SALUS SECURITY

MAR 2024

CODE SECURITY ASSESSMENT

HEDGEY FINANCE

Overview

Project Summary

- Name: Hedgey Finance - NFTLocks
- Platform: EVM-compatible chains
- Language: Solidity
- Repository:
    - https://github.com/hedgey-finance/NFTLocks
- Audit Range: See Appendix - 1

Project Dashboard

Application Summary

Name	Hedgey Finance - NFTLocks
Version	v2
Type	Solidity
Dates	Mar 31 2024
Logs	Mar 28 2024; Mar 31 2024

Vulnerability Summary

Total High-Severity issues	0
Total Medium-Severity issues	0
Total Low-Severity issues	2
Total informational issues	1
Total	3

Contact

E-mail: support@salusec.io

Risk Level Description

High Risk	The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for clients' reputations or serious financial implications for clients and users.
Medium Risk	The issue puts a subset of users' sensitive information at risk, would be detrimental to the client's reputation if exploited, or is reasonably likely to lead to a moderate financial impact.
Low Risk	The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client's business circumstances.
Informational	The issue does not pose an immediate risk, but is relevant to security best practices or defense in depth.

Content

Introduction

1.1 About SALUS

At Salus Security, we are in the business of trust.

We are dedicated to tackling the toughest security challenges facing the industry today. By building foundational trust in technology and infrastructure through security, we help clients to lead their respective industries and unlock their full Web3 potential.

Our team of security experts employ industry-leading proof-of-concept (PoC) methodology for demonstrating smart contract vulnerabilities, coupled with advanced red teaming capabilities and a stereoscopic vulnerability detection service, to deliver comprehensive security assessments that allow clients to stay ahead of the curve.

In addition to smart contract audits and red teaming, our Rapid Detection Service for smart contracts aims to make security accessible to all. This high calibre, yet cost-efficient, security tool has been designed to support a wide range of business needs including investment due diligence, security and code quality assessments, and code optimisation.

We are reachable on Telegram (https://t.me/salusec), Twitter (https://twitter.com/salus_sec), or Email (support@salusec.io).

1.2 Audit Breakdown

The objective was to evaluate the repository for security-related issues, code quality, and adherence to specifications and best practices. Possible issues we looked for included (but are not limited to):

- Risky external calls
- Integer overflow/underflow
- Transaction-ordering dependence
- Timestamp dependence
- Access control
- Call stack limits and mishandled exceptions
- Number rounding errors
- Centralization of power
- Logical oversights and denial of service
- Business logic specification
- Code clones, functionality duplication

1.3 Disclaimer

Note that this security audit is not designed to replace functional tests required before any software release and does not give any warranties on finding all possible security issues with the given smart contract(s) or blockchain software, i.e., the evaluation result does not guarantee the nonexistence of any further findings of security issues.

Findings

2.1 Summary of Findings

ID	Title	Severity	Category	Status
1	Use safeTransferFrom instead of transferFrom	Low	Risky External Calls	Ackownledged
2	Missing emergency withdrawal function	Low	Centralization	Acknowledged
3	PUSH0 bytecode might not be deployed on EVM-compatible chains	Informational	Compiler	Resolved

2.2 Notable Findings

Significant flaws that impact system confidentiality, integrity, or availability are listed below.

1. Use safeTransferFrom instead of transferFrom

Severity: Low	Category: Risky External Calls
Target: - contracts/NFTLock.sol	

Description

NFTLock.sol:L125

```
IERC721(locks[lockId].nft).transferFrom(address(this), msg.sender,
locks[lockId].tokenId);
```

There are some smart contracts that don't support ERC721, using transferFrom() may result in the NFT being sent to such contracts and the NFT will be locked in those contracts forever. According to EIP721, a wallet/broker/auction application MUST implement the wallet interface if it will accept safe transfers.

Recommendation

Consider using the safeTransferFrom() method instead of transferFrom() for NFT transfers.

Status

The issue has been acknowledged by the team.

2. Missing emergency withdrawal function	
Severity: Low	Category: Centralization
Target: - contracts/NFTLock.sol	

Description

According to README.md, the contract will lock the UniswapV3 position NFT. Users can withdraw the corresponding underlying assets by burning. If the UniswapV3 pool is attacked or asset prices fluctuate sharply during the UniswapV3 position NFT locked in the contracts, users need to call unlockNFT() to warp the UniswapV3 position NFT immediately to avoid losses. But the UniswapV3 position NFT will still lock in the contacts before unlockDate which means that users can't avoid losses.

Recommendation

Consider adding an emergency withdrawal function.

Status

The issue has been acknowledged by the team.

2.3 Informational Findings

3. PUSH0 bytecode might not be deployed on EVM-compatible chains

Severity: Informational	Category: Compiler
Target: - contracts/NFTLock.sol	

Description

```
pragma solidity 0.8.24;
```

The protocol is expected to be deployed on multiple EVM-compatible chains (Optimism, ArbitrumOne, Polygon, etc) but the pragma statement shows usage of 0.8.24 version of the Solidity compiler. This version (and every version after 0.8.19) will use the PUSH0 opcode, which is still not supported on some EVM-compatible chains, for example, Fantom Opera.

Verify PoC:

```
cast call  --rpc-url "https://1rpc.io/ftm" --create 0x5f
```

Recommendation

To ensure that the same deterministic bytecode can be deployed to all EVM-compatible chains, consider using version 0.8.19.

Status

The issue has been resolved by the team.

Appendix

Appendix 1 - Files in Scope

This audit covered the following files in commit dd771c2:

File	SHA-1 hash
contracts/NFTLock.sol	11e16baa8d8c2487500d52e2e13a35203b67f4cc