E12. Attribution and build-artifact assessment

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.

Evidence classes

The case records account and repository relationships, repeated public identifiers, registration data, a path embedded in a decrypted Python docstring, and build/archive metadata. These are attribution leads with different strengths. This report evaluates their evidentiary role without reproducing personal contact values, the workstation path or provider-account identifiers. [S1–S3]

Embedded development path

Usage examples in the Python payload retain a development-directory path and references to supporting scripts. Its presence in decrypted source can connect artifacts through shared text. It does not prove that the path came from the machine that compiled or deployed the malware: examples can be copied, templated or deliberately fabricated. The master report's description of the path as unforgeable is not sustained. [S1, S3]

Archive clock settings

The timezone analysis compares local DOS timestamps with UTC extended-timestamp fields where both are present. It reports approximately UTC−4 for one Windows-agent archive and UTC−7 for the VSIX and nested native-agent archive. DOS timestamps have two-second precision, which can account for small apparent differences. These comparisons describe stored time fields and possible packaging offsets; they do not uniquely identify two physical hosts or geographic locations. [S2]

A concentration of activity during business hours in a selected timezone is weak national-attribution evidence. Automated builds, repackaging, remote machines, chosen clock settings and the small selected sample all complicate interpretation. A cgo-enabled macOS arm64 build constrains the build environment but does not identify an individual workstation.

Identity and financial boundaries

Names, mailbox suffixes and persona reuse can guide further comparison but are not proof of civil identity. Registration records may provide provider-side leads; they do not establish who supplied payment or controlled an account at a relevant time. Likewise, a funding relationship or exchange association is not evidence of victim cash-out. The separate development application in E13 cannot fill that gap. [S1]

Assessment

Artifact reuse and corroborated account relationships warrant investigation. Real-world operator identity, nationality, physical location and quantified victim loss remain unestablished. No new provider inquiry, identity search or financial transaction analysis was performed for this report.

Exhibits

These exhibits show only structural evidence: stored time fields, build settings and the shape of the embedded path. Persona names, handles, the workstation account name, mailbox values and provider-account identifiers are redacted or left out of every excerpt. ZIP times were read with `TZ=UTC` so that the analyst's own timezone cannot enter the comparison.

Exhibit E12-1. Windows-agent archive: local DOS time vs UTC extended time (UTC−4)

Artifact: Windows Node agent archive (win_node_agent.zip) · 603347 bytes · SHA-256
`be5859f23f0781242d854dc24c102c694dabfc14a159b56429e313f0521141b6`

Location: central-directory entry `lib/server-resolve.js` (local header at offset 550020)

Extraction: `TZ=UTC zipinfo -v win_node_agent.zip lib/server-resolve.js | grep "^ lib/\|file last modified"`

Supports: "approximately UTC−4 for one Windows-agent archive"

```
lib/server-resolve.js
file last modified on (DOS date/time):          2026 Jul 13 02:13:14
file last modified on (UT extra field modtime): 2026 Jul 13 06:13:14 local
file last modified on (UT extra field modtime): 2026 Jul 13 06:13:14 UTC
```

The DOS field (02:13:14) is four hours behind the UTC field (06:13:14). The "local" line is zipinfo's rendering of the UTC field in the reader's timezone (UTC here), not the creator's. All 192 entries carry the UTC field (`TZ=UTC zipinfo -v win_node_agent.zip | grep -c "UT extra field modtime): .* UTC"` returns 192).

Exhibit E12-2. VSIX and nested agent archive: the same comparison gives UTC−7

Artifact: VSIX package (google-update-support.vsix) · 17647582 bytes · SHA-256
`52886aab179f26421678ff23af1b0fabf0a17ffbb534369cdbbac8008cbed8e7`

Artifact: Nested native-agent archive (google-update-support-agent.zip) · 17283435 bytes · SHA-256
`e1bf1b29e6fa3525d7f32f429290a88d6ea2890e61c06574b8ff6372aa5d0667`

Location: VSIX entry `extension/extension.js`; nested entry `google-update-support-darwin-arm64`

Extraction: `TZ=UTC zipinfo -v google-update-support.vsix extension/extension.js | grep "^ extension\|file last modified"; TZ=UTC zipinfo -v google-update-support-agent.zip google-update-support-darwin-arm64 | grep "^ google\|file last modified"`

Supports: "UTC−7 for the VSIX and nested native-agent archive"; two-second DOS precision

```
extension/extension.js
file last modified on (DOS date/time):          2026 May 12 22:10:42
file last modified on (UT extra field modtime): 2026 May 13 05:10:41 local
file last modified on (UT extra field modtime): 2026 May 13 05:10:41 UTC
google-update-support-darwin-arm64
file last modified on (DOS date/time):          2026 May 12 22:10:40
file last modified on (UT extra field modtime): 2026 May 13 05:10:39 local
file last modified on (UT extra field modtime): 2026 May 13 05:10:39 UTC
```

The difference is seven hours plus one second in both entries: DOS times are stored at two-second resolution and round up from the odd UTC second, which is the small apparent discrepancy the report mentions. The nested archive is the member extracted from the VSIX.

Exhibit E12-3. The macOS arm64 agent was built with cgo enabled

Artifact: Go agent, darwin/arm64 (google-update-support-darwin-arm64) · 13304624 bytes · SHA-256
`339907b44f161f57ff30819f422c552382ff437b3ae437463b4222cfe86bd943`

Location: embedded Go build information

Extraction: `go version -m google-update-support-darwin-arm64 | grep -E "^\S|build\s"`

Supports: "A cgo-enabled macOS arm64 build constrains the build environment"

```
google-update-support-darwin-arm64: go1.26.2
	build	-buildmode=exe
	build	-compiler=gc
	build	-trimpath=true
	build	DefaultGODEBUG=cryptocustomrand=1,tlssecpmlkem=0,urlstrictcolons=0
	build	CGO_ENABLED=1
	build	GOARCH=arm64
	build	GOOS=darwin
	build	GOARM64=v8.0
	build	vcs=git
	build	vcs.revision=85669fbcaa34ba565229b491db96ece6722cc346
	build	vcs.time=2026-05-13T05:09:31Z
	build	vcs.modified=false
```

`go version -m` only reads the build-info section; the binary was not run. The sibling darwin/amd64 and linux/amd64 agents report `CGO_ENABLED=0` with the same `vcs.revision`.

Exhibit E12-4. Shape of the development path in the decrypted Python docstring

Artifact: Decrypted Python payload (detect_malware.py) · 60054 bytes · SHA-256
`4c3fdd86a59d52a0e29f472666cbf9472b6fd83c38ef2a4b547f31c72b8ee87e`
Location: module docstring, lines 29–32 (six path occurrences across four usage lines)
Extraction: `sed -n '29,32p' detect_malware.py`
Supports: "Usage examples in the Python payload retain a development-directory path and references to supporting scripts"

```
pip install -r "c:/Users/[REDACTED: workstation account]/Documents/_Task/PShell/[REDACTED: pe\
rsona name]/[REDACTED: persona handle]/utils/requirements-chrome-decrypt.txt"
python "c:/Users/[REDACTED: workstation account]/Documents/_Task/PShell/[REDACTED: persona na\
me]/[REDACTED: persona handle]/utils/detect_malware.py" …[elided: credential-store arguments]…
powershell …[elided: execution flags]… -File "C:/Users/[REDACTED: workstation account]/Docume\
nts/_Task/PShell/[REDACTED: persona name]/[REDACTED: persona handle]/utils/Setup-ChromeEmbedd\
edPythonAndDecrypt.ps1" …[elided: arguments]…
Start-Process powershell -Verb RunAs …[elided: elevated relaunch of the same .ps1 path]…
```

Only the path structure (`c:/Users/<account>/Documents/_Task/PShell/<name>/<handle>/utils/`) and the supporting-script names are kept; the three name segments are redacted as identity leads and the command arguments are elided. Lines are hard-wrapped.

Exhibit E12-5. The dead-drop commits carry a UTC offset of +0000

Artifact: Dead-drop repository Git bundle · 4900 bytes · SHA-256
`804df377a0425d0aa7433ec2402d55ce8c913e1e4063d32565ddfda8ae807245`
Location: author and committer date fields of all eight commits
Extraction: `git log --format='%h %ad %cd' --date=raw` (in a clone of the bundle)
Supports: hour-of-day readings require a *selected* timezone; the commit times themselves carry none

```
c3c71d6 1783926658 +0000 1783926658 +0000
47b13d3 1780294802 +0000 1780294802 +0000
87b9974 1779762224 +0000 1779762224 +0000
52f6173 1779761849 +0000 1779761849 +0000
1f5c121 1779761745 +0000 1779761745 +0000
80cbe5a 1779761466 +0000 1779761466 +0000
bb62474 1779760642 +0000 1779760642 +0000
dcc229b 1779760409 +0000 1779760409 +0000
```

All offsets are `+0000`, so any business-hours reading of these commits depends on a timezone chosen by the analyst. Author names and addresses are excluded by the format string.

Claims without a retained primary artifact

- Account and repository relationships, repeated public identifiers, registration data, mailbox suffixes, persona reuse, funding or exchange associations: these rest on analysis notes and provider pages that are not retained as primary records, and are in any case excluded here.
- The hour-of-day distributions and the comparison across candidate timezones (UTC+8/+9 versus US zones) are analyst calculations, not stored fields. The raw inputs are the ZIP and Git time fields shown above.

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.

Windows Node agent archive (win_node_agent.zip) · 603,347 bytes

`SHA-256 be5859f23f0781242d854dc24c102c694dabfc14a159b56429e313f0521141b6`

VSIX package (google-update-support.vsix) · 17,647,582 bytes

`SHA-256 52886aab179f26421678ff23af1b0fabf0a17ffbb534369cdbbac8008cbed8e7`

Nested native-agent archive extracted from the VSIX · 17,283,435 bytes

`SHA-256 e1bf1b29e6fa3525d7f32f429290a88d6ea2890e61c06574b8ff6372aa5d0667`

Go agent, darwin/arm64 · 13,304,624 bytes

`SHA-256 339907b44f161f57ff30819f422c552382ff437b3ae437463b4222cfe86bd943`

Decrypted Python payload (detect_malware.py) · 60,054 bytes

`SHA-256 4c3fdd86a59d52a0e29f472666cbf9472b6fd83c38ef2a4b547f31c72b8ee87e`

Dead-drop repository Git bundle (drop-indexing/tasks) · 4,900 bytes

`SHA-256 804df377a0425d0aa7433ec2402d55ce8c913e1e4063d32565ddfda8ae807245`

Source records and fingerprints

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

[S1] Master investigation record · 65,312 bytes

`SHA-256 9f3736d9c68564553945d8427d0ba6d5b7cf12a92f559bc0d615ef2fd0e26bed`

[S2] Archive timezone analysis, 22 July 2026 · 9,389 bytes

`SHA-256 53d713bdefacffc4aa9e8b7d0ec0cac19c2bf895fe3dd7ceaaa215a4804b4954`

[S3] Embedded development-path analysis, 22 July 2026 · 8,997 bytes

`SHA-256 6eab87f118bdd56cee048f344131ef1e189556ea1155b6c880f89f8a553e9cf6`