E15. Companion tooling and validation record

Ilja Krasnjanski
24 October 2026 · Edited evidence report
Historical evidence through 24 July 2026; later engineering checks identified separately.

This report synthesizes the cited case records. Exhibits distinguish retained artifact excerpts from secondary analytical records and claims whose original inputs are unavailable. Excerpts may be redacted or reformatted; extraction methods and limitations are stated locally. Fingerprints identify inputs, not the truth of their claims. Bracketed S labels identify the analysis records listed at the end.

Tool scope

The companion implements a parse-only JavaScript triage scanner, an experimental YARA rule and two Sigma rules. It is separate from the original key-recovery deobfuscator. The scanner parses with Acorn, traverses the AST, recognizes selected clipboard calls and campaign strings, and reports locations and an input hash. It folds limited literal strings but does not evaluate dynamic arguments or execute the input. [S1, S2]

The documented CLI checks file type, UTF-8 encoding and an 8 MiB size limit. Parsing occurs in-process, so those checks do not prove resistance to resource exhaustion. Syntactic receiver names can be misleading; a clipboard-call finding is not proof of malicious behavior or displayed/copied-content divergence. [S2]

Detection logic

The YARA rule requires a framework marker, an environment marker and at least two wallet-related strings in a file smaller than 64 MiB. This deliberately combines framework and collection indicators. The Linux Sigma rule selects the recorded system or user systemd unit path; the Windows rule selects schtasks process creation containing the capture-task name. Renaming, different registration APIs and absent telemetry remain blind spots. [S3–S5]

Recorded validation

The 21 October validation record reports eight AST tests and four detection tests passing. YARA compiled and matched positive ASCII/wide fixtures while rejecting framework-only and partial-marker fixtures. Both Sigma documents parsed, and their positive and negative field fixtures passed. The eight AST tests were rerun offline on 24 October 2026 and passed (Exhibit E15-5). The four detection-test results remain historical and were not rerun for this exhibit set. [S1]

Held-file spot checks reported two clipboard-write findings in a 73,545-byte lure bundle, five campaign-marker findings in a 1,140,214-byte JavaScript payload, and a YARA match on a Go implant. The corresponding hashes appear in E02 and E05. [S1]

Limits

Fixture success and three spot checks are not estimates of production precision or recall. No fleet-wide evaluation, cross-platform execution test or production SIEM conversion was reported. The supplied CI configuration had not run on GitHub at the time of the record. Source and test files are hashed below to identify the reviewed implementation; the separate execution record in Exhibit E15-5 identifies the AST rerun. No public release identifier has been assigned. [S6–S8]

Exhibits

The companion repository is the authors' own code and is quoted directly. Its JavaScript unit tests were rerun for this exhibit set (Exhibit E15-5): they run offline and their inputs are inline strings that are only parsed. The Python detection tests were not rerun because `yara-python` and pySigma are not installed in this environment, so their result is quoted from the validation record. No held sample was executed. Its held inputs were only hashed.

Exhibit E15-1. Experimental wallet-agent YARA rule

Artifact: companion YARA rule `deaddrop_wallet_agent.yar` · 771 bytes · SHA-256 `4cc26ce2b70b19a1a18dc92ed05c5503d7ceaea76a1443126a9addd4c862419e`
Location: lines 1–18 (whole file)
Extraction: `cat -n rules/yara/deaddrop_wallet_agent.yar`
Supports: "requires a framework marker, an environment marker and at least two wallet-related strings in a file smaller than 64 MiB"

```
rule UNK_DeadDrop_Overlord_Wallet_Agent_Experimental
{
    meta:
        description = "Overlord framework plus wallet-collection markers; static hunting only"
        author = "UNK_DeadDrop research contributors"
        date = "2026-10-21"
        evidence_cutoff = "2026-07-24"
        reference = "docs/claim-ledger.md#c5--operator-oriented-detection"
        status = "experimental"
    strings:
        $framework = "overlord-client/cmd/agent" ascii wide
        $env = "OVERLORD_SERVER_RAW" ascii wide
        $wallet1 = "OVERLORD_COMPANY_WALLET_EP" ascii wide
        $wallet2 = "X-Company-Wallet-Client-Id" ascii wide
        $wallet3 = "/api/company-wallet/upload" ascii wide
    condition:
        filesize < 64MB and $framework and $env and 2 of ($wallet*)
}
```

Look at the condition line. A framework-only match is not enough. YARA's `MB` unit is 2^20 bytes, so "64 MiB" in E15 is exact.

Exhibit E15-2. Sigma detection logic (Linux unit path; Windows schtasks)

Artifact: companion Sigma rule `linux_google_agent_persistence.yml` · 831 bytes · SHA-256 `34047a57d90536fd90d6829298ecb7bbc4e37e1039eae261121e572671394a23`
Artifact: companion Sigma rule `windows_capture_task.yml` · 796 bytes · SHA-256 `d6efe8045af6b031829ac881026b48e191e0d834a03d2ffff765084af8591cad`
Location: Linux rule lines 7–20; Windows rule lines 12–21 (file-relative)
Extraction: `cat -n rules/sigma/linux_google_agent_persistence.yml rules/sigma/windows_capture_task.yml`
Supports: Linux rule "selects the recorded system or user systemd unit path"; Windows rule "selects schtasks process creation containing the capture-task name"

```
references:
  - 'hxxps://attack[.]mitre[.]org/techniques/T1543/002/'
tags:
  - attack.persistence
  - attack.t1543.002
logsource:
  product: linux
  category: file_event
detection:
  selection_system:
    TargetFilename: '/etc/systemd/system/google-agent.service'
  selection_user:
    TargetFilename|endswith: '/.config/systemd/user/google-agent.service'
  condition: selection_system or selection_user
---- windows_capture_task.yml ----
logsource:
  product: windows
  category: process_creation
detection:
  selection:
    Image|endswith: '\schtasks.exe'
    CommandLine|contains|all:
      - '/create'
      - 'GoogleUpdateSupporterNode'
  condition: selection
```

The `----` separator line is not part of either file. The reference URL is defanged. The Windows rule's own description says it "Does not cover task registration through COM or other APIs". This matches E15's blind-spot statement.

Exhibit E15-3. Scanner size limit and literal-folding bound

Artifact: companion scanner `src/inspect-js.mjs` · 4841 bytes · SHA-256
`adc936d308f486ab522d5696069a52591d71099028b3cf2d3def7f5b9bffdf9b`
Location: lines 8–24, 33–39, 91, 100, 105
Extraction: `sed -n '8,24p;33,39p;91p;100p;105p' src/inspect-js.mjs`
Supports: "8 MiB size limit"; "folds limited literal strings but does not evaluate dynamic arguments"; parsing in-process with Acorn; UTF-8 check

```
export const MAX_BYTES = 8 * 1024 * 1024;
const MARKERS = /OVERLORD_COMPANY_WALLET_EP|X-Company-Wallet-Client-Id|\/api\/company-wallet\/\
upload|GoogleUpdateSupporterNode/g;

// Only literal syntax is folded. No variable resolution, calls or coercion.
export function literalString(node, depth = 0) {
  if (!node || depth > 32) return undefined;
  if (node.type === 'Literal' && typeof node.value === 'string') return node.value;
  if (node.type === 'TemplateLiteral' && node.expressions.length === 0) {
    return node.quasis[0].value.cooked ?? undefined;
  }
  if (node.type === 'BinaryExpression' && node.operator === '+') {
    const left = literalString(node.left, depth + 1);
    const right = literalString(node.right, depth + 1);
    if (left !== undefined && right !== undefined) return left + right;
  }
  return undefined;
}
  if (Buffer.byteLength(source) > MAX_BYTES) throw new Error('Input exceeds 8 MiB limit');
  let ast;
  try {
    ast = parse(source, { ecmaVersion: 'latest', sourceType: 'module', locations: true });
  } catch {
    ast = parse(source, { ecmaVersion: 'latest', sourceType: 'script', locations: true });
  }
    if (stat.size > MAX_BYTES) throw new Error('Input exceeds 8 MiB limit');
    if (count > MAX_BYTES) throw new Error('Input exceeds 8 MiB limit');
  const source = new TextDecoder('utf-8', { fatal: true }).decode(bytes);
```

Folding covers only string literals, interpolation-free template literals and `+` of literals. Recursion stops at depth 32. The limit is checked on `stat`, again after a bounded read of `MAX_BYTES + 1`, and inside `inspect()`. `TextDecoder` with `fatal: true` rejects malformed UTF-8. One line is hard-wrapped with `\`.

Exhibit E15-4. Recorded validation results (21 October 2026)

Artifact: companion validation record `docs/validation.md` · 1873 bytes · SHA-256
`2a252efae1075c45709f131ddf58c6068720f9450d85c25c3572c85ef2495caf`
Location: lines 3, 10–17, 24–28
Extraction: `sed -n '3p;10,17p;24,28p' docs/validation.md`
Supports: "eight AST tests and four detection tests passing"; YARA/Sigma fixture results; held-file spot checks

```
Local validation performed 21 October 2026. Evidence remains dated July 2026.
| Check | Result |
|---|---|
| Node.js 22.22.2, Acorn 8.17.0 | 8 AST tests passed |
| Python 3.14, yara-python 4.5.4, PyYAML 6.0.3, pySigma 2.0.1 | 4 detection tests passed |
| YARA | Compiled; positive ASCII/wide fixtures matched; framework-only and partial-marker \
fixtures did not |
| Sigma | Both documents parsed with pySigma; positive and negative field fixtures passed |
| Whitepaper | Exactly 4 A4 pages; all four pages rendered and visually reviewed |
| Input execution | No held sample was executed or installed |
| Input SHA-256 | Check | Result |
|---|---|---|
| `5ee915bb8d0363bd92176cc595d7425fd7e7830508d8824d90b5285a3d2b11bd` | AST scan of \
73,545-byte lure JavaScript | 2 clipboard-write findings |
| `694626bed2df1f0d0026a3f7ead7692b44c7e73041cdd799b79a025335533cdc` | AST scan of \
1,140,214-byte JavaScript | 5 campaign-marker findings |
| `ef183bbb40f4c7c3f4b52994cdba3295f3eab2604fdad99eefa3b67b8d966e8d` | YARA scan of held Go \
implant | Wallet-agent rule matched |
```

Three table rows are hard-wrapped with `\`. Lines 30–33 of the record add that "The CI workflow is supplied but has not run on GitHub". This is a recorded result, not attested output. The record has uncommitted edits (5 added lines, 4–8) relative to companion commit `e4fddd8`. The table rows did not change.

Exhibit E15-5. Fresh offline rerun of the AST test suite

Artifact: companion test file `test/inspect-js.test.mjs` · 1956 bytes · SHA-256 `864bc2ea0f4814e7bf1370ea7a1cd7bcfd959fd2e0465f5db76eca5c073019c3`

Location: TAP output of `node --test` (Node v22.22.2, Acorn 8.17.0 from the companion's node_modules), run 24 October 2026

Extraction: `node --test test/inspect-js.test.mjs`

Supports: "eight AST tests ... passing"

```
ok 1 - recognises literal-computed clipboard method and preserves location
ok 2 - optional clipboard chain with dynamic argument remains unresolved
ok 3 - comments and unrelated methods do not create clipboard findings
ok 4 - finds campaign marker across literal concatenation
ok 5 - does not execute submitted code
ok 6 - does not infer identifiers or interpolated strings
ok 7 - module and script syntax supported; malformed syntax rejected
ok 8 - oversized input rejected before parsing
1..8
# tests 8
# pass 8
# fail 0
# cancelled 0
# skipped 0
```

Only the `ok` and summary lines are shown. The per-test YAML blocks (durations) are omitted. The earlier E15 text said all checks were historical. The report now distinguishes this AST rerun from the historical detection checks. This rerun reproduces the 8/8 AST result on the same Node and Acorn versions. The four Python detection tests were not reproduced, because `yara` is not importable here.

Exhibit E15-6. Held spot-check inputs are present and match the recorded hashes

Artifact: held lure JavaScript `app.js` · 73545 bytes · SHA-256 `5ee915bb8d0363bd92176cc595d7425fd7e7830508d8824d90b5285a3d2b11bd`

Artifact: held stage-three JavaScript `stage3.js` · 1140214 bytes · SHA-256 `694626bed2df1f0d0026a3f7ead7692b44c7e73041cdd799b79a025335533cdc`

Artifact: held Go implant ELF · 15790242 bytes · SHA-256 `ef183bbb40f4c7c3f4b52994cdba3295f3eab2604fdad99eefa3b67b8d966e8d`

Location: whole files (hashed only; not opened, scanned or executed)

Extraction: `sha256sum app.js stage3.js overlord_go_implant_linux-amd64.elf; stat -c '%s %n' <same>`

Supports: held-file spot checks, 73,545-byte and 1,140,214-byte inputs, Go implant

```
5ee915bb8d0363bd92176cc595d7425fd7e7830508d8824d90b5285a3d2b11bd  app.js
694626bed2df1f0d0026a3f7ead7692b44c7e73041cdd799b79a025335533cdc  stage3.js
ef183bbb40f4c7c3f4b52994cdba3295f3eab2604fdad99eefa3b67b8d966e8d  overlord_go_implant_linux-\
amd64.elf
73545 app.js
1140214 stage3.js
15790242 overlord_go_implant_linux-amd64.elf
```

One line is hard-wrapped with `\`. All three hashes and both stated byte counts match the validation table in Exhibit E15-4. The finding counts (2 clipboard writes, 5 markers, YARA match) were not regenerated. Regenerating them would mean running the scanner and YARA over held malicious inputs, which is outside this exhibit pass.

Claims without a retained primary artifact

- The four Python detection tests passing and the YARA/Sigma fixture outcomes. The test code is retained (`tests/test_detections.py`, 3531 bytes, SHA-256 `1fbcb9f791fd6cb3c4e3dbfd7699a1ec0d88614aa7b0b15da8e358d395e56193`), but no captured run output is retained. Only the validation table records the result.
- The spot-check finding counts (2, 5, match). No saved scanner JSON or YARA output is retained. The record says raw finding text was deliberately not published.
- "The supplied CI configuration had not run on GitHub": this is stated in the record. A never-run CI cannot have an artifact.

Exhibit inputs

Retained inputs identified by size and SHA-256. These include artifacts and analytical records; each exhibit identifies its evidence type and extraction limitations. The files themselves are not distributed.

Companion validation record · 1,873 bytes

SHA-256 2a252efae1075c45709f131ddf58c6068720f9450d85c25c3572c85ef2495caf

Companion YARA rule · 771 bytes

SHA-256 4cc26ce2b70b19a1a18dc92ed05c5503d7ceaea76a1443126a9addd4c862419e

Companion Linux Sigma rule · 831 bytes

SHA-256 34047a57d90536fd90d6829298ecb7bbc4e37e1039eae261121e572671394a23

Companion Windows Sigma rule · 796 bytes

SHA-256 d6efe8045af6b031829ac881026b48e191e0d834a03d2ffff765084af8591cad

Companion parse-only scanner · 4,841 bytes

SHA-256 adc936d308f486ab522d5696069a52591d71099028b3cf2d3def7f5b9bffdf9b

Companion AST test suite (rerun offline) · 1,956 bytes

SHA-256 864bc2ea0f4814e7bf1370ea7a1cd7bcfd959fd2e0465f5db76eca5c073019c3

Companion detection tests (not rerun) · 3,531 bytes

SHA-256 1fbcb9f791fd6cb3c4e3dbfd7699a1ec0d88614aa7b0b15da8e358d395e56193

Held lure JavaScript (hashed only) · 73,545 bytes

SHA-256 5ee915bb8d0363bd92176cc595d7425fd7e7830508d8824d90b5285a3d2b11bd

Held stage-three JavaScript (hashed only) · 1,140,214 bytes

SHA-256 694626bed2df1f0d0026a3f7ead7692b44c7e73041cdd799b79a025335533cdc

Held Overlord Go implant ELF (hashed only) · 15,790,242 bytes

SHA-256 ef183bbb40f4c7c3f4b52994cdba3295f3eab2604fdad99eefa3b67b8d966e8d

Source records and fingerprints

SHA-256 identifies the reviewed source bytes, not acquisition time or every artifact described. Restricted originals are not reproduced.

[S1] Companion validation record, 21 October 2026 · 2,562 bytes

SHA-256 613d16e0bed012c96e00a7db6e2f7260b263fa5cd3941d39d523a71dfc94a74d

[S2] AST tool design · 2,011 bytes

SHA-256 b1d14b75bf59ee6c2e432f8631665e5a393ca93dda94b1e0cae38c9a35c45fe4

[S3] Experimental wallet-agent YARA rule · 771 bytes

SHA-256 4cc26ce2b70b19a1a18dc92ed05c5503d7ceaea76a1443126a9addd4c862419e

[S4] Experimental Linux persistence Sigma rule · 831 bytes

SHA-256 34047a57d90536fd90d6829298ecb7bbc4e37e1039eae261121e572671394a23

[S5] Experimental Windows task Sigma rule · 796 bytes

SHA-256 d6efe8045af6b031829ac881026b48e191e0d834a03d2ffff765084af8591cad

[S6] Parse-only scanner implementation · 4,841 bytes

SHA-256 adc936d308f486ab522d5696069a52591d71099028b3cf2d3def7f5b9bffdf9b

[S7] Scanner test implementation · 1,956 bytes

SHA-256 864bc2ea0f4814e7bf1370ea7a1cd7bcfd959fd2e0465f5db76eca5c073019c3

[S8] Detection test implementation · 3,531 bytes

SHA-256 1fbcb9f791fd6cb3c4e3dbfd7699a1ec0d88614aa7b0b15da8e358d395e56193