COINMERCENARY / SMART CONTRACT AUDIT

Dinngo Token

25 JUNE 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 Dinngo 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 June 25, 2018 using git hash 6468e1f9b863262a0a676604c89ab67e8ed2487a the following contract files and their respective SHA256 fingerprints were reviewed:

Filename	SHA256 Fingerprint
ICO.sol	6f68efec85cc1d7a89c144e889e25e61ea5a6acd6798a4c8eefd9eb7af7b3288

Remediation Audit

On July 4, 2018 using git hash 565e94e4a9fc991684d32f4cc6187158c4024a36 the following contract files and their respective SHA256 fingerprints were reviewed:

Filename	SHA256 Fingerprint
ICO.sol	42699d7dc143d7744830e56c02e5fc63c3d00aa1815436396beea37e560e89cf

Remediation Audit #2

On August 8, 2018 using git hash c5d89da11431b00b0825750d94f3800efc378b9f the following contract files and their respective SHA256 fingerprints were reviewed:

Filename	SHA256 Fingerprint
DinngoCompany.sol	ea94b59c2d3e14f95b28cdec83def811f2b44803824da3fbfeb18fec8111c00a
DinngoCrowdsale.sol	e328de69ee550ca7a306e43e1009d1a15ab9f984ce5909d66f9673d785d09183
DinngoFounding.sol	a051b0e10c51e3b7aac802e19924abeaba6892b169870d85ae7018108d483d97
DinngoToken.sol	23ecb9d46a95ea98ac76b74cc9f1532d3a52035761ec07528bf589a6ace39611

Remediation Audit #3

On August 14, 2018 using git hash 512e980e98487833250a4a70275094606c8b2013 the following contract files and their respective SHA256 fingerprints were

reviewed:

Filename	SHA256 Fingerprint
DinngoFounding.sol	72358fa361d6bcd6ab6da6bf0c9d96c5e63ac53d2e7fb1820c72c60743d4d941

AUDIT SUMMARY

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

Analysis Results

	Initial Audit	Audit #1	Audit #2	Audit #3
Design Patterns	Updates Recommended	Passed	Passed	Passed
Static Analysis	Passed	Passed	Passed	Passed
Manual Analysis	Updates Recommended	Passed	Passed	Passed
Token Allocation	Passed	Passed	Passed	Passed
Network Behavior	Updates Recommended	Passed	Passed	Passed

Test Results

- No unit test coverage available.

Token Allocation Results

- Symbol: DGO
- Locking contract 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

DGO-1 / Low: Ensure Transfer event is emitted during initial assignment of token balances

Present in ICO.sol, line 512

Explanation

Several third party services such as Etherscan use the Transfer event to maintain an external record of transfers. When assigning the initial DGO tokens in the constructor of the CustomToken contract, Transfer events must be emitted to ensure proper tracking of token allocation when viewed using these third party services.

Resolution

Resolved in 97029e13c08d8c94fecc1b985e29d4e6ca4d6f0f.

DGO-2 / Informational: Contract name should reflect token name

Present in ICO.sol, line 506

Explanation

The token contract name is currently CustomToken. Best practices dictate that the token contract name should reflect the name of the token. DinngoToken would be an acceptable contract name.

Resolution

Resolved in 306ff4ab065a01732dc58e1828f3250b73dc8cc4.

DGO-3 / Informational: Remove reference to _updatePurchasingState() function to optimize gas usage

Present in ICO.sol, line 612

Explanation

The Crowdsale contract references _updatePurchasingState() inside of the heavily used buyTokens() function. The _updatePurchasingState() function contains no code, only a comment about it being an “optional override”. Removing the reference to _updatePurchasingState() will optimize gas usage.

Resolution

Resolved in 565e94e4a9fc991684d32f4cc6187158c4024a36.

DGO-4 / Informational: Remove reference to _postValidatePurchase() function to optimize gas usage

Present in ICO.sol, line 615

Explanation

The Crowdsale contract references _postValidatePurchase() inside of the heavily used buyTokens() function. The _postValidatePurchase() function contains no code, only a comment about it being an “optional override”. Removing the reference to _postValidatePurchase() will optimize gas usage.

Resolution

Resolved in 565e94e4a9fc991684d32f4cc6187158c4024a36.

CONCLUSION

The reviewed smart contracts are free of security issues and well crafted.

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