VIDMA

milestoneBased

SMART CONTRACT AUDIT

Project: milestoneBased
Date: March 23rd, 2022

TABLE OF CONTENTS

VIDMA

SUMMARY

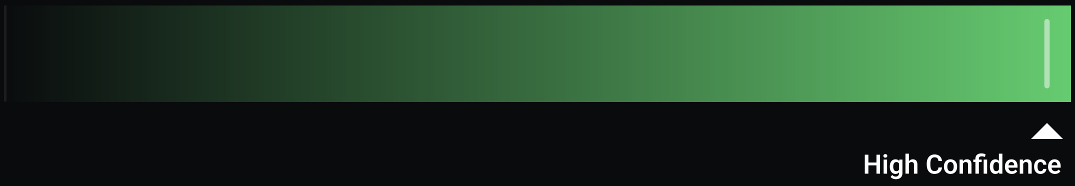
Vidma team has conducted a smart contract audit for the given token and vesting contracts. Both contracts are in excellent condition and are well written.

During the auditing process, the Vidma security team hasn't found any issues stating that the audited contracts are fully production-ready and are safe to use.

A detailed summary of the issues and their current state is displayed in the table below.

The severity of the issue	Total found	Resolved	Unresolved
Critical	0 issues	0 issues	0 issues
High	0 issues	0 issues	0 issues
Medium	0 issues	0 issues	0 issues
Low	0 issues	0 issues	0 issues
Informational	0 issues	0 issues	0 issues
Total	**0 issues**	**0 issues**	**0 issues**

Evaluating the findings, we can assure that the contracts are fully operational, optimized and have no security issues. Under the given circumstances we can set the following risk level:



Vidma auditors are evaluating the initial commit given for the scope of the audit and the last commit with the fixes. Hence, it helps to adequately evaluate the development quality. Code style, optimization of the contracts, amount, and risk level of the issues are taken into consideration. The Vidma team has developed the transparent scoring system presented below.

Severity of the issue	Resolved	Unresolved
Critical	1	10
High	0.8	7
Medium	0.5	5
Low	0.2	0.5
Informational	0	0.1

Based on the given findings, risk level, performance, and code style, the Vidma team can grant the following overall score:

100

Vidma auditing team has conducted a bunch of integrated autotests to ensure that the given codebase has decent performance and security levels. The test results and the coverage can be found in the accompanying section of this audit report.

Please mind that this audit does not certify the definite reliability and security level of the contracts. This document describes all vulnerabilities, typos, performance issues, and security issues found by Vidma auditing team. If the code is under development, we recommend running one more audit once the code is finalized.

SCOPE OF WORK

milestoneBased

milestoneBased is on a mission to fix a legacy system of VC capital inefficiency by revolutionizing collaboration on milestone management between crypto investors and startups. It is the first company to leverage a blockchain DAO and escrow smart contract capabilities, in an automated governance and milestone achievement management platform. Early-stage investors achieve greater capital and process efficiency, improved security and transparency, and data-driven insights, for a faster path to liquidity and monetization of deployed funds. Startup teams become milestone focused and motivated for strengthened performance.

Within the scope of this audit, two independent auditors deeply investigated the given codebase and analyzed the overall security and performance of the smart contracts.

The debrief took place from March 21st to March 23rd, 2022 and the final results are present in this document.

Vidma auditing team has made a review of the following contracts:

- MilestoneBasedVesting;
- MilestoneBasedToken.

The source code was taken from the following **source**:
https://bitbucket.org/applicature/milestonebased.contracts

Initial commit submitted for the audit:
92369ff117273eb2bd930a3e490f30f11991e4d2

In order to conduct a more detailed audit, milestoneBased has provided the following **documentation**:
https://drive.google.com/drive/folders/1yqt6Xzr5g8hm-JragY9ow239BoZ6IKkm?usp=sharing

WORKFLOW OF THE AUDITING PROCESS

During the manual phase of the audit, Vidma team manually looks through the code in order to find any security issues, typos, or discrepancies with the logic of the contract.

Within the testing part, Vidma auditors run integration tests using the Truffle testing framework. The test coverage and the tests themselves are inserted into this audit report.

Vidma team uses the most sophisticated and contemporary methods and techniques to ensure the contract does not have any vulnerabilities or security risks:

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

STRUCTURE AND ORGANIZATION OF THE FINDINGS

For the convenience of reviewing the findings in this report, Vidma auditors classified them in accordance with the severity of the issues. (from most critical to least critical). The acceptance criteria are described below.

All issues are marked as "Resolved" or "Unresolved", depending on whether they have been fixed by milestoneBased or not. The latest commit, indicated in this audit report should include all the fixes made.

To ease the explanation, the Vidma team has provided a detailed description of the issues and recommendations on how to fix them.

Hence, according to the statements above, we classified all the findings in the following way:

Finding	Description
🟪 Critical	The issue bear a definite risk to the contract, so it may affect the ability to compile or operate.
🟥 High	Major security or operational risk found, that may harm the end-user or the overall performance of the contract.
🟧 Medium	The issue affects the contract to operate in a way that doesn’t significantly hinder its performance.
🟨 Low	The found issue has a slight impact on the performance of the contract or its security.
🟩 Informational	The issue does not affect the performance or security of the contract/recommendations on the improvements.

MANUAL REPORT

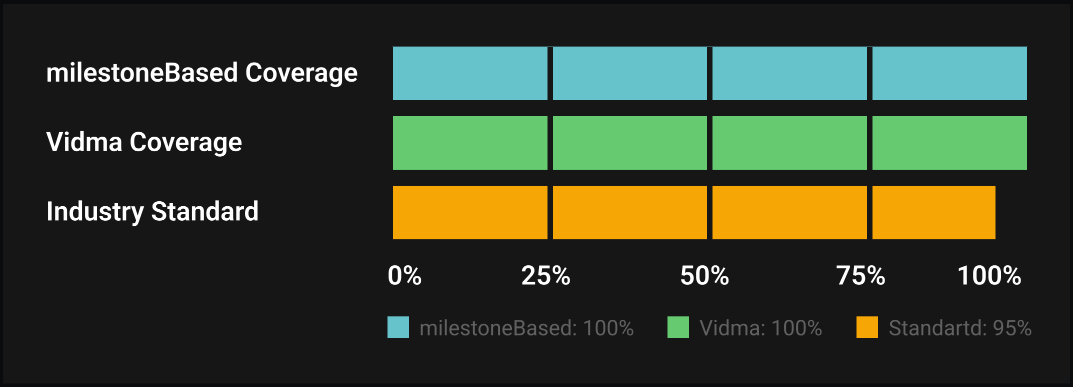
Vidma auditors has conducted a deep analysis of the smart contracts. As the outcome, no issues were identified. The contracts are in excellent condition and no fixes were required by the auditing team.

TEST RESULTS

To verify the contract security and performance a bunch of integration tests were made using the Truffle testing framework.

Tests were based on the functionality of the code, business logic, and requirements and for the purpose of finding the vulnerabilities in the contacts.

In this section, we provide both tests written by milestoneBased and Vidma auditors.



It's important to note that Vidma auditors do not modify, edit or add tests to the existing tests provided in the milestoneBased repo. We write totally separate tests with code coverage of a minimum of 95%, to meet the industry standards.

Tests are written by milestoneBased

Test Coverage

File	% Stmts	% Branch	% Funcs	% Lines
contracts\	100.00	100.00	100.00	100.00
MilestoneBasedToken.sol	100.00	100.00	100.00	100.00
MilestoneBasedVesting.sol	100.00	100.00	100.00	100.00
All Files	**100.00**	**100.00**	**100.00**	**100.00**

Test Results

```
Contract: MilestoneBasedVesting
  Constructor
    ✓ Must fail if passed zero address of token
    ✓ Must set up correctly (58ms)
  addTokensForVesting
    ✓ Must fail if sender isn't owner
    ✓ Must fail if passed amount equal to zero
    ✓ Must fail if sender hasn't approved tokens for vesting
      contract (57ms)
    ✓ Must add tokens for vesting to vesting contract (62ms)
  createVesting
    ✓ Must fail if sender isn't owner
    ✓ Must fail if passed zero address of beneficiary
    ✓ Must fail if amount of vesting equal to 0
    ✓ Must fail if passed incorrect vesting period (45ms)
    ✓ Must fail if passed incorrect vesting period (45ms)
    ✓ Must fail if amount of vesting bigger then allocation (68ms)
    ✓ Must create vesting correctly (60ms)
  createVestingBatch
```

```
    ✓ Must fail if sender isn't owner (38ms)
    ✓ Must fail if passed arrays have different length (38ms)
    ✓ Must create vesting multiple times for different users (94ms)
    ✓ Must create vestings with different type for one user (82ms)
  emergencyWithdraw
    ✓ Must fail if sender isn't owner
    ✓ Must withdraw correct amount of tokens from vesting
      contract (215ms)
    ✓ Must fail if there is no available tokens to withdraw
  testing vesting types calculation and withdraw
    ✓ Withdraw function must fail if user doesn't have vestings
    ✓ All vesting must return 0 if vestings haven't
      started yet (102ms)
    ✓ Vesting type "Marketing" must give small part of vested
      tokens after creation of vesting (55ms)
    ✓ Vestings with types "Founder" and "Rewards" must return 0
      while lock period
    ✓ Vesting with type "Rewards" must return part of tokens after
      lock period
    ✓ Must return correct amount of withdrawable tokens for user
      with multiple vestings after withdraw (141ms)
    ✓ Vesting type "Founder" must return correct amount of
      tokens (433ms)
    ✓ Vestings must return all vested tokens after
      vesting's end (307ms)
  revokeVestingOfUser
    ✓ Must fail if user's vesting is irrevocable (142ms)
    ✓ Must revoke vesting correctly (50ms)
    ✓ Must fail if vesting has been already revoked
    ✓ Must revoke correct amount of user's tokens if user has c
      laimed tokens (103ms)
  burn
    ✓ Must fail if sender doesn't have enough tokens to burn (95ms)
    ✓ Must burn tokens correctly (48ms)
  burnFrom
    ✓ Must fail if sender doesn't have enough allowance (40ms)
    ✓ Must fail if owner doesn't have enough tokens
    ✓ Must burn tokens from another user correctly (47ms)

37 passing (5s)
```

Tests are written by Vidma

Test Coverage

File	% Stmts	% Branch	% Funcs	% Lines
contracts\	100.00	100.00	100.00	100.00
MilestoneBasedToken.sol	100.00	100.00	100.00	100.00
MilestoneBasedVesting.sol	100.00	100.00	100.00	100.00
All Files	**100.00**	**100.00**	**100.00**	**100.00**

Test Results

```
Contract: MilestoneBasedToken
  ✓ has a name
  ✓ has a symbol
  ✓ has 18 decimals
  total supply
    ✓ returns the total amount of tokens
  balanceOf
    when the requested account has no tokens
      ✓ returns zero
    when the requested account has some tokens
      ✓ returns the total amount of tokens
  transfer
    when the recipient is not the zero address
      when the sender does not have enough balance
        ✓ reverts (1198ms)
      when the sender transfers all balance
        ✓ transfers the requested amount (1062ms)
        ✓ emits a transfer event (1063ms)
      when the sender transfers zero tokens
      ✓ transfers the requested amount (1156ms)
      ✓ emits a transfer event (1136ms)
  when the recipient is the zero address
    ✓ reverts (1100ms)
transfer from
  when the token owner is not the zero address
    when the recipient is not the zero address
      when the spender has enough approved balance
        when the token owner has enough balance
          ✓ transfers the requested amount (1088ms)
          ✓ decreases the spender allowance (1068ms)
          ✓ emits a transfer event (1070ms)
          ✓ emits an approval event (1071ms)
        when the token owner does not have enough balance
          ✓ reverts (1063ms)
      when the spender does not have enough approved balance
        when the token owner has enough balance
          ✓ reverts (115ms)
        when the token owner does not have enough balance
          ✓ reverts (1076ms)
    when the recipient is the zero address
      ✓ reverts (1088ms)
  when the token owner is the zero address
    ✓ reverts (1049ms)
approve
  when the spender is not the zero address
    when the sender has enough balance
      ✓ emits an approval event (1029ms)
      when there was no approved amount before
        ✓ approves the requested amount (1081ms)
      when the spender had an approved amount
        ✓ approves the requested amount and replaces the
          previous one (1058ms)
    when the sender does not have enough balance
      ✓ emits an approval event (1049ms)
      when there was no approved amount before
        ✓ approves the requested amount (1048ms)
      when the spender had an approved amount
        ✓ approves the requested amount and replaces the
          previous one (1050ms)
  when the spender is the zero address
    ✓ reverts (1076ms)
```

```
decrease allowance
  when the spender is not the zero address
    when the sender has enough balance
      when there was no approved amount before
        ✓ reverts (1065ms)
      loremipsum
        ✓ emits an approval event (1043ms)
        ✓ decreases the spender allowance subtracting the
          requested amount (1097ms)
        ✓ sets the allowance to zero when all allowance is
          removed (1054ms)
        ✓ reverts when more than the full allowance is
          removed (1072ms)
    when the sender does not have enough balance
      when there was no approved amount before
        ✓ reverts (1070ms)
      when the spender had an approved amount
        ✓ emits an approval event (1065ms)
        ✓ decreases the spender allowance subtracting the
          requested amount (1066ms)
        ✓ sets the allowance to zero when all allowance is
          removed (1048ms)
        ✓ reverts when more than the full allowance is
          removed (1068ms)
  when the spender is the zero address
    ✓ reverts (1067ms)
increase allowance
  when the spender is not the zero address
    when the sender has enough balance
      ✓ emits an approval event (1054ms)
      when there was no approved amount before
        ✓ approves the requested amount (1039ms)
      when the spender had an approved amount
        ✓ increases the spender allowance adding the
          requested amount (1067ms)
    when the sender does not have enough balance
      ✓ emits an approval event (1071ms)
      when there was no approved amount before
        ✓ approves the requested amount (1070ms)
      when the spender had an approved amount
        ✓ increases the spender allowance adding the
          requested amount (1041ms)
  when the spender is the zero address
    ✓ reverts (1092ms)
_transfer
  when the recipient is not the zero address
    when the sender does not have enough balance
      ✓ reverts (1061ms)
    when the sender transfers all balance
      ✓ transfers the requested amount (1089ms)
      ✓ emits a transfer event (1075ms)
    when the sender transfers zero tokens
      ✓ transfers the requested amount (1140ms)
      ✓ emits a transfer event (1108ms)
  when the recipient is the zero address
    ✓ reverts (1047ms)
_approve
  when the spender is not the zero address
    when the sender has enough balance
      ✓ emits an approval event (1048ms)
      when there was no approved amount before
        ✓ approves the requested amount (1048ms)
      when the spender had an approved amount
        ✓ approves the requested amount and replaces the
          previous one (1066ms)
    when the sender does not have enough balance
      ✓ emits an approval event (1047ms)
      when there was no approved amount before
        ✓ approves the requested amount (1062ms)
      when the spender had an approved amount
        ✓ approves the requested amount and replaces the
          previous one (1072ms)
  when the spender is the zero address
    ✓ reverts (1075ms)
burn
  when the given amount is not greater than balance of the sender
    for a zero amount
      ✓ burns the requested amount
      ✓ emits a transfer event
    for a non-zero amount
      ✓ burns the requested amount
      ✓ emits a transfer event
  when the given amount is greater than the balance of the sender
    ✓ reverts (1065ms)
  burnFrom
    on success
      for a zero amount
        ✓ burns the requested amount
        ✓ decrements allowance
        ✓ emits a transfer event
      for a non-zero amount
        ✓ burns the requested amount
        ✓ decrements allowance
        ✓ emits a transfer event
    when the given amount is greater than the balance of the sender
      ✓ reverts (2094ms)
    when the given amount is greater than the allowance
      ✓ reverts (2135ms)
  Ownable
    ✓ has an owner (6ms)
    transfer ownership
      ✓ changes owner after transfer (1059ms)
      ✓ prevents non-owners from transferring (1094ms)
      ✓ guards ownership against stuck state (1064ms)
    lorerenounce ownership
      ✓ loses owner after renouncement (1066ms)
      ✓ prevents non-owners from renouncement (1059ms)

Contract: MilestoneBasedVesting
  MilestoneBasedVesting Deploy Test Cases
    ✓ should deploy with with token setted to the zero
      address (1145ms)
    ✓ should deploy with correct owner (191ms)
    ✓ should deploy with correct tokens (79ms)
    ✓ should deploy with correct initial total tokens
      allocation (110ms)
    ✓ should deploy with correct initial total tokens in
      vesting (79ms)
  MilestoneBasedVesting Owner Test Cases
    ✓ shouldn't create vesting by not the current owner (456ms)
    ✓ shouldn't create vesting with beneficiary with zero
      address (211ms)
    ✓ shouldn't create vesting with zero amount (246ms)
    ✓ shouldn't create vesting with duration equal to zero (266ms)
    ✓ shouldn't create vesting if vesting ends before current
      timestamp (191ms)
    ✓ shouldn't create vesting if amount exceed available amount
      for vesting (255ms)
    ✓ shouldn't create vestings if parameters length mismatch (532ms)
    ✓ shouldn't emergency withdraw tokens by not the current
      owner (222ms)
    ✓ shouldn't emergency withdraw if no available tokens (224ms)
    ✓ shouldn't add tokens to vesting by not the current
      owner (198ms)
    ✓ shouldn't add tokens to vesting if amount is equal 0 (198ms)
    ✓ should add tokens to vesting correctly (454ms)
    ✓ should allow to add tokens to vesting multiple times
      correctly (460ms)
    ✓ should create vesting correctly (595ms)
    ✓ should allow to create multiple vestings for one user
      correctly (558ms)
    ✓ should create batch of vestings correctly (972ms)
    ✓ shouldn't revoke vesting by not the current owner (196ms)
    ✓ shouldn't revoke vesting if vesting is irrevocable (193ms)
    ✓ should revoke vesting correctly (761ms)
    ✓ shouldn't revoke vesting if it was already revoked (207ms)
    ✓ should emergency withdraw all tokens which are not
      in vesting correctly (635ms)
  MilestoneBasedVesting User Test Cases
    ✓ should return correct withdrawable amount before vesting
      start date (313ms)
    ✓ should return correct withdrawable amount when vesting is
      running (367ms)
    ✓ should return correct withdrawable amount when vesting is
      over (203ms)
    ✓ should return correct withdrawable amount when vesting is
      revoked (391ms)
    ✓ shouldn't withdraw if user not in vesting (211ms)
    ✓ should withdraw correctly (883ms)
    ✓ should withdraw multiple times correctly (907ms)
    ✓ should withdraw all tokens from vesting (1411ms)

112 passing (3m)
```

We are delighted to have a chance to work together with milestoneBased team and contribute to their success by reviewing and certifying the security of the smart contracts.

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.