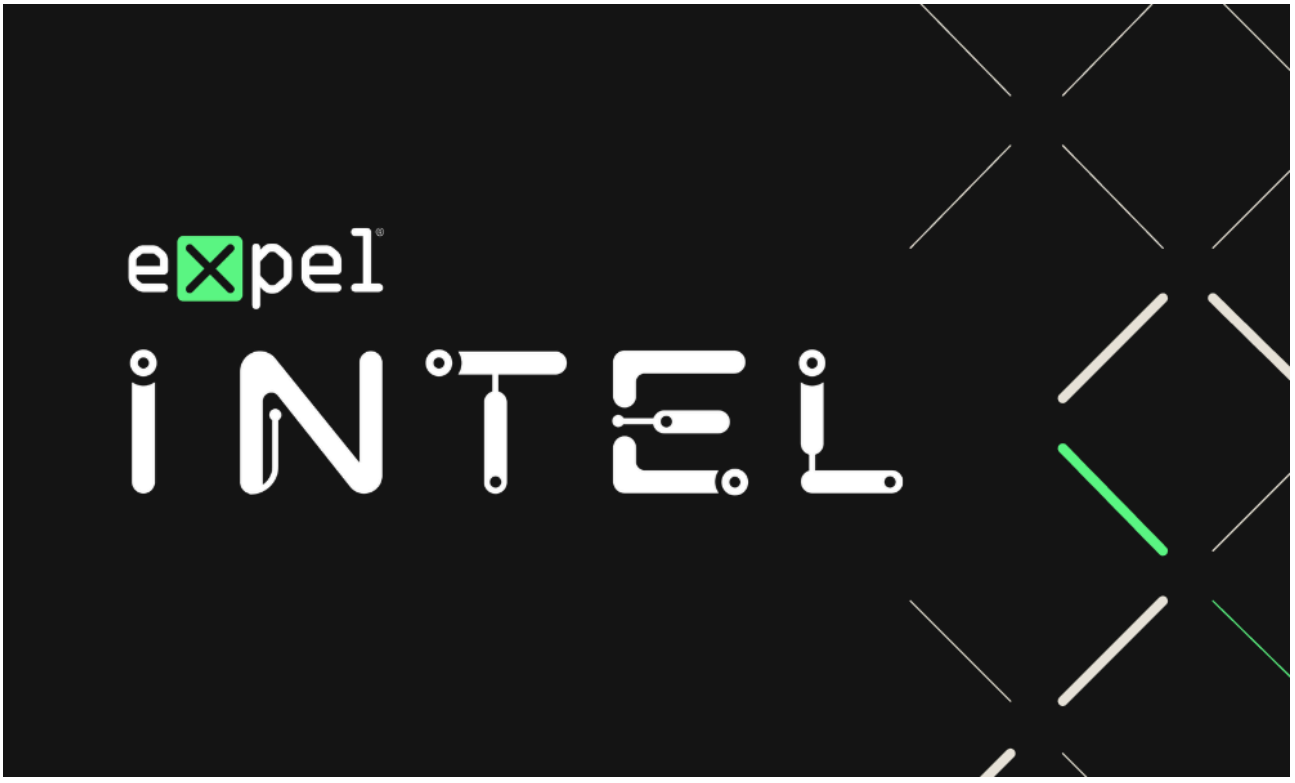
Inside Lazarus: How North Korea uses AI to industrialize attacks on developers

By Marcus Hutchins

Published: 2026-04-22 · Archived: 2026-05-01 02:14:13 UTC



This research was also highlighted in WIRED. You can read the full article from Andy Greenberg and Matt Burgess here.

TL;DR

- Expel is actively tracking an APT group that we assess with high confidence to be North Korean (DPRK) state-sponsored. We suspect that the threat actor is a subgroup or spin-off of a larger organization, potentially starting out as fraudulent IT workers before pivoting to malware.
- The group is extremely active in targeting Web3 developers and is primarily focused on stealing high-value digital assets such as cryptocurrency and NFTs.
- As much as $12M worth of cryptocurrency wallets were exfiltrated by the threat actor in 3 months, though hardware security tokens may limit damage.
- Whilst this specific group is financially motivated, many of their techniques overlap with other DPRK APTs, including those engaged in espionage.
- The group makes heavy use of Generative AI, often abusing tools like Cursor and ChatGPT.

Introducing Expel-TA-0001 (AKA HexagonalRodent)

Why yet another threat actor name?

Like many of you, we're also frustrated with the endless creation of new names for threat actors. With that said, we did find it necessary on this occasion. Vendors name threat actors based on their own internal intelligence, but the less of that intelligence that gets published, the harder it is for us to know if what we're seeing maps perfectly to activity named by another vendor.

HexagonalRodent is an active user of three pieces of malware known as BeaverTail, OtterCookie, and InvisibleFerret. BeaverTail and OtterCookie are both multi-functional malware toolkits written in NodeJS. They possess a wide array of features from password stealers to reverse shell capabilities. InvisibleFerret, on the other hand, is written in Python and acts purely as a reverse shell.

From the outside, we cannot assess the internal team structures of DPRK's military and intelligence groups. Within the threat intelligence community, we divide groups based on distinctions in motivations, techniques, and tooling. While all of these tools have been widely attributed to DPRK, they aren't exclusive to just one group within DPRK. There are multiple different variants of each piece of malware, which we believe are operated by different groups.

For the most part, all of the groups have typically been lumped together under a single name, or referred to by the malware they use.

Below is a non-comprehensive list of vendor names for groups seen using this malware:

- Famous Chollima
- PurpleBravo
- UNC5342
- DEV#POPPER
- DeceptiveDevelopment
- Contagious Interview

Without access to the internal intelligence vendors use to make their group determinations, we cannot reliably know how or if the activity we're tracking overlaps with any specific name. Additionally, we believe this group to be a subset of a larger group. Therefore, it made sense to name this specific activity cluster.

We do, however, assess with medium-high confidence that this group is a subset of what CrowdStrike refers to as Famous Chollima. Although this name encompasses both DPRK-sponsored fraudulent IT workers and several malware-based crypto theft groups, we've seen no evidence to suggest that this specific subgroup engages in fraudulent IT work.

Targeting and modus operandi

HexagonalRodent primarily targets Web3 developers with the goal of stealing crypto assets. They achieve this via social engineering developers with the promise of high-paying tech jobs, a technique that has likely become more

successful as a result of consistent mass layoffs in the industry and the resulting hiring glut. The threat actors may reach out to targets directly via platforms like LinkedIn, or publish fake job openings to popular career portals.

Once the threat actors have lured in a developer with a fake job offer, they then request that the developer undergo a coding skills assessment. For software engineering jobs, it's not unusual for companies to test developers' skills by providing them with a 'take home assessment' (a coding project that the developer must debug, add features to, or audit, and turn in for review at a later date).

HexagonalRodent's skills assessments are subtly backdoored with malware. One technique often leveraged is the tasks.json feature in VSCode (an extremely popular code editor). The config file enables developers to configure automated tasks to be run by VSCode when certain events occur. The threat actors abuse this by shipping their own tasks.json with a malicious runOn:"folderOpen"command configured. This causes VSCode to execute malware simply as a result of the target opening the source code folder with VSCode.

Additionally, the skills assessments have backdoors in the actual code, which are designed to be executed when the code is run. This serves as a primary infection vector for targets who are not using VSCode, as well as a fallback in cases where the user opens the project in safe mode, or has VSCode tasks disabled.

Difference in TTPs between HexagonalRodent and other DPRK-aligned actors

Groups like Stardust Chollima (AKA Sapphire Sleet), and Pressure Chollima (AKA JadeSleet/TraderTraitor) conduct sophisticated, highly targeted intrusions into the networks of large crypto exchanges. HexagonalRodent, in comparison, is much more opportunistic. The group may capitalize on credentials they stumble across, but tend to stick to exfiltrating crypto wallets and passwords directly from individual systems. We have not seen any evidence of attempts to move laterally within corporate networks.

Despite their relatively untargeted and unsophisticated nature, HexagonalRodent have found plenty of success in operating high-volume malware campaigns. While cryptocurrency exchanges tend to have the largest holdings, they also have the most security. On the other hand, there are plenty of small Web3 projects and crypto investors who hold significant funds, but lack the appropriate means to secure them.

A rare supply chain attack

Recently, HexagonalRodent appears to have successfully pulled off a supply chain attack. This is not something we've seen from this group before. On March 18, it was revealed that the 'fast-draft' VSX extension had been compromised and used to install malware.

The malware command-and-control server (C2) listed in the blog is 195.201.104[.]53, which belongs to the OtterCookie malware family. This specific server and OtterCookie variant is one we'd already attributed to HexagonalRodent and were tracking prior to the fast-draft attack.

While it's possible the developer was deliberately targeted, this is the first time we've seen HexagonalRodent conduct a supply chain attack. It may simply be the case that they went digging through their exfiltrated credentials for any access to repositories that they could use to help spread their malware.

We can confirm that a user matching the full name of the fast-draft extension's developer was infected by OtterCookie on March 9, 2026.

Generative AI use

HexagonalRodent makes significant use of generative AI in their campaigns. Through our telemetry, we were able to identify the use of ChatGPT and Cursor. We reached out to both the companies who operate both these products and notified them of the threat actor's abuse of their services.

Cursor's team got back to us within one business day and let us know they were investigating the accounts provided, and provided the following statement:

"Cursor was recently made aware of the issue and has blocked both the user and the IP addresses used in the attack per our Terms of Service. We are investigating further and are in communication with other model providers on the incident."

The company behind ChatGPT, OpenAI, provided the following statement:

"Based on our visibility, a small number of accounts associated with this activity sought assistance from our models on dual use cyber topics that have legitimate security, software development, and administrative use cases, but that can also be misused—e.g. password recovery and credential-security workflows, server and infrastructure security, developer troubleshooting, and crypto wallet recovery processes.

"This was limited ChatGPT usage rather than sustained or broad malware development activity. Where our safety systems detected more overtly malicious intent, our models refused or redirected those requests toward safer, dual-use responses. We did not identify any novel capabilities in these interactions.

"We'd like to thank Expel for contacting us and sharing their findings, and we encourage others to do so as well. Collaboration with external researchers and industry partners helps surface abuse faster and improve collective defense."

AI-Powered malware development

Identifying use of malware in AI development

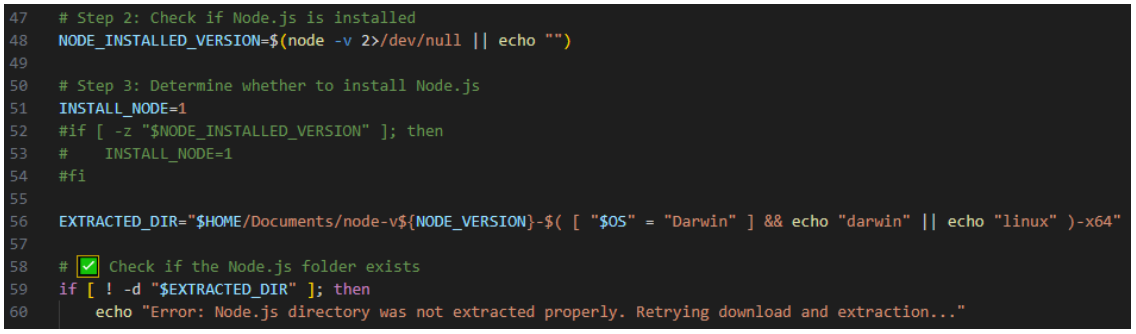
Code written using generative AI often contains verbose comments written using very formal language in perfect English. Handwritten malware, on the other hand, rarely ever has comments. When it does, they're typically very short and to the point, using slang terms and abbreviations.

One strange artifact of code generated with certain AI tools is the presence of emojis in both the code and code comments. This is highly unusual, since it's not prevalent in real world code, including the code that AI models were trained on. To type emojis on most desktop systems, developers would either need to memorize emoji keyboard shortcuts, or bring up the emoji panel and pick one by hand. These are both time-consuming practices, which interfere with most developer's desire to write code efficiently.

Evidence of AI use in threat actor tooling development

Unfortunately, both the BeaverTail and OtterCookie tooling has been around for some time, so we don't have any insights into the initial development of this code. The malware is also obfuscated with the commercial JavaScript obfuscator obfuscator.io. Obfuscation strips original code structure and comments, which makes it much harder to identify telltale signs of AI use.

However, the initial loader for the BeaverTail and OtterCookie malware did, at one point, contain artifacts consistent with AI-generated code. The comments are overly verbose, including literal step-by-step explanations of what each fragment does, as well as a slew of emojis.



```
# Step 2: Check if Node.js is installed
NODE_INSTALLED_VERSION=$(node -v 2>/dev/null || echo "")

# Step 3: Determine whether to install Node.js
INSTALL_NODE=1
#if [ -z "$NODE_INSTALLED_VERSION" ]; then
#    INSTALL_NODE=1
#fi

EXTRACTED_DIR="$HOME/Documents/node-v${NODE_VERSION}-$( [ "$OS" = "Darwin" ] && echo "darwin" || echo "linux" )-x64"

# ✅ Check if the Node.js folder exists
if [ ! -d "$EXTRACTED_DIR" ]; then
    echo "Error: Node.js directory was not extracted properly. Retrying download and extraction..."
```

A small snippet of the malware loader showing the verbose comments and emoji use.

While not a smoking gun, these kinds of artifacts are well known signs of AI-generated code, but extremely rare to see in handwritten code.

Additionally, according to a threat intelligence report published by Anthropic in 2025, several DPRK-sponsored threat actors registered accounts for Claude. The report claims that the users likely intended to, among other things, refine the BeaverTail, OtterCookie, and InvisibleFerret malware. What's notable about this specific instance, is that Anthropic states that they were aware of the threat actors prior to them registering accounts, and were able to ban them before they were able to send even a single prompt.

More conclusively, we did find two new in-development tools being tested in the wild. Both of these tools are entirely 'vibe coded', which we were able to confirm as a result of the threat actor accidentally leaking some of the prompts used to generate the malware. Additionally, the code wasn't obfuscated, revealing extensive debug code littered with emojis.

```
// Handle key:forward events for all active clients
(0,react__WEBPACK_IMPORTED_MODULE_0__.useEffect)(() => {
  if (!socket || !socket.connected) return;
  const handleKeyForward = payload => {
    console.log('🔍🔍🔍 handleKeyForward called with payload:', payload);
    console.log('🔍 Payload keys:', payload ? Object.keys(payload) : 'null');
    console.log('🔍 Payload type:', payload === null || payload === void 0 ? void 0 : payload.ty
    console.log('🔍 Payload clientId:', payload === null || payload === void 0 ? void 0 : payloa
    console.log('🔍 Payload copiedText:', payload === null || payload === void 0 ? void 0 : payl
    console.log('🔍 Payload clipboardText:', payload === null || payload === void 0 ? void 0 : p
    const clientId = (payload === null || payload === void 0 ? void 0 : payload.clientId) || (pay

    // Check if this is a clipboard event (type: 'clipboard' OR has clipboard data)
    if (payload.type === 'clipboard' || payload.copiedText || payload.clipboardText) {
      const copiedText = payload.clipboardText || payload.copiedText || payload.text;
      console.log('✅✅✅ Clipboard event detected in key:forward for client:', clientId, 'text
      if (!clientId) {
        console.error('❌ No clientId in clipboard event');
        return;
      }
      if (!copiedText || !copiedText.trim()) {
        console.warn('⚠️ No clipboard text in clipboard event, copiedText:', copiedText);
        return;
      }
      console.log('✅✅✅ Processing clipboard event, copiedText length:', copiedText.length);
```

A snippet of one tool (a web-based keylogger panel) which contains egregious emoji use.

We also saw evidence of several of the threat actors prompting various US-owned AI models to audit their skills assessments' code for malware. We believe this was likely part of an attempt to AI-proof their backdoors.

Previously, several of the threat actor's campaigns had been burned as a result of their targets using AI to audit the skills assessment's source code. Frontier AI models could often find the backdoors with ease, resulting in several targets publicly outing the threat actor's personas.

Elaborate phishing schemes

HexagonalRodent's abuse of front companies and fake websites

Although the threat actors appear to have a strong preference for reaching out to targets directly (posing as tech recruiters), we have seen some far more elaborate schemes. In several cases, the group set up fake company websites, along with associated LinkedIn accounts and employee profiles. In one case, they even registered a corporate entity in Mexico.

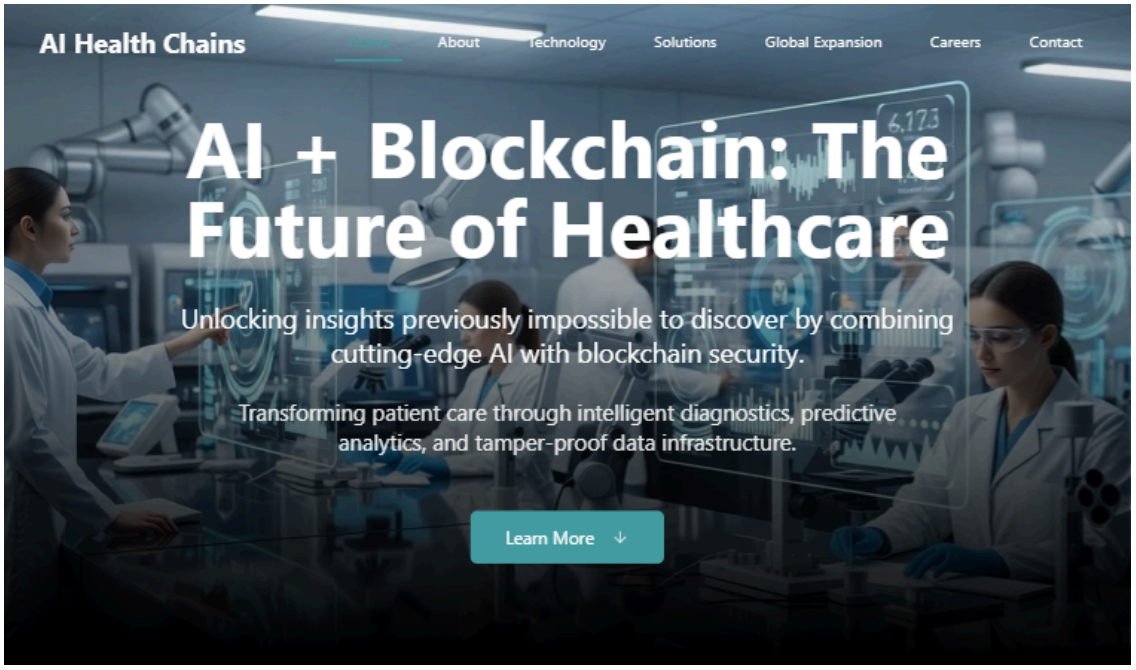
Through their fake companies, the threat actors would list job openings on a multitude of Web3-focused career platforms. The job postings would direct applicants to submit their resumes via the fake company website. Following receipt of a job application, one of the group's members would then reach out to the applicant via an official company email and ask them to complete a skill assessment.

We suspect that these campaigns likely have a much higher success rate compared to cold calling targets via LinkedIn. However, it took significant time and resources to establish these front companies, which could be burned by even a single applicant realizing they've been targeted with malware.

Generative AI use in the construction of front companies

Throughout the process of setting up front companies, the threat actors make heavy use of generative AI. Some of the more elaborate operations create entire fake leadership teams. Each persona would get its own LinkedIn account. The fake C-suites would be listed on the company's LinkedIn page, and often its website too. Many of the headshots used were AI-generated, but in some cases they'd steal photos of real people from social media posts.

The websites themselves are also built with AI. One such example is aihealthchains[.]com. Although we were unable to confirm that this specific site belongs to Expel-TA-0001, it was involved in the distribution of BeaverTail and OtterCookie malware. We personally reverse engineered several malicious skills-assessments published by the company's GitHub account and confirmed this. However, the GitHub account has since been taken down.



The homepage of AI Health Chains, which features an AI-generated video of lab workers as the website background.

When we inspected the source code of the website, we could see that all the assets are hosted via c.animaapp.com. This url is the default CDN used by websites built with Anima, an online AI-powered website design and development platform.

```
<section id="mission" class="py-24 md:py-32 px-4 sm:px-6 lg:px-8 bg-t
  <div class="container mx-auto max-w-7xl text-foreground">
    <div class="grid md:grid-cols-2 gap-16 items-center">  grid
      <div style="opacity: 1; transform: none;">…</div>
      <div class="relative" style="opacity: 1; transform: none;">
        <img src="https://c.animaapp.com/mhaqco34WrlaWG/img/ai_2.png"
      </div>
    </div>
  </div>
```

> On this specific website, none of the assets are hosted locally, they're all downloaded from the animaapp CDN.

While we do not believe that Anima is at any fault, as it would have been impossible to know whether such a website was being designed for a legitimate web3 project or a front company, the Anima CEO was very proactive in reaching out and working with us to investigate the threat actor's use of their product. The abuse of AI web design tools in this way was also not limited to Anima, simply, this was the only website active at the time of publication.

Another example, which is no longer online, was codepointlab[.]com. We were able to tie this campaign back to Expel-TA-0001 because the threat actors hosted their malware control infrastructure on the same server as the company website. While one could argue that this may be a legitimate website that the actors simply compromised, it is hosted via one of the bulletproof hosting providers they use for their malware infrastructure.

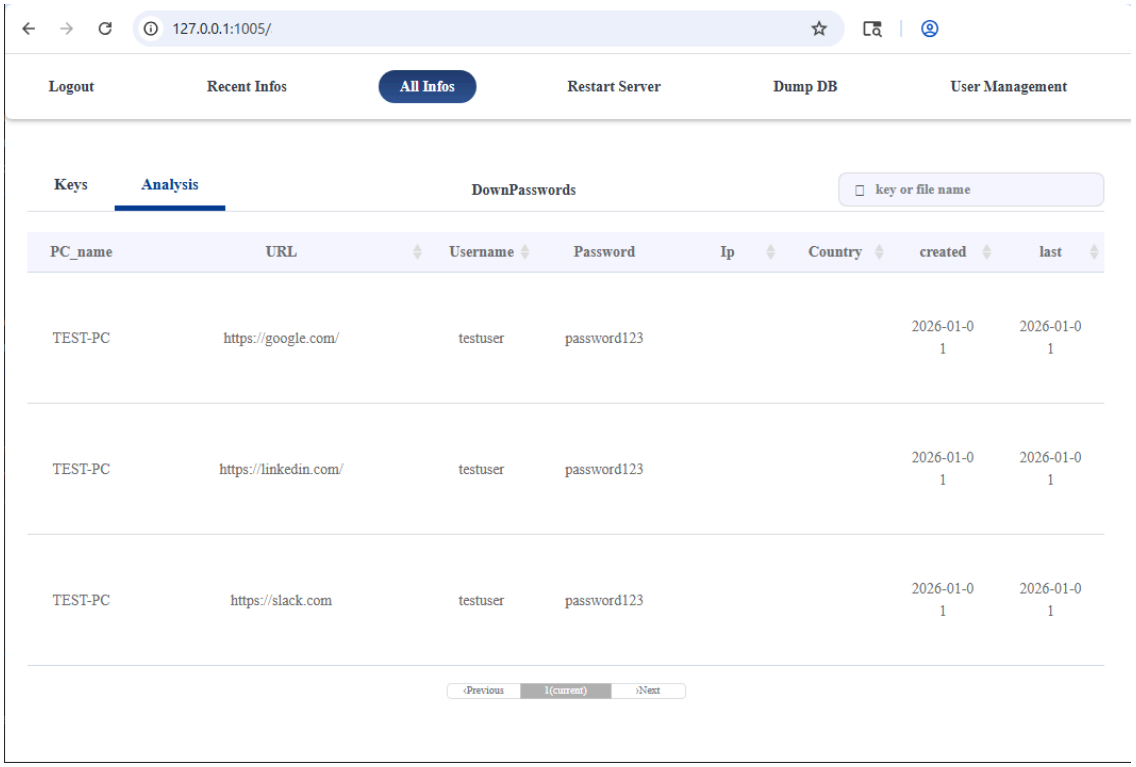
How a single thread lead to us uncovering a vast hacking operation

In October 2025, following an incident involving a BeaverTail malware infection on a customer network, we began investigating the threat actor's command-and-control (C2) infrastructure. During our research, we stumbled across a trove of C2 panels. The panels, which were written in ReactJS, provided several key insights into the inner workings of the group.

ReactJS web applications, or WebApps, consist of two parts: a frontend and a backend. The frontend code is what gets loaded into the user's web browser when they visit the website. Meanwhile, the backend code runs on the web server and does most of the data processing. The frontend and backend talk to each other via an API.

Since the design and layout of the WebApp is embedded in the frontend code, we don't need to be able to log into the threat actor's server to see what the user interface looks like. Instead, we reverse engineered the frontend code, then built our own backend to recreate the panel with synthetic data. This allows us to show you what it looks like.

The first WebApp, which we found on an exposed FTP server, is a prototypical infostealer panel. It allows the threat actors to search, view, and download credentials stolen from victim systems by their malware.



The infostealer panel displaying our synthetic data from a fake system named ‘TEST-PC’.

Based on analysis of the panel’s communication protocol, we believe this is for the BeaverTail malware, specifically its infostealer component. The malware has the ability to exfiltrate credentials from most web browsers’ built-in password manager, the macOS Keychain, Linux’s Keyring, and 1Password.

Pivoting off the threat actor’s infrastructure

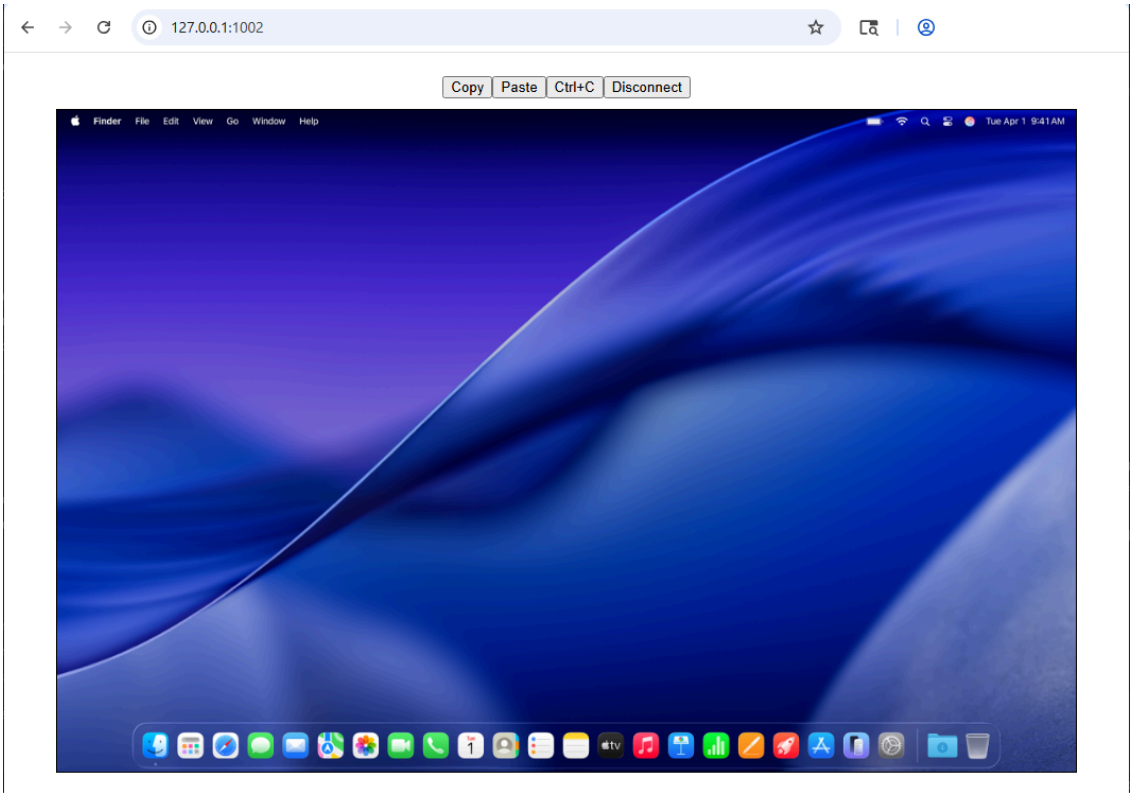
Once we’d gained an understanding of how the threat actors operate, we built custom tooling to identify more of their infrastructure. We spent several weeks cataloging servers related to BeaverTail, OtterCookie, and InvisibleFerret. With this information, we were able to derive many high-confidence detections to help strengthen our customer’s security.

Via reverse engineering and careful analysis, we then identified which systems were involved in campaigns specific to HexagonalRodent, and excluded those belonging to other DPRK-sponsored activity clusters. While all the servers we found proved valuable for detection engineering, they weren’t all relevant to this investigation.

During our triaging of DPRK-associated malware infrastructure, we found several new panels. However, they contained different backend server IPs embedded within the code, and at first appeared to have no obvious links to HexagonalRodent.

A whole host of new panels is uncovered

The next panel we found was a browser-based remote control utility. It provided the threat actor with VNC-like capabilities (the ability to view the victim’s screen, as well as control their keyboard, mouse, and clipboard). All of this could be done directly from the attacker’s web browser.

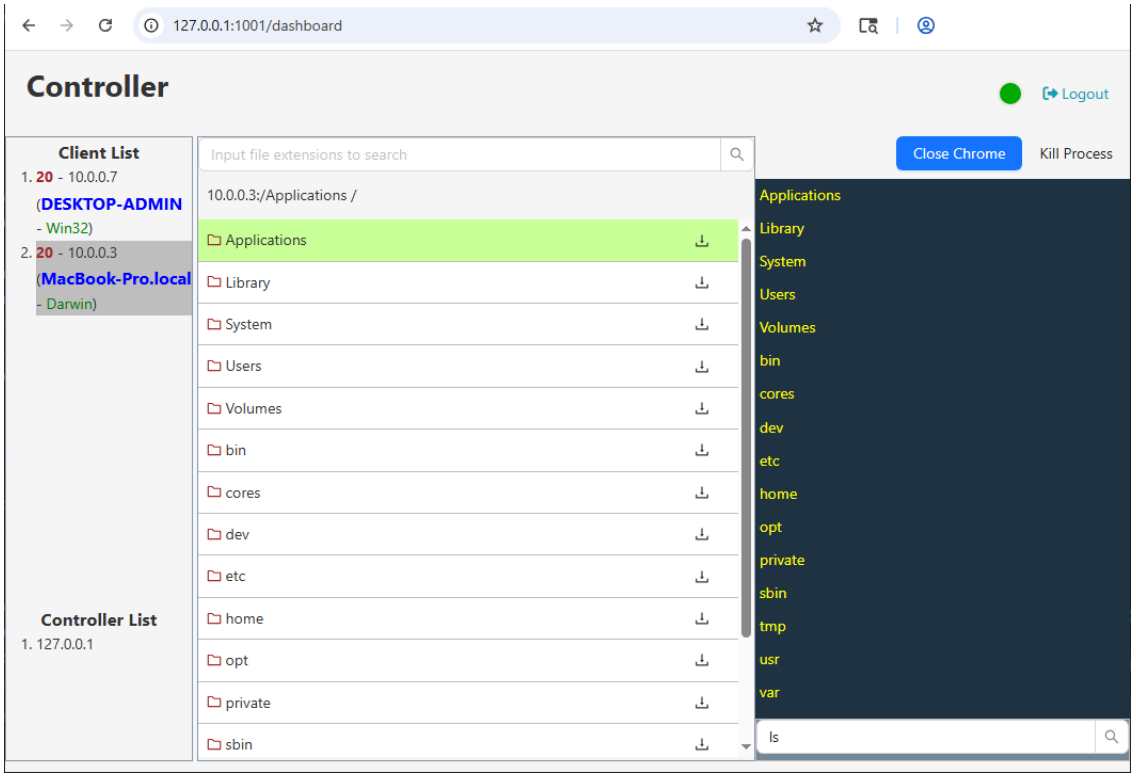


The remote control panel, presenting a stock image of a desktop for illustrative purposes.

The third panel we found was even more interesting. It had two separate components. The first, a remote file manager, allowed the threat actors to browse through the victim's filesystem. This feature—again, entirely browser-based—felt completely seamless.

The panel communicates with the malware on the victim's system in real time. Clicking into a directory sends a command to the victim's system to list said directory, then the results are sent back to the WebApp. The panel updates the displayed directories and files dynamically, so it feels almost like you're browsing files on your own computer.

The second feature is a 'reverse shell' controller. It provided the threat actor with the ability to run arbitrary commands on the victim's system. This is facilitated via cmd.exe on Windows, sh on Linux, and zsh on Mac. The results are then dynamically displayed on the righthand side of the panel.



The file explorer/reverse shell panel interacting with our test system. To simulate the panel, we built a backend server from scratch by reverse engineering the WebApp's communication protocol.

In the left-most column is the 'Client List' and 'Controller List' segment. The Client List displays currently connected victim machines, showing their IP address, hostname, and operating system. The target of the file explorer window and reverse shell is set by simply clicking on one of the systems in the list.

The 'Controller List', on the other hand, seems to just list the IP addresses of anyone currently logged into the panel (i.e., the threat actors). It doesn't seem to serve any purpose other than just showing who's currently connected.

Both the file explorer and reverse shell functionality is implemented via WebSocket, a protocol that enables realtime streaming communication between a website and WebSocket server. Both the threat actor's web browser and the malware on the victim's system maintain persistent connections to a custom WebSocket server. This server is responsible for relaying commands between the threat actor's panel and the victim system.

Essentially, the panel facilitates real-time communication between the threat actors and the victim's system, rather than relying on having to constantly send web requests to poll for updates. This does, however, mean that the malware can be easily identified by checking for NodeJS processes which maintain a persistent TCP connection to one of the threat actor's servers.

Below are some example commands to check if a system is connected to the C2 server used in the fast-draft supply-chain breach:

MacOS/Linux: netstat -an | grep 195.201.104.53

Windows: netstat -an | findstr 195.201.104.53

Know your animals: Beavers vs. otters

While reversing the file explorer panel's communication protocol, we realized something. It doesn't match up with the protocol used by the BeaverTail malware. It does, however, match the protocol used by OtterCookie.

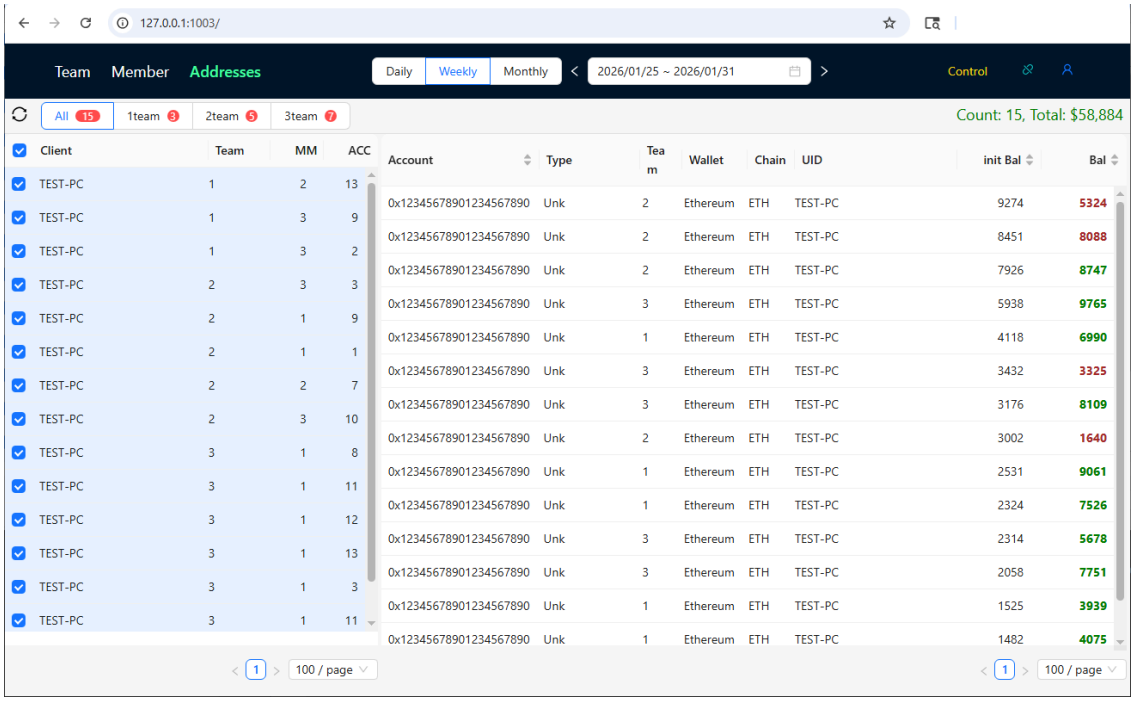
Previously, we'd seen a malicious skills assessment that was backdoored with both BeaverTail and OtterCookie at the same time. This seemed unusual, since BeaverTail and OtterCookie are both pieces of NodeJS malware and have an almost 100% overlap in features. Why deploy two near identical pieces of malware?

Initially, we'd chalked it up to an operator error. We'd been assuming that different teams were sharing skills assessment templates with each other, and one had forgotten to remove their backdoor before passing it over to another team.

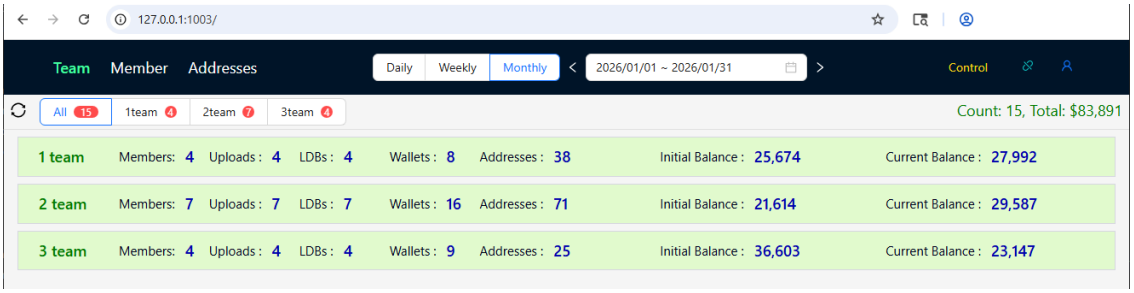
However, after obtaining credible intelligence from a third-party source, we confirmed with high certainty that this panel, along with others on the server, are utilized by HexagonalRodent. This still doesn't answer the question as to why they're co-deploying two extremely similar pieces of malware, but it could be for redundancy in case one infection gets detected.

Our first real insight into the group's inner workings

The final panel we found and reverse engineered was by far the most fascinating, but also equal parts confusing. It lists cryptocurrency wallets, along with their balance, but doesn't appear to contain any features to do anything with them.



The panel's 'Addresses' page, filled with our backend's placeholder addresses.

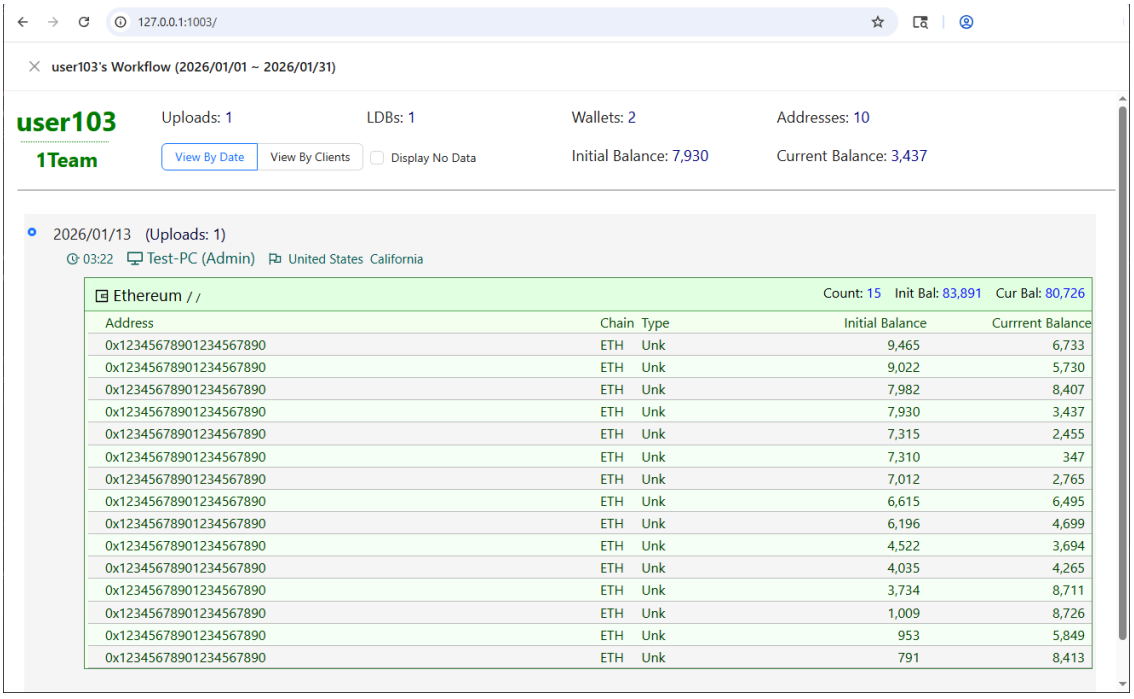


The panel ‘Teams’ page (populated with synthetic data).

The panel’s hardcoded hierarchy provides some unique insight. There is a ‘Members’ and a ‘Teams’ page. The main page subdivides data by team, and then again by member. It implies that HexagonalRodent consists of multiple teams, each composed of multiple members.

The OtterCookie malware provides further evidence of this. Each sample contains two hardcoded values: ‘t‘ and ‘userKey’. The former maps to a value in the front end code which is also named ‘t’. This value is then used to build the “Teams” list. The second value, userKey, is not directly referenced in the code, but we believe it likely maps to a Member ID.

We’ve also been tracking malicious skills assessments in the wild, cataloging which t and userKey values they contain. The variations and volumes of these campaigns supports our theory that HexagonalRodent consists of multiple teams, each composed of several members.

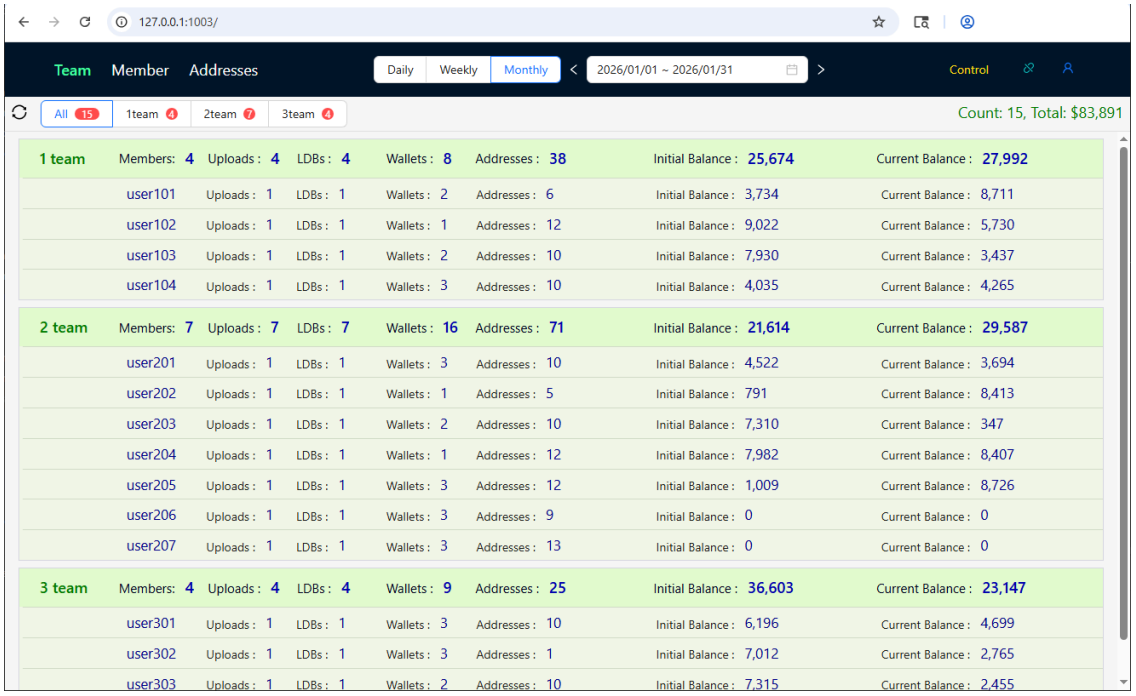


The panel’s ‘workflow’ page.

The frontend code also has a page presenting a ‘Workflow’ for each ‘Member’. The workflow page lists cryptocurrency addresses grouped by system hostname (likely the specific system they were exfiltrated from). If <userKey is indeed the member ID, then this page would list all the cryptocurrency wallets exfiltrated by a specific team member.

What's notable is the fact panel does not appear to contain any pages to display private keys, nor features for interacting with the wallets outside of just simply viewing their 'initial' and 'current' balance. We believe the 'initial' balance pertains to either the balance of the wallet when it was exfiltrated from the victim system, or when it was first ingested into this panel.

Additionally, part of each user's username is hardcoded. The panel's code removes the word team from the user's username, then sets a variable named userTeam to whatever is left. This means that the username can only contain the word team and the team number. In essence, there can only be one username per team.



The main page as viewed by a user with admin privileges.

Also embedded in the panel's code is a function checking if the current user's name is "admin". If so, then the user is granted the ability to view all members across all teams. Regular users are limited to viewing only members within their team.

The read-only nature of the panel, limitation of one account per team, and ranking-system like interface leads us to believe that this is a workforce tracker rather than a command-and-control server. This is also consistent with other reports pertaining to DPRK's fraudulent IT worker operations, where threat intelligence companies have discovered the actor's timesheet and daily status update tracking systems.

The panel seems intended for team leaders to view their team member's performance, as well as a global manager to view the performance of all members and all teams.

The final score

At one point during the course of our investigation, a misconfiguration exposed the workflow tracker's backend database, allowing us to obtain the raw data from it. With this data, we were able to gain a much deeper insight into the threat actor's operations.

The data contained a t and id value for each wallet entry. These match up with the t and userKey variables we've observed across the threat actor's malware campaigns. With this, we can infer HexagonalRodent's internal structure.

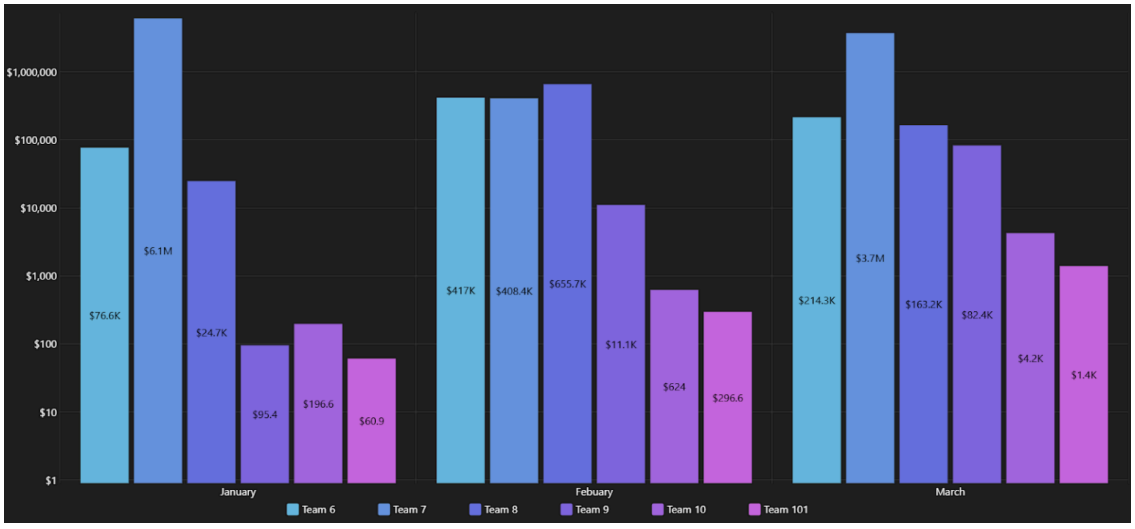
There are 31 unique campaign IDs and 6 unique team numbers. The teams are named 6team, 7team, 8team, 9team, 10team, and 101team. While we're not certain there is a 1:1 map between campaign IDs and team members, if there is, that would imply that HexagonalRodent consists of 31 operators split across 6 teams.

Curiously, there is a hardcoded reference to a 'team 4′ in the panel's source code, but we found no data relating to a team 4 in either the workflow data or any of the HexagonalRodent campaigns we analyzed. We did, however, confirm the existence of a Team 2 and Team 3, though both teams operate their own separate infrastructure entirely disconnected from any belonging to HexagonalRodent.

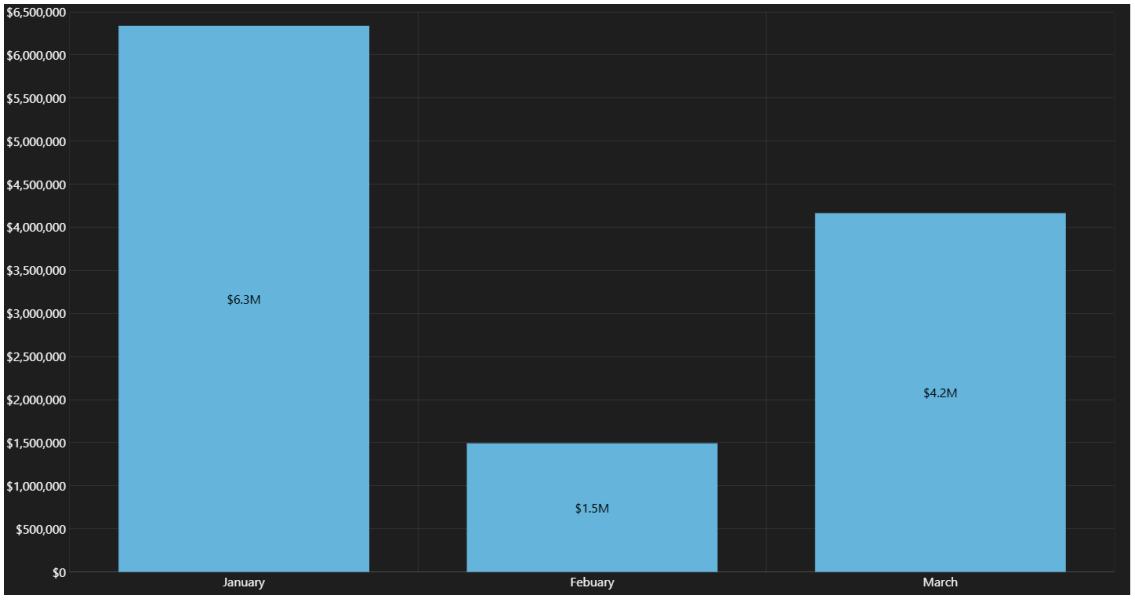
Our leading theory is that some of the earlier teams may have spun off and started their own independent operations. One of these teams, we believe, is behind a more sophisticated version of BeaverTail, which stores payloads via the Blockchain (a technique known as EtherHiding).

From victim IP addresses and system hostnames contained within the data, we are able to deduce that the threat actor's campaigns exfiltrated a total of **26,584** cryptocurrency wallets from **2,726** infected developer's systems.

Below is a graph of the total USD value of wallets ingested into the threat actor's system between January 1 and March 31 of 2026.



Total value of all wallets ingested each month broken down by team.



Total value of all wallets ingested monthly by all teams.

In total, it appears that public keys for wallets holding a total of up to $12 million dollars worth of crypto assets were exfiltrated from victim systems in the first 3 months of 2026.

Limitations in assessing actual losses

While we did attempt to evaluate the total value of assets successfully stolen, this task proved essentially impossible. The exfiltrated wallets held thousands of tokens across a multitude of different blockchains, which made tracking transactions extremely difficult. Furthermore, for most transactions, the threat actors used entirely new wallets.

Additionally, since the workflow tracker does not contain any private keys, we're unable to confirm which wallets the threat actors were successfully able to exfiltrate full, unencrypted, private keys from. The data lists many of the wallets as being protected by hardware tokens, which means that even with persistent access to the owner's system, the threat actors would be unlikely able to drain those wallets.

We were able to confirm that the funds from at least 13 of the wallets eventually made their way to a known DPRK-operated Ethereum address. The wallet, which was created in 2023, has received over $1.1 million in funds, with around $500k of that in a single month. Though it's unknown if this wallet is specific to this campaign.

Conclusion and after thought

It's important to note how successful HexagonalRodent's campaigns have been in terms of the number of infected systems, despite the lack of sophisticated tooling. Their AI-generated malware is neither novel nor particularly evasive, but it is effective.

A self-refutation

Admittedly, this article refutes a theory of mine; well at least partially, at least. I had proposed that generative AI is unlikely to significantly lower the bar for malware development. This was predicated on the fundamentals of how LLMs work. They predict likely text based on patterns in their training data.

Since writing evasive malware typically requires coming up with novel techniques, LLMs should, in theory, do the opposite. The more widely documented a malware technique is, the more prevalent it is likely to be in an LLM's training data. But the more documented a malware technique, the more detected it is going to be, so LLMs should skew towards writing highly detectable malware. Unless, of course, they are being guided by a skilled malware developer who is familiar with which techniques to avoid.

However, HexagonalRodent makes for an important edge case. Unlike most modern financially motivated threat actors, they target individuals rather than corporate networks. Thus, they don't have to contend with high end cybersecurity products such as EDRs and NDRs. In fact, most personal computers don't run any antimalware products at all. Which makes AI-generated malware's lack of evasiveness somewhat irrelevant here.

Vibe coded malware vs. high-end EDR product

What came as a shock to us is what happened in the few cases we encountered where targets opened the malicious skills assessments on their work machine. Despite the customers' networks running flagship EDR products, the malware did not immediately trigger any detections. Alerts came much later, after the EDR vendors deployed rules to flag connections to the attacker's C2 IP addresses. This was what motivated us to build our own in-house detection.

We theorize that the lack of detections may be due to the accidental novelty of HexagonalRodent's malware. The programming languages they chose to develop their malware in are NodeJS and Python. Neither language is particularly common for writing malware, since neither the Python or NodeJS interpreter is installed by default on the average operating system.

Because the threat actors target software developers, and often those who develop applications in NodeJS and Python, they can afford to use those languages for their malware. Not only is the presence of both interpreters on a developer's system more likely, it's highly inconspicuous. Essentially, the malware inadvertently blends in with normal developer activity. NodeJS code doing things on a NodeJS developer's system is expected behavior.

Additionally, the group makes use of the commercial JavaScript obfuscator obfuscator.io, which is used by legitimate developers to protect their source code from reverse engineering and/or theft. This makes it extremely difficult to write antimalware signatures for, since the obfuscated JS malware just looks like any other JS obfuscated code.

The combination of NodeJS malware naturally blending in with developer behavior, and the use of languages for which security products appear not to have good visibility into, HexagonalRodent appears to have stumbled into creating a perfect storm of low-effort/high-impact malware. They've essentially inadvertently built a living-off-the-land attack which weaponizes software developer's tooling against them.

HexagonalRodent appears to have stumbled into creating a perfect storm of low-effort/high-impact malware.

This might only be the beginning

For the past 4 years, the tech industry has been flooded with mass-layoff after mass-layoff. This has likely heavily impacted DPRK's fraudulent IT worker scheme, forcing them to reallocate resources towards other means of generating revenue. With so many software engineers out of work, and so few job opportunities available, it makes it all the more easier for North Korean state-sponsored hackers to ensnare targets. With developers applying to hundreds or thousands of jobs without receiving a call back, they're likely to have their guard down when that one job offer finally comes in.

In addition, DPRK is known to exploit foreign workers to enable their operations. With IT worker schemes, they'd often need people to take delivery of company-issue mobile devices. Since many of these devices had GPS trackers, they could not simply ship them out of the country. Instead, they'd recruit people within the country to host the laptops from their houses. To the companies, it'd appear that the laptop was being operated from within the country. In reality, it was being remotely controlled by a North Korean operative using tools like IP KVMs.

Some of these operations were rather elaborate, with participants hosting 10s of laptops at a time, all from their personal residence. Since payment amounts were often relative to how many operations the subject helped facilitate, many were willing to look the other way and not ask questions if it meant a stable paycheck.

As economic uncertainty increases, so do the number of people willing to turn a blind eye to questionable remote jobs that help facilitate foreign adversaries. This opens the door for many opportunities that go beyond fraudulent IT work and cryptocurrency stealing malware.

Whilst financially motivated cybercrime is highly unappealing to almost every nation-state, since the monetary loss from the resulting sanctions would far outweigh any financial gain, this is not the case for North Korea. The heavy sanctions already levied against the country mean there is little more that can be done to deter them, but a lot to be gained for a nation whose economic activity is severely constrained. It's estimated that revenue from cybercrime makes up between 3 and 7 percent of North Korea's GDP, which is not insignificant.
