COINMERCENARY / SMART CONTRACT AUDIT

SmarterThanCrypto (STC)

27 FEBRUARY 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 SmarterThanCrypto (STC) token generation event contract.

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

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 27, 2018 using git hash b059d164af43897c73254f98845c8149b61f7d91, the following contract files and their respective SHA256 fingerprints were reviewed:

Filename	SHA256 Fingerprint
STC.sol	c98358003a488a8f3c6c3e3fed371db33914e8680266f53b30f8335097df118b
STCVesting.sol	c41c53860165ef2016296957737a45cafc32ceb2e1f8491a4aebc414190169b9
SafeMath.sol	de1e89d77c58eb83018b44faa578799d98092901344511962dfff416b18ffc3b
StandardToken.sol	7a474a33653ba66ad1701868aca7e1a51856a40787b5fd0f129b83e6f7796283
Token.sol	b054a9acb46ea2bb9d5d58408a0a53ed0364b6a4d61e437d8ca2b839a256747d

Remediation Audit

On March 12, 2018 using git hash 5c3c0443cde24b3389eeeddd9a90135a609ae119, the following updated contract files and their respective SHA256 fingerprints were reviewed:

Filename	SHA256 Fingerprint
STC.sol	1abe2451d4f5736255b264410455e4830ef3ed050930d72b299750dbab3181ef
STCVesting.sol	693d70a3c58910464ba9c70df0bdc37fda552bdfad036e22df578e65dcba9fbc
SafeMath.sol	c8ffe7f97deefea189edd915b096ea797d361618a2df957a75c5a9a8ee75488c
StandardToken.sol	b5d2a3e709e8fa8e6b353ab3d888208c47cf2ebf96fa13d3120358292cde89ce
Token.sol	751b542f840558b265edeae08248abfc38ea86326d6252e8ca656583809a630b

AUDIT SUMMARY

The contracts have been found to be free of security issues. The SmarterThanCrypto token is a compliant ERC-20 token.

Analysis Results

	Initial Audit	Remediation Audit
Design Patterns	Passed	Passed
Static Analysis	Failed	Passed
Manual Analysis	Failed	Passed
Token Allocation	Passed	Passed
Network Behavior	Passed	Passed

Test Results

No test coverage available for contracts. CoinMercenary recommends extensive test coverage.

Token Allocation Results

Expense Allocation: 4.5%
Team Allocation: 8.5% (vested over 2 years)

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

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

STC-2 / Medium: Compiler version not fixed

Present in StandardToken.sol
Present in Token.sol
Present in STCVesting.sol
Present in STC.sol

Explanation

Solidity source files indicate the versions of the compiler they can be compiled with. It is recommended to specify the compile version the contract was built with, since future compiler versions may handle certain language constructions in a way the developer did not foresee.

Resolution

Resolved in 5c3c0443cde24b3389eeeddd9a90135a609ae119

STC-3 / Informational: Use ‘view’ rather than ‘constant’ for functions

Present in StandardToken.sol, lines: L23, L42
Present in Token.sol, lines: L3, L7
Present in STCVesting.sol, lines: L57, L27
Present in STC.sol, lines: L251, L189, L100, L244

Explanation

The function is declared as constant. Currently, for functions the constant modifier is a synonym for view, which is the preferred option. Consider using view for functions and constant for state variables.

Resolution

Resolved in 5c3c0443cde24b3389eeeddd9a90135a609ae119

STC-4 / Informational: Implicit visibility level

Present in StandardToken.sol, lines: L45, L46, L42, L5
Present in Token.sol, lines: L3, L4, L5, L7
Present in STCVesting.sol, lines: L57, L50, L37
Present in STC.sol, lines: L86, L314, L244, L251

Explanation

The default function visibility level in Solidity is public. Explicitly define function visibility to prevent confusion and avoid ambiguity.

Resolution

Resolved in 5c3c0443cde24b3389eeeddd9a90135a609ae119

STC-5 / Informational: Style guide violation

Present in STCVesting.sol, lines: L27
Present in STC.sol, lines: L100

Explanation

In Solidity, function and event names usually start with a lower and uppercase letter respectively:

```
Function getTime(); // good
event GetTime(); // good
```

Violating the style guide decreases readability and leads to confusion. Start function names with lowercase, events with uppercase.

Resolution

Resolved in 5c3c0443cde24b3389eeeddd9a90135a609ae119

CONCLUSION

The reviewed smart contract is well crafted and follows common security practices. No critical problems have been found.

The care and attention to detail by the SmarterThanCrypto team for the token generation event shows their commitment to security. We’re proud to have SmarterThanCrypto as a customer, and look forward to seeing their upcoming success.