SmartDec

MozoCoin Smart Contracts Security Analysis

This report is public.

Published: July 30, 2018

Abstract

In this report, we consider the security of the MozoCoin project. Our task is to find and describe security issues in the smart contracts of the platform.

Disclaimer

The audit does not give any warranties on the security of the code. One audit cannot be considered enough. We always recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts. Besides, security audit is not an investment advice.

Summary

In this report, we have considered the security of MozoCoin smart contracts. We performed our audit according to the procedure described below.

The audit has shown two critical issues. We do not recommend using this contract before they are fixed. Besides, several medium and low severity issues were found. We recommend addressing them too.

General recommendations

Our audit has revealed two critical issues: Code logic violation and Vulnerable modifier. These are serious issues; moreover, they can be exploited by an external attacker. Besides, we recommend limiting Power of the owner, fixing Documentation mismatch and Possible front-running attack.

In addition, if the developers decide to improve the code quality, we recommend fixing Redundant code and Lack of documentation issue, implementing Tests and Deploy scripts.

However, these issues are minor and do not influence code operation.

Checklist

Security

The audit has shown vulnerabilities. Here by vulnerabilities we mean security issues that can be exploited by an external attacker. This does not include low severity issues, documentation mismatches, overpowered contract owner, and some other kinds of bugs.

Compliance with the documentation

The audit has shown some discrepancies between the code and the provided documentation. See Documentation mismatch issue.

ERC20 compliance

We have checked ERC20 compliance during the audit. The audit has shown that HoldToken contract is fully ERC20 compliant.

ERC20 MUST

The audit has shown no ERC20 “MUST” requirements violations.

ERC20 SHOULD

The audit has shown no ERC20 “SHOULD” requirements violations.

Tests

The audit has shown that the code is not covered with tests. See No test issue.

The text below is for technical use; it details the statements made in Summary, General recommendations and Checklist.

Procedure

In our audit, we consider the following crucial features of the smart contract code:

1. Whether the code is secure.
2. Whether the code corresponds to the documentation (including whitepaper).
3. Whether the code meets best practices in efficient use of gas, code readability, etc.

We perform our audit according to the following procedure:

- automated analysis
    - we scan project's smart contracts with our own Solidity static code analyzer SmartCheck
    - we scan project's smart contracts with several publicly available automated Solidity analysis tools such as Remix and Solhint
    - we manually verify (reject or confirm) all the issues found by tools
- manual audit
    - we manually analyze smart contracts for security vulnerabilities
    - we check smart contracts logic and compare it with the one described in the whitepaper
    - we check ERC20 compliance
- report
    - we reflect all the gathered information in the report

Checked vulnerabilities

We have scanned MozoCoin smart contracts for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that we considered (the full list includes them but is not limited to them):

- Reentrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with (Unexpected) Throw
- DoS with (Unexpected) revert
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- Unsafe type inference
- Implicit visibility level
- Address hardcoded
- Using delete for arrays
- Integer overflow/underflow
- Locked money
- Private modifier
- Revert/require functions
- Using var
- Visibility
- Using blockhash
- Using SHA3
- Using suicide
- Using throw
- Using inline assembly

Project overview

Project description

In our analysis we consider MozoCoin documentation (Mozo-Smart contracts.docx, sha1sum: fcf0353b9b0d564ad0b4d61cda0f114447cd4172) and smart contracts code (version on commit ebf6bf0a011034e4342d04e7f1c00c2c72742e70).

Project architecture

For the audit, we have been provided with the following set of files:

- **Agency.sol**
- **Agentable.sol**
- **ChainCoOwner.sol**
- **ChainOwner.sol**
- **Closable.sol**
- **ContributionBonus.sol**
- **ICO.sol**
- **InvestmentDiscount.sol**
- **MozoSaleToken.sol**
- **MozoToken.sol**
- **Owner.sol**
- **OwnerERC20.sol**
- **Referral.sol**
- **RevocableVested.sol**
- **Sale.sol**
- **Timeline.sol**
- **TimelineBonus.sol**
- **TimeLock.sol**
- **Upgradable.sol**

Files successfully compile with solc command (with some warnings, see Compilation output in Appendix).

Files contain the following contracts:

- **Agency** (inherits **Sale**, **Timeline**, and **Agentable** contracts)
- **Agentable**
- **ChainCoOwner** (inherits **ChainOwner** contract)
- **ChainOwner**
- **Closable**
- **ContributionBonus**

- **ICO** (contract with functions declarations)
- **InvestmentDiscount** (inherits **Sale**, **ContributionBonus**, and **Timeline** contracts)
- **MozoSaleToken** (inherits **Timeline**, **ChainCoOwner**, and **ICO** contracts and **BasicToken** contract from **OpenZeppelin** library)
- **MozoToken** (inherits **OwnerERC20** contract and **StandardToken** contract from **OpenZeppelin** library)
- **Owner** (contract with functions declarations)
- **OwnerERC20** (inherits **Owner** contract and **ERC20Basic** contract from **OpenZeppelin** library)
- **Referral** (inherits **Sale**, **ContributionBonus**, **Timeline**, **Agentable**, and **Upgradable** contracts)
- **RevocableVested** (inherits **Sale**, **ContributionBonus**, **Timeline**, **Agentable**, **Upgradable** contracts)
- **Sale** (inherits **ChainOwner** and **Closable** contracts)
- **Timeline**
- **TimelineBonus** (inherits **Sale** and **Timeline** contracts)
- **TimeLock** (inherits **Timeline**, **ChainOwner**, and **Agentable** contracts)
- **Upgradable** (contract with functions declarations)

The total volume of audited files is 953 lines of Solidity code.

Automated analysis

We used several publicly available automated Solidity analysis tools. Here are the combined results of SmartCheck, Solhint, and Remix.

All the issues found by tools were manually checked (rejected or confirmed).

False positives are constructions that were discovered by the tools as vulnerabilities but do not consist a security threat.

True positives are constructions that were discovered by the tools as vulnerabilities and can actually be exploited by attackers or lead to incorrect contracts operation.

Cases when these issues lead to actual bugs or vulnerabilities are described in the next section.

Tool	Rule	False positives	True positives
Remix	Fallback function requires too much gas		5
	Gas requirement of function high: infinite	67	2
	Potential Violation of Checks-Effects-Interaction pattern	6	1
	Should be constant but is not	1	
	Use of "now"	3	
Total Remix		77	8
SmartCheck	Address Hardcoded	1	
	Dos With Revert	10	
	Erc20 Transfer Should Throw	1	
	Gas Limit And Loops	7	3
	Locked Money	2	

	Private Modifier	25	
	Redundant Fallback Reject		2
	Reentrancy External Call	10	1
	Unchecked Math	25	
	Underflow Overflow		9
	Visibility		2
Total SmartCheck		81	17
Solhint	Avoid to make time-based decisions in your business logic	11	
	Avoid to use inline assembly	1	
	Constructors should use the new constructor keyword		14
	Explicitly mark visibility in function		1
	Explicitly mark visibility of state		1
	Possible reentrancy vulnerabilities	4	
Total Solhint		16	16
Overall Total		174	41

Manual analysis

The contracts were completely manually analyzed, their logic was checked and compared with the one described in the documentation. Besides, the results of the automated analysis were manually verified. All confirmed issues are described below.

Critical issues

Critical issues seriously endanger smart contracts security. We highly recommend fixing them.

Code logic violation

`close()` function has `public` visibility modifier (**Closable.sol**, line 21):

```
function close() public {
```

This function allows to change `isClosed` variable to `true`. Thus, everyone can close the sale using this `public` function. We highly recommend fixing this vulnerability.

Vulnerable modifier: onlySameChain

We found that `onlySameChain` modifier is vulnerable (**MozoSaleToken.sol**, line 102):

```
require(sm.owner() == owner());
```

Attacker's contract can implement `owner()` function that returns required address.
This implies that `transferByEth()` function in **MozoSaleToken** contract is accessible by an attacker. Thus, the attacker can set `totalCapInWei` variable to 2**256-1 and it will be impossible to buy tokens. We highly recommend fixing this vulnerability.

Medium severity issues

Medium issues can influence smart contracts operation in current implementation. We highly recommend addressing them.

Overpowered owner

The token contract owner has the following powers:

1. **RevocableVested.sol**, line 80:

```
function revoke() public onlyOwner notClosed
```

The owner can always call `revoke()` function and return back remaining tokens. During `cliff` period, the owner can return all the tokens back.

2. **MozoSaleToken.sol**, line 361:

```
function release() public onlyOwnerOrCoOwner
```

`release()` function can be called at any time by the `owner` or `coowners`. It can lead to premature exchange of SMZO tokens to MOZO tokens.

3. **Sale.sol**, line 82:

```
function withdraw() public onlyOwner
```

Sale.sol, line 90:

```
function returnToken() public onlyOwner notClosed
```

These functions have `internal` analogs called from `release()` function. That is why we recommend removing `public` analogs in order to restrict owner power.

In the current implementation, the system depends heavily on the owner of the contract. In this case, there are scenarios that may lead to undesirable consequences for investors, e.g. if the owner's private keys become compromised.
Thus, we recommend designing contracts in a trustless manner.

Documentation mismatch

There is a discrepancy between the smart contracts code and the documentation:

- According to the documentation ("Mozo-Smart contracts.docx", p. 22, item 5.1.2):

```
"If hard cap is reached: burn all unsold tokens"
```

However, all these tokens remain at the `owner` and `coowners`' wallets. Thus, `owner` and `coowners` can transfer these tokens to another account and exchange them to MOZO tokens using `release()` function.
The same situation happens in case hard cap is not reached: `owner` and `coowners` can transfer their tokens to new addresses and, thus, these tokens will be counted as sold and exchanged to MOZO tokens.

We recommend either improving contracts' functionality so that it matches the documentation or modifying the documentation in order to avoid any discrepancies.

Possible front-running

MozoSaleToken.sol, line 190:

```
function addAddressToWhitelist(address _address)
onlyOwnerOrCoOwner public returns (bool success)
```

After calling `addAddressesToWhitelist(user)`, any user can call `transfer()` function with `_value` greater than `AML_THRESHOLD`. Thus, this amount will be added to `pendingAmounts[user]` variable and `addAddressesToWhitelist()` function will send SMZO tokens to user from owner or coowner's address. We recommend taking into account this vulnerability.

Low severity issues

Low severity issues can influence smart contracts operation in future versions of code. We recommend taking them into account.

Lack of documentation

MozoSaleToken.sol, line 380:

```
function bonusToken(address[] _recipients, uint[] _amount)
public onlyOwnerOrCoOwner onlyStopping
```

`bonusToken()` function is not documented. Thus, it is not clear how this function will be used in the workflow. Moreover, bonuses are sent without `AML_THRESHOLD` restriction. We recommend completing the documentation by adding the relevant information.

Vulnerable modifier: validOwner

We found that `validOwner` modifier is vulnerable (**ChainOwner.sol**, line 40):

```
require(ico.owner() == _smzoToken.owner());
```

Attacker's contract can implement `owner()` function that returns required address. Thus, `upgrade()` function in **Referral** contract (line 105) is accessible by an attacker. However, it will only break view functions and will not affect code execution. We highly recommend taking into account this vulnerability.

Constructor keyword

Defining constructor as function of the same name as the contract is deprecated. However, in considered project, constructors of all the contracts are implemented in the deprecated way. We highly recommend not using deprecated constructions and following the best practices instead.

Redundant code

The following lines are redundant:

1. **Referral.sol**, line 21:

   ```
   uint public level;
   ```

   This variable is redundant as it is never used. Moreover, `public` modifier is redundant as getter function is implemented explicitly. We recommend using `internal` modifier or removing getter function.
2. **Timeline.sol**, line 23:

   ```
   modifier notEnded()
   ```

   This modifier is redundant as it is never used.
3. **MozoSaleToken.sol**, line 8:

```
import "./Upgradable.sol";
```

This import is redundant as it is never used.

4. **MozoSaleToken.sol**, line 450:

```
transferredIndex = i - 1;
```

This line is redundant. In case this line is executed, the last iteration of the loop will be repeated in the next loop.

5. **Sale.sol**, lines 18, 21:

```
ICO public smzoToken;
uint public sold;
```

These variables have `public` modifier, however it is redundant as getter function is implemented explicitly. We recommend using `internal` modifier or removing getter function.

6. **Sale.sol**, lines 63-67:

```
bool ret = smzoToken.transferByEth(msg.sender, msg.value,
tokenToFund);
if (ret) {
      sold = sold.add(tokenToFund);
}
return ret;
```

`ret` variable is redundant, because `transferByEth()` function always returns `true`.

7. **RevocableVested.sol**, line 89:

```
if (token > total)
```

This case is impossible since `token` variable is `total*(time/vestedDuration)`, and `time <= vestedDuration`.

8. **MozoSaleToken.sol**, lines 328,354:

```
return false;
```

`transfer()` and `transferByEth()` functions always return `true`, therefore these lines are redundant.

9. **Agency.sol**, lines 54-55:

```
require(_minContribution >= 0);
require(_bonusPercentage >= 0);
```

ContributionBonus.sol, lines 36-37:

```
require(_weiContributionTranches[0] >= 0);
require(_bonusPercentageTranches[0] >= 0);
```

TimelineBonus.sol, lines 63-65:

```
require(_bonusPercentage >= 0);
require(_minContribution >= 0);
```

```
require(_minContributionAfterBonus >= 0);
```

RevocableVested.sol, lines 51-52:

```
require(cliff >= 0);
require(_vestedDuration >= 0);
```

Variables of `uint` type are always more or equal than zero, thus these lines are redundant.

10. **RevocableVested.sol**, lines 72-74:

```
function() public payable {
      revert();
}
```

TimeLock.sol, lines 53-55:

```
function() public payable {
      revert();
}
```

Contracts should reject unexpected payments. Before Solidity 0.4.0, it was done manually. Starting from Solidity 0.4.0, contracts without `fallback` function automatically revert payments, making the code above redundant.

11. **MozoSaleToken.sol**, line 70:

```
modifier onlyWhitelisted() {
```

This modifier is redundant as it is never used.

We highly recommend removing redundant code in order to improve code readability and transparency and decrease cost of deployment and execution.

Visibility level

The following variables/functions have an implicit visibility level:

- **MozoSaleToken.sol**, line 37:

```
mapping(address => bool) bonus_transferred_repicients;
```

- **MozoSaleToken.sol**, line 250:

```
function setStop() onlyOwnerOrCoOwner {
```

We recommend specifying visibility levels (`public`, `private`, `external`, `internal`) explicitly and correctly in order to improve code readability.

Not implemented functionality

There are several logic steps in documentation that are not implemented in the contract code:

- Figures 3,4: steps 9, 11 - if the hard cap is reached, it is necessary to release the sale
- Figure 5: step 7 - creation of **TimeLock** contract

We highly recommend improving smart contract's code by adding this functionality in new versions.

Fallback function abuse

Several contracts (**Sale.sol, TimelineBonus.sol, Referral.sol, InvestmentDiscount.sol, Agency.sol**) contain fallback function:

```
function() public payable {
      buyToken();
}
```

We recommend implementing additional check: `require(msg.data.length == 0)`. It eliminates the possibility of accidental fallback function call.

No deploy script

There are no deploy scripts. Bugs and vulnerabilities in deploy appear often and severely endanger contracts' security.
We highly recommend not only developing deploy scripts very carefully but also performing audit of them.

No tests

The provided code does not contain tests. Testing is crucial for code security. We highly recommend not only covering the code with tests but also making sure that the coverage is sufficient.

OpenZeppelin library

The project uses OpenZeppelin library. However, OpenZeppelin files are added to the repo instead of being connected via npm.
We highly recommend using standard and latest versions of library contracts.

Typo

We found several typos:

- **MozoToken.sol**, line 18:

  ```
  //token symbol
  ```

  There should be `decimals` instead of `symbol`.
- **Referral.sol**, line 130:

  ```
  return PACKAGE_5_TOKENS_REQUIRE;
  ```

  There should be `PACKAGE_5_BONUS` variable instead of `PACKAGE_5_TOKENS_REQUIRE` variable.

We highly recommend removing all found typos in the contract code.

Notes

Gas limit and loops

The following loops traverse through arrays of variable length:

- **MozoSaleToken.sol**, line 211:

```
for (uint i = 0; i < length; i++) {
```

- **MozoSaleToken.sol**, line 387:

```
for (i = 0; i < len; i++) {
```

- **ChainCoOwner.sol**, line 29:

```
for (uint i=0; i < len; i++) {
```

The traversed arrays are passed as functions parameters. Therefore, if there are too many items in these arrays, the execution of the corresponding functions will fail due to an out-of-gas exception.
In these cases, we recommend separating the calls into several transactions.

Max participation

We found that it is possible to buy more than `AML_THRESHOLD` tokens for non-whitelisted addresses. One just need to buy tokens from different addresses. However, it will deprive the buyer of the bonus in **investmentDiscount** contract logic. We recommend taking into account this opportunity.

Checks-Effects-Interaction

There is a Checks-Effects-Interaction violation (**Sale.sol**, line 63):

```
bool ret = smzoToken.transferByEth(msg.sender, msg.value,
tokenToFund);
if (ret) {
      sold = sold.add(tokenToFund);
}
```

In this case the violation does not lead to actual vulnerabilities. However, we highly recommend following best practices including Checks-Effects-Interactions pattern since it helps to avoid many serious vulnerabilities.

This analysis was performed by SmartDec.

Alexander Seleznev, Chief Business Development Officer
Evgeniy Marchenko, Lead Developer
Pavel Kondratenkov, Analyst
Igor Sobolev, Analyst
Alexander Drygin, Analyst

Sergey Pavlin, Chief Operating Officer 

July 30, 2018

Appendix

Compilation output

```
Compiling ./contracts/Agency.sol...
Compiling ./contracts/Agentable.sol...
Compiling ./contracts/ChainCoOwner.sol...
Compiling ./contracts/ChainOwner.sol...
Compiling ./contracts/Closable.sol...
Compiling ./contracts/ContributionBonus.sol...
Compiling ./contracts/ICO.sol...
Compiling ./contracts/InvestmentDiscount.sol...
Compiling ./contracts/MozoSaleToken.sol...
Compiling ./contracts/MozoToken.sol...
Compiling ./contracts/Owner.sol...
Compiling ./contracts/OwnerERC20.sol...
Compiling ./contracts/Referral.sol...
Compiling ./contracts/RevocableVested.sol...
Compiling ./contracts/Sale.sol...
Compiling ./contracts/TimeLock.sol...
Compiling ./contracts/Timeline.sol...
Compiling ./contracts/TimelineBonus.sol...
Compiling ./contracts/Upgradable.sol...
Compiling ./../open-zeppelin/contracts/math/SafeMath.sol...
Compiling ./../open-
zeppelin/contracts/token/ERC20/BasicToken.sol...
Compiling ./../open-
zeppelin/contracts/token/ERC20/ERC20.sol...
Compiling ./../open-
zeppelin/contracts/token/ERC20/ERC20Basic.sol...
Compiling ./../open-
zeppelin/contracts/token/ERC20/StandardToken.sol...

Compilation warnings encountered:

/Mozo-SC/compile/contracts/ChainOwner.sol:19:5: Warning:
Defining constructors as functions with the same name as the
contract is deprecated. Use "constructor(...) { ... }"
instead.
    function ChainOwner(OwnerERC20 _parent) internal {
    ^ (Relevant source part starts here and spans across
multiple lines).
,/Mozo-SC/compile/contracts/Sale.sol:27:5: Warning: Defining
```

```
constructors as functions with the same name as the contract
is deprecated. Use "constructor(...) { ... }" instead.
    function Sale(ICO _ico) internal ChainOwner(_ico)
onlyOwner() {
    ^ (Relevant source part starts here and spans across
multiple lines).
,/Mozo-SC/compile/contracts/Timeline.sol:44:5: Warning:
Defining constructors as functions with the same name as the
contract is deprecated. Use "constructor(...) { ... }"
instead.
    function Timeline(
    ^ (Relevant source part starts here and spans across
multiple lines).
,/Mozo-SC/compile/contracts/Agentable.sol:35:5: Warning:
Defining constructors as functions with the same name as the
contract is deprecated. Use "constructor(...) { ... }"
instead.
    function Agentable(address _agency) internal
onlyWalletAddress(_agency) {
    ^ (Relevant source part starts here and spans across
multiple lines).
,/Mozo-SC/compile/contracts/Agency.sol:40:5: Warning: Defining
constructors as functions with the same name as the contract
is deprecated. Use "constructor(...) { ... }" instead.
    function Agency(
    ^ (Relevant source part starts here and spans across
multiple lines).
,/Mozo-SC/compile/contracts/ChainCoOwner.sol:23:5: Warning:
Defining constructors as functions with the same name as the
contract is deprecated. Use "constructor(...) { ... }"
instead.
    function ChainCoOwner(OwnerERC20 _parent, address[]
_coOwner) ChainOwner(_parent) internal {
    ^ (Relevant source part starts here and spans across
multiple lines).
,/Mozo-SC/compile/contracts/ContributionBonus.sol:25:5:
Warning: Defining constructors as functions with the same name
as the contract is deprecated. Use "constructor(...) { ... }"
instead.
    function ContributionBonus(
    ^ (Relevant source part starts here and spans across
multiple lines).
,/Mozo-SC/compile/contracts/InvestmentDiscount.sol:25:5:
Warning: Defining constructors as functions with the same name
as the contract is deprecated. Use "constructor(...) { ... }"
instead.
    function InvestmentDiscount(
```

```
    ^ (Relevant source part starts here and spans across
multiple lines).
,/Mozo-SC/compile/contracts/MozoSaleToken.sol:117:5: Warning:
Defining constructors as functions with the same name as the
contract is deprecated. Use "constructor(...) { ... }"
instead.
    function MozoSaleToken(
    ^ (Relevant source part starts here and spans across
multiple lines).
,/Mozo-SC/compile/contracts/MozoToken.sol:34:5: Warning:
Defining constructors as functions with the same name as the
contract is deprecated. Use "constructor(...) { ... }"
instead.
    function MozoToken(uint256 _totalSupply) public {
    ^ (Relevant source part starts here and spans across
multiple lines).
,/Mozo-SC/compile/contracts/Referral.sol:73:5: Warning:
Defining constructors as functions with the same name as the
contract is deprecated. Use "constructor(...) { ... }"
instead.
    function Referral(
    ^ (Relevant source part starts here and spans across
multiple lines).
,/Mozo-SC/compile/contracts/RevocableVested.sol:37:5: Warning:
Defining constructors as functions with the same name as the
contract is deprecated. Use "constructor(...) { ... }"
instead.
    function RevocableVested(
    ^ (Relevant source part starts here and spans across
multiple lines).
,/Mozo-SC/compile/contracts/TimeLock.sol:29:5: Warning:
Defining constructors as functions with the same name as the
contract is deprecated. Use "constructor(...) { ... }"
instead.
    function TimeLock(OwnerERC20 _mozoToken, address
_beneficiary, uint _total, uint _start, uint _lockPeriod)
    ^ (Relevant source part starts here and spans across
multiple lines).
,/Mozo-SC/compile/contracts/TimelineBonus.sol:48:5: Warning:
Defining constructors as functions with the same name as the
contract is deprecated. Use "constructor(...) { ... }"
instead.
    function TimelineBonus(
    ^ (Relevant source part starts here and spans across
multiple lines).
,/Mozo-SC/compile/contracts/MozoSaleToken.sol:250:5: Warning:
No visibility specified. Defaulting to "public".
```

```
    function setStop() onlyOwnerOrCoOwner {
    ^ (Relevant source part starts here and spans across
multiple lines).
,/Mozo-SC/compile/contracts/Sale.sol:130:5: Warning: Function
state mutability can be restricted to pure
    function _holdTokens() internal view returns(uint) {
    ^ (Relevant source part starts here and spans across
multiple lines).
,/Mozo-SC/compile/contracts/Sale.sol:134:5: Warning: Function
state mutability can be restricted to pure
    function _bonusProcess() internal {
    ^ (Relevant source part starts here and spans across
multiple lines).

Writing artifacts to ./build/contracts
```

Solhint output

```
Agency.sol
  40:5  warning  Constructors should use the new constructor
keyword  constructor-syntax

Agentable.sol
  16:45  error    Open bracket must be on same line. It must
be indented by other constructions by space  bracket-align
  20:9   warning  Avoid to use inline assembly. It is
acceptable only in rare cases                       no-inline-
assembly
  35:5   warning  Constructors should use the new constructor
keyword                                      constructor-syntax

ChainCoOwner.sol
  23:67  error    Visibility modifier must be first in list of
modifiers                             visibility-modifier-order
  23:5   warning  Constructors should use the new constructor
keyword                                 constructor-syntax
  56:5   error    Function order is incorrect, public function
can not go after internal function  func-order

ChainOwner.sol
  19:5  warning  Constructors should use the new constructor
keyword  constructor-syntax

ContributionBonus.sol
```

```
  25:5   warning  Constructors should use the new constructor
keyword                 constructor-syntax
  31:3   error    Expected indentation of 8 spaces but found
2                       indent
  31:1   error    Mixed tabs and spaces. Allowed only
spaces                          no-mix-tabs-and-spaces
  32:3   error    Expected indentation of 8 spaces but found
2                       indent
  32:1   error    Mixed tabs and spaces. Allowed only
spaces                          no-mix-tabs-and-spaces
  33:1   error    Mixed tabs and spaces. Allowed only
spaces                          no-mix-tabs-and-spaces
  34:3   error    Expected indentation of 8 spaces but found
2                       indent
  34:1   error    Mixed tabs and spaces. Allowed only
spaces                          no-mix-tabs-and-spaces
  35:1   error    Mixed tabs and spaces. Allowed only
spaces                          no-mix-tabs-and-spaces
  35:3   error    Expected indentation of 8 spaces but found
2                       indent
  36:6   error    Expected indentation of 8 spaces but found
5                       indent
  36:1   error    Mixed tabs and spaces. Allowed only
spaces                          no-mix-tabs-and-spaces
  37:6   error    Expected indentation of 8 spaces but found
5                       indent
  37:1   error    Mixed tabs and spaces. Allowed only
spaces                          no-mix-tabs-and-spaces
  40:3   error    Expected indentation of 8 spaces but found
2                       indent
  40:1   error    Mixed tabs and spaces. Allowed only
spaces                          no-mix-tabs-and-spaces
  41:1   error    Mixed tabs and spaces. Allowed only
spaces                          no-mix-tabs-and-spaces
  41:7   error    Expected indentation of 12 spaces but found
6                      indent
  42:1   error    Mixed tabs and spaces. Allowed only
spaces                          no-mix-tabs-and-spaces
  42:7   error    Expected indentation of 12 spaces but found
6                      indent
  43:3   error    Expected indentation of 8 spaces but found
2                       indent
  43:1   error    Mixed tabs and spaces. Allowed only
spaces                          no-mix-tabs-and-spaces
  44:3   error    Expected indentation of 8 spaces but found
2                       indent
  44:1   error    Mixed tabs and spaces. Allowed only
```

```
spaces                              no-mix-tabs-and-spaces
  45:1   error    Mixed tabs and spaces. Allowed only
spaces                              no-mix-tabs-and-spaces
  45:3   error    Expected indentation of 8 spaces but found
2                          indent
  52:5   error    Definitions inside contract / library must
be separated by one line  separate-by-one-line-in-contract
  55:14  error    Statement indentation is incorrect. Required
space after ;           statement-indent

InvestmentDiscount.sol
  25:5  warning  Constructors should use the new constructor
keyword  constructor-syntax
  37:5  warning  Code contains empty
block                               no-empty-blocks

MozoSaleToken.sol
   19:28  error    Constant name must be in capitalized
SNAKE_CASE                            const-name-snakecase
   22:28  error    Constant name must be in capitalized
SNAKE_CASE                            const-name-snakecase
   25:27  error    Constant name must be in capitalized
SNAKE_CASE                            const-name-snakecase
   37:5   warning  Explicitly mark visibility of
state                                          state-visibility
   37:30  error    Variable name must be in
mixedCase                                        var-name-
mixedcase
  117:5   error    Definitions inside contract / library must
be separated by one line      separate-by-one-line-in-contract
  117:5   warning  Constructors should use the new constructor
keyword                        constructor-syntax
  190:54  error    Visibility modifier must be first in list
of modifiers                     visibility-modifier-order
  209:60  error    Visibility modifier must be first in list
of modifiers                     visibility-modifier-order
  224:59  error    Visibility modifier must be first in list
of modifiers                     visibility-modifier-order
  237:65  error    Visibility modifier must be first in list
of modifiers                     visibility-modifier-order
  250:24  warning  Explicitly mark visibility in
function                                       func-visibility
  351:13  warning  Possible reentrancy vulnerabilities. Avoid
state changes after transfer  reentrancy
  393:9   warning  Possible reentrancy vulnerabilities. Avoid
state changes after transfer  reentrancy
  394:9   warning  Possible reentrancy vulnerabilities. Avoid
```

```
state changes after transfer  reentrancy
  450:9   warning  Possible reentrancy vulnerabilities. Avoid
state changes after transfer  reentrancy

MozoToken.sol
  13:28  error    Constant name must be in capitalized
SNAKE_CASE                      const-name-snakecase
  16:28  error    Constant name must be in capitalized
SNAKE_CASE                      const-name-snakecase
  19:27  error    Constant name must be in capitalized
SNAKE_CASE                      const-name-snakecase
  34:5   error    Definitions inside contract / library must
be separated by one line  separate-by-one-line-in-contract
  34:5   warning  Constructors should use the new constructor
keyword                 constructor-syntax

Referral.sol
   73:5  warning  Constructors should use the new constructor
keyword                 constructor-syntax
   95:5  error    Definitions inside contract / library must
be separated by one line  separate-by-one-line-in-contract
  154:5  error    Definitions inside contract / library must
be separated by one line  separate-by-one-line-in-contract
  195:5  error    Definitions inside contract / library must
be separated by one line  separate-by-one-line-in-contract

RevocableVested.sol
   37:5   warning  Constructors should use the new constructor
keyword                              constructor-syntax
   72:5   error    Function order is incorrect, fallback
function can not go after public function  func-order
   97:9   error    Statement indentation is incorrect.
Required space after if                      statement-indent
  117:13  warning  Avoid to make time-based decisions in your
business logic                        not-rely-on-time
  126:21  warning  Avoid to make time-based decisions in your
business logic                        not-rely-on-time

Sale.sol
   27:5   warning  Constructors should use the new constructor
keyword                 constructor-syntax
   82:5   error    Definitions inside contract / library must
be separated by one line  separate-by-one-line-in-contract
  134:39  warning  Code contains empty
block                                          no-empty-
blocks
```

```
TimelineBonus.sol
  28:13  warning  Avoid to make time-based decisions in your
business logic  not-rely-on-time
  31:17  warning  Avoid to make time-based decisions in your
business logic  not-rely-on-time
  48:5   warning  Constructors should use the new constructor
keyword        constructor-syntax
  92:13  warning  Avoid to make time-based decisions in your
business logic  not-rely-on-time

Timeline.sol
  19:17  warning  Avoid to make time-based decisions in your
business logic            not-rely-on-time
  24:17  warning  Avoid to make time-based decisions in your
business logic            not-rely-on-time
  29:17  warning  Avoid to make time-based decisions in your
business logic            not-rely-on-time
  34:17  warning  Avoid to make time-based decisions in your
business logic            not-rely-on-time
  34:37  warning  Avoid to make time-based decisions in your
business logic            not-rely-on-time
  44:5   error    Definitions inside contract / library must
be separated by one line  separate-by-one-line-in-contract
  44:5   warning  Constructors should use the new constructor
keyword                  constructor-syntax
  50:30  warning  Avoid to make time-based decisions in your
business logic            not-rely-on-time

TimeLock.sol
  29:5  warning  Constructors should use the new constructor
keyword                               constructor-syntax
  53:5  error    Function order is incorrect, fallback
function can not go after public function  func-order

Upgradable.sol
  8:1  error  Definition must be surrounded with two blank
line indent  two-lines-top-level-separator

✖ 86 problems (52 errors, 34 warnings)
```

Solium output

```
mozo/contracts/Agentable.sol
  20:8    error    Avoid using Inline
```

```
Assembly.    security/no-inline-assembly

mozo/contracts/ChainCoOwner.sol
  29:13    warning    Assignment operator must have exactly
single space on both sides of it.    operator-whitespace
  42:13    warning    Assignment operator must have exactly
single space on both sides of it.    operator-whitespace

mozo/contracts/ContributionBonus.sol
  31:2    error      Only use indent of 8
spaces.                                                  indentat
ion
  32:2    error      Only use indent of 8
spaces.                                                  indentat
ion
  34:2    error      Only use indent of 8
spaces.                                                  indentat
ion
  35:2    error      Only use indent of 8
spaces.                                                  indentat
ion
  36:5    error      Only use indent of 8
spaces.                                                  indentat
ion
  37:5    error      Only use indent of 8
spaces.                                                  indentat
ion
  40:2    error      Only use indent of 8
spaces.                                                  indentat
ion
  40:7    warning    Assignment operator must have exactly
single space on both sides of it.    operator-whitespace
  43:0    error      Only use indent of 8
spaces.                                                  indentat
ion
  44:2    error      Only use indent of 8
spaces.                                                  indentat
ion
  45:2    error      Only use indent of 8
spaces.                                                  indentat
ion

mozo/contracts/MozoSaleToken.sol
  250:4    error    No visibility specified explicitly for
setStop function.    security/enforce-explicit-visibility

mozo/contracts/RevocableVested.sol
```

```
  117:12    warning    Avoid using 'now' (alias to
'block.timestamp').    security/no-block-members
  126:20    warning    Avoid using 'now' (alias to
'block.timestamp').    security/no-block-members

mozo/contracts/Timeline.sol
  19:16    warning    Avoid using 'now' (alias to
'block.timestamp').    security/no-block-members
  24:16    warning    Avoid using 'now' (alias to
'block.timestamp').    security/no-block-members
  29:16    warning    Avoid using 'now' (alias to
'block.timestamp').    security/no-block-members
  34:16    warning    Avoid using 'now' (alias to
'block.timestamp').    security/no-block-members
  34:36    warning    Avoid using 'now' (alias to
'block.timestamp').    security/no-block-members
  50:29    warning    Avoid using 'now' (alias to
'block.timestamp').    security/no-block-members

mozo/contracts/TimelineBonus.sol
  28:12    warning    Avoid using 'now' (alias to
'block.timestamp').    security/no-block-members
  31:16    warning    Avoid using 'now' (alias to
'block.timestamp').    security/no-block-members
  92:12    warning    Avoid using 'now' (alias to
'block.timestamp').    security/no-block-members

✖ 12 errors, 14 warnings found.
```