SmartDec

ROOBEE Token
Security Analysis

This report is public.

Published: April 19, 2019.

Abstract . . . 2
Disclaimer . . . 2
Summary . . . 2
General recommendations . . . 2
Checklist . . . 3
Procedure . . . 4
Checked vulnerabilities . . . 4
Project overview . . . 5
The latest version of the code . . . 5
Audit results . . . 5
Critical issues . . . 5
Medium severity issues . . . 5
Overpowered owner . . . 5
Misleading comment (fixed) . . . 6
Low severity issues . . . 6
Notes . . . 7
Approve function of the ERC20 token standard . . . 7

Abstract

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

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 considered the security of ROOBEE smart contracts. We performed our audit according to the procedure described below.

The initial audit showed neither critical issues nor low severity issues. However, two medium severity issues were found. They do not endanger project security. One of the issues was fixed in the latest version of the code.

After the fixes, the token contract was deployed. The issues found during the audit do not endanger the project security.

General recommendations

The contracts code is of good code quality. The audit did not reveal any issues that endanger project security. However, we recommend implementing token distribution logic.

Checklist

Security

The audit showed 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.

Arithmetics

Token smart contracts are secure from arithmetics issues.

ERC20 compliance

We checked ERC20 compliance during the audit. The audit showed that **Roobee** contract was fully ERC20 compliant.

ERC20 MUST

The audit showed no ERC20 “MUST” requirements violations.

ERC20 SHOULD

The audit showed no ERC20 “SHOULD” requirements violations.

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

Procedure

We perform our audit according to the following procedure:

- we manually analyze smart contracts for security vulnerabilities
- we scan the code with several publicly available automated Solidity analysis tools (SmartCheck, Remix) during manual analysis
- we check ERC20 compliance
- we check smart contracts logic and compare it with the one described in the specification
- we reflect all the gathered information in the report

Checked vulnerabilities

We have scanned Roobee token's code 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

Project overview

In our analysis we consider Roobee token's code (version on commit 9cae8606992965b9f32d43903105a4b14d329b4a) and specification ("README.md" in the repository).

The latest version of the code

After the initial audit, some fixes were applied and the code was updated to the latest version (commit 16431b13ca376a8ad5375b49cacc2d71a1029040).

Audit results

The token's code was completely manually analyzed and checked for ERC20 standard compliance.

Token's logic was also checked and compared with the one described in the specification.

All the found issues (bugs, vulnerabilities, and ERC20 standard violations) are described below.

Critical issues

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

The audit showed no critical issues.

Medium severity issues

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

Overpowered owner

The documentation contains information about token distribution. However, the code does not contain such logic. In the current implementation, the contract owner should manually mint tokens to desired addresses. Since there is no restrictions on destination addresses, 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.

Misleading comment (fixed)

The comment at **Roobee.sol**, lines 359–364 is controversial:

```
// This portion is copied from ConsenSys's Standard Token
Contract. It
// calls the receiveApproval function that is part of the
contract that
// is being approved (`_spender`). The function should look
like:
// `receiveApproval(address _from, uint256 _amount, address
// _tokenContract, bytes _extraData)` It is assumed that the
call
// *should* succeed, otherwise the plain vanilla approve
would be used
```

According to the comment, the following code should invoke function `receiveApproval(address _from, uint256 _amount, address _tokenContract, bytes _extraData)`. However, at line 365 `approvalFallback(address _from, uint256 _value, address _token, string memory _extraData)` function is called. Note that the function names are different and types of the last argument do not match different. This can result in contract interaction issues.

We highly recommend checking `approvalFallback()` function's arguments and name and comparing them with the ones that will be used with the token contract.

The issue has been fixed 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.

The audit showed no low severity issues.

Notes

Approve function of the ERC20 token standard

There is ERC20 approve issue: changing the approved amount from a nonzero value to another nonzero value allows a double spending with a front-running attack.

We recommend instructing users to follow one of two ways:

- not to use `approve()` function directly and to use `increaseAllowance()`/`decreaseAllowance()` 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

This analysis was performed by SmartDec.

Boris Nikashin, Project Manager
Igor Sobolev, Analyst
Pavel Kondratenkov, Analyst

April 19, 2019