

UNCX Liquidity Locker V3 Security Audit Report

Contents

Contents	1
Executive Summary	2
Project Overview	2
Audit Scope	3
Audit Methodology	4
Findings Summary	6
Findings	7
Conclusion	23
Disclaimer	23

Executive Summary

Title	Description
Client	UNCX
Project	Locker V3
Platform	Ethereum
Language	Solidity
Repository	-
Initial commit	-
Timeline	November 23 2023 - December 18 2023

Project Overview

UNCX Network is a one-stop shop DeFi protocol providing several main services with several sub-features.

Audit Scope

File	Link
UNCX_ProofOfReservesV2_UniV3.sol	<u>0x7f5c649856f900d15c83741f45ae46f5c6858234</u>

Audit Methodology

General Code Assessment

The code is reviewed for clarity, consistency, style, and whether it follows code best practices applicable to the particular programming language used, such as indentation, naming convention, commented code blocks, code duplication, confusing names, irrelevant or missing comments, etc. This part is aimed at understanding the overall code structure and protocol architecture. Also, it seeks to learn overall system architecture and business logic and how different parts of the code are related to each other.

Code Logic Analysis

The code logic of particular functions is analyzed for correctness and efficiency. The code is checked for what it is intended for, the algorithms are optimal and valid, and the correct data types are used. The external libraries are checked for relevance and correspond to the tasks they solve in the code. This part is needed to understand the data structures used and the purposes for which they are used. At this stage, various public checklists are applied in order to ensure that logical flaws are detected.

Entities and Dependencies Usage Analysis

The usages of various entities defined in the code are analyzed. This includes both: internal usage from other parts of the code as well as possible dependencies and integration usage. This part aims to understand and spot overall system architecture flaws and bugs in integrations with other protocols.

Access Control Analysis





Access control measures are analyzed for those entities that can be accessed from outside. This part focuses on understanding user roles and permissions, as well as which assets should be protected and how.

Use of checklists and auditor tools



Auditors can perform a more thorough check by using multiple public checklists to look at the code from different angles. Static analysis tools (Slither) help identify simple errors and highlight potentially hazardous areas. While using Echidna for fuzz testing will speed up the testing of many invariants, if necessary.

Vulnerabilities





The audit is directed at identifying possible vulnerabilities in the project's code. The result of the audit is a report with a list of detected vulnerabilities ranked by severity level:































Severity	Description
 Critical	Vulnerabilities leading to the theft of assets, blocking access to funds, or any other loss of funds.
 High	Vulnerabilities that cause the contract to fail and that can only be fixed by modifying or completely replacing the contract code.
 Medium	Vulnerabilities breaking the intended contract logic but without loss of funds and need for contract replacement.
 Low	Minor bugs that can be taken into account in order to improve the overall quality of the code

After the stage of bug fixing by the Customer, the findings can be assigned the following statuses:

Status	Description
 Fixed	Recommended fixes have been made to the project code and no longer affect its security.
 Acknowledged	The Customer took into account the finding. However, the recommendations were not implemented since they did not affect the project's safety.

Findings Summary

Severity	# of Findings
 Critical	0
 High	5
 Medium	3
 Low	7

ID	Severity	Title	Status
H-1	 High	An incorrect tick range	
H-2	 High	Refund tokens may be stolen	
H-3	 High	Missing slippage control	
H-4	 High	The old collector have access to the lock	
H-5	 High	The ability to manipulate NPM addresses.	
M-1	 Medium	No range of values	
M-2	 Medium	Crosscontract read-only reentrancy	
M-3	 Medium	A disabled fee is not supported	
L-1	 Low	Null address checks	
L-2	 Low	Typos	
L-3	 Low	The country list doesn't work	
L-4	 Low	Max unlock timestamp is too high	
L-5	 Low	User can provide poisoned nftManager	
L-6	 Low	Function <code>safeApprove</code> is deprecated	
L-7	 Low	Magic numbers	

Findings

Critical

Not Found

High

H-1

High

An incorrect tick range

Fixed

Description

[0x7f5c649856f900d15c83741f45ae46f5c6858234](#)

The `lock` function checks that a position is full-range. If the position is not full range, then the `lock` function converts it to full-range.

When the user locks funds with `tickLower = -100` and `tickUpper = maxTick`, this position will not be converted.

Recommendation

We recommend replacing:

```
position.tickLower != - maxTick && position.tickUpper != maxTick
```

by

```
position.tickLower != - maxTick || position.tickUpper != maxTick
```


Description

[0x7f5c649856f900d15c83741f45ae46f5c6858234](#)

[0x7f5c649856f900d15c83741f45ae46f5c6858234](#)

[0x7f5c649856f900d15c83741f45ae46f5c6858234](#)

Only the owner can withdraw funds with the `adminRefundERC20` function.

But anyone can call the `lock` contract and appropriate money for themselves.

Let's consider the following case:

1. Alice sends `100 ABC` and Bob sends `1 XYZ` to a contract by mistake;
2. Charlie:
 - Charlie creates a position in UniV3, with the minimal amount of tokens (`ABC & XYZ`), the position must be non-full;
 - Charlie gives an approve for the token to the contract;
 - Charlie calls the `lock` function with `lockPeriod = 1 seconds`;
 - As the position is not full, this position will be converted by `_convertPositionToFullRange`;
 - The function removes Charlie's position, and after that determines how many tokens should be processed using the following formula:

```
mintParams.amount0Desired =
    IERC20(mintParams.token0).balanceOf(address(this));
mintParams.amount1Desired =
    IERC20(mintParams.token1).balanceOf(address(this));
```

- After a new position is created, unused tokens are refunded:

```
uint256 balance0 =
    IERC20(mintParams.token0).balanceOf(address(this));
uint256 balance1 =
    IERC20(mintParams.token1).balanceOf(address(this));
```

- After the lock is created, Charlie's position grabs all Bob's and Alice's tokens.

Recommendation

We recommend checking `balanceBefore = token.balanceOf(this)` and `balanceAfter = token.balanceOf(this) - balanceBefore` for the `_convertPositionToFullRange` function.

H-3

High

Missing slippage control

Fixed

Description

[0x7f5c649856f900d15c83741f45ae46f5c6858234](#)

When the `mint` function is called, it consumes the following mint params:

```
mintParams.amount0Min = 0;  
mintParams.amount1Min = 0;
```

The attacker can move price from actual price and mint can happen with unfavorable conditions for the user.

Recommendation

We recommend adding the `amount0Min` and the `amount1Min` to the `lock` params.

H-4

● High

The old collector have access to the lock

Fixed

Description

When transferring a lock from the "old" owner to the "new" owner, only the `userLock.owner` and `userLock.pendingOwner` variables are changed.

[0x7f5c649856f900d15c83741f45ae46f5c6858234](#)

The old `collectAddress` can collect fees.

Recommendation

We recommend setting `userLock.additionalCollector` and `userLock.collectAddress` to `address(0)` when transferring ownership.

Description

[0x7f5c649856f900d15c83741f45ae46f5c6858234](#)

The `lock (LockParams calldata params)` function allows users to lock their NFT positions, converts nft to full range, and collects fees and sends them back to the collector. It also supports different NPM addresses that are forks of Uniswap NPM, thus users have to input a NPM address in `LockParams` to successfully lock nft.

Due to the weak code logic, here are some scenarios where an attacker can steal ERC20 tokens from the contract or NFT tokens (which actually NFT UniV3 positions or others). The core of vulnerability is in the `_convertPositionToFullRange()` function, especially this part:

```
TransferHelper.safeApprove(
    mintParams.token0,
    address(_nftPositionManager),
    mintParams.amount0Desired
);
TransferHelper.safeApprove(
    mintParams.token1,
    address(_nftPositionManager),
    mintParams.amount1Desired
);
```

This part of code makes approve:

- of `mintParams.token0` which are actually controlled by a custom NPM (controlled by the attacker, the NPM can return any token address);
- to `address(_nftPositionManager)` the custom NPM address (controlled by the attacker);
- by the amount `mintParams.amount1Desired` which are actually `balanceOf()` token0 on the current address.

It means that if there is some ERC20 (USDC for ex.) token balance, the attacker:

- calls `function lock()`, with custom `LockParams.nftPositionManager`;
- `_convertPositionToFullRange` has to be triggered during `function lock()`.
- `IERC20(mintParams.token0).balanceOf(address(this))` will be stored in the `mintParams.amount0Desired` variable (for ex 1000 USDC).
- `TransferHelper.safeApprove(mintParams.token0, address(_nftPositionManager), mintParams.amount0Desired);`

and consequently approve will be made to the custom NPM address of these 1000 USDC.

As a result of these executions, the attacker then can call the `USDC.transferFrom()` function from the custom NPM contracts and steal assets.

Now, let's consider an attack on NFT assets and dive into the `_convertPositionToFullRange()` function again:

- `TransferHelper.safeApprove(mintParams.token0, address(_nftPositionManager), mintParams.amount0Desired);`

To successfully steal some NFT, we need that this contract makes approve of target NFT to the attacker's custom NPM address.

It means the custom NPM address should return `mintParams.token0`, a real Uni NFT NPM address.

Also currently we see `balanceOf()` NFTs at `UNCX_ProofOfReservesV2_UniV3` is 127.

It means that if the custom NPM (controlled by the attacker) returns `mintParams.token0` as a real NPM, `mintParams.amount0Desired` will be 127. But actually `UNCX_ProofOfReservesV2_UniV3` does not have 127 NFT ID at balance. Let's assume it has NFT ID 500, 501, 502...627.

Thus, the attacker can mint new NFT positions with the minimal range and a LP position with 1 wei, let's say, 400 times (400 new NFTs). Then the attacker sends all of them to `UNCX_ProofOfReservesV2_UniV3` and now `UNCX_ProofOfReservesV2_UniV3` has $127 + 400 = 527$ nft balances.

It means `mintParams.amount0Desired` now will be 527 in expression:

```
TransferHelper.safeApprove(mintParams.token0, address(_nftPositionManager),  
mintParams.amount0Desired);
```

and we can steal NFT id = 527 from this contract by calling `transferFrom` as in the ERC20 scenario.

As a result, the attacker can mint necessary NFTs and send them to the smart contract to manipulate this variable: `mintParams.amount0Desired`.

Recommendation

We recommend using only white-listed NPM addresses which have to be controlled by `onlyOwner`.

Medium

M-1

Medium

No range of values

Acknowledged

Description

Fee values have no limits.

[0x7f5c649856f900d15c83741f45ae46f5c6858234](#)

Recommendation

We recommend adding ranges for each value.

Description

[0x7f5c649856f900d15c83741f45ae46f5c6858234](#)

The `withdraw` function updates the state after the `safeTransferFrom` function.

An attacker can use the contract to attack other contracts. The attack will include calling the view function to check the assets. Thus, the attacker will have double assets.

Also, a cross reentrancy attack is possible, but we cannot provide the scenario.

Recommendation

We recommend changing the state before `saveTransferFrom`.

Description

In protocol in the `lock` function:

```
int24 maxTick =  
tickSpacingToMaxTick(factory.feeAmountTickSpacing(position.fee))
```

As docs say, `feeAmountTickSpacing(uint24)` – Returns 0 in case of unenabled fee.

If 0 is returned, it gives zero division in `tickSpacingToMaxTick`:

```
function tickSpacingToMaxTick(  
    int24 tickSpacing  
) public pure returns (int24 maxTick) {  
    // @audit tickSpacing can be 0 in case if fee is not enabled.  
    // Division by zero in that case  
    maxTick = (887272 / tickSpacing) * tickSpacing;  
}
```

[0x7f5c649856f900d15c83741f45ae46f5c6858234](#)

[0x7f5c649856f900d15c83741f45ae46f5c6858234](#)

Recommendation

Process a case when `feeAmountTickSpacing` returned 0.

Low

L-1

● Low

Null address checks

Acknowledged

Description

The `_receiver` param is not checked:

[0x7f5c649856f900d15c83741f45ae46f5c6858234](#)

The `_recipient` param is not checked:

[0x7f5c649856f900d15c83741f45ae46f5c6858234](#)

[0x7f5c649856f900d15c83741f45ae46f5c6858234](#)

Recommendation

We recommend adding null address checks.

Client's commentary

L-2

● Low

Typos

Acknowledged

Description

Typos:

- `adress:`

[0x7f5c649856f900d15c83741f45ae46f5c6858234](#)

Recommendation

We recommend fixing the typos.

L-3

● Low

The country list doesn't work

Acknowledged

Description

Any user can change the country code:

[0x7f5c649856f900d15c83741f45ae46f5c6858234](#)

Also, the country code is not checked in the `transferLockOwnership` function:

[0x7f5c649856f900d15c83741f45ae46f5c6858234](#)

Recommendation

We recommend moving this functionality to the frontend.

Client's commentary

L-4

● Low

Max unlock timestamp is too high

Acknowledged

Description

The maximum value of `_unlock_date` is 1e10 (2286 year).

[0x7f5c649856f900d15c83741f45ae46f5c6858234](#)

[0x7f5c649856f900d15c83741f45ae46f5c6858234](#)

Recommendation

We recommend using several years as the maximum value for locks and a unique value for infinite locks.

L-5

● Low

User can provide poisoned nftManager

Acknowledged

Description

User can provide any address for `nftPositionManager` in params
[0x7f5c649856f900d15c83741f45ae46f5c6858234](#).

Recommendation

We recommend using only white-listed NPM addresses which have to be controlled by `onlyOwner`.

L-6

Low

Function `safeApprove` is deprecated

Acknowledged

Description

The `safeApprove` function is deprecated ([SafeERC20.sol#L38-L49](#))

[0x7f5c649856f900d15c83741f45ae46f5c6858234](#)

[0x7f5c649856f900d15c83741f45ae46f5c6858234](#)

[0x7f5c649856f900d15c83741f45ae46f5c6858234](#)

[0x7f5c649856f900d15c83741f45ae46f5c6858234](#)

Recommendation

We recommend using the `safeIncreaseAllowance` and `safeDecreaseAllowance` functions instead of `safeApprove`.

L-7

● Low

Magic numbers

Acknowledged

Description

[0x7f5c649856f900d15c83741f45ae46f5c6858234](#)

Recommendation

We recommend using constants instead of numbers.

Conclusion

Altogether, the audit process has revealed 5 HIGH, 3 MEDIUM and 7 LOW severity findings.

Disclaimer

The Stronghold audit makes no statements or warranties about the utility of the code, the safety of the code, the suitability of the business model, investment advice, endorsement of the platform or its products, the regulatory regime for the business model, or any other statements about the fitness of the contracts to purpose, or their bug-free status. The audit documentation is for discussion purposes only.