

From Interview to Compromise: A Case Study of a Targeted Web3 Supply-Chain Attack



Paper 594

Presenting author

Ramon Garcia Seuma

reymon.devs@gmail.com



The Problem: Developer-Targeted Attacks

Crypto asset access

Devs hold private keys, deploy contracts, manage wallets and infrastructure credentials

Untrusted code everywhere

Daily workflow: clone repos, run npm install, execute code from strangers

Informal hiring processes

Interviews via LinkedIn DMs, take-home repos, screen-sharing sessions with strangers



Attackers are now embedding malware directly into developer hiring workflows



Attack Chain: 5 Phases

Phase 0 — Initial Access

LinkedIn recruiter impersonation → Calendly → Google Meet → “clone this repo”

Phase 1 — Local Trigger

npm install triggers prepare hook — executes before dev inspects the code

Phase 2 — Covert Execution

Background Node.js process spawned: `nohup node server &`

Phase 3 — Dynamic Payload

Exfiltrates `process.env` to Vercel → response executed: `new Function("require", data)(require)`

Phase 4 — Persistent C2

TCP beacon to 216.250.249.176:1224 every ~5s — 204 exchanges captured in 30 min

Key insight: delivery (Vercel, ephemeral, rotatable) separated from control (persistent TCP C2)

Taking down one does not affect the other — both must be neutralized simultaneously





Dataset and Methodology

Forensic Dataset

Network capture (PCAP)

71 MB, 30-min observation window

C2 stream extractions

204 TCP conversations via tshark

Repository snapshot

26 MB archived malicious repo

Social engineering artifacts

LinkedIn, hiring PDF, Calendly links

Reproduction Environment

Hetzner CX23 VM, Ubuntu 24.04

Docker (node:20-bullseye), restricted privileges
tcpdump on host + tshark extraction

Ephemeral SSH keypair, VM destroyed after

Analysis Approach

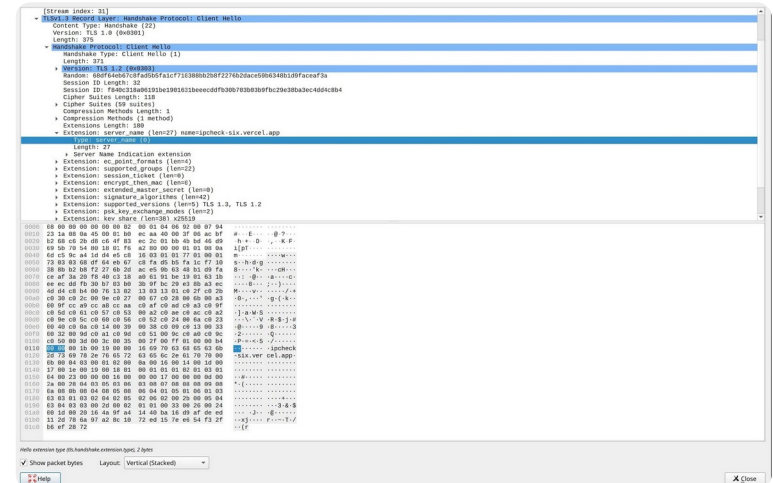
1. **Static** Source code + obfuscation → 2. **Dynamic** Sandbox + capture → 3. **Protocol** PCAP + git forensics



Key Finding: Dynamic Payload Delivery

No static malicious payload in the repository

1. prepare hook executes silently on npm install
2. Background Node.js process (cross-platform)
3. Base64 endpoint decoded, process.env exfiltrated
4. Server response executed as code:
`new Function("require", resp.data)(require)`



TLS Client Hello to ipcheck-six.vercel.app (Stage 1 loader)

Signature-based detection fails — attacker controls code at runtime, not at rest





C2 Protocol and Threat Intelligence

C2 Protocol

Target 216.250.249.176:1224, plaintext HTTP/TCP

Beacon GET /api/checkStatus every ~5s, 204 captured

Tag “crash the bad guys” (base64 tid)

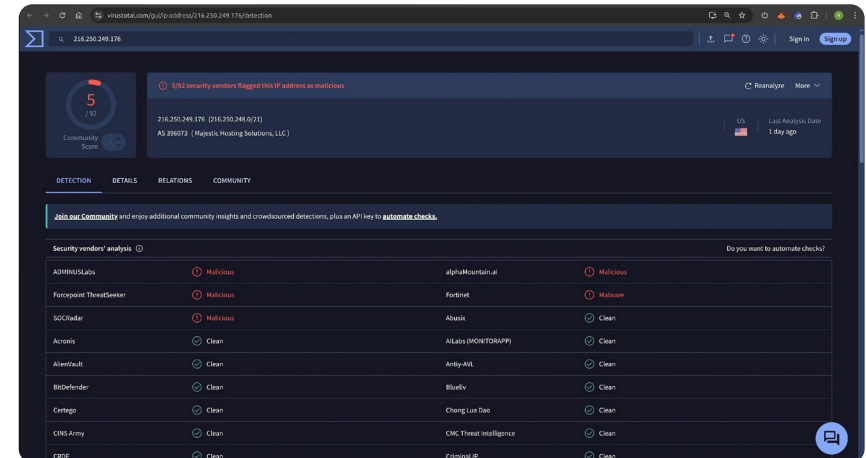
Hosting Majestic Hosting / SpinServers, TX

Threat Intelligence

VirusTotal: 5/92 vendors flag as malicious

AbuseIPDB: 1 report (email spam, Dec 2025)

Shodan / Censys: no historical scan data



VirusTotal report for 216.250.249.176 — 5/92 detections

Operationally fresh infrastructure — no confirmed overlap with Contagious Interview clusters





PWD /tmp or CI runner vs real project tree



7



Detection: Implementable Sigma Rule

Target signal

npm lifecycle hook spawning a backgrounded Node.js process during dependency installation

Detection logic

Parent npm/node with install/lifecycle/prepare

Child node with nohup / start /b / detach

Filter node-gyp, electron-rebuild, webpack

MITRE ATT&CK: T1195.002 + T1059.007

```
title: Suspicious npm Lifecycle Hook
status: experimental
logsource:
  category: process_creation
detection:
  parent_npm:
    ParentImage|endswith: ['/npm', '/node']
    ParentCommandLine|contains:
      ['install', 'lifecycle', 'prepare']
  background_node:
    Image|endswith: ['/node', '\node.exe']
    CommandLine|contains:
      ['nohup', 'start /b', 'detach']
  filter_legitimate:
    CommandLine|contains:
      ['node-gyp', 'electron-rebuild']
condition: parent_npm and background_node
               and not filter_legitimate
level: high
```

Limitation: does not detect attacks that defer process spawning until after npm install completes





Conclusions

The hiring process as a trusted distribution channel

Key contributions

1. Real-world forensic dataset of a targeted supply-chain attack
2. Full reverse-engineering of a dynamic JavaScript implant
3. Sandbox detection surface analysis + countermeasures
4. Implementable Sigma detection rule

The observed model: stealth, adaptability, selective targeting

Advanced persistent threat tradecraft applied to developer workflows





iSCSi 2023 official Sponsors

Platinum



Gold



Silver



Institutional

