COINMERCENARY / SMART CONTRACT AUDIT

Coupon Chain Token Audit

30 JULY 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 Coupon Chain token and crowdsale contract.

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 July 29th, 2018 using git hash ab4544f2f3b83302456949e5d907642ffa2dead4 the following contract files and their respective SHA256 fingerprints were reviewed:

Filename	SHA256 Fingerprint
CouponToken.sol	8135c3662fbf2636975537dd4484656ea2ab20a8f5a92641243282691ae9c185
CouponTokenBounty.sol	9f0372edcd9e93a9fcdac3fd7b4aa7ca22fc071151eb195bb53aba91be1b0253
CouponTokenCampaign.sol	3024f05def133f71cd2f8579d2f34aef5a29c04dfe9b0c662f37c7557ecf2003
CouponTokenConfig.sol	fed606f32b5f6c221eb2a89a7b802aa7e1c3292dbd12248b4eaa2cf69e43978d
CouponTokenSale.sol	3cce2528fd266554a9074860ddc674ddd50560ecb87d3eeb3eeecf0b66678ade
CouponTokenSaleConfig.sol	6bd89ec2d864863e2326bf7e75ae144263ccc0d0ce7dc6b852d33677f102c765

Remediation Audit

On August 5th, 2018 using git hash 572c613c6e917a8baa96fbb36c297e3ae8f6dd94 the following contract files and their respective SHA256 fingerprints were reviewed:

Filename	SHA256 Fingerprint
CouponToken.sol	6af9608b4a42496f2d2ca17b101dcd91e6577312329c93cc33d513279ec1301d
CouponTokenBounty.sol	ead9b2f1f4af9ca3cf4ce711d24cfafdd4f98b1ca1a3de8c12ad072c687a9ca3
CouponTokenCampaign.sol	5d274b2c448aed9a1694df66000d1c180f461cbf2e0245a2903d40fe53e168b6
CouponTokenConfig.sol	979e08051c8a1da23bf91b03de48020da3fd4344b8695b6a2145c912bce44237
CouponTokenSale.sol	b3ab34409ae347287f9fc580f7de53a9cde78bd3d402cac03a6eaef163091051
CouponTokenSaleConfig.sol	05cf09ddf1960ebd5ab8d397c3c186905c4a935eb219ccc8f566247ee6159c5e

AUDIT SUMMARY

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

Analysis Results

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

Test Results

- Extensive unit test coverage available.

Token Allocation Results

- Dynamically mintable token capped by total coupon supply.

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

CTT-1 / Informational: 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 d507c1f42d160080e475a30aa818ca92330dda70.

CONCLUSION

The reviewed smart contracts are free of security issues. The contracts are well crafted and extensive unit tests are available. We commend the Mezzofy team on the quality of their smart contracts.

We look forward to seeing the success of the Mezzofy team and appreciate the opportunity to be a part of their story.