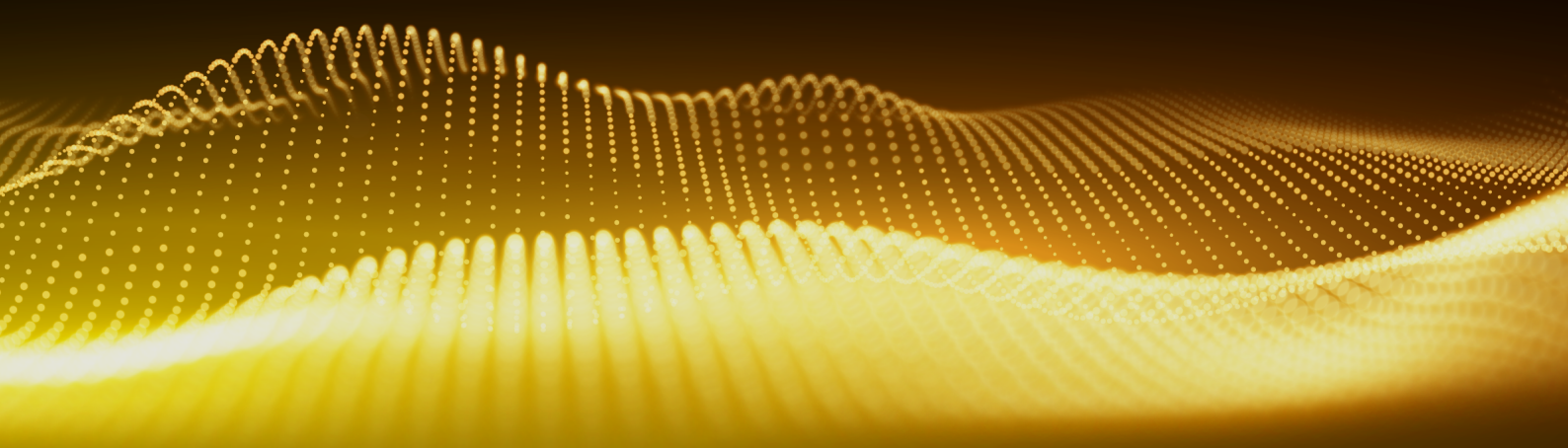
ChainAudits

Incident Report

14.09.2024

BaseBros Fi



On 13.09.2024, BaseBros Fi on the Base blockchain deleted their entire social presence, including all accounts and messages, after gaining control of and draining ecosystem funds through an unaudited and unverified Vault contract. Our blockchain security company, ChainAudits, had audited **4 out of the 5** key smart contracts used in the project. Unfortunately, the contract that facilitated the rug pull (**Vault Contract**) was not included in our audit scope, nor is it verified on the blockchain.

Audit Scope

ChainAudits accepted the BaseBros Fi audit request that included the **Brewery, Strategy, FeeManager, and Staking contracts**, all of which were later audited by our team. The **Brewery** and **Strategy contracts** included in the scope were 1:1 forks of Beefy Finance, that the team communicated to have sourced from their public Github repository. The **Vault Contract** however, which contained the backdoor vulnerability leading to the rug pull, was neither audited by us nor verified on the blockchain.

Below is a breakdown of the contracts provided by the BaseBros team and their differences compared to Beefy Finance contracts:

StrategyVelodromeGaugeV2:
https://www.diffchecker.com/incvyt9U/

Audited Contract:
https://github.com/ChainAudits/Projects/tree/main/2024/BaseBrosFi/contracts/strategies/Aerodrome

Beefy Contract:
https://github.com/beefyfinance/beefy-contracts/blob/master/contracts/BIFI/strategies/Velodrome/StrategyVelodromeGaugeV2.sol

StratFeeManagerInitializable:
https://www.diffchecker.com/rQCITNTE/

Audited Contract:
https://github.com/ChainAudits/Projects/blob/main/2024/BaseBrosFi/contracts/strategies/Common/StratFeeManagerInitializable.sol

Beefy Contract:
https://github.com/beefyfinance/beefy-contracts/blob/master/contracts/BIFI/strategies/Common/StratFeeManagerInitializable.sol

The Attack Summary

Name	Description	Link
A	Attacker Wallet	https://basescan.org/address/0x022ecd2a5fd5c831d7070e32577d15d3f1fb8f73
B	Brewery Contract	https://basescan.org/address/0x6aEA7234bAaaf3E0838B4e271171f3B5165be620#code
C	Strategy Contract	https://basescan.org/address/0x615b684D9204B2659C978e15F4E0f25e802478C6#code
D	Vault Address	https://basescan.org/address/0xd6B7f964974F5857A6ea57469EE94A4601B91eEE#code
E	Owner of All Contracts' Address	https://basescan.org/address/0xFE53193a5AA993581619D345b505507c9ae07Fc4

The attacker exploited a backdoor in **Contract D**, allowing them to drain the "*want*" balance of **Contract C**. Since this contract was neither audited nor verified, its presence and functionality were hidden from public review.

GitHub Link to ChainAudits' Report (commit unchanged):
https://github.com/ChainAudits/Projects/blob/main/2024/BaseBrosFi/ChainAudits_SmartContract_Audit_BaseBrosFi.pdf
Commit: 9c3ef23

As can be observed in the link above, the smart contract audit report and the contracts provided to us by the BaseBros Fi team remain unchanged since the audit was published.

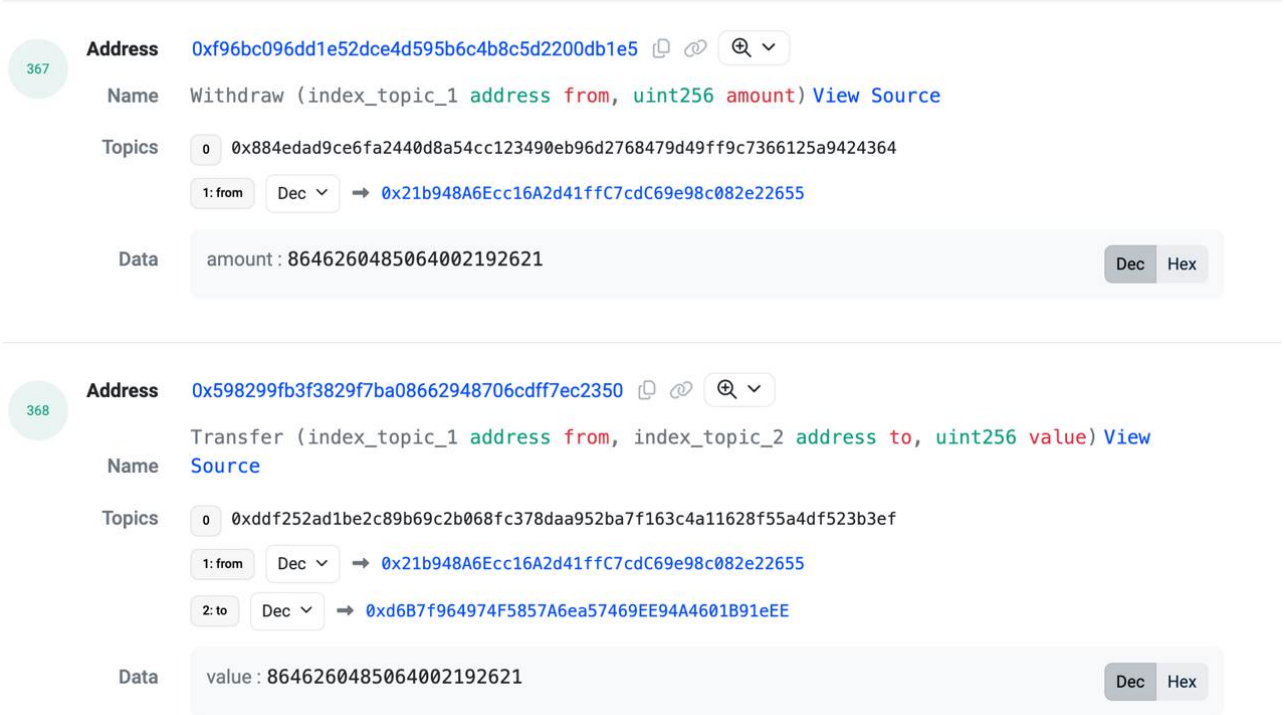
Note – After the audit was finished, ChainAudits did not receive deployed contract addresses from the BaseBros Fi team. Moreover, the other strategy addresses used in the exploit were not part of the audit scope, and we never received their code.

Root Cause

The funds deposited in the Strategy Contract could be withdrawn by the Vault address, which was the case here. It is also evident that the same person has control over both **Wallet A** and **Contract E**.

1. The malicious actors set the unverified Vault address in **Contract C**, Txid: https://basescan.org/tx/0x2bfaa7a8f0536bb39ba6fef2ac4c14d9a763547a95e0e1344dd40e85d39b47fc

2. They then called an unknown (not visible as the contract is not verified) method in **Contract D** (Vault Contract) and in that single transaction, the funds were withdrawn from the **Strategy Contract** to the **Vault Contract**.



3. To achieve this, the team overrode the "*beforeDeposit*" function in the **Strategy Contract**. This function is triggered during the deposit process in the **Brewery Contract**. By doing so, they moved funds from the Brewery Contract to the Strategy Contract. During the deposit process, the state variable "*harvestOnDeposit*" was set to true, which was controlled by the team. By automatically setting fees to 0, the team manipulated the contract in their favour. Afterward, they called the "*harvest*" function, passing their own address to complete the exploit. These actions were performed by the unverified **Vault contract**.

4. The malicious actors also "retired" all **Strategy** Contracts, withdrawing their balances to the vault. Here are the transactions for reference: https://app.blocksec.com/explorer/tx/base/0xa0bd1f66141d00fb9340e91403066f49d02cbbdcacd6a499f64834dd5f2d269e

List of all involved strategy addresses:

- 0x615b684d9204b2659c978e15f4e0f25e802478c6
- 0x34d643c3d1002c1d222c3f4347a56907aed0aa7d
- 0xc75e8dfbc301a128519ef11ebc40d07fa28f68ad
- 0x3af181de50bbc952c8c1ab97e517a3f2649c002b
- 0x21b948a6ecc16a2d41ffc7cdc69e98c082e22655
- 0x3ce0708c5a8364fb18d56ec6ad56463b544999e7
- 0x58d97f3772939c4d55120b334815b9f0b7bc1c77
- 0x36ee61b39f0a5b4dbdc9ebfedd05d27d33d658a8
- 0xc99cb8e221984a983927cc268e76cdb1d0fb4c21
- 0xcd30a02e50b9ec3c0521fb2ddba7bd6eb2820ecd
- 0xb33a0d6b4a96f4f01aafdc58a698f93cbac09a42
- 0x277f6919042e4ea76b69f8e6e97cb79d96fe3bfa

In all the addresses, it can be observed that the vault address is set to the Attacker's Wallet address, which is why it was possible to call the "*retireStrat*" function.

Note – This rug pull/attack was misinterpreted as affecting the Seamless protocol due to the labeling of the contracts used in this attack. BaseBros Fi is the only affected protocol which led to the team draining multiple pools associated with the BaseBros Fi platform.

Example of the Potential Exploit:

```
function exploit(StrategyVelodromeGaugeV2[] _strategyAddresses⬆) external onlyOwner {
for(uint256 i = 0; i < _strategyAddresses⬆.length; i++) {
    StrategyVelodromeGaugeV2 strat = _strategyAddresses⬆[i];
    strat.retireStrat();
    IERC20 _token = IERC20(strat.want());
    uint256 vaultBalance = _token.balanceOf(address(this));
    _token.transfer(msg.sender, vaultBalance);

}
```

1. The attacker passed the strategy addresses via a dynamic array in the exploit function

2. The "*retireStrat*" function is called in a loop to call all strategy contracts.

3. The tokens received from the strategy contracts were then transferred to the attacker's wallet.

Disclaimer

ChainAudits reports should not be construed as an endorsement or disapproval of any particular business or group. These reports do not reflect the economic value of any of the products or assets developed by the group. Also, ChainAudits does not consider integration with external contracts or services (e.g., Unicrypt, Uniswap, PancakeSwap).

ChainAudits reports aim to identify successful audit processes to help our clients improve the quality of the code and manage the risks associated with cryptographic tokens and blockchain technology. It is important to understand that blockchain technology and cryptographic assets pose significant ongoing risks. Each company and individual should conduct its own due diligence to maintain consistent safety measures. ChainAudits makes no representations about the security or performance of the technologies we audit.

ChainAudits does not provide any warranty or guarantee that the analyzed technology is completely defect-free, nor does it imply approval by the technology's owners. These

audits should not be used to make input or output decisions; they will be involved in any project. They are not giving financial advice and should not be construed as such.