Swiss Finance Governance Audit

The BlockChain Auditor

Prepared by: Jorge Martinez

Version: 2
Date: Feb. 1, 2020

Summary of Findings

This document expresses all security concerns of the Swiss Finance Smart Contracts as expressed by Jorge Martinez. I took care to attempt to find as many ways to improve the security, code efficiency, best practices, and overall function of the smart contracts.

Contract Status: Ready for Deployment

5 Critical Issues found in the governance.sol were found.
0 Medium Issues were found.
4 Low Issues in the governance.sol were mitigated.
0 Informational Issues were found.

Solidity Code Coverage

Jorge's Test Suite	Swiss Finance	Industry Standard
100%	0%	95%

For this audit, I wasn't provided with a testing suite but as part of my audit methodology I developed a test suite to verify the functionality of the governance contract, check its security, and to help reveal any underlying issues.

This audit should be seen as one step in the development process with the intent of raising awareness on the meticulous work involved in secure development and making no material statements or guarantees to the operational state of the smart contract(s) once they are deployed. This document is not an endorsement of the reliability or effectiveness of the smart contracts. This is an assessment of the smart contract logic, implementation, and best practices. I cannot take responsibility for any potential consequences of the deployment or use of the smart contract(s) related to the audit.

Test Suite Results

Jorge’s Test Suite

```
Swiss Governance Security Testing
  Swiss Token
    Deployment
      ✓ should set the erc20 details (4191ms)
      ✓ should have 18 decimals
      ✓ deployer should have 9435 tokens
  Governance
    Deployment
      ✓ should be the owner of Swiss Token
      ✓ governance owner should be able to change the swiss fee wallet (144ms)
      ✓ governance owner should be able to change the desh fee wallet (128ms)
    Voting
      ✓ should let addresses with more than 10 tokens vote (398ms)
      ✓ should not let addresses with less than 10 tokens vote (144ms)
      ✓ should let addresses with 100 tokens or more make proposals (106ms)
      ✓ should not let addresses with less than 100 tokens make proposals (92ms)
      ✓ an address should not be able to vote yes and no on any given proposal (273ms)
      ✓ addresses should be able to add and remove vote during the voting phase (329ms)
      ✓ cannot remove or add 0 votes (201ms)
      ✓ a single address should not be able to pass a proposal (192ms)
    Malicious Contract Attacks
      ✓ should not let contracts interact with the governance contract (44ms)
    Pausability
      ✓ only the owner should be able to pause the contract (44ms)
      ✓ only the owner should be able to unpause the contract (69ms)
      ✓ when governance is paused there can be no new proposals (85ms)
      ✓ when governance is paused proposals cannot be executed (612ms)
      ✓ when governance is paused votes cannot be added or removed (617ms)
      ✓ when governance is paused votes can be removed from proposals up for vote (560ms)
      ✓ when governance is paused votes can be removed from expired proposals (452ms)
    Proposal Functions
      ✓ text proposal with a yes majority with quorum should not execute (418ms)
      ✓ text proposal with a yes majority without quorum should not execute (369ms)
      ✓ text proposal with a no majority with quorum should not execute (512ms)
      ✓ text proposal with a no majority without quorum should not execute (442ms)
      ✓ new swiss fee that is greater than 10000 is reverted
      ✓ new swiss fee with yes majority with quorum should execute (370ms)
      ✓ new swiss fee with yes majority without quorum should not execute (431ms)
      ✓ new swiss fee with no majority with quorum should not execute (571ms)
      ✓ new swiss fee with no majority without quorum should not execute (603ms)
      ✓ new desh fee with yes majority with quorum should execute (446ms)
      ✓ new desh fee with yes majority without quorum should not execute (505ms)
      ✓ new desh fee with no majority with quorum should execute (529ms)
      ✓ new desh fee with no majority without quorum should not execute (564ms)
      ✓ only the governance owner should be able to propose a governance upgrade (65ms)
      ✓ new governance contract with yes majority with quorum should pass (491ms)
      ✓ new governance contract with yes majority without quorum should not pass (427ms)
      ✓ new governance contract with no majority with quorum should not pass (497ms)
      ✓ new governance contract with no majority without quorum should not pass (460ms)
      ✓ only the governance owner should be able to propose a new quorum (84ms)
      ✓ new quorum with yes majority with quorum should pass (363ms)
      ✓ new quorum with yes majority without quorum should not pass (391ms)
      ✓ new quorum with no majority with quorum should not pass (559ms)
      ✓ new quorum with no majority without quorum should not pass (483ms)

-------------------------|----------|----------|----------|----------|----------------|
File                     |  % Stmts | % Branch |  % Funcs |  % Lines |Uncovered Lines |
-------------------------|----------|----------|----------|----------|----------------|
 contracts/              |    64.29 |    49.32 |    54.84 |    63.35 |                |
  MaliciousContract.sol  |      100 |      100 |      100 |      100 |                |
  governance.sol         |    73.21 |    58.65 |    53.33 |    71.86 |... 714,720,726 |
  token.sol              |       50 |    27.27 |    54.35 |       50 |... 948,954,957 |
-------------------------|----------|----------|----------|----------|----------------|
All files                |    64.29 |    49.32 |    54.84 |    63.35 |                |
-------------------------|----------|----------|----------|----------|----------------|
```

* 100% of relevant code tested

Table of Contents

Audit Methodology & Techniques

The BlockChain Auditor has the following auditing process:

1. Our audits include
   a. Review of the specifications, source code, and instructions provided to the TheBlockchainAuditor to clearly identify the desired functionality of the smart contract(s).
   b. Manual line by line review of contract code to spot potential vulnerabilities.
   c. Identification of deviations between desired functionality expressed to the TheBlockchainAuditor and what the smart contract(s) are doing.
2. Automated static and symbolic analysis, as well as verifying testing coverage using the provided test suite.
   a. Automated static and symbolic analysis help determine what inputs cause each part of the smart contract to execute. Analysis of how much of the code base is tested and comparison to industry standard.
3. Examination of smart contracts and development process as a whole, ensuring best practices are followed, allowing improved efficiency and security based on established industry and academic practices.
4. Specific, itemized, and actionable recommendations to assist in securing the smart contract(s) in question.

Contract Checklist

governance.sol

Contract Vulnerability	
Integer Overflow	Pass
Race Condition	Pass
Denial of Service	Pass
Logical Vulnerability	Pass
Hardcoded Address	Pass
Function Input Parameter Check	Pass
Function Access Control Check	Pass
Random Number Generation	N/A
Random Number Use	N/A
Contract Specification	
Solidity Compiler Version	Pass
Event Use	Pass
Fallback Function Use	Pass
Constructor Use	Pass
Function Visibility Declaration	Pass
Variable Storage Declaration	Pass
Deprecated Keyword Use	Pass
ERC20/223 Standard	N/A
ERC721 Standard	N/A
Business Risk	
Able to Arbitratrily Create Token	N/A
Able to Arbitrarily Destroy Token	N/A
Can Suspend Transactions	Pass
Short Address Attack	Pass
Gas Optimization	
assert()/require()/revert() misused	Pass
Loop Optimization	Pass
Storage Optimization	Pass

Issue Classification

Critical

These issues in the smart contract can have catastrophic implications that could ruin your reputation, disrupt the contract's functionality, or impact the client and your users' sensitive information.

Informational

This issue relates to style and security best practices but does not pose an immediate risk.

Medium

An issue classified as medium has relatively small risk and isn't exploitable to circumvent desired functionality and could not have financial consequences but could put user's sensitive information at risk.

Acknowledged

The issue remains in the code but is a result of an intentional business or design decision.

Low

An issue classified as informational does not pose an immediate threat to disruption of functionality and could not be exploited on a recurring basis, however, it should be considered for security best practices or code integrity.

Unresolved

Although the client has been informed of the risk, it was decided to accept it because it was not relevant in the functionality of the smart contract.

Undetermined

The impact of the issue is uncertain and more investigation is required to understand the repercussions of the issue.

Mitigated

Actions were taken to minimize the impact or likelihood of the risk.

Detailed Analysis

5.1 executeProposal()

governance.sol

A single user can pass a proposal

Explanation:

Upon developing the testing suite it became apparent that a single user is able to initialize a proposal, reach quorum, wait five minutes, and execute the proposal.You can see this is one of the test cases that failed.

Recommendation:

I recommend you make it such that a single individual is unable to pass a proposal by themselves by requiring a minimum amount of different addresses or a minimum percentage of token holders to participate in the voting process.

5.2 freeze()

governance.sol

Unable to Freeze Governance

Explanation:

If for whatever reason the governance is exploited and your investor’s tokens are at risk there is no freeze function to stop voting.

Recommendation:

I recommend that a freeze function is implemented that would not allow votes or proposals to be made. There of course would have to be an unfreeze function to resume governance if you were to upgrade to a more secure governance or simply protect your investors.

5.3 proposeNewSwissFee()

governance.sol

No upper limit on Swiss fee proposal

Explanation:

Since there is no upper limit users could set the fee to be 10,000 or higher and as a result the fee could be higher than the amount of tokens being transferred.

Recommendation:

Add a require statement to proposeNewSwissFee() that forces the proposed fee to be within what you believe to be a reasonable interval that will not hurt the Swiss protocol.

5.4 proposeNewDeshFee()

governance.sol

No upper limit on Decash fee proposal

Explanation:

Since there is no upper limit users could set the fee to be 10,000 or higher and as a result the fee could be higher than the amount of tokens being transferred.

Recommendation:

Add a require statement to proposeNewDeshFee() that forces the proposed fee to be within what you believe to be a reasonable interval that will not hurt the Decash protocol.

5.5 executeProposal()

governance.sol

Multiple Proposals Simulateneously

Explanation:

As coded, executeProposal() allows multiple proposals to be in vote simultaneously and combined with the fact that a single user can pass a proposal by themselves this allows a user with a large enough balance to implement an unfavorable change without many users knowing about it until its too late. Users could also have their funds locked in a vote while another more pressing vote is going on and be unable to participate.

Recommendation:

This is a more difficult problem to solve. One possible solution would be to only allow a single proposal to be active at a time and keep a backlog of proposals such that they are voted upon sequentially. Another possibility is that you can give users the ability to vote on multiple proposals with up to the number of tokens locked.

5.6 isProposalOpen()

governance.sol

Unclear Function Naming

Explanation:

The isProposalOpen() function is checking if the proposal is in voting but returns false during the RESULT_EXECUTION_ALLOWANCE_PERIOD.

Recommendation:

I recommend it to be renamed to isProposalInVoting() or something that more clearly reflects its purpose.

5.7 isProposalExecutable() governance.sol

Unclear Reverts in Function

Explanation:

There is only one way that isProposalExecutable() can succeed but there are two ways that it can fail. It succesfully returns false if the proposal is not executable but it doesn't differentiate between it not being executable because the proposal is still in the voting phase or if it has gone beyond the RESULT_EXECUTION_ALLOWANCE_PERIOD.

Recommendation:

Handle both failure cases seperately so that it can be more clear why it is failing during testing.

5.8 isProposalExecutable() governance.sol

Repeated checks

Explanation:

isProposalExecutable() checks if the votes are tied but executeProposal() is already checking for a tie immediately before it calls isProposalExecutable().

Recommendation:

I recommend removing the check from isProposalExecutable() as it will simplify isProposalExecutable() a little.

5.9 minBalanceToInitProposal

governance.sol

Code Integrity

Explanation:

According to the whitepaper, the minimum Swiss required to initialize the proposal is 100 tokens but according to the code the minimum is actually 50 tokens.

Recommendation:

To ensure code integrity either the whitepaper needs to be updated or the MIN_BALANCE_TO_INIT_PROPOSAL set to 100 tokens.

Notable Concerns & Suggestions

Overall Thoughts

The team took the necessary steps to correct the issues I found in their governance contract. I modifed my test suite to take the changes into consideration and found that the governance contract functioned as expected. I feel that steps were taking to ensure the safety the user/investor of this contract and that this governance contract does not put the Swiss Decash ecosystem at risk. It allows users to make proposals and handles access control well. It defends against malicious contracts. Now proposals can only happen sequentially which helps protect investors from sneaky proposals while their tokens would have been otherwise encumbered.

I now believe that this governance contract is finalized and okay for deployment.

Appendix A

File Fingerprints

governance.sol	27ff2b87db005d32a745d1f6d98687ae

The Blockchain Auditor is honored to have the opportunity to be partnered with Swiss Finance and help provide enhanced security and efficiency for the Swiss Finance platform.

The Swiss Finance Governance protocol is great demonstration of the power of decentralization and we are looking forward to seeing what else the team that developed DeCash will develop in the future.

The BlockChain Auditor

- Jorge Martinez