SmartDec

Tolar Smart Contracts Security Analysis

This report is public.

Published: September 12, 2018

Abstract

In this report, we consider the security of the Tolar 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 Tolar smart contracts. We performed our audit according to the procedure described below.
The audit showed no critical issues. However, two medium severity and two low severity issues were found. They do not endanger project security. Nevertheless, we highly recommend addressing them.

All these issues were fixed by the developers and are not present in The latest version of the code.

General recommendations

The contracts' code is of good code quality. Thus, we do not have any additional recommendations.

Checklist

Security

The audit has shown no 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 no discrepancies between the code and the whitepaper except the minor ones (see Whitepaper mismatches).

ERC20 compliance

We have checked ERC20 compliance during the audit. The audit has shown that **TolarToken.sol** 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 covered with tests sufficiently.

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
    - we run tests and check code coverage
- report
    - we check the issues fixed by the developer
    - we reflect all the gathered information in the report

Checked vulnerabilities

We have scanned Tolar 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
- Front Running
- DoS with (Unexpected) revert
- DoS with Block Gas Limit
- Gas Limit and Loops
- Locked money
- Integer overflow/underflow
- Unchecked external call
- ERC20 Standard violation
- Authentication with tx.origin
- Unsafe use of timestamp
- Using blockhash for randomness
- Balance equality
- Unsafe transfer of ether
- Fallback abuse
- Using inline assembly
- Short Address Attack
- Private modifier
- Compiler version not fixed
- Style guide violation
- Unsafe type deduction
- Implicit visibility level
- Use delete for arrays
- Byte array
- Incorrect use of assert/require
- Using deprecated constructions

Project overview

Project description

In our analysis we consider Tolar whitepaper ("Whitepaper-Tolar.pdf", sha1sum: a0ad0e2af78ee785fa7b533b46fa4b7b3d835d3c) and smart contracts code (version on commit fcc46608b4ffd2cf1980fd8f9c6794afa8f7cb0e).

The latest version of the code

We have performed the check of the fixed vulnerabilities in the latest version of the code (version on commit b247bb3f8151383727e849c0b770071d726b03fc).

All the outputs in Appendix are performed for the latest version of the code.

Project architecture

For the audit, we have been provided with the truffle project. The project is an npm package and includes tests.

- The project successfully compiles with `truffle compile` command (with some warnings, see Compilation output in Appendix)
- The project successfully passes all the tests, however, code coverage is not sufficient (`truffle test` command, see Tests output and Code coverage in Appendix)

The total LOC of audited Solidity code is 530.

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 the 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 where 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	1	
	Potential Violation of Checks-Effects-Interaction pattern	2	
	Should be constant but is not	8	
	Use of "block.timestamp"	14	
Total Remix		25	
SmartCheck	Address Hardcoded	2	
	Gas Limit In Loops	2	1
	Private Modifier Do not Hide Data	1	
	Safemath	4	
Total SmartCheck		9	1

Solhint	Avoid to make time-based decisions in your business logic	2	
	Code contains empty block	4	
	Variable "_payee" is unused		1
Total Solhint		6	1
Overall total		40	2

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.

The audit has shown no critical issues.

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. The token contract creator initially owns all Tolar tokens. This means that the distribution of the tokens will be performed manually. Hence, there is no way to check, whether it will be done according to the whitepaper.
2. Bonus tokens will be deposited manually by the owner of **TokenDistributor** contract.
3. Token lock and vesting are done manually by calling `depositAndLock()` and `depositAndVest()` functions. These functions accept number of tokens and locking/vesting period as arguments. Thus, there is no guarantee that token lock and vesting will be done according to the whitepaper.

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.
The issues have been fixed by the developers and are not present in the latest version of the code.

No deployment script

The provided code does not contain deployment script. Thus, there is no possibility to check constructor parameters, mentioned in the whitepaper. Bugs and vulnerabilities often appear in deployment scripts and severely endanger system's security.

We highly recommend developing deployment scripts.
The issue has been fixed by the developers and is not present in the latest version of the code.

Low severity issues

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

Redundant code

The following lines are redundant:

1. **TokenTimelockEscrow.sol**, line 28

```
function withdrawalAllowed(address _payee) public view returns
(bool)
```

`_payee` variable is defined, but never used.

2. **TokenDistributor.sol**, line 72:

```
require(isFinalized, "Contract not finalized.");
```

This check is redundant, since it is true only if the next check is also true.
TokenDistributor.sol, line 73:

```
require(crowdsale != address(0), "Crowdsale not started.");
```

`crowdsale` address is set in the internal function `finalization()`, which is called in the `finalize()` function of the **Finalizable** contract. In the same function, `isFinalized` variable value is set to `true` and is never changed after. Thus, if `crowdsale` address is not null, then `isFinalized` variable is set to `true`.
The issue has been fixed by the developers and is not present in the latest version of the code.

3. **TokenDistributor.sol**, line 336:

```
super.finalization();
```

This function is empty in super class. Thus, there is no need in calling it.
The issue has been fixed by the developers and is not present in the latest version of the code.

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

Whitepaper mismatches

There are several discrepancies between the smart contracts code and the whitepaper.

1. According to the whitepaper, Tolar token should have name “Tolar”. Nevertheless, in the current implementation, Tolar token’s name is "Example Token".
   TolarToken.sol, line 31:

```
string public constant NAME = "Example Token";
```

The issue has been fixed by the developers and is not present in the latest version of the code.

2. Some token lock periods differ in the deploy script from the ones, stated in the whitepaper.
   According to the whitepaper:

```
 20% Founders: 36 months after ICO with 5% vesting possible
after 24 months
```

However, in the configuration scripts, 20% of the tokens will be allocated to founders after two years:
scripts/config.js, lines 36–39, 42–45:

```
id: 'founder_1',
address: '0x03C3A6aa06d2130d67Ba62AcFdf6B0a311fee659',
percentage: 0.1,
releaseTime: closingTime + duration.years(2),
…
id: 'founder_2',
address: '0x3c9F55570fb4c3bb8e056AF064914f037E2a46D0',
percentage: 0.1,
releaseTime: closingTime + duration.years(2),
```

According to the whitepaper:

```
8% Proof of Stake Network Start Nodes: 36 months after ICO
```

However, in the configuration scripts, lock period equals to:
scripts/config.js, line 29:

```
bonusTime: withdrawTime + duration.days(180),
```

According to the whitepaper:

```
2,5% Advisors: 24 months after ICO
```

However, in the configuration scripts, lock period equals to:

```
releaseTime: closingTime + duration.years(1),
```

Comment from the developers:
"the configuration in the deployment scripts is confirmed and double checked by the Tolar team. They said that the whitepaper will be updated with the new info before the ICO."

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

Notes

Gas limit and loops

The loop at **TokenCrowdsale.sol**, line 81 traverses through an array of variable length:

```
for (uint32 i = 0; i < _beneficiaries.length; i ++)
```

`_beneficiaries` array is passed as `withdrawTokens()` function parameter. Therefore, if there are too many items in `_beneficiaries` array, the execution of `withdrawTokens()` function will fail due to an out-of-gas exception.
In this case, we recommend separating the call into several transactions.

This analysis was performed by SmartDec.

Evgeny Marchenko, Lead Developer
Alexander Seleznev, Chief Business Development Officer
Boris Nikashin, Business Developer
Alexander Drygin, Analyst
Pavel Kondratenkov, Analyst

Sergey Pavlin, Chief Operating Officer 

September 12, 2018

Appendix

Code coverage

```
-----------------------------------|----------|----------|----------|----------|----------------|
File                               |  % Stmts | % Branch |  % Funcs |  % Lines |Uncovered Lines |
-----------------------------------|----------|----------|----------|----------|----------------|
 contracts/                        |      100 |      100 |      100 |      100 |                |
  TokenCrowdsale.sol               |      100 |      100 |      100 |      100 |                |
  TokenDistributor.sol             |      100 |      100 |      100 |      100 |                |
  TokenTimelockEscrowImpl.sol      |      100 |      100 |      100 |      100 |                |
  TolarToken.sol                   |      100 |      100 |      100 |      100 |                |
 contracts/crowdsale/distribution/ |      100 |       75 |      100 |      100 |                |
  PostDeliveryCrowdsale.sol        |      100 |       75 |      100 |      100 |                |
 contracts/lifecycle/              |      100 |      100 |      100 |      100 |                |
  Factory.sol                      |      100 |      100 |      100 |      100 |                |
  Finalizable.sol                  |      100 |      100 |      100 |      100 |                |
 contracts/payment/                |      100 |       95 |      100 |      100 |                |
  TokenConditionalEscrow.sol       |      100 |      100 |      100 |      100 |                |
  TokenEscrow.sol                  |      100 |     87.5 |      100 |      100 |                |
  TokenTimelockEscrow.sol          |      100 |      100 |      100 |      100 |                |
  TokenTimelockFactory.sol         |      100 |      100 |      100 |      100 |                |
  TokenTimelockFactoryImpl.sol     |      100 |      100 |      100 |      100 |                |
  TokenVestingFactory.sol          |      100 |      100 |      100 |      100 |                |
  TokenVestingFactoryImpl.sol      |      100 |      100 |      100 |      100 |                |
 contracts/token/ERC20/            |      100 |      100 |      100 |      100 |                |
  SafeStandardToken.sol            |      100 |      100 |      100 |      100 |                |
-----------------------------------|----------|----------|----------|----------|----------------|
All files                          |      100 |    97.92 |      100 |      100 |                |
-----------------------------------|----------|----------|----------|----------|----------------|
```

Compilation output

```
Compiling ./contracts/Migrations.sol...
Compiling ./contracts/TokenCrowdsale.sol...
Compiling ./contracts/TokenDistributor.sol...
Compiling ./contracts/TokenTimelockEscrowImpl.sol...
Compiling ./contracts/TolarToken.sol...
Compiling
./contracts/crowdsale/distribution/PostDeliveryCrowdsale.sol..
.
Compiling ./contracts/lifecycle/Factory.sol...
Compiling ./contracts/lifecycle/Finalizable.sol...
Compiling ./contracts/mocks/FinalizableMock.sol...
Compiling ./contracts/mocks/SafeStandardTokenMock.sol...
Compiling ./contracts/mocks/TokenTimelockEscrowMock.sol...
Compiling ./contracts/payment/TokenConditionalEscrow.sol...
Compiling ./contracts/payment/TokenEscrow.sol...
Compiling ./contracts/payment/TokenTimelockEscrow.sol...
Compiling ./contracts/payment/TokenTimelockFactory.sol...
Compiling ./contracts/payment/TokenTimelockFactoryImpl.sol...
Compiling ./contracts/payment/TokenVestingFactory.sol...
```

```
Compiling ./contracts/payment/TokenVestingFactoryImpl.sol...
Compiling ./contracts/token/ERC20/SafeStandardToken.sol...
Compiling openzeppelin-
solidity/contracts/crowdsale/Crowdsale.sol...
Compiling openzeppelin-
solidity/contracts/crowdsale/emission/AllowanceCrowdsale.sol..
.
Compiling openzeppelin-
solidity/contracts/crowdsale/validation/CappedCrowdsale.sol...
Compiling openzeppelin-
solidity/contracts/crowdsale/validation/IndividuallyCappedCrow
dsale.sol...
Compiling openzeppelin-
solidity/contracts/crowdsale/validation/TimedCrowdsale.sol...
Compiling openzeppelin-solidity/contracts/math/SafeMath.sol...
Compiling openzeppelin-
solidity/contracts/ownership/CanReclaimToken.sol...
Compiling openzeppelin-
solidity/contracts/ownership/HasNoContracts.sol...
Compiling openzeppelin-
solidity/contracts/ownership/HasNoEther.sol...
Compiling openzeppelin-
solidity/contracts/ownership/HasNoTokens.sol...
Compiling openzeppelin-
solidity/contracts/ownership/NoOwner.sol...
Compiling openzeppelin-
solidity/contracts/ownership/Ownable.sol...
Compiling openzeppelin-
solidity/contracts/token/ERC20/BasicToken.sol...
Compiling openzeppelin-
solidity/contracts/token/ERC20/BurnableToken.sol...
Compiling openzeppelin-
solidity/contracts/token/ERC20/DetailedERC20.sol...
Compiling openzeppelin-
solidity/contracts/token/ERC20/ERC20.sol...
Compiling openzeppelin-
solidity/contracts/token/ERC20/ERC20Basic.sol...
Compiling openzeppelin-
solidity/contracts/token/ERC20/SafeERC20.sol...
Compiling openzeppelin-
solidity/contracts/token/ERC20/StandardBurnableToken.sol...
Compiling openzeppelin-
solidity/contracts/token/ERC20/StandardToken.sol...
Compiling openzeppelin-
solidity/contracts/token/ERC20/TokenTimelock.sol...
Compiling openzeppelin-
solidity/contracts/token/ERC20/TokenVesting.sol...
```

```
Compilation warnings encountered:

openzeppelin-solidity/contracts/crowdsale/Crowdsale.sol:136:5:
Warning: Unused function parameter. Remove or comment out the
variable name to silence this warning.
    address _beneficiary,
    ^------------------^
,openzeppelin-
solidity/contracts/crowdsale/Crowdsale.sol:137:5: Warning:
Unused function parameter. Remove or comment out the variable
name to silence this warning.
    uint256 _weiAmount
    ^----------------^
,openzeppelin-
solidity/contracts/crowdsale/Crowdsale.sol:178:5: Warning:
Unused function parameter. Remove or comment out the variable
name to silence this warning.
    address _beneficiary,
    ^------------------^
,openzeppelin-
solidity/contracts/crowdsale/Crowdsale.sol:179:5: Warning:
Unused function parameter. Remove or comment out the variable
name to silence this warning.
    uint256 _weiAmount
    ^----------------^
,contracts/payment/TokenTimelockEscrow.sol:28:30: Warning:
Unused function parameter. Remove or comment out the variable
name to silence this warning.
  function withdrawalAllowed(address _payee) public view
returns (bool) {
                             ^------------^
,contracts/lifecycle/Finalizable.sol:46:3: Warning: Function
state mutability can be restricted to pure
  function finalization() internal {
  ^ (Relevant source part starts here and spans across
multiple lines).
,contracts/mocks/FinalizableMock.sol:9:3: Warning: Function
state mutability can be restricted to view
  function finalized() public onlyFinalized {
  ^ (Relevant source part starts here and spans across
multiple lines).
,contracts/mocks/FinalizableMock.sol:13:3: Warning: Function
state mutability can be restricted to view
  function notFinalized() public onlyNotFinalized {
  ^ (Relevant source part starts here and spans across
multiple lines).
```

```
,openzeppelin-
solidity/contracts/crowdsale/Crowdsale.sol:120:3: Warning:
Function state mutability can be restricted to pure
  function _preValidatePurchase(
  ^ (Relevant source part starts here and spans across
multiple lines).
,openzeppelin-
solidity/contracts/crowdsale/Crowdsale.sol:135:3: Warning:
Function state mutability can be restricted to pure
  function _postValidatePurchase(
  ^ (Relevant source part starts here and spans across
multiple lines).
,openzeppelin-
solidity/contracts/crowdsale/Crowdsale.sol:177:3: Warning:
Function state mutability can be restricted to pure
  function _updatePurchasingState(
  ^ (Relevant source part starts here and spans across
multiple lines).

Writing artifacts to ./build/contracts
```

Tests output

```
 Contract: Token Generation Event
    ✓ should create ERC20 token contract (225ms)
    ✓ should create distributor contract (217ms)
    ✓ should init TokenTimelockFactory (99ms)
Dev fund - 1yr 5%
Timelock:  0x1d29693ab073f5a81ac431706427fc26be50cf91
Dev fund - 2yr 5%
Timelock:  0x2e5744eb9018b50908e44990918dee4d26a6d390
Dev fund - 3yr 22%
Timelock:  0x7ca84589462ef8916d5aa1dac1ca26f4190eb4d1
    ✓ should distribute dev fund tokens (32%) (433ms)
Founder 1 - 2yr 10%
Timelock:  0x83dce60066f2bb03101e054d81bcb20b849a33e0
Founder 2 - 2yr 10%
Timelock:  0xf10eeb8b0a9e8ec82920ff40d0285650e9dea151
    ✓ should distribute founder tokens (2 * 10%) (199ms)
Nodes - 6m 8%
Timelock:  0x47dae07bdc3d2c00132304072ce5e65346c0a805
    ✓ should distribute start node tokens (8%) (108ms)
Developers - 2yr 2.5%
Timelock:  0x134aeff96d8c162f113e7a6544206ad01d53ca7d
```

```
    ✓ should distribute developer tokens (2.5%) (101ms)
Advisors - 2yr 2.5%
Timelock:  0xb2c70bb020e393b28b976c4cfe4a019d924fcf21
    ✓ should distribute advisor tokens (2.5%) (109ms)
    ✓ should be able to finalize (93ms)
    after Finalization
      ✓ should create crowdsale (91ms)
      ✓ should disable deposits

  Contract: TokenCrowdsale
    ✓ fails to create if withdraw time is before closing time
(64ms)
    after opening
      ✓ can buy tokens (51ms)
      ✓ fails to withdraw (553ms)
      ✓ fails to withdraw for address (536ms)
      ✓ fails to withdraw for multiple address (591ms)
      ✓ fails to sell more than cap (87ms)
      ✓ can sell exactly until cap (111ms)
    after closing
      ✓ report as closed
      ✓ fails to withdraw (558ms)
      ✓ fails to withdraw for address (513ms)
      ✓ fails to withdraw for multiple address (506ms)
    after withdrawal
      ✓ can withdraw (521ms)
      ✓ fails to withdraw if not bought (563ms)
      ✓ can withdraw for address (532ms)
      ✓ can withdraw for multiple addresses (600ms)

  Contract: TokenDistributor
    when creating
      ✓ fails if benefactor address is 0x0
      ✓ fails if rate is zero (43ms)
      ✓ fails if wallet address is 0x0 (45ms)
      ✓ fails if token contract address is 0x0 (43ms)
      ✓ fails if cap is 0 (43ms)
      ✓ fails if opening time has passed (130ms)
      ✓ fails if opening time if after closing time (55ms)
      ✓ fails if withdraw time if before closing time (45ms)
      ✓ fails if bonus time if before closing time (55ms)
```

```
    as a Crowdsale Factory
      as a Presale Escrow
        ✓ can not deposit presale to distributor address
(611ms)
        ✓ can deposit presale tokens LoE to cap (783ms)
        ✓ fail to deposit bonus tokens above allowance (718ms)
        ✓ fail to deposit presale tokens above cap (572ms)
        after Finalization
          ✓ fails to deposit more presale tokens (484ms)
          after withdraw time
            ✓ can withdraw presale tokens for himself (523ms)
            ✓ can withdraw presale tokens for others (527ms)
            ✓ can withdraw presale tokens for others (533ms)
      as a Bonus Escrow
        ✓ can not deposit bonus to distributor address (63ms)
        ✓ can deposit bonus tokens LoE to cap (252ms)
        ✓ can deposit presale and bonus tokens LoE to cap
(888ms)
        ✓ can deposit presale and bonus tokens with wei LoE to
cap (728ms)
        ✓ fail to deposit bonus tokens above allowance (59ms)
        after Finalization
          ✓ fails to deposit more bonus tokens
          after withdraw time
            ✓ can withdraw bonus tokens for himself (844ms)
            ✓ can withdraw bonus tokens for others (707ms)
            ✓ can withdraw bonus tokens for multiple others
(550ms)
    as an IndividuallyCappedCrowdsale proxy
      ✓ fails to whitelist a user
      ✓ fails to whitelist a group
      after Finalization
        ✓ can whitelist users (59ms)
        ✓ can whitelist groups (76ms)
        before Crowdsale
          ✓ fails to claim leftover tokens
        during Crowdsale
          ✓ fails to claim leftover tokens
        after Crowdsale
          ✓ fails to claim leftover tokens
        after Withdraw time
          ✓ can claim leftover tokens (63ms)
```

```
          ✓ can claim leftover tokens twice (98ms)
    as a TokenTimelockFactory proxy
      ✓ can set Token Timelock Factory (72ms)
      when TimelockFactory set
        ✓ fails to reset Token Timelock Factory to another
address (66ms)
        ✓ fails to reset Token Timelock Factory to 0x0
        ✓ can not deposit and lock to distributor address
(97ms)
        ✓ can deposit and lock tokens (115ms)
        ✓ fails to deposit more than approved
        ✓ fails to deposit to 0x0 (41ms)
        ✓ fails to deposit with release time before withdraw
time
        ✓ can deposit twice to same user (173ms)
        ✓ fails to withdraw (174ms)
        after release time
          ✓ can withdraw (115ms)
        after Finalization
          ✓ fails to deposit and lock tokens
    as a TokenVestingFactory proxy
      ✓ can set Token Vesting Factory (93ms)
      when VestingFactory set
        ✓ fails to reset Token Vesting Factory to another
address (69ms)
        ✓ fails to reset Token Vesting Factory to 0x0
        ✓ can not deposit and vest to distributor address
        ✓ can deposit and vest tokens (108ms)
        ✓ fails to deposit to 0x0
        ✓ fails to deposit with release time before withdraw
time
        ✓ fails to deposit more than approved
        ✓ can deposit twice to same user (256ms)
        ✓ fails to withdraw (145ms)
        after duration time
          ✓ can withdraw (133ms)
        after Finalization
          ✓ fails to deposit and vest tokens
    as a Crowdsale Instatiator
      after finalization
        if cap reached
```

```
        ✓ there is no CrowdsaleInstantiated event
        ✓ crowdsale address is 0x0
        ✓ fails to whitelist a user
      if cap not reached
        ✓ emits a CrowdsaleInstantiated event
        ✓ instantiates the Crowdsale with correct allowance

Contract: TolarToken
  ✓ total suply is 1e+27
  ✓ owner owns all the tokens
  ✓ refuses ether
  as Finalizable Burnable Token
    when not finalized
      ✓ owner can burn tokens (82ms)
      ✓ other fail to burn tokens (57ms)
    when finalized
      ✓ should return unpaused
      ✓ owner can burn tokens (82ms)
      ✓ fails to burn tokens if not approved
      ✓ can burn own tokens (105ms)
      ✓ can burn all approved tokens (76ms)
      ✓ can burn some approved tokens (71ms)
      ✓ fails to burn more than approved amount (40ms)

Contract: Finalizable
  ✓ can be finalized (39ms)
  ✓ can execute onlyNotFinalized
  ✓ fails to execute onlyFinalized
  when finalized
    ✓ logs finalized
    ✓ cannot be finalized twice
    ✓ fails to execute onlyNotFinalized
    ✓ can execute onlyFinalized

Contract: TokenEscrow
  ✓ fails if benefactor is zero address
  as a TokenEscrow
    deposits
      ✓ can accept a single deposit (65ms)
      ✓ can not accept an deposit for itself
      ✓ can accept an empty deposit (73ms)
```

```
        ✓ only the owner can deposit (55ms)
        ✓ fails to deposit more than escrow allowance (50ms)
        ✓ can increase escrow allowance (88ms)
        ✓ fails to deposit to 0x0
        ✓ fails to deposit to escrow address
        ✓ emits a deposited event
        ✓ can add multiple deposits on a single account
(105ms)
        ✓ can track deposits to multiple accounts (161ms)
      withdrawals
        ✓ can withdraw payments (145ms)
        ✓ can do an empty withdrawal (42ms)
        ✓ only the owner can withdraw
        ✓ emits a withdrawn event (95ms)

  Contract: TokenTimelockEscrow
    ✓ fails if benefactor is zero address
    before release time
      ✓ reverts on withdrawals (218ms)
    after release time
      as a TokenEscrow
        deposits
          ✓ can accept a single deposit (58ms)
          ✓ can not accept an deposit for itself
          ✓ can accept an empty deposit (60ms)
          ✓ only the owner can deposit
          ✓ fails to deposit more than escrow allowance (73ms)
          ✓ can increase escrow allowance (104ms)
          ✓ fails to deposit to 0x0
          ✓ fails to deposit to escrow address
          ✓ emits a deposited event
          ✓ can add multiple deposits on a single account
(117ms)
          ✓ can track deposits to multiple accounts (244ms)
        withdrawals
          ✓ can withdraw payments (165ms)
          ✓ can do an empty withdrawal (64ms)
          ✓ only the owner can withdraw
          ✓ emits a withdrawn event (129ms)

  Contract: TokenTimelockFactory
```

```
    ✓ should not allow token 0x0 (52ms)
    ✓ should not allow beneficiary 0x0
    ✓ should not allow beneficiary factory
    as a TokenTimelockFactory
      ✓ can count istantiatinos
      ✓ emits ContractInstantiation event
      ✓ should track instantiations
      ✓ should have only one instantiation
      when child contract exists as TokenTimelock
        ✓ cannot be released before time limit (42ms)
        ✓ cannot be released just before time limit (53ms)
        ✓ can be released just after limit (76ms)
        ✓ can be released after time limit (100ms)
        ✓ cannot be released twice (135ms)

  Contract: TokenVestingFactory
    as a TokenVestingFactory
      ✓ should not allow beneficiary 0x0
      ✓ should not allow beneficiary factory
      ✓ can create TokenVesting contracts (65ms)
      ✓ can count istantiatinos (50ms)

  Contract: SafeStandardToken
    transfer
      ✓ returns true on success (87ms)
      ✓ fails to transfer tokens to 0x0 (54ms)
      ✓ fails to transfer tokens to token address (41ms)
    transferFrom
      ✓ returns true on success (141ms)
      ✓ fails to transferFrom tokens to 0x0 (57ms)
      ✓ fails to transferFrom tokens to token address (103ms)

  164 passing (1m)
```

Solhint output

```
contracts/lifecycle/Finalizable.sol
  46:36  warning  Code contains empty block  no-empty-blocks

contracts/Migrations.sol
```

```
   6:15  error  Variable name must be in mixedCase        var-
name-mixedcase
  20:28  error  Function param name must be in
mixedCase  func-param-name-mixedcase

contracts/mocks/FinalizableMock.sol
   9:45  warning  Code contains empty block  no-empty-blocks
  13:51  warning  Code contains empty block  no-empty-blocks

contracts/mocks/TokenTimelockEscrowMock.sol
  13:3  error    Open bracket must be on same line. It must be
indented by other constructions by space  bracket-align
  13:3  warning  Code contains empty
block
      no-empty-blocks

contracts/payment/TokenTimelockEscrow.sol
  20:28  warning  Avoid to make time-based decisions in your
business logic  not-rely-on-time
  28:38  warning  Variable "_payee" is
unused                                 no-unused-vars
  30:12  warning  Avoid to make time-based decisions in your
business logic  not-rely-on-time

contracts/TokenCrowdsale.sol
   28:1   error    Open bracket must be on same line. It must
be indented by other constructions by space  bracket-align
   54:3   error    Open bracket must be on same line. It must
be indented by other constructions by space  bracket-align
   81:51  error    Expression indentation is incorrect.
Required no spaces after i                          expression-
indent
  122:13  warning  Avoid to make time-based decisions in your
business logic                                 not-rely-on-time

contracts/TokenDistributor.sol
   88:3   error    Open bracket must be on same line. It must
be indented by other constructions by space  bracket-align
   95:28  warning  Avoid to make time-based decisions in your
business logic                                 not-rely-on-time
  183:51  error    Expression indentation is incorrect.
Required no spaces after i                          expression-
indent
  219:51  error    Expression indentation is incorrect.
Required no spaces after i                          expression-
indent
```

```
  353:13  warning  Avoid to make time-based decisions in your
business logic                                not-rely-on-time

contracts/TokenTimelockEscrowImpl.sol
  15:3  error    Open bracket must be on same line. It must be
indented by other constructions by space  bracket-align
  15:3  warning  Code contains empty
block
      no-empty-blocks

✖ 21 problems (10 errors, 11 warnings)
```

Solium output

```
No issues found.
```