BAIL security

BAILSEC.IO

EMAIL : OFFICE@BAILSEC.IO

TWITTER : @BAILSECURITY

TELEGRAM : @HELLOATBAILSEC

FINAL REPORT:

Trustswap LockToken - Update

May 2024

Disclaimer:

Security assessment projects are time-boxed and often reliant on information that may be provided by a client, its affiliates, or its partners. As a result, the findings documented in this report should not be considered a comprehensive list of security issues, flaws, or defects in the target system or codebase.

The content of this assessment is not an investment. The information provided in this report is for general informational purposes only and is not intended as investment, legal, financial, regulatory, or tax advice. The report is based on a limited review of the materials and documentation provided at the time of the audit, and the audit results may not be complete or identify all possible vulnerabilities or issues. The audit is provided on an "as-is," "where-is," and "as-available" basis, and the use of blockchain technology is subject to unknown risks and flaws.

The audit does not constitute an endorsement of any particular project or team, and we make no warranties, expressed or implied, regarding the accuracy, reliability, completeness, or availability of the report, its content, or any associated services or products. We disclaim all warranties, including the implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

We assume no responsibility for any product or service advertised or offered by a third party through the report, any open-source or third-party software, code, libraries, materials, or information linked to, called by, referenced by, or accessible through the report, its content, and the related services and products. We will not be liable for any loss or damages incurred as a result of the use or reliance on the audit report or the smart contract.

The contract owner is responsible for making their own decisions based on the audit report and should seek additional professional advice if needed. The audit firm or individual assumes no liability for any loss or damages incurred as a result of the use or reliance on the audit report or the smart contract. The contract owner agrees to indemnify and hold harmless the audit firm or individual from any and all claims, damages, expenses, or liabilities arising from the use or reliance on the audit report or the smart contract.

By engaging in a smart contract audit, the contract owner acknowledges and agrees to the terms of this disclaimer.

1. Project Details

Important:
Please ensure that the deployed contract matches the source-code of the last commit hash.

Project	TrustSwap - LockToken - Update
Website	trustswap.com
Type	Locking contract
Language	Solidity
Methods	Manual Analysis
Github repository	Latest audited commit: https://github.com/trustswap/teamfinance-contract-tokenlock/blob/f8486570d0a2b1e07fbf903c632473d3f7336f29/contracts/LockToken.sol Newly provided commit: https://github.com/trustswap/teamfinance-contract-tokenlock/blob/f10643c6b8bc4a0d094a13d344575684d59064c1/contracts/LockTokenUpgrade.sol
Resolution 1	https://github.com/trustswap/teamfinance-contract-locktoken/blob/842a9301e6b4bfa673f80eb614e47af28afe7e1f/contracts/LockToken.sol

2. Detection Overview

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
High	1	1		
Medium				
Low	2	1		1
Informational				
Governance				
Total	3	2		1

2.1 Detection Definitions

Severity	Description
High	The problem poses a significant threat to the confidentiality of a considerable number of users' sensitive data. It also has the potential to cause severe damage to the client's reputation or result in substantial financial losses for both the client and the affected users.
Medium	While medium level vulnerabilities may not be easy to exploit, they can still have a major impact on the execution of a smart contract. For instance, they may allow public access to critical functions, which could lead to serious consequences.
Low	Poses a very low-level risk to the project or users. Nevertheless, the issue should be fixed immediately
Informational	Effects are small and do not pose an immediate danger to the project or users
Governance	Governance privileges which can directly result in a loss of funds or other potential undesired behavior

3. Detection

TrustSwap - LockToken - Update

The focus of this audit is to analyze and validate the changes introduced between two commits on the contract's GitHub repository. Our aim is to assess the implications of these changes on the contract's security, functionality, and overall performance.

Background

TrustSwap's LockToken contract serves as a crucial component in the ecosystem, providing mechanisms to lock tokens and manage locked positions. The latest audited version of this contract, found at commit f848657, has been in operation with acknowledged bugs for which resolutions have not yet been provided.

Recent Changes

The newly provided commit f10643c introduces two significant updates to the LockToken contract, aimed at enhancing its functionality and user flexibility:

1. collectUniswapV3LPFees: This function has been added to allow any party to claim accrued LP fees from a Uniswap V3 position. This update is intended to facilitate easier management and recovery of liquidity provider fees from their CLAMM positions.

2. recoverAssets: This function provides the ability to migrate a locked position from one address to another, potentially increasing the administrative flexibility and handling of assets under unexpected circumstances or for strategic reallocation.

The following diff-check illustrates the changes:

https://www.diffchecker.com/DYTA5kU9/

Audit Scope and Objectives

The audit will focus on these updates to evaluate:

- The correctness and efficiency of the new code implementations.
- The security aspects, specifically checking for vulnerabilities introduced or unresolved from the previous version.
- The potential impacts these changes may have on the contract's existing functionalities and its interactions with external contracts and services.

Our review will include a thorough diff-check, using tools such as Diffchecker, to precisely identify and analyze every modification made. This structured approach ensures transparency and comprehensive coverage of the contract's updated features.

By the end of this audit, we aim to provide a detailed report that discusses the safety, efficiency, and appropriateness of the changes, alongside any recommendations for further improvements or corrections necessary.

Issue_01	Lack of validation allows users to steal all fees from all positions
Severity	**High**
Description	The collectUniswapV3LPFees function allows a user to pass the _id which is corresponding to the lockedNFTs mapping. The lockedNFT.tokenId is then used to collect the fees from the position manager. It is however not checked if this tokenId is corresponding to the NonfungiblePositionManager address. This allows a malicious user to create a lock position with any tokenId that mimics the tokenId of an already deposited, valid position. Consider the following scenario: a) Alice has a UniswapV3 position with tokenId 15_000 and locks this position into the contract. b) Bob detects that Alice has locked this position into the contract and creates a malicious ERC721 contract, which mints Bob the tokenId 15_000. c) Bob locks tokenId 15_000 and receives id = 100 d) Bob invokes the collectUniswapV3Fees function with id = 100 and the valid uniswapPositionManager address. Due to the fact that id = 100 corresponds to tokenId = 15_000 and there is no validation that Bob's tokenId is in fact corresponding to Uniswap's PositionManager, Bob can now withdraw Alice's fees.
Recommendations	Consider ensuring that the provided _id corresponds to the uniswapPositionManager address: "require lockedNFT.tokenAddress == uniswapPositionManager" Optionally, it might be even more secure to just fetch the

	tokenAddress from the mapping and use this as the uniswapPositionManager address.
Comments / Resolution	Resolved.

Issue_02	Return value of fee0Collected/fee1Collected can be manipulated
Severity	**Low**
Description	The collectUniswapV3LPFees function allows a user to not only claim any accrued LP fees but also idle tokens which have been sitting there due to a “decreaseLiquidity” call. The clue here is that the caller can provide an arbitrary uniswapPositionManager address. While this does not have a direct negative impact, it can be abused to manipulate the return values for fee0Collected and fee1Collected, which can be misleading. It is moreover clear that it is not an option to hardcode the uniswapPositionManager because there can be multiple position managers for other projects which have been forked from UniswapV3.
Recommendations	Consider simply removing these return values.
Comments / Resolution	Resolved.

Issue_03	recoverAssets may revert if “user” has large array of assets
Severity	**Low**
Description	The recoverAssets function loops over all deposits for a specific user and assigns these deposits to the “newRecipient”. In the very rare scenario that a user has an incredibly large amount of deposits (>= 500 or even more), the block gas limit may become insufficient to fulfill the operation.
Recommendations	Consider if this will become an issue in practice. If yes, consider implementing a paginated approach. In our opinion, this issue can be safely acknowledged but should be kept in mind.
Comments / Resolution	Acknowledged.

Appendix: UniswapV3 NonfungiblePositionManager

This contract allows users to create a standard UniswapV3 position, which is wrapped into a NFT. The NonfungiblePositionManager is the owner of the NFT and all privileges are on behalf of this contract. It exposes logic that allows the tokenID holder to access these privileges.

1. mint Function
The mint function is the entry function for creating new liquidity positions within UniswapV3, which are then encapsulated within an NFT. This NFT symbolizes ownership and rights over the liquidity position. Once a position is created, the PositionManager holds the ownership, managing all associated rights.

2. increaseLiquidity Function
This function allows NFT holders to add more liquidity to their existing positions. A unique feature of increaseLiquidity is its flexibility; it permits users to add to positions even if they are not the token holders, even without any permissions.

3. decreaseLiquidity Function
Users can reduce their stake in a liquidity pool using the decreaseLiquidity function. Importantly, it takes into account the accrued fees, ensuring that liquidity providers are compensated for the transaction fees their liquidity has generated. The liquidity which is removed is then converted to tokenA/B and can be collected via the "collect" function.

4. collect Function
The collect function enables liquidity providers to withdraw accrued fees and any residual tokens from their positions.

5. burn Function
The burn function allows users to eliminate the NFT representing their liquidity position, provided there is no remaining liquidity associated with it. It is important to mention that there is a griefing vector for this function: Due to the fact that the liquidity must be zero, a user can simply add a small amount on behalf of this position, which effectively makes the burn call revert.