SmartDec

Temporal Smart Contracts Security Analysis

This report is public.

Published: August 22, 2018

Abstract

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

The audit showed no critical issues. However, a number of medium and low severity issues were found. They do not endanger project security. However, we highly recommend addressing them.

General recommendations

The contracts code is of good code quality. The contracts code does not contain issues that endanger project security. However, we recommend limiting Power of the owner and fixing ERC20 mismatch.

In addition, if the developer decides to improve the code, we recommend following best practices for Hardcoded addresses, removing Redundant code, and fixing Fallback function.

However, mentioned above are minor issues. They do not influence code operation.

The text below is for technical use; it details the statements made in Summary and General 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 provided documentation.

ERC20 compliance

We have checked ERC20 compliance during the audit. The audit has shown that RTCoin contract is not ERC20 compliant.

ERC20 MUST

The audit has shown one ERC20 “MUST” requirements violations.

ERC20 SHOULD

The audit has shown no ERC20 “SHOULD” requirements.

Tests

Tests could not be run.

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
- report
    - we reflect all the gathered information in the report

Checked vulnerabilities

We have scanned Temporal 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 Temporal documentation and smart contracts code (version on commit f45d59a59c3fdcbe0b6e1ac9f8f195582a0b0445).

Project architecture

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

- **Administration.sol**
- **MergedMiner.sol**
- **PaymentV2.sol**
- **RTC-ETH.sol**
- **RTCoin.sol**
- **Stake.sol**
- **Vesting.sol**

The files successfully compile with compiler version 0.4.24.

These files contain the following contracts:

- **Vesting** - contains vesting functionality
- **Stake** - contains token deposit functionality
- **RTCoin** - token contract
- **RTCETH** - token sale contract
- **MergedMinerValidator** - contains block validator functionality
- **Payments** - contains receipt and validation of signed payments functionality

The total LOC of audited Solidity code is 918.

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	Constant but potentially should not be	2	
	Fallback function requires too much gas	1	
	Gas requirement of function high: infinite	28	
	Potential Violation of Checks-Effects-Interaction pattern	5	
	Use of "now"	4	
Total Remix		40	
SmartCheck	Address Hardcoded	2	6
	Gas Limit And Loops	3	
	Private Modifier Dont Hide Data	10	
	Should Return Struct	1	
	Timestamp Dependence	1	

Total SmartCheck		17	6
Solhint	Avoid to make time-based decisions in your business logic	6	
	Event and function names must be different	1	
Total Solhint		7	
Total Overall		**64**	**6**

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.

ERC20 standard violation

According to ERC20 standard:

```
Allows _spender to withdraw from your account multiple times,
up to the _value amount. If this function is called again it
overwrites the current allowance with _value.
```

However, in `approve()` function (**RTCoin.sol**, line 129):

```
allowed[msg.sender][_spender] =
allowed[msg.sender][_spender].add(_amount);
```

Moreover, making `approve()` function irreversible may affect some types of business logics.

We highly recommend following standard and implementing `increaseApproval()/decreaseApproval()` functions (they can be found in OpenZeppelin library).

Also, we recommend instructing users to follow one of two ways:

- not to use `approve()` function directly and to use `increaseApproval()/decreaseApproval()` functions instead
- to change the approved amount to 0, wait for the transaction to be mined, and then to change the approved amount to the desired value

Overpowered owner

RTCoin contract owner has the following powers:

1. The owner can grant minting role to anyone, including himself.
   RTCoin.sol, lines 138,165:

```
function setMergedMinerValidator(address
_mergedMinerValidator) external onlyOwner returns (bool)
function setFailOverStakeContract(address _contractAddress)
external onlyOwner returns (bool)
```

2. The owner can mint tokens without any limits. **RTCoin.sol**, line 182:

```
function mint(address _recipient, uint256 _amount) public
onlyMinters returns (bool)
```

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.

Bad design of the ownership transfer

There are two issues in `startOwnerTransferDelay()` function (**Administration.sol**, line 78):

1. `_newOwner` address is not checked for being non-zero.
2. `startOwnerTransferDelay()` function can be called second time only after `transferOwnership()` function call. The reason is `noPendingDelay` modifier, which can be changed only in `transferOwnership()` function. As a result, during the call of the `startOwnerTransferDelay()` function with a wrong argument, functionality of the owner transfer would be lost. Moreover, admin has a permission to this function.

We highly recommend checking admin permissions, logic of ownership transfer procedure, and adding the following line:

```
require(_newOwner != address(0));
```

Low severity issues

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

Hardcoded address

The following files contain hardcoded addresses in their contracts:

- **Stake.sol**, line 28:

```
address  public TOKENADDRESS =
0xE9AEc23c620681a59e2111785b0D35a90498128f;
```

- **PaymentV2.sol**, lines 18, 19, 20:

```
address constant private SIGNER =
0xC6C35f43fDD71f86a2D8D4e3cA1Ce32564c38bd9;
address constant private TOKENADDRESS =
0xC6C35f43fDD71f86a2D8D4e3cA1Ce32564c38bd9;
address constant private HOTWALLET =
0xC6C35f43fDD71f86a2D8D4e3cA1Ce32564c38bd9;
```

- **RTC_ETH.sol**, line 16:

```
address constant private TOKENADDRESS = address(0);
```

- **Vesting.sol**, line 15:

```
address constant public TOKENADDRESS =
0xB8fe3B2C83014566733B766a27d94CB9AC167Dc6;
```

Using hardcoded addresses is not a bad practice, however these addresses might be used for some malicious activity. We recommend checking these addresses.

Deploy

There is no deployment script. However, the contracts deployment does not seem trivial. Bugs and vulnerabilities often appear in deployment scripts and severely endanger system's security.
We highly recommend developing and testing deployment scripts very carefully.

Misleading comments

The comments at **RTC-ETH.sol**, lines 45 and 56 are misleading:

```
// place holder
```

We recommend fixing this comment in order to improve code readability.

Rounding

There is a rounding issue at **RTC-ETH.sol**, line 116:

```
uint256 rtcPurchased = (msg.value.div(weiPerRtc)).mul(1
ether);
```

We highly recommend multiplying before division to increase the rounding precision.

Repeated emit

There are two repeatedly emitted events at **RTC-ETH.sol**, lines 117, 119:

```
emit RtcPurchased(rtcPurchased);
```

We recommend removing duplicated lines and emitting events in the end of the function in order to improve contract design.

Locked tokens

The following contracts have no ability to withdraw mistakenly sent tokens:

- **RTC-ETH**
- **Stake**

We recommend implementing an additional function to withdraw tokens from contracts which can receive tokens directly.

Redundant code

The following lines are redundant:

1. **Stake.sol**, line 309:

   ```
   function calculateTotalCoinsMinted(uint256 _numRTC) internal
   pure returns (uint256 totalCoinsMinted)
   ```

   The function is internal and is not used anywhere.

2. **MergedMiner.sol**, line 95:

   ```
   unction submitBlock() public nonSubmittedBlock(block.number)
   notCurrentSetBlock(block.number) returns (bool)
   ```

   `notCurrentSetBlock()` modifier implements a redundant check that is already made in `nonSubmittedBlock()` modifier.

3. **MergedMiner.sol**, line 116:

```
function claimReward(uint256 _blockNumber) internal
isCoinbase(_blockNumber) unclaimed(_blockNumber)
submittedBlock(_blockNumber) returns (uint256)
```

`unclaimed()` and `submittedBlock()` modifiers contain similar checks.

4. **MergedMiner.sol**, line 136:

```
require(blocks[_blockNumbers[i]].state ==
BlockStateEnum.claimed, "block state is not claimed");
```

This check will always pass due to the line 121.

5. **Administration.sol**, line 83:

```
delayExpirationTime: now.add(uint256(86400).mul(1 seconds)),
```

Checking the multiplication by one is excessive.

RTC-ETH.sol, lines 79, 81:

```
uint256 oneUsdOfEth = oneEth.div(ethUSD);
weiPerRtc = oneUsdOfEth.div(8);
```

Checking the division by eight is excessive as the division by zero is reverted by EVM.

Stake.sol, line 169:

```
stakes[msg.sender][_stakeNumber].coinsMinted =
stakes[msg.sender][_stakeNumber].coinsMinted.add(mintAmount);
```

Checking the overflow after adding `mintAmount` is excessive, as it is impossible to mint more than `totalSupply`, which is **uint256**.

RTCoin.sol, lines 84,111:

```
balances[_recipient] = balances[_recipient].add(_amount);
```

Checking the overflow after adding `_amount` is excessive, as balance is limited by `totalSupply`.

RTCoin.sol, lines 83, 107, 109 .

```
balances[msg.sender] = balances[msg.sender].sub(_amount);
allowed[_owner][msg.sender] =
allowed[_owner][msg.sender].sub(_amount);
balances[_owner] = balances[_owner].sub(_amount);
```

Using of **SafeMath** is excessive according to the operations in the previous lines

6. **Stake.sol**, line 164:

```
require(stakes[msg.sender][_stakeNumber].coinsMinted.add(mintA
mount) <= stakes[msg.sender][_stakeNumber].totalCoinsToMint,
"total coins minted does not add up");
```

This check is redundant according to similar check at lines 301-303.

7. **RTCoin.sol**, line 212:

```
require(eI.balanceOf(address(this)) >= _amount, "attempting to
send more tokens than current balance");
```

This check is redundant according to similar check in `transfer()` function called in the next line.

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

Code logic issue

ETH is transferred to `HOTWALLET` address at every `makePayment()` function call (**PaymentV2.sol**, line 106):

```
HOTWALLET.transfer(msg.value);
```

We recommend transferring ETH from contract to the wallet using additional `withdraw()` function in order to decrease gas consumption.

Fallback abuse

There is a missing check at **RTC-ETH.sol**, line 66:

```
function () external payable {
        require(buyRtc());
    }
```

We highly recommend implementing the following line:

```
require(msg.data.length == 0);
```

in order to prevent accidental purchase of tokens.

Burn functionality

There are no checks for zero address in `transfer()`, `transferFrom()`, `transferForeignToken()` functions. It can lead to the lock of tokens.

We highly recommend implementing missing check or making additional function implementing burn functionality.

Flaw in the documentation

There is a use of “Proof Of Stake” term in the provided documentation. This term is already used for the different concept.
We recommend using another term in order to avoid confusion.

This analysis was performed by SmartDec.

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

Sergey Pavlin, Chief Operating Officer

August 22, 2018

Appendix

Solhint output

```
./Administration.sol
   6:1   error    Definition must be surrounded with two blank
line indent             two-lines-top-level-separator
  16:5   error    Definitions inside contract / library must
be separated by one line  separate-by-one-line-in-contract
  46:17  warning  Avoid to make time-based decisions in your
business logic            not-rely-on-time
  83:34  warning  Avoid to make time-based decisions in your
business logic            not-rely-on-time

./ERC20Interface.sol
   7:1  error  Definition must be surrounded with two blank
line indent                                    two-lines-top-
level-separator
  10:5  error  Function order is incorrect, external function
can not go after external constant function  func-order
  11:5  error  Function order is incorrect, external function
can not go after external constant function  func-order
  12:5  error  Function order is incorrect, external function
can not go after external constant function  func-order

./MergedMiner.sol
   10:2  error  Line length must be no more than 120 but
current length is 140                     max-line-length
   11:1  error  Definition must be surrounded with two blank
line indent                              two-lines-top-level-
separator
  113:5  error  Definitions inside contract / library must be
separated by one line                 separate-by-one-line-in-
contract
  129:5  error  Function order is incorrect, external function
can not go after internal function  func-order
  148:5  error  Function order is incorrect, public function
can not go after internal function    func-order
  156:5  error  Function order is incorrect, public function
can not go after internal function    func-order

./PaymentV2.sol
    9:2   error  Line length must be no more than 120 but
current length is 140                          max-line-length
```

```
   10:1   error  Definition must be surrounded with two blank
line indent                                  two-lines-top-
level-separator
   16:2   error  Line length must be no more than 120 but
current length is 143                          max-line-length
   26:22  error  Open bracket must be on same line. It must be
indented by other constructions by space  bracket-align
   28:23  error  Open bracket must be on same line. It must be
indented by other constructions by space  bracket-align
   75:2   error  Line length must be no more than 120 but
current length is 131                          max-line-length
  110:2   error  Line length must be no more than 120 but
current length is 127                          max-line-length
  114:2   error  Line length must be no more than 120 but
current length is 127                          max-line-length
  196:91  error  Open bracket must be indented by other
constructions by space                           bracket-align
  202:5   error  Function order is incorrect, public function
can not go after internal function         func-order

./RTC-ETH.sol
  11:2  error  Line length must be no more than 120 but
current length is 140                        max-line-length
  12:1  error  Definition must be surrounded with two blank
line indent                               two-lines-top-level-
separator
  60:5  error  Function order is incorrect, constructor
function can not go after public function  func-order
  66:5  error  Function order is incorrect, fallback function
can not go after public function     func-order

./RTCoinInterface.sol
   4:1  error  Definition must be surrounded with two blank
line indent                                     two-lines-top-
level-separator
  22:5  error  Function order is incorrect, external function
can not go after external constant function  func-order
  29:5  error  Function order is incorrect, external function
can not go after external constant function  func-order
  31:5  error  Function order is incorrect, external function
can not go after external constant function  func-order

./RTCoin.sol
   11:2  error    Line length must be no more than 120 but
current length is 140                    max-line-length
```

```
   12:1  error    Definition must be surrounded with two blank
line indent                          two-lines-top-level-
separator
   73:5  warning  Event and function names must be
different                                        no-simple-
event-func-name
  138:5  error    Definitions inside contract / library must
be separated by one line               separate-by-one-line-in-
contract
  138:5  error    Function order is incorrect, external
function can not go after public function  func-order
  148:5  error    Function order is incorrect, external
function can not go after public function  func-order
  151:2  error    Line length must be no more than 120 but
current length is 126                    max-line-length
  160:2  error    Line length must be no more than 120 but
current length is 129                    max-line-length
  161:2  error    Line length must be no more than 120 but
current length is 123                    max-line-length
  162:2  error    Line length must be no more than 120 but
current length is 152                    max-line-length
  165:5  error    Function order is incorrect, external
function can not go after public function  func-order
  262:5  error    Definitions inside contract / library must
be separated by one line               separate-by-one-line-in-
contract

./SafeMath.sol
  4:1  error  Definition must be surrounded with two blank
line indent  two-lines-top-level-separator

./StakeInterface.sol
  4:1  error  Definition must be surrounded with two blank
line indent  two-lines-top-level-separator

./Stake.sol
   10:2   error    Line length must be no more than 120 but
current length is 140                    max-line-length
   11:1   error    Definition must be surrounded with two
blank line indent                          two-lines-top-level-
separator
   23:2   error    Line length must be no more than 120 but
current length is 142                    max-line-length
   24:2   error    Line length must be no more than 120 but
current length is 175                    max-line-length
```

```
   28:21  error    Variable name must be in
mixedCase                                                  var-
name-mixedcase
   30:30  error    Variable name must be in
mixedCase                                                  var-
name-mixedcase
   65:2   error    Line length must be no more than 120 but
current length is 144                   max-line-length
   82:2   error    Line length must be no more than 120 but
current length is 124                   max-line-length
   82:13  warning  Avoid to make time-based decisions in your
business logic                         not-rely-on-time
   85:2   error    Line length must be no more than 120 but
current length is 127                   max-line-length
   92:2   error    Line length must be no more than 120 but
current length is 144                   max-line-length
  152:2   error    Line length must be no more than 120 but
current length is 139                   max-line-length
  165:2   error    Line length must be no more than 120 but
current length is 126                   max-line-length
  207:10  error    Expected indentation of 8 spaces but found
9                                      indent
  251:5   error    Definitions inside contract / library must
be separated by one line               separate-by-one-line-in-
contract
  269:23  warning  Avoid to make time-based decisions in your
business logic                         not-rely-on-time
  336:5   error    Function order is incorrect, public
function can not go after internal function  func-order
  337:2   error    Line length must be no more than 120 but
current length is 139                   max-line-length

./Vesting.sol
    9:2   error    Line length must be no more than 120 but
current length is 140       max-line-length
   10:1   error    Definition must be surrounded with two
blank line indent             two-lines-top-level-separator
   40:2   error    Line length must be no more than 120 but
current length is 145       max-line-length
   46:17  warning  Avoid to make time-based decisions in your
business logic            not-rely-on-time
   97:2   error    Line length must be no more than 120 but
current length is 135       max-line-length
  102:21  warning  Avoid to make time-based decisions in your
business logic            not-rely-on-time
  112:2   error    Line length must be no more than 120 but
current length is 133       max-line-length
```

```
  118:2   error    Line length must be no more than 120 but
current length is 150       max-line-length
  119:2   error    Line length must be no more than 120 but
current length is 133       max-line-length
  122:5   error    Definitions inside contract / library must
be separated by one line  separate-by-one-line-in-contract
  138:2   error    Line length must be no more than 120 but
current length is 128       max-line-length

✖ 75 problems (68 errors, 7 warnings)
```

Solium output

```
Administration.sol
  2:0      warning    Avoid using experimental features in
production code    no-experimental
  46:16    warning    Avoid using 'now' (alias to
'block.timestamp').         security/no-block-members
  83:33    warning    Avoid using 'now' (alias to
'block.timestamp').         security/no-block-members

ERC20Interface.sol
  2:0    warning    Avoid using experimental features in
production code    no-experimental

MergedMiner.sol
  2:0    warning    Avoid using experimental features in
production code    no-experimental

PaymentV2.sol
  2:0    warning    Avoid using experimental features in
production code    no-experimental

RTC-ETH.sol
  2:0    warning    Avoid using experimental features in
production code    no-experimental

RTCoin.sol
  2:0    warning    Avoid using experimental features in
production code    no-experimental

RTCoinInterface.sol
  2:0    warning    Avoid using experimental features in
production code    no-experimental
```

```
SafeMath.sol
  2:0    warning    Avoid using experimental features in
production code    no-experimental

Stake.sol
  2:0       warning    Avoid using experimental features in
production code    no-experimental
  82:12     warning    Avoid using 'now' (alias to
'block.timestamp').         security/no-block-members
  269:22    warning    Avoid using 'now' (alias to
'block.timestamp').         security/no-block-members

StakeInterface.sol
  2:0    warning    Avoid using experimental features in
production code    no-experimental

Vesting.sol
  2:0       warning    Avoid using experimental features in
production code    no-experimental
  46:16     warning    Avoid using 'now' (alias to
'block.timestamp').         security/no-block-members
  102:20    warning    Avoid using 'now' (alias to
'block.timestamp').         security/no-block-members

✖ 17 warnings found.
```