Security Assessment

Convergence Finance

Apr 3rd, 2021

Summary

This report has been prepared for Convergence Finance smart contracts, to discover issues and vulnerabilities in the source code of their Smart Contract as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Dynamic Analysis, 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.

Additionally, this audit is based on a premise that all external smart contracts are implemented safely.

The security assessment resulted in 4 informational findings. 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 given they are currently missing in the repository;
- Provide more comments per each function for readability, especially contracts are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	Convergence Finance
Description	DeFi
Platform	Ethereum
Language	Solidity
Codebase	Convergence-Finance/token-contracts
Commits	ccf7cca086dd28d3742d75cfe209e519888041d5

Audit Summary

Delivery Date	Apr 03, 2021
Audit Methodology	Manual Review
Key Components	

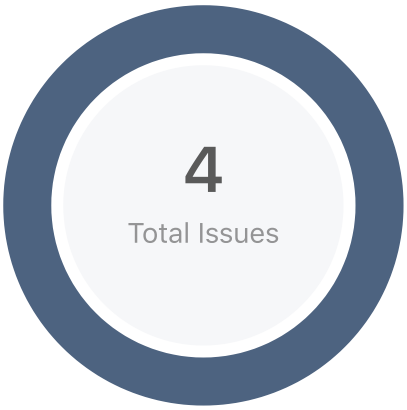
Vulnerability Summary

Total Issues	4
Critical	0
Major	0
Minor	0
Informational	4
Discussion	0

Audit Scope

ID	file	SHA256 Checksum
SPS	contracts/StakingPools.sol	253b061f4d3a9dfb531f4a7afca557aab5c6cdd7a404c6572fd8307ced29289e
ISP	contracts/interfaces/IStakingPoolMigrator.sol	d551d1c8683951b692466e111d1675cb6b13a8d3078145318b8d13cb0c5b4114
ISR	contracts/interfaces/IStakingPoolRewarder.sol	f37e76621ef8fc9d55439ce1031bf9270b1c846a5fe17f4496dc9ef0366835de

Findings



Critical	0 (0.00%)
Major	0 (0.00%)
Minor	0 (0.00%)
Informational	4 (100.00%)
Discussion	0 (0.00%)

ID	Title	Category	Severity	Status
SPS-1	Missing Some Important Checks	Logical Issue	Informational	Declined
SPS-2	Tautology or Contradiction	Logical Issue	Informational	Resolved
SPS-3	Unlocked Compiler Version Declaration	Language Specific	Informational	Resolved
SPS-4	Migrator and Rewarder in StakingPools.sol	Logical Issue	Informational	Resolved

SPS-1 | Missing Some Important Checks

Category	Severity	Location	Status
Logical Issue	Informational	contracts/StakingPools.sol: 126	Declined

Description

There is no validation to check whether the pool exists.

There is no validation to check whether `staker` is zero address.

Recommendation

Consider adding checks for them. For example:

```
    function getReward(uint256 poolId, address staker) external view
onlyPoolExists(poolId) returns (uint256) {
        require(staker  != address(0), "StakingPools: getReward address required");
    ...
}
```

Alleviation

The development team responded that this is an external view function and no need to make any changes.

SPS-2 | Tautology or Contradiction

Category	Severity	Location	Status
Logical Issue	Informational	contracts/StakingPools.sol: 152	Resolved

Description

`_migratorSetterDelay` is a uint256, so `_migratorSetterDelay >= 0` will be always true.

Recommendation

Consider fixing the incorrect comparison by changing the value type or the comparison. For example:

```
require(_migratorSetterDelay > 0, "StakingPools: zero setter delay");
```

Alleviation

The development team heeded our advice and resolved this issue in commit `ccf7cca086dd28d3742d75cfe209e519888041d5`.

SPS-3 | Unlocked Compiler Version Declaration

Category	Severity	Location	Status
Language Specific	Informational	contracts/StakingPools.sol: 2	Resolved

Description

The compiler version utilized throughout the project uses the "^" prefix specifier, denoting that a compiler version which is greater than the version will be used to compile the contracts. Recommend the compiler version should be consistent throughout the codebase.

Recommendation

It is a general practice to instead lock the compiler at a specific version rather than allow a range of compiler versions to be utilized to avoid compiler-specific bugs and be able to identify ones more easily. We recommend locking the compiler at the lowest possible version that supports all the capabilities wished by the codebase. This will ensure that the project utilizes a compiler version that has been in use for the longest time and as such is less likely to contain yet-undiscovered bugs.

Alleviation

The development team responded that they will lock the version by `hardhat config` when deploy.

SPS-4 | Migrator and Rewarder in StakingPools.sol

Category	Severity	Location	Status
Logical Issue	Informational	contracts/StakingPools.sol: 273~274, 314	Resolved

Description

1. What's the strategy of migration? What's the implementation of `migrate()` in `IStakingPoolMigrator`?
2. What's the implementation of `onReward()` in `IStakingPoolRewarder`?

Alleviation

The development team responded that:

1. The migration is to convert the `lp token` from `uniswap` to their `lp token`.
2. Another contract has the responsibility for the distribution rewards.

Appendix

Finding Categories

Gas Optimization

Gas Optimization findings refer to exhibits that 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.

Mathematical Operations

Mathematical Operation exhibits entail findings that relate to mishandling of math formulas, such as overflows, incorrect operations etc.

Logical Issue

Logical Issue findings are exhibits that 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.

Data Flow

Data Flow findings describe faults in the way data is handled at rest and in memory, such as the result of a struct assignment operation affecting an in-memory struct rather than an in storage one.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete .

Coding Style

Coding Style findings usually do not affect the generated byte-code and comment on how to make the codebase more legible and as a result easily maintainable.

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.

Compiler Error

Compiler Error findings refer to an error in the structure of the code that renders it impossible to compile using the specified version of the project.

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 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 without CertiK's prior written consent.

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.

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.