SmartDec

Ethex Smart Contracts Security Analysis

This report is public.

Published: November 1, 2019.

Abstract

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

The audit showed neither critical issues nor low severity issues. However, one medium severity issue was found. It does not endanger project security.

General recommendations

The contracts code is of medium code quality. The audit did not reveal any issues that endanger project security.

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 types of bugs.

Compliance with the documentation

The audit showed no discrepancies between the code and the provided documentation.

Tests

The audit showed that the code was covered with tests sufficiently.

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 categorize all the found security issues in accordance with the classification in order to identify developers' shortcomings
  - we check smart contracts logic and compare it with the one described in the documentation
  - 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 Ethex 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 Ethex specification (”README.md” in the project repo) and smart contracts’ code (version on commit 65c8bd1761d87d508648f6d06c5195761754ac42).

Project architecture

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

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

The total LOC of audited Solidity sources is 670.

Automated analysis

We used several publicly available automated Solidity analysis tools. Here are the combined results of SmartCheck, Solhint, and Remix scanning. 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	True positives	False positives
SmartCheck	Overpowered role	7	3
	Costly loop		3
	Non-initialized return value		1
	Extra gas consumption		1
	Replace multiple return values with a struct		1
Total SmartCheck		7	9
Solhint	Variable name must be in mixedCase		2
	Avoid multiple calls of ”send” method in single transaction		14
	Avoid to use tx.origin		1
Total Solhint		0	17
Total Overall		**7**	**26**

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 the 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.

Overpowered role

Project contains upgradability logic:

- **EthexJackpot.sol**, lines 109, 113
- **EthexLoto.sol**, lines 224, 228
- **EthexSuperprize.sol**, lines 43, 97, 101

According to this logic, owner of contracts can change **EthexJackpot** and **EthexSuperprize** contracts' addresses. Thus, if they replace these contracts with invalid ones, the user will not receive these prizes (Jackpot and Superprize).

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.

This analysis was performed by SmartDec.

Boris Nikashin, Project Manager
Pavel Kondratenkov, Analyst
Alexander Drygin, Analyst

November 1, 2019

Appendix

Code coverage

```
--------------------------|----------|----------|----------|----------|----------------|
File                      |  % Stmts | % Branch |  % Funcs |  % Lines |Uncovered Lines |
--------------------------|----------|----------|----------|----------|----------------|
contracts/                |    85.47 |    68.42 |    72.73 |    86.29 |                |
DeliverFunds.sol          |        0 |      100 |        0 |        0 |              5 |
EthexHouse.sol            |        0 |      100 |       50 |        0 |             16 |
EthexJackpot.sol          |    81.05 |    53.85 |    66.67 |    81.25 |... 175,176,216 |
EthexLoto.sol             |    88.52 |       80 |    69.23 |    89.66 |... 225,229,233 |
EthexSuperprize.sol       |    86.96 |       50 |      100 |    86.96 |... 67,77,91,92 |
EthexTestOldJackpot.sol   |      100 |      100 |      100 |      100 |                |
IERC20.sol                |      100 |      100 |      100 |      100 |                |
Ownable.sol               |       50 |       25 |    66.67 |    57.14 |       19,20,21 |
--------------------------|----------|----------|----------|----------|----------------|
All files                 |    85.47 |    68.42 |    72.73 |    86.29 |                |
--------------------------|----------|----------|----------|----------|----------------|
```

Compilation output

```
Compiling your contracts...
===========================
> Compiling ./contracts/DeliverFunds.sol
> Compiling ./contracts/EthexHouse.sol
> Compiling ./contracts/EthexJackpot.sol
> Compiling ./contracts/EthexLoto.sol
> Compiling ./contracts/EthexSuperprize.sol
> Compiling ./contracts/EthexTestOldJackpot.sol
> Compiling ./contracts/IERC20.sol
> Compiling ./contracts/Ownable.sol
```

Tests output

```
  Contract: Ethex Jackpot
 jackpot contract rises (2865ms)
 registerTicket doesn't fail (136ms)
 settleJackpot doesn't fail (57ms)
 jackpot works (48327ms)
 redrawing jackpot doesn't work (65ms)

Contract: Ethex Loto
 placebet doesn't fail (146ms)
 settlebets doesn't fail (without bets) (39ms)
 settlebets doesn't fail (with bets) (177ms)
```

```
 bets with empty params is prohibited
 bets with no id is prohibited
 bets with empty combination is prohibited (55ms)
 bet with totally incorrect combination fails (40ms)
 bet with partly incorrect combination fails (no significant
fields)
 bets with partly incorrect combination works (one significa
nt field) (3390ms)
 25 draws for random bets with 1 symbol are correct (3505ms)
 25 draws for random bets with 2 symbol are correct (4201ms)
 25 draws for random bets with 3 symbol are correct (4234ms)
 25 draws for random bets with 4 symbol are correct (4480ms)
 25 draws for random bets with 5 symbol are correct (4275ms)
 25 draws for random bets with 6 symbol are correct (4666ms)
 25 draws for all even bets are correct (11515ms)
 25 draws for all odd bets are correct (14618ms)
 drawing 25 all numbers bets (5688ms)
 drawing 25 all letters bets (3859ms)
 Draw generates refund if bet is too old (3178ms)
 bets with amount less then minimal is prohibited (42ms)
 placebet should not create jackpot ticket for bets with one
cell (61ms)
 placebet should create jackpot ticket for bets with more th
en one cell (106ms)
 settleBets should process 10 bets (2144ms)
 placeBet should fail if there is not enough eth on contract
(1072ms)
 placeBet should fail if bet amount is greater then max win
amount (38ms)

Contract: Ethex Superprize
 superprize contract rises (2779ms)
 superprize migration works (108ms)
 superprize: initSuperprize (109ms)
 superprize: paySuperprize (no action) (1280ms)


35 passing (3m)
```