CERTIK

Preliminary Comments

ShieldNetwork Token

Jun 16th, 2021

Table of Contents

Summary

This report has been prepared for ShieldNetwork Token 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 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 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	ShieldNetwork Token
Description	A deflationary yield token, based on SafeMoon
Platform	BSC
Language	Solidity
Codebase	0xf2e00684457de1a3c87361bc4bfe2de92342306c
Commit	

Audit Summary

Delivery Date	Jun 16, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	ShieldNetwork.sol

Vulnerability Summary

Total Issues	6
Critical	0
Major	0
Medium	1
Minor	3
Informational	2
Discussion	0

Audit Scope

ID	file	SHA256 Checksum
SNC	ShieldNetwork.sol	c8b021eb678c14b90d8a61e0deb36aca2c129bdd724ac07149e3c5cfedf74bc8

Findings

Critical	**0** (0.00%)
Major	**0** (0.00%)
Medium	**1** (16.67%)
Minor	**3** (50.00%)
Informational	**2** (33.33%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
SNC-01	Unlocked Compiler Version	Language Specific	Informational	Pending
SNC-02	Functionality Optimization	Gas Optimization	Informational	Pending
SNC-03	Ambiguous Calculation	Logical Issue	Medium	Pending
SNC-04	Usage of `transfer()` for sending Ether	Volatile Code	Minor	Pending
SNC-05	Unchecked Value of ERC-20 `transfer()`/`transferFrom()` Call	Volatile Code	Minor	Pending
SNC-06	Potential Over-centralization of Functionality	**Centralization / Privilege**	**Minor**	**Pending**

SNC-01 | Unlocked Compiler Version

Category	Severity	Location	Status
Language Specific	Informational	ShieldNetwork.sol: 78	Pending

Description

The contract specifies an unlocked compiler version. An unlocked compiler version in the source code of the contract permits the user to compile it at or above a particular version. This, in turn, leads to differences in the generated bytecode between compilations due to differing compiler version numbers. This can lead to an ambiguity when debugging as compiler specific bugs may occur in the codebase that would be hard to identify over a span of multiple compiler versions rather than a specific one.

Recommendation

We advise that the compiler version is instead locked at the lowest version possible that the contract can be compiled at. For example, for version `v0.6.2` the contract should contain the following line:

```
pragma solidity 0.6.2;
```

SNC-02 | Functionality Optimization

Category	Severity	Location	Status
Gas Optimization	Informational	ShieldNetwork.sol: 1139~1141	Pending

Description

The linked conditional will cause the one in L1144 to always yield `true`.

Recommendation

We advise to optimize the linked code blocks.

SNC-03 | Ambiguous Calculation

Category	Severity	Location	Status
Logical Issue	Medium	ShieldNetwork.sol: 1006	Pending

Description

The linked statement will subtract the `rBurnAmount` amount from the `_rTotal`, even in the case where the user does not own enough reflections in L1000.

Recommendation

We advise to revise the linked statements.

SNC-04 | Usage of `transfer()` for sending Ether

Category	Severity	Location	Status
Volatile Code	Minor	ShieldNetwork.sol: 1274	Pending

Description

After EIP-1884 was included in the Istanbul hard fork, it is not recommended to use `.transfer()` or `.send()` for transferring ether as these functions have a hard-coded value for gas costs making them obsolete as they are forwarding a fixed amount of gas, specifically `2300`. This can cause issues in case the linked statements are meant to be able to transfer funds to other contracts instead of EOAs.

Recommendation

We advise that the linked `.transfer()` and `.send()` calls are substituted with the utilization of the `sendValue()` function from the `Address.sol` implementation of OpenZeppelin either by directly importing the library or copying the linked code.

SNC-05 | Unchecked Value of ERC-20 `transfer()`/`transferFrom()` Call

Category	Severity	Location	Status
Volatile Code	Minor	ShieldNetwork.sol: 1269	Pending

Description

The linked `transfer()`/`transferFrom()` invocations do not check the return value of the function call which should yield a `true` result in case of a proper ERC-20 implementation.

Recommendation

As many tokens do not follow the ERC-20 standard faithfully, they may not return a `bool` variable in this function's execution meaning that simply expecting it can cause incompatibility with these types of tokens. Instead, we advise that OpenZeppelin's `SafeERC20.sol` implementation is utilized for interacting with the `transfer()` and `transferFrom()` functions of ERC-20 tokens. The OZ implementation optionally checks for a return value rendering compatible with all ERC-20 token implementations.

SNC-06 | Potential Over-centralization of Functionality

Category	Severity	Location	Status
Centralization / Privilege	Minor	ShieldNetwork.sol: 1266, 1273	Pending

Description

The linked function is meant to be used in an edge-case situation whereby the contract owner can arbitrarily transfer tokens/Ether to any address.

Recommendation

We advise this functionality to be guarded by a time delay to ensure that the normal course of operation of the contract has progressed.

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.

Volatile Code

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

Language Specific

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

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 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.

CERTIK