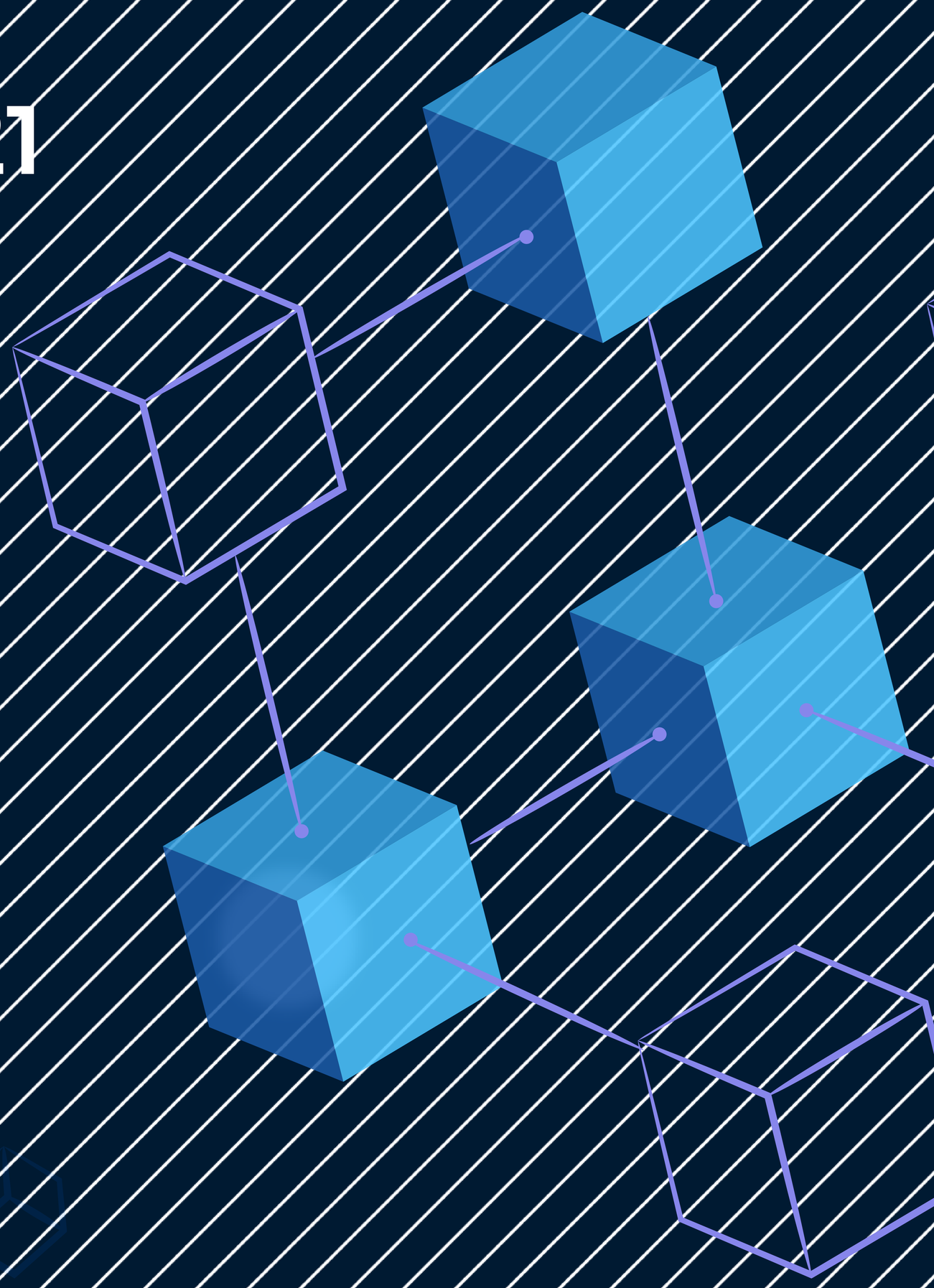
QuillAudits

Audit Report
November, 2021



For

SPLINTERLANDS

Contents

Scope of the Audit

The scope of this audit was to analyze and document the SPS Token smart contract 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
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- Unsafe type inference
- Implicit visibility level

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 BEP-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 analysed 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.

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 the security of smart contracts.

Code Review / Manual Analysis
Manual analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analysed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the 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 the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit
Remix IDE, Solhint, Mythril, Slither.

Issue Categories

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

Risk-level	Description
High	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 be fixed before moving to a live environment.
Medium	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	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 issues that 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

Type	High	Medium	Low	Informational
Open	0	0	0	0
Acknowledged	0	0	1	2
Closed	0	0	0	0

Introduction

During the period of **November 9, 2021 to November 16 , 2021** - QuillAudits Team performed a security audit for the **SPS** smart contract.

The code for the SPS contract was obtained from:
- https://github.com/steem-monsters/sps-dao/blob/main/SPS.sol

The contract was deployed and tested on Ropsten and you can find it here:
- SPS: 0x45686cC6a199580A6efc609291dca54Ab72f2E91

Issues Found

A. Contract – SPS.sol

High severity issues

No issues were found.

Medium severity issues

No issues were found.

Low severity issues

1. **Pragma Version not Locked**

   Description
   The pragma version is not locked.

   Remediation
   It is a good practice to lock the solidity version for a live deployment (use **0.5.16** instead of **^0.5.16**). Contracts should be deployed with the same compiler version and flags that they have been tested the most with. Locking the pragma helps ensure that contracts do not accidentally get deployed using, for example, the latest compiler which may have higher risks of undiscovered bugs. Contracts may also be deployed by others and the pragma indicates the compiler version intended by the original authors.

   Status: Acknowledged

Informational issues

2. Events

Description
Events in the **bridgeTransfer** and the **bridgeTransferFrom** functions are emitted at the beginning of the function.

Remediation
It is a good practice to emit events towards the end of the function.

Status: Acknowledged

3. Indentation

Description
The indentation is incorrect on the following lines:
- 103-108
- 320-329
- 414-429

Suggestion
Fix the indentation according to solidity's guideline.

Status: Acknowledged

Functional test

Function Names	Testing results
transfer	Passed
transferFrom	Passed
mint	Passed
approve	Passed
bridgeTransfer	Passed
bridgeTransferFrom	Passed
delegate	Passed
delegateBySig	Passed
setAdmin	Passed
setMinter	Passed
setStakeModifier	Passed

Functionality Tests Performed

SPS.sol

- Users should be able to transfer tokens not more than their balance. **PASS**
- approve. **PASS**
- Users should be able to bridgeTransfer tokens not more than their balance. **PASS**
- Users should be able to bridgeTransferFrom tokens not more than their approval and also not more than the owner’s(token owner not the contract owner) balance. **PASS**
- Users should be able to transferFrom tokens not more than their approval and also not more than the owner’s(token owner not the contract owner) balance. **PASS**
- Only the minter should be able to call the mint function and mint tokens to their own account. **PASS**
- Users should be able to delegate their votes without tokens actually being transferred to the delegatee. **PASS**
- Only the admin should be able to setAdmin. **PASS**
- Only the admin should be able to setMinter. **PASS**
- Only the admin should be able to setStakeModifier. **PASS**

Automated Tests

Slither

Slither, an open-source static analysis framework. This tool provides rich information about Ethereum smart contracts and has critical properties. While Slither is built as a security-oriented static analysis framework, it is also used to enhance the user's understanding of smart contracts, assist in code reviews, and detect missing optimizations. Slither detected the following issues:

```
INFO:Detectors:
SPS._writeCheckpoint(address,uint32,uint96,uint96) (dist/SPSDAO.sol#321-332) uses a dangerous strict equality:
        - nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber (dist/SPSDAO.sol#324)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
SPS.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (dist/SPSDAO.sol#205-214) uses timestamp for comparisons
       Dangerous comparisons:
       - require(bool,string)(now <= expiry,SPS::delegateBySig: signature expired) (dist/SPSDAO.sol#212)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
SPS.getChainId() (dist/SPSDAO.sol#355-359) uses assembly
       - INLINE ASM (dist/SPSDAO.sol#357)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
delegate(address) should be declared external:
       - SPS.delegate(address) (dist/SPSDAO.sol#192-194)
delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) should be declared external:
       - SPS.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (dist/SPSDAO.sol#205-214)
getPriorVotes(address,uint256) should be declared external:
       - SPS.getPriorVotes(address,uint256) (dist/SPSDAO.sol#234-268)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:dist/SPSDAO.sol analyzed (2 contracts with 46 detectors), 6 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Mythril

Mythril is a security analysis tool for EVM bytecode. It detects security vulnerabilities in smart contracts built for Ethereum, Hedera, Quorum, Vechain, Roostock, Tron and other EVM-compatible blockchains. It uses symbolic execution, SMT solving and taint analysis to detect a variety of security vulnerabilities. Mythril raised the following concerns:

```
The analysis was completed successfully. No issues were detected.
```

Contract Library

Contract-library contains the most complete, high-level decompiled representation of all Ethereum smart contracts, with security analysis applied to these in real-time. We performed analysis using the contract Library on the Ropsten address of the SPS contract used during manual testing:

- SPSDAO: 0x45686cC6a199580A6efc609291dca54Ab72f2E91

Results

No major issues were found. All the other issues have been categorized above according to their level of severity.

Closing Summary

In this report, we have considered the security of SPS. We performed our audit according to the procedure described above.

The audit showed one low level severity issue which has been Acknowledged by the Client.

Disclaimer

Quillhash audit is not a security warranty, investment advice, or an endorsement of the **SPS** platform. This audit does not provide a security or correctness guarantee of the audited smart contracts. The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the **SPS** Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

Audit Report
November, 2021

For

SPLINTERLANDS

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com