SmartDec

Debitum Smart Contracts Security Audit

Abstract

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

Procedure

In our analysis we consider Debitum whitepaper (version on 27 October 2017, sha1sum 3e22f59ee30af02d2acba0294c315c40d0da8a19 *Debitum-TGE-Document-v2.0.pdf) and smart contracts code (version on commit bdbe014).
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, Oyente and Securify (beta version since full version was unavailable at the moment this report was made)
  - 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 run tests
- report
  - we report all the issues found to the developer during the audit process
  - we reflect all the gathered information in the report

Disclaimer

The audit does not give any warranties on the security of the code. One audit can not be considered enough. We always recommend proceeding to several independent audits and a public bug bounty program to ensure the security of the smart contracts.

Checked vulnerabilities

We have scanned Debitum 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 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

Automated Analysis

We used several publicly available automated Solidity analysis tools. Here are the combined results of their analysis. All the issues found by tools were manually checked (rejected or confirmed).

Tool	Vulnerability	false positives	true positives
SmartCheck	Address hardcoded	12	2
	Constant functions	6	
	Dos with revert	1	
	ERC20 approve		1
	ERC20 transfer	2	
	ERC20 transfer return false		4
	Gas limit and loops	9	7
	Pragmas version		9
	Private modifier		3
	Reentrancy external call	27	4
	Should be pure but is not	14	
	Should be view but is not	19	1
	Timestamp dependence	23	22
	Unchecked math	19	16
	Visibility		20
Total SmartCheck		132	89
Remix	Could potentially lead to re-entrancy vulnerability	1	1
	Fallback function requires too much gas	1	1
	Function declared as view but potentially is not	10	
	Function state mutability can be restricted to pure		4
	Function state mutability can be restricted to view		2
	Gas requirement unknown or not constant	42	8
	Is constant but potentially should not be	3	
	No visibility specified		29
	Potentially should be constant but is not	1	2
	Inline assembly		1
	Unused function parameter.	4	4
	use of "now"	9	11

	Variables have very similar names	11	2
Total Remix		82	65
Oyente	Assertion Failure		16
Total Oyente			16
Overall total		214	170

Securify* — beta version, full version is unavailable.

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

Manual Analysis

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

High severity issues

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

Not a multisignature wallet

The `changeRequirement` function at MultisigWallet.sol, line 271 allows any owner to change the number of owners required for transaction confirmation:

```
function changeRequirement(uint _required)
      public
      ownerExists(msg.sender)
      validRequirement(owners.length, _required)
{
      required = _required;
      RequirementChange(_required, now);
}
```

One of the owners will be able to change the required number of owners and confirm transactions by himself. This ruins the whole idea of multisignature wallet.
We highly recommend adding a requirement on the number of signatures for access to this functionality.
The issue has been fixed by the developer and is not present on commit 6f701a0.

No tokens distribution

The contracts do not implement tokens distribution (to the team and supporters). This is a discrepancy with the whitepaper, p. 13:
"10% of Debitum tokens will be directly distributed to Debitum Network advisors and contributors. ... 15% of Debitum tokens will be directly distributed to the Debitum Network team - … . The other 15% of Debitum tokens (150,000,000 tokens) will be stored in Reserve."
Tokens are simply sent to one address (Crowdsale.sol, line 352):

```
token.transfer(address(wallet), reserve);
```

This is a serious problem. We highly recommend implementing the corresponding logics in the code or fixing the whitepaper and informing the investors about it.
Besides, the whitepaper states (p.14):
"All purchased Debitum tokens will be distributed within 15 calendar days after the closing of the Crowdsale."
In the code the tokens are sent immediately, without any 15 days period. We recommend fixing this issue, too.

Reentrancy

There is a reentrancy in TokenVesting.sol, line 96, the `release` method. The `safeTransfer` method is called, which calls the `transfer` of ERC223 token, which in his turn calls `tokenFallback` of the initial caller. If the `tokenFallback` function of the initial caller calls `release`, the attacker will be able to get the same tokens again.
We highly recommend following the withdrawal pattern (e.g., every investor withdraws his/her funds) here as well as following the Checks-Effects-Interactions pattern.
The code does not contain severe reentrancy on commit 6f701a0; however, the CEI violation persists (line 123), we highly recommend fixing it.

Reentrancy

There is a reentrancy in TokenVesting.sol, line 105, the `removeBeneficiary` method. The `safeTransfer` method is called, which calls the `transfer` of ERC223 token, which in his turn calls `tokenFallback` of the initial caller. We highly recommend following the Checks-Effects-Interactions pattern.
The code does not contain severe reentrancy on commit 6f701a0; however, the CEI violation persists, we highly recommend fixing it (by moving `transfer` to the end of the function).

Vesting

It is unclear how vesting contract is connected to the crowdsale contract. The shares of participants that are declared in the whitepaper are not defined in contracts. Thus, investors are forced to trust the deploy script that it uses the vesting contract with correct shares and terms. We highly recommend to guarantee to users that the vesting will run as declared inside of the contracts and not only in deploy script.

Vesting logics violated

In the `removeBeneficiary` function (TokenVesting, line 105) the vesting logics is violated: `totalVestingTokenAmount` is not reduced when the beneficiary is removed. Tokens are sent to the owner, although he/she did not send them to the beneficiary in `addBeneficiary` (TokenVesting, line 74). This means that tokens can be transferred from the contract to the owner by calling `addBeneficiary` and `removeBeneficiary`.
We highly recommend fixing this vulnerability.
The `totalVestingTokenAmount` issue has been fixed by the developer and is not present on commit 6f701a0, but the tokens are still being sent to the owner, thus, the problem persists.

Multisignature only in deploy

The multisignature wallet is not used in the crowdsale contract. Thus, investors are forced to trust the deploy script that it uses the multisignature wallet and not something else. We highly recommend to guarantee that the multisignature wallet is used inside of the contracts (not in deploy script).
The issue has been fixed by the developer and is not present on commit 6f701a0. Still, we recommend checking that the specified address contain the correct contract.

Capitalization can be changed

The `changeHardCap` (Crowdsale.sol, line 251) method allows changing `CROWDFUND_HARD_CAP` during the crowdsale, which would change the token distribution. We highly recommend prohibiting this.

Wrong variable

In Crowdsale.sol, line 245 the crowdsale stores information about invested ethers of each contributor. But it saves `weiRaised` (all raised ethers to this moment) instead of `weiAmount`. We highly recommend fixing this bug.
The issue has been fixed by the developer and is not present on commit 6f701a0.

Last contributor

The crowdsale will not return unspent ethers to the last contributor when cap is reached (but this contributor will get tokens according to `CROWDFUND_HARD_CAP.safeSub(weiRaised)`, not all sent ethers). We highly recommend fixing this bug.
The issue has been fixed by the developer and is not present on commit 6f701a0.

Out of gas

The `finalizeCrowdsale` function (Crowdsale.sol, line 332) has two loops with unknown number of iterations. If there are a lot of investors the function will run out of gas and fail. Thus, there will be no opportunity to finalize crowdsale. We highly recommend following the withdrawal pattern (e.g., every investor withdraws his/her funds) here.
The issue has been fixed by the developer and is not present on commit 6f701a0.

Checks-Effects-Interactions

The `finalizeCrowdsale` method (Crowdsale.sol, line 332) violates the Checks-Effects-Interactions pattern at lines 345 and 358. It calls `transfer` inside the loop, so finalization process can get stuck (if contributor has expensive fallback function). We highly recommend following the withdrawal pattern (e.g., every investor withdraws his/her funds) here as well as following the Checks-Effects-Interactions pattern.
The issue has been fixed by the developer and is not present on commit 6f701a0.

Medium severity issues

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

Deploy

The provided code contains dummy deploy scripts. Bugs and vulnerabilities in deploy scripts often appear and severely endanger contracts' security. We highly recommend not only developing deploy scripts very carefully but also performing audit of them.
The issue has been fixed by the developer and is not present on commit 6f701a0.

Out of gas

The `release` method (TokenVesting.sol, line 91) and the `removeBeneficiary` method (TokenVesting.sol, line 105) loop over beneficiaries, which may make contract run out of gas in case of big number of beneficiaries. We highly recommend following the withdrawal pattern (e.g., every investor withdraws his/her funds) here.
Though there are `withdraw`'s for releasing, still `released = true` is set only in `release` function (and `release` function is not deleted) on commit 6f701a0.

ERC223 violation

The `tokenFallback` function (iEC23Receiver.sol, line 20) violates ERC223 standard since it contains an extra parameter `_sender`. Besides, the `ContractTransfer` event violates the standard: the event name should be `Transfer` instead of `ContractTransfer`, the parameters should be `indexed`. Also, there is violation of the following requirement: "If the receiver does not implement the `tokenFallback` function, consider the contract is not designed to work with tokens, then the transaction must fail and no tokens will be transferred." There may be circumstances in which it is not satisfied - see https://github.com/ethereum/EIPs/issues/223#issuecomment-326758995 for details. We highly recommend changing the contracts so that they would satisfy the declared ERC223 standard.
The `tokenFallback` has been fixed by the developer and is not present on commit 6f701a0; `Transfer` and `indexed` on `_value` parameter have not been fixed.

Maximum owner number exceeded

The constructor in MultisigWallet.sol, line 201 uses the following check:

```
        _owners.length <= MAX_OWNER_COUNT
```

The `MAX_OWNER_COUNT` might be exceeded since the nonstrict inequality is used and `_owners` and `msg.sender` become the owners of the wallet. We recommend using strict inequality here.
The issue has been fixed by the developer and is not present on commit 6f701a0.

Unchecked address

The `StandardToken(transactions[transactionId].token)` call (MultisigWallet.sol, lines 374 and 376) implies that the `transactions[transactionId].token` address is an address of token contract. However, this is not checked when the transaction is created. We recommend adding the corresponding check.
The issue has been fixed by the developer and is not present on commit 6f701a0.

removeOwner bug

If the following conditions in `removeOwner` function are met:

- `owners.length == required`
- one of the owners who confirm the removing is the owner that is being removed
- this owner is the last one to confirm the removing

then the removing will fail, because `isOwner[owner] = false;` is executed on line 253 and thus `changeRequirement` will throw on `ownerExists(msg.sender)` modifier. We recommend fixing this bug.
The issue has been fixed by the developer and is not present on commit 6f701a0.

Owner removing logics

When the owner is removed, his confirmations are not removed from the corresponding mappings. This means that the confirmations of the removed user that had been made before he/she was removed will still be valid after the user is removed. We recommend fixing this issue.
The issue has been fixed by the developer and is not present on commit 6f701a0.

Bad code

The `getTransactionIds` function (MultisigWallet.sol, line 478) iterates first over all transactions and then over transactions between `from` and `to`. It would be better to execute only one loop over transactions between `from` and `to`. Besides, the last two arguments of the function `bool pending` and `bool excuted` are supposed to be opposite and can be replaced with any one of them. We highly recommend rewriting the `getTransactionIds` function.

ERC20 increase/decrease

No `increaseApproval`/`decreaseApproval` are implemented in DebitumToken.sol. We recommend fixing this issue because there is ERC20 approve issue and `increaseApproval`/`decreaseApproval` might be used to avoid it (link).
The issue has been fixed by the developer and is not present on commit 6f701a0. However, we recommend implementing zero checks in `approve` function (either `_value` is `0` or `allowed[msg.sender][_spender]` is `0`). Also, we recommend informing users to use `increaseApproval/decreaseApproval` functions or to use `approve` with caution (wait for the first transaction to be mined).

Test coverage

The test coverage of the contracts is not full. The following crucial functions are not tested:

- TokenVesting.sol:
  - removeBeneficiary
- MultisigWallet.sol:
  - addOwner
  - changeRequirement
  - revokeConfirmation
  - getConfirmationCount
  - tokenBalance
    The issue has been fixed by the developer and is not present on commit 6f701a0.
  - getTransactionIds
  - deleteOwnersAddApproval
  - deleteOwnersRemoveApproval

We highly recommend creating test for them.

assert instead of require

There are many places in the code where `assert` is used instead of `require`. `assert` spends all the gas; `require` does not spend remaining gas (after Metropolis release). We highly recommend using `require`.
The issue has been fixed by the developer and is not present on commit 6f701a0.

No information on minimum contribution

The whitepaper does not give any information on minimal contribution of 0.1 ether (Crowdsale.sol, line 118). We highly recommend adding this information to the whitepaper.

Vesting VS frozen

In several places in the whitepaper terms ***vesting*** and ***frozen*** are used interchangeably. We highly recommend using only one of them to make the text more clear; we recommend using ***frozen*** since it is closer to what is implied.
The issue has been fixed by the developer and is not present on commit 6f701a0.

Low severity issues

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

No wallet check

The `Crowdsale` constructor (Crowdasle.sol, line 167) performs many checks but does not check that the given wallet address is non-zero. We recommend adding this check to avoid sending funds to an unexisting address.
The issue has been fixed by the developer and is not present on commit 6f701a0.

Unnecessary fallback

The fallback function in DebitumToken.sol, line 23 is not needed. We recommend removing it since it will make the deploy cheaper.
The issue has been fixed by the developer and is not present on commit 6f701a0.

Outdated safeMath

The version of `safeMath` that is used in the contracts is outdated. We recommend using the latest version.
The issue has been fixed by the developer and is not present on commit 6f701a0.

Codestyle issues

Codestyle issues influence code conciseness and readability and in some cases may lead to bugs in future. We recommend to take them into account.

Pragmas version

Solidity source files indicate the versions of the compiler they can be compiled with.
Example:
```
      pragma solidity ^0.4.18; // bad: compiles w 0.4.18 and above
      pragma solidity 0.4.18; // good : compiles w 0.4.18 only
```
We recommend following the latter example, as future compiler versions may handle certain language constructions in a way the developer did not foresee. Besides, we recommend using the latest compiler version (0.4.18 at the moment).
The issue has been fixed by the developer and is not present on commit 6f701a0 in the main contracts; in the interfaces and OpenZeppelin contracts the problem persists.

Implicit visibility level

In many places in the code there are functions and variables with implicit visibility level. We recommend specifying visibility level explicitly and correctly.
The issue has been fixed by the developer and is not present on commit 6f701a0.

Function can be declared as view

In many places in the code there are functions that can be declared as `view`. If a function is not supposed to modify the state, consider declaring it as `view` (alias for `constant`, which will be deprecated for functions). This provides additional checks in case these assumptions are violated. If a function is not supposed to modify the state or read from state, consider declaring it as `pure`.
The issue has been fixed by the developer and is not present on commit 6f701a0.

Redundant code

The following entities in Crowdsale.sol are not used in the code:

- `MAX_OWNER_COUNT`
- `signerAddress`
- `State`
- `forwardFunds`
- `minimalCapReached`

We recommend removing them from the code.
The issue has been fixed by the developer and is not present on commit 6f701a0.

Redundant code

In Crowdsale.sol, line 140 the `verifiedForCrowdsale` modifier takes arguments `participant` and `weiAmount`. However, the modifier is used only once, with the pair of values `(msg.sender, msg.value)`. Thus we recommend hardcoding these values into the modifier and removing its arguments.
The issue has been fixed by the developer and is not present on commit 6f701a0.

Redundant code

The loop in MultiSigWallet.sol, line 499 is redundant. We recommend using `delete ownersConfirmedOwnerAdd[_confirmedOwner];` to delete the array.
The issue has been fixed by the developer and is not present on commit 6f701a0.

Redundant code

The loop in MultiSigWallet.sol, line 507 is redundant. We recommend using `delete ownersConfirmedOwnerRemove[_confirmedOwner];` to delete the array.
The issue has been fixed by the developer and is not present on commit 6f701a0.

Conclusion

In this report we have considered the security of Debitum smart contracts. We performed our audit according to the procedure described above.
The audit showed many critical issues, as well as many lower issues. They were reported to the developer. Most of the issues have been fixed by the developer and are not present on commit 6f701a0. However, we highly recommend fixing the remaining issues.
The token contract does not contain any critical issues.

This analysis was performed by SmartDec

COO Sergey Pavlin [illegible]

December 1, 2017

Appendix

Code coverage

Here is the output of the `solidity-coverage` test coverage calculation tool:

```
------------------------------------------------------------------------------|----------|----
------|----------|----------|----------------|
File                                                                           |  % Stmts | %
Branch |  % Funcs |  % Lines |Uncovered Lines |
------------------------------------------------------------------------------|----------|-----
-----|----------|----------|----------------|
C:/Users/alexandrov/work/sc/audit/debitum/ico-contracts/contracts/helpers\   |      100
|      100 |      100 |      100 |                |
 ERC23ReceiverMock.sol                                                        |      100
|      100 |      100 |      100 |                |
C:/Users/alexandrov/work/sc/audit/debitum/ico-contracts/contracts/interface\ |      100
|      100 |      100 |      100 |                |
 iEC23Receiver.sol                                                            |      100
|      100 |      100 |      100 |                |
 iERC20Token.sol                                                              |      100
|      100 |      100 |      100 |                |
 iERC23Token.sol                                                              |      100
|      100 |      100 |      100 |                |
C:/Users/alexandrov/work/sc/audit/debitum/ico-contracts/contracts/zeppelin\  |    76.92
|       50 |    73.33 |     77.5 |                |
 Ownable.sol                                                                  |       40
|       50 |    66.67 |       50 |       39,40,41 |
 SafeERC20.sol                                                                |    33.33
|    16.67 |    33.33 |    33.33 |          15,19 |
 SafeMath.sol                                                                 |    66.67
|       30 |       75 |    66.67 |    16,17,18,19 |
 StandardToken.sol                                                            |      100
|       90 |      100 |      100 |                |
C:/Users/alexandrov/work/sc/audit/debitum/ico-contracts/contracts\           |    78.02
|    54.67 |    76.47 |    74.63 |                |
 Crowdsale.sol                                                                |     96.3
|    67.86 |    88.89 |    94.57 |... 293,322,374 |
 DebitumToken.sol                                                             |    93.75
|     87.5 |      100 |    93.33 |             38 |
 MultisigWallet.sol                                                           |    63.96
|    44.12 |    63.64 |    61.36 |... 500,501,504 |
 TokenVesting.sol                                                             |    70.83
|    38.89 |    77.78 |    63.64 |... 120,121,123 |
------------------------------------------------------------------------------|----------|-----
-----|----------|----------|----------------|
All files                                                                      |     78.1
|    53.89 |    76.47 |    75.24 |                |
------------------------------------------------------------------------------|----------|-----
-----|----------|----------|----------------|
```

Compilation output

```
Compiling .\contracts\Crowdsale.sol...
Compiling .\contracts\DebitumToken.sol...
Compiling .\contracts\Migrations.sol...
Compiling .\contracts\MultisigWallet.sol...
Compiling .\contracts\TokenVesting.sol...
Compiling .\contracts\helpers\ERC23ReceiverMock.sol...
Compiling .\contracts\interface\iEC23Receiver.sol...
Compiling .\contracts\interface\iERC20Token.sol...
Compiling .\contracts\interface\iERC23Token.sol...
Compiling .\contracts\zeppelin\Ownable.sol...
Compiling .\contracts\zeppelin\SafeERC20.sol...
Compiling .\contracts\zeppelin\SafeMath.sol...
Compiling .\contracts\zeppelin\StandardToken.sol...


Compilation warnings encountered:
```

```
/C/Users/user/debitum/ico-contracts/contracts/zeppelin/Ownable.sol:20:3: Warning: No
visibility specified. Defaulting to "public".
 function Ownable() {
 ^
Spanning multiple lines.
,/C/Users/user/debitum/ico-contracts/contracts/interface/iERC20Token.sol:10:5: Warning: No
visibility specified. Defaulting to "public".
   function balanceOf(address _owner) constant returns (uint256 balance);
   ^--------------------------------------------------------------------^
,/C/Users/user/debitum/ico-contracts/contracts/interface/iERC20Token.sol:16:5: Warning: No
visibility specified. Defaulting to "public".
   function transfer(address _to, uint256 _value) returns (bool success);
   ^--------------------------------------------------------------------^
,/C/Users/user/debitum/ico-contracts/contracts/interface/iERC20Token.sol:23:5: Warning: No
visibility specified. Defaulting to "public".
   function transferFrom(address _from, address _to, uint256 _value) returns (bool success);
   ^--------------------------------------------------------------------------------------^
,/C/Users/user/debitum/ico-contracts/contracts/interface/iERC20Token.sol:29:5: Warning: No
visibility specified. Defaulting to "public".
   function approve(address _spender, uint256 _value) returns (bool success);
   ^------------------------------------------------------------------------^
,/C/Users/user/debitum/ico-contracts/contracts/interface/iERC20Token.sol:34:5: Warning: No
visibility specified. Defaulting to "public".
   function allowance(address _owner, address _spender) constant returns (uint256 remaining);
   ^---------------------------------------------------------------------------------------^
,/C/Users/user/debitum/ico-contracts/contracts/zeppelin/StandardToken.sol:21:5: Warning: No
visibility specified. Defaulting to "public".
   function transfer(address _to, uint _value) returns (bool success) {
   ^
Spanning multiple lines.
,/C/Users/user/debitum/ico-contracts/contracts/zeppelin/StandardToken.sol:32:5: Warning: No
visibility specified. Defaulting to "public".
   function transferFrom(address _from, address _to, uint _value) returns (bool success) {
   ^
Spanning multiple lines.
,/C/Users/user/debitum/ico-contracts/contracts/zeppelin/StandardToken.sol:44:5: Warning: No
visibility specified. Defaulting to "public".
   function balanceOf(address _owner) constant returns (uint balance) {
   ^
Spanning multiple lines.
,/C/Users/user/debitum/ico-contracts/contracts/zeppelin/StandardToken.sol:48:5: Warning: No
visibility specified. Defaulting to "public".
   function approve(address _spender, uint _value) returns (bool success) {
   ^
Spanning multiple lines.
,/C/Users/user/debitum/ico-contracts/contracts/zeppelin/StandardToken.sol:54:5: Warning: No
visibility specified. Defaulting to "public".
   function allowance(address _owner, address _spender) constant returns (uint remaining) {
   ^
Spanning multiple lines.
,/C/Users/user/debitum/ico-contracts/contracts/interface/iEC23Receiver.sol:20:5: Warning: No
visibility specified. Defaulting to "public".
   function tokenFallback(address _sender, address _origin, uint _value, bytes _data) returns
(bool ok);


^-----------------------------------------------------------------------------------------------
------^
,/C/Users/user/debitum/ico-contracts/contracts/interface/iERC23Token.sol:5:5: Warning: No
visibility specified. Defaulting to "public".
   function transfer(address to, uint value, bytes data) returns (bool ok);
   ^----------------------------------------------------------------------^
,/C/Users/user/debitum/ico-contracts/contracts/interface/iERC23Token.sol:6:5: Warning: No
visibility specified. Defaulting to "public".
   function transferFrom(address from, address to, uint value, bytes data) returns (bool ok);
   ^----------------------------------------------------------------------------------------^
,/C/Users/user/debitum/ico-contracts/contracts/DebitumToken.sol:19:5: Warning: No visibility
specified. Defaulting to "public".
   function DebitumToken() {
   ^
Spanning multiple lines.
,/C/Users/user/debitum/ico-contracts/contracts/DebitumToken.sol:23:5: Warning: No visibility
specified. Defaulting to "public".
   function () payable {
   ^
```

```
Spanning multiple lines.
,/C/Users/user/debitum/ico-contracts/contracts/DebitumToken.sol:28:5: Warning: No visibility
specified. Defaulting to "public".
   function transfer(address _to, uint _value, bytes _data) returns (bool success) {
   ^
Spanning multiple lines.
,/C/Users/user/debitum/ico-contracts/contracts/DebitumToken.sol:35:5: Warning: No visibility
specified. Defaulting to "public".
   function transferFrom(address _from, address _to, uint _value, bytes _data) returns (bool
success) {
   ^
Spanning multiple lines.
,/C/Users/user/debitum/ico-contracts/contracts/DebitumToken.sol:43:5: Warning: No visibility
specified. Defaulting to "public".
   function transfer(address _to, uint _value) returns (bool success) {
   ^
Spanning multiple lines.
,/C/Users/user/debitum/ico-contracts/contracts/DebitumToken.sol:47:5: Warning: No visibility
specified. Defaulting to "public".
   function transferFrom(address _from, address _to, uint _value) returns (bool success) {
   ^
Spanning multiple lines.
,/C/Users/user/debitum/ico-contracts/contracts/Crowdsale.sol:167:5: Warning: No visibility
specified. Defaulting to "public".
   function Crowdsale(
   ^
Spanning multiple lines.
,/C/Users/user/debitum/ico-contracts/contracts/Crowdsale.sol:211:5: Warning: No visibility
specified. Defaulting to "public".
   function () payable {
   ^
Spanning multiple lines.
,/C/Users/user/debitum/ico-contracts/contracts/Migrations.sol:11:3: Warning: No visibility
specified. Defaulting to "public".
 function Migrations() {
 ^
Spanning multiple lines.
,/C/Users/user/debitum/ico-contracts/contracts/Migrations.sol:15:3: Warning: No visibility
specified. Defaulting to "public".
 function setCompleted(uint completed) restricted {
 ^
Spanning multiple lines.
,/C/Users/user/debitum/ico-contracts/contracts/Migrations.sol:19:3: Warning: No visibility
specified. Defaulting to "public".
 function upgrade(address new_address) restricted {
 ^
Spanning multiple lines.
,/C/Users/user/debitum/ico-contracts/contracts/MultisigWallet.sol:188:5: Warning: No
visibility specified. Defaulting to "public".
   function() payable {
   ^
Spanning multiple lines.
,/C/Users/user/debitum/ico-contracts/contracts/TokenVesting.sol:61:3: Warning: No visibility
specified. Defaulting to "public".
 function TokenVesting(StandardToken _token, uint _start, uint _duration) {
 ^
Spanning multiple lines.
,/C/Users/user/debitum/ico-contracts/contracts/helpers/ERC23ReceiverMock.sol:9:5: Warning: No
visibility specified. Defaulting to "public".
   function ERC23ReceiverMock(bool _isReceiver) {
   ^
Spanning multiple lines.


,/C/Users/user/debitum/ico-contracts/contracts/helpers/ERC23ReceiverMock.sol:13:5: Warning: No
visibility specified. Defaulting to "public".
   function tokenFallback(address _sender, address _origin, uint _value, bytes _data) returns
(bool ok){
   ^
Spanning multiple lines.
,/C/Users/user/debitum/ico-contracts/contracts/MultisigWallet.sol:517:28: Warning: Unused
function parameter. Remove or comment out the variable name to silence this warning.
   function tokenFallback(address _sender, address _origin, uint _value, bytes _data) public
returns (bool ok) {
                          ^-------------^
```

```
,/C/Users/user/debitum/ico-contracts/contracts/MultisigWallet.sol:517:45: Warning: Unused
function parameter. Remove or comment out the variable name to silence this warning.
   function tokenFallback(address _sender, address _origin, uint _value, bytes _data) public
returns (bool ok) {
                                           ^-------------^
,/C/Users/user/debitum/ico-contracts/contracts/MultisigWallet.sol:517:62: Warning: Unused
function parameter. Remove or comment out the variable name to silence this warning.
   function tokenFallback(address _sender, address _origin, uint _value, bytes _data) public
returns (bool ok) {
                                                            ^---------^
,/C/Users/user/debitum/ico-contracts/contracts/MultisigWallet.sol:517:75: Warning: Unused
function parameter. Remove or comment out the variable name to silence this warning.
   function tokenFallback(address _sender, address _origin, uint _value, bytes _data) public
returns (bool ok) {
                                                                         ^---------^
,/C/Users/user/debitum/ico-contracts/contracts/TokenVesting.sol:126:26: Warning: Unused
function parameter. Remove or comment out the variable name to silence this warning.
 function tokenFallback(address _sender, address _origin, uint _value, bytes _data) public
returns (bool ok) {
                        ^-------------^
,/C/Users/user/debitum/ico-contracts/contracts/TokenVesting.sol:126:43: Warning: Unused
function parameter. Remove or comment out the variable name to silence this warning.
 function tokenFallback(address _sender, address _origin, uint _value, bytes _data) public
returns (bool ok) {
                                         ^-------------^
,/C/Users/user/debitum/ico-contracts/contracts/TokenVesting.sol:126:60: Warning: Unused
function parameter. Remove or comment out the variable name to silence this warning.
 function tokenFallback(address _sender, address _origin, uint _value, bytes _data) public
returns (bool ok) {
                                                          ^---------^
,/C/Users/user/debitum/ico-contracts/contracts/TokenVesting.sol:126:73: Warning: Unused
function parameter. Remove or comment out the variable name to silence this warning.
 function tokenFallback(address _sender, address _origin, uint _value, bytes _data) public
returns (bool ok) {
                                                                       ^---------^
,/C/Users/user/debitum/ico-contracts/contracts/Crowdsale.sol:270:31: Warning: Function
declared as view, but this expression (potentially) modifies the state and thus requires non-
payable (the default) or payable.
               tokenAmount =
weiLimitOfCurrentStep(_weiRaised).safeMul(currentRate(_weiRaised));
                             ^----------------------------------------------------------------
^
,/C/Users/user/debitum/ico-contracts/contracts/Crowdsale.sol:271:71: Warning: Function
declared as view, but this expression (potentially) modifies the state and thus requires non-
payable (the default) or payable.


uint256 tokenAmountNextStep =
calculateTokenAmountFor(_weiRaised.safeAdd(weiLimitOfCurrentStep(_weiRaised)),
_weiAmount.safeSub(weiLimitOfCurrentStep(_weiRaised)));
                                                                     ^------------------------
---------------------------^
,/C/Users/user/debitum/ico-contracts/contracts/Crowdsale.sol:271:126: Warning: Function
declared as view, but this expression (potentially) modifies the state and thus requires non-
payable (the default) or payable.
               uint256 tokenAmountNextStep =
calculateTokenAmountFor(_weiRaised.safeAdd(weiLimitOfCurrentStep(_weiRaised)),
_weiAmount.safeSub(weiLimitOfCurrentStep(_weiRaised)));

                              ^---------------------------------------------------^
,/C/Users/user/debitum/ico-contracts/contracts/Crowdsale.sol:272:31: Warning: Function
declared as view, but this expression (potentially) modifies the state and thus requires non-
payable (the default) or payable.
               tokenAmount = tokenAmount.safeAdd(tokenAmountNextStep);
                             ^--------------------------------------^
,/C/Users/user/debitum/ico-contracts/contracts/Crowdsale.sol:274:31: Warning: Function
declared as view, but this expression (potentially) modifies the state and thus requires non-
payable (the default) or payable.
               tokenAmount = _weiAmount.safeMul(currentRate(_weiRaised));
                             ^-----------------------------------------^
,/C/Users/user/debitum/ico-contracts/contracts/Crowdsale.sol:287:20: Warning: Function
declared as view, but this expression (potentially) modifies the state and thus requires non-
payable (the default) or payable.
           return FIRST_STEP_UPPER_LIMIT.safeSub(_weiRaised);
                  ^----------------------------------------^
```

```
,/C/Users/user/debitum/ico-contracts/contracts/Crowdsale.sol:289:20: Warning: Function
declared as view, but this expression (potentially) modifies the state and thus requires non-
payable (the default) or payable.
           return SECOND_STEP_UPPER_LIMIT.safeSub(_weiRaised);
                  ^-----------------------------------------^
,/C/Users/user/debitum/ico-contracts/contracts/Crowdsale.sol:291:20: Warning: Function
declared as view, but this expression (potentially) modifies the state and thus requires non-
payable (the default) or payable.
           return CROWDFUND_HARD_CAP.safeSub(_weiRaised);
                  ^------------------------------------^
,/C/Users/user/debitum/ico-contracts/contracts/Crowdsale.sol:303:35: Warning: Function
declared as view, but this expression (potentially) modifies the state and thus requires non-
payable (the default) or payable.
       if (CROWDFUND_HARD_CAP <= weiRaised.safeAdd(_weiAmount)) {
                                 ^----------------------------^
,/C/Users/user/debitum/ico-contracts/contracts/Crowdsale.sol:304:20: Warning: Function
declared as view, but this expression (potentially) modifies the state and thus requires non-
payable (the default) or payable.
           return CROWDFUND_HARD_CAP.safeSub(weiRaised);
                  ^-----------------------------------^
,/C/Users/user/debitum/ico-contracts/contracts/DebitumToken.sol:59:5: Warning: Function state
mutability can be restricted to view
   function isContract(address _addr) private returns (bool is_contract) {
   ^
Spanning multiple lines.
,/C/Users/user/debitum/ico-contracts/contracts/TokenVesting.sol:130:3: Warning: Function state
mutability can be restricted to view
 function supportsToken(address _token) public returns (bool){
 ^
Spanning multiple lines.


,/C/Users/user/debitum/ico-contracts/contracts/zeppelin/SafeMath.sol:9:3: Warning: Function
state mutability can be restricted to pure
 function safeMul(uint a, uint b) internal returns (uint) {
 ^
Spanning multiple lines.
,/C/Users/user/debitum/ico-contracts/contracts/zeppelin/SafeMath.sol:15:3: Warning: Function
state mutability can be restricted to pure
 function safeDiv(uint a, uint b) internal returns (uint) {
 ^
Spanning multiple lines.
,/C/Users/user/debitum/ico-contracts/contracts/zeppelin/SafeMath.sol:22:3: Warning: Function
state mutability can be restricted to pure
 function safeSub(uint a, uint b) internal returns (uint) {
 ^
Spanning multiple lines.
,/C/Users/user/debitum/ico-contracts/contracts/zeppelin/SafeMath.sol:27:3: Warning: Function
state mutability can be restricted to pure
 function safeAdd(uint a, uint b) internal returns (uint) {
 ^
Spanning multiple lines.
```

Tests output

```
Contract: Crowdsale.sol
   √ When hard cap is reached then 60% of token supply is sent (102ms)
   √ Should return token rate by raised wei (79ms)
   √ Should count left wei till step limit (59ms)
   √ When hard cap is reached then crowdsale is finished (2150ms)
   √ If minimal cap is not reached till crowdsale end then all investments returned to
investors (4859ms)
   √ Must not let invest more then 30eth for not verified investors (528ms)
   √ Only signer can register crowdsale participant (387ms)
   √ Let decrease initial hard cap if wei is not raised till new hard cap (889ms)
   √ Hard cap can not be increased above of initial hard cap (290ms)
   √ Wont accept investments bellow 0.1 ether (830ms)
   √ After crowdsale all not sold tokens are transfered to wallet (4622ms)


 Contract: DebitumToken.sol
   √ Token contract should return the correct total supply after construction (214ms)
   √ Token should throw an error when trying to transfer to 0x0 (248ms)
   √ Token should throw an error when trying to transferFrom to 0x0 (253ms)
```

```
    √ Token creator at the start has all tokens (396ms)
    √ Should return the correct allowance amount after approval (723ms)
    √ Should throw an error when trying to transfer more than allowed (779ms)
    √ Should return correct balances after transfering from another account (1323ms)
    √ Token should return correct balances after transfer (295ms)
    √ Should throw an error when trying to transfer more than balance (652ms)
    √ Ether cannot be sent to token contract (738ms)
    √ Tokens may not by sent to another contract if it does not implement ERC23Receiver
standard (752ms)
    √ Tokens may by sent to another contract if it implement ERC23Receiver standard (695ms)


 Contract: MultiSigWallet
    √ Requires multiple confirmations (952ms)
    √ After getting required confirmations ether transferred to destination (1207ms)
    √ After getting required confirmations tokens transferred to destination (862ms)
    √ Eth transaction confirmation cannot be revoked by owner who not confirmed transaction
earlier (676ms)
    √ Owner removal needs confirmations of required owners (947ms)


 Contract: TokenVesting.sol
    √ Only owner can add beneficiary (945ms)
    √ Tokens cannot be transferred more, then added to contract (733ms)
    √ Should have released all after end (576ms)



 31 passing (33s)
```