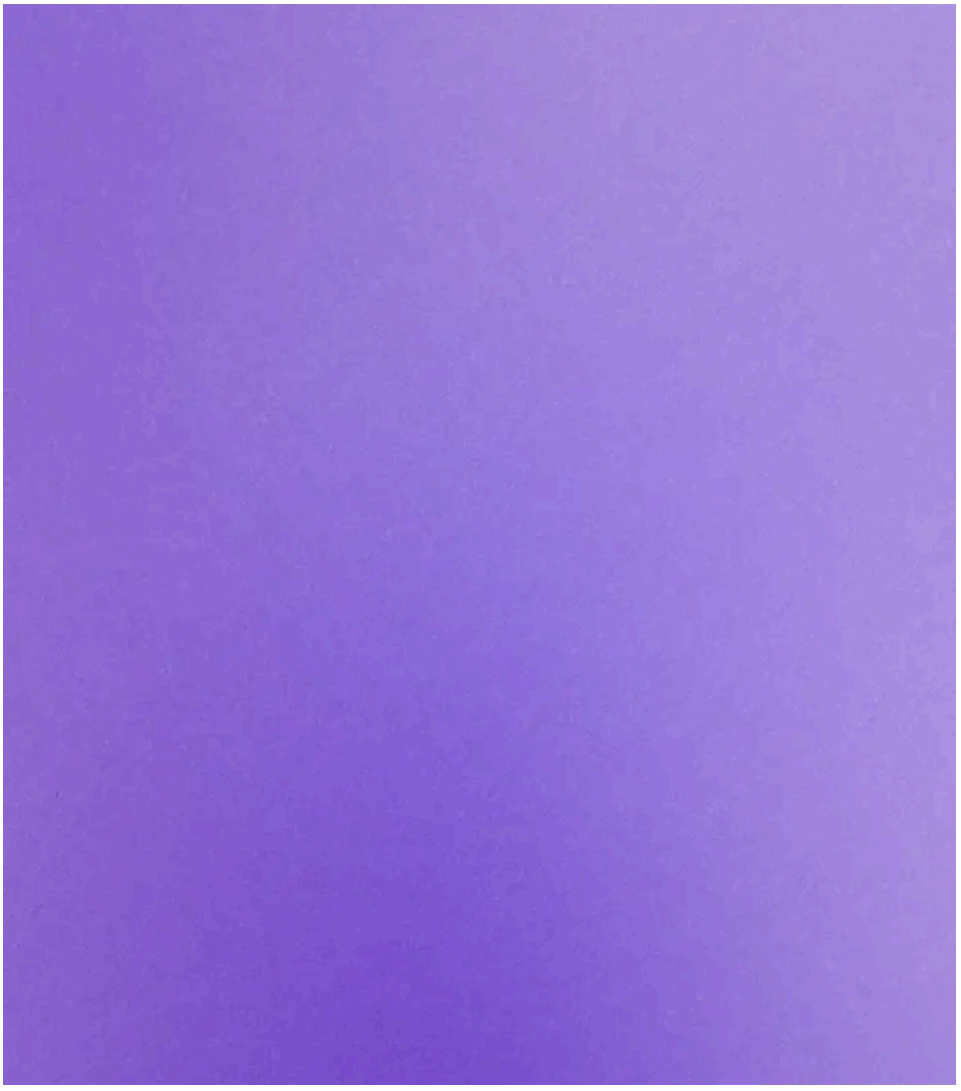
North Korea’s Contagious Interview Campaign Escalates: 338 M...

Archived: 2026-04-05 23:39:55 UTC



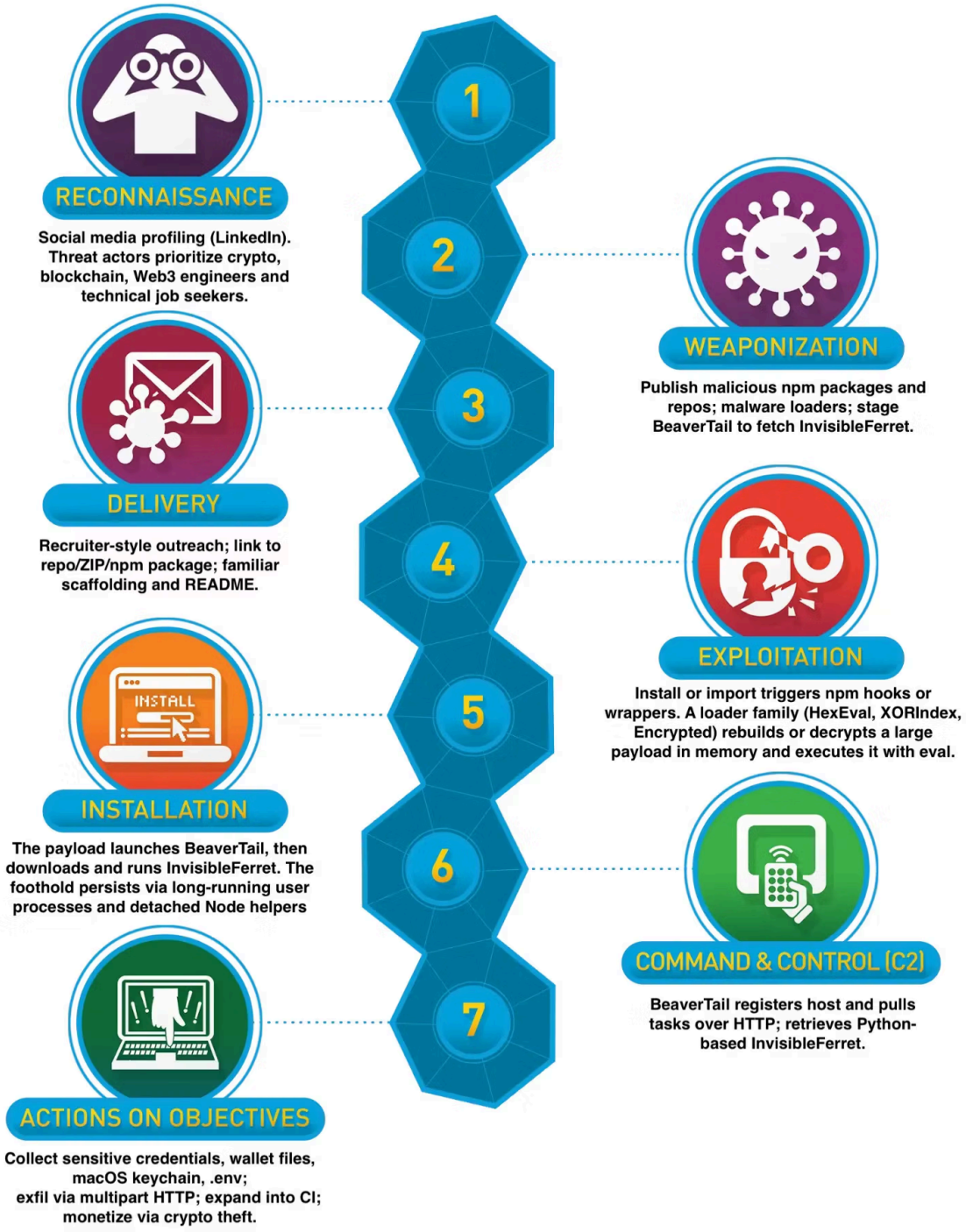
The Contagious Interview operation continues to weaponize the npm registry with a repeatable playbook. Since our July 14, 2025 update, we have identified and analyzed more than 338 malicious packages with over 50,000 cumulative downloads.

25 of these packages remain live on the npm registry at the time of writing. We have submitted takedown requests to the npm security team and petitioned for suspension of the associated publisher accounts.

In this latest wave, North Korean threat actors used more than 180 fake personas tied to new npm aliases and registration emails, and ran over a dozen command and control (C2) endpoints (see IOCs). Their tooling has evolved from direct BeaverTail malware droppers to HexEval, XORIndex, and encrypted loaders. Each executes at install or import, reconstructs obfuscated BeaverTail in memory, then typically fetches the InvisibleFerret backdoor for persistence. New malicious packages appear weekly, including this week.

The pattern is wave-based and iterative. The threat actors ship typosquatted packages, tweak the loader code, and scale distribution across new aliases.

Targets include Web3, cryptocurrency, and blockchain developers, as well as technical job seekers approached with recruiting lures, leading to multi-stage compromise and financial loss.



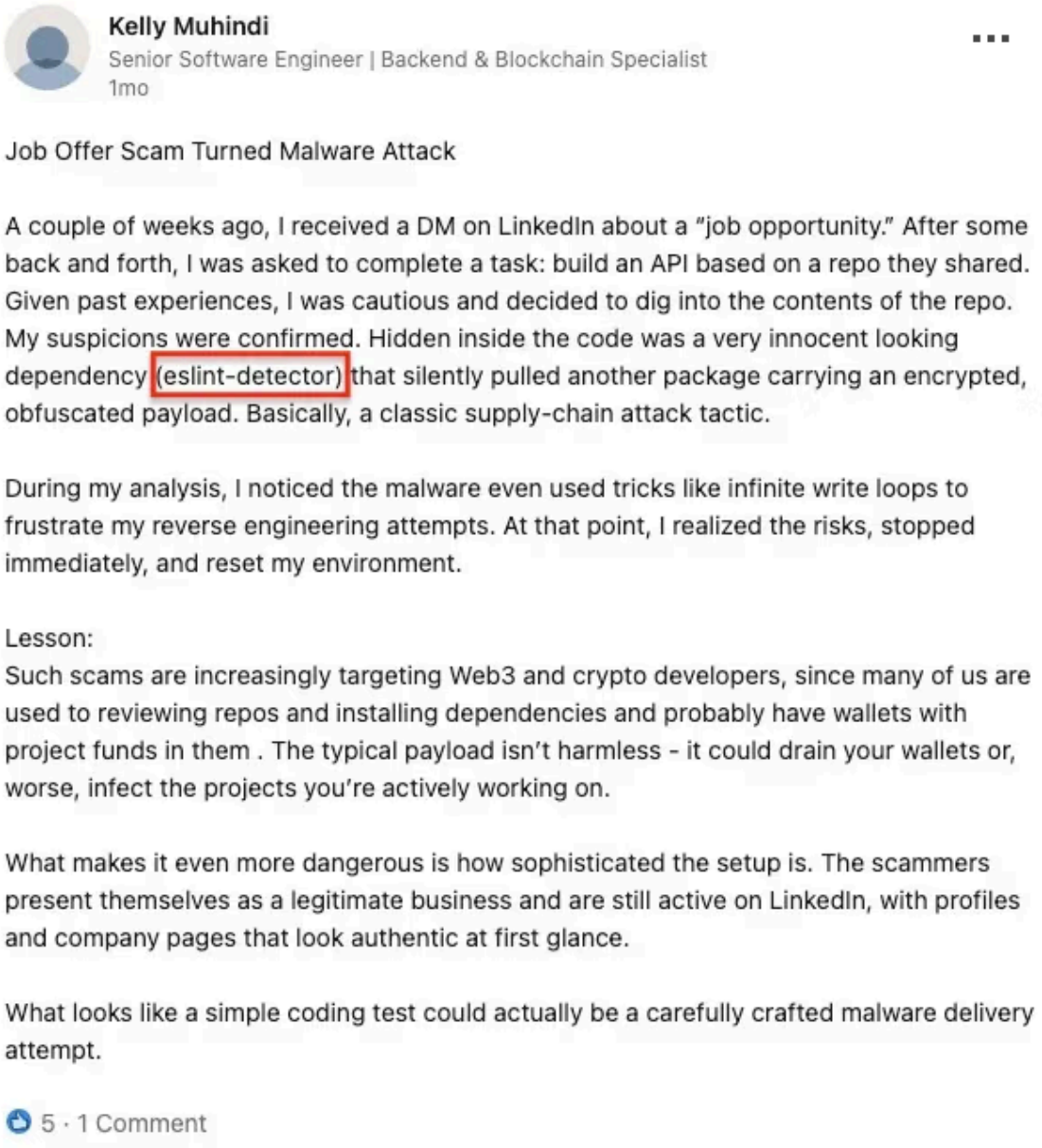
Lockheed Martin Cyber Kill Chain framework mapped to the current Contagious Interview campaign. ***Reconnaissance*** *on LinkedIn,* ***weaponization*** *with published malicious packages,* ***delivery*** *via recruiter lures,* ***exploitation*** *by malware loaders that execute in memory,* ***installation*** *of BeaverTail and the InvisibleFerret backdoor,* ***C2*** *over web protocols, then* ***actions on objectives*** *that establish initial access, and steal sensitive credentials and wallet keys.*

Stage 1: Reconnaissance#

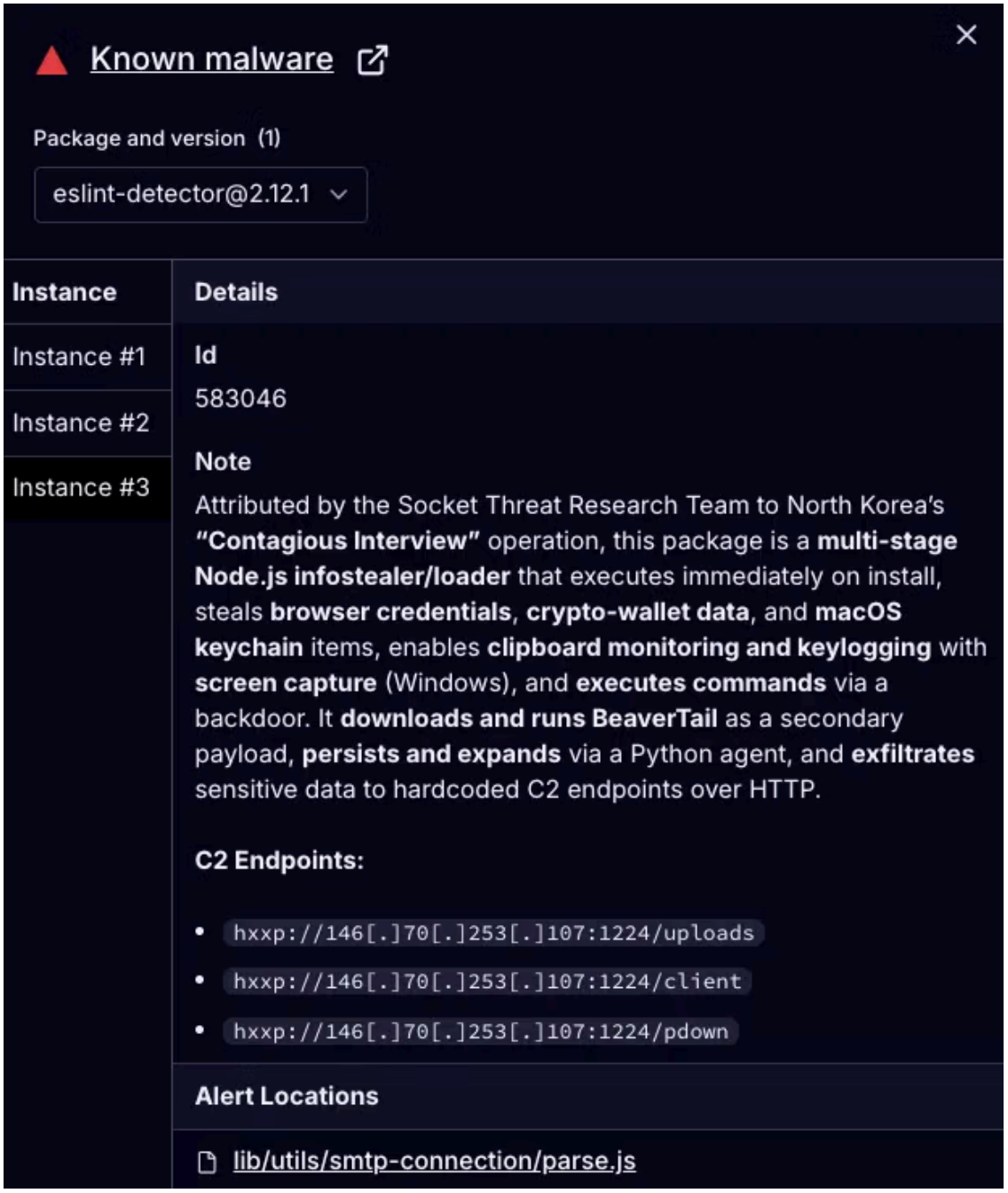
The campaign opens with focused reconnaissance. Threat actors approach targets on social media, most often LinkedIn, posing as recruiters or hiring managers. They screen for technical fit and financial upside, prioritizing

cryptocurrency and blockchain developers, Web3 engineers, and technical job seekers. The objective is to compromise machines that are likely to hold credentials, private keys, tokens, and other monetizable secrets.

A recent victim account on LinkedIn illustrates this stage. A software engineer received a “job opportunity” message, was given a repository for a quick assignment, and found an innocuous dependency named `eslint-detector` that contained an encrypted, obfuscated payload. The lure targeted a Web3 and crypto profile, relied on routine dependency installation, and used a polished company persona. What looked like a part of the recruitment assignment was a staged malware delivery.



LinkedIn victim report of a job-offer lure that delivered a malicious npm package, eslint-detector, which silently fetched an encrypted payload, illustrating Contagious Interview reconnaissance and supply chain delivery tactics.



Socket AI Scanner's analysis of the malicious eslint-detector package highlights install-time execution of a multi-stage infostealer/loader, theft of browser credentials and crypto-wallet data, macOS Keychain access, clipboard monitoring and Windows keylogging with screen capture, remote command execution, BeaverTail download with Python-based persistence (i.e. InvisibleFerret staging), and HTTP exfiltration to hardcoded C2 endpoints.

Stage 2: Weaponization#

We continue to see weekly upload bursts, rapid re-uploads after takedowns, and iterative changes to loaders and `postinstall` scripts. Independent and excellent research by Kieran Miyamoto on the DPRK Research blog (https://dprk-research.kmsec.uk/) also corroborates this pattern and closely tracks the campaign's weekly cadence across the npm registry.

Over 335 malicious packages in this wave align with the documented Contagious Interview techniques that combine job-seeker social engineering with open source supply chain abuse, notably npm typosquats, brand impersonation, and obfuscated loaders that fetch the second and third-stage malware and backdoors. Threat actors' objective is developer endpoint access, CI/CD persistence, and ultimately cryptocurrency theft and strategic espionage across blockchain, Web3, and broader tech firms.

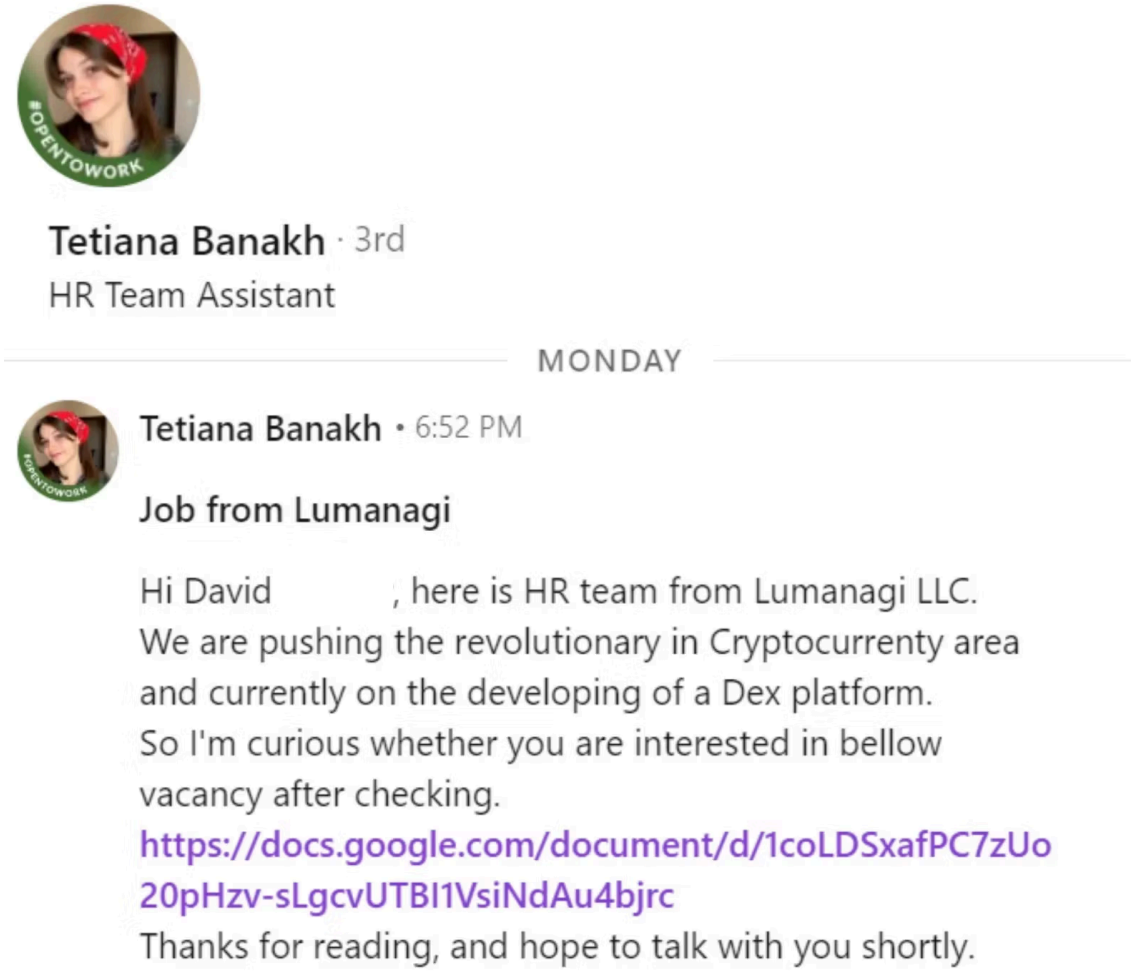
Over 335 names cluster around everyday dependencies that interview candidates and working developers install on autopilot, especially in the Node/Express stack. We see close misspellings and plausible add-ons of staples like `express`, `dotenv`, `body-parser`, `validator`, `cors`, `helmet`, `morgan`, `nodemailer`, and `nodemon`. Examples include `epxreso` / `epxresso` / `epxressoo` (Express), `dotevn` (dotenv), `boby_parser` (body-parser), `vaildator` (validator), `cors-validator` (cors), `http-helmet` (helmet), `morgan-logger` (morgan), `nodemailer-helper` (nodemailer), and `nodemon-pkg` (nodemon). As some victims report, play on deadline pressure in fake job interview assignments ("just run `npm install`") turn routine setup into initial access.

Beyond server basics, the current wave targets what developers touch constantly during quick prototypes: frontend/framework and toolchain surface area (e.g., `react-router`, `tailwindcss`, `next`, `vite`, `webpack`, `eslint`, `prettier`). We see lookalikes such as `react-router-html`, `react-redirect-router`, `nextjs-babel-toastify`, numerous [`vite`]-prefixed lookalikes like `vite-plugin-react-ping` and the near-duplicate `vvite-plugin-react-ping`, plus `vitejs-plugin-react-refresh` and `webpack-css-branch-loader`.

When it comes to crypto hiring, the Web3 kits are also targeted: `ethers.js` is typosquatted as `ethrs.js` and `ethres.js`; `web3.js` is typosquatted as `we3.js` and `wb3.js`; and there are systematic typosquats of `truffle` (e.g., `truffel`), `ganache` (e.g., `ganacche`), and `foundry` (e.g., `foudry`), as well as hardhat-themed packages like `hardhat-deploy-notifier` and `hardhat-deploy-notification`. We also see brand impersonation such as `metamask-api`. The typosquatted names mirror what candidates are most likely to search, typo, or accept in a template.

Stage 3: Delivery#

Targets often receive a series of interview messages followed by a link to a code repository. Cloning and running the project executes an initialization script on first use, which starts the malware chain. Some victims also receive links to documents or forms on common productivity platforms (e.g. Google Docs), setting up a "take home" task that delivers the payload.



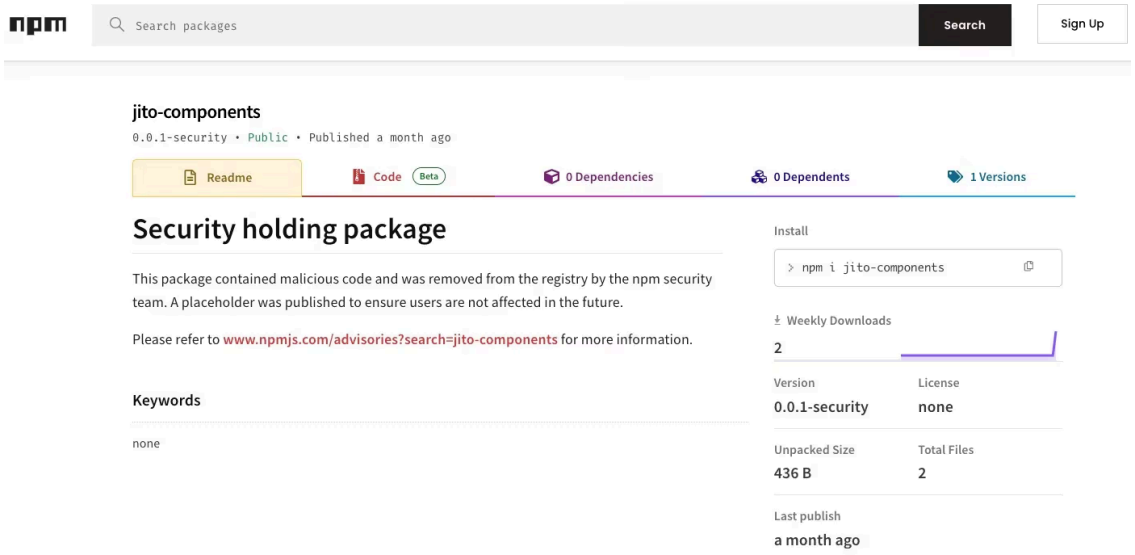
LinkedIn DM lure directing the target to a Google Docs link, a stage-one tactic that establishes a hiring pretext, pivots off-platform, and sets up delivery of a coding test with malicious dependency.

Additionally, we found that threat actors registered email addresses to look like recruiter/HR or “tech” personas that would resonate with developers and job-seekers. We see (1) recruiting/business veneer, e.g. `bob.berg.business@gmail[.]com`, `soft.business0987@gmail[.]com`, `astroglobal.work@gmail[.]com`, `jiayingzhang.contact@gmail[.]com`; (2) developer/engineering cues, e.g. `goldenrhynodev@gmail[.]com`, `luis.fernando.dev1214@gmail[.]com`, `sean_tech208@hotmail[.]com`, `stromdev712418@gmail[.]com`, `ryon_dev_3@outlook[.]com`; and (3) crypto/Web3 flavor, e.g. `jackson.tf7.eth@gmail[.]com`. These match how threat actors in Contagious Interview campaigns build plausible recruiting identities while keeping infrastructure disposable.

Stage 4: Exploitation#

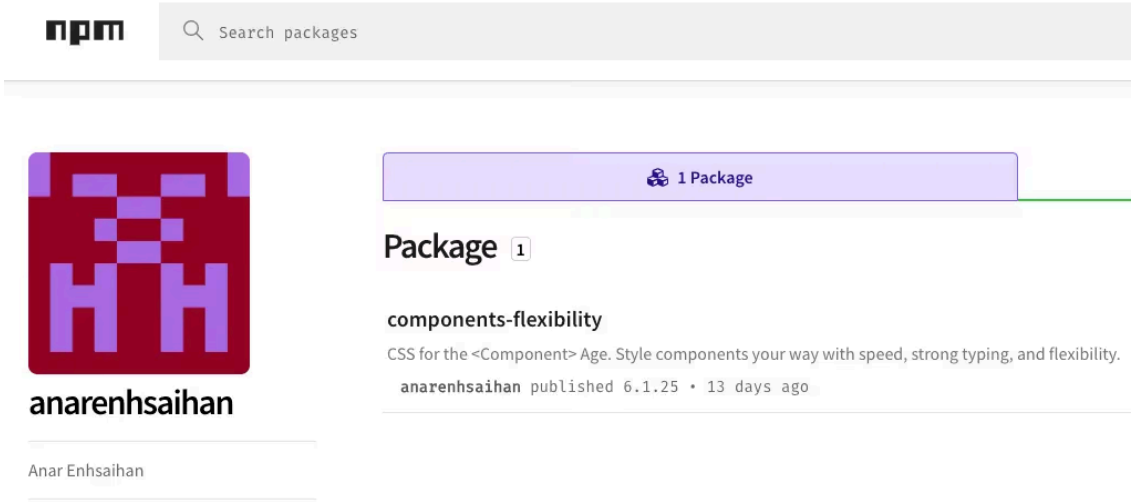
Exploitation begins the moment threat actor code executes on the target machine. In this campaign, execution is user-driven, not a vulnerability exploit. Install or import triggers threat actor logic via npm lifecycle hooks such as `postinstall`, through entry points that run code at module load, or via small cross-platform wrappers. Three loader families (described in more detail below) implement this pivot from delivery to code run.

A note on the exploitation of npm registry mechanisms by Contagious Interview threat actors. In the current wave of the npm ecosystem infiltrations, we found cases highlighting some gaps in account-level enforcement on the npm registry that threat actors are targeting for abuse. For example, the threat actors' alias `anarenhsaihan` published two malicious packages: `jito-components`, which has since been removed and replaced by a security holding page, and `components-flexibility`, which remains live at the time of writing. Both packages serve as loaders for the BeaverTail malware.



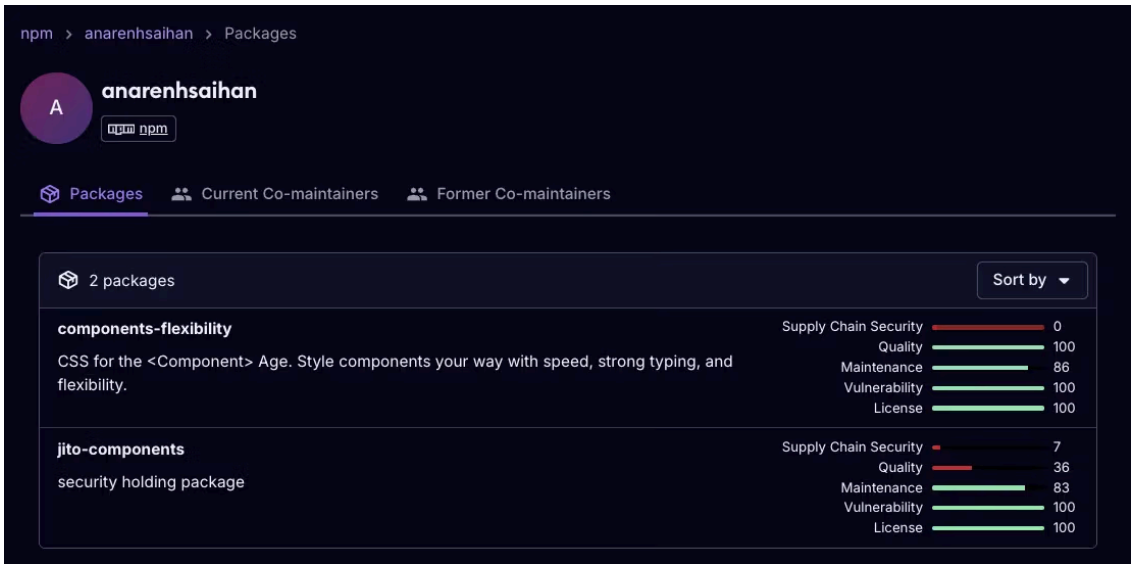
The npm registry marks jito-components as a security holding package after detecting malicious code, replacing the original with placeholder version 0.0.1-security to block installs and protect users.

Despite the `jito-components` package being flagged and removed by the npm security team, the threat actor's account was not suspended. This allowed the same alias to publish a second malicious package under the guise of a legitimate UI styling utility.



npm account anarenhsaihan with a live package, components-flexibility, indicating the alias remains active and able to publish after the jito-components takedown.

Cleaning up the ecosystem is not a trivial task, especially against advanced persistent threat (APT) actors. Contagious Interview is not a cybercrime hobby, it operates like an assembly line or a factory-model supply chain threat. It is a state-directed, quota-driven operation with durable resourcing, not a weekend crew, and removing a malicious package is insufficient if the associated publisher account remains active.

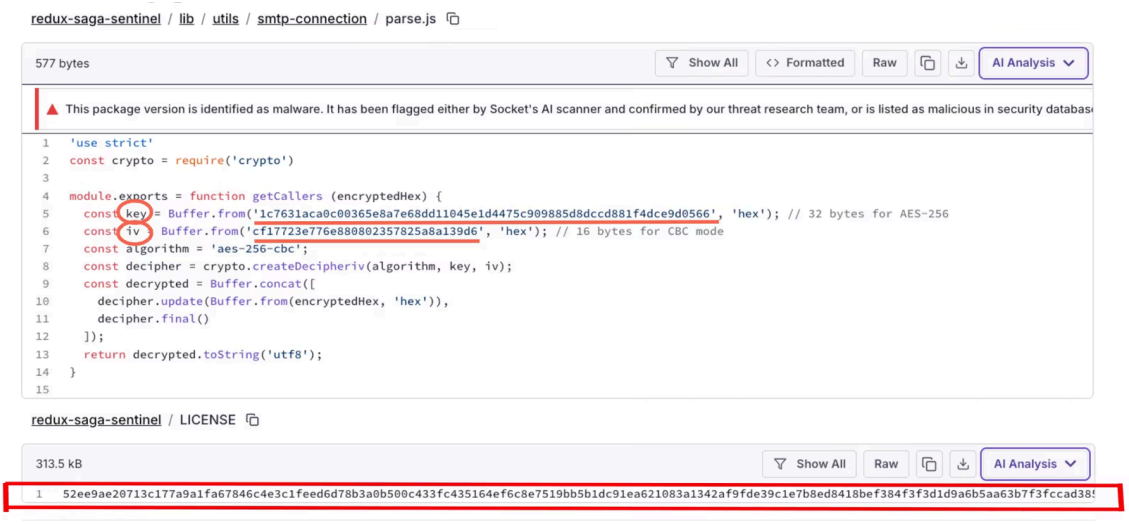


Socket AI Scanner's view of the npm alias anarenhsaihan shows jito-components replaced with a security holding package while components-flexibility remains live. Our analysis of components-flexibility highlights install-time loader behavior, in-memory execution via eval, and delivery of BeaverTail malware.

Stage 5: Installation#

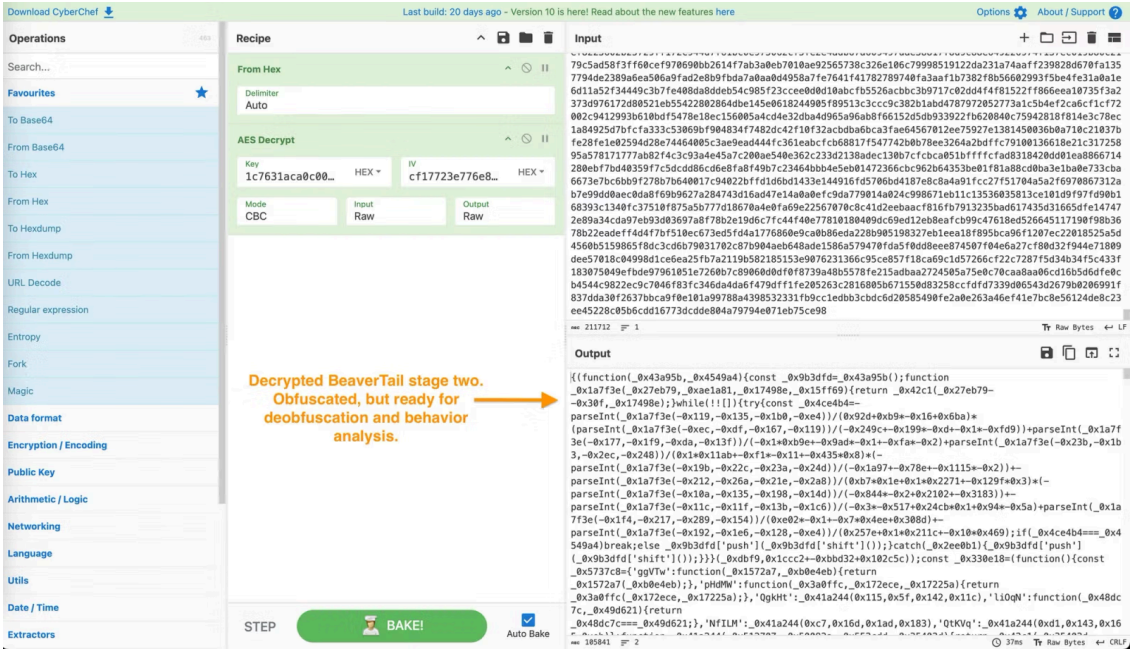
Contagious Interview packages install like nesting dolls, a small loader runs first, reconstructs BeaverTail in memory, then BeaverTail drops and fetches the InvisibleFerret backdoor. Earlier waves relied on two families of loaders. HexEval stores stage-two as long hex strings, decodes them at runtime, and evaluates the plaintext with `eval`, which transfers control to BeaverTail. XORIndex hides strings and code as XORed byte tables and rebuilds them with simple index math before executing the result. Both approaches avoid leaving a readable second stage on disk, and both appear across hundreds of malicious packages.

Recent wave added encrypted loaders. The goal is obfuscation versus cryptographic safety. The malicious packages with encrypted loaders ship a small module that imports Node's `crypto`, fixes the algorithm to AES-256-CBC, and hardcodes both the key and the initialization vector (IV). The ciphertext, a large hex blob, is stashed elsewhere in the package, sometimes in a file named `LICENSE`. At install or import, the module reads that blob, decrypts it, converts it to UTF-8, and evaluates the plaintext in process.



Socket AI Scanner's analysis of the malicious redux-saga-sentinel package highlights an encrypted loader split across two files. The top file, lib/utils/smtp-connection/parse.js, imports Node crypto and hardcodes an AES-256-CBC key and IV. The bottom file, LICENSE, stores the large hex ciphertext. At runtime, parse.js decrypts the LICENSE blob to plaintext JavaScript and executes it, enabling in-memory loader execution within the same package.

The below CyberChef panel shows how defenders can reproduce decryption: convert the hex ciphertext to raw bytes, apply AES-256-CBC with the embedded key and IV, and recover the stage-two JavaScript. The recovered body remains obfuscated, but deobfuscation confirms BeaverTail, based on its file and wallet targeting, control-flow patterns, and the handoff logic for launching the InvisibleFerret backdoor.



CyberChef reproduces the decrypt of the package's encrypted loader. Converting the hex ciphertext and applying AES-256-CBC with the embedded key and IV recovers BeaverTail stage-two JavaScript in the Output pane, still obfuscated but ready for deobfuscation and behavior analysis.

Operationally, installation means a long-running foothold rather than a guaranteed autorun. The loader starts BeaverTail, which fetches and launches InvisibleFerret. With BeaverTail active and InvisibleFerret staged, the malware is ready to register the host and begin tasking, which leads directly into Stage 6.

Stage 6: Command and Control#

BeaverTail establishes C2 over HTTP(S) and sometimes WebSocket, registers the host, fetches tasking, and stages InvisibleFerret, a cross-platform Python backdoor for Windows, macOS, and Linux.

The campaign blends raw IP C2 with platform C2. Fixed IPs on commodity VPS providers act as backends, while front-end beacons often use legitimate hosting such as `*.vercel.app` to blend into developer traffic. URIs are deliberately plain and work-adjacent, with paths such as `/api/ipcheck`, `/process-log`, and `/apikey` that masquerade as health checks or logging hooks, so a quick glance by a developer or code reviewer raises little suspicion. Infrastructure recycles across waves with small mutations. Threat actors reuse domain patterns and URL shapes, periodically switch between raw IPs and platform subdomains, and reappear on non-standard ports, historically including port `1224` and in this wave additional high ports, to evade simple egress filters.

Stage 7: Actions on Objectives#

Monetization and follow-on objectives focus on cryptocurrency theft and maintaining persistent access for further compromise. There is no vetted dollar total for this specific campaign, but independent reporting estimates that North Korea-linked threat actors have already stolen $2 billion in 2025 and approximately $1.34 billion in 2024. The social engineering workflow described here, fake recruiter personas that push candidates into running take-home assignments or “tests”, aligns with tactics Reuters reported across the crypto sector in 2025. Stolen assets typically move through layered mixers, cross chain swaps, and lower visibility networks, with investigators observing multi hop flows across Bitcoin, Ethereum, BTTC, and Tron.

Outlook and Recommendations

The campaign’s trajectory points to a durable, factory-style operation that treats the npm ecosystem as a renewable initial access channel. Across waves, we document a steady push of new malware loader variants, including recent encrypted loaders.

We anticipate more loader riffs that split decryption and staging across files to defeat static scans, continued reuse of URL shapes and hosting platforms for cover traffic, and rapid re-uploads after takedowns, especially when publisher accounts remain active.

Account suspensions help, but they are not sufficient, since accounts can be created on a whim. Registries should adopt layered controls: suspend and revoke tokens for confirmed malicious publishers; require re-verification with 2FA and provenance signing; apply pre-publish and prefetch screening to quarantine high-risk uploads; throttle suspicious velocity and namespace churn; and cluster related aliases by shared infrastructure, email patterns, and code templates so enforcement follows the operator, not the name.

Defenders should harden the points where this campaign succeeds: pull requests, installs, and CI. Treat every `npm install` as code execution and block risky behavior before it reaches developer machines or pipelines. Shift left

by scanning code and PRs in real time; require a clean report before merge and vet external libraries for provenance, maintainer trust, and pinned versions.

Socket's security tooling is purpose-built to address these challenges. The Socket GitHub App provides real-time PR scanning, flagging suspicious or malicious packages before merge. The Socket CLI surfaces red flags during installs and lets teams enforce allow/deny rules, blocking risky behaviors such as `postinstall` scripts, unexpected network egress, decrypt-and-eval loaders, or native binaries. Socket Firewall blocks known malicious packages before the package manager fetches them, including transitive dependencies, by mediating dependency requests; use it alongside the CLI for behavior-level gating. The Socket browser extension alerts users to suspicious packages while browsing. Socket MCP extends protection into AI-assisted coding, detecting and warning on malicious or hallucinated packages before they are introduced through LLM suggestions. Integrating these tools into development pipelines will help proactively detect and prevent malware, reducing exposure to Contagious Interview-style supply chain attacks.

MITRE ATT&CK#

- T1195.002 — Supply Chain Compromise: Compromise Software Supply Chain
- T1608.001 — Stage Capabilities: Upload Malware
- T1204.002 — User Execution: Malicious File
- T1059.007 — Command and Scripting Interpreter: JavaScript
- T1027.013 — Obfuscated Files or Information: Encrypted/Encoded File
- T1546.016 — Event Triggered Execution: Installer Packages
- T1005 — Data from Local System
- T1082 — System Information Discovery
- T1083 — File and Directory Discovery
- T1217 — Browser Information Discovery
- T1555.003 — Credentials from Password Stores: Credentials from Web Browsers
- T1555.001 — Credentials from Password Stores: Keychain
- T1041 — Exfiltration Over C2 Channel
- T1105 — Ingress Tool Transfer
- T1119 — Automated Collection
- T1657 — Financial Theft

Indicators of Compromise (IOCs)#

C2 Endpoints

1. `135[.]181[.]123[.]177`
2. `138[.]201[.]50[.]5`
3. `144[.]172[.]105[.]235`
4. `144[.]172[.]112[.]106`
5. `146[.]70[.]253[.]107`
6. `23[.]127[.]202[.]249`
7. `23[.]227[.]202[.]244`

8. `http://fashdefi[.]store:6168/defy/v7`
9. `https://0927[.]vercel[.]app/api/ipcheck`
10. `https://api[.]npoint[.]io/b964566497d98298d32c`
11. `https://ip-check-server[.]vercel[.]app/api/ip-check/208`
12. `https://json-project-hazel[.]vercel[.]app/apikey/QWERTYU890T12HML`
13. `https://log-server-lovat[.]vercel[.]app/api/ipcheck/703`
14. `https://process-log[.]vercel[.]app/api/ipcheck`
15. `https://process-log-update[.]vercel[.]app/api/ipcheck`

Malicious npm Packages:

1. alchmey-sdk
2. alert-codestreamer
3. async-chai
4. babel-cli-ganache
5. bind-error
6. bingo-abstract-transport
7. bingo-log
8. bingo-logger
9. bingo-pretty
10. boby_parser
11. btrez-logger
12. case-sensitive-paths
13. chai-utils
14. chartable-utils
15. checking-ip
16. checking-ips
17. chunk-logger
18. colorful-buttons
19. common-js-support
20. common-logify
21. components-flexibility
22. config-log
23. cookie-logger
24. cookie-loggers
25. cookie-loggo
26. cookie-parsing
27. cookies-logger
28. cors-validator
29. cross-session
30. ddok-escapes
31. display-notifications
32. dotevn

33. `dragon0905-vite-tsconfig-assistant`
34. `emittery-up`
35. `epxreso`
36. `epxresso`
37. `epxressoo`
38. `err-notification`
39. `error-analysis`
40. `error-fallback`
41. `error-loggerjs`
42. `eslint-config-detector`
43. `eslint-detector`
44. `eslint-logger`
45. `eslint-plugin-react-purify`
46. `eslint-ts-view`
47. `eslint-validation-cli`
48. `eslints-logger`
49. `eth-node-utils`
50. `etherres`
51. `ethres.js`
52. `ethrs.js`
53. `express-prisma`
54. `express-xmlrequest`
55. `file-uploading-advance`
56. `filigrean-icon`
57. `filigren-icon`
58. `filigron-icon`
59. `filiogrean-ico`
60. `financial-utils`
61. `flowhint`
62. `flowico`
63. `flowmint`
64. `foudry`
65. `foundary`
66. `foundrey`
67. `foundri`
68. `frontend-cron`
69. `func-analys`
70. `func-analyst`
71. `func-analysis`
72. `func-logger`
73. `fundry`
74. `gad-logger`

75. `ganac`
76. `ganacche`
77. `ganacha`
78. `ganachee`
79. `ganachhe`
80. `gannache`
81. `gatepass`
82. `glow-admin`
83. `gnach`
84. `gridmind`
85. `hardhat-deploy-notification`
86. `hardhat-deploy-notifier`
87. `hashsentinel`
88. `http-err-notification`
89. `http-helmet`
90. `http-req-logger`
91. `httpreslog`
92. `httpreqlog`
93. `husky-es`
94. `husky-logger`
95. `icon-sea`
96. `ip-checkers`
97. `ip-checking`
98. `ip-checks`
99. `item-box`
100. `jito-components`
101. `jnscript`
102. `js-notifiers`
103. `json-configs`
104. `json-confs`
105. `json-log-stream`
106. `json-weqjoken`
107. `json-webhooks`
108. `jsonlise-conf`
109. `jsons-logger`
110. `jsonstylizer`
111. `layzr`
112. `log-task`
113. `log4action`
114. `logger-cookie`
115. `logger-extjs`
116. `logger-pino`

117. `logging-winston`
118. `logflow-json`
119. `login-tokenizer`
120. `lovable-ci`
121. `lovable-cli`
122. `lovable-cookie-logger`
123. `lovable-cookies-logger`
124. `lovable-js`
125. `lovable-logger`
126. `lovable-loggers`
127. `lovable-react`
128. `lovable-ts`
129. `luma-glow-db`
130. `matrix-charts`
131. `mega-compress`
132. `metamask-api`
133. `middleware-loggers`
134. `mongodb-cd`
135. `mongodb-ci`
136. `mongodb-orn`
137. `mongose-ci`
138. `mongose-cli`
139. `morgan-logger`
140. `motionflow`
141. `mongoose-ci`
142. `muxflux`
143. `my-ttt`
144. `next-plugin-uni-i18n`
145. `nextjs-babel-toastify`
146. `node-log-config`
147. `node-log-stream`
148. `node-logflow`
149. `node-logger-sdk`
150. `node-loggerx`
151. `node-notifications`
152. `node-nvm-ssh`
153. `node-orm-logger`
154. `node-vite-config`
155. `node-winston`
156. `node-winston-logger`
157. `nodeapi-json`
158. `nodemailer-helper`

159. `nodemon-pkg`
160. `nodelog-lite`
161. `nodespode`
162. `notification-clients`
163. `notification-displayer`
164. `notification-layer`
165. `notifications-client`
166. `notifications-layer`
167. `notifications-log`
168. `orbital-ledger`
169. `parse-logger`
170. `parser-session`
171. `parser-tson`
172. `pino-node`
173. `pixzen`
174. `preset-log`
175. `prepare-config`
176. `prettier-utils`
177. `pretty-format-setting`
178. `proc-log-cmd`
179. `proc-log-error`
180. `process-load`
181. `qrcode-pretty-react`
182. `query-logger`
183. `randly`
184. `rc-logger`
185. `react-babel-purify`
186. `react-context-stylizer`
187. `react-copack`
188. `react-content-provider`
189. `react-dhtml`
190. `react-dropzone-log`
191. `react-eslint-type`
192. `react-fs-cofnig`
193. `react-fs-config`
194. `react-hook-eslint`
195. `react-icons-loader`
196. `react-lovable`
197. `react-milton`
198. `react-outcome-error-alert`
199. `react-prop`
200. `react-repack`

201. `react-redux-stylizer`
202. `react-redirect-router`
203. `react-router-html`
204. `react-router-purify`
205. `react-stylizer`
206. `react-tediter`
207. `react-thunk-log`
208. `react-toast-ui`
209. `real-socket-rt`
210. `recharts-smart`
211. `redux-eslint-saga`
212. `redux-lint-saga`
213. `redux-saga-devtool`
214. `redux-saga-guard`
215. `redux-saga-help`
216. `redux-saga-inspector`
217. `redux-saga-sentinel`
218. `redux-saga-validator`
219. `redux-thunk-action`
220. `redux-toolkit-rts`
221. `request-guard`
222. `request-kraken`
223. `request-sentry`
224. `router-kit`
225. `rtk-log`
226. `rtk-logger`
227. `rtk-service`
228. `rtk-sleep`
229. `rtk-wake`
230. `safe-winston`
231. `sensitive-paths-focus`
232. `session-logger`
233. `sessionfiy`
234. `sessions-logger`
235. `simple-icon-maker`
236. `some-promise`
237. `stake-config`
238. `stream-loggers`
239. `strictor`
240. `succgdess`
241. `tai1wind-configs-viewer`
242. `tailwind-beauty-icon`

243. tailwind-book-icon
244. tailwind-class-overrides
245. tailwind-classname-overrides
246. tailwind-classes-overrides
247. tailwind-color-icon
248. tailwind-computer-icon
249. tailwind-config-overrides
250. tailwind-config-setting
251. tailwind-configs
252. tailwind-configs-viewer
253. tailwind-cup-icon
254. tailwind-desktop-icon
255. tailwind-glass-icon
256. tailwind-icon
257. tailwind-icon-animate
258. tailwind-mouse-icon
259. tailwind-mui-modal
260. tailwind-nbr-icon
261. tailwind-next-icon
262. tailwind-react-icon
263. tailwind-react-mui
264. tailwind-round-icon
265. tailwind-scrollbar-show
266. tailwind-scrollmenu
267. tailwind-style-components
268. tailwind-style-overrides
269. tailwind-supabase
270. tailwind-theme-colors
271. tailwindcss-animatexs
272. tailwindcss-animators
273. tailwindcss-color-icons-lite
274. tailwindcss-config-overrides
275. tailwindcss-remotion
276. theta-tv-charts
277. tjsontype
278. trslip
279. truflee
280. truffel
281. tsleep
282. uidraftism
283. uxlift
284. uxline

285. `vaildator`
286. `viam`
287. `vite-audit-plugin`
288. `vite-auditlog`
289. `vite-babel-plugin-es6-promise`
290. `vite-binding-js`
291. `vite-chunk-tools`
292. `vite-chunk-manager`
293. `vite-configs-viewer`
294. `vite-css-icon`
295. `vite-jsconfig`
296. `vite-lightsparse`
297. `vite-linting-js`
298. `vite-log-plugin`
299. `vite-logeidit`
300. `vite-mobcss-log`
301. `vite-next-logger`
302. `vite-next-loggers`
303. `vite-parse`
304. `vite-plugin-chunk-chop`
305. `vite-plugin-es6-babel`
306. `vite-plugin-js-support`
307. `vite-plugin-morgan`
308. `vite-plugin-opticompress`
309. `vite-plugin-parse`
310. `vite-plugin-parse-js`
311. `vite-plugin-parse-json`
312. `vite-plugin-react-ping`
313. `vite-plugin-reactjs-refresh`
314. `vite-plugin-uni-i18n`
315. `vite-plugin-vue-layout`
316. `vite-postcss-bootstrap`
317. `vite-postcss-helper`
318. `vite-postcss-kit`
319. `vite-postcss-nested`
320. `vite-react-chunker`
321. `vite-simpleparse`
322. `vite-singleparse`
323. `vite-ts-icon`
324. `vite-tsauditlog`
325. `vite-tsconfig-assistant`
326. `vite-tsconfig-optimized`

327. `vitejs-plugin-react-refresh`
328. `vortex-logger`
329. `vvite-plugin-react-ping`
330. `wb3.js`
331. `we3.js`
332. `webpack-css-branch-loader`
333. `winstem-logging`
334. `winston-datalog`
335. `winston-log`
336. `x-session-parser`
337. `xml-request-parser`
338. `tailwindcss-theme-icons`

npm Aliases

1. `adammorris533`
2. `alexander0110818`
3. `alexander0110820`
4. `alexander0110828`
5. `anarenhsaihan`
6. `andrey0212`
7. `andrii_matsiuk`
8. `anthony_smith`
9. `ariel02`
10. `artemsdefi`
11. `artemsnpm`
12. `asd123123123123`
13. `astro123456`
14. `aylin_alkan`
15. `behrad80515`
16. `bellyache`
17. `benmilam727510`
18. `benzonjohn`
19. `bobbb`
20. `brian_sanders`
21. `brian_scott`
22. `bryankoh0604`
23. `bryanlee604`
24. `butleralvin510`
25. `carolina32123`
26. `caroline727`
27. `castiblanco`
28. `cesar510727`

29. `chain1107saw`
30. `charles1236542`
31. `charles987456`
32. `cheapdev009`
33. `cheekaide`
34. `christrotman`
35. `danicaagawin`
36. `daniel604`
37. `darielfrias`
38. `david0604`
39. `david1003`
40. `david_fernandez`
41. `david_raynolds`
42. `davidjambis`
43. `ddok`
44. `denys604`
45. `diego123123`
46. `dkeosleff`
47. `dmitriy1023`
48. `dmytro604`
49. `dragon0905`
50. `dyani-steras`
51. `elodieblanc0707`
52. `emily0102`
53. `evalinevaraza63`
54. `fanhaoming`
55. `felip2342`
56. `fukdev`
57. `fulldev0418`
58. `goldenrhyno`
59. `grayce1024`
60. `guograce902`
61. `harry1988051211`
62. `harukitanaka`
63. `hector008`
64. `hector9299`
65. `hendriksenelise727`
66. `hmax`
67. `holppkgaske6i75`
68. `iandavies`
69. `ip_checknpm`
70. `jacksoneth`

71. `jahmiekstreetmanx`
72. `jaya_lubiszn`
73. `jeffbennett862`
74. `jenny-jenkins`
75. `jenis19970102`
76. `jiaopin0813`
77. `jinping0813`
78. `jinping0824`
79. `jiupaladin`
80. `joko_seti`
81. `johnasten`
82. `julianohoffmann`
83. `kaitlyndynamo`
84. `kanaan7407751`
85. `kevin_c`
86. `kevincarol`
87. `kevinyamada`
88. `kencheng1291`
89. `kingwords`
90. `kingsley19960304`
91. `kentadev0114`
92. `kik.ita`
93. `kurnia_utama4q`
94. `lauren01`
95. `leahu0604`
96. `loraine-packman09164`
97. `lucastyler`
98. `luka1291`
99. `luka1293`
100. `lukapro518`
101. `luis1214`
102. `maggie01`
103. `malarkey1992`
104. `marcsanford`
105. `math4324`
106. `meirjacob`
107. `melnikoleg`
108. `michaeldante`
109. `milton_sanders`
110. `monky1003`
111. `mykola1214`
112. `mykolakostenko`

113. `natin933`
114. `n99114`
115. `npmlover56`
116. `oliverwilson1976`
117. `ossargd324625`
118. `paerhui1102`
119. `patrickweberman`
120. `pavlo123123`
121. `perumal_balak`
122. `peter_soria525582`
123. `protonsra`
124. `quongekitti8vs6cx`
125. `riccardotala798`
126. `rodolfguerr`
127. `royalcat`
128. `royalking`
129. `royalpanda`
130. `royalpandagungfu`
131. `royaltiger`
132. `ruplles0308`
133. `ryon2080`
134. `ryon_tim`
135. `satodev`
136. `satrias`
137. `sasin`
138. `savioh`
139. `scarlet1290`
140. `scott_david`
141. `seed1996001`
142. `sean-tech`
143. `seren_quasarmzfjn49235`
144. `sergio12`
145. `setiawanet`
146. `skydev777`
147. `smartdevuser`
148. `storm0418`
149. `suhkuv.competition.tel`
150. `terralindenwhytk82974`
151. `tetiana0102`
152. `thiago_chiago`
153. `tim_blosser`
154. `timothygaffney08`

155. `trailer`
156. `trenton_alexander`
157. `uoenkpense`
158. `valerii73718`
159. `vandesaw`
160. `venjamin`
161. `venjamin1`
162. `vespero1011`
163. `victoria88`
164. `viktoria115`
165. `vinkeyasmael`
166. `vladsupernpm`
167. `vladislavkarniushka`
168. `web3chessdefi`
169. `wilder_keatingrmtuw64788`
170. `wilkinson310`
171. `william1024`
172. `winston1`
173. `wonderful123`
174. `world47`
175. `world4dev`
176. `xinrong83`
177. `yasmin9`
178. `yevheniikasymchuk`
179. `yonismith`
180. `zane29879`
181. `zhang.j`
182. `zybinantone241`

Email Addresses

1. `adammorris533@gmail[.]com`
2. `aidanphillips721@gmail[.]com`
3. `alexander0110818@outlook[.]com`
4. `alexander0110820@outlook[.]com`
5. `alexander0110828@outlook[.]com`
6. `anastasiiakoziar02@gmail[.]com`
7. `anthonysmith0979@outlook[.]com`
8. `anto[.]nost[.]athakos194@gmail[.]com`
9. `arslan310[.]kiran@gmail[.]com`
10. `astroglobal[.]work@gmail[.]com`
11. `aylin_fintech@hotmail[.]com`
12. `behrad[.]daniel@outlook[.]com`

13. `bellyache@alightmotion[.]id`
14. `bob[.]berg[.]business@gmail[.]com`
15. `briansanders0126@gmail[.]com`
16. `bryankoh64@outlook[.]com`
17. `bryanlee604@outlook[.]com`
18. `butleralvin510@outlook[.]com`
19. `carolina32123@hotmail[.]com`
20. `carolinefruet727@gmail[.]com`
21. `chain1107saw@gmail[.]com`
22. `chaparrocesaryed510727@outlook[.]com`
23. `cheekaide1992@gmail[.]com`
24. `ChinneryMarcia5425@hotmail[.]com`
25. `christrotman727@outlook[.]com`
26. `cibin87216@exitbit[.]com`
27. `ctwajstj8948@hotmail[.]com`
28. `danicaagawin5@gmail[.]com`
29. `darielfrias89@outlook[.]com`
30. `davidfernandez420@outlook[.]com`
31. `davidjambis@outlook[.]com`
32. `decovenjamin@gmail[.]com`
33. `denise[.]ward0418@outlook[.]com`
34. `desmondwynn144@gmail[.]com`
35. `devkotacorrado@googlemail[.]com`
36. `dl249995@gmail[.]com`
37. `dmytro604@outlook[.]com`
38. `dreamjobsato@gmail[.]com`
39. `dv6305655@gmail[.]com`
40. `dyanisteras15091999cuunn@hotmail[.]com`
41. `elodieblanc0707@gmail[.]com`
42. `emilylida0923@outlook[.]com`
43. `ethoszephyrtrcac76000@hotmail[.]com`
44. `EvalineVaraza63@hotmail[.]com`
45. `farrelvillarrealdngp170616@hotmail[.]com`
46. `felip2342@techspirehub[.]com`
47. `fhaoming7@gmail[.]com`
48. `galihmxf11@hotmail[.]com`
49. `garavitovillamilj@gmail[.]com`
50. `garycorn@loopsoft[.]tech`
51. `goldenrhynodev@gmail[.]com`
52. `grayce@xuchuyen[.]com`
53. `guograce902@gmail[.]com`
54. `guilddmelihb2r@hotmail[.]com`

55. `hmax23410@gmail[.]com`
56. `hectorramirez008@outlook[.]com`
57. `hendriksenelise727@gmail[.]com`
58. `hiroshi[.]watanabe1011@gmail[.]com`
59. `holppkgaske6i75@outlook[.]com`
60. `iandavies2313@gmail[.]com`
61. `jackson[.]tf7[.]eth@gmail[.]com`
62. `jahmiekstreetmanxlj126940778@hotmail[.]com`
63. `jaya[.]lubiszn@hotmail[.]com`
64. `jeffbennett862@gmail[.]com`
65. `jessikamoreira015@gmail[.]com`
66. `jh0333224@gmail[.]com`
67. `jiaopin0813@outlook[.]com`
68. `jiayingzhang[.]contact@gmail[.]com`
69. `jinping0813@outlook[.]com`
70. `jinping0824@outlook[.]com`
71. `joko[.]setiawan9l@hotmail[.]com`
72. `jokohjj80@hotmail[.]com`
73. `johnas12121@hotmail[.]com`
74. `johnbenzon510727@outlook[.]com`
75. `jiupaladin@gmail[.]com`
76. `jonatasfrnancisco887@gmail[.]com`
77. `juancastiblanco1998@gmail[.]com`
78. `julianohoffmann33@gmail[.]com`
79. `k7407751@gmail[.]com`
80. `kaitlyndynamofwtsc28771@outlook[.]com`
81. `kencheng1291@proton[.]me`
82. `kevincarol00001@gmail[.]com`
83. `kevincarol00002@gmail[.]com`
84. `kevinyamada71@gmail[.]com`
85. `kik[.]ita[.]aylen701@gmail[.]com`
86. `kingsley19960304@hotmail[.]com`
87. `korovalerii0803@gmail[.]com`
88. `kurnia[.]utama4q@hotmail[.]com`
89. `lauren[.]washco@hotmail[.]com`
90. `leahucosmin0720@gmail[.]com`
91. `leeuna@xvism[.]site`
92. `littebaby232355@gmail[.]com`
93. `lucastyler195@gmail[.]com`
94. `luis[.]fernando[.]dev1214@gmail[.]com`
95. `luka1291@outlook[.]com`
96. `luka1293@outlook[.]com`

97. malarkeyclayton5@gmail[.]com
98. marcsanford22@gmail[.]com
99. marinella@basemindway[.]com
100. matiushkodenys@gmail[.]com
101. matheuslealcardoso86@gmail[.]com
102. matheusserra0133@gmail[.]com
103. meirjacob727@gmail[.]com
104. melnikoleg995@gmail[.]com
105. melnicenkosergij119@gmail[.]com
106. michal[.]kaim99@outlook[.]com
107. milamben510@outlook[.]com
108. miltonsanders1234@gmail[.]com
109. mischenko0604@gmail[.]com
110. mykolakostenko16@gmail[.]com
111. mykolasvyryd20@gmail[.]com
112. natinbusiness[.]work@gmail[.]com
113. ninaquigleyfgsja22730@outlook[.]com
114. oka[.]setiawanet@hotmail[.]com
115. ohmlsnwz1502@hotmail[.]com
116. oliverwilson1976@hotmail[.]com
117. ossargd@xuseca[.]cloud
118. pandaroyal48@outlook[.]com
119. patterson[.]ariel@outlook[.]com
120. pattersonariel988@gmail[.]com
121. patrickweberman@outlook[.]com
122. pavlovainerman@gmail[.]com
123. peterdwtp525582@hotmail[.]com
124. perumalbalak727@outlook[.]com
125. pineye0212@outlook[.]com
126. plyn_rider@protonmail[.]com
127. proluka80518@outlook[.]com
128. quongekitti8vs6cx@hotmail[.]com
129. quintonverdantgsbxf26081@hotmail[.]com
130. ramirezhector9299@gmail[.]com
131. realonlinethiago@gmail[.]com
132. reichenausteve@gmail[.]com
133. riccardotala798@outlook[.]com
134. robertwarr1011@gmail[.]com
135. rodolfguerr717@outlook[.]com
136. royalcat3982@outlook[.]com
137. royalking066@outlook[.]com
138. royalpandagungfu06@outlook[.]com

139. `royaltiger06@outlook[.]com`
140. `runedrakesdmty71479@hotmail[.]com`
141. `ryon2080@outlook[.]com`
142. `ryon_dev_3@outlook[.]com`
143. `ryon_dev_4@outlook[.]com`
144. `ryon_dev_5@outlook[.]com`
145. `ryon_dev_6@outlook[.]com`
146. `ryonteam@outlook[.]com`
147. `sasakidev581@gmail[.]com`
148. `satriapkp91@hotmail[.]com`
149. `seed1996009@outlook[.]com`
150. `serenquasarmzfjn49235@hotmail[.]com`
151. `sergio1997121400@gmail[.]com`
152. `sean_tech208@hotmail[.]com`
153. `shubertlarvp286287@hotmail[.]com`
154. `slobodanprluv@gmail[.]com`
155. `smartinezquitian20@gmail[.]com`
156. `smarttmpacc@hotmail[.]com`
157. `soft[.]business0987@gmail[.]com`
158. `stromdev712418@gmail[.]com`
159. `suhkuv[.]competition[.]tel@gmail[.]com`
160. `tetianabanakh34@gmail[.]com`
161. `terralindenwhytk82974@outlook[.]com`
162. `timothygaffney08@gmail[.]com`
163. `top1152025@outlook[.]com`
164. `top6042025@outlook[.]com`
165. `trentonwork105@gmail[.]com`
166. `vandesaw@dewacid[.]store`
167. `venjamindeco0305@gmail[.]com`
168. `victoria88@celestiad[.]tech`
169. `vinkeyasmael@hotmail[.]com`
170. `vladkashka56@gmail[.]com`
171. `vladzane569@gmail[.]com`
172. `warfelbyeon95om0@hotmail[.]com`
173. `wilderkeatingrmtuw64788@hotmail[.]com`
174. `williammorphy37@gmail[.]com`
175. `wondereleven1@gmail[.]com`
176. `xinrong83@outlook[.]com`
177. `yevheniikasymchuk@gmail[.]com`
178. `yonismith727@outlook[.]com`
179. `yuleseraphxyvoi89853@hotmail[.]com`
180. `yusufsnz95@hotmail[.]com`

181. `yusufuyn94@hotmail[.]com`
182. `zanevlad3@gmail[.]com`
183. `zybinanton241@gmail[.]com`

Source: https://socket.dev/blog/north-korea-contagious-interview-campaign-338-malicious-npm-packages