QuillAudits

Audit Report
June, 2021

Secret Sky .finance

Contents

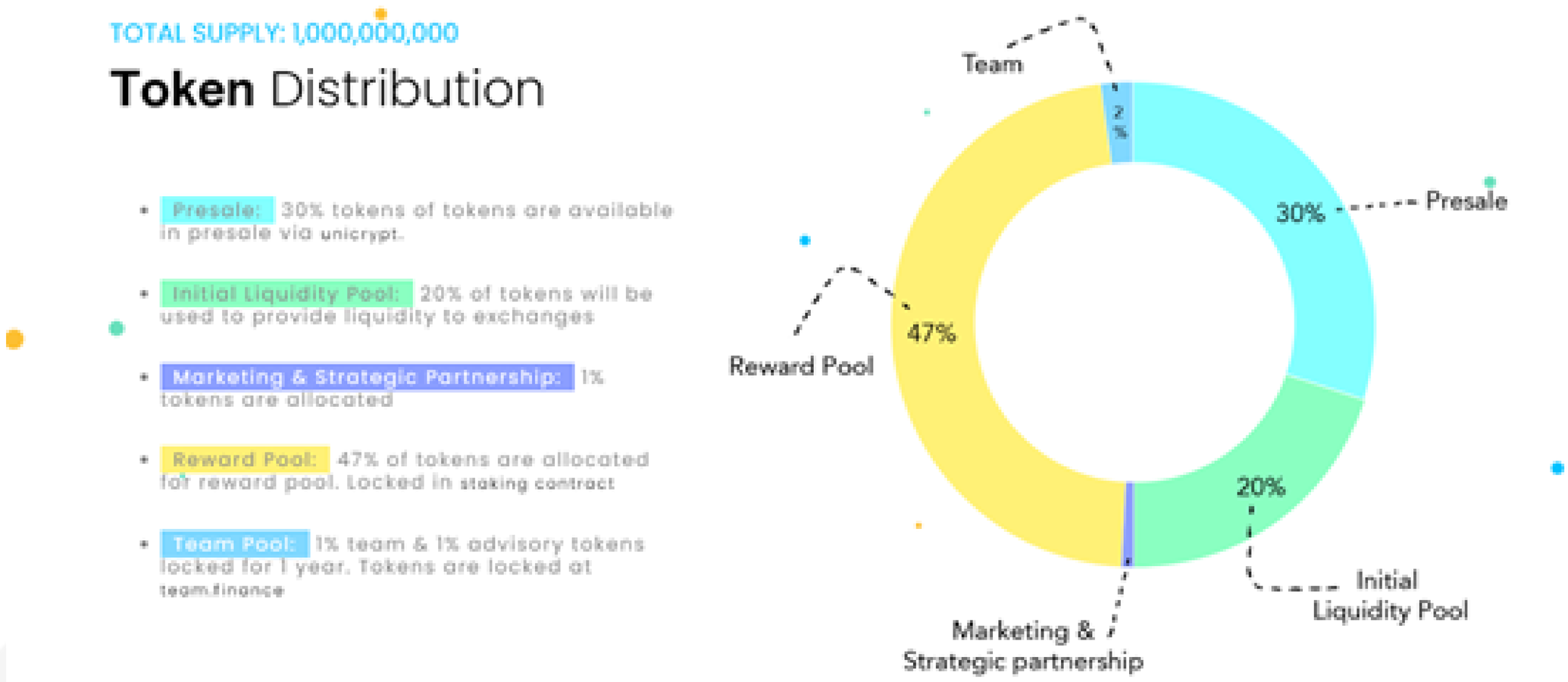
Overview

SSFToken by Secret Sky Finance

A Decentralized Communication platform

Name: SecretSky.finance
Symbol: SSF
Decimals: 18
Total Supply: 1000Million

Contract: SSFToken.sol



Scope of Audit

The scope of this audit was to analyse **SSFToken.sol** smart contract's codebase for quality, security, and correctness.

Checked Vulnerabilities

We have scanned the smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that we considered:

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- Exception Disorder
- Gasless Send
- Use of tx.origin
- Malicious libraries
- Compiler version not fixed
- Address hardcoded
- Divide before multiply
- Integer overflow/underflow
- ERC20 transfer() does not return boolean
- ERC20 approve() race
- Dangerous strict equalities
- Tautology or contradiction
- Return values of low-level
- calls
- Missing Zero Address
- Validation
- Private modifier
- Revert/require functions
- Using block.timestamp
- Multiple Sends
- Using SHA3
- Using suicide
- Using throw
- Using inline assembly

Techniques and Methods

Throughout the audit of smart contract, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

Structural Analysis

In this step we have analyzed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.
SmartCheck.

Static Analysis

Static Analysis of Smart Contracts was done to identify contract vulnerabilities. In this step a series of automated tools are used to test security of smart contracts.

Code Review / Manual Analysis

Manual Analysis or review of code was done to identify new vulnerability or verify the vulnerabilities found during the static analysis. Contracts were completely manually analyzed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of automated analysis were manually verified.

Gas Consumption

In this step we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit

Mythril, Slither, SmartCheck, Surya, Solhint.

Issue Categories

Every issue in this report has been assigned with a severity level. There are four levels of severity and each of them has been explained below.

High severity issues

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality and we recommend these issues to be fixed before moving to a live environment.

Medium level severity issues

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems and they should still be fixed.

Low level severity issues

Low level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are severity four issues which indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Number of issues per severity

	High	Medium	Low	Informational
Open	0	1	6	1
Closed	0	0	0	0

Issues Found – Code Review / Manual Testing

High severity issues

None.

Medium severity issues

1. **Unused owner Functionality**

   The contract implements a powerful role **owner**, but currently, there are no functionalities/features that are specific to this powerful role.

Low Severity Issues

1. [#L182-184] function **transferOwnership()**:
   Missing events to track down the transfer of ownership to the **newOwner**

2. **Missing Zero Address Validation**

   - [#269-278] function **transfer()**:
     Missing Zero Address Check for to address
   - [#L280-288] function **approve()**:
     Missing Zero Address Check for spender address
   - [#L290-296]function **increaseAllowance()**:
     Missing Zero Address Check for spender address
   - [#L298-304]function **decreaseAllowance()**:
     Missing Zero Address Check for spender address
   - [#306-316] function **transferFrom()**:
     Missing Zero Address Check for to and from address

3. **Missing Zero Token Transfer Validation**

   - [#269-278] function **transfer()**: Add check for tokens > 0
   - [#306-316] function **transferFrom()**: Add check for tokens > 0

4. **Reentrancy Issue**

   [#L327-340]As mentioned in the **Mythril** report in the **Automated Tools** section. External call to **spender** address can result in the **reentrancy**. If the **spender** is a **contract**, it can issue a reentrant call to any of this contract's function from its **fallback** function.

5. [#L342]Unused Event LogInt()

6. **approve() race**

   The standard ERC20 implementation contains a widely-known racing condition in its approve function, wherein a spender is able to witness the token owner broadcast a transaction altering their approval and quickly sign and broadcast a transaction using transferFrom to move the current approved amount from the owner's balance to the spender. If the spender's transaction is validated before the owner's, the spender is able to spend their entire approval amount twice.

Reference:

- https://docs.google.com/document/d/1YLPtQxZu1UAvO9cZ1O2RPXBbT0mooh4DYKjA_jp-RLM/edit
- https://medium.com/mycrypto/bad-actors-abusing-erc20-approval-to-steal-your-tokens-c0407b7f7c7c
- https://eips.ethereum.org/EIPS/eip-20

Informational

1. [#L273, 303, 311, 312]Missing Error Statements for **Subtraction Underflow Conditions**

Gas Optimization

Public functions that are never called by the contract should be declared external to save gas.

```
transferOwnership(address) should be declared external:
        - Owned.transferOwnership(address) (SSFToken.sol#182-184)
totalSupply() should be declared external:
        - SSFToken.totalSupply() (SSFToken.sol#256-258)
balanceOf(address) should be declared external:
        - SSFToken.balanceOf(address) (SSFToken.sol#260-267)
transfer(address,uint256) should be declared external:
        - SSFToken.transfer(address,uint256) (SSFToken.sol#269-278)
increaseAllowance(address,uint256) should be declared external:
        - SSFToken.increaseAllowance(address,uint256) (SSFToken.sol#290-296)
decreaseAllowance(address,uint256) should be declared external:
        - SSFToken.decreaseAllowance(address,uint256) (SSFToken.sol#298-304)
transferFrom(address,address,uint256) should be declared external:
        - SSFToken.transferFrom(address,address,uint256) (SSFToken.sol#306-316)
allowance(address,address) should be declared external:
        - SSFToken.allowance(address,address) (SSFToken.sol#318-325)
approveAndCall(address,uint256,bytes) should be declared external:
        - SSFToken.approveAndCall(address,uint256,bytes) (SSFToken.sol#327-340)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

Automated Testing

Slither

Slither didn’t detect any high severity issues

Smartcheck

Smartcheck didn’t detect any high severity issues

Solhint

Solhint didn’t detect any high severity issues

Mythril

```
==== External Call To User-Supplied Address ====
SWC ID: 107
Severity: Low
Contract: SSFToken
Function name: approveAndCall(address,uint256,bytes)
PC address: 4104
Estimated Gas Usage: 9508 - 68801
A call to a user-supplied address is executed.
An external message call to an address specified by the caller is executed. Note that the callee account might contain arbitrary code and could re-enter any function within this
contract. Reentering the contract in an intermediate state may lead to unexpected behaviour. Make sure that no state modifications are executed after this call and/or reentrancy
guards are in place.
--------------------
In file: SSFToken.sol:333

ApproveAndCallFallBack(spender).receiveApproval(
            msg.sender,
            tokens,
            address(this),
            data
        )

--------------------
Initial State:

Account: [CREATOR], balance: 0x1c1080000000902, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x1, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x0, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], calldata: , value: 0x0
Caller: [SOMEGUY], function: approveAndCall(address,uint256,bytes), txdata: 0xcae9ca51000000000000000000000000000000000000000000000000000000000000000300000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001, value: 0x0
Caller: [ATTACKER], function: approveAndCall(address,uint256,bytes), txdata: 0xcae9ca5100000000000000000000000deadbeefdeadbeefdeadbeefdeadbeefdeadbeef00000000000000000000000000000000
0000000000000000000000000000000400000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000000000000000000000000000, value: 0x0
```

Disclaimer

The audit does not give any warranties on the security of the code. One audit cannot be considered enough. We always recommend proceeding with several independent audits and a public bug bounty program to ensure the security of the code. Besides a security audit, please don’t consider this report as investment advice.

Closing Summary

Some issues of medium and low severity have been reported during the audit. Some suggestions have also been made to improve the code quality and gas optimization. There were NO critical or major issues found that can break the intended behaviour.

Secret Sky.finance

QuillAudits

Canada, India, Singapore and United Kingdom

audits.quillhash.com

audits@quillhash.com