Beyond the Pond Phish: Unraveling Lazarus Group's Evolving Tactics - BitMEX Blog

By BitMEX

Published: 2025-05-30 · Archived: 2026-04-05 18:11:37 UTC

The Lazarus Group is a prominent hacking group associated with the North Korean government with a long history of targeting companies and individuals within the cryptocurrency space. They have been linked to the breaches of Phemex, WazirX, Bybit, Stake, among others.

Our security team frequently responds to attempts to attack us, many of which use techniques or infrastructure that have been tied to the Lazarus Group by other researchers.

A common pattern in their major operations is the use of relatively unsophisticated methods, often starting with phishing, to gain a foothold in their target's systems.

For example, in the Bybit breach, the group tricked a Safe Wallet employee into running malicious code on their computer to establish initial access. Once this foothold was obtained, what looks like a more sophisticated "division" of the group took over and continued post-exploitation, obtaining access to Safe's AWS account and modifying the wallet's front-end source code, which resulted in the ultimate theft of their cold wallets.

Throughout the last few years, it appears that the group has divided into multiple subgroups that are not necessarily of the same technical sophistication. This can be observed through the many documented examples of bad practices coming from these "frontline" groups that execute social engineering attacks when compared to the more sophisticated post-exploitation techniques applied in some of these known hacks.

Recently, a BitMEX employee was contacted through LinkedIn for a potential "NFT Marketplace" web3 project collaboration. This pretext was similar enough to other attacks common in this industry that the employee suspected it was an attempt to trick them into running malicious code on their device. They alerted the security team, who investigated with the objective of understanding how this campaign worked and how to protect ourselves from it.

The interaction is pretty much known if you are familiar with Lazarus' tactics. After some back and forth with the attacker, our employee was invited to a private GitHub repository which contained code for a Next.js/React website. The goal was to make the victim run the project, which includes malicious code, on their computer. After a few minutes of inspection of the repository (just grepping for "eval", really), we found some very suspicious pieces of code:


carbon-10

The first instance of calls to the eval function was commented out, suggesting this code was used in a previous campaign or was an older version of the malicious code being distributed. If it was not commented out, it would

send a HTTP request to*hxxp[://]regioncheck[.]net/api/user/thirdcookie/v3/726*and execute the response's "cookie" value. This domain has been previously attributed to the Lazarus Group by Palo Alto's Unit 42.


carbon-4

The second eval call we found was not commented out. The code here sends a HTTP request to hxxp[://]fashdefi[.]store:6168/defy/v5 and executes the JavaScript code returned by the server.

We then sent this request out manually and saved its response for further analysis. The JavaScript code returned by the server was obfuscated, making it hard to analyse at a glance.


image14

To understand what this is really doing, we used webcrack, a JavaScript deobfuscation tool, which yields a slightly better "unminified" version.

This javascript file looked like a result of joining three different scripts together. We can see multiple code blocks that separate the different stages of the malware.


image5-1

At first glance, the second part of the script contained strings that were similar to what we would expect from a credential stealer: references to Chrome extension IDs and to other Browsers.


image12

This "p.zi" string looked familiar to us as well, even without deobfuscating the code - it is similar to other pieces of malware that have been previously tied to the DPRK and resembles the "BeaverTail" campaign, originally described by Palo Alto's Unit 42 in this report. Since Unit 42 has already extensively analysed this second component, we will not cover it here.


image3-3

After getting confirmation of who we were dealing with, we decided to continue deobfuscating the code in an attempt to dig some IoCs that could be added to our internal tools.

JavaScript deobfuscation is pretty fun once you know the patterns the obfuscation tools use, and usually boils down to finding and replacing references to array strings or calls to "decryption" functions and renaming variables. Starting from the first code block, we manually replaced all of the references to a string array with their corresponding values and used webcrack's symbol renaming tool to rename variables based on their context, which results in human-readable code:


carbon-7

As this was not our first time reverse engineering malware related to this kind of campaign, we were already reasonably familiar with the code. However, this initial part of the file was new to us: it connects to a Supabase instance and writes metadata (username, hostname, os, ip, geolocation, time) about the computer that has been infected.

Supabase is a free managed database service akin to Google's Firebase. It allows developers to quickly set up databases that have easy-to-use interfaces for applications which, if configured properly, allow you to implement almost all functionality that would usually be tied to an API layer (such as authentication, access control, etc) without the need for one.

A common issue with these services is that developers do not take the time to configure permissions properly and end up leaving significant parts of the database accessible to anyone. With this in mind, it was one of the first things we decided to test using this simple script:

To our surprise, at the time, this returned 37 records with data from computers that had previously been compromised.

If we take a closer look at the data, some logs stand out: a lot of username/hostname combinations are repeated, and some of those have patterns that look like test runs, potentially done by developers. We also see a pattern with many hostnames of the form of 3-XXX.

The IP addresses logged for these entries mostly belong to VPN providers. One of the recurring usernames, "Victor", consistently uses IP addresses that appear to be managed by Touch VPN, while "GHOST72" uses IP addresses that map to Astrill VPN servers (source: spur.us).

By looking at the logs for "Victor", we found an entry that stands out: the IP address and location do not match the previously observed Touch VPN exit nodes, but rather a residential China Mobile IP address (223.104.144.97) located in Jiaxing, China. We believe that this was an operational security mistake, which ended up leaking the attacker's original IP address.

Once we had this information, we created a simple program that would query this database on a regular basis and log new infections with the goal of understanding the general profile of victims and potentially spotting new mistakes by the operators. This program has been running since May 14, 2025 and our data has all logs dating back to March 31st. So far, this amounts to 856 entries with 174 unique user/hostname combinations.

Unique new infections by day (UTC):

By looking at the username, hostname and IPs of past infections, we were also able to identify other computers and accounts used to test or develop the malware used in this campaign:

- Victor@3-KZH (12x Touch VPN, 2x China Mobile, 1x Unknown/US)
- Victor@3-KZH-1 (9x Touch VPN)

- GHOST72@3-UJS-2 (3x Astrill VPN, 1x Zoog VPN)
- ghost@GHOST-3 (3x Astrill VPN, 1x Hotspot VPN)
- Super@3-AHR-2 (1x Astrill VPN, 1x Touch VPN)
- Admin@3-HIJ (2x Astrill VPN)
- Lenovo@3-RKS (1x Astrill VPN)
- firebird@3-KJH (1x Touch VPN)
- degen@Alli (1x Astrill VPN)
- GoldRock@DESKTOP-N4VEL23 (1x Astrill VPN)
- Muddy@DESKTOP-MK87CBC (1x Astrill VPN)

With these hostnames in mind, we can also plot a chart that shows active hours for the operators behind this campaign:

Interestingly, we identified a consistent period of downtime for the operators from ~8am to ~1pm UTC (5pm to 10pm Pyongyang time), which suggests that they do have a structured schedule or consistent “working hours”, with activity occurring throughout the rest of the 24-hour cycle.

Conclusion

Investigating this Lazarus Group campaign shows a stark contrast between their entry-level phishing strategies and advanced post-exploitation techniques. The accidental exposure of the Supabase database revealed not only their tracking methods but also significant lapses in operational security, such as the leakage of Chinese IP addresses, offering interesting insights about the inner workings of the group.

Contact

If you want to get in touch with us regarding this topic, or the idea of working in an organisation that investigates these kinds of attacks interests you, contactsecurity-research atbitmex dot com.

Indicators Of Compromise (IoCs)

Supabase URL	https://mkswbddldpyiqkyu.supabase.co/
C2 URL	http://144.172.96.35/

Threat Actor: Victor@3-KZH	107.182.231.193, 107.182.231.196, 120.226.22.28, 184.174.5.149, 223.104.144.97, 31.13.189.10, 31.13.189.178, 37.120.216.226, 38.134.148.94, 45.141.153.154, 89.116.158.156, 89.116.158.164, 89.116.158.188, 89.116.158.228, 89.116.158.68
Threat Actor: Victor@3-KZH-1	107.182.231.196, 31.13.189.10, 31.13.189.26, 38.132.106.130, 45.141.153.130, 89.116.158.156, 89.116.158.228, 89.116.158.68, 89.116.158.84
Threat Actor: GHOST72@3-UJS-2	108.181.57.127, 195.146.5.31, 199.168.113.31, 89.187.185.11
Threat Actor: ghost@GHOST-3	129.232.193.253, 195.146.5.31, 209.127.117.234, 45.56.197.79
Threat Actor: GoldRock@DESKTOP-N4VEL23	38.170.181.10
Threat Actor: Lenovo@3-RKS	38.170.181.10
Threat Actor: Super@3-AHR-2	217.138.198.34, 89.187.161.220
Threat Actor: degen@Alli	167.88.61.148
Threat Actor: firebird@3-KJH	146.70.63.2

Source: https://blog.bitmex.com/bitmex-busts-lazarus-group/