COINMERCENARY / SMART CONTRACT AUDIT

Securrency Platform

5 NOVEMBER 2018 / TABLE OF CONTENTS

INTRODUCTION

CoinMercenary provides comprehensive, independent smart contract auditing.

We help stakeholders confirm the quality and security of their smart contracts using our comprehensive and standardized audit process. Each audit is unbiased and verified by multiple reputable auditors.

The scope of this audit was to analyze and document the Securrency platform contracts.

This audit provides practical assurance of the logic and implementation of the contracts.

AUDIT METHODOLOGY

CoinMercenary audits consist of four categories of analysis.

Design Patterns

We first inspect the overall structure of the smart contract, including both manual and automated analysis.

The design pattern analysis checks appropriate test coverage, utilizes a linter to ensure consistent style and composition, and code comments are reviewed. Overall architecture and safe usage of third party smart contracts are checked to ensure the contract is structured in a way that will not result in future issues.

Static Analysis

The static analysis portion of our audit is performed using a series of automated tools, purposefully designed to test the security of the contract. These tools include:

- **Manticore** - Dynamic binary analysis tool with EVM support.
- **Mythril** - Reversing and bug hunting framework for the Ethereum blockchain.
- **Oyente** - Analyzes Solidity code to find common vulnerabilities.
- **Solgraph** - DOT graph creation for visualizing function control flow of a Solidity contract to highlight potential security vulnerabilities.

Data flow and control flow are also analyzed to identify vulnerabilities.

Manual Analysis

Performing a hands on review of the smart contract to identify common vulnerabilities is the most intensive portion of our audit. Checks for race conditions, transaction-ordering dependence, timestamp dependence, and denial of service attacks are part of our standardized process.

Network Behavior

In addition to our design pattern check, we also specifically look at network behavior. We model how the smart contract will operate once in production,

then determine the answers to questions such as: how much gas will be used, are there any optimizations, how will the contract interact?

Contracts Reviewed

On October 31st, 2018 using git hash a23435e1387a76571748f4587074a8dd2dfac5ff from the repository located at:

https://github.com/Securrency/EthereumSmartContracts

Each contract file located in the contracts/* directory were reviewed. In addition to the contract files, unit tests and the setup process were reviewed as well.

Remediation Audit #1

On November 19th, 2018 using git hash 2f216c0d33dfa0615cb71fc80c161224b2a7425d from the repository located at:

https://github.com/Securrency/EthereumSmartContracts

Each contract file located in the contracts/* directory were reviewed. In addition to the contract files, unit tests and the setup process were reviewed as well.

AUDIT SUMMARY

The contracts have been found to be free of security issues.

Analysis Results

	Initial Audit	**Remediation Audit**
Design Patterns	Updates Recommended	Passed
Static Analysis	Updates Recommended	Passed
Manual Analysis	Updates Recommended	Passed
Token Allocation	Passed	Passed
Network Behavior	Passed	Passed

Test Results

- Unit test coverage available. 211 passing tests, 0 failing tests.

Explicit Vulnerability Check Results

Known Vulnerability	**Results**
Parity Multisig Bug 2	Not vulnerable
Callstack Depth Attack	Not vulnerable
Transaction-Ordering Dependence	Not vulnerable
Timestamp Dependency	Not vulnerable
Re-Entrancy Vulnerability	Not vulnerable
Proxy and Buffer Overflow	Not vulnerable

ISSUES DISCOVERED

Issues below are listed from most critical to least critical. Severity is determined by an assessment of the risk of exploitation or otherwise unsafe behavior.

Severity Levels

- **Informational** - No impact on the contract.
- **Low** - Minimal impact on operational ability.
- **Medium** - Affects the ability of the contract to operate.
- **High** - Affects the ability of the contract to work as designed in a significant way.
- **Critical** - Funds may be allocated incorrectly, lost or otherwise result in a significant loss.

Issues

SC-1 / Critical: transferOwnership should be a two-step process and require acceptance from the new owner.

Present in SymbolRegistry.sol, line 117
Present in NetworkRolesManager.sol, line 59

Explanation

Transferring ownership is a common pattern in smart contracts and can be implemented in many different ways. Due to the extreme unforgiving nature of smart contracts, and the inability to easily rollback ownership changes, CoinMercenary recommends implementing a two-step process for transferring ownership of contracts.

Part 1 of the two-step process is the current owner initiating the transfer to the newOwner address.

Part 2 of the two-step process is for the newOwner to accept the transfer of ownership, which then completes the transfer.

Resolution

Resolved in 184bf1aa01e393fee9ada6d01416d3a09dd674eb.

SC-2 / Medium: verifySymbol modifier requires symbols between 1 and 5, not 1 and 6 characters long

Present in SymbolRegistry.sol, line 29

Explanation

The verifySymbol modifier logic requires a symbol to be greater than 0 or less than 6 characters, but the validation comment specifies the symbol can be between 0 and 6 characters.

The validation comment should be updated to reflect symbols between 1 and 5 characters long.

Resolution

Resolved in 423f2bd57a091d22b1ed9242b4880da1dab5e86e.

SC-3 / Low: Use latest Solidity compiler version

Present in all contract files

Explanation

Update all contract files to use the latest version of Solidity compiler in order to ensure the latest performance enhancements, features and bug fixes are available.

Resolution

Resolved in 4322935a7b97f1c09c17ae86814decceef06e2f2.

SC-4 / Informational: exprationInterval naming typo, and unit contents of variable not documented

Present in SRStorage.sol, line 12

Explanation

exprationInterval should be expirationInterval, and the comment above the declared variable should note the storage units (seconds? minutes?) for clarity.

Resolution

Resolved in 47e09d23f5b166fea08d4e9a43d52fe83ff74c0d.

SC-5 / Informational: Parameters in functions not documented in declaration comment above function

Present in IMultiChainToken.sol, line 25
Present in ISecuritiesToken.sol, line 21
Present in SecuritiesNFT.sol, line 27
Present in SecuritiesStandardToken.sol, line 67
PResent in SecuritiesStandardToken.sol, line 113
Present in SecuritiesToken.sol, line 45, 80
Present in CAT20Verification.sol, line 30
Present in CAT721Verification.sol, line 30
Present in FCStorage.sol, line 25, 48
Present in TCStorage.sol, line 27, 54
Present in IFCStorage.sol, line 19, 21,
Present in IToChain.sol, line 25, 36,
Present in ITransferModule.sol, line 25, 54

Explanation

A variable is present in the function declaration that is not documented in the comment prior to the function declaration. For consistency purposes, each variable and its purpose should be declared.

Resolution

Resolved in a6d0a73f8ef57f05884a7419588f8a8d0d333a7c.

SC-6 / Informational: Declared parameters do not match order of parameters for Event

Present in WhiteList.sol, line 20, 27

Explanation

The comment above the function specifies a different order of parameters.

Resolution

Resolved in a52a7e55f79f2d4afb9be183bebb32d1d0af44a7.

SC-7 / Informational: Various typos in variable names and functions

Present in PMTokenRolesStorage.sol, lines 14, 19, 27, 35, 45, 53
Present in IPMTokenRolesStorage.sol, lines 14, 19
Present in PMRolesManagerStorage.sol, line 250
Present in IPMRolesManagerStorage.sol, line 170
Present in RolesManager.sol, lines 120, 149
Present in TransferModule.sol, lines 30, 70, 153

Explanation

Typos in function and/or variable names that should be corrected prior to deployment.

Resolution

Resolved in 2f216c0d33dfa0615cb71fc80c161224b2a7425d.

AUDIT CONCLUSION

November 21st, 2018

CoinMercenary provides comprehensive, independent smart contract auditing.

We help stakeholders confirm the quality and security of their smart contracts using our comprehensive and standardized audit process. Each audit is unbiased and verified by multiple reputable auditors.

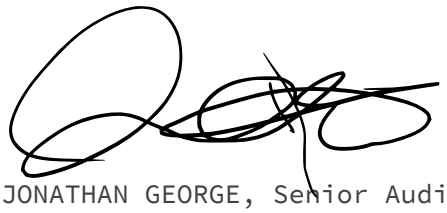
The scope of this audit was to analyze and document the Securrency platform contracts. The audit provides practical assurance of the logic and implementation of the contracts.

CoinMercenary has reviewed the Securrency platform contracts and found them to be free of security issues and logic errors.

The audit began on October 31st, ending on November 21st. One critical issue, one medium, one low and 4 informational issues were documented.

Working with the Securrency team has been a pleasure and we look forward to seeing their continued success.

Sincerely,



JONATHAN GEORGE, Senior Auditor