SmartDec

Joyso Smart Contracts Security Analysis

This report is public.
Version 1.3

Published: May 22, 2018

Abstract

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

The initial audit has shown no critical issues. However, several medium and low severity issues have been found. All of them were successfully fixed by the developer in the latest version of the code.

General recommendations

The latest version of the code is of good code quality and does not contain issues that endanger project security.

Nevertheless, if the developer decides to improve the code, we recommend addressing Unchecked math issue. In addition, we recommend conducting an audit of tokens added to the project.

However, these are minor issues, which do not influence code operation.

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, Oyente, 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 report all the issues found to the developer during the audit process
    - we check the issues fixed by the developer
    - we reflect all the gathered information in the report

Checked vulnerabilities

We have scanned Joyso 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 Joyso smart contracts code (Git repository, version on commit 67b0c6c or joyso-contracts-67b0c6c210fd2034734df57987b917fa72b9d103.zip, sha1sum 4fd1b984f2ca9de4e90d4943e1c1cec31de25f5d).

The latest version of the code

We have performed the check of the fixed vulnerabilities in the latest version of the code (Git repository, version on commit 1c54bf8c0ffc041a18e1f03f59e0fed1798322ab).

Project architecture

For the audit, we have been provided with the truffle project. The project also contains tests, deploy scripts, and several files that are beyond the scope of the audit.

- The project successfully compiles with `truffle compile` command (see Compilation output in Appendix)
- The project successfully passes all the tests (`truffle test` command, see Tests output in Appendix)

The project includes the following files:

- Joyso.sol
- JoysoDataDecoder.sol
- Migratable.sol
- libs/SafeMath.sol

The main scope of the audit is Joyso.sol and its dependencies. This file contains a contract of the same name Joyso. It inherits Ownable contract from OpenZeppelin library version 1.6.0 or higher, JoysoDataDecoder contract, and uses SafeMath library.

The total volume of audited files is 563 lines of Solidity code.

Code logic

Joyso is token exchange contract that allows users to deposit allowed tokens and ETH, exchange tokens and ETH, withdraw tokens and ETH. Administrator and contract owner can process user orders, add new administrators, add new tokens, process user request for withdrawal.

The contract inherits Ownable contract from OpenZeppelin library. The contract has complex data encoding and decoding logic, uses several cryptographic functions to sign and verify user data.

List of publicly available functions with descriptions by SmartDec team:

```
function Joyso (address _joysoWallet, address _joyToken) public
```

Call restrictions:
- everyone can make a call

Parameters requirements:
- no requirements

Logic:
- contract constructor
- the function initializes several variables
  - beneficiary wallet as `_joysoWallet`
  - fee payment token as `_joyToken`
  - first admin as contract creator address

```
function depositToken (address token, uint256 amount) external
```

Call restrictions:
- everyone can make a call

Parameters requirements:
- `token` address must be listed in `tokenAddress2Id` mapping

Logic:
- function adds caller to user list – `userAddress2Id` mapping
- function transfers tokens of `token` contract from caller address to contract address

```
function depositEther () external payable
```

Call restrictions:
- everyone can make a call

Parameters requirements:
- no requirements

Logic:
- function adds caller to user list – `userAddress2Id` mapping
- function accepts ETH from caller address to Joyso contract address

```
function withdraw (address token, uint256 amount) external
```

Call restrictions:

- everyone can make a call

Parameters requirements:

- `token` address must be listed in `address2Id` mapping or be `0`
- current timestamp must be greater than user locked block

Logic:

- function transfers tokens of `token` contract from Joyso contract address to caller if `balances` allows it
- function transfers ETH from Joyso contract address to caller if `token` is `0` and `balances` mapping allows it

```
function lockMe () external
```

Call restrictions:

- everyone can make a call

Parameters requirements:

- no requirements

Logic:

- function locks caller account for withdrawal for 30 days

```
function unlockMe () external
```

Call restrictions:

- everyone can make a call

Parameters requirements:

- no requirements

Logic:

- function unlocks caller account for withdrawal

```
function getBalance (address token, address account) external view
returns (uint256)
```

Call restrictions:

- everyone can make a call

Parameters requirements:

- no requirements

Logic:

- function returns current `token` balance of `address` user

```
function registerToken (address tokenAddress, uint256 index)
external onlyAdmin
```

Call restrictions:

- only from address listed in `isAdmin` mapping or owner address

Parameters requirements:

- no requirements

Logic:

- function adds `token` address as token listed with `index` in corresponding mappings

```
function addToAdmin (address admin, bool isAdd) onlyAdmin external
```

Call restrictions:

- only from address listed in `isAdmin` mapping or owner address

Parameters requirements:

- no requirements

Logic:

- function adds or removes `admin` address from `isAdmin` mapping

```
function withdrawByAdmin_Unau (uint256[] inputs) external onlyAdmin
```

Call restrictions:

- only from address listed in `isAdmin` mapping or owner address

Parameters requirements:

- all input parameters must meet set of requirements, critical ones are
  - data for withdraw request must be signed by corresponding user
  - user must have enough `balance` of tokens or ETH
  - checks whether orders were not canceled

Logic:

- call forces withdraw of tokens and ETH only by user signed request

```
function matchByAdmin_TwH36 (uint256[] inputs) external onlyAdmin
```

Call restrictions:

- only from address listed in `isAdmin` mapping or owner address

Parameters requirements:

- all input parameters must meet set of requirements, critical ones are
  - data for order matching must be signed by corresponding users
  - all users must have enough `balance` of tokens or ETH
  - maker’s price must not be worse than the taker's order
  - orders must not be canceled

Logic:

- function exchanges tokens and ETH with pre-matched taker and makers orders

```
function cancelByAdmin(uint256[] inputs) external onlyAdmin
```

Call restrictions:

- call must be only from address listed in `isAdmin` mapping or owner address

Parameters requirements:

- all input parameters must meet set of requirements, critical one is
  - data for order matching must be signed by corresponding users

Logic:

- function makes exchange or withdraw order canceled

```
function matchTokenOrderByAdmin_k44j (uint256[] inputs) external
onlyAdmin
```

Call restrictions:

- call must be only from address listed in `isAdmin` mapping or owner address

Parameters requirements:

- all input parameters must meet set of requirements, critical ones are
  - data for order matching must be signed by corresponding users
  - all users must have enough `balance` of tokens
  - maker's price must not be worse than the taker's order
  - orders must not be canceled

Logic:

- function exchanges tokens with pre-matched taker's and maker's orders

```
function migrateByAdmin_DQV(uint256[] inputs) external onlyAdmin
```

Call restrictions:

- call must be only from address listed in `isAdmin` mapping or owner address

Parameters requirements:

- all input parameters must meet set of requirements, critical ones are
  - data for migrate must be signed by corresponding users
  - all users must have enough `balance` of tokens or ETH for migration request

Logic:

- function migrate balances to new contract

```
function collectFee(address token) external onlyOwner
```

Call restrictions:

- call must be only from owner

Parameters requirements:

- no requirements

Logic:

- function allows owner to withdraw tokens and ETH stored on contract as fee

```
function changeLockPeriod(uint256 periodInDays) external onlyOwner
```

Call restrictions:

- call must be only from owner

Parameters requirements:

- `periodInDays` should be between 1 and 30

Logic:

- function allows owner to change `LockPeriod` from 1 day to 30 days

```
function transferForAdmin(address token, address account, uint256
amount) onlyAdmin external
```

Call restrictions:

- call must be only from address listed in `isAdmin` mapping or owner address

Parameters requirements:

- `token` address must be listed in `tokenAddress2Id` mapping

Logic:

- function transfers `amount` of `token` from admin to `account` address if `balances` allows it

Automated analysis

We used several publicly available automated Solidity analysis tools. Here are the combined results of SmartCheck, Solhint, and Remix. Oyente has found no issues.
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	Vulnerability	False positives	True positives
Remix	Defines a return type but never explicitly returns a value	1	
	Gas requirement of function is high	17	
	Potential Violation of Checks-Effects-Interaction pattern	1	1
	Variables have very similar names	7	
Total Remix		26	1
SmartCheck	Dos With Revert	8	
	Gas Limit And Loops		1
	No Payable Fallback	5	
	Pragmas Version	4	2
	Reentrancy External Call	9	1

	Unchecked Math	8	
	Visibility		5
Total SmartCheck		34	9
Solhint	Compiler version must be fixed	5	1
	Explicitly mark visibility of state	5	
Total Solhint		10	1
Overall Total		**70**	**11**

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 the current implementation. We highly recommend addressing them.

Mixing entities in mapping

Both `user` and `token` entities are stored in the same `address2Id` mapping (`addUser` and `registerToken` functions). This allows an attacker to add malicious token contracts address to `address2Id` mapping via `depositEther` and thus bypass the token check-in `depositToken` function (Joyso.sol, line 56). Furthermore, more complex social engineering attacks are possible. We highly recommend implementing two separate mappings for `user` and `token` entities.

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

Overpowered owner

The contract owner can change tokens' IDs to address mapping using `registerToken` function with already used `index` parameter. We highly recommend banning this possibility since it might be undesirable for contract users.

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.

Possible out of gas

During tests, some functions consume a lot of gas (>4800000, which would even exceed the block gas limit until recent changes). `matchByAdmin` function iterates over the array passed as a parameter, which may consume much gas, too. We highly recommend checking gas amount in tests. Besides, we recommend avoiding loops with a big or unknown number of steps.

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

Potential violation of Checks-Effects-Interaction pattern

There is a Checks-Effects-Interaction violation in Joyso.sol, lines 58-59:

```
require(Token(token).transferFrom(msg.sender, this, amount));
balances[token][msg.sender] =
balances[token][msg.sender].add(amount);
```

In this case the CEI violation does not lead to an actual vulnerability. However, we highly recommend following best practices including Checks-Effects-Interactions pattern since it helps to avoid many serious vulnerabilities.

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

Using OpenZeppelin files in repo

OpenZeppelin files are added to the repo instead of being connected via npm. We highly recommend using npm in order to guarantee that original OpenZeppelin contracts are used with no modifications.

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

Redundant code

There are both `require` check of overflow and safeMath functions in Joyso.sol, lines 71, 166, 169-170. We recommend removing `require` checks since they are redundant.

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

Pragmas version

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

```
pragma solidity ^0.4.19; // bad: compiles w 0.4.19 and above
pragma solidity 0.4.19; // good: compiles w 0.4.19 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 – 0.4.19 at the moment (0.4.13 is used in the contracts).

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

Implicit visibility level

There are variables and functions with implicit visibility level in the code (Joysol.sol, lines 11, 12, 13, 14; JoysoDataDecoder.sol, line 5).
We recommend specifying visibility levels (`public`, `private`, `external`, `internal`) explicitly and correctly in order to improve code readability.

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

Mismatched event parameter

In `collectFee` function `Withdraw` event is emitted incorrectly:

```
Withdraw(
    token,
    msg.sender,
    amount,
    balances[token][joysoWallet]
);
```

The event indicates `msg.sender` address as `user` address. However, `balance` argument of this event is `joysoWallet`'s remaining balance. It can lead to incorrect interpretation of this event. We recommend using another event in order to keep admin's address in event info in `collectFee` function.

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

Misspelling

There is a typo in the code in Joyso.sol, line 151:

```
require(periodInDays * 1 days < 30 * 1 days && periodInDays >=
1 * 1 days);
```

In the second check `periodInDays` is not multiplied by `1 days`. Thus, this check will always fail and `changeLockPeriod` function will never be executed.
We recommend fixing this typo.

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

Unchecked math

Solidity is prone to an integer over- and underflow. Overflow leads to unexpected effects and can lead to loss of funds if exploited by malicious account. The values in the following cases are not checked:

- Joyso.sol, line 708:

```
txFee = txFee * (10 ** 12) / joyPrice;
```

In the current implementation it is almost impossible to overflow there. However, if the code will be reused it can cause problems. We recommend including additional checks.

Added tokens audit

Since Joyso contract interacts with third-party token contracts, we recommend auditing these contracts before adding tokens to the platform.

This analysis was performed by SmartDec.

Pavel Yushchenko, Chief Technical Officer
Yaroslav Alexandrov, Head of Development Department
Ivan Ivanitskiy, Chief Analytics Officer
Elizaveta Kharlamova, Analyst
Alexander Seleznev, Chief Business Development Officer
Igor Sobolev, Analyst

Sergey Pavlin, Chief Operating Officer 

May 22, 2018

Appendix

Compilation output

```
Compiling ./contracts/Joyso.sol...
Compiling ./contracts/JoysoDataDecoder.sol...
Compiling ./contracts/Migratable.sol...
Compiling ./contracts/Migrations.sol...
Compiling ./contracts/libs/SafeMath.sol...
Compiling ./contracts/testing/JoysoMock.sol...
Compiling ./contracts/testing/NewJoyso.sol...
Compiling ./contracts/testing/TestToken.sol...
Compiling ./node_modules/zeppelin-
solidity/contracts/math/SafeMath.sol...
Compiling ./node_modules/zeppelin-
solidity/contracts/ownership/Ownable.sol...
Compiling ./node_modules/zeppelin-
solidity/contracts/token/ERC20/BasicToken.sol...
Compiling ./node_modules/zeppelin-
solidity/contracts/token/ERC20/ERC20.sol...
Compiling ./node_modules/zeppelin-
solidity/contracts/token/ERC20/ERC20Basic.sol...
Compiling ./node_modules/zeppelin-
solidity/contracts/token/ERC20/StandardToken.sol...
```

Code coverage

```
----------------------|----------|----------|----------|----------|----------------|
File                  |  % Stmts | % Branch |  % Funcs |  % Lines |Uncovered Lines |
----------------------|----------|----------|----------|----------|----------------|
 contracts/           |    94.35 |    88.14 |     91.3 |    93.97 |                |
  Joyso.sol           |    93.84 |    88.14 |     87.1 |    93.43 |... 525,561,708 |
  JoysoDataDecoder.sol |      100 |      100 |      100 |      100 |                |
  Migratable.sol      |      100 |      100 |      100 |      100 |                |
 contracts/libs/      |       90 |     62.5 |      100 |       90 |                |
  SafeMath.sol        |       90 |     62.5 |      100 |       90 |             11 |
 contracts/testing/   |      100 |    66.67 |      100 |      100 |                |
  JoysoMock.sol       |      100 |      100 |      100 |      100 |                |
  NewJoyso.sol        |      100 |    66.67 |      100 |      100 |                |
  TestToken.sol       |      100 |      100 |      100 |      100 |                |
----------------------|----------|----------|----------|----------|----------------|
All files             |    94.49 |    85.61 |    92.73 |    94.14 |                |
----------------------|----------|----------|----------|----------|----------------|
```

Tests output

```
TestDecoder
    ✓ testDecodeWithdraw (112ms)
    ✓ testDecodeOrderData (56ms)

  Contract: cancel.js
    ✓ cancelByAdmin should update the user nonce (1001ms)
    ✓ nonce should more than current nonce (1132ms)
    ✓ pay joy for fee to cancel the order (956ms)
    ✓ cancel should fail if user's signature is wrong (1010ms)
    ✓ cancel should fail if user's balance is not enough
(998ms)
    ✓ match should fail if the taker order's nonce is less
than userNonce (1261ms)
    ✓ match should fail if the maker order's nonce is less
than userNonce (1184ms)
    ✓ tokenMatch should fail if the taker order's nonce is
less than userNonce (1147ms)
    ✓ match should fail if the maker order's nonce is less
than userNonce (1164ms)

  Contract: debug.js
    ✓ case1, details in google doc (1161ms)

  Contract: match.js
    ✓ case1, details in google doc (1076ms)
    ✓ case2, details in google doc (1292ms)
    ✓ case3, details in google doc (1114ms)
    ✓ case4 (1046ms)
    ✓ case5 (929ms)
    ✓ case6 trade all the user balance (1024ms)
    ✓ taker paid Joy for fee (1135ms)
    ✓ gasFee can only charge once for each order (1327ms)
    ✓ gasFee (JOY) can only charge once for each order
(1385ms)
    ✓ it should fail if taker's signature is wrong. (996ms)
    ✓ it should fail if the maker's signature is wrong
(1011ms)
    ✓ it should fail if the price taker's price is worse than
maker's (951ms)
```

```
    ✓ split a taker order into two transactions (1294ms)

  Contract: test migrate.js
    ✓ test new version contract (151ms)
    ✓ combination of new and old contract (816ms)
    ✓ token migrate (869ms)
    ✓ token migrate, pay by free (979ms)
    ✓ token migrate, pay by ether (1003ms)
    ✓ token migrate, pay by joy (1048ms)
    ✓ token migrate, pay by token (940ms)
1 user migrate cost:  81611
2 users migrate cost:  109018
    ✓ gas consumption  (1319ms)

  Contract: Joyso misc.js
    ✓ it should fail if not admin send the match (940ms)
    ✓ deposit should fail, if the deposit token is not
registered  (342ms)
    ✓ it should fail if the token is not approved to the joyso
contract (273ms)
    ✓ registerToken's index should more than 1 (212ms)
    ✓ the same token can not registered twice  (210ms)
    ✓ add new admin (1072ms)
    ✓ for case1, maker and taker order exchage the place
should still success (1010ms)
    ✓ register token can not use other token's index (237ms)

  Contract: Joyso mock
    ✓ withdraw ether directly by user (1042ms)
    ✓ withdraw token directly by user (976ms)
    ✓ unlockMe should reset the user lock (896ms)
    ✓ withdraw ether should fail if no balance (775ms)
    ✓ withdraw token should fail if no balance (877ms)
    ✓ withdraw directly should fail (829ms)

  Contract: gas analysis
2 order match: 151083
3 order match: 193017
4 order match: 250011
5 order match: 307591
6 order match: 364545
7 order match: 421985
```

```
8 order match: 479584
9 order match: 537248
10 order match: 595267
withdraw by admin (ether): 74720
withdraw by admin (token): 103749
    ✓ case 1  (9095ms)

  Contract: tokenMatch.js
2 order match: 151863
    ✓ try token base match (1074ms)
    ✓ token by token match (1103ms)
    ✓ try token base match, taker is a sell order (1094ms)
    ✓ it should fail if taker's signature is wrong. (1053ms)
    ✓ it should fail if the maker's signature is wrong (993ms)
    ✓ a filled taker order should not be trade again (993ms)
    ✓ a filled maker order should not be trade again (1159ms)
    ✓ it should fail if the price taker's price is worse than
maker's (941ms)

  Contract: joyso withdraw
    ✓ withdraw token, pay by ether (1037ms)
    ✓ withdraw joy, pay by ether (876ms)
    ✓ withdraw ether, pay by ether (974ms)
    ✓ withdraw token, pay by JOY (984ms)
    ✓ withdraw joy, pay by JOY (973ms)
    ✓ withdraw ether, pay by JOY (1033ms)
    ✓ withdraw token, pay by token (967ms)
    ✓ it should fail if use the same withdraw hash (871ms)
    ✓ it should fail if the signature is wrong (964ms)
    ✓ withdraw token, pay by ether. Should fail if no token
balance. (818ms)
    ✓ withdraw token, pay by ether. Should fail if no ether
balance. (943ms)

  67 passing (1m)
```

Solhint output

```
contracts/Joyso.sol
   12:1   error    Definition must be surrounded with two
blank line indent           two-lines-top-level-separator
   92:5   warning  Event and function names must be
different                          no-simple-event-func-name
  110:2   error    Line length must be no more than 120 but
current length is 123     max-line-length
  189:5   error    Function name must be in
mixedCase                                    func-name-mixedcase
  253:5   error    Function name must be in
mixedCase                                    func-name-mixedcase
  253:70  error    Function body contains 60 lines but allowed
no more than 50 lines  function-max-lines
  347:79  error    Function body contains 58 lines but allowed
no more than 50 lines  function-max-lines
  347:5   error    Function name must be in
mixedCase                                    func-name-mixedcase
  471:5   error    Function name must be in
mixedCase                                    func-name-mixedcase
  521:79  error    Visibility modifier must be first in list
of modifiers             visibility-modifier-order
  549:2   error    Line length must be no more than 120 but
current length is 122     max-line-length
  561:16  warning  Avoid to make time-based decisions in your
business logic          not-rely-on-time
  583:2   error    Line length must be no more than 120 but
current length is 145     max-line-length
  585:2   error    Line length must be no more than 120 but
current length is 145     max-line-length

contracts/JoysoDataDecoder.sol
   6:1  error  Definition must be surrounded with two blank
line indent        two-lines-top-level-separator
  33:2  error  Line length must be no more than 120 but
current length is 126  max-line-length
  51:2  error  Line length must be no more than 120 but
current length is 122  max-line-length
  67:2  error  Line length must be no more than 120 but
current length is 138  max-line-length

✖ 18 problems (16 errors, 2 warnings)
```

Solium output

```
contracts/Joyso.sol
  561:15    warning    Avoid using 'now' (alias to
'block.timestamp').    security/no-block-members

✖ 1 warning found.
```