UNK_DeadDrop: An Artifact-Based Study of Recruitment Malware, Credential Collection, and Operator Infrastructure

Ilja Krasnjanski

Independent researcher

ORCID: 0009-0007-1039-5930

27 October 2026 · Preprint

Abstract

Three recruitment-themed phishing messages led to UNK_DeadDrop delivery code, payloads, configuration history, administration interfaces and targeting databases. We traced the code and repository histories, checked the recorded authenticated decryptions, and queried the databases read-only; they show a mutable command-and-control resolver, screen-capture implementations and frontend workflows linking delivery maintenance with intended wallet operations. The two databases contain 142,365 distinct address strings across 27,698 repository values; these totals measure collected targeting material, not infections. The editor-extension and Windows execution branches reproduce mechanisms in prior reports; resolver history and capture artifacts add implementation detail, while the administration source and database aggregates expose supporting operations absent from the five analyses reviewed. Historical evidence ends on 24 July 2026. Later reporting is assessed separately. Endpoint and backend telemetry were unavailable. Execution outcomes, financial loss and operator identity remain unknown.

Keywords: malware analysis; recruitment phishing; developer security; credential theft; threat intelligence; UNK_DeadDrop

1. Introduction and related work

Recruitment lures expose developers to malicious repositories, editor extensions and terminal commands presented as technical assessments; published analyses describe these delivery mechanisms, but their relationships to targeting data and administration infrastructure require additional evidence. [1–5]

This investigation began with three phishing messages received on 22 June and 21–22 July 2026, and their links and associated artifacts led to samples using Nodit, Category Labs, D2S AI Trade, Apex Systems, PredictTogether and ECHOES branding. The brands identify lure themes; links between samples depend on their code and configuration, together with the records of how they were obtained. [E01]

Proofpoint described the repository-task and Visual Studio Code extension-package (VSIX) route, native agents and Windows browser-decryption branch before this study; Vitali's earlier case provides a direct comparison, and Jamf, Microsoft and Unit 42 describe related recruitment campaigns. The original contribution comparison covers these five publications, checked on 21 October 2026. The delivery and Windows mechanisms were already public. [1–5]

A bounded public-source refresh through 25 October added an UNK_DeadDrop-specific passive-DNS investigation, a first-person TaikoTalent incident report and its same-origin follow-up material, later Overlord and MALFEX research, and current ATT&CK pages. Section 11.1 assesses these sources separately from the corpus collected through 24 July. [6–16]

The additional findings concern configuration resolution, capture implementation, administration interfaces and targeting databases. The contribution comparison is limited to details absent from those five publications. Proofpoint retains UNK_DeadDrop as a distinct cluster. We use G1052 for behavioural comparison; common attribution remains unresolved. [E03–E08; 1]

The recovered links trace how targeting and delivery feed into collection and administration, while campaign prevalence and effectiveness require population and outcome data absent from this corpus.

2. Materials and methods

The corpus combines delivery and collection artifacts with repository history, administration source and targeting data. Figure 1 maps the artifact relationships. Appendix B identifies the evidence reports and restricted source records. [E01–E21]

Collection followed the case's static-analysis and passive-retrieval rules, using public resources, research services and public blockchain reads; no sample was installed or detonated, and recovered operator credentials were not used. Historical network collection was not repeated for this revision; service availability statements therefore refer to the observation date, principally June–July 2026. [E01]

Inspection compared source control flow, configuration, archive contents, binary symbols and file hashes. The abstract syntax tree (AST) analysis parsed JavaScript without evaluating the target. The original decryption record reports successful AES-GCM authentication of the recovered plaintexts, and this study relies on that record and uses hashes to identify the inspected bytes. [E03–E05]

Both targeting databases were opened read-only for counts, an equality join and UNION queries, and the latest retained aggregate check, dated 24 October 2026, reproduced all five results without exporting contact rows. Ingestion lowercased and trimmed addresses; the aggregate queries applied no additional normalisation or person-level identity resolution. Counts therefore refer to stored address strings and repository values. Appendix C specifies the queries. [E08]

Source code supports conclusions about configured actions and control flow; endpoint telemetry is needed to confirm execution, and backend or transaction records to establish subsequent

operations. Comments, dependencies and binary symbols offer narrower evidence than a complete implementation. These distinctions determine the strength of each capability claim.

The post-cutoff refresh used the half-open window `(24 July, 25 October 2026]`, bounded public-web search and passive registry or repository access. Search results were treated as leads until the underlying page was retained, and items were classified as direct case updates, corrections, related context, derivative reporting or unverified leads. No listed lure or C2 endpoint was contacted. [6–16]

Command and control is abbreviated C2, and an indicator of compromise is abbreviated IOC. Platform mechanisms include App-Bound Encryption (ABE), the Windows Data Protection API (DPAPI), Component Object Model (COM) elevation, and macOS Transparency, Consent, and Control (TCC); their effectiveness depends on operating-system, application, policy and session conditions.

The detection-rule results in Section 10 come from evaluations recorded on 26 October 2026 and reviewed for this revision on 27 October, and they replay retained material and selected software controls; they do not extend the historical collection period. No malware execution, new infrastructure collection or fresh replay was performed for this manuscript update.

2.1 Corpus selection and retained records

The corpus is a case-derived collection assembled from three received phishing messages and the artifacts associated with their links, and follow-up examination covered linked delivery repositories, lure bundles, staged payloads, configuration resources and related administration material. Inclusion reflects documented relationships examined during the investigation; it is not a random sample of recruitment campaigns or a complete inventory of deployed variants. Lure branding alone was not treated as evidence of common ownership. [E01–E03, E07]

The starting messages date to 22 June and 21–22 July 2026, and the historical evidence cutoff is 24 July. Earlier timestamps within repository histories and build metadata describe artifact events rather than their acquisition. October aggregate checks and companion tests are reported separately from historical collection, and no single acquisition date is assigned to the entire corpus. [E01, E08, E11–E12, E15, E21]

Variant identification uses SHA-256 and byte size for retained files, supplemented by configuration, source comparisons, archive members and recorded revision identifiers. A matching hash identifies byte-equivalent inputs; different hashes require content comparison before attributing the difference to configuration or behaviour (for example, the two equal-length JavaScript payloads differ in embedded ZIP timestamps rather than configuration). Separate delivery branches and the incomplete ECHOES delivery chain remain distinct where their payload relationships are unproven. [E03–E05, E11]

Retained inputs include messages, lure JavaScript, the VSIX, encrypted and recovered plaintext payloads, native binaries, repository material, frontend source and two SQLite databases. Availability varies within each evidence report. Some ownership, infrastructure, upstream-comparison and development-network observations are preserved only in analytical notes because the original responses or snapshots are unavailable. The ECHOES final payload and the administration backend were not recovered. These gaps constrain independent checking of the affected claims. [E01–E21]

The supporting reports identify retained inputs by size and SHA-256 and distinguish artifact excerpts from secondary analytical records. Fingerprints describe the bytes reviewed at preparation; they do not authenticate acquisition times or certify an unbroken chain of custody. Selected excerpts omit credentials, raw contact data and personal attribution leads. The public reports therefore support review of the analysis without releasing a complete forensic corpus.

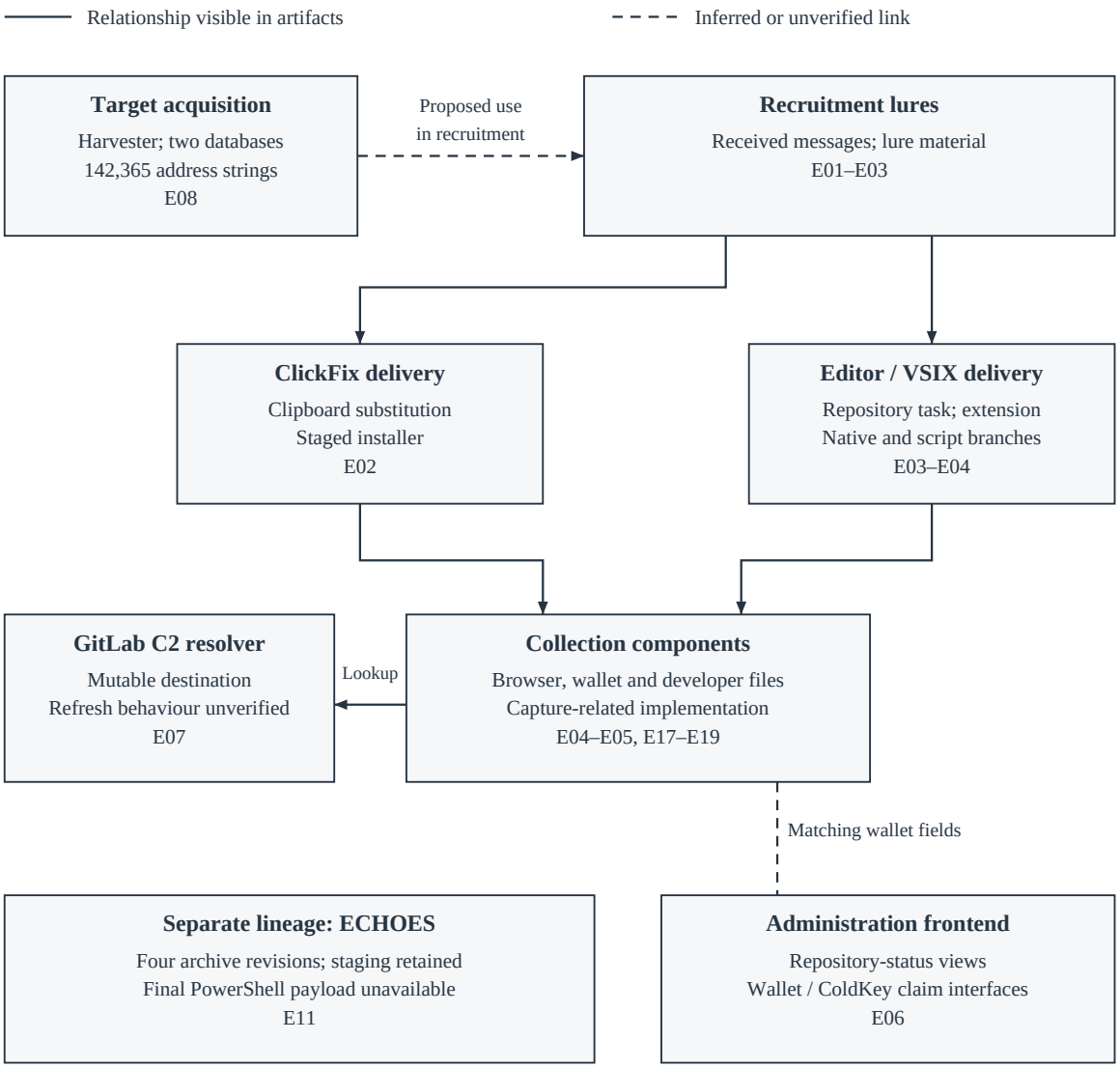


Figure 1. Relationships supported by the recovered corpus. Solid arrows represent relationships visible in source or retained artifacts, not observed execution. Dashed links indicate proposed use of targeting data in recruitment and correspondence between collection targets and frontend wallet fields; delivery and backend transactions remain unverified. The resolver link applies to the inspected payload, not necessarily every component. ECHOES is shown separately because its final payload is unavailable. Evidence IDs refer to Appendix B.

3. Target acquisition and recruitment lures

The recovered harvester selects repositories by star range, visits their branches to extract author and committer addresses from commit metadata, and lowercases and trims addresses before filtering them. Worker configurations partition collection, while scheduler and merge code consolidate results into two databases. A saved scheduler log describes an April 2026 run. [E08]

The analysis note describes sender configurations referencing both databases, with different intervals, cooldowns and warm-up settings, and also records Ondo and NXLog message templates. Those sender configurations are not retained among the exhibit inputs, so their relationship to harvesting remains a reported code observation. Actual delivery volume requires mail records. [E08]

The safe pool contains 79,698 rows and the antispam pool 62,667; their address fields have no exact overlap under the tested comparison. Their union contains 142,365 distinct address strings and 27,698 distinct repository values; repository values overlap between the pools, so adding the separate repository counts would overstate the union.

The union measures targeting material, not unique people, delivered messages or compromised endpoints. The three received messages provide separate evidence of delivery. [E01, E08]

Mail-provider relationships derive from configured hosts and message headers rather than variable names. The guide describes Namecheap Private Email settings, while the analysis record reports Mailgun-related evidence in later messages. The retained guide also discusses spam filtering and a shared outbound relay; that relay supports a provider association without linking unrelated tenants to the campaign. [E09]

Lure material includes company impersonation, portfolios, testimonials, aged accounts and functional-looking projects. The clean PredictTogether source differs from its weaponised deployment, making the delivered repository, archive and revision the relevant unit of inspection. Four payload-bearing forks and public recruitment replies provide exposure leads, but task execution and extension installation require endpoint records. Creation times and related artifacts suggest that one apparently victim-owned repository may instead have been staged by an operator; its ownership remains unresolved without provider records. [E01–E03, E10]

4. Delivery mechanisms and payload staging

The ClickFix branch embeds a verification step in a recruitment-themed web application. Its copy handler substitutes installation logic for the displayed command while preserving a plausible verification result; this design allows the verification interaction to accompany a background installation attempt. Device checks and command builders vary across the recovered bundles (Unix-like commands use detachment and output suppression, while Windows commands use hidden-window execution). Some copied strings include a trailing newline, whose effect depends on terminal paste behaviour. The handler substitutes the clipboard contents. Terminal execution was not reproduced. [E02]

The retained ClickFix chain has three stages. The first-stage samples contain approximately 41 KB of shell or 39 KB of batch content, including an encoded installer and installation-status reporting, and verification, bot-check and design-check filenames recur across related droppers. Stage two handles prior instances, removes stale files, configures primary and fallback stores, and can stage Node.js 20.18.1; an official-distribution fallback reduces dependence on an existing Node installation. The macOS branch removes the downloaded runtime's quarantine attribute. Retrieval still depends on the configured locations being reachable. [E01, E02]

Stage three contains an AES-256-GCM envelope comprising a 12-byte nonce, a 16-byte authentication tag and ciphertext. The loader uses an embedded key, decrypts in memory, invokes Node's module-compilation machinery and schedules deletion of its own file. Two recovered plaintexts are each 1,140,214 bytes but have different hashes, and content comparison attributes their differences to timestamps for two package-lock members inside embedded Windows bundles; the configuration is identical. [E01, E05]

The repository route uses editor metadata to invoke an installer. A `runOn: folderOpen` task points into a plausible project directory, while labels and cleanup reduce conspicuous material remaining in the workspace. The retained VSIX is 17,647,582 bytes, identifies itself as `google-dev-tools.google-update-support`, contains native agents matching published hashes and declares startup activation. Workspace settings supply the server URL, client identifier and token, allowing package reuse across deployments. Execution depends on editor trust, task permissions and platform-specific control flow. Startup behaviour varies with the operating-system branch. [E03; 1]

The ECHOES archives preserve a separate, incomplete delivery chain, where four revisions preserve lure and encoding structure while changing repository-based retrieval configuration. The third revision adds VBS persistence and an elevation request; the fourth changes only the destination and token. The referenced final PowerShell stage was unavailable through the examined

unauthenticated routes and was not recovered. The revisions document changes to staging. The final payload's capabilities remain unknown. [E11]

The restricted repository catalogue dated 24 July places ECHOES and the weaponised `predict-together` copy under the same account; of 172 public repositories catalogued across the examined accounts, only these two and one further assessment repository carried delivery kits. This catalogue is not reproduced in E01–E21. Common account control connects the delivery repositories. A shared final payload remains unverified.

ECHOES adds a third entry pattern: a password-protected archive and shortcut launch PowerShell, while repository-held retrieval configuration changes between commits. It shares repository-based staging and destination rotation with other branches. Its collection capabilities remain unknown because the final payload was not recovered. [E11]

5. Static deobfuscation and platform-specific execution

The VSIX loader used string-array indirection that prevented simple literal recovery. The recorded Acorn workflow parsed wrapper functions and mapped their decoder indices and offsets without evaluating the target. Candidate rotations were compared across resolved call sites, and the selected rotation produced printable decodes at all fifteen tested sites, compared with three for the next candidate. Key reconstruction preserved the source order of nine decoder results and seven literal fragments. [E04]

The resulting key authenticated three payloads: the credential-handling Python component, the Windows Node agent and its orchestration script. Successful authentication, rather than plaintext readability, supports the recovered key and payload association. This procedure recovered the key and Windows architecture already described by Proofpoint. [E04; 1]

The Windows collection branch uses Node.js and Python without requiring the native Go agent, and it can run the editor's Electron executable under `ELECTRON_RUN_AS_NODE=1`, allowing a legitimately signed binary to interpret an unrelated collection script. Native-agent hashes alone therefore do not cover this branch. Process arguments and ancestry show how the interpreter was invoked; the script path identifies its input. [E04]

5.1 Privilege handling

The recovered Windows source constructs PowerShell that requests an elevated COM object and invokes its shell-execution method through a runtime-created delegate. Interface identifiers are assembled from numeric fragments. Similar CMSTPLUA-related builder logic appears in the VSIX and ClickFix branches, supporting shared tooling. The intended child process is the editor executable acting as a Node interpreter; fallback behaviour is also present. [E19]

The source attempts elevation. Whether it completes without a prompt requires runtime testing; successful activation depends on policy, integrity level, user group and component registration. Correlated COM activation, process ancestry and browser-data access would distinguish an attempted launch from completed elevated collection. [E19]

Linux and macOS branches can solicit passwords through desktop dialogs and test submitted values using `sudo`, and recorded loops allow repeated prompts for approximately ten minutes on Linux and thirty minutes on macOS. Accepted input can support subsequent credential recovery and persistence. The record contains no representative endpoint test of these paths. [E01, E04]

5.2 Framework components and sample-specific modifications

The Go implant derives from the public Overlord client framework; its module path and runtime dependencies connect it to Overlord, as do its remote-access functions and transport configuration. Operator identity remains unresolved. The reported upstream comparison did not locate the sample's build revision in the public history examined, and an unpublished clone, pruned history or other metadata differences could explain the missing revision. [E18]

The comparison identified SQLite-backed browser collection and wallet-related modules absent from the selected upstream files. The Windows JavaScript/Python branch, staged collection and GUS configuration namespace provide additional evidence of deployment-specific logic. The original upstream snapshot is not retained, and the comparison does not cover every historical version, fork or plugin. [E18]

Capture evidence also differs by artifact type. Selected capture and audio dependencies were absent from one Linux module listing, while later binary inspection identified capture and frame-difference symbols; the symbols support capture-related implementation even where selected dependencies are absent. Working capture remains untested across the builds. [E05, E18]

6. Credential collection and screen capture

Browser collection targets Chromium profile files, Firefox `key4.db` and `logins.json`, and platform-specific credential stores. The Windows Python component includes DPAPI handling and logic intended to obtain ABE material through browser elevation interfaces, and staging paths and comments describe attempts to satisfy caller-path requirements. Unix-like branches target Safe Storage, desktop keyrings and macOS Keychain material. [E01, E04]

Collection phases stage browser and wallet artifacts before attempting additional credential recovery. An elevated process does not automatically obtain secrets protected by the original user's session; recoverability depends on browser version, policy, session state and execution context. [E04]

Wallet selectors include 35 extension identifiers, standalone wallet paths, mnemonic-related filenames and Bittensor ColdKeys; ColdKey and mnemonic fields also appear in the administration frontend, linking collection targets to intended wallet workflows. Developer-secret selectors cover SSH material, cloud and package-manager configuration, environment files and private-key extensions. These selectors extend potential exposure to source control, package publication and cloud administration, subject to file presence and access. [E01, E06]

The `.kdbx` extension appears in a generic file-selection set, but no dedicated KDBX parser or demonstrated master-password attack was identified in the reviewed material. Collection would expose the encrypted database. Reading its protected contents requires a separate decryption step. Other password managers may still fall within generic file selectors. [E05]

Screen-capture evidence combines binary symbols, dependencies and installer checks. The Go implant contains JPEG encoding, previous-frame state, block comparison, and frame-rate and bandwidth telemetry, and symbols including `CaptureAndSend`, `encodeBlocks`, `blockChanged` and `copyPrev` support a design consistent with transmitting changed image regions. The Windows bundle contains a native capture dependency, JPEG encoding, MessagePack and WebSocket support, and installer checks can reject bundles missing the capture module; the associated task is `GoogleUpdateSupporterNode`. These artifacts support a capture-streaming implementation. No session was recorded and frame rate remains unmeasured. [E03, E05]

The Windows bundle also contains synthetic input-injection and clipboard modules, which enable synthetic input, not necessarily keystroke capture. The reviewed material did not establish a keylogger: some apparent matches were Go TLS key-log facilities, and searches found no corresponding keyboard-hook implementation. Dynamically supplied components remain outside that assessment. [E05]

6.1 Collection phases and filesystem scope

The stage-three JavaScript accepts main, pass and phase3 upload variants, but these labels do not define exclusive classes of collected data. The phase3 branch traverses the home directory using filename and extension selectors; terms include keys, wallets, phrases, passwords, seeds, mnemonics, secrets, credentials, backups, recovery data, keystores and tokens. Candidate formats include JSON, YAML, TOML, INI, property lists and environment files. Selection is lexical; it does not identify every secret within arbitrary file contents. [E17]

Recorded defaults limit traversal to eight minutes overall, two minutes per root and fifteen seconds per directory read, and additional caps allow 2,500 directories, depth six and 500 candidate files. Environment variables can override several limits, while system, cache, dependency and cloud-storage locations are excluded. Actual collection depends on the files encountered and the effective configuration. [E17]

On macOS, readability probes precede access to selected protected directories, and failed or timed-out probes cause those directories to be skipped. Skipping unreadable directories can avoid prompts in some conditions. No universal TCC bypass was identified. Previously granted access and changing permission states can alter the outcome. [E17]

6.2 Data transfer and configuration

Uploads use chunks of two mebibytes by default, with bounded retries per chunk. A separate retry loop begins at five-second intervals and increases to a two-minute cap and is designed to continue until the work completes or execution ends. Successful transfer still depends on connectivity, accepted credentials and server responses. The inspected request path disables TLS certificate verification. Interception and upload outcomes were not observed. [E17]

Build-time placeholders identify a server address, agent token and client seed, with fallback values if substitution does not occur; GUS settings also control traversal budgets, retry behaviour and chunk sizing. Substitution and environment overrides can change effective settings between deployments. Sensitive bearer values are omitted from the public evidence reports. [E17]

7. C2 configuration and infrastructure history

A retained payload resolves its C2 address through `gitlab[.]com/drop-indexing/tasks/-/raw/main/config.json`, a dependency that separates destination selection from the distributed payload. The recorded history includes configuration rotation associated with a July domain batch, and a client consulting the updated resource could select a different server without any change to its file hash. [E07]

7.1 Resolver dependency and containment

The resolver makes its configuration object, hosting account and revision history relevant to containment. Blocking a resolved destination may leave the client able to obtain a replacement; blocking the resolver may leave a cached destination usable. Refresh frequency and failure caching remain unknown. Resolver use was confirmed only in the inspected components. Monitoring the configuration request together with subsequent connections would help distinguish these cases. [E07]

7.2 Historical service associations

July observations associate `207.241.172[.]222` with C2 and administration, `170.205.31[.]182` with lure hosting, and `23.137.105[.]75` with an earlier C2 role. These assignments draw on certificates, passive DNS and application content collected at different times, and they describe historical service associations, not continuous control of the addresses or attribution of neighbouring hosted assets. [E01, E07]

The evidence for `207.241.172[.]222` has a specific retention gap. Analysis notes report a certificate and management-panel response in third-party scans, but the raw responses and certificate are unavailable. Retained panel HTML and JavaScript came from `api.domatisc[.]ink`, not the bare IP. The panel content can therefore be inspected independently; the reported direct-IP response cannot. [E20]

Other infrastructure signals are weaker. A limited port scan cannot exclude unqueried services, and an empty vulnerability lookup says little about host security. Common OpenSSH banners and handshake fingerprints are poor attribution signals; host-key reuse is more specific, although cloned images and compromised systems can also explain it. Mail-tenant values and repeated account identifiers offer leads for provider-side comparison; shared defaults do not identify an operator. [E01, E20]

Certificate Transparency records provide timing context but no cross-domain certificate link; the 23 July enumeration recorded each registered base separately, sometimes with a wildcard and subdomains such as `api.`, `www.` and `email.`. No certificate combined two bases. Wildcards also limit what subsequent logs can reveal about new subdomains.

Several dated observations suggest recurring provisioning patterns. Certificates for four bases were first issued on 13 July, when the recorded resolver configuration changed to `api.cklomino[.]ink` and `api.dsfmot[.]xyz`. On 23 July, `aazdias[.]ink` and `fallaazdis[.]ink` were registered about six minutes apart through the same registrar and privacy service, using one Cloudflare nameserver pair, and their primary-plus-`fall` naming resembles the earlier `aavzx[.]ink` and `fallbaav[.]ink` pair. Several older lure domains also received certificates that day.

These enumeration and registration observations survive in restricted 23 July records, not in E01–E21. Shared automation, hosting panels or registrar defaults can produce the same timing patterns; the dates therefore supplement configuration and content links rather than independently establishing a cluster. A previously proposed domain family was withdrawn after review found the association insufficient. [E07, E16, E21]

The resulting topology is a sequence of observations: configured destinations, DNS resolutions, certificates and matching content. Preserving those types and dates avoids presenting rotation, parallel service roles and changing scan coverage as one simultaneous network. [E21]

8. Administration interfaces

Recovered frontend source from a development deployment of `rms-platform` combines message metrics, database views, task boards, repository monitoring and wallet operations. Repository views expose clone URLs, availability status and refresh controls, while wallet interfaces accept imported material, display or derive addresses, request balance previews, and expose single or bulk claim actions. Chain types, mnemonics and ColdKeys correspond to endpoint collection targets. Together, these features connect delivery maintenance and intended wallet handling within the recovered application. [E06]

Only the frontend was recovered. Backend records are needed to verify endpoint operation and stored data, determine who could execute privileged actions, and confirm transfers. The same gap covers repository alerts and completed destination changes. [E06]

Recipient selection appears to occur on the server side, limiting what endpoint artifacts can reveal about a later wallet operation; resolving recipients would require backend evidence, additional configuration or confirmed transactions. A separately recovered guide covers server setup, mail configuration, GitHub-user collection and deliverability, and records deployment procedures. Team size and operator expertise remain unknown. [E06, E09]

9. Attribution leads and financial context

Repository history, commit metadata, mailbox patterns, domain records and collaboration artifacts provide attribution leads of unequal strength; reuse of specific artifacts or credentials among linked samples carries more information than similar public names. A workstation path embedded in a Python docstring offers a lead for locating source records; paths are copyable, and configured time offsets may reflect build-server settings. Physical location, nationality and developer identity remain unresolved. A cgo-enabled `darwin/arm64` build identifies its target platform; the build machine remains unknown. [E01, E12]

Victim cash-out and control of the exchange account remain unverified. The separate application ran on a development network and was distinct from the administration wallet console; its deployed bundle and raw chain-query responses are missing from the exhibit inputs, so the account depends on analytical notes. The encrypted environment archive's contents and relationship to the deployed configuration remain unresolved. [E13]

The ECHOES poster records image-generation assertions in its embedded metadata, which describe an AI-image pipeline but do not identify the person who created or deployed the artwork. No independent signature-verifier output is retained; embedded ingredient-validation assertions are not a substitute for such verification. Certificate validity alone also cannot date image creation. [E14]

9.1 Artifact chronology

The mail and build records place two events close together on 21 July: a message Date header at 01:08:32 UTC and the implant's `vcs.time` five minutes and thirty-four seconds later. The latter records a source revision, not compilation. The compilation time and its relationship to the message remain unknown. [E21]

The retained Windows ZIP contains member timestamps spanning 18 June to 20 July, including entries dated 13 July and two package-lock members dated 20 July. ZIP times can reflect build settings, repackaging or assigned values, and their apparent offset from UTC does not locate an operator, just as the Go `trimpath` setting does not identify a build host. The chronology preserves these fields as artifact metadata; linking them to particular accounts or machines would require independent records. [E12, E21]

10. Detection and incident response

Detection benefits from combining branch-specific process context with collection and configuration markers. Framework names, folder-open tasks and updater labels occur in benign or unrelated software, while wallet-upload strings, unexpected extension installation and resolver access provide more specific context when correlated with process ancestry. ClickFix begins with a terminal interaction, whereas the VSIX route depends on editor tasks and extension behaviour. A single editor-to-shell sequence therefore does not describe both branches. [E02–E05, E15]

10.1 File-content detection and validation

The cited companion snapshot contains 33 named YARA rules in 28 files, pinned to the public source revision in [17], and these cover retained binary, installer, collector, loader and administration artifacts, alongside exact hashes for selected packages and messages. The repository contains no Sigma rules: authentic endpoint logs were unavailable for replay. Workspace-configuration signatures were also removed because the original task and settings files were absent from the retained repository copies. A source signature matches persistence code. Testing an endpoint event detector requires authentic logs. [17]

The recorded 26 October replay examined 192 selected input files and 395 raw or archive-member views, and every rule matched at least one retained case object; 90 input files had a match and 102 had none. ZIP, VSIX, TAR, gzip and nested ZIP content was inspected in memory

without executing samples. Five RAR containers were scanned only as raw bytes; four ECHOES containers matched exact hashes. The two targeting databases were excluded. Copied variants and nested views inflate object counts, and exact-hash rules recognize only the listed bytes. These results measure coverage of the selected retained artifacts, while unknown variants and independent incidents fall outside that test set. [17]

The same report records zero matches across 1,914 local standard-library and installed-package files, evaluated as 2,379 raw or nested views. This local software sample is too limited to estimate operational precision. A later control review identified seven source rules that also matched research prose, rendered exhibits or public security tools; those controls reproduced the source strings used by the rules. Narrower conditions removed those hits while preserving the artifact matches.

The wallet-agent rule illustrates the remaining limits. It requires ELF magic at byte zero, a size below 64 MiB, an Overlord module path, an environment marker and at least two wallet-related strings. The ELF requirement rejects a marker-rich plain-text fixture, but does not prove that matched strings belong to reachable code. Source-pattern rules can also match sufficiently complete quoted implementations, while exact-hash rules miss any changed bytes. Analysts must interpret matches using file type, acquisition context and the matched content. [17]

10.2 Static indicators of the JavaScript branches

The JavaScript branches differ in what they expose before execution. In the examined lures, the complete ClickFix command is assembled at copy time from a nonce, a phrase and the visitor's platform, through builders that add detachment, output suppression or a hidden window. Inspecting only literal arguments to clipboard calls misses the assembled command; the indicative structure is the builder code together with a copy handler that replaces the user's selection. [E02]

The VSIX hides module names and configuration behind an RC4-encoded, rotated string array, and Section 5 shows that the array can be recovered without execution. Obfuscation is a weak indicator on its own, because commercial extensions ship the same obfuscator. The Windows branch's use of `ELECTRON_RUN_AS_NODE` is weak in the same way, since language servers and command-line shims launch the editor binary as Node legitimately. A generated PowerShell script that sets the variable and runs hidden adds suspicion, although legitimate software can also combine these behaviours [18]. The stage-two loaders expose a tighter combination: embedded base64, AES-GCM authentication and decoded text passed to module compilation. [E04, E05]

Automated detection of these structures without execution, and its evaluation on registry corpora, are the subject of a companion paper. [18]

10.3 Endpoint event correlation

Relevant sequences include repository opening, task execution, script or runtime retrieval, and an editor executable subsequently acting as a Node interpreter. Browser-database access, wallet-file access and outbound transfers can provide evidence of later collection stages. Individual events remain ambiguous because developer tools routinely launch processes and access project data; correlation should account for delivery branch, operating system and alternative event ordering. [E02–E05, E19]

A task definition supplies a name and action to search for in execution logs, but those logs are needed to confirm it ran. Likewise, a wallet selector identifies files to examine for exposure, while capture-task activity and session timing are needed to assess screen exposure. Incident scoping depends on correlating these artifacts with host events.

10.4 Incident scoping and response assessment

Scoping should cover installed extensions, services, scheduled tasks and staged runtimes beyond the lure directory. Credential exposure depends on the executed component, collection phase and accessible files; browser sessions, password-manager databases, cloud tokens, SSH keys and wallet material consequently require different assessments. Screen exposure additionally depends on which capture component ran and when. [E01, E04, E05]

Network investigations should retain resolved addresses alongside resolver dependencies and observation times. Historical DNS and TLS records support retrospective reconstruction; current blocking decisions require current telemetry and ownership checks. Shared provider relationships alone do not justify blocking an entire range. [E07, E20]

The response notes record one preservation request as filed and others as drafted or proposed. The exhibits contain no provider acknowledgement or measured disruption outcome. They therefore document intended response actions rather than confirmed suspension or reduced victimisation. [E01]

11. Discussion and limitations

The combined artifacts connect targeting, delivery maintenance and collection within a broader administration workflow; the resolver shows how destination selection can change independently of payload bytes, while wallet interfaces correspond to selectors in collection code. These relationships extend the examined literature beyond the already reported VSIX and Windows mechanisms. Their contribution is an implementation-level account of dependencies, with no inference of successful end-to-end operation. [E06–E08; 1–5]

The evidence varies substantially in completeness. Retained source supports control-flow inspection, whereas symbols and dependencies support narrower capability assessments. Some infrastructure, upstream-comparison and development-network findings survive only in analytical notes; the ECHOES final payload remains unavailable, and authenticated decryption is reported from the original record. Restricted inputs further prevent independent public replication of the database totals and several artifact analyses.

Attribution remains unresolved because several observations admit competing explanations. Shared hosting can connect unrelated domains. Repository ownership requires more than creation timestamps; empty RPC results may reflect query coverage rather than wallet inactivity. These limits are particularly important where the underlying responses are no longer retained. [E05, E10, E13, E16]

The main unresolved questions require different forms of evidence. Representative endpoint telemetry would test resolver refresh, capture availability and platform-specific execution; backend records or confirmed transactions would distinguish intended wallet operations from completed transfers; and provider-side records would help assess account control and infrastructure relationships. None of these outcomes can be inferred from the targeting totals or frontend interfaces alone.

Technique mappings must follow the implemented behaviour: encrypted payload storage is distinct from encrypted C2 transport, and a wallet-claim interface is insufficient evidence of completed theft. Recruitment themes alone also cannot assign these behaviours to another group. [E01]

11.1 Post-cutoff public-source update

Hopkins's UNK_DeadDrop-specific passive-DNS investigation was published on 23 July and updated on 16 August, and it reports `api.chestsoi[.]ink` in the same GitLab dead-drop sequence examined here. The original publication predates the evidence cutoff, although it was identified during the later review. The retained July capture lacks that resolver generation, so the domain is listed separately as a source-reported lead. [E07; 6]

On 9 October, a first-person TaikoTalent report described a recruitment lure whose launcher used the `deepfake_guard` name, the same POSIX double-fork form, `/var/tmp/.kdm.XXXXXX`, and `/guard-launcher-result`, with paired primary and fallback `.ink` hosts. That combination is sufficiently distinctive to treat the incident as a probable UNK_DeadDrop continuation lead. The article, Gist, repository and mailing-list follow-up nevertheless share one reporting origin; the reported sample was not independently obtained in this refresh. The mailing-list exchange also disputed the claimed disclosure history, which the author subsequently corrected. The new domains and hash therefore remain research leads rather than independently verified operational indicators. [E01, E02; 7–11]

Later WatchGuard reporting documents an Overlord deployment through Brazilian legal-notice lures and a Windows VBS chain, while SafeDep describes the MALFEX `imgdecode` native Node add-on. A September MalwareBazaar record supplies another Overlord sample with distinct infrastructure. These sources document other uses of the framework and related tooling. Shared operators or campaign identity with UNK_DeadDrop remain unverified. [E18; 12–14]

The G1052 page, last modified 31 July, did not list `T1102.001`, `T1566.002` or `T1552.001` when checked on 25 October; the separate T1102.001 page likewise did not name G1052. This is a bounded page-state observation, not evidence that MITRE considered and rejected the case mappings. [15, 16]

12. Conclusion

The UNK_DeadDrop artifacts connect recruitment delivery to a mutable C2 resolver, browser and wallet collection, screen-capture implementation and an administration frontend. Read-only queries reproduce 142,365 distinct stored address strings. That is the size of the targeting corpus; victim counts remain unknown. The resolver history, capture artifacts, frontend source and database aggregates extend the examined reporting, while the VSIX and Windows branches reproduce known mechanisms.

Resolver revisions identify destination changes. Installed extensions and staged runtimes provide host artifacts to correlate with process events, while collection selectors bound the files at risk. Successful execution and wallet transfers require endpoint and backend records; operator identity remains unresolved. Later public reports provide a probable continuation lead and framework comparisons, kept separate from the historical corpus. [6–16]

Data availability and ethical handling

The underlying archive contains malware, harvested contact data, credentials, provider identifiers and unverified identity leads, so it is not deposited with this preprint. Supporting reports E01 to E21 contain edited observations, selected exhibits, methods, limitations and source fingerprints. Appendix B lists their roles. These reports differ from the public references 1 to 18 and do not provide independent replication.

The public companion repository contains inert fixtures and 33 experimental YARA rules. This paper pins the source snapshot to the commit cited in [17].

The companion paper [18] describes and evaluates the static scanner in the same repository. No software archival DOI was verified for this revision. The restricted case archive and targeting databases are not included.

Public materials include aggregate counts, sample hashes, defanged historical indicators and analytical descriptions. Raw contact rows, usable credentials and personal attribution leads are excluded. Public-interface collection does not itself establish consent from people represented in the recovered data. No sample execution or recovered-credential use occurred during this editorial revision.

Appendix C makes the aggregate method inspectable, but public replication requires access to the withheld databases. Hashes permit comparison by readers who independently possess the

relevant files. The YARA checks are artifact-coverage and selected-control validations; the scanner has a separate evaluation [18]. Historical acquisition and decryption remain attributed to the original case records.

References

[1] Proofpoint. Don't Fear the Repo: UNK_DeadDrop Phishing Campaign Targets Developers to Steal Cryptocurrency. 8 June 2026.

[2] Denys Vitali. I was likely targeted by DPRK in a sophisticated developer malware campaign. 25 May 2026, with June update.

[3] Jamf Threat Labs. Threat Actors Expand Abuse of Microsoft Visual Studio Code. 20 January 2026.

[4] Microsoft Defender Experts and Microsoft Defender Security Research Team. Contagious Interview: Malware delivered through fake developer job interviews. 11 March 2026.

[5] Unit 42. Contagious Interview: DPRK Threat Actors Lure Tech Industry Job Seekers to Install New Variants of BeaverTail and InvisibleFerret Malware. 9 October 2024.

[6] Ben Hopkins. Hunting UNK_DeadDrop - Uncovering the Foxtopia Stealer Through Passive DNS Pivoting. 23 July 2026; updated 16 August 2026.

[7] Edison Flores. A fake recruiter sent me a job offer that ended in my terminal: the trojan is still live. 9 October 2026.

[8] Edison Flores. taikotalent.xyz fake-recruiting terminal trojan campaign: IoCs, hashes, attack chain. Public Gist; checked 25 October 2026.

[9] Edison Flores. FOLLOW-UP: taikotalent.xyz fake-recruiter trojan: RESOLVED. 15 October 2026.

[10] Solar Designer. Re: FOLLOW-UP: taikotalent.xyz fake-recruiter trojan: RESOLVED. 15 October 2026.

[11] Edison Flores. Re: FOLLOW-UP: taikotalent.xyz fake-recruiter trojan: RESOLVED. 17 October 2026.

[12] Euler Neto, WatchGuard Threat Lab. How VBScript Malware Hides Behind Trusted Tools to Evade Detection. 22 October 2026.

[13] SafeDep Team. MALFEX Moves Its npm Loader into a Native Windows DLL. 13 October 2026.

[14] MalwareBazaar. SHA-256 2da7c151676e991ebee73da26db28e6872f655b5487dc4b424a6024a3381472a (OverlordRAT). First seen 8 September 2026; checked 25 October 2026.

[15] MITRE ATT&CK. Contagious Interview, Group G1052. Last modified 31 July 2026; checked 25 October 2026.

[16] MITRE ATT&CK. Web Service: Dead Drop Resolver, T1102.001. Last modified 11 September 2026; checked 25 October 2026.

[17] KrasForge. `unk-deaddrop-companion`, artifact-tested YARA rules, static scanner and validation report.

GitHub source snapshot, commit `ae74de8a927a8eda1b0cf5bd9136ffc86ed0f522`, 26 October 2026; public branch identity checked 27 October 2026.

[18] Ilja Krasnjanski. Static AST-Based Triage of Socially-Engineered Developer Lures: Proving Obfuscator Rotations Without Execution. Companion preprint, 29 October 2026. Software: `unk-scanner`.

The five-source contribution comparison was checked on 21 October 2026. References 6–16 were added by a separate bounded refresh on 25 October. It did not re-open the historical evidence cutoff or contact listed lure or C2 infrastructure. The new web references were retained in a dated Noesis refresh; dedicated citation- archive operations were unavailable to the current principal, so the paper does not claim an independent archival deposit. Reference 17 identifies the 26 October public companion snapshot. The historical E15 record remains unchanged.

Appendix A. Sample hashes and historical indicators

The held VSIX has SHA-256 `52886aab179f26421678ff23af1b0fabf0a17ffbb534369cdbbac8008cbed8e7`. The Go implant used in the companion's static YARA spot check has SHA-256 `ef183bbb40f4c7c3f4b52994cdba3295f3eab2604fdad99eefa3b67b8d966e8d`. The 1,140,214-byte JavaScript payload has SHA-256 `694626bed2df1f0d0026a3f7ead7692b44c7e73041cdd799b79a025335533cdc`. The lure JavaScript used in the clipboard-call spot check has SHA-256 `5ee915bb8d0363bd92176cc595d7425fd7e7830508d8824d90b5285a3d2b11bd`. These identify specific held files and should not be read as complete family coverage. [E03, E15]

Historical July lure or campaign observations include `apexsyshire[.]ink`, `d2saitrade[.]ink`, `joincategory[.]xyz`, `3dscenebox[.]online`, `surgecareer[.]ink` and `validator.aavzx[.]ink`. The record also associates `cklomino[.]ink`, `dsfmot[.]xyz`, `aazdias[.]ink` and `fallaazdis[.]ink` with backend or store roles. These are dated research indicators, not a current blocklist. Each infrastructure association has a recorded role and date. Malicious use requires additional evidence beyond the association itself. [E01]

Post-cutoff research leads are kept separate from those historical indicators. They include `api.chestsoi[.]ink`, `taikotalent[.]xyz`, `api.gaganata[.]ink`, `api.fallgganata[.]ink` and reported SHA-256 `c7c22e322c17bda8198257342d4034af6250a5c433242461b870879ba9aa285d`. The TaikoTalent values come from one reporting origin and the sample was not independently recovered in this refresh. These entries are research leads, not a current blocklist or an extension of the 24 July evidence corpus. [6–11]

Useful content and host artifacts include the extension identity, `GoogleUpdateSupporterNode`, `google-agent.service`, wallet-upload endpoint strings, the resolver path and the recorded collection selectors. Operator credentials, harvested addresses, personal contact details and provider account identifiers are intentionally omitted to protect sensitive data.

Appendix B. Supporting evidence register

The supporting reports E01–E21 were prepared on 24 October 2026 from restricted case records, principally from June–July 2026, and the October companion engineering record. Each report presents the relevant observations, methods and limitations and identifies its source records by descriptive title and SHA-256. They are edited analytical reports, not independent literature sources or original forensic exports. The underlying restricted archive is not included.

[E01] Master investigation report. Scope, staging, infrastructure, corrections and historical response record.

[E02] Clipboard-lure deobfuscation analysis. Recovered handler, command construction and telemetry.

[E03] Editor-extension package analysis. VSIX identity, configuration, native-agent hashes and build differences.

[E04] Authenticated payload-decryption analysis. Recorded AST-assisted key recovery, authentication and Windows source.

[E05] Payload capability re-examination. Capture artifacts, selectors and bounded negative searches.

[E06] Administration frontend analysis. Recovered repository-monitoring and wallet-management interfaces; no backend.

[E07] Configuration and infrastructure history. Resolver dependency and historical hosting associations.

[E08] Target-acquisition code and two SQLite databases. Harvester implementation and read-only aggregates; queries in Appendix C.

[E09] Recovered deployment and mail guide. Mail-provider configuration and workflow interpretation.

[E10] Repository-ownership reassessment. Corrected ownership hypothesis and comparison with published indicators.

[E11] Archive lineage and configuration rotation. Four archive revisions and an unrecovered final stage.

[E12] Attribution and build-artifact notes. Identity and chronology leads; no independent personal attribution.

[E13] Separate development-application analysis. Development-network context; not proof of wallet-console recipients or theft.

[E14] Image-provenance examination. Recorded content-provenance analysis; no fresh signature verification.

[E15] Companion tooling and validation record. Parse-only AST scanner, experimental rules, an offline AST-test rerun and historical detection tests; no public release identifier had been assigned at the time of the record.

[E16] Earlier-phase retraction. Withdrawal of an attribution based on shared hosting.

[E17] Stage-three collection and configuration analysis. Filename selectors, traversal limits, directory checks and transfer mechanics.

[E18] Framework comparison. Comparison with selected public upstream files; not an exhaustive historical audit.

[E19] Windows elevation and Electron analysis. Static COM elevation attempt and script execution path; no runtime validation.

[E20] Historical origin observations. Analytical notes describing third-party scans and certificates, plus retained domain-served panel files; raw scan responses unavailable.

[E21] Build and hosting chronology. Timestamp semantics and evolving topology; fields do not independently certify compile times.

Appendix C. Aggregate verification method

Open both databases in SQLite read-only mode and attach the antispam pool as `other`; the safe pool is `main`. The relation is `emails`, with fields `email` and `repo`. Do not export row-level data. Ingestion lowercased and trimmed addresses; these queries apply no additional normalisation or person-level identity resolution. All five aggregates were reproduced read-only on 24 October 2026, as recorded in E08. The queries and results are:

`SELECT count(*) FROM main.emails;`: 79,698.

`SELECT count(*) FROM other.emails;`: 62,667.

`SELECT count(*) FROM main.emails s JOIN other.emails a ON s.email = a.email;`: 0.

`SELECT count(*) FROM (SELECT email FROM main.emails UNION SELECT email FROM other.emails);`: 142,365.

`SELECT count(*) FROM (SELECT repo FROM main.emails UNION SELECT repo FROM other.emails);`: 27,698.

An address string is the unit of analysis. Repository values are stored references, not a verified count of extant repositories. SQL equality and UNION semantics apply to the stored values; the counts do not resolve aliases, ownership or successful delivery. Re-execution requires authorised access to the restricted inputs.