milestoneBased

SMART CONTRACT AUDIT

ZOKYO.

Nov 15th, 2021 | v. 1.0

PASS

Zokyo Security Team has concluded that this smart contract passes security qualifications and bear no security or operational risk

TECHNICAL SUMMARY

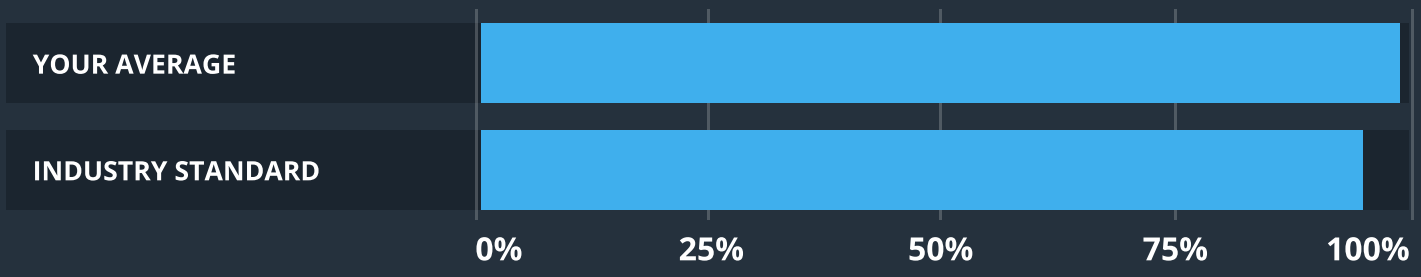
This document outlines the overall security of the milestoneBased smart contracts, evaluated by Zokyo's Blockchain Security team.

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

Contract Status

There were no critical issues found during the audit.

Testable Code



The testable code is 99%, which is above 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 milestoneBased 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 milestoneBased repository.

Repository:
https://bitbucket.org/applicature/milestonebased.contracts/src/release-v1/

Last commit:
31dddb3

Contracts under the scope:

- Voting;
- SingleSignVotingStrategy;
- Roadmap;
- MilestoneBased;
- IVotingStrategy.

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 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

Zokyo security team has conducted a security audit for the given list of smart contracts. The contracts are in good condition. They are well written and structured. All the issues and vulnerabilities found are presented in the “Complete Analysis” section of this report. Among them, there are 2 issues with low severity and 3 informational issues. All of the mentioned findings were successfully resolved by the milestoneBased team. After a review of the fixes and comments, one issue was marked as not valid.

Zokyo security team states that the smart contracts bear no security or operational risks and are full production-ready. Based on the quality of the codebase and the outcome of the audit, the score is set to 99.

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

LOW | RESOLVED

Function setTrustedSigner should do sanity checks even if the caller is the owner, the owner can make mistakes too.

Recommendation:
Do a sanity check where you are checking if the new trusted signer has a different value from the old one, to not set the same value twice and consume gas.

LOW | RESOLVED

Function setUrl should do sanity checks even if the caller is the owner, the owner can make mistakes too.

Recommendation:
Do a sanity check where you are checking if the new uri has a different value from the old one, to not be able to set the same value twice and consume gas.

INFORMATIONAL | RESOLVED

There are some known bugs in the older versions of solidity that have been fixed in the most recent one, https://docs.soliditylang.org/en/latest/bugs.html , it's part of best practices to always use the most recent updated version.

Recommendation:
Set the solidity pragma to the most recent one (=0.8.9).

INFORMATIONAL | RESOLVED

Event Withdrawn from Roadmap contract should index the recipient address too, to be able to filter based on it.

Recommendation:
Make the recipient parameter indexed in the Withdrawn event.

INFORMATIONAL | NOT VALID

Error messages should be more straightforward than just an abbreviation, even if there is documentation to support them, openzeppelin way of handling them by adding the contract source followed by a straight and short explanation it's a good practice.

Recommendation:
Make error messages more human-readable and straightforward, look into how openzeppelin contracts handle the error messages.

	Voting	Roadmap	MilestoneBased
Re-entrancy	Pass	Pass	Pass
Access Management Hierarchy	Pass	Pass	Pass
Arithmetic Over/Under Flows	Pass	Pass	Pass
Unexpected Ether	Pass	Pass	Pass
Delegatecall	Pass	Pass	Pass
Default Public Visibility	Pass	Pass	Pass
Hidden Malicious Code	Pass	Pass	Pass
Entropy Illusion (Lack of Randomness)	Pass	Pass	Pass
External Contract Referencing	Pass	Pass	Pass
Short Address/ Parameter Attack	Pass	Pass	Pass
Unchecked CALL Return Values	Pass	Pass	Pass
Race Conditions / Front Running	Pass	Pass	Pass
General Denial Of Service (DOS)	Pass	Pass	Pass
Uninitialized Storage Pointers	Pass	Pass	Pass
Floating Points and Precision	Pass	Pass	Pass
Tx.Origin Authentication	Pass	Pass	Pass
Signatures Replay	Pass	Pass	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass	Pass	Pass

	SingleSignVotingStrategy	IVotingStrategy
Re-entrancy	Pass	Pass
Access Management Hierarchy	Pass	Pass
Arithmetic Over/Under Flows	Pass	Pass
Unexpected Ether	Pass	Pass
Delegatecall	Pass	Pass
Default Public Visibility	Pass	Pass
Hidden Malicious Code	Pass	Pass
Entropy Illusion (Lack of Randomness)	Pass	Pass
External Contract Referencing	Pass	Pass
Short Address/ Parameter Attack	Pass	Pass
Unchecked CALL Return Values	Pass	Pass
Race Conditions / Front Running	Pass	Pass
General Denial Of Service (DOS)	Pass	Pass
Uninitialized Storage Pointers	Pass	Pass
Floating Points and Precision	Pass	Pass
Tx.Origin Authentication	Pass	Pass
Signatures Replay	Pass	Pass
Pool Asset Security (backdoors in the underlying ERC-20)	Pass	Pass

CODE COVERAGE AND TEST RESULTS FOR ALL FILES

Tests written by milestoneBased 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\	97.56	93.00	97.67	97.64	
MilestoneBased.sol	100.00	100.00	100.00	100.00	
Roadmap.sol	98.59	95.65	100.00	98.65	337
SingleSignVotingStrategy.sol	100.00	100.00	100.00	100.00	
Voting.sol	96.36	90.00	93.75	96.49	270, 271, 326, 336
contracts\interfaces\	100.00	100.00	100.00	100.00	
IVotingStrategy.sol.sol	100.00	100.00	100.00	100.00	
contracts\mocks\	83.33	100.00	83.33	83.33	
ERC20Mock.sol	100.00	100.00	100.00	100.00	
VotingStrategyMock.sol	75.00	100.00	75.00	75.00	18
contracts\tests\	60.00	25.00	100.00	66.67	
RoadmapTest.sol	100.00	100.00	100.00	100.00	
Utils.sol	50.00	25.00	100.00	60.00	30, 40
All files	**96.30**	**90.38**	**96.08**	**96.43**	

Test Results

Contract: Integration
- ✓ should go through first use case correctly (8363ms)

Contract: MilestoneBased

Upgrade roadmap
- ✓ should change implementation of roadmaps (1408ms)

Create Roadmap
- ✓ should initialize created roadmap with expected data (1309ms)
- ✓ should set roadmap mapping (905ms)
- ✓ should emit event (612ms)

Contract: Roadmap

funding roadmap
- ✓ cannot funding if state is not funding (442ms)
- ✓ should funding roadmap correctly (945ms)
- ✓ should emit event on funding (556ms)

withdraw milestone funds
- ✓ cannot withdraw if state is not funding (1308ms)
- ✓ cannot withdraw if caller is not admin (1460ms)
- ✓ cannot withdraw if voting status is suspended (1658ms)
- ✓ cannot withdraw if voting status is not correct in MilestoneStartDate funds release type (1147ms)
- ✓ should withdraw if voting status is correct in MilestoneStartDate funds release type (2715ms)
- ✓ cannot withdraw if voting status is not correct in MilestoneEndDate funds release type (1609ms)
- ✓ should withdraw if voting status is correct in MilestoneEndDate funds release type (2829ms)
- ✓ cannot withdraw more than milestone available funds (2383ms)
- ✓ should emit event on withdraw (2346ms)
- ✓ should be able to withdraw from admin (2817ms)

Contract: Voting

Is Valid
- ✓ should revert if not one U256 parameter (791ms)
- ✓ should revert if not two B32 parameter (602ms)
- ✓ should return true with a valid signature (593ms)
- ✓ should return false with an invalid signature (737ms)

Set trusted signer
- ✓ should revert if not owner (247ms)

✓ should actually change a trusted signer (351ms)
✓ should emit event (381ms)

Set url

✓ should revert if not owner (291ms)
✓ should actually change a trusted signer (656ms)
✓ should emit event (390ms)

Contract: Voting

Create proposal

✓ should revert on options array length mismatch (531ms)
✓ should revert on concrete option array length mismatch (400ms)
✓ should revert if no options provided (285ms)
✓ should actually create proposal (585ms)
✓ should create consecutive proposals (1715ms)
✓ should emit creation event (506ms)

Vote

✓ should revert if proposal does not exist (189ms)
✓ should revert if too late to vote (359ms)
✓ should revert if signature check failed (553ms)
✓ should revert if option id is out of bounds (321ms)
✓ should revert if voting for the same option (536ms)
✓ should cancel previous vote (963ms)
✓ should change voting power on a vote (394ms)
✓ should change overall voting values on a vote (500ms)
✓ should emit Vote event (304ms)

Cancel Vote

✓ should revert if proposal does not exist (221ms)
✓ should revert if too late to vote (354ms)
✓ should revert if no vote exist (475ms)
✓ should remove voting power and relative values (445ms)
✓ should emit ProposalVoteCancelled event (269ms)

Execute

✓ should revert on a reentrancy (1433ms)
✓ should fail if proposal does not exist (263ms)
✓ should fail if called too late (395ms)
✓ should fail if not maximum power option (297ms)
✓ should fail if equal voting powers (2009ms)
✓ should fail if already executed (827ms)
✓ should fail if voting power consensus is not reached (1701ms)

✓ should fail if voters consensus is not reached (1729ms)
✓ should execute empty option (1912ms)
✓ should actually add milestone on an execute (995ms)
✓ should actually add several milestones on an execute (2501ms)
✓ should emit ProposalExecuted event with expected data (697ms)
✓ should correctly handle execution failure (2697ms)
✓ should execute empty option (1879ms)

Get Options

✓ should return expected data (683ms)

Contract: Roadmap

✓ cannot initialize twice (718ms)
✓ should initialize correctly (734ms)
✓ should initialize admin correctly (175ms)

set refunding contract

✓ cannot set if caller is not voting contract (302ms)
✓ cannot set if address is zero (255ms)
✓ cannot set in not Funding state (476ms)
✓ should set correctly (428ms)
✓ should emit event on set (309ms)

set voting contract

✓ cannot set if caller is not voting contract (243ms)
✓ cannot set if address is zero (241ms)
✓ cannot set in not Funding state (618ms)
✓ should set correctly (340ms)
✓ should emit event on set (244ms)

add milestone

✓ cannot add if caller is not voting contract (300ms)
✓ cannot add if in not Funding state (578ms)
✓ cannot add if roadmap balance is less than required for the milestone (244ms)
✓ cannot add if milestone already exists (767ms)
✓ cannot add if start end date is 0 (225ms)
✓ cannot add if start end is later than end date (301ms)
✓ should add correctly (1085ms)
✓ should add multi milestones correctly (1531ms)
✓ should emit event on add (489ms)

update milestone

✓ cannot update if caller is not voting contract (290ms)
✓ cannot update if in not Funding state (452ms)

```
    ✓ cannot update non-existent milestone (269ms)
    ✓ cannot update if start end date is 0 (177ms)
    ✓ cannot update already started milestone (248ms)
    ✓ should update correctly (1567ms)
    ✓ should emit event on update (309ms)
  remove milestone
    ✓ cannot remove if caller is not voting contract (295ms)
    ✓ cannot remove non existent milestone (252ms)
    ✓ cannot remove if in not Funding state (506ms)
    ✓ cannot remove already started milestone (311ms)
    ✓ should remove correctly (635ms)
    ✓ should emit event on remove (192ms)
  update milestone voting status
    ✓ cannot update if caller is not voting contract (281ms)
    ✓ cannot update if in not Funding state (639ms)
    ✓ cannot update if milestone does not exist (282ms)
    ✓ cannot update back to Active (355ms)
    ✓ cannot update back to Suspended in Finished Status (606ms)
    ✓ cannot update to Finished in Suspended Status (567ms)
    ✓ cannot update to Suspended in Suspended Status (729ms)
    ✓ cannot update to Finished in Finished Status (790ms)
    ✓ cannot update before start date (249ms)
    ✓ should update to Suspended correctly (696ms)
    ✓ should update to Finished correctly (627ms)
    ✓ should emit event on update (443ms)
  update roadmap state
    ✓ cannot update if caller is not voting contract (397ms)
    ✓ should update to refunding state correctly (1544ms)
    ✓ cannot update back to funding state (764ms)

  113 passing (3m)
```

Tests written by Zokyo Security team

As part of our work assisting milestoneBased in verifying the correctness of their contract code, our team was responsible for writing integration tests using the Truffle testing framework.

Tests were based on the functionality of the code, as well as a review of the milestoneBased contract requirements for details about issuance amounts and how the system handles these.

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\	99.02	97.00	100.00	98.58	
MilestoneBased.sol	100.00	100.00	100.00	100.00	
Roadmap.sol	98.59	97.83	100.00	98.58	337
SingleSignVotingStrategy.sol	100.00	100.00	100.00	100.00	
Voting.sol	99.09	96.00	100.00	98.25	422, 425
contracts\interfaces\	100.00	100.00	100.00	100.00	
IVotingStrategy.sol	100.00	100.00	100.00	100.00	
All files	**99.02**	**97.00**	**100.00**	**98.58**	

Test Results

```
Contract: MilestoneBased
  functions:
    createRoadmap
      ✓ should create roadmap and voting contract by provided parameters correctly (1288ms)
      ✓ should catch event
```

Contract: Roadmap

functions:

fundRoadmap

- ✓ should send tokens from sender to contract correctly (367ms)
- ✓ should catch event
- ✓ should revert if state not Funding (2113ms)

withdraw

- ✓ should revert if melistone does not withdrawable (644ms)
- ✓ should revert if caller is not admin or voting (458ms)
- ✓ should revert if amount of withdrawing tokens is greater than milestone balance (343ms)
- ✓ should revert if voting status is suspended (462ms)
- ✓ should withdraw fund from milestone correctly (787ms)
- ✓ should catch event

setRefundingContract

- ✓ should revert if caller is not voting contract (231ms)
- ✓ should revert if new refunding contract is zero address (232ms)
- ✓ should change refunding contract correctly (365ms)
- ✓ should catch event

setVotingContract

- ✓ should revert if caller is not voting (255ms)
- ✓ should revert if new voting contract is zero address (179ms)
- ✓ should change voting contract correctly (500ms)
- ✓ should catch event

addMilestone

- ✓ should revert if caller is not voting contract (236ms)
- ✓ should revert if melistone already exist (273ms)
- ✓ should revert if dates are incorrect (263ms)
- ✓ should revert if amount of fund tokens for a milestone is greater than roadmap balance without locked funds (238ms)
- ✓ should add milestone correctly (829ms)
- ✓ should catch event

updateMilestone

- ✓ should revert if caller is not voting contract (300ms)
- ✓ should revert if melistone does not exist (256ms)
- ✓ should revert if new dates are incorrect (261ms)
- ✓ should revert if milestone already started (251ms)
- ✓ should update information about already existing milestone correctly (548ms)
- ✓ should catch event

```
    removeMilestone
      ✓ should revert if caller is not voting contract (317ms)
      ✓ should revert if melistone does not exist (261ms)
      ✓ should revert if milestone already started (225ms)
      ✓ should remove milestone correctly (657ms)
      ✓  should catch event
    updateMilestoneVotingStatus
      ✓ should revert if caller is not voting contract (318ms)
      ✓ should revert if melistone does not exist (273ms)
      ✓ should revert if milestone has not started (648ms)
      ✓ should revert if transitions of a voting status is not supported (243ms)
      ✓ should update milestone voting status correctly (390ms)
      ✓ should catch event
    updateRoadmapState
      ✓ should revert if caller is not voting contract (259ms)
      ✓ should update roadmap state correctly and send all funds to refunding contract (677ms)
      ✓ should catch Refunded event
    checkIsMilestoneWithdrawable
      FundsReleaseType: MilestoneStartDate
        ✓ should return true if startDate is less than current time (75ms)
        ✓ should return false if startDate is greater than current time (104ms)
      FundsReleaseType: MilestoneEndDate
        ✓ should return true if endDate is less than current time (555ms)
        ✓ should return false if endDate is greater than current time (608ms)

Contract: SingleSignVotingStrategy
  constructor
    ✓ should deploy with correct trusted signer address (122ms)
    ✓ should deploy with correct signature generation resource url (143ms)
  functions:
    setTrustedSigner
      ✓ should revert if caller is not owner (244ms)
      ✓ should set trusted signer correctly (637ms)
      ✓ should catch event
    setUrl
      ✓ should revert if caller is not owner (275ms)
      ✓ should set url correctly (446ms)
      ✓ should catch event
```

```
    url
      ✓ should return url correctly (131ms)
    isValid
      ✓ should return true with a valid signature (261ms)
      ✓ should return false with an invalid signature (529ms)
      ✓ should revert if argumentsU256 not one (692ms)
      ✓ should revert if argumentsB32 not two (400ms)

Contract: Voting
  functions:
    addProposal
      ✓ should add new propola correctly (890ms)
      ✓ should revert if arrays have different length (272ms)
      ✓ should revert if subarrays have different length (653ms)
      ✓ should revert if targets array is empty (342ms)
      ✓ should catch event
    vote
      ✓ should votes in proposal correctly (1166ms)
      ✓ should revote correctly (891ms)
      ✓ should catch event
      ✓ should revert with invalid signature (981ms)
      ✓ should revert with invalid optionId (365ms)
      ✓ cannot vote with the same options (352ms)
    cancelVote
      ✓ should cancel vote correctly (766ms)
      ✓ should revert if vote does not exist (473ms)
      ✓ should revert if vote not in proposal time interval (391ms)
      ✓ should catch event
    execute
      ✓ should execute an option in a proposal correctly (2522ms)
      ✓ should catch event
      ✓ should revert if proposal does not exist (328ms)
      ✓ should revert if not maximum power option (846ms)
      ✓ should revert if option already executed (625ms)
      ✓ should revert if minConsensusVotingPower greater than total voting power in
        this proposal (3010ms)
      ✓ should revert if minConsensusVotersCount greater than total voters in this proposal (2798ms)
      ✓ should revert with invalid callTargets (2597ms)
```

```
  getOptionCount
    ✓ should return option count correctly (447ms)
    ✓ should revert if proposal does not exist (182ms)
  getOptions
    ✓ should return options correctly (789ms)
    ✓ should revert if proposal does not exist (211ms)
  proposalsCount
    ✓ should return proposals count correctly (153ms)
  proposalExists
    ✓ should return true if proposal exist (158ms)
    ✓ should return false if proposal does not exist (110ms)
  proposalTimeInterval
    ✓ should return LockBeforeVoting time interval correctly (610ms)
    ✓ should return Voting time interval correctly (292ms)
    ✓ should return LockBeforeExecution time interval correctly (287ms)
    ✓ should return Execution time interval correctly (183ms)
    ✓ should return AfterExecution time interval correctly (330ms)
  maxVotingPowerOption
    ✓ should return information about option with a maximum voting power
      for proposal correctly (452ms)

98 passing (1m)
```

We are grateful to have been given the opportunity to work with the milestoneBased 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 milestoneBased team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

ZOKYO.