BAIL security

BAILSEC.IO

EMAIL : OFFICE@BAILSEC.IO

TWITTER : @BAILSECURITY

TELEGRAM : @HELLOATBAILSEC

FINAL REPORT:

Trustswap

LockNFT - PriceEstimator

March 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 SwapToken
Website	trustswap.com
Type	Auxiliary
Language	Solidity
Methods	Manual Analysis
Github repository	https://github.com/trustswap/teamfinance-contract-tokenlock/blob/f8486570d0a2b1e07fbf903c632473d3f7336f29/contracts
Resolution 1	https://github.com/trustswap/teamfinance-contract-locktoken/blob/c4e987e09ab2b165ed1d96c7300d9f39b0a1d34f/contracts/LockToken.sol

2. Detection Overview

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

LockNFT

The LockNFT contract is the NFT contract which is used as property for positions within the LockToken contract. It is using OpenZeppelin's ERC721 contract as base. Whenever users create positions via locking tokens, they can optionally mint a NFT which represents this position. This is done via the mintLiquidityLockNFT function. Furthermore, on any transfer of the NFT, the LockToken.transferLocks function is invoked, which manipulates the storage to remove ownership from the sender and grant it to the new recipient.

It is important to point out that **the owner of the LockNFT contract is ideally the LockToken contract**, this would ensure that the mintLiquidityLockNFT is invoked correctly. Moreover, it would eliminate all governance privileges as long as the LockToken has no functionality to change the owner of the LockNFT contract, which is the case.

The following governance functions are therefore not callable, which greatly increases the decentralization of the protocol:

- setLockTokenAddress
- setNFTDescriptorAddress

BailSec greatly appreciates such an architecture.

No issues found.

PriceEstimator

The PriceEstimator is an oracle-like contract which is used by the LockToken contract in an effort to fetch the current ETH amount for 1 USDT.

It leverages a UniswapV2 pool for that purpose and allows the owner to change the router contract arbitrarily.

Furthermore, it is meant to be used as an implementation contract for a proxy.

Issue	Incorrect setting of uniswapRouter can break the contract functionality
Severity	**Governance**
Description	The owner has the privilege to change the uniswapRouter. This function has two validations: a) The uniswapRout er must be a contract b) The uniswapRouter must not be address(0) While the second condition is already ensured via the first condition, it is notable that the owner can set a non-compatible contract as uniswapRouter which would essentially break the whole functionality.
Recommendations	Consider incorporating a Gnosis Multisignature contract as owner and ensuring that the Gnosis participants are trusted entities.
Comments / Resolution	Acknowledged.

Issue	UniswapV2 pool is inherently manipulatable
Severity	**High**
Description	Currently, the getEstimatedETHforERC20 function is used within the LockToken to fetch the ETH amount for 1 USDT. Anyone can manipulate the corresponding UniswapV2 pool to achieve any desired exchange rate.
Recommendations	Consider simply using a Chainlink Oracle for that purpose. Best practices like staleness checks should be incorporated. This requires a complete refactoring of this contract and thus a small nominal fee to audit the changed logic. Additionally, it is important to note that even if this contract is used as implementation for a proxy, it might make more sense to completely redeploy a new proxy contract after this code has been refactored.
Comments / Resolution	**Partially resolved with comment:** The getFeeInETHWithOracle function aims to return the equivalent ETH value of the provided USD amount. This calculation is done as follows: ((_feesInUSD * 10 ** 18) / uint256(answer)) * 100 For Chainlink, the ETH/USD feed returns the value with 8 decimals, this means that the return value will be 310000000000. If now the provided feesInUSD value is denominated with 6 decimals, as example 3100e6, this will result in the following calculation: (3100e6 * 10 ** 18) / 310000000000 * 100 This calculation is correct.

	BUT, if the provides feesInUSD value is denominated with 18 decimals, this would result in the following calculation: (3100e18 * 10 ** 18) / 310000000000 * 100 which is incorrect. It is now important to understand that the provided USD value is based on the decimals of the USD token in the LockToken contract. **Thus if this contract is deployed on BSC and the corresponding USD token has 18 decimals (such as USDT; USDC), this implementation will not work.** Additionally to mention: It is not always granted that the CL oracle returns the value with 8 decimals. If the answer is denominated with 18 decimals, this will not work. Staleness checks are not incorporated as well, however, this is negligible