COINMERCENARY / SMART CONTRACT AUDIT

# Securrency Platform

## 14 FEBRUARY 2019 / 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 February 7th, 2019 using git hash 1bbb5c620668c54d215cdd961c60fa4cd28c5b03, the contracts from the repository located below were reviewed.

https://github.com/Securrency/EthereumSmartContracts

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** |
|---|---|
| **Design Patterns** | Passed |
| **Static Analysis** | Passed |
| **Manual Analysis** | Updates Recommended |
| **Token Allocation** | Passed |
| **Network Behavior** | Passed |

### Test Results

- Unit test coverage available.

### 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 / High: pushNewCATApp should use the onlyApplicationRegistry modifier

Present in ACATRStorage.sol, line 98

Explanation

The pushNewCATApp function is open for anyone to call. This function should be restricted to onlyApplicationRegistry(msg.sender).

Resolution

Not yet resolved.

---

#### SC-2 / Low: Function named isRegistredApp should be named isRegisteredApp

Present in ApplicationRegister.sol, line 180

Explanation

The function named isRegistredApp contains a typo, and should be “isRegisteredApp”.

Resolution

Not yet resolved.

---

### SC-3 / Low: emitCreatedRole should use onlyPermissionModule modifier

Present in PMRolesManagerStorage.sol, line 113

Explanation

The emitCreatedRole function is open for anyone to call. This function should be restricted to onlyPermissionModule(msg.sender).

Resolution

Not yet resolved.

---


# SECURRENCY AUDIT CONCLUSION

February 14th, 2019

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 February 7th, 2019, ending on (NOT YET COMPLETED). One ‘high’ issue and two ‘low’ 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