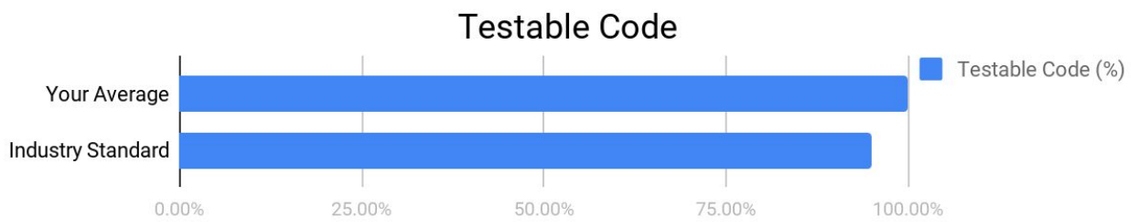
Codex Protocol Contract Audit

by Hosho, April 2018

Executive Summary

This document outlines the overall security of Codex Protocol's smart contract as evaluated by Hosho's Smart Contract auditing team. The scope of this audit was to analyze and document Codex Protocol's token contract codebase for quality, security, and correctness.

All issues have been remediated and suggestions implemented. (See Complete Analysis)



Testable code is higher than industry standard. (See Coverage Report)

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contract, rather limited to an assessment of the logic and implementation. In order to ensure a secure contract that's able to withstand the Ethereum network's fast-paced and rapidly changing environment, we at Hosho recommend that the Codex Protocol Team put in place a bug bounty program to encourage further and active analysis of the smart contract.

Table Of Contents

1. Auditing Strategy and Techniques Applied

The Hosho Team has performed a thorough review of the smart contract code, the latest version as written and updated on April 12, 2018. All main contract files were reviewed using the following tools and processes. (See All Files Covered)

Throughout the review process, care was taken to ensure that the token contract:

- Implements and adheres to existing ERC-20 Token standards appropriately and effectively;
- Documentation and code comments match logic and behavior;
- Distributes tokens in a manner that matches calculations;
- Follows best practices in efficient use of gas, without unnecessary waste; and
- Uses methods safe from reentrance attacks.
- Is not affected by the latest vulnerabilities

The Hosho Team has followed best practices and industry-standard techniques to verify the implementation of Codex Protocol's token contract. To do so, reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as they were discovered. Part of this work included writing a unit test suite using the Truffle testing framework. In summary, our strategies consist largely of manual collaboration between multiple team members at each stage of the review:

1. Due diligence in assessing the overall code quality of the codebase.
2. Cross-comparison with other, similar smart contracts by industry leaders.
3. Testing contract logic against common and uncommon attack vectors.
4. Thorough, manual review of the codebase, line-by-line.
5. Deploying the smart contract to testnet and production networks using multiple client implementations to run live tests.

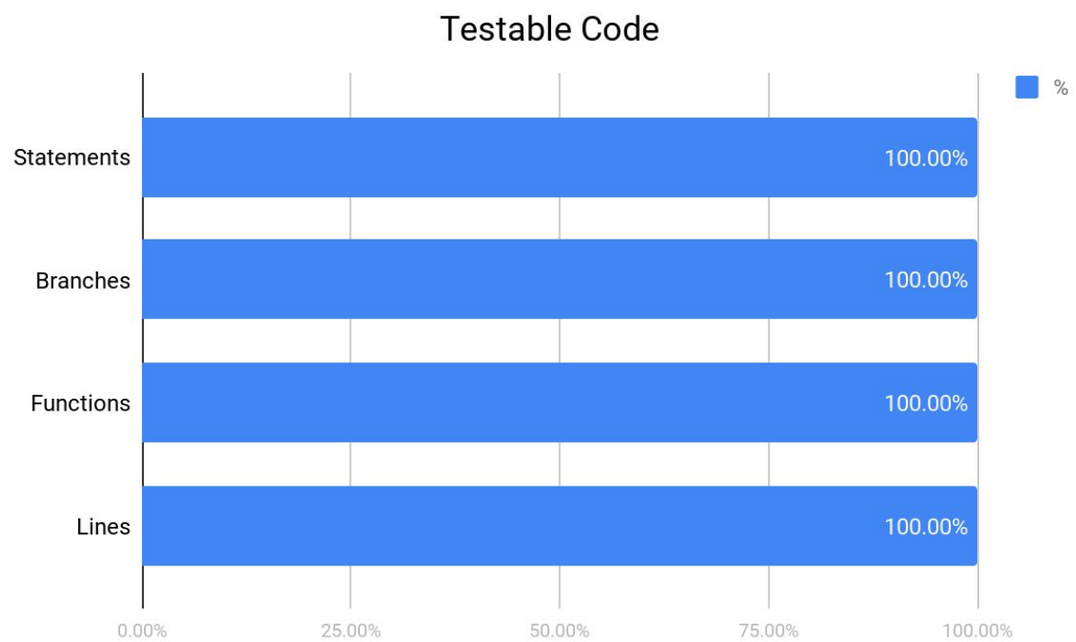
2. Structure Analysis and Test Results

2.1. Summary

The Codex Protocol contracts comprise an ERC-20 compliant token along with an escrow system. There is also a token reclaim system, inherited from OpenZeppelin that allows for tokens that get trapped within the contract to be released. The token contains mintable and ownable functionality as well.

2.2 Coverage Report

As part of our work assisting Codex Protocol in verifying the correctness of their contract code, our team was responsible for writing a unit test suite using the Truffle testing framework.



For individual files see Additional Coverage Report

2.3 Failing Tests

No failing tests.

See Test Suite Results for all tests.

3. Complete Analysis

For ease of navigation, sections are arranged from most critical to least critical. Issues are tagged "Resolved" or "Unresolved" depending on whether they have been fixed or addressed. Furthermore, the severity of each issue is written as assessed by the risk of exploitation or other unexpected or otherwise unsafe behavior:

- **Informational** - The issue has no impact on the contract's ability to operate.
- **Low** - The issue has minimal impact on the contract's ability to operate.
- **Medium** - The issue affects the ability of the contract to operate in a way that doesn't significantly hinder its behavior.
- **High** - The issue affects the ability of the contract to compile or operate in a significant way.
- **Critical** - The issue affects the contract in such a way that funds may be lost, allocated incorrectly, or otherwise result in a significant loss.

3.1. Resolved, Low: Invalid Check

MintingERC20

Explanation

The `mint` function contains a check validating that `disableMinting` is set to true. Because there exists nowhere in the codebase that alters `disableMinting` to a false value, this check is invalid.

Resolution

The Codex Protocol team updated this function to `finishMinting` which is initially false by default and can be set to true by calling the `finishMinting()` function.

3.2. Resolved, Low: Invalid Check

BiddableERC20

Explanation

An if statement is contained in this contract that relies on the boolean value `_transferAllSupplyToOwner`. This contains a non-accessible code path due to the boolean value being statically set to false using the constructor in the Biddable contract.

Resolution

The boolean variable `_transferAllSupplyToOwner` and all references to it have been removed by the Codex Protocol Team.

3.3. Resolved, Informational: No Token Escape

Biddable

Explanation

With the increasing number of ERC-20 tokens getting stuck in contracts, Hosho highly suggests implementing a token escape, so issues like GNT, which has a large number of tokens locked up inside of it's contract, do not occur.

Resolution

The Codex Protocol Team imported the CanReclaimToken contract from Open Zeppelin in order to add the ability to escape tokens for each contract that requires it.

3.4. Resolved, Informational: No Token Escape

BiddableEscrow

Explanation

With the increasing number of ERC-20 tokens getting stuck in contracts, Hosho highly suggests implementing a token escape, so issues like GNT, which has a large number of tokens locked up inside of it's contract, do not occur.

Resolution

The Codex Protocol Team imported the CanReclaimToken contract from Open Zeppelin in order to add the ability to escape tokens for each contract that requires it.

---

4. Closing Statement

We are grateful to have been given the opportunity to work with the Codex Protocol Team.

The team of experts at Hosho, having backgrounds in all aspects of blockchain, cryptography, and cybersecurity, can say with confidence that the Codex Protocol contract is free of any critical issues.

The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them.

We at Hosho recommend that the Codex Protocol Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

5. Test Suite Results

Contract: ERC-20 Tests for Biddable

√ Should deploy a token with the proper configuration (165ms)

√ Should allocate tokens per the minting function, and validate balances (1138ms)

√ Should transfer tokens from 0xd86543882b609b1791d39e77f0efc748dfff7dff to 0x42adbad92ed3e86db13e4f6380223f36df9980ef (377ms)

√ Should not transfer negative token amounts (149ms)

√ Should not transfer more tokens than you have (142ms)

√ Should allow 0xa3883a50d7d537cec8f9bad8e8404aa8ff3078f3 to authorize 0x341106cb00828c87cd3ac0de55eda7255e04933f to transfer 1000 tokens (181ms)

√ Should allow 0xa3883a50d7d537cec8f9bad8e8404aa8ff3078f3 to zero out the 0x341106cb00828c87cd3ac0de55eda7255e04933f authorization (203ms)

√ Should allow 0x667632a620d245b062c0c83c9749c9bfadf84e3b to authorize 0x53353ef6da4bbb18d242b53a17f7a976265878d5 for 1000 token spend, and 0x53353ef6da4bbb18d242b53a17f7a976265878d5 should be able to send these tokens to 0x341106cb00828c87cd3ac0de55eda7255e04933f (805ms)

√ Should not allow 0x53353ef6da4bbb18d242b53a17f7a976265878d5 to transfer negative tokens from 0x667632a620d245b062c0c83c9749c9bfadf84e3b (127ms)

√ Should not allow 0x53353ef6da4bbb18d242b53a17f7a976265878d5 to transfer tokens from 0x667632a620d245b062c0c83c9749c9bfadf84e3b to 0x0 (109ms)

√ Should not transfer tokens to 0x0 (119ms)

√ Should not allow 0x53353ef6da4bbb18d242b53a17f7a976265878d5 to transfer more tokens than authorized from 0x667632a620d245b062c0c83c9749c9bfadf84e3b (154ms)

√ Should allow an approval to be set, then increased, and decreased (950ms)

Contract: Extended ERC-20 Tests for CodexProtocol

Minting

√ Should allow adding/removing of minters (269ms)

√ Should restrict minting to only minters (81ms)

√ Should not allow minting 0 tokens (152ms)

√ Should not allow minting more tokens than are in max supply (235ms)

√ Should not allow minting after finishMinting (158ms)

Ownership

√ Should allow ownership transfers (276ms)

Contract: BiddableEscrow

√ Should allow ownership transfer (205ms)

√ Should allow a new arbitrator to be set (113ms)

√ Should be able to return back the escrow deposit state (389ms)

Deposit

√ Should not allow deposits of non-matching deposit amounts (77ms)

√ Should not allow deposits if the ID already exists (332ms)

√ Should require that the correct signer signs the message to handle an escrow deposit (160ms)

Escrow Release

√ Should only allow the arbitrator to call the release (53ms)

√ Should only allow valid escrows to be released on (100ms)

√ Should release the funds to the depositor, minus the gas fees (393ms)

√ Should allow the gas fees to be withdrawn from the contract (500ms)

Contract: Token Reclaim Tests for Biddable

Token reclaim

√ Should allow the reclaim of tokens from the tested contract (228ms)

√ Should safely handle a failed token transfer (260ms)

Contract: Ownership Tests for Biddable

Deployment

√ Should deploy with the owner being the deployer of the contract

Transfer

√ Should not allow a non-owner to transfer ownership (77ms)

√ Should not allow the owner to transfer to 0x0 (94ms)

√ Should allow the owner to transfer ownership (188ms)

Contract: Pause Tests for Biddable

Deployment

√ Should deploy in an un-paused state

Pause configuration

√ Should be able to be paused (195ms)

√ Should be able to be unpaused (281ms)

√ Should not be able to be unpaused while unpaused (82ms)

√ Should not be able to be paused while paused (257ms)

Contract: SafeERC20 Tests

Failure mode tests

√ Should revert on transfer failure (76ms)

√ Should revert on transferFrom failure (91ms)

√ Should revert on approve failure (82ms)

Success mode tests

√ Should pass on transfer success (99ms)

√ Should pass on transferFrom success (95ms)

√ Should pass on approve success (101ms)

Contract: SafeMath

√ Should skip operation on multiply by zero (54ms)

√ Should revert on multiply overflow (75ms)

√ Should allow regular multiple (87ms)

√ Should revert on divide by zero

√ Should allow regular division (58ms)

√ Should revert on subtraction overflow (42ms)

√ Should allow regular subtraction (66ms)

√ Should revert on addition overflow (55ms)

√ Should allow regular addition (72ms)

6. All Contract Files Tested

Commit Hashes

Contract.biddex - f6fe5203ebde0c2e5620d8ec1f2404004cc7b70c

Contract.biddable-escrow - febb005c4b5cdd6f235655bc159f9430cc527406

File	Fingerprint (SHA256)
contracts/Biddable.sol	8c7ac31548865f4181bc3f237f0b8ddc3340e13d8865b2ada3f5f099cb297ff1
contracts/BiddableERC20.sol	6ad893fdc578088e5de01d338983fa8495e9345c49fcc598e54c3b2be602fddb
contracts/BiddableEscrow.sol	79eb781229ab23fe21f1b4529846e6b0bd8152a534c3b6de7f7c33290118be3d
contracts/MintingERC20.sol	a8315eee1f9e02a180237f793c21741cb80dcc90e9d9c900a30b102a68a9f482
contracts/zeppelin-solidity/lifecycle/Pausable.sol	d7bc1d8a10773084a5d9ea538fa24adc812015cc366682ac9982b3ae2ac0e95f
contracts/zeppelin-solidity/math/SafeMath.sol	f342da26f32f5510cda63479c1ea825445c04e26b1cb42660ce7416a21784714
contracts/zeppelin-solidity/ownership/CanReclaimToken.sol	e8467dbdc6db5fc9d49a424402b7486c2278d0f8f626384f42a579b502d60024
contracts/zeppelin-solidity/ownership/Ownable.sol	88697fcbf64e8989905463351b5734824764485ab76cf4a708c59e3b0adb21c4
contracts/zeppelin-solidity/token/ERC20/BasicToken.sol	9b2a3fa925c0d13087218cf312be43ea643b04cde737e0ebeba8f1d12a5dc2f9
contracts/zeppelin-solidity/token/ERC20/ERC20.sol	f0e1fef2f8d0e75c830f855695e7822851db926f7e62492845139cd665dbb885
contracts/zeppelin-solidity/token/ERC20/ERC20Basic.sol	72535169112b070ae299eddfc8e743e0d3d4997031b305e360db1b1aa38ecc26
contracts/zeppelin-solidity/token/ERC20/SafeERC20.sol	6f7a2d5c680598b994ed8f0c228bd78d1a863d5fea29a00dea1832be3f2dc0c2
contracts/zeppelin-solidity/token/ERC20/StandardToken.sol	d8ea271d3e03064a4a7694a379d4e897c2be335bf2bddc9ec6dd61814bdf1cc7

7. Individual File Coverage Report

File	% Statements	% Branches	% Functions	% Lines
contracts/Biddable.sol	100.00%	100.00%	100.00%	100.00%
contracts/BiddableERC20.sol	100.00%	100.00%	100.00%	100.00%
contracts/BiddableEscrow.sol	100.00%	100.00%	100.00%	100.00%
contracts/MintingERC20.sol	100.00%	100.00%	100.00%	100.00%
contracts/zeppelin-solidity/lifecycle/Pausable.sol	100.00%	100.00%	100.00%	100.00%
contracts/zeppelin-solidity/math/SafeMath.sol	100.00%	100.00%	100.00%	100.00%
contracts/zeppelin-solidity/ownership/CanReclaimToken.sol	100.00%	100.00%	100.00%	100.00%
contracts/zeppelin-solidity/ownership/Ownable.sol	100.00%	100.00%	100.00%	100.00%
contracts/zeppelin-solidity/token/ERC20/BasicToken.sol	100.00%	100.00%	100.00%	100.00%
contracts/zeppelin-solidity/token/ERC20/ERC20.sol	100.00%	100.00%	100.00%	100.00%
contracts/zeppelin-solidity/token/ERC20/ERC20Basic.sol	100.00%	100.00%	100.00%	100.00%
contracts/zeppelin-solidity/token/ERC20/SafeERC20.sol	100.00%	100.00%	100.00%	100.00%
contracts/zeppelin-solidity/token/ERC20/StandardToken.sol	100.00%	100.00%	100.00%	100.00%
All files	**100.00%**	**100.00%**	**100.00%**	**100.00%**