MALWARE ANALYSIS / REVERSE ENGINEERING

PackClient

Reverse Engineering a Modular RAT Framework

ABSTRACT

This report reconstructs the PackClientLauncher build carried by the Tax Notice sideload chain: its worker-side 1RCP screenshot interface, transport and authentication, PLK1 delivery/cache path, and active-session handoff. Two runtime evidence sets connect the recovered launcher to private executable mappings, persistence and attempted connectivity. Core bytes, the external 1RCP peer, successful C2 and a completed real-worker exchange remain unrecovered or unproven.

Analysis scope — PackClientLauncher code and runtime evidence

Ivan Immanuel Shaji

Published: 6 September 2026

Research cutoff: 5 September 2026 (UTC)

CONTENTS

The research

PackClientLauncher: reconstructed interfaces, runtime evidence and the limits of the recovered build.

Online article and full-resolution figures · Technical references and tooling

01 / TECHNICAL SUMMARY

A launcher with a recoverable contract

The worker interface is precise. Its external peer is missing. Two runtime evidence sets show where the launcher was active and how far it got.

PackClient was already a publicly documented malware family when this research began. This report examines one recovered launcher build: its private screenshot interface, the authentication and delivery path alongside it, and its correspondence with two separately collected runtime evidence sets.

The recovered worker implements `1RCP`: a 20-byte header, four message types, a raw top-down BGRX framebuffer, and an endpoint opened by the worker rather than created by it. The worker writes pixels to that endpoint. The endpoint creator and downstream consumer were not identified, and the recovered launcher does not establish how they use those pixels. Screenshot IPC reference[1]

Finding	Evidence basis	What it establishes
Launcher interfaces	Static reconstruction and field-level dataflow	Framing, authentication, cache acceptance and worker-side screenshot IPC contracts in recovered B
Runtime correspondence	Two process dumps, native process records and focused screenshots	The reconstructed launcher resides in private mappings in two surrogate processes
Synthetic tooling validation	Offline decoder, metadata dissector and IPC-kit tests	Agreement with the reconstructed contracts on synthetic inputs; real-capture compatibility remains unvalidated

The strongest runtime result is **launcher residence and attempted connectivity**. PID 5812 preserves a concrete pull attempt ending in `WSA=10060`. The second evidence set preserves a launcher mapping in PID 3696 and failed-SYN evidence. Neither demonstrates a completed C2 session, downloaded Core, or successful real screenshot exchange.

The worker failed before READY; the debugger record does not establish the causal failure sequence. Evidence and method[2], limitations[3]

02 / RESEARCH

From a signed host to the embedded launcher

The staged helper name explains the load relationship. Pinned bytes and package structure establish what it contains.

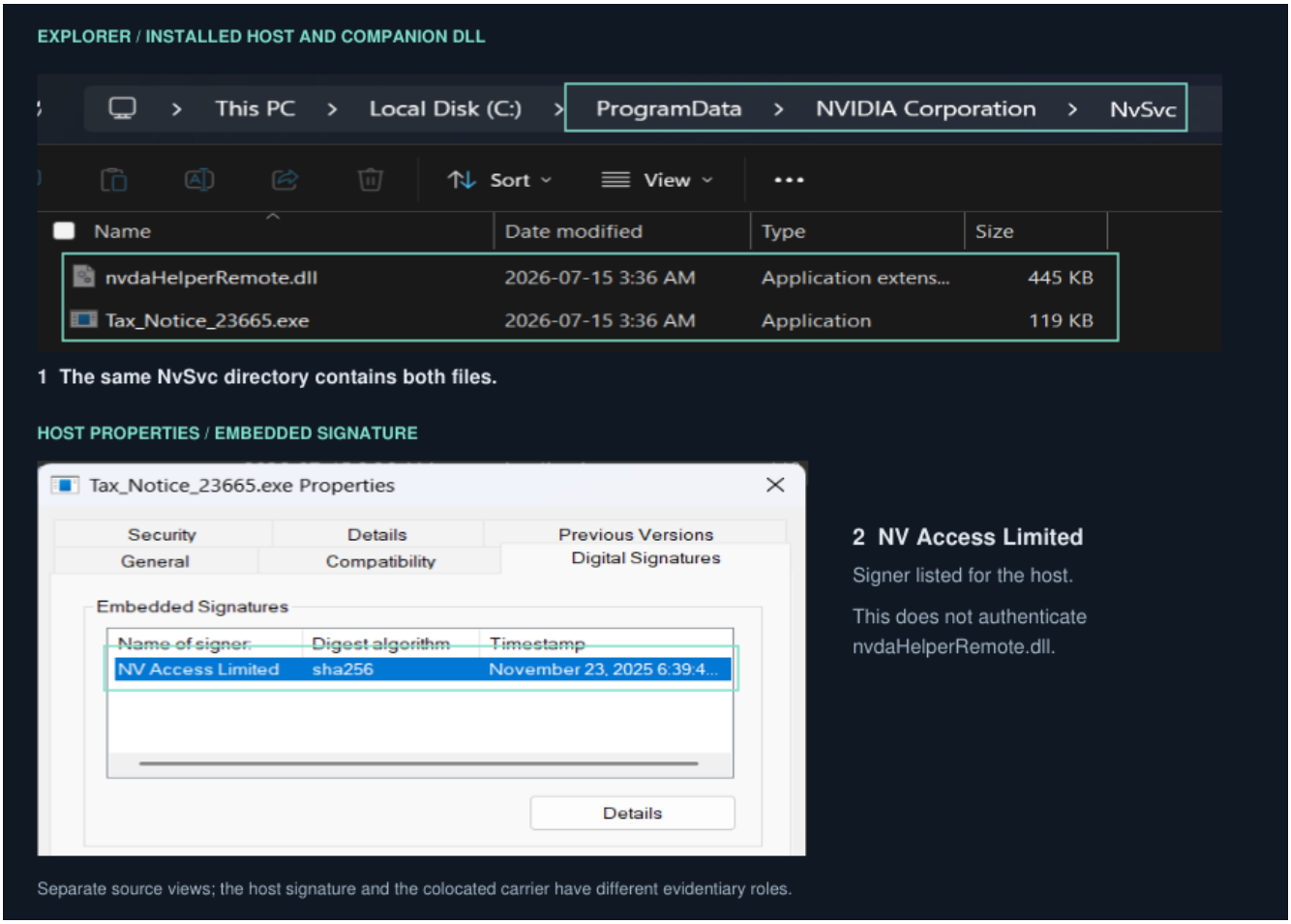
The July 2026 Tax Notice lineage stages a signed NV Access/NVDA executable beside a malicious `nvdaHelperRemote.dll`. The host imports that helper by bare filename, allowing the colocated carrier to satisfy the import. The legitimate helper name is documented by NV Access. This is the recovered DLL sideload relationship; the signed host's identity does not make the staged pair benign. Artifact identity and lineage[2], NVDA helper documentation[4]

Inside the carrier, the declared transformed record resolves to a coherent x86 call-over-data package containing two PE32 executables, A and B, and a terminal loader candidate. Image B identifies as `PackClientLauncher.exe`. Automated reports used Donut-related labels, but the byte-level reconstruction did not establish that specific generator. The report therefore attributes the recovered structure, without promoting the automated label. Transformed record[5], Image B[5]

A is the wrapper/mapper: it maps its embedded B inside the current process and calls B's entry point. The carrier separately contains task creation, a `runas` self-elevation request and suspended `svchost.exe` creation followed by thread-context operations. These recovered call contracts support the runtime process observations; the exact upstream code-placement primitive remains unresolved. Carrier launch paths and A-to-B mapping[5]

FIGURE 01 · RUNTIME OBSERVATION

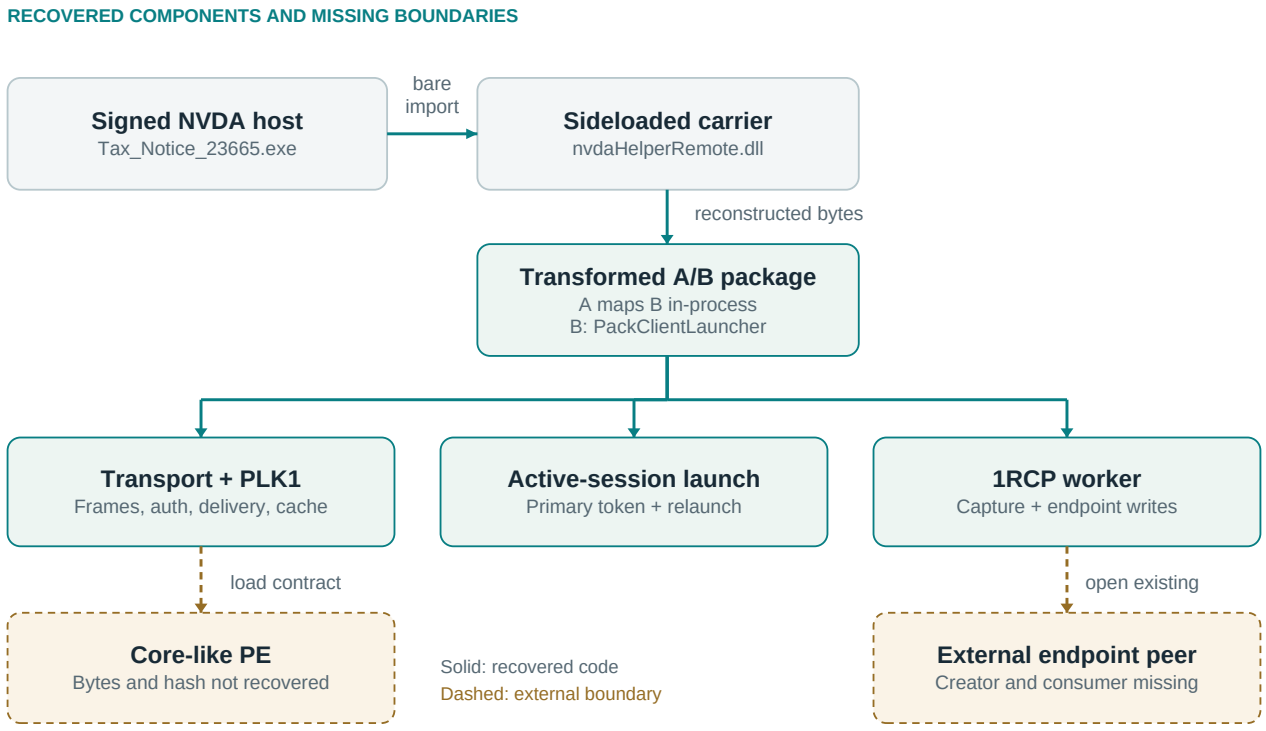
The installed pair uses a signed host and a separate companion DLL.



Explorer shows Tax_Notice_23665.exe beside nvdaHelperRemote.dll under C:\ProgramData\NVIDIA Corporation\NvSvc. A separate properties view lists an embedded NV Access Limited signature for the host. The signature belongs to the host; this view does not authenticate the companion DLL.

Source and analysis · Full-resolution figure

FIGURE 02 · TECHNICAL DIAGRAM



The launcher explains four planes, with two external boundaries. Static reconstruction; not a captured exchange.

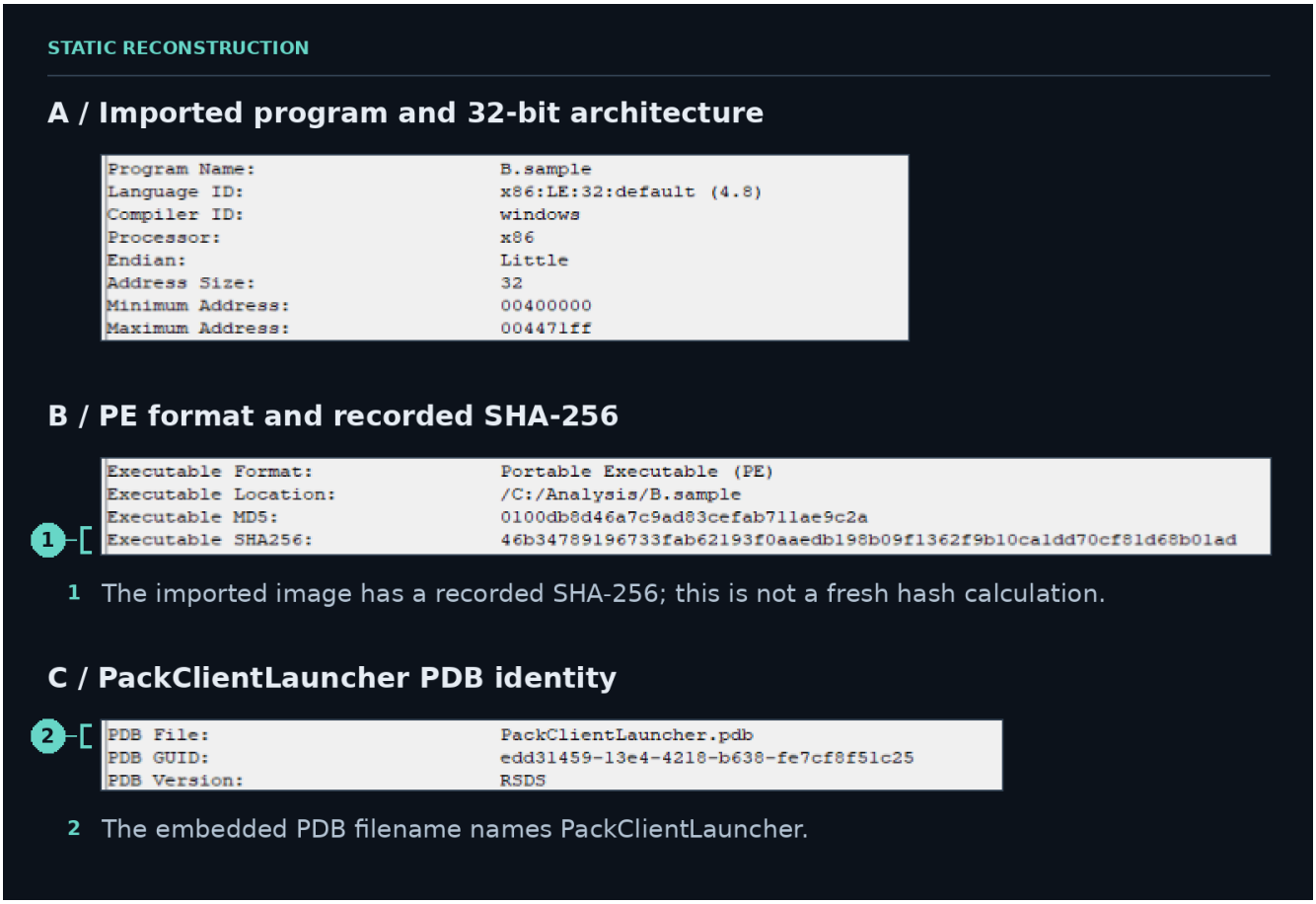
The launcher implements transport, PLK1 transfer/cache, active-session handoff and an early screenshot-worker mode. These are related capabilities in B; their coexistence does not prove that a screenshot is sent directly over the lower network transport.

Object	Recorded size	Identity anchor
Tax Notice host	120,984 bytes	SHA-256 begins `93DD8B7B393289F8`
Sideload carrier	455,527 bytes	SHA-256 begins `7295090C2CB63EBC`
Reconstructed image B	PE32 launcher	SHA-256 begins `46B34789196733FA`

The identity ledger[6] publishes complete hashes. Short hashes above are reading aids, not substitute identifiers.

FIGURE 03 · STATIC RECONSTRUCTION

The recovered B image has a concrete identity.



The image summary identifies the imported PE and its PackClientLauncher PDB name. The recorded SHA-256 identifies recovered image B.

Source and analysis · Full-resolution figure

03 / RESEARCH

The 1RCP screenshot boundary

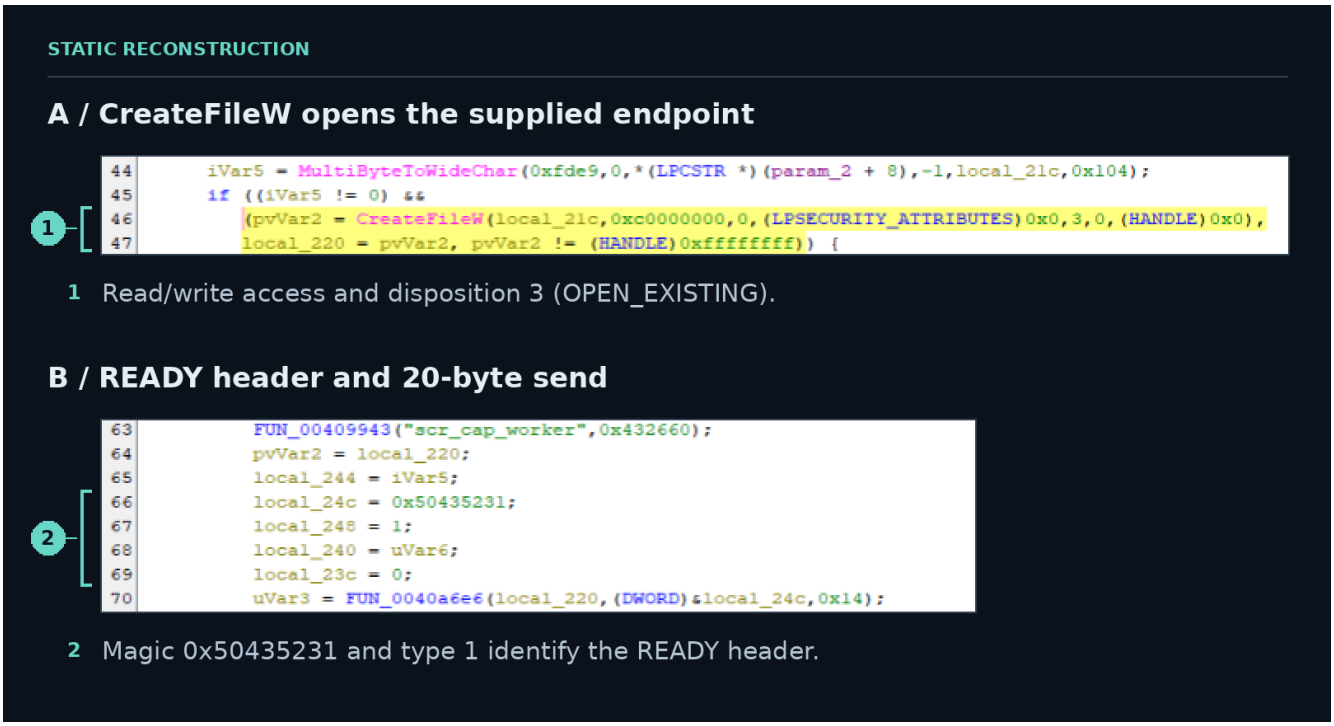
The worker implements a small, exact protocol. The endpoint creator and downstream consumer remain unidentified.

The worker is selected by an early argument-dispatch branch, before the later active-session/Core path. The interface is `/scr_cap_worker <endpoint-utf8> [monitor-index]`. Its body runs in the already launched process and inherits that process's token, session and environment. The wrapper exits after the worker returns; the later token-based launch path cannot be used to infer the worker's original security context. Activation and context[1]

The endpoint argument is converted to UTF-16 and passed to `CreateFileW` with read/write access, no sharing and `OPEN_EXISTING`. There is one open attempt. The recovered path contains no endpoint creator, named-pipe server, wait-for-pipe loop or peer authentication. A named pipe is strongly supported by context, but the argument's namespace is not recovered; the API itself also accepts other paths. Screenshot IPC reference[1]

FIGURE 04 · STATIC RECONSTRUCTION

The worker opens an existing endpoint.

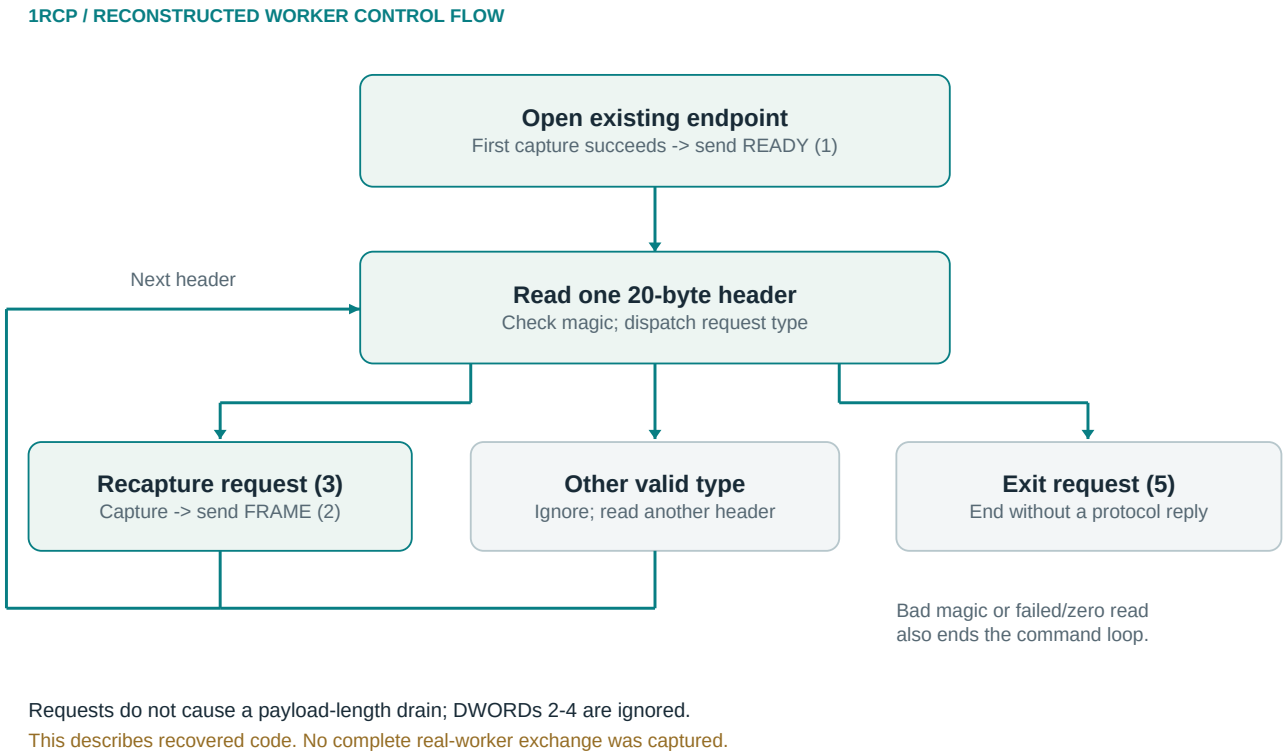


1. CreateFileW receives the supplied endpoint, read/write access and disposition 3 (OPEN_EXISTING). 2. The READY header uses magic 0x50435231 and type 1. The endpoint namespace and external peer are not recovered.

Source and analysis · Full-resolution figure

After the first successful capture, the worker emits READY. It then accumulates exactly 20 command bytes, checks the magic and dispatches the request type. Correct-magic types other than 3 and 5 are ignored. Bad magic, a failed read or a zero-byte read ends the command loop without a protocol reply. The diagram scopes the successful capture paths and command dispatch; it is not a packet capture.

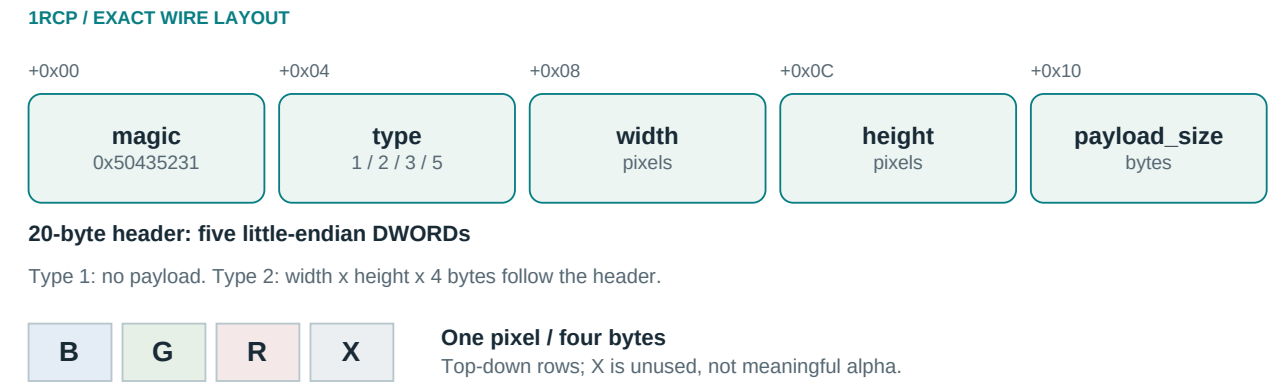
FIGURE 05 · TECHNICAL DIAGRAM



The recovered 1RCP worker opens, captures, announces, then reads commands. Static reconstruction; not a captured exchange.

The header is five little-endian DWORDs. The bytes `31 52 43 50` spell `1RCP`; as a DWORD the magic is `0x50435231`.

FIGURE 06 · TECHNICAL DIAGRAM



The 20-byte header and the raw BGRX payload have separate meanings. Static reconstruction; not a captured exchange.

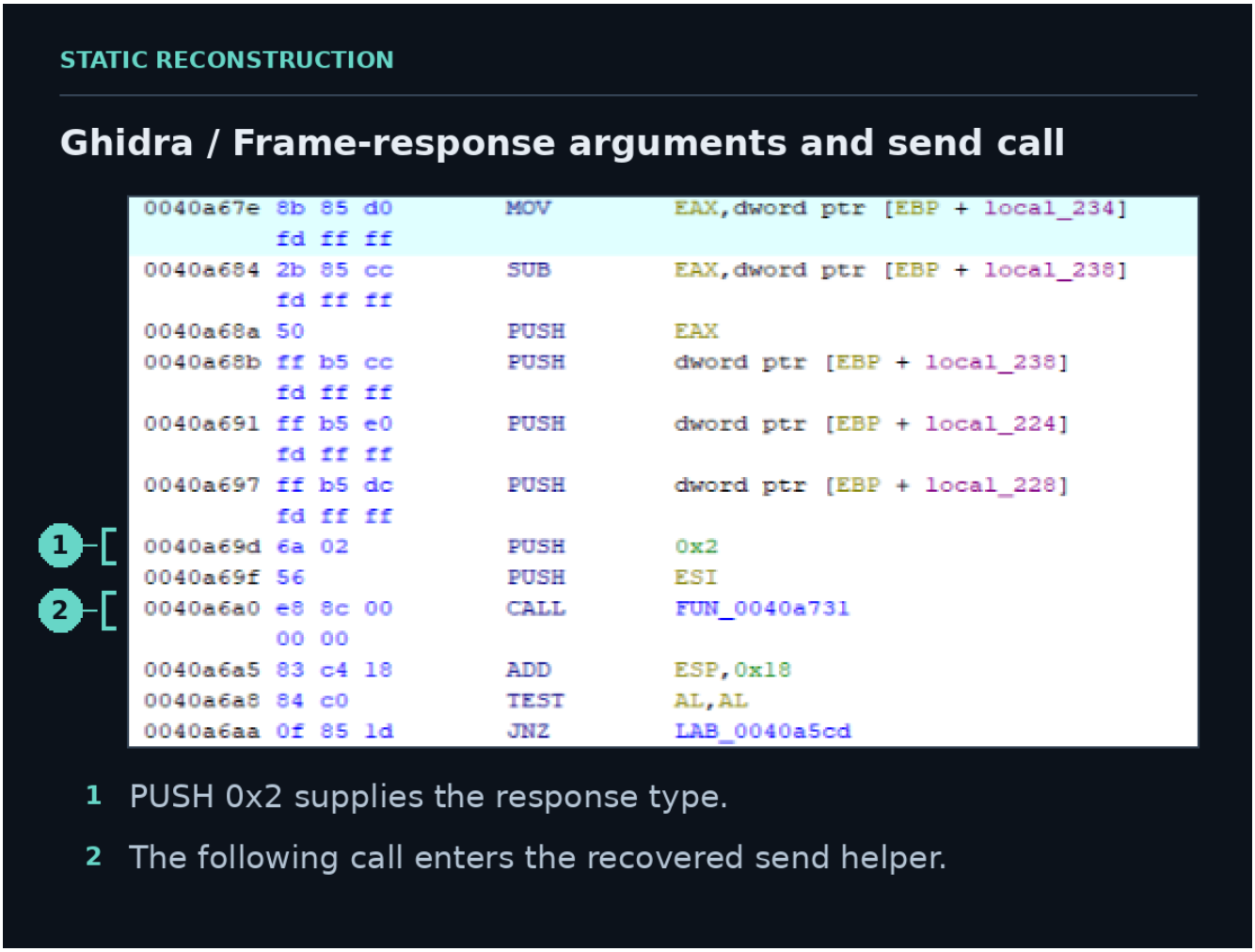
Type	Direction	Meaning	Dimensions and payload
1	Worker → peer	READY	Captured width and height; payload length zero
3	Peer → worker	Recapture	Request DWORDs 2–4 are ignored
2	Worker → peer	Frame response	Width, height and following pixel-byte length
5	Peer → worker	Exit	Request DWORDs 2–4 are ignored; no reply

This distinction matters for an independent parser. Zero-filled request fields are a synthetic fixture convention, not a requirement imposed by the worker. The request loop also does not drain extra bytes according to the request's last DWORD; it reads the next 20-byte header. Worker process exit codes are separate from these message types.

The selected monitor supplies positive width and height. The DIB uses 32-bit `BI_RGB` with a negative height, producing top-down rows. Each pixel is blue, green, red and an unused fourth byte; meaningful alpha is not established. Stride is `width × 4`, with no additional per-row padding. The expected payload is `width × height × 4`; the IA-32 reconstruction found no separate arithmetic overflow guard. The actual response length is taken from the framebuffer vector's end-minus-begin distance. Pixel calculation and layout[1]

FIGURE 07 · STATIC RECONSTRUCTION

The recapture branch constructs a type-2 response.



The literal type 2 and the send-helper call are visible in the recovered recapture path. The field semantics and BGRX payload layout come from the linked static dataflow analysis. This is reconstructed code, not a captured live framebuffer.

Source and analysis · Full-resolution figure

The verified dataflow ends at `WriteFile` on the supplied endpoint. Within recovered B, no scoped call/data edge carries those pixels into Winsock, PLK1, the encrypted transport, or a JPEG/PV10 conversion path. An external peer could transform or forward the pixels, but its implementation is missing. The recovered worker interface does not establish end-to-end exfiltration. External peer boundaries[1]

04 / RESEARCH

Framing, authentication and the missing key writer

The lower transport can be reconstructed more precisely than the available runtime traffic can validate it.

The outer frame starts with a little-endian word formed by `0x5A400000 | body_length`. The receiver checks `(frame_word >> 22) == 0x169`, bounds the nonzero body at `0x3FFFFF`, reads it exactly, and requires at least four bytes for the type. Thus `24 00 40 5A` means a 36-byte body and a 40-byte total frame. Treating the prefix as an entire message would produce an incorrect decoder. Framing and handshake[7]

Offset	Bytes	Outer-frame meaning
`0x00`	4	Prefix combined with body length
`0x04`	4	Little-endian type DWORD, part of the body
`0x08`	Variable	Payload; body length minus four

B sends PLH1 and PLA1 in plaintext type `0x15`. Its first framed receive accepts PLC1 either directly in type `0x15` or through an authenticated outer `0x16` envelope with inner type `0x15`, provided the envelope keys are initialized. This receive precedes Core loading. PLH1 is a 32-byte client hello; PLC1 is the 24-byte server challenge; PLA1 is the 40-byte client authentication object. PLH1 includes version 1, an unresolved word set to `0x20`, fixed fields, a tick count and the process ID. PLC1 contributes a 16-byte challenge at offsets `0x08..0x17`; challenge generation lives in the missing server.

PLA1 uses HMAC-SHA-256 over exactly **42 bytes**. It is not a hash of the entire hello and challenge:

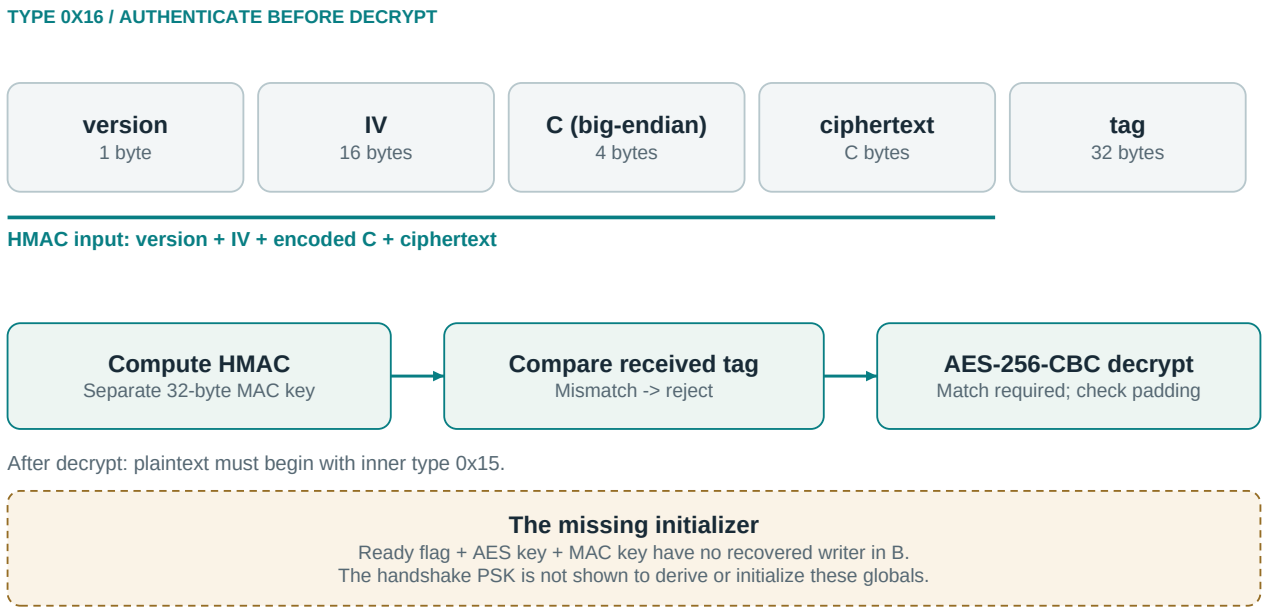
```
PLH1[0x10:0x20]  // 16 bytes
|| PLC1[0x08:0x18]  // 16 bytes
|| PLH1[0x0C:0x10]  //  4 bytes
|| PLH1[0x06:0x08]  //  2 bytes
|| "PLK1"          //  4 bytes
```

The PSK comes from `PACK_LAUNCH_PSK`; missing or empty values select the 19-byte fallback `pack-launch-dev-psk`. The passive tool only uses that fallback when explicitly requested. After sending PLA1, the recovered function does not check a separate post-authentication acknowledgement. These are build-specific static facts, not observations of a successful live handshake. Handshake evidence[2]

Authentication gates encrypted receive

Type `0x16` is an inbound encrypted envelope. In this B image the send path constructs only type `0x15`; symmetric support must not be assumed. The envelope carries a version byte, 16-byte IV, four-byte **big-endian** ciphertext length, ciphertext, and 32-byte HMAC tag. The length field's byte order differs from the surrounding little-endian framing.

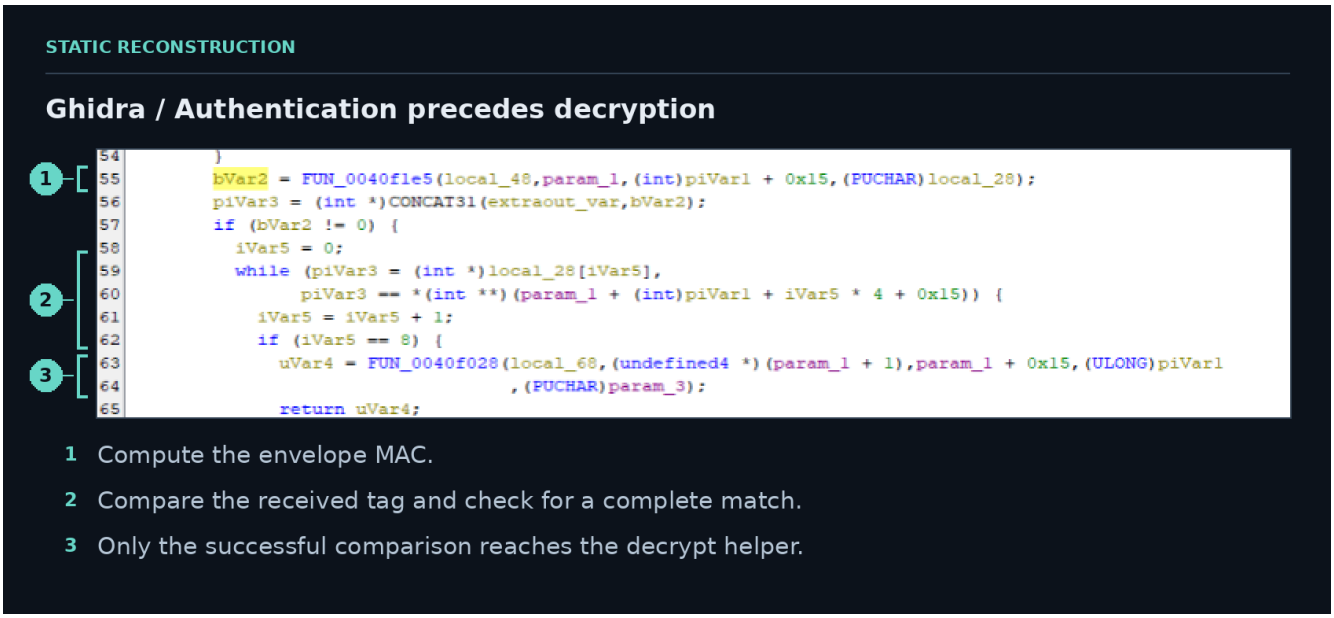
FIGURE 08 · TECHNICAL DIAGRAM



A valid HMAC is a precondition for AES-CBC decryption. Static reconstruction; not a captured exchange.

FIGURE 09 · STATIC RECONSTRUCTION

Authentication gates decryption.



1. Compute the envelope MAC. 2. Compare the received tag. 3. Enter the decrypt helper only after the comparison succeeds. Helper roles are established in the transport analysis; the screenshot shows their control-flow order.

Source and analysis · Full-resolution figure

The MAC covers version, IV, encoded length and ciphertext, excluding the outer frame/type and the trailing tag. A tag match gates AES-256-CBC decryption with BCrypt block padding; the resulting plaintext must begin with inner type `0x15`. The recovered tag comparison exits early, so the report does not describe it as constant-time.

A ready byte at B RVA `0x41C44` gates a 32-byte AES key at `0x41C48` and a separate 32-byte HMAC key at `0x41C68`. The completed B census found reads but no initializer, setter or derivation. The handshake PSK/HMAC was not shown to flow into those globals. An unrecovered in-process component could initialize them; absent that state, encrypted receive remains disabled. Envelope-key initialization remains unresolved. Envelope key origin[7]

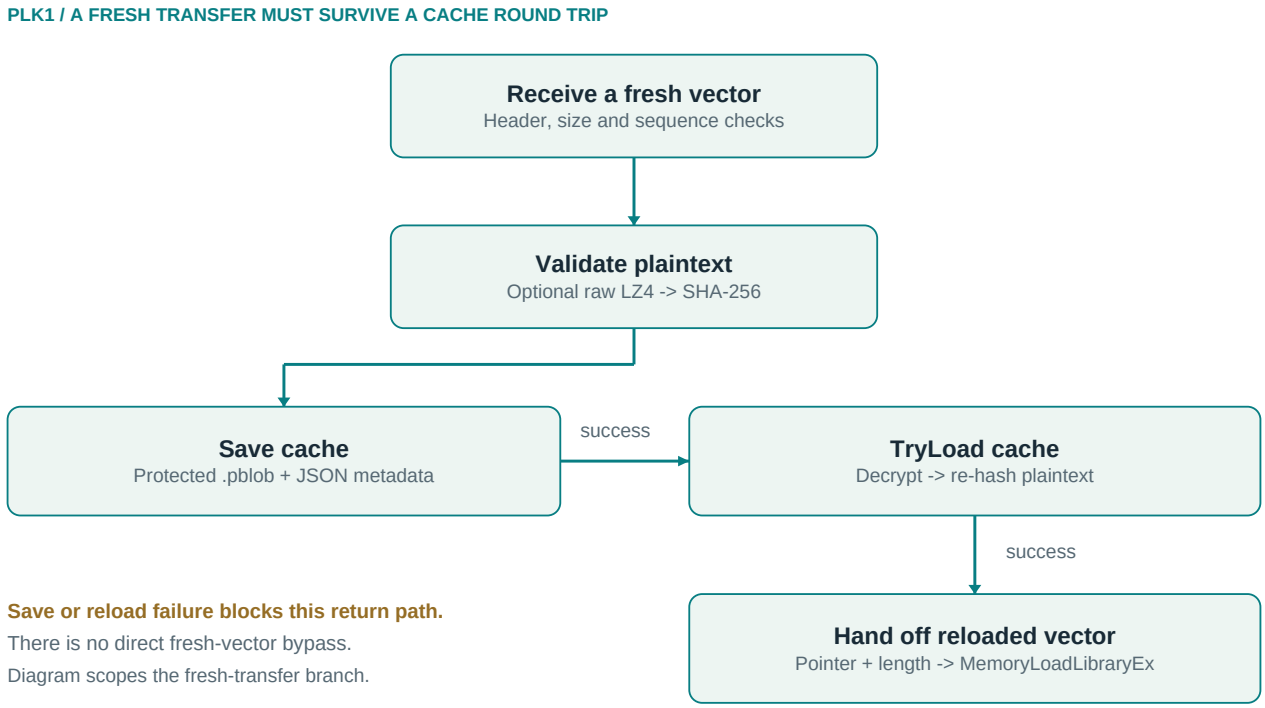
05 / RESEARCH

Delivery is gated by a cache round trip

Freshly received bytes do not pass directly to the PE loader. The cache is part of the acceptance path.

PLK1 validates magic/version, original and transferred size bounds, sequence and chunk-copy state, optional raw LZ4 decompression, and the final plaintext SHA-256. Active callers request slot 0, corresponding to `PackClientCore.primary.dll`. Generic shared support can name a secondary slot, but the bounded caller census found no reachable request for it in this build. Delivery and Core boundary[2]

FIGURE 10 · TECHNICAL DIAGRAM



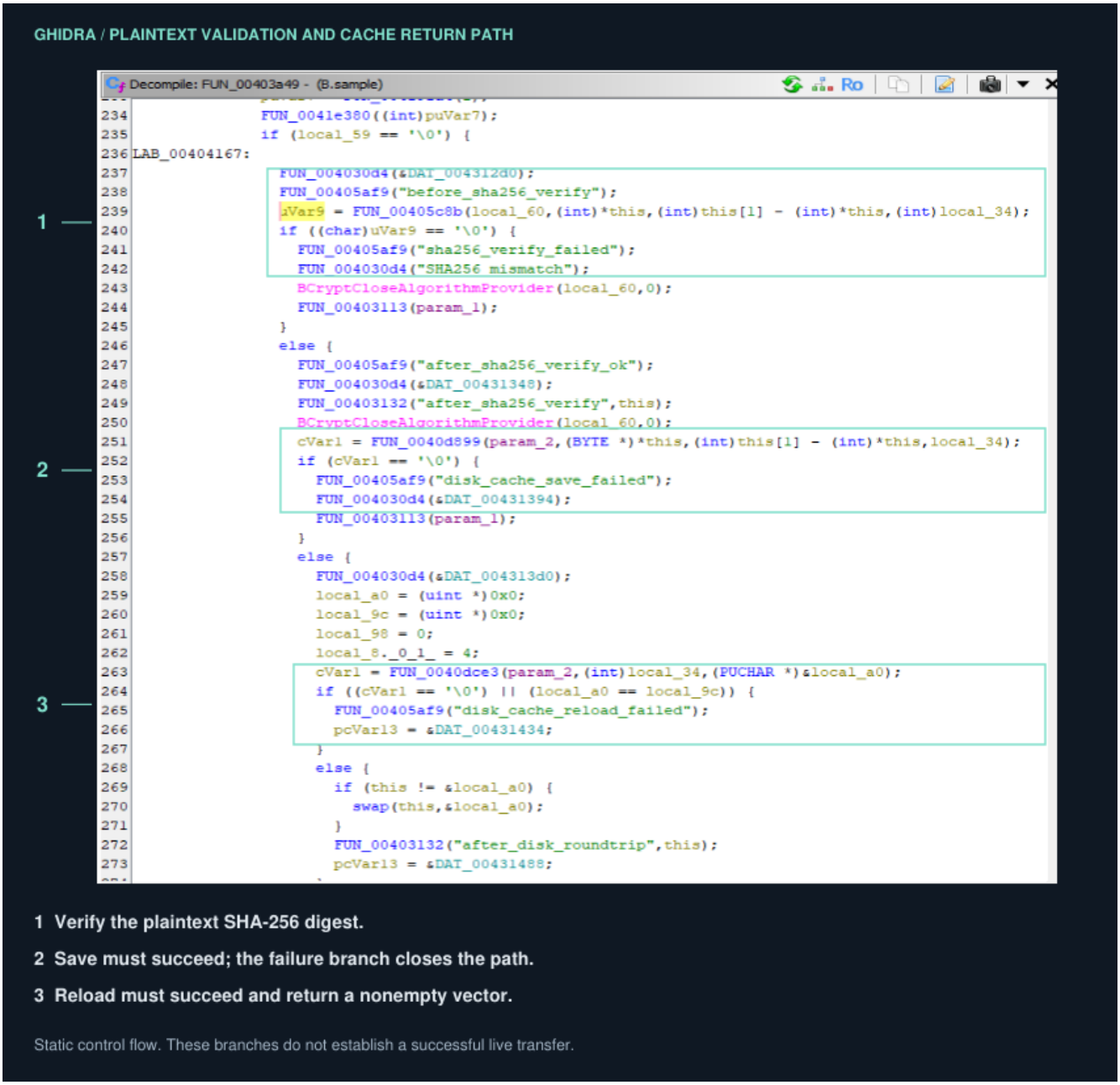
Freshly received bytes reach the loader only after Save and TryLoad. Static reconstruction; not a captured exchange.

For a fresh transfer, `Save` must succeed. `TryLoad` must then decrypt and independently re-hash the stored plaintext. Only that reloaded vector is returned to the PE wrapper and `MemoryLoadLibraryEx`. This exact branch order matters: a decoder that accepts the network object alone models only part of the launcher's acceptance conditions.

The capability resolution and CryptProtect/Unprotect dataflow strongly support current-user DPAPI protection for a `.pblob` plus JSON metadata under a relative `pluginsdata\x86` tree. The exact runtime base path is unresolved. Cache names, expected role/exports and the immediate PE-loading contract strongly suggest Core's role; **no Core bytes or independent Core hash were recovered**. A string naming Core inside B is evidence of a contract, not another resident PE. Vector and cache provenance[7], Core acquisition gap[5]

FIGURE 11 · STATIC RECONSTRUCTION

The plaintext must pass digest verification, cache save and cache reload.



The recovered transfer body checks the plaintext digest, requires Save to succeed, and immediately requires TryLoad to return a nonempty vector. The visible failure branches are SHA256 mismatch, disk_cache_save_failed and disk_cache_reload_failed. This is statically reconstructed control flow, not evidence of successful delivery.

Source and analysis · Full-resolution figure

06 / RESEARCH

Staying with the active console session

The launcher contains an active-console relaunch mechanism with a 30-attempt spawn loop. It should not be conflated with the original chain’s elevation step.

The active-session path resolves a bootstrap executable, preserves a constrained set of special arguments, duplicates the current process token as a primary launch token and assigns it to the target console session before calling `CreateProcessAsUserW`. A separately selected user token can supply the environment; it is not the launch token. This is separate from the early screenshot mode, which exits through its own wrapper. Token selection[8], Spawn contract[8]

Parameter or state	Recovered behavior
Application and command	Null application-name argument; mutable quoted command line
Child environment	Optional Unicode environment; handle inheritance disabled
Desktop and visibility	`WinSta0\Default`; hidden/no-window startup
Creation flags	`0x09000400`: breakaway, no window, Unicode environment
Retry bound	At most 30 attempts; one-second sleep after failure
Session-drift polling	200 ms; a valid different console ID triggers replacement

On successful drift replacement, the launcher closes child handles and exits the old process. It uses canonical `-acsi` state for this handoff. The recovered API makes the continuity mechanism concrete; it does not establish which elevation API produced the original high-integrity host in the second runtime evidence set. Session drift[8]

FIGURE 12 · STATIC RECONSTRUCTION

The active-session path retries a token-based launch.

The recovered call uses CreateProcessAsUserW with flags 0x09000400. The loop sleeps for 1,000 ms after failure and stops after at most 0x1E attempts. This is the active-session helper, not proof of the original chain's elevation API.

Source and analysis · Full-resolution figure

07 / RESEARCH

The reconstructed code appears in memory

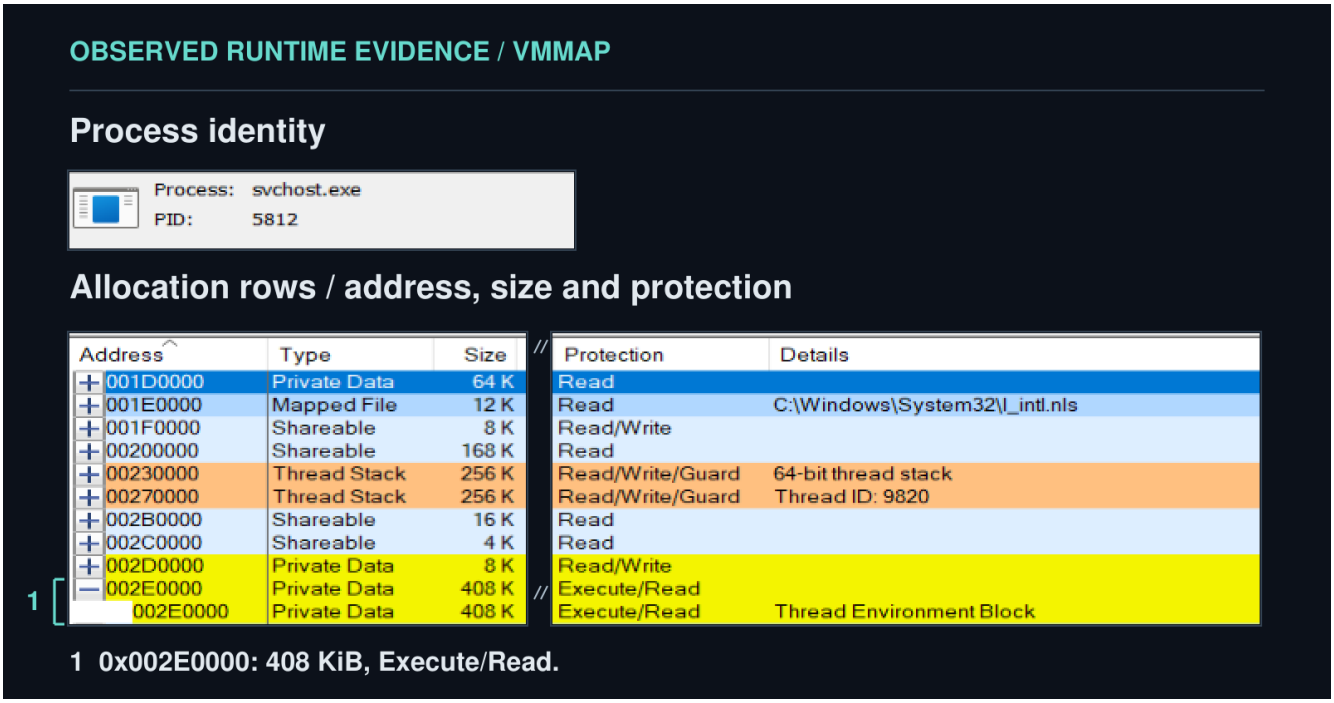
The first runtime set links static structure, private mappings, a thread start and launcher-specific live state.

The first runtime capture preserves a bare 32-bit `SysWOW64\svchost.exe`, PID 5812, an NvSvc task, VMMap state, a user-mode dump and a late Process Monitor capture. The runtime correspondence below comes from passive examination of those artifacts.

VMMap's 408 KiB private executable allocation at `0x002E0000` was an investigative lead. Its displayed "Thread Environment Block" label did not identify its contents. The dump's package structure, embedded offsets, PE metadata and section hashes supplied that identification. Memory correspondence[2]

FIGURE 13 · RUNTIME OBSERVATION

Private executable allocation in svchost.exe, PID 5812.



VMMap shows a 408 KiB Execute/Read allocation at 0x002E0000. Corresponding dump analysis identifies its contents as the transformed A/B package.

Source and analysis · Full-resolution figure

Object in PID 5812	Runtime location	Why the match matters
Transformed package allocation	`0x002E0000`; 417,792 bytes	A/B occur at the static package offsets `0x128D` / `0x1CC1`
Mapped A	`0x02C70000`; image size `0x64000`	Entry RVA `0x1308`, timestamp `0x6A27AE4B` and PE layout agree
Mapped B	`0x05410000`; image size `0x48000`	Entry RVA `0x12FAC`, timestamp `0x6A3CB0B5` and immutable sections agree
Thread 1884	Start `0x02C71308`	Exactly mapped A plus its entry RVA

The mapped images are outside the registered module list. B-specific state includes the named mutant `PackClientLauncher.Session.9b2126fc5ed31443` under the BaseNamedObjects namespace, effective configuration and formatted launcher diagnostics. This converging evidence supports B's activity more strongly than a lone Core-related string or a suspicious process name would.

The bounded PE/marker census found no independently identifiable Core image in the dump. That does not exclude headerless content, missing pages, another moment in the process's life, or unavailable artifacts. It also does not reveal the instruction-level mechanism that populated the surrogate. Runtime validation[9]

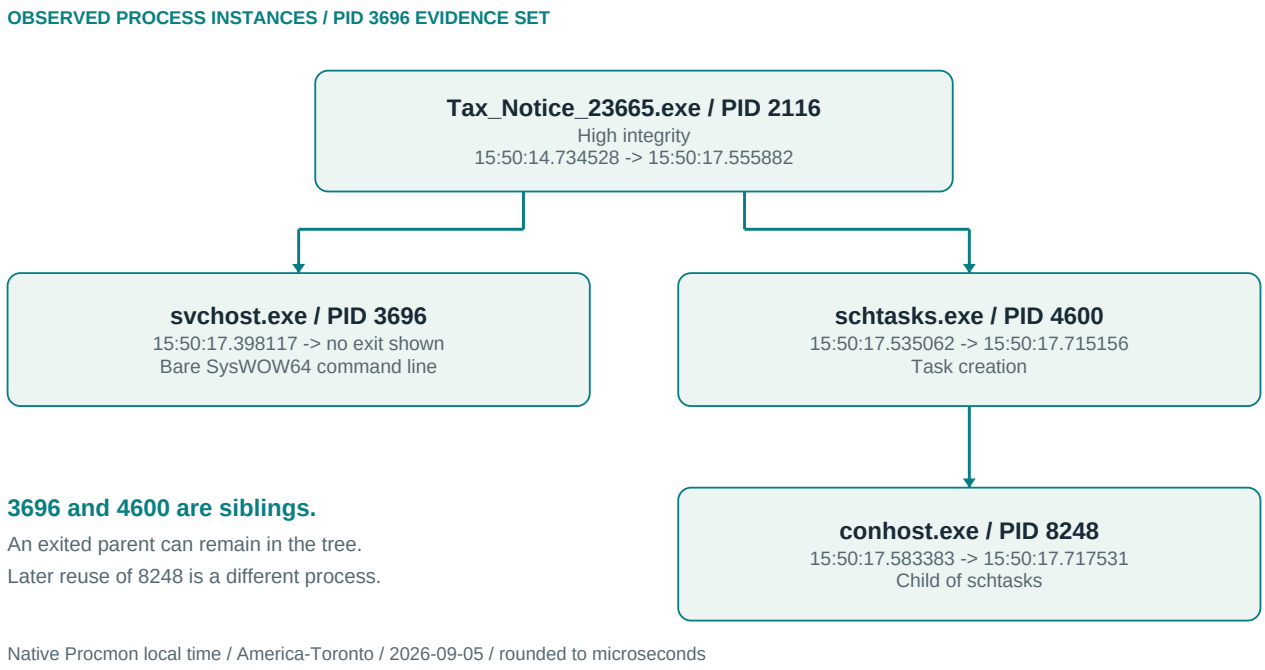
08 / RESEARCH

A second runtime set preserves the creation chain

The September 5 capture connects the adjacent helper load, elevation, persistence and long-lived PID 3696 surrogate.

During controlled execution in Windows 11, Process Monitor records successful loading of the adjacent helper after two earlier `0xC0000906` failures. A medium-integrity Tax Notice process, PID 5952, relaunches as high-integrity PID 2116. The elevated host creates the bare surrogate and the task-creation branch below. The session includes separate launch attempts; the process relationships are tied to their recorded creation events. Native process evidence[9]

FIGURE 14 · TECHNICAL DIAGRAM



The recorded host creates a long-lived surrogate and a short-lived task branch. Native-event correlation; process times rounded to microseconds.

The relationship is specific: **svchost 3696 and schtasks 4600 are siblings under 2116**. Conhost 8248 belongs below schtasks. The parent exits shortly after creating its children; its continued appearance in a process tree preserves ancestry, not liveness. Later reuse of a numeric PID must be joined to a new process start and lifetime.

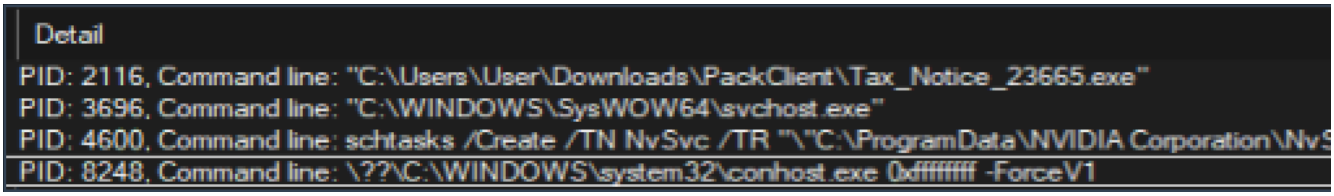
FIGURE 15 · RUNTIME OBSERVATION

One parent creates two siblings.

A / Creating process, PID, operation and result

Process Name	PID	Operation	Path	Result
Tax_Notice_23665.exe	5952	Process Create	C:\Users\User\Downloads\PackClient\Tax_Notice_23665.exe	SUCCESS
Tax_Notice_23665.exe	2116	Process Create	C:\WINDOWS\SysWOW64\svchost.exe	SUCCESS
Tax_Notice_23665.exe	2116	Process Create	C:\WINDOWS\SYSTEM32\schtasks.exe	SUCCESS
schtasks.exe	4600	Process Create	C:\WINDOWS\System32\Conhost.exe	SUCCESS

B / Corresponding child PIDs and command lines



PID 4600 command, continued (path segment repeated):

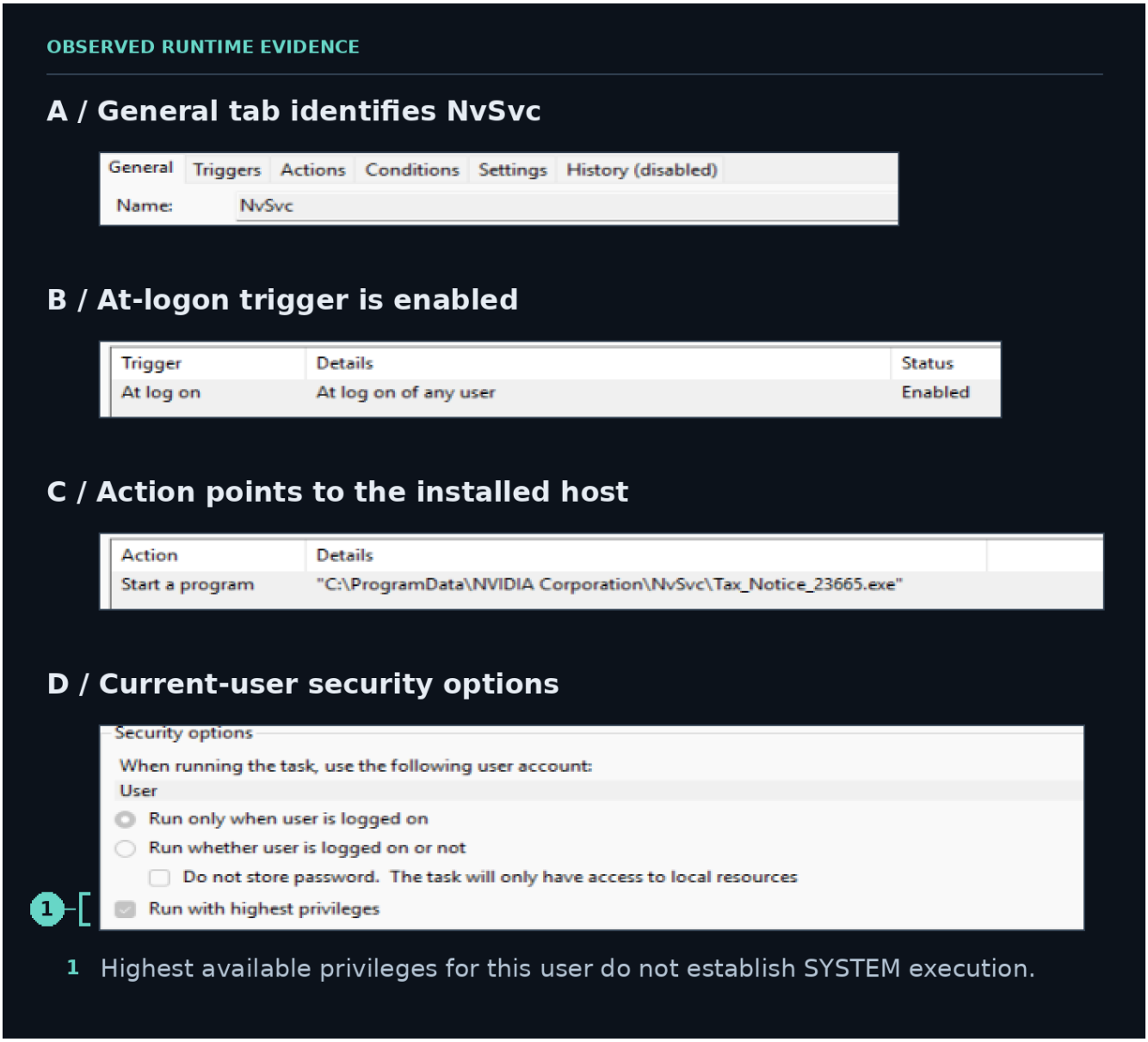
Process Monitor records parent PID 2116 creating svchost PID 3696 and schtasks PID 4600; schtasks then creates conhost PID 8248. SUCCESS is the Process Create result, not the child's exit code.

Source and analysis · Full-resolution figure

Native file and registry records establish installation beneath `C:\ProgramData\NVIDIA Corporation\NvSvc`, hidden/backdated copies and HKCU Run/RunOnce writes in this chain. Task XML and the creation record establish NvSvc persistence. Task Scheduler shows the at-logon trigger, ProgramData action and configured user context. Unlike the first set’s timestamp mismatch alone, the second runtime evidence set includes records of the backdating operation. Persistence evidence[2]

FIGURE 16 · RUNTIME OBSERVATION

NvSvc combines an at-logon trigger with a ProgramData action.



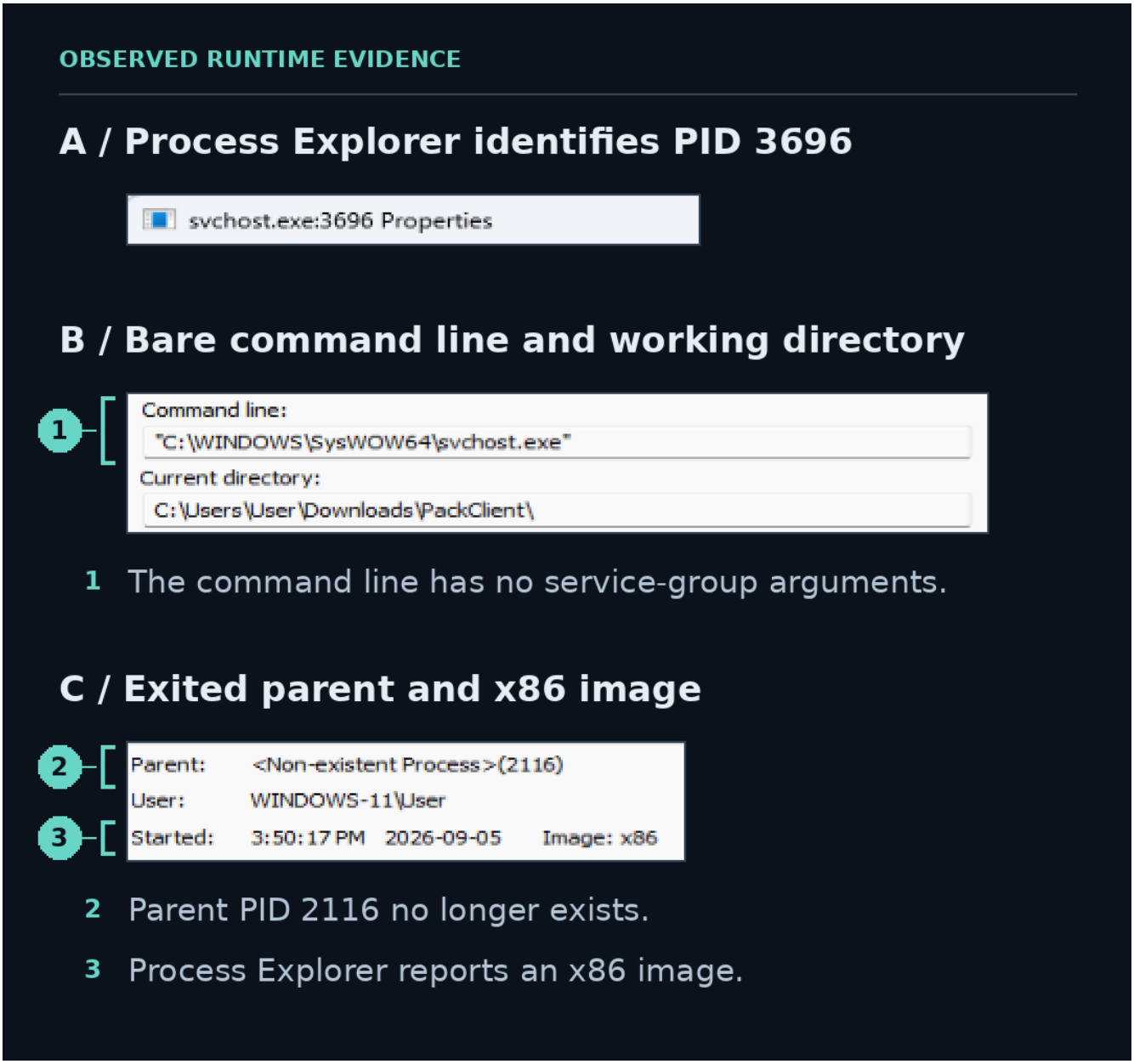
Task Scheduler shows NvSvc's enabled at-logon trigger, ProgramData action and configured principal. Highest available privileges for this user do not establish SYSTEM execution.

Source and analysis · Full-resolution figure

“Highest privileges” here belongs to the selected current user; it does not mean SYSTEM. A later persisted-path launch is observed, but the saved evidence does not resolve every trigger as logon versus on-demand. The bare surrogate properties support an anomalous launcher host; they do not identify process hollowing or another specific injection subtype.

FIGURE 17 · RUNTIME OBSERVATION

The surviving surrogate has a bare command line.



1. Bare SysWOW64 svchost command line. 2. Parent 2116 has exited. 3. The process is x86. The Downloads\PackClient current directory supplies additional context. These properties support an anomalous surrogate, without identifying an injection subtype.

Source and analysis · Full-resolution figure

The second dump records a 415,071-byte transformed package and a private B mapping. The recorded comparison reports eight differing bytes at three package sites, plus normalized B correspondence after accounting for relocations and the import address table. **The package/B residence result stands separately from the proposed explanation for those eight bytes.** Three simple 6666-to-443 port substitutions would ordinarily account for six changed bytes, so the exact per-site before/after transcript is still required. The eight-byte result is not a proved port patch. Recorded comparison and open gap[2]

09 / RESEARCH

An attempt is not an established session

Different artifacts answer different network questions. Their clocks, coverage and attribution limits must stay attached to the result.

PID 5812’s preserved environment and concrete formatted buffers identify `154[.]36[.]188[.]201:443`, a slot-0 pull attempt and `WSA=10060`, a connection timeout. The PSK environment variable was absent in the captured environment. This is positive evidence of an attempt; it is not a captured PLH1/PLC1/PLA1 exchange or evidence that the encrypted receive keys were initialized. First-set network evidence[2]

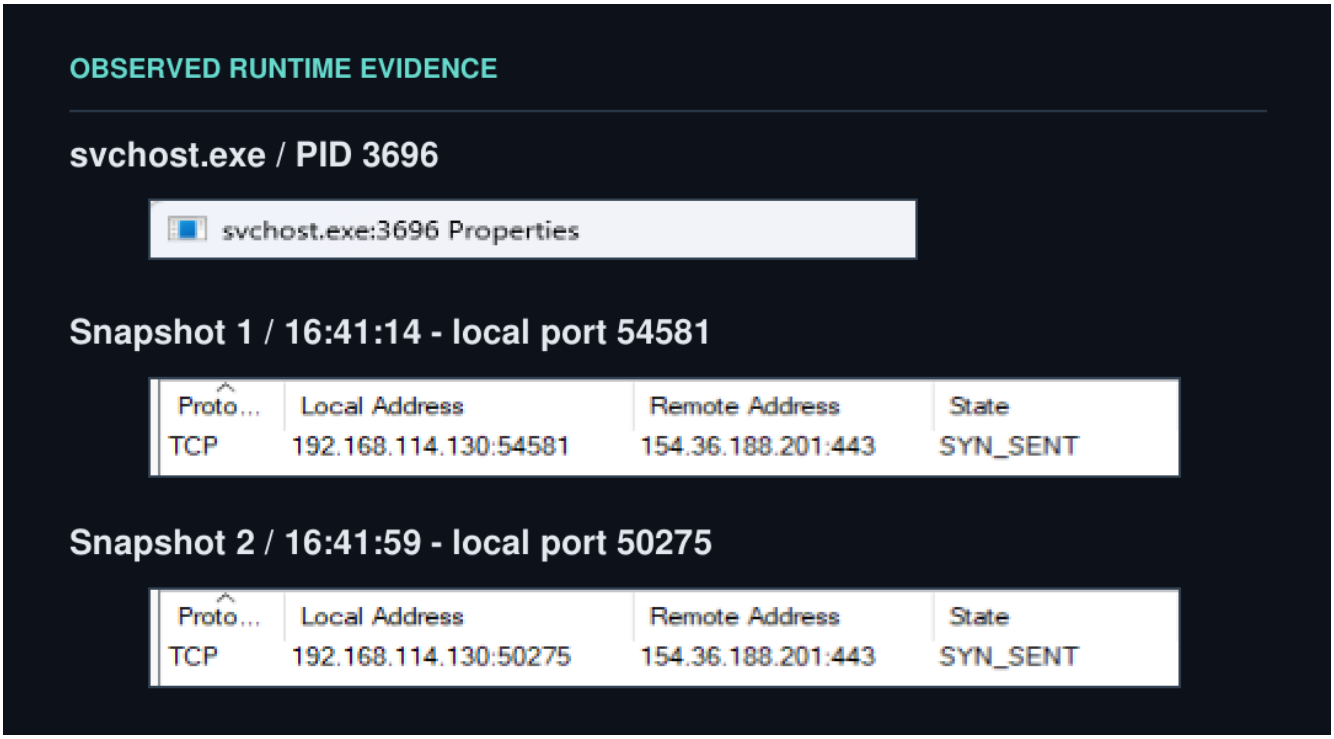
The second runtime evidence set adds 373 ICMP host-unreachable messages quoting TCP SYNs to the same destination. The packet census found no SYN-ACK, established flow or application payload. Process Explorer separately associates PID 3696 with SYN_SENT sockets at two captured moments. Those views supply direct UI-level process/socket association; the PCAP itself has no PID metadata.

Evidence object	What it establishes	Limit
PID 5812 dump buffers	Configured attempt and timeout	No successful handshake or delivery object
PID 3696 TCP/IP views	The process owns the displayed SYN_SENT entries	Only the displayed moments; no TCP completion
PID 3696 evidence-set packet capture	Failed SYN evidence quoted by ICMP	Process attribution is correlation, not packet metadata

Port 443 alone does not establish TLS or HTTPS. The strong association between the PID 3696 launcher and the failed-SYN set comes from endpoint, timing, process lifetime and configuration together. It remains a correlation when applied to individual packets. Network attribution[2]

FIGURE 18 · RUNTIME OBSERVATION

Connection attempts from svchost.exe, PID 3696.



Process Explorer shows two SYN_SENT attempts to 154.36.188.201:443, from local ports 54581 and 50275.

Source and analysis · Full-resolution figure

Capture coverage

The PID 5812 Process Monitor capture spans about 9.5 seconds and begins roughly 44 minutes 45 seconds after the process started. A missing network event in this short, late interval cannot negate the earlier attempt recorded in memory. Similarly, optional Sysmon ImageLoad or network collection was not verified as enabled; event absence is meaningful only after collection coverage is established. Collection coverage[2]

10 / RESEARCH

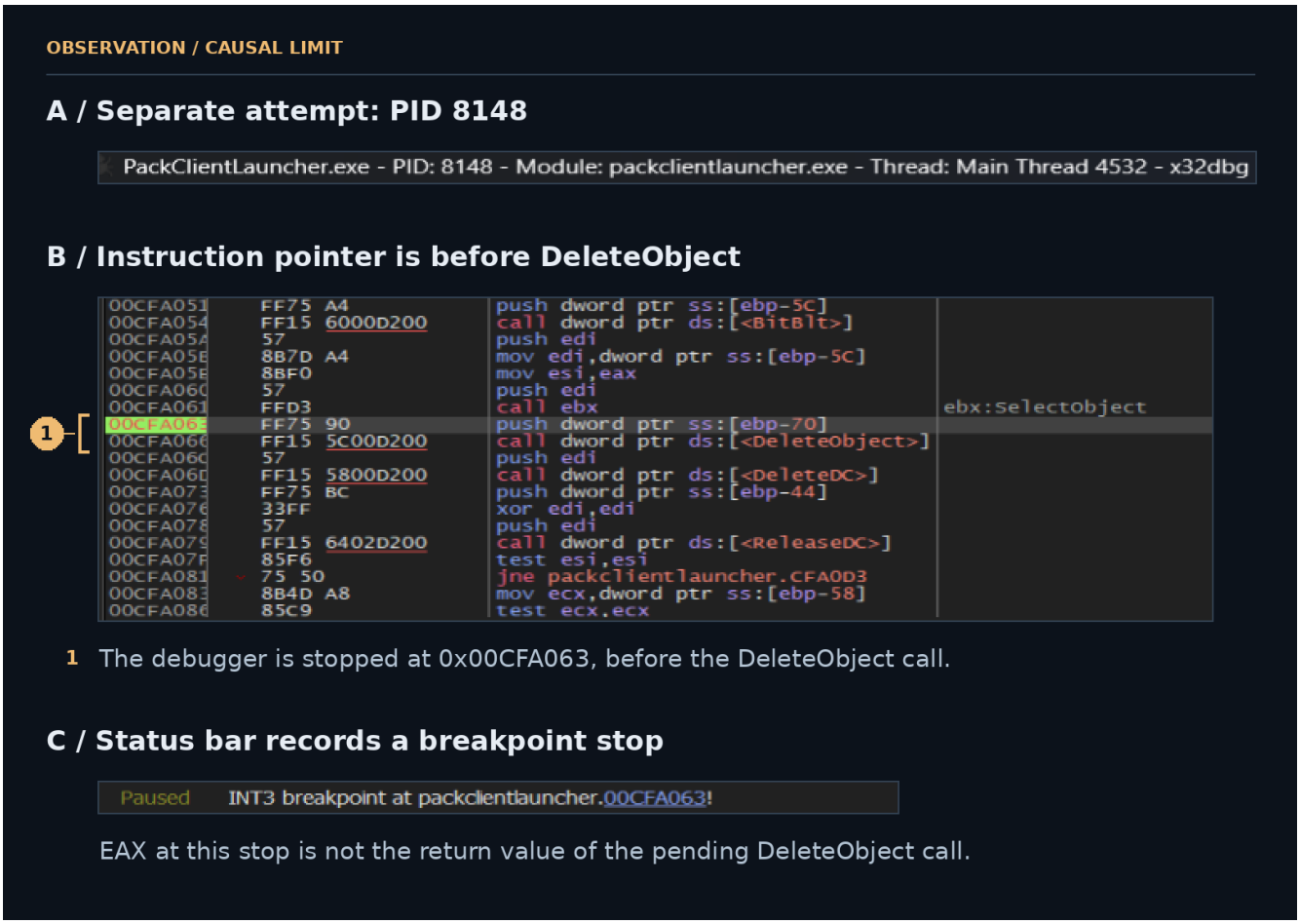
The worker failed before READY

A potentially unsafe lifetime sequence is real static evidence. The available debugger record does not prove it caused the final failure.

The recovered capture path creates a DIB, selects it, calls `BitBlt`, restores the old selection, calls `DeleteObject`, performs DC cleanup and later copies from the saved bits pointer. Microsoft's API contract explains why a successful deletion would matter: deleting the DIB frees system-owned bits. But `DeleteObject` can fail, and a breakpoint before its call does not capture a successful return. CreateDIBSection[10], DeleteObject[11]

FIGURE 19 · OBSERVATION · CAUSALITY UNRESOLVED

The debugger is stopped before deletion.



This is the PID 8148 pre-call stop. EAX at this point is not DeleteObject's return value. The image supplies call-site context; it does not prove successful deletion or establish the cause of the later PID 8040 failure.

Source and analysis · Full-resolution figure

Debugger evidence spans three separate attempts with manual changes. The process IDs and addresses cannot be combined into a controlled before/after experiment.

PID	Observation	Limit
8892	Source `0x03260000`, copy length `0x510000`; memory-map lookup rejects the source	Successful deletion and an unmodified baseline
8148	Pre-delete breakpoint; later copy breakpoint uses source `0x02830000`	A breakpoint is not an access violation or a deletion result
8040	AV at RVA `0xA567`, `push [edi+8]`, `EDI=0x13C`; zero READY bytes received	Correct application of every manual change and the crash's causal chain

The copy length `0x510000` equals `1536 × 864 × 4`, consistent with the captured monitor geometry. Arithmetic agreement does not establish buffer lifetime or ownership. Attempt-level validation[1]

FIGURE 20 · OBSERVATION · CAUSALITY UNRESOLVED

The final worker attempt failed before READY.

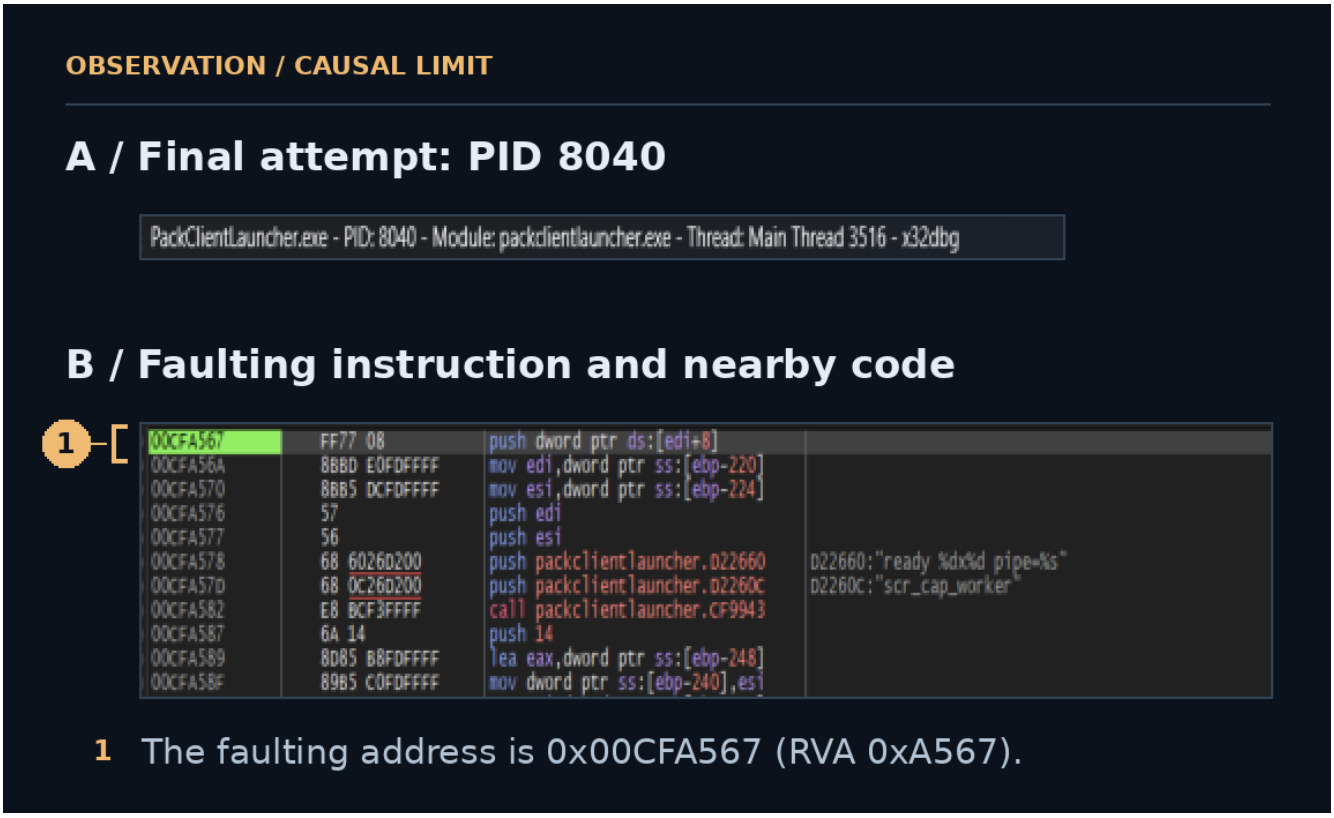
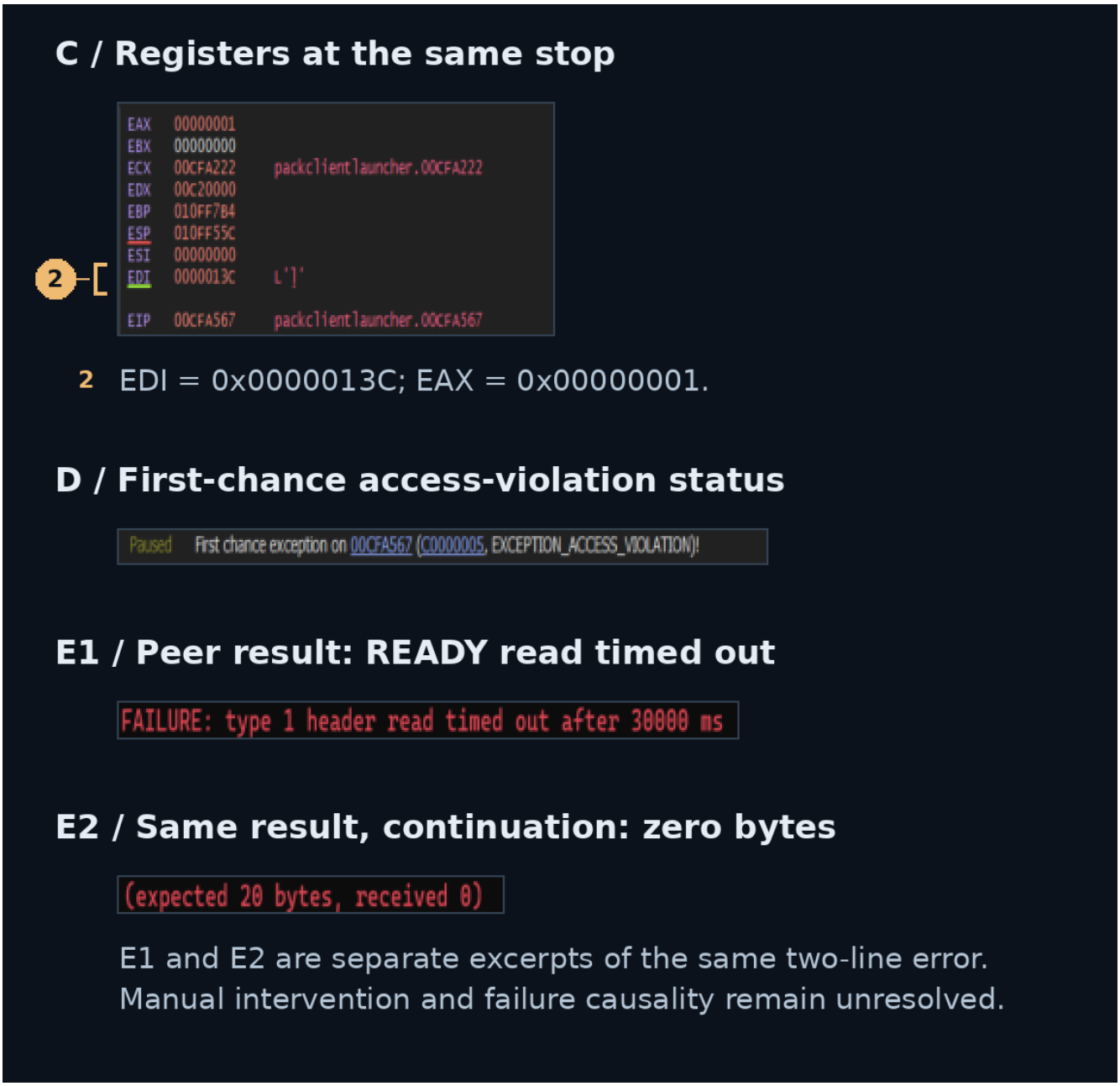


Figure continues with the remaining evidence panels.

FIGURE 20 / CONTINUED · OBSERVATION · CAUSALITY UNRESOLVED



The saved record shows an access violation at RVA 0xA567 with EDI=0x13C, followed by a zero-byte READY timeout. The attempt included manual debugger changes, so the crash does not establish a failure in unmodified execution or a causal use-after-free diagnosis.

Source and analysis · Full-resolution figure

The peer received zero of the expected 20 READY bytes before timeout. No supplied artifact captures the real `1 → 3 → 2 → 5` exchange, a type-2 framebuffer, or a successful correction. The publication therefore reports a **lifetime-hazard lead and a failed worker attempt**, while leaving successful deletion, a causal use-after-free diagnosis and a proven fix unresolved. A single build/environment record also cannot establish that PackClient screenshots universally fail. Worker validation boundary[2]

11 / RESEARCH

Tools, reproducibility and defensive use

Three passive interfaces inspect inert protocol material. An optional local IPC kit includes synthetic worker-contract tests.

The supported tooling comprises a raw-stream decoder, a PCAP/PCAPNG decoder and a Wireshark metadata dissector. The Python decoders do not create sockets, resolve domains, transmit, replay traffic or launch samples. Optional decryption and decompression are explicit; sensitive key material is redacted by default. Installation, documented inputs, examples and limits are in the tooling guide[12].

Tool	Purpose	Scope
Raw-stream decoder	Inspect framing, authentication objects and PLK1 data	Offline byte streams and synthetic fixtures
PCAP/PCAPNG decoder	Extract and inspect reconstructed protocol streams	No IP fragment reassembly or full connection-epoch separation
Wireshark Lua dissector	Display launcher protocol metadata	No encrypted-envelope decryption

Synthetic fixtures exercise the reconstructed contracts; agreement does not validate a real malware session. Independent real-capture compatibility has not been established. The published tests and documented inputs let readers check those supported components.

Synthetic screenshot IPC validation

The optional synthetic IPC appendix[13] includes a benign Windows PowerShell 5.1 peer, a fixed-pixel simulator and runnable tests. The tests cover protocol validation, image output, finite I/O timeouts and preservation of existing output files. Known BGRX and BMP hashes let a reader verify the generated image bytes. These results demonstrate agreement with the reconstructed contract; they do not establish a completed real-worker exchange or reproduce the GDI failure.

Detection interpretation

Useful defensive pivots combine an unusual colocated helper, NvSvc persistence, a bare user-context SysWOW64 surrogate, launcher-specific arguments and structurally valid protocol objects. A filename or magic string alone has weaker specificity. The detection guide[14] describes the included Sigma, Suricata and YARA detection rules and their validation boundaries. Representative production detection accuracy and false-positive rates have not been measured.

12 / RESEARCH

Contribution and remaining questions

Recovered launcher contracts extend the public campaign account; Core bytes, the external peer and several runtime paths remain unresolved.

Proofpoint’s 27 August 2026 report is the public campaign anchor. It already describes the Tax Notice chain, PackClient’s modular Launcher/Core architecture and representative protocol markers. This work builds on that context with detailed launcher contracts, passive tooling and static-to-runtime linkage. Proofpoint campaign report[15]

The previously underdocumented `1RCP` worker contract is reconstructed here alongside the authentication, cache and session-handoff behavior in this build. Prior work and attribution[16]

Documented gap	Evidence needed to close it	Effect on the present claim
Core bytes and exports	Independently identifiable Core bytes with provenance and a hash	Launcher/Core contract remains a role inference
External 1RCP peer	Identified endpoint-creation and type-2 consumption paths	No confirmed screenshot-to-network or JPEG/PV10 bridge
Successful real worker exchange	Preserved real READY/frame output tied to one controlled attempt	Static contract and synthetic agreement remain separate from live validation
Deletion/crash causality	A captured deletion result and complete intervention/attempt record	Lifetime hazard is not a proved causal diagnosis or fix
Envelope-key initialization	The writer and runtime values for the gated key globals	No inferred PSK-to-envelope-key derivation
Successful C2 and delivery	A completed session with attributable protocol/application evidence	Existing network evidence establishes attempts only
Surrogate population mechanism	Evidence of the upstream operation that places/starts the code	Private mapping does not identify an injection subtype
Eight package-byte differences	Per-site offsets and exact before/after bytes	Do not assert the three-port-substitution explanation
Runtime measurement reproducibility	Sources and invocations for the analysis helpers	Some event counts and memory comparisons cannot be independently reproduced from this repository

The recovered 20-byte protocol, authentication ordering, cache gate, process relationships and static/runtime correspondence remain the supported findings. The limitations reference[3] records the unresolved components and validation gaps.

APPENDIX A

Recorded identities

Complete SHA-256 values for the artifacts cited in the research. Short identifiers in the article are reading aids.

Campaign ZIP / Not republished

```
7108FF29916D0642 16AA2ECE7FB395F1 E3A73D12D19895BF FC0BD46806CBF85A
```

Staged IMG / Not republished

```
38EC1F5E23F65B10 AE3027BEABFA0BF7 F9FB686355A9E33C 7E7E44E6A998E04C
```

Tax_Notice_23665.exe / 120,984 bytes

```
93DD8B7B393289F8 8493596FAA4AE700 54D9EB4FE47F2DD3 34F0C6BB5262F2A8
```

nvdaHelperRemote.dll / 455,527 bytes

```
7295090C2CB63EBC 43F932451971C41F 9D015D2741E97AE3 D9855F5AE87CFF94
```

Reconstructed image B / PE32 launcher

```
46B34789196733FA B62193F0AAEDB198 B09F1362F9B10CA1 DD70CF81D68B01AD
```

PID 5812 dump / 42,481,455 bytes

```
B67AA7F4176AF979 3C0D2C53C12B757C 95C213BF82710869 7ED5B60D2C1ED663
```

PID 3696 dump / 38,687,630 bytes

```
E635CC1F72112E10 FEA9B9B9574FD7D2 F7A5D101EDBF3C33 B61FDD12071FA49E
```

Source access

Raw malware, virtual-machine disks, dumps and captures are not redistributed. The technical references preserve artifact identities and the limits of the available evidence. The repository contains the research, selected figures, inspection tools and the optional synthetic IPC kit with its tests.

APPENDIX B

Figure index

APPENDIX C

References

Numbered references correspond to the citations in the article. Technical references provide the focused supporting analysis; external sources establish prior work and API contracts.

[1] Screenshot IPC reference
https://ivanimmanuel-dev.github.io/PackClient/references/screenshot-ipc.html

[2] Evidence and method
https://ivanimmanuel-dev.github.io/PackClient/references/evidence.html

[3] limitations
https://ivanimmanuel-dev.github.io/PackClient/references/limitations.html

[4] NVDA helper documentation
https://github.com/nvaccess/nvda/blob/master/nvdaHelper/readme.md

[5] Transformed record
https://ivanimmanuel-dev.github.io/PackClient/references/architecture.html

[6] identity ledger
https://ivanimmanuel-dev.github.io/PackClient/evidence.html#identities

[7] Framing and handshake
https://ivanimmanuel-dev.github.io/PackClient/references/protocol-reference.html

[8] Token selection
https://ivanimmanuel-dev.github.io/PackClient/references/active-session-handoff.html

[9] Runtime validation
https://ivanimmanuel-dev.github.io/PackClient/references/runtime-validation.html

[10] CreateDIBSection
https://learn.microsoft.com/en-us/windows/win32/api/wingdi/nf-wingdi-createdibsection

[11] DeleteObject
https://learn.microsoft.com/en-us/windows/win32/api/wingdi/nf-wingdi-deleteobject

[12] tooling guide
https://ivanimmanuel-dev.github.io/PackClient/references/tooling.html

[13] synthetic IPC appendix
https://ivanimmanuel-dev.github.io/PackClient/references/screenshot-ipc-validation.html

[14] detection guide
https://ivanimmanuel-dev.github.io/PackClient/references/detection-guide.html

[15] Proofpoint campaign report
https://www.proofpoint.com/us/blog/threat-insight/carry-compromise-ta4922-packs-packclient

[16] Prior work and attribution
https://ivanimmanuel-dev.github.io/PackClient/references/prior-work.html