SMART CONTRACT AUDIT

ZOKYO.

Nov 3, 2020 | v. 1.0

PASS

Zokyo Security Team has concluded that there are no issues that can have an impact on contract security.
The contract is well written and is production-ready.

TECHNICAL SUMMARY

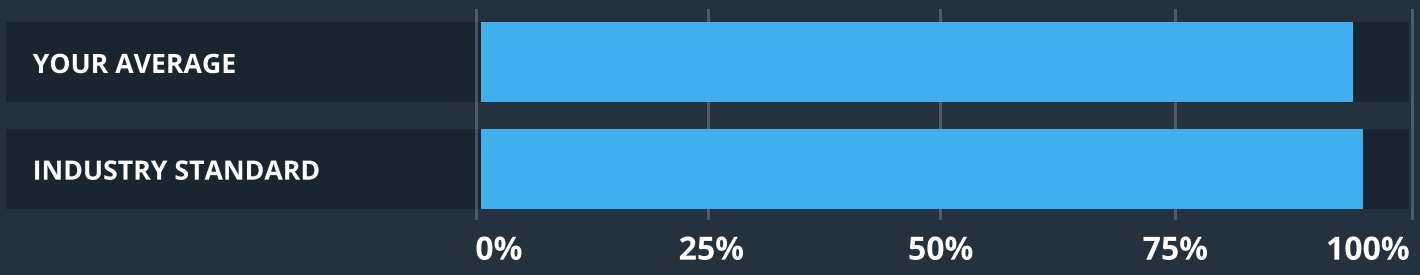
This document outlines the overall security of the Fuse Network smart contracts, evaluated by Zokyo's Blockchain Security team.

The scope of this audit was to analyze and document the Fuse Network smart contract codebase for quality, security, and correctness

Contract Status

There were no critical issues found during the audit.

Testable Code



Testable code is 94%, which is close to the industry standard of 95%.

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 Zokyo recommend that the Fuse team put in place a bug bounty program to encourage further and active analysis of the smart contract.

TABLE OF CONTENTS

AUDITING STRATEGY AND TECHNIQUES APPLIED

The Smart contract's source code was taken from the fuse-network repo pull request (last commit – taking care of block reward without total stake) and master branch (last commit – Fix errors in my last commit).

Requirements: FIP8 spec

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

- Implements and adheres to existing 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;
- Uses methods safe from reentrance attacks;
- Is not affected by the latest vulnerabilities;
- Whether the code meets best practices in code readability, etc.

Zokyo's Security Team has followed best practices and industry-standard techniques to verify the implementation of Fuse Network smart contracts. To do so, the code is reviewed line-by-line by our smart contract developers, documenting any issues as they are discovered. Part of this work includes 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.	3	Testing contract logic against common and uncommon attack vectors.
2	Cross-comparison with other, similar smart contracts by industry leaders.	4	Thorough, manual review of the codebase, line-by-line.

SUMMARY

There were no critical issues found during the **manual audit**. There is only one low level finding which may have effect only in case of specific conditions when the list of validators will be too long.

Contracts are well written and structured however, there still are some improvements to contract code marked with an informational rank.

All the findings during manual audit have no impact on contract performance or security, so it is fully production-ready.

STRUCTURE AND ORGANIZATION OF DOCUMENT

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:

Critical

The issue affects the ability of the contract to compile or operate in a significant way.

High

The issue affects the ability of the contract to compile or operate in a significant way.

Medium

The issue affects the ability of the contract to operate in a way that doesn’t significantly hinder its behavior.

Low

The issue has minimal impact on the contract’s ability to operate.

Informational

The issue has no impact on the contract’s ability to operate.

COMPLETE ANALYSIS

pendingValidators and currentValidators storing structure can be optimized

LOW | UNRESOLVED

All the addresses are stored addressArrayStorage and when it is required to get/update an address contract loops overall array. It can cause a **Loop Issue** and exceed the block gas limit.

Recommendation:
Use mapping to mark address index in addressArrayStorage.

getBallotLimitPerValidator function structure can be optimized

INFORMATIONAL | UNRESOLVED

The function getBallotLimitPerValidator of VotingUtils contract lines 88-98 includes unneeded condition with limit value update.

Recommendation:
Add condition to check if MAX_LIMIT_OF_BALLOTS< validatorsCount in line 92 and return 1 instead of value update and spend gas on math operation in 93 line.

Outdated Compiler Version

LOW | UNRESOLVED

CFXQuantum contract source files contain the ^0.4.24 pragma version. The most recent version at the time of audit performing is 0.7.4 which is minimum in 36 releases newer from current.

Recommendation:
The contracts should be updated to the latest compiler version.

Functions order in contract file

INFORMATIONAL | UNRESOLVED

The function order is wrong according to the code style.

Functions should be grouped according to their visibility and ordered in the following way:

- constructor;
- fallback function (if exists);
- external;
- public;
- internal;
- private.

Recommendation:
Change functions order.

Unused functions are present in contract code

INFORMATIONAL | UNRESOLVED

The contract ConsensusUtils contains the functions which are not used in the contract or contract which inherits ConsensusUtils:

- _setSnapshot (232-246 lines);
- _setNextSnapshotId (224-225 lines);
- _setLastSnapshotTakenAtBlock (216-218 lines);
- _getRandom (482-484 lines).

Recommendation:
Remove unneeded code from SC.

Unneeded function implementation

INFORMATIONAL | UNRESOLVED

The getSnapshotsPerCycle function is not needed as it is a copy of a public variable.

Recommendation:
Remove unneeded function from SC.

CODE COVERAGE AND TEST RESULTS FOR ALL FILES

Tests written by Fuse Network team

Code Coverage

The resulting code coverage (i.e., the ratio of tests-to-code) is as follows:

FILE	% STMTS	% BRANCH	% FUNCS	% LINES	UNCOVERED LINES
contracts/	93.31	79.80	93.10	93.41	
BlockReward.sol	100.00	80.77	100.00	100.00	
Consensus.sol	100.00	83.33	100.00	100.00	
ConsensusUtils.sol	86.00	90.32	82.61	85.71	... 278, 479, 483
ProxyStorage.sol	78.79	50.00	100.00	90.00	99, 103, 105
Voting.sol	100.00	91.67	100.00	100.00	
VotingUtils.sol	95.04	73.81	100.00	96.75	91, 95, 107, 175
contracts/abstracts/	100.00	100.00	100.00	100.00	
BlockRewardBase.sol	100.00	100.00	100.00	100.00	
ValidatorSet.sol	100.00	100.00	100.00	100.00	
VotingBase.sol	100.00	100.00	100.00	100.00	
contracts/eternal-storage/	76.47	50.00	82.35	80.00	
EternalStorage.sol	100.00	100.00	100.00	100.00	
EternalStorageProxy.sol	75.00	50.00	80.00	78.79	... 121, 122, 123
contracts/interfaces/	100.00	100.00	100.00	100.00	
IBlockReward.sol	100.00	100.00	100.00	100.00	
IConsensus.sol	100.00	100.00	100.00	100.00	

IVoting.sol	100.00	100.00	100.00	100.00
All files	**91.18**	**77.57**	**92.15**	**92.45**

Test Results

Contract: BlockReward

initialize
- ✓ default values (113644 gas)

reward
- ✓ can only be called by system address (314292 gas)
- ✓ should revert if input array contains more than one item (58160 gas)
- ✓ should revert if lengths of input arrays are not equal (57749 gas)
- ✓ should revert if "kind" parameter is not 0 (57759 gas)
- ✓ should give reward to validator and total supply should be updated (337219 gas)
- ✓ should give rewards to validator and its delegators (33111904 gas)
- ✓ reward amount should update after BLOCKS_PER_YEAR and total yearly inflation should be calculated correctly (6936463 gas)
- ✓ call reward with 0 blockReward (172836 gas)

#getBlockRewardAmountPerValidator
- ✓ block reward with one validator (186171 gas)
- ✓ block reward of one validator staking 100% of the total stake (186171 gas)
- ✓ block reward of 1 validator of 2, staking 10% of the total stake (229739 gas)
- ✓ block reward of 1 validator of 2, staking 50% of the total stake (229739 gas)
- ✓ block reward does not change if the propotion stays the same (319095 gas)
- ✓ block reward for two validators (359237 gas)
- ✓ block reward without the total stake (140917 gas)

emitRewardedOnCycle
- ✓ should fail if not called by validator (41705 gas)
- ✓ should be successful if `shouldEmitRewardedOnCycle` and `consensus.isFinalized` are true (50057 gas)
- ✓ should be successful and emit event (1335441 gas))

upgradeTo
- ✓ should only be called by ProxyStorage (131316 gas))
- ✓ should change implementation address (108303 gas)
- ✓ should increment implementation version (108303 gas)
- ✓ should work after upgrade (221947 gas)

✓ should use same proxyStorage after upgrade (80290 gas)
✓ should use same storage after upgrade (126714 gas)

Contract: Consensus
initialize
✓ default values (228559 gas)
✓ initial validator address not defined - owner should be initial validator (228328 gas)
✓ initial validator address defined (174607 gas)
✓ initial validator is added to pending after sending fuse (389123 gas)
setProxyStorage
✓ setProxyStorage should fail if no address (24782 gas)
✓ setProxyStorage should fail if not called by owner (105256 gas)
✓ setProxyStorage successfully (53952 gas)
✓ setProxyStorage should not be able to set again if already set (80278 gas)
emitInitiateChange
✓ should fail if not called by validator (30677 gas)
✓ should fail if newValidatorSet is empty (76143 gas)
✓ should fail if `shouldEmitInitiateChange` is false (118515 gas)
✓ should be successful and emit event (162746 gas)
finalizeChange
✓ should only be called by SYSTEM_ADDRESS (105938 gas)
✓ should set finalized to true (80887 gas)
✓ should not update current validators set if new set is empty (107708 gas)
✓ should update current validators set (217504 gas)
with staking
✓ adding validators should update the total stake (507131 gas)
✓ removing validators should update the total stake (507131 gas)
stake (fallback function)
basic
✓ should not allow zero stake (22877 gas)
with the initial validator
✓ less than minimum stake, should update the total stake (96307 gas)
✓ minimum stake amount (160564 gas)
✓ should allow more than minimum stake (160564 gas)
✓ should allow the maximum stake (160564 gas)
✓ should not allow more the maximum stake (68538 gas)
with first candidate
✓ less than minimum stake - should not be added to pending validators (200158 gas)

```
        ✓ minimum stake amount (343375 gas)
        ✓ should allow more than minimum stake (343375 gas)
        ✓ should allow the maximum stake (343375 gas)
        ✓ should not allow more the maximum stake (68538 gas)
    advanced
      with first candidate
        ✓ minimum stake amount, in more than one transaction (390035 gas)
        ✓ more than minimum stake amount, in more than one transaction (390035 gas)
        ✓ more than one validator (522921 gas)
        ✓ multiple validators, multiple times (784725 gas)
      with existing stake
        ✓ minimum stake amount, in more than one transaction (246324 gas)
        ✓ more than minimum stake amount, in more than one transaction (246324 gas)
        ✓ more than one validator (379337 gas)
        ✓ multiple validators, multiple times (641141 gas)
      total stake
        ✓ two validators (802862 gas)
        ✓ validator with delegators (771535 gas)
stake
  basic
    ✓ should not allow zero stake (23511 gas)
    ✓ less than minimum stake - should not be added to pending validators (77303 gas)
    ✓ minimum stake amount (344018 gas)
    ✓ should allow more than minimum stake (344018 gas)
    ✓ should allow the maximum stake (344018 gas)
    ✓ should not allow more the maximum stake (69172 gas)
  advanced
    ✓ minimum stake amount, in more than one transaction (391321 gas)
    ✓ more than minimum stake amount, in more than one transaction (391321 gas)
    ✓ more than one validator (524207 gas)
    ✓ multiple validators, multiple times (629657 gas)
delegate
  basic
    ✓ should not allow zero stake (24077 gas)
    ✓ should fail if no staker address (23881 gas)
    ✓ less than minimum stake - should not be added to pending validators (121602 gas)
    ✓ minimum stake (388318 gas)
    ✓ should allow the maximum stake (388318 gas)
```

```
      ✓ should not allow more the maximum stake (113469 gas)
    advanced
      ✓ minimum stake amount, in more than one transaction (439916 gas)
      ✓ more than one validator (612803 gas)
      ✓ multiple times according to staked amount, in more than one transaction (292682 gas)
  cycles and snapshots
    ✓ hasCycleEnded
    ✓ cycle (334761 gas)
    ✓ cycle function should only be called by BlockReward (92049 gas)
  withdraw
    stakers
      ✓ cannot withdraw zero (23679 gas)
      ✓ cannot withdraw more than staked amount (166058 gas)
      stakers are not curent validators
        ✓ can withdraw all staked amount (320990 gas)
        ✓ can withdraw less than staked amount (197216 gas)
        ✓ can withdraw multiple times (257998 gas)
      stakers are curent validators
        ✓ cannot withdraw the min stake for active validator (373181 gas)
        ✓ cannot withdraw one wei from the min stake for active validator (373139 gas)
        ✓ can withdraw until the min stake for active validator (407426 gas)
        ✓ can withdraw multiple times (471429 gas)
    delegators
      ✓ cannot withdraw zero (25096 gas)
      ✓ cannot withdraw if no staker address defined (210783 gas)
      ✓ cannot withdraw more than staked amount (212418 gas)
      ✓ can withdraw all staked amount (421720 gas)
      ✓ can withdraw less than staked amount (243578 gas)
      ✓ can withdraw multiple times (305779 gas)
  setValidatorFee
    ✓ should only be called by validator (220342 gas)
    ✓ should only be able to set a valid fee (3468120 gas)
  getDelegatorsForRewardDistribution
    ✓ validator without delegators (220342 gas)
    ✓ validator with many delegators (3468120 gas)
    validator with one delegator
      ✓ total delegated more than total staked - no fee (262294 gas)
      ✓ total delegated less than total staked - no fee (262294 gas)
```

```
        ✓ total delegated more than total staked - 100% fee (277139 gas)
        ✓ total delegated less than total staked - 100% fee (277139 gas)
        ✓ total delegated more than total staked - other fee (277151 gas)
        ✓ total delegated less than total staked - other fee (277151 gas)
      validator with multiple delegators
        ✓ total delegated more than total staked - no fee (357581 gas)
        ✓ total delegated less than total staked - no fee (357581 gas)
        ✓ total delegated more than total staked - 100% fee (372426 gas)
        ✓ total delegated less than total staked - 100% fee (372426 gas)
        ✓ total delegated more than total staked - other fee (372426 gas)
        ✓ total delegated less than total staked - other fee (372426 gas)
    upgradeTo
      ✓ should only be called by ProxyStorage (220342 gas)
      ✓ should change implementation address (220342 gas)
      ✓ should increment implementation version (220342 gas)
      ✓ should work after upgrade (220342 gas)
      ✓ should use same proxyStorage after upgrade (220342 gas)
      ✓ should use same storage after upgrade (220342 gas)

  Contract: ProxyStorage
    initialize
      ✓ should be successful (69220 gas)
    initializeAddresses
      ✓ should fail if not called from owner (25360 gas)
      ✓ should be successful (93748 gas)
      ✓ should not be called twice (120524 gas)
    upgradeTo
      ✓ should only be called by ProxyStorage (this) (103303 gas)
      ✓ should change implementation address (80290 gas)
      ✓ should increment implementation version (80290 gas)
      ✓ should work after upgrade (149510 gas)

  Contract: Voting
    initialize
      ✓ should be successful (47511 gas)
    newBallot
      ✓ should be successful (412121 gas)
      ✓ should fail if not called by valid voting key (139989 gas)
```

✓ should fail if duration is invalid (560392 gas)
✓ should fail if proposed value is invalid (139999 gas)
✓ should fail if contract type is invalid (145744 gas)
✓ should fail if creating ballot over the ballots limit (5454904 gas)

vote

✓ should vote "accept" successfully (52898 gas)
✓ should vote "reject" successfully (52935 gas)
✓ multiple voters should vote successfully (211666 gas)
✓ should be successful even if called by non validator (52898 gas)
✓ should fail if voting before start time (27071 gas)
✓ should fail if voting after end time (28509 gas)
✓ should fail if trying to vote twice (83111 gas)
✓ should fail if trying to vote with invalid choice (60632 gas)
✓ should fail if trying to vote for invalid id (57414 gas)

onCycleEnd

✓ should only be called by Consensus (94175 gas)
✓ should work when there are no validators (should not happen) (60662 gas)
✓ should work when there are no active ballots (65883 gas)
✓ should work when there is one active ballot - no votes yet (508694 gas)
✓ golden flow should work (3423733 gas)
✓ golden flow should work wih a lot of validators, a lot of votes

153 passing (26m)

Tests written by Zokyo Secured team

As part of our work assisting Fuse in verifying the correctness of their contract code, our team was responsible for writing additional tests using the Truffle testing framework.
Tests were based on the functionality of the code, as well as a review of the Fuse contract to cover untested lines of code.

Code Coverage

The resulting code coverage (i.e., the ratio of tests-to-code) is as follows:

FILE	% STMTS	% BRANCH	% FUNCS	% LINES	UNCOVERED LINES
contracts/	94.12	83.33	93.68	94.51	
BlockReward.sol	100.00	80.77	100.00	100.00	
Consensus.sol	100.00	83.33	100.00	100.00	
ConsensusUtils.sol	86.67	90.32	84.06	86.34	... 253, 479, 483
ProxyStorage.sol	93.94	69.23	100.00	100.00	
Voting.sol	100.00	91.67	100.00	100.00	
VotingUtils.sol	96.69	78.57	100.00	97.56	91, 107, 175
contracts/abstracts/	100.00	100.00	100.00	100.00	
BlockRewardBase.sol	100.00	100.00	100.00	100.00	
ValidatorSet.sol	100.00	100.00	100.00	100.00	
VotingBase.sol	100.00	100.00	100.00	100.00	
contracts/eternal-storage/	94.12	62.50	100.00	100.00	
EternalStorage.sol	100.00	100.00	100.00	100.00	
EternalStorageProxy.sol	93.75	62.50	100.00	100.00	
contracts/interfaces/	100.00	100.00	100.00	100.00	
IBlockReward.sol	100.00	100.00	100.00	100.00	
IConsensus.sol	100.00	100.00	100.00	100.00	

IVoting.sol	100.00	100.00	100.00	100.00
All files	**94.12**	**81.78**	**94.24**	**94.90**

Most uncovered lines are useless functions that were described in the Complete Analysis section.

We are grateful to have been given the opportunity to work with the Fuse Network team.

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.

Zokyo's Security Team recommends that the Fuse Network team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

ZOKYO.