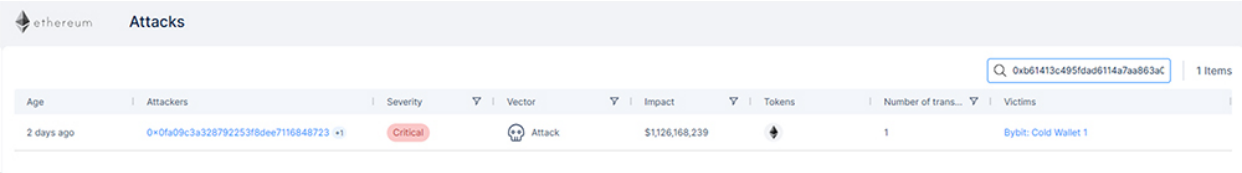
research.checkpoint.com /2025/the-bybit-incident-when-research-meets-reality/

The Bybit Incident: When Research Meets Reality

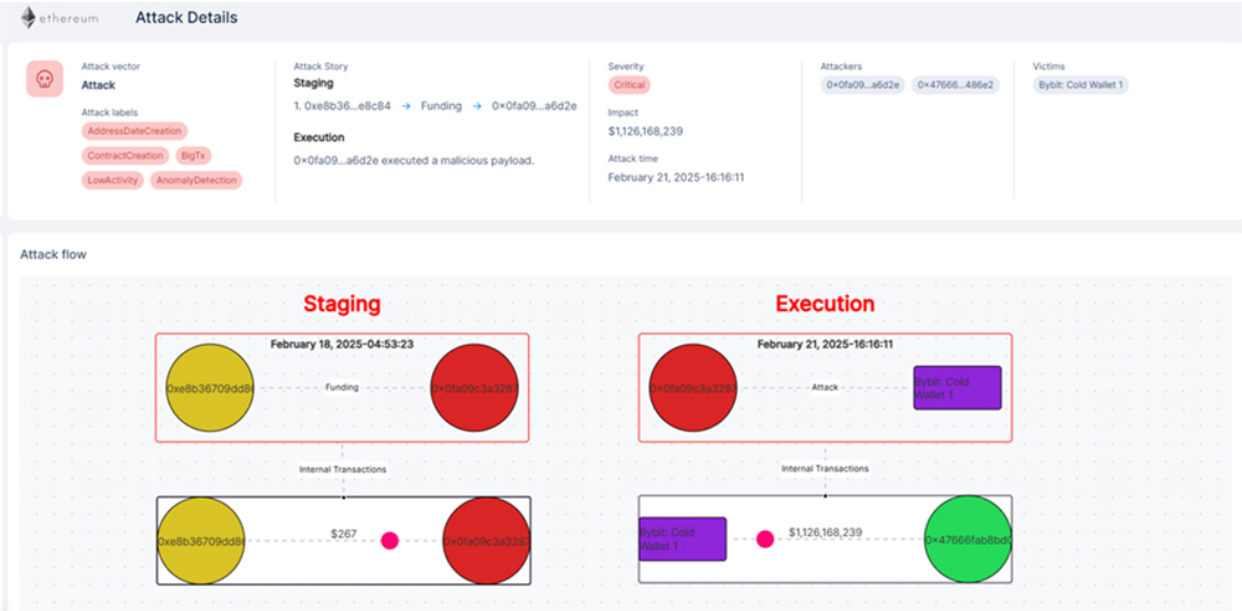
⋮ 2/23/2025

Research by Dikla Barda, Roman Ziakin and Oded Vanunu

On February 21st, Check Point Blockchain Threat Intel System alerted on a critical attack log on the Ethereum blockchain network.



The log indicated that the AI engine identify anomality change with this transaction and categorize it as critical attack in real time. It was indicated that ByBit cold wallet got hacked, resulting in the theft of approximately $1.5 billion worth of digital assets, primarily in Ethereum tokens. This incident marks one of the largest thefts in the history of the digital asset industry.



Executive Summary:

- In one of the largest thefts in digital asset history, hackers gained unauthorized access to a multisig Ethereum wallet and stole $1.5 billion worth of digital assets, primarily consisting of Ethereum tokens.
- The recent incident with Bybit marks a new phase in attack methods, featuring advanced techniques for manipulating user interfaces. Rather than just targeting protocol flaws, the attackers used sophisticated infrastructure compromise to manipulate the UI that signers interacted with.

- This past July, Check Point's Threat Intelligence Blockchain system identified and published a concerning new findings where attackers manipulating legitimate transactions through the Safe Protocol's execTransaction function.
- The recent hack highlights that multisig cold wallets are not secure if signers can be deceived, emphasizing the growing sophistication of supply chain and user interface manipulation attacks.
- The Bybit hack challenges previous beliefs about crypto security, showing that despite strong smart contracts and multisig protections, the human-interface layer remains vulnerable. This incident highlights how UI manipulation can compromise even the most secure wallets.

The Evolution of Protocol Exploitation

Published in July 2024, our research provided technical analysis of how the execTransaction function operates within the Safe framework and documented cases where it was used in attack chains.

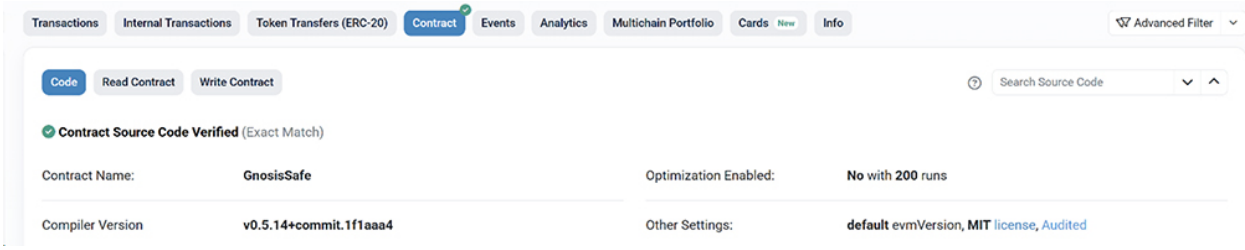
The research focused on understanding the technical mechanics of the Safe Protocol's execTransaction function and its potential for misuse, highlighting the importance of understanding how legitimate protocol features could be leveraged unexpectedly.

The recent Bybit incident represents a significant evolution of these attack patterns, introducing sophisticated UI manipulation techniques not previously seen. Instead of just exploiting protocol mechanics, the attackers compromised Safe's infrastructure to manipulate what signers saw when approving transactions, allowing them to compromise a significant institutional multisig setup.

The Attack Breakdown

Initial Compromise

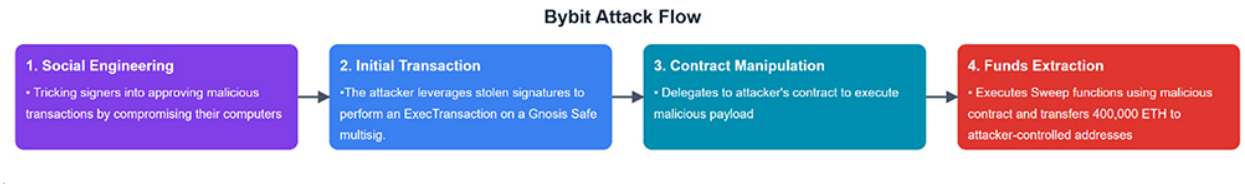
The attack originated from wallet address 0x47666fab8bd0ac7003bce3f5c3585383f09486e2, which executed an ExecTransaction on the Gnosis Safe multisig proxy contract.



This widely used multisig implementation relies on externally provided bytes signatures to authorize actions on behalf of the multisig.

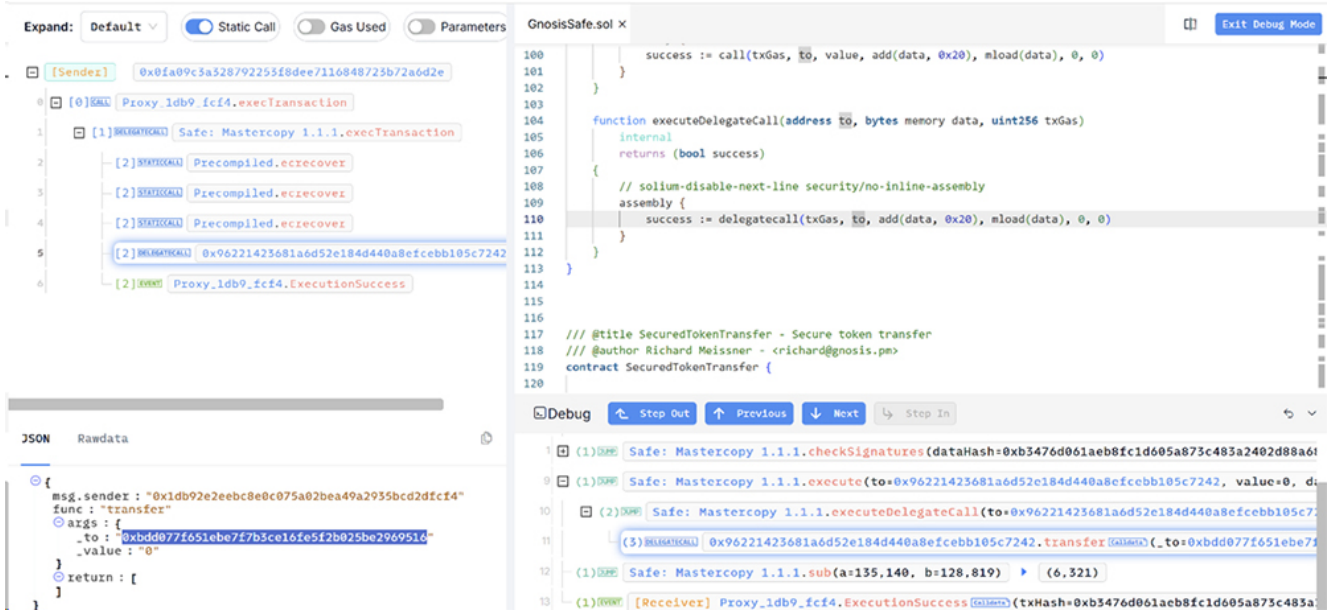
A critical vulnerability in this setup stems from Gnosis Safe's reliance on externally generated signatures rather than on-chain voting. This design choice makes it susceptible to:

- UI manipulation
- Supply-chain attacks
- Manipulation of transaction data



Attack Execution Flow

1. The attacker executed an ExecTransaction on the Gnosis Safe multisig proxy contract, manipulating what signers saw in the interface.
2. The signers interacted with what appeared to be the legitimate Safe interface to approve what they believed was a legitimate transaction.
3. The attack involved manipulation of the transaction data between what was displayed to signers and what was actually executed.
4. The attacker then executed a delegate call to their contract at 0xbdd077f651ebe7f7b3ce16fe5f2b025be2969516, triggering the transfer function



5. Since the malicious contract was unverified, it needed to be decoded to understand the function of the transfer:

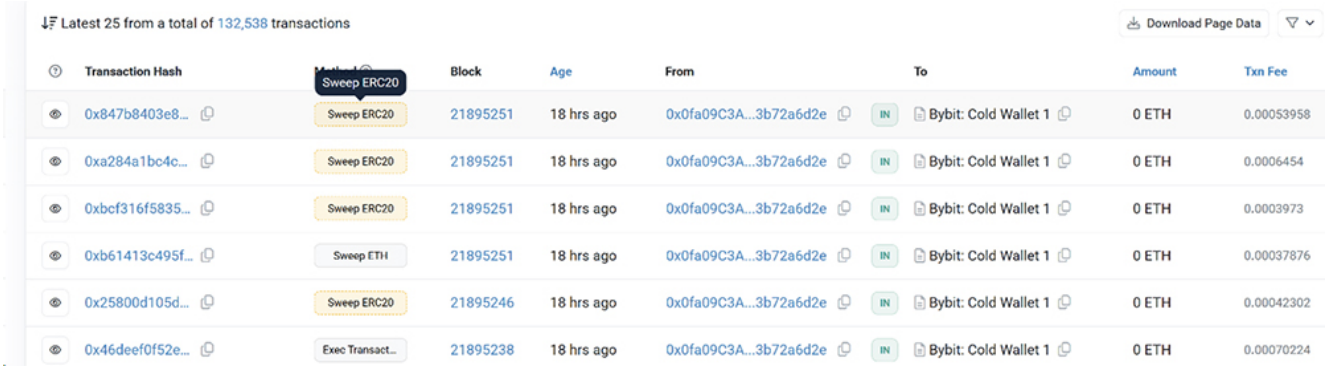
```
function transfer(address recipient, uint256 amount) public payable {  find similar
    require(msg.data.length - 4 >= 64);
    require(msg.sender == 0xfa09c3a328792253f8dee7116848723b72a6d2e, Error('Ownable: caller is not the owner'));
    _transfer = recipient;
}
```

   It contained code that overwrote the _transfer address in SLOT[0] of the contract, which executes the function in the original ByBit contract. Since the attacker's contract declares _transfer as its first state variable, when called via delegatecall, this storage slot maps to the target contract's SLOT[0]. This means that when the Safe multisig executes the delegatecall to this contract, the malicious contract's code runs in the context of the Safe contract. By overwriting this slot, the attacker could alter the contract's behaviour.
6. This manipulation redirected further delegations to the attacker's contract at 0xbDd077f651EBe7f7b3cE16fe5F2b025BE2969516
7. The attacker's contract includes Sweep functions, which were executed via delegation to transfer funds.

```
function sweepETH(address receiver) public payable {  find similar
    require(msg.data.length - 4 >= 32);
    require(msg.sender == 0xfa09c3a328792253f8dee7116848723b72a6d2e, Error('Ownable: caller is not the owner'));
    v0 = receiver.call().value(this.balance).gas(2300 * !this.balance);
    require(bool(v0), 0, RETURNDATASIZE()); // checks call status, propagates error data on error
}

function sweepERC20(address token, address to) public payable {  find similar
    require(msg.data.length - 4 >= 64);
    require(msg.sender == 0xfa09c3a328792253f8dee7116848723b72a6d2e, Error('Ownable: caller is not the owner'));
    v0, /* uint256 */ v1 = token.balanceOf(this).gas(msg.gas);
    require(bool(v0), 0, RETURNDATASIZE()); // checks call status, propagates error data on error
    require(MEM[64] + RETURNDATASIZE() - MEM[64] >= 32);
    MEM[MEM[64] + 68] = v1;
    v2 = v3 = 0;
    while (v2 >= 68) {
        MEM[v2 + MEM[64]] = MEM[32 + (MEM[64] + v2)];
        v2 += 32;
    }
    MEM[MEM[64] + 68] = 0;
    v4, /* uint256 */ v5 = token.call(68, 0xa9059cbb, to).gas(msg.gas);
    if (RETURNDATASIZE() != 0) {
        v6 = new bytes[](RETURNDATASIZE());
        RETURNDATACOPY(v6.data, 0, RETURNDATASIZE());
    }
    require(v4, Error('Token transfer failed'));
}
```

8. This series of transactions resulted in the theft of over $1 billion in assets, including 400,000 ETH

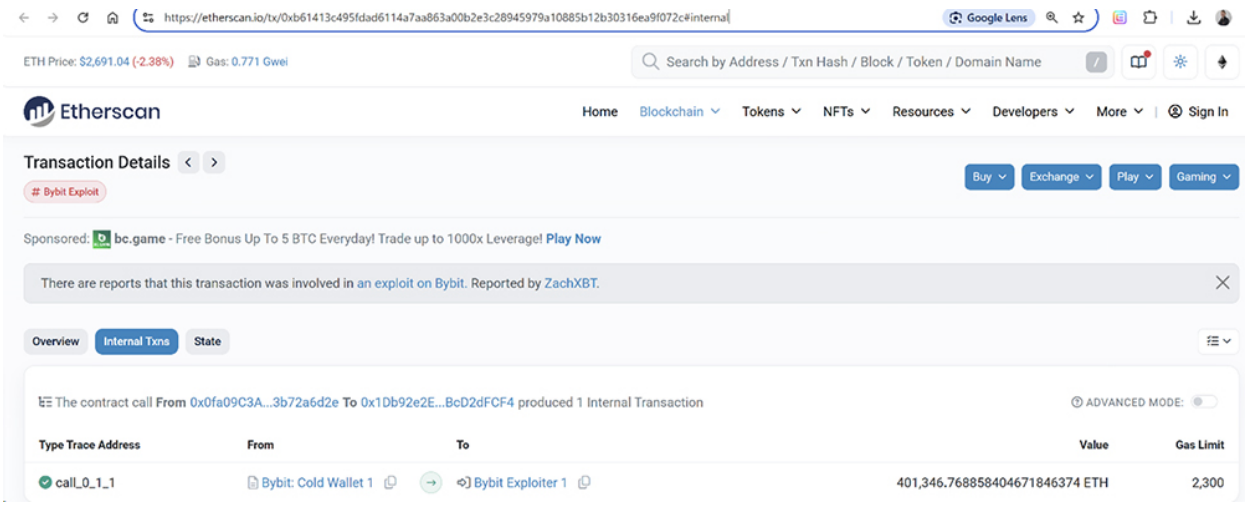


Latest 25 from a total of 132,538 transactions — Download Page Data

Transaction Hash	Method	Block	Age	From	To	Amount	Txn Fee
0x847b8403e8...	Sweep ERC20	21895251	18 hrs ago	0x0fa09C3A...3b72a6d2e	Bybit: Cold Wallet 1	0 ETH	0.00053958
0xa284a1bc4c...	Sweep ERC20	21895251	18 hrs ago	0x0fa09C3A...3b72a6d2e	Bybit: Cold Wallet 1	0 ETH	0.0006454
0xbcf316f5835...	Sweep ERC20	21895251	18 hrs ago	0x0fa09C3A...3b72a6d2e	Bybit: Cold Wallet 1	0 ETH	0.0003973
0xb61413c495f...	Sweep ETH	21895251	18 hrs ago	0x0fa09C3A...3b72a6d2e	Bybit: Cold Wallet 1	0 ETH	0.00037876
0x25800d105d...	Sweep ERC20	21895246	18 hrs ago	0x0fa09C3A...3b72a6d2e	Bybit: Cold Wallet 1	0 ETH	0.00042302
0x46deef0f52e...	Exec Transact...	21895238	18 hrs ago	0x0fa09C3A...3b72a6d2e	Bybit: Cold Wallet 1	0 ETH	0.00070224

This transaction is responsible for stealing over 1 billion dollars in lost:

For example, 400,000 ETH:



Why This Attack is So Significant

This hack sets a new precedent in crypto security by **bypassing a multisig cold wallet without exploiting any smart contract vulnerability**. Instead, it exploited **human trust and UI deception**:

- **Multisigs are no longer a security guarantee** if signers can be compromised.
- **Cold wallets aren't automatically safe** if an attacker can manipulate what a signer sees.
- **Supply chain and UI manipulation attacks are becoming more sophisticated**.

Conclusion

The Bybit hack has shattered long-held assumptions about crypto security. No matter how strong your smart contract logic or multisig protections are, the interface layer remains vulnerable. This attack proves that UI manipulation through infrastructure compromise can bypass even the most secure wallets. The industry needs to move to end-to-end prevention, where each transaction must be validated through multiple independent channels.

GO UP
BACK TO ALL POSTS