Code Security Assessment

Vault Tec

Feb 1st, 2022

Table of Contents

Summary

Overview

- Project Summary
- Audit Summary
- Vulnerability Summary
- Audit Scope

Findings

- CON-01 : Improper usage of `public` and `external` type
- CON-02 : Unbounded Loop
- CON-03 : Purpose of `ingore dust`
- LMM-01 : Centralization Risk
- LMM-02 : Potential Reentrancy Attack
- LMM-03 : The Reward Approval For `poolContract` Should Be Removed As Well In `removePool`
- LMM-04 : `RewardsDistributed` Event Logged ` amount` Would Not Accurate
- LMM-05 : Unchecked Varibale `weight` for the pools
- LMM-06 : Variables that could be declared as `constant`
- TSB-01 : Centralization Risk
- VIE-01 : The Type Of `poolContract` Is Not Decalred as `TimeLockPool`

Appendix

Disclaimer

About

Summary

This report has been prepared for Vault Tec to discover issues and vulnerabilities in the source code of the Vault Tec project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	Vault Tec
Description	Liquidity Mining Manager & TimeLocked Pool
Platform	other
Language	Solidity
Codebase	https://github.com/vault-tec-team/vault-tec-core
Commit	b1c3e0450a39e614b95ec21bf88690bb93172cba a0cd3ec5612da2dcdaa9747eaa49d6ceaae77d42

Audit Summary

Delivery Date	Feb 01, 2022
Audit Methodology	Static Analysis, Manual Review
Key Components	TimeLockPool, LiquidityMiningManager, AbstractRewards, TokenSaver

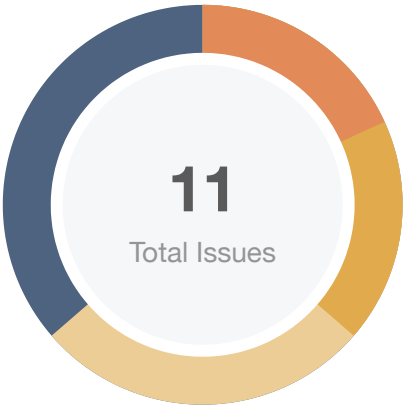
Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Mitigated	Resolved
Critical	0	0	0	0	0	0	0
Major	2	0	0	2	0	0	0
Medium	2	0	0	0	0	0	2
Minor	3	0	0	3	0	0	0
Informational	4	0	0	4	0	0	0
Discussion	0	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
ARB	contracts/base/AbstractRewards.sol	
BPB	contracts/base/BasePool.sol	2d466b469fd6079fdca2305b8716320d460c40e2060a6dc081f76bd0b223eab7
TSB	contracts/base/TokenSaver.sol	4df1f949bfcddf7305dfba1ef6021842105ebd1c793a5c440217af60f69743b2
LMM	contracts/LiquidityMiningManager.sol	32e20dc9c7834eb1abcb11483bc16b9467e6329d27fa4ea0f310bfdc854052c2
TLN	contracts/TimeLockNonTransferablePool.sol	65d66fa08c13c10901a59a4c3c19d9c1075396715491e8b4f48d659bf5dd77c5
TLP	contracts/TimeLockPool.sol	cc36e4786ebc928814354f1cc6d43bac72b48c4d9102678daaf55f9f8dc263c7
VIE	contracts/View.sol	c8c79af8a80a435b466588e5291d154bf31285aba868d30b0bc44326c1bcdaf9

Findings



Critical	0 (0.00%)
Major	2 (18.18%)
Medium	2 (18.18%)
Minor	3 (27.27%)
Informational	4 (36.36%)
Discussion	0 (0.00%)

ID	Title	Category	Severity	Status
CON-01	Improper usage of `public` and `external` type	Gas Optimization	Informational	Acknowledged
CON-02	Unbounded Loop	Logical Issue	Informational	Acknowledged
CON-03	Purpose of `ingore dust`	Magic Numbers	Informational	Acknowledged
LMM-01	Centralization Risk	**Centralization / Privilege**	**Major**	Acknowledged
LMM-02	Potential Reentrancy Attack	Volatile Code	Medium	Resolved
LMM-03	The Reward Approval For `poolContract` Should Be Removed As Well In `removePool`	Logical Issue, Control Flow	Medium	Resolved
LMM-04	`RewardsDistributed` Event Logged `_amount` Would Not Accurate	Inconsistency	Minor	Acknowledged
LMM-05	Unchecked Varibale `weight` for the pools	Inconsistency	Minor	Acknowledged
LMM-06	Variables that could be declared as `constant`	Gas Optimization	Informational	Acknowledged
TSB-01	Centralization Risk	**Centralization / Privilege**	**Major**	Acknowledged
VIE-01	The Type Of `poolContract` Is Not Decalred as `TimeLockPool`	Inconsistency	Minor	Acknowledged

CON-01 | Improper Usage Of `public` And `external` Type

Category	Severity	Location	Status
Gas Optimization	Informational	contracts/TimeLockPool.sol (1): 99~101, 90~97, 103~105 contracts/base/AbstractRewards.sol (1): 47~49	Acknowledged

Description

`public` functions that are never called by the contract could be declared as `external`. `external` functions are more efficient than `public` functions.

Recommendation

Consider using the external attribute for public functions that are never called within the contract.

CON-02 | Unbounded Loop

Category	Severity	Location	Status
Logical Issue	Informational	contracts/View.sol (1): 99, 73 contracts/TimeLockPool.sol (1): 92~94	Acknowledged

Description

The `for` loop takes the following variable `depositsOf[_account].length`, as the maximal iteration times. If the size of the array is very large, it could exceed the gas limit to execute the functions. In this case, the contract might suffer from DoS (Denial of Service) situation.

Recommendation

We recommend to limit the max deposit index to ensure this would not cause loss to the project.

CON-03 | Purpose Of `ingore dust`

Category	Severity	Location	Status
Magic Numbers	Informational	contracts/LiquidityMiningManager.sol (1): 136~139 contracts/base/BasePool.sol (1): 79~82	Acknowledged

Description

When the contract ignore dust, the 1 reward would be kept in the contracts.

Recommendation

We would like to know the purpose of this design.

LMM-01 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	**Major**	contracts/LiquidityMiningManager.sol (1): 103, 90, 74, 53, 110	Acknowledged

Description

In the contract `LiquidityMiningManager`, the role `REWARD_DISTRIBUTOR_ROLE` has the authority over the following function:

- distributeRewards

In the contract `LiquidityMiningManager`, the role `GOV_ROLE` has the authority over the following function:

- addPool
- removePool
- adjustWeight
- setRewardPerSecond

Any compromise to the `REWARD_DISTRIBUTOR_ROLE` & `GOV_ROLE` accounts may allow the hacker to take advantage of this.

Recommendation

We advise the client to carefully manage the `REWARD_DISTRIBUTOR_ROLE` & `GOV_ROLE` accounts' private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the different level in term of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

[Vault Tec Team]:

1. We will be adding a note in the frontend regarding the admin role.

2. After the system is stable, we willreplace the current admin with multi-sig with time-lock

LMM-02 | Potential Reentrancy Attack

Category	Severity	Location	Status
Volatile Code	Medium	contracts/LiquidityMiningManager.sol (1): 110~142	Resolved

Description

Reentrancy in LiquidityMiningManager.distributeRewards() (LiquidityMiningManager.sol#110-142). There are External calls:

- returndata = address(token).functionCall(data,SafeERC20: low-level call failed) (@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol#93)
- (success,returndata) = target.call{value: value}(data) (@openzeppelin/contracts/utils/Address.sol#132)
- reward.safeTransferFrom(rewardSource,address(this),totalRewardAmount) (LiquidityMiningManager.sol#125)
- address(pool.poolContract).call(abi.encodeWithSelector(pool.poolContract.distributeRewards.selector,poolRewardAmount)) (LiquidityMiningManager.sol#131)
- reward.safeTransfer(rewardSource,leftOverReward) (LiquidityMiningManager.sol#138)

If an untrusted pool contract is added into the pools, there would be a risk of a reentrancy attack. These functions which invoke `distributeRewards` would be impacted because they change the statements after `distributeRewards`.

- addPool
- removePool
- adjustWeight
- setRewardPerSecond

Recommendation

We recommend using the Checks-Effects-Interactions Pattern to avoid the risk of calling unknown contracts or applying OpenZeppelin ReentrancyGuard library - `nonReentrant` modifier for the aforementioned functions to prevent reentrancy attack.

Alleviation

[Vault Tec Team]: Fixed in commit: https://github.com/vault-tec-team/vault-tec-core/commit/a0cd3ec5612da2dcdaa9747eaa49d6ceaae77d42

LMM-03 | The Reward Approval For `poolContract` Should Be Removed As Well In `removePool`

Category	Severity	Location	Status
Logical Issue, Control Flow	Medium	contracts/LiquidityMiningManager.sol (1): 75~87	Resolved

Description

As the implementation of `LiquidityMiningManager` contract. The the max amount reeard token approval is set to `poolContract`. The Approval should be removed when contract remove the pool to protect the asset from unexpected transfer operation.

Recommendation

Advise to remove the approval as well in `removePool`, or use the SafeApproval of the exact amount for each reward distribution in implementation.

Alleviation

[Vault Tec Team]: We are adding `reward.safeApprove(poolContract, 0);` to the removePoolfunction to ensure the contract allowance is 0 after pool is removed. https://github.com/vault-tec-team/vault-tec-core/commit/ab44d108e137717d5fb69f56547c696880f6d53a

LMM-04 | `RewardsDistributed` Event Logged `_amount` Would Not Accurate

Category	Severity	Location	Status
Inconsistency	Minor	contracts/LiquidityMiningManager.sol (1): 141	Acknowledged

Description

If there are rewards token amounts left over after the distribution of all pools in the contract address. The emitted log would not accurate. The actual number of total distributed reward amounts should be the contract balance difference between before and after the distribution.

Recommendation

Advise to use the contract balance difference between before and after the distribution as distributed reward amount to emit the event RewardsDistributed.

LMM-05 | Unchecked Varibale `weight` For The Pools

Category	Severity	Location	Status
Inconsistency	Minor	contracts/LiquidityMiningManager.sol (1): 98, 61	Acknowledged

Description

Unchecked Varibale `weight` for the pools. The weight of a pool could be set to 0, which would deprive its rewards in reward distribution.

Recommendation

Advise to validate the wieght of a pool should be greater than 0 when set it.

Alleviation

[Vault Tec Team]: The design is intended. In some cases, we would like to suspend the reward emission for a pool temporarily.

LMM-06 | Variables That Could Be Declared As `constant`

Category	Severity	Location	Status
Gas Optimization	Informational	contracts/LiquidityMiningManager.sol (1): 14	Acknowledged

Description

The linked variables could be declared as `constant` since these state variables are never modified.

Recommendation

We recommend to declare these variables as `constant`.

TSB-01 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	**Major**	contracts/base/TokenSaver.sol (1): 24	ⓘ Acknowledged

Description

In the contract `TokenSaver`, the role `TOKEN_SAVER_ROLE` has the authority over the following function:

- saveToken

Any compromise to the `DEFAULT_ADMIN_ROLE` & `TOKEN_SAVER_ROLE` accounts may allow the hacker to take advantage of this.

Recommendation

We advise the client to carefully manage the `DEFAULT_ADMIN_ROLE` & `TOKEN_SAVER_ROLE` & `TOKEN_SAVER_ROLE` accounts' private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the different level in term of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

[Vault Tec Team]:

1. We will be adding a note in the frontend regarding the admin role.
2. After the system is stable, we willreplace the current admin with multi-sig with time-lock

VIE-01 | The Type Of `poolContract` Is Not Decalred As `TimeLockPool`

Category	Severity	Location	Status
Inconsistency	Minor	contracts/View.sol (1): 57	Acknowledged

Description

Within functino `fetchData`, the pools[i].poolContract is cast to `TimeLockPool`. But the poolContract is decalred as `IBasePool` in the contract `liquidityMiningManager`. Any mismatched type or interface casting would cause failures in the contract execution.

Recommendation

Advise to modify the interface usage to make sure the contract functionality would not blocked by mismatched type cast.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different require statements on the input variables than a setter function.

Magic Numbers

Magic Number findings refer to numeric literals that are expressed in the codebase in their raw format and should otherwise be specified as constant contract variables aiding in their legibility and maintainability.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK's prior written consent in each instance.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS

AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING

MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.