

# Harmony Bridge Hacked for \$100M due to Suspected Private Key Compromise

On June 24, Harmony Bridge was hacked for about \$100M due to suspected private key compromise. Here's our analysis of this incident.

## Background

Harmony Bridge is a cross-chain bridge with five validators for operational verification. The main reason for this attack is that the private keys of two validators are suspected to be compromised, resulting in the *confirmTransaction* function of the contract to be called successfully.

```
198      /// @param transactionId transaction ID.
199      function confirmTransaction(uint256 transactionId)
200          public
201          ownerExists(msg.sender)
202          transactionExists(transactionId)
203          notConfirmed(transactionId, msg.sender)
204      {
205          confirmations[transactionId][msg.sender] = true;
206          emit Confirmation(msg.sender, transactionId);
207          executeTransaction(transactionId);
208      }
```



Harmony



**Stolen assets  
on BSC**

**5000** BNB  
**640,000** BUSD



**Stolen assets  
on Ethereum**

<b>592</b> WBTC	<b>13,100</b> ETH
<b>43</b> WETH	<b>110,000</b> FXS
<b>990</b> AAVE	<b>9,981,000</b> USDT
<b>41,200,000</b> USDC	<b>5,530,000</b> BUSD
<b>84,620,000</b> AAG	<b>6,070,000</b> DAI
<b>415,000</b> SUSHI	<b>5,652,000</b> Frax

**Attacker Address:**  
**0x0d043128146654c7683**  
**fbf30ac98d7b2285ded00**

**Balance: \$1,801,587**

**Attacker Address:**  
**0x0d043128146654c7683**  
**fbf30ac98d7b2285ded00**

**Balance: \$99,002,488**

SWAP

**Attacker transit address1:**  
**0x9e91ae672e7f7330fc6b**  
**9bab9c259bd94cd08715**

**Attacker transit address2:**  
**0x58f4baccb411acef70a**  
**5f6dd174af7854fc48fa9**

The attacker transfers the stolen token assets to the transit address, then goes to the exchange through the transit address to exchange them for ETH, and finally transfers them back to the attacker's address



**BEOSIN**  
Blockchain Security

0xf845A7ee8477AD1FB4446651E548901a2635A915

0x812d8622C6F3c45959439e7ede3C580dA06f8f25

**Victim contract:**

0x715cdda5e9ad30a0ced14940f9997ee611496de6

**Example hash:**

0x6e5251068aa99613366fd707f3ed99ce1cb7ffdea05b94568e6af4f460cecd65

**Related transactionId:**

21106–21118(ETH),120515–120518(BNB)

The private key compromise address 0x812d86 calls the *confirmTransaction* function of the 0x715cdd contract for operation verification, and the transactionId for verification is 21107 (here the transactionId of 21107 is used as an example).

[illegible]

It can be found that in this transaction, the validation of `isConfirmed` returns `true`.

**Transaction**  
0x4a59c3e5c4...a9ced191

**Execution** **Function Trace**

- confirmTransaction
  - executeTransaction
    - isConfirmed
    - external\_call
    - unlockToken
      - safeTransfer
        - callOptionalReturn
          - isContract
          - transfer
            - \_msgSender
            - trackVotes
              - sub96
              - \_writeCheckpoint

**Stack Trace**

- confirmTransaction in MultiSigWallet.sol:195
- executeTransaction in MultiSigWallet.sol:203
- isConfirmed in MultiSigWallet.sol:226

**Source Code**

```

219 /// @param transactionId Transaction ID.
220 function executeTransaction(uint256 transactionId)
221     public
222     ownerExists(msg.sender)
223     confirmed(transactionId, msg.sender)
224     notExecuted(transactionId)
225 {
226     if (isConfirmed(transactionId)) {
227         Transaction storage txn = transactions[transactionId];
228         txn.executed = true;
229         if (
230             external_call(
231                 txn.destination,
232                 txn.value,
233                 txn.data.length,
234                 txn.data
235             )
236         ) emit Execution(transactionId);
237     } else {
238         emit ExecutionFailure(transactionId);
239         txn.executed = false;
240     }
241 }
  
```

**Output**

```

{
  "address": "0x10cdda0e9ad3090ced1494079997ee011490de0",
  "caller": {
    "address": "0x812d8622c6f3c45959439e7ede3c580da06f8f25",
    "balance": "5092258803158613132"
  },
  "input": {
    "transactionId": "21114"
  },
  "output": {
    "0": true
  }
}
  
```

However, a validator node query in the contract shows that although there are five owners, only two have been verified.

**Read Contract Information** [Expand all] [Reset]

- MAX\_OWNER\_COUNT
- confirmations
- getConfirmationCount
- getConfirmations
  - transactionId (uint256)
  - Query
  - confirmations address[]
 

```

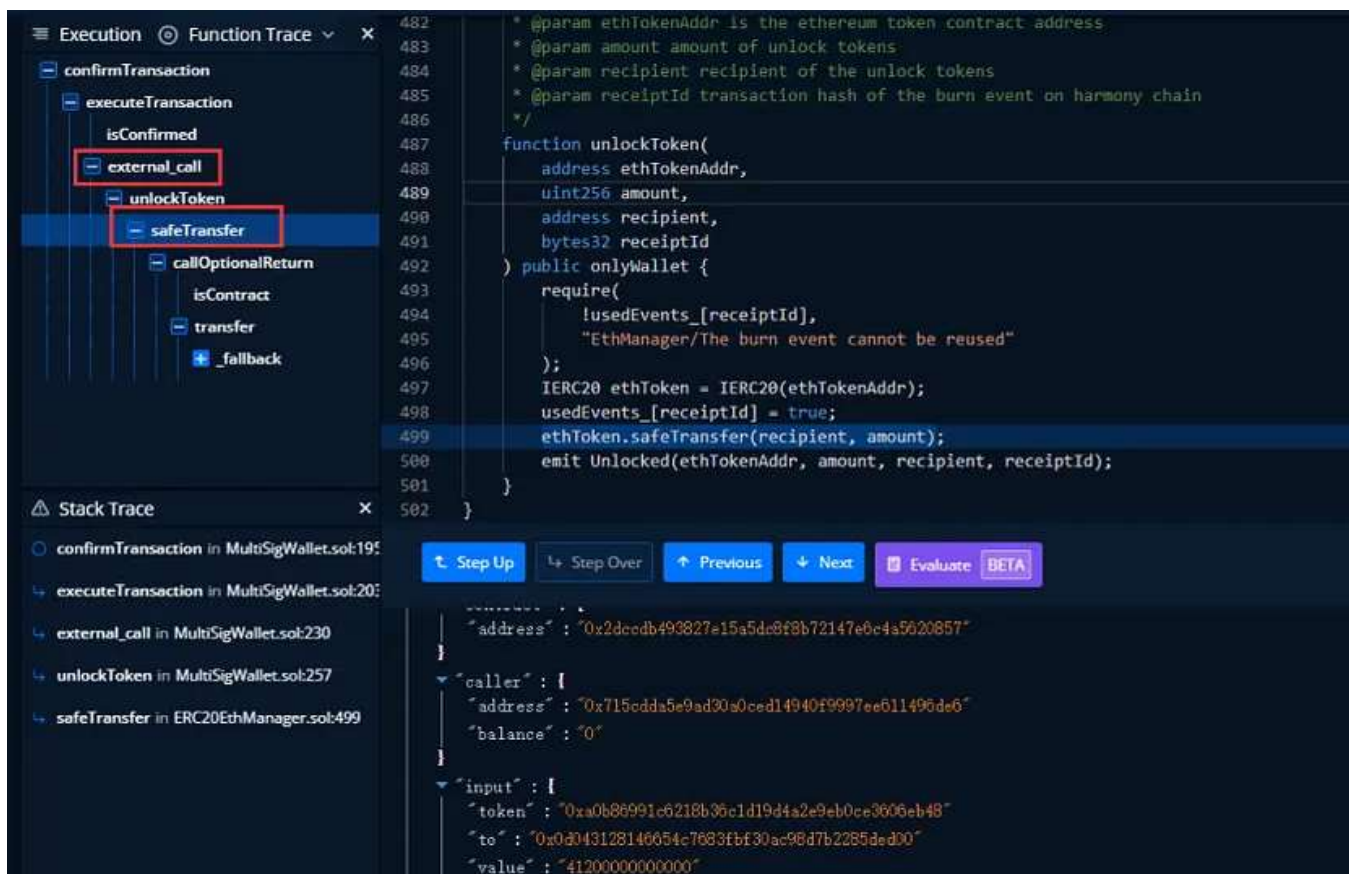
[ getConfirmations(uint256) method Response ]
confirmations address[]: 0x1045a7ee8477AD1FB4446651E548901a2635A915, 0x812d8622c6f3c45959439e7ede3c580da06f8f25
          
```
- getOwners
  - Returns list of owners.
 

```

0x1045a7ee8477ad1fb4446651e548901a2635a915, 0x40bd066666995353d9451ec9897e2003caf50ce, 0x812d8622c6f3c45959439e7ede3c580da06f8f25, 0xc608047252b3f524257d62b68aa0543818767a9d, 0xbca79094dc02418270c1374efe3118d05e9fa29 address[]
          
```

The attacker then uses these two validator nodes to successfully get the corresponding tokens using `external_call` and repeatedly exploits this attack to profit.





The project subsequently changed the number of validator nodes required to pass from 2 to 4 via transactionId of 21126 (120531 on BNB Chain).

[illegible]

## Fund Flow

The attack resulted in the loss of 85,867 ETH, 990 AAVE and 78,500,000 AAG on Ethereum, and 5,000 BNB and 640,000 BUSD on BNB Chain, for a total of about \$100,428,116. The stolen funds are still held at the attacker's address.

## Summary

The attacker took advantage of the low number of validator node verification requirements and used two validator nodes to steal millions of dollars in assets. It is recommended that the project owner try to choose more nodes when designing the number of validator verification requirements and do a good job of validator security.

## **More**

1. ***How to Steal User's Signature in NFT Phishing Attacks?***
2. ***How to Ensure the Security of NFT Under the Web 3.0 Boom?***
3. ***A Research Into NFT Whitelist Bypass Vulnerability (1/2)***
4. ***Investigation of Common Phishing Attacks in Web 3.0: Discord, Google Ads, Fake Domains and Others***
5. ***[RECAP] AMA About How to Keep Your Smart Contract Secure During Development With Beosin VaaS***
6. ***Hype, Plagiarism, Insider Fraud, NFT Scams on OpenSea and Security Advice***

## **Contact**

If you have need any blockchain security services, please contact us:

**[Website](#) [Email](#) [Official Twitter](#) [Alert](#) [Telegram](#) [LinkedIn](#)**