SmartDec

Faireum Smart Contracts Security Analysis

This report is public.

Published: January 17, 2019

Abstract

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

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

General recommendations

The contracts code is of good quality. The contracts code does not contain issues that endanger project security. However, we recommend covering Code with tests, fixing Documentation mismatch, and removing Redundant code.

In addition, if the developers decide to improve the code, we recommend following best practices for Pragmas version. However, these issues are minor and do not influence code operation.

Checklist

Smart Contracts 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.

Compliance with the documentation

The audit showed discrepancies between the code and the provided documentation.

ERC20 compliance

We checked ERC20 compliance during the audit. The audit showed that **FaireumToken** 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.

Tests

There are no tests provided with the code.

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

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 reflect all the gathered information in the report

Checked vulnerabilities

We have scanned Faireum 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 Faireum whitepaper (“Faireum_whitepaper2.0_En.pdf”, sha1sum: 630f7cd253fa9e61f56e0b51e777857bb943b6da) and smart contracts code (“FaireumToken.sol”, sha1sum: 637e2aa0797f95d5853cf4f3c692a36853977c03).

For the audit, we were provided with the **FaireumToken.sol** file.

- The file successfully compiles with solc command.

The total LOC of audited Solidity sources is 273.

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).

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.

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

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

Tool	Rule	False positives	True positives
Remix	Gas requirement of function high: infinite	24	
	Should be constant but is not	2	
	Use of "now"	3	
	Use of "this" for local functions	1	
Total Remix		29	
SmartCheck	Erc20 Approve		2
	Pragmas Version		1
	Private Modifier Don’t Hide Data	4	
	Safemath	2	
Total SmartCheck		6	3

Solhint	Avoid to make time-based decisions in your business logic	3	
	Compiler version must be fixed		1
	Event and function names must be different	3	
Total Solhint		6	1
Total Overall		**42**	**4**

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 showed no critical issues.

Medium severity issues

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

No tests and deployment script

The provided code does not contain tests. Testing is crucial for code security and audit does not replace tests in any way.
We highly recommend both covering the code with tests and making sure that the test coverage is sufficient.

There is also 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.

Low severity issues

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

Documentation mismatch

There is a discrepancy between the smart contracts code and the whitepaper:
According to the documentation (“Faireum_whitepaper2.0_En.pdf”, page 35):

```
Team/Developer/Advisor: Locked for a minimum of 12 months and
followed by 1 to 2 years vesting schedule to the long-term
benefit of the team/developer/advisor.
```

However, half of the tokens locked for 6 months and another half for a year in the code.

```
function lockTeamTokens(address _beneficiary, uint256
_tokensAmount) public onlyAdmin {
    uint256 _half = _tokensAmount.div(2);
    _lockTokens(address(teamAdvisorsTokensVault), false,
_beneficiary, _half);
    _lockTokens(address(teamAdvisorsTokensVault), true,
_beneficiary, _half);
}
```

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

Redundant code

SafeERC20 library at **FaireumToken.sol**, lines 357-369 is redundant, as it is not used in the project.

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

Pragma version

Solidity source files indicate the versions of the compiler they can be compiled with.
Example:

```
pragma solidity ^0.4.24; // bad: compiles w 0.4.24 and above
pragma solidity 0.4.24; // good: compiles w 0.4.24 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 compiler version 0.4.25. Moreover, new version of compiler (0.5.2 at the moment) is available and contains several important changes. In order to update your contracts to compiler version 0.5.2 the developers should follow solidity documentation.

Notes

ERC20 approve issue

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 `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.

This analysis was performed by SmartDec.

Boris Nikashin, Project Manger
Alexander Drygin, Analyst
Pavel Kondratenkov, Analyst
Kirill Gorshkov, Analyst

Sergey Pavlin, Chief Operating Officer

January 17, 2019

Appendix

Solhint output

```
FaireumToken.sol
    1:17  warning  Compiler version must be
fixed                                                              c
ompiler-fixed
    7:1   error    Definition must be surrounded with two blank line
indent                                    two-lines-top-level-
separator
   71:1   error    Definition must be surrounded with two blank line
indent                                    two-lines-top-level-
separator
  106:1   error    Definition must be surrounded with two blank line
indent                                    two-lines-top-level-
separator
  150:1   error    Definition must be surrounded with two blank line
indent                                    two-lines-top-level-
separator
  157:5   error    Function order is incorrect, external function
can not go after external constant function  func-order
  159:5   error    Function order is incorrect, external function
can not go after external constant function  func-order
  161:5   error    Function order is incorrect, external function
can not go after external constant function  func-order
  163:5   warning  Event and function names must be
different                                                    no-
simple-event-func-name
  173:2   error    Line length must be no more than 120 but current
length is 128                              max-line-length
  179:1   error    Definition must be surrounded with two blank line
indent                                    two-lines-top-level-
separator
  219:5   warning  Event and function names must be
different                                                    no-
simple-event-func-name
  357:1   error    Definition must be surrounded with two blank line
indent                                    two-lines-top-level-
separator
  376:1   error    Definition must be surrounded with two blank line
indent                                    two-lines-top-level-
separator
  405:1   error    Definition must be surrounded with two blank line
```

```
indent                                        two-lines-top-level-
separator
  408:28  error    Constant name must be in capitalized
SNAKE_CASE                                                  const-name-
snakecase
  409:28  error    Constant name must be in capitalized
SNAKE_CASE                                                  const-name-
snakecase
  410:27  error    Constant name must be in capitalized
SNAKE_CASE                                                  const-name-
snakecase
  437:49  warning  Avoid to make time-based decisions in your
business logic                                        not-rely-on-time
  438:13  warning  Avoid to make time-based decisions in your
business logic                                        not-rely-on-time
  441:16  error    Statement indentation is incorrect. Required
space after if                                      statement-indent
  449:26  warning  Avoid to make time-based decisions in your
business logic                                        not-rely-on-time
  487:9   error    Statement indentation is incorrect. Required
space after if                                      statement-indent
  497:5   error    Function order is incorrect, public function can
not go after internal function              func-order
  502:5   error    Function order is incorrect, public function can
not go after internal function              func-order
  507:5   error    Function order is incorrect, public function can
not go after internal function              func-order
  514:5   error    Function order is incorrect, public function can
not go after internal function              func-order
  519:5   error    Function order is incorrect, public function can
not go after internal function              func-order
  524:5   error    Function order is incorrect, public function can
not go after internal function              func-order
  528:5   error    Function order is incorrect, public function can
not go after internal function              func-order
  532:5   warning  Event and function names must be
different                                                      no-
simple-event-func-name
  532:5   error    Function order is incorrect, public function can
not go after internal function              func-order
  536:5   error    Function order is incorrect, public function can
not go after internal function              func-order
  541:59  error    Visibility modifier must be first in list of
modifiers                                         visibility-modifier-
order
  541:5   error    Function order is incorrect, external function
can not go after internal function            func-order
```

```
  543:9   error    Statement indentation is incorrect. Required
space after if                                    statement-indent

✖ 36 problems (29 errors, 7 warnings)
```

Solium output

```
FaireumToken.sol
  437:48    warning    Avoid using 'now' (alias to
'block.timestamp').    security/no-block-members
  438:12    warning    Avoid using 'now' (alias to
'block.timestamp').    security/no-block-members
  449:25    warning    Avoid using 'now' (alias to
'block.timestamp').    security/no-block-members

✖ 3 warnings found.
```