
# SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

**Customer**: Boobanker.org
**Prepared on**: 11/11/2020
**Platform**: Ethereum
**Language**: Solidity

audit@etherauthority.io

# Table of contents

THIS DOCUMENT MAY CONTAIN CONFIDENTIAL INFORMATION ABOUT IT SYSTEMS AND INTELLECTUAL PROPERTY OF THE CUSTOMER AS WELL AS INFORMATION ABOUT POTENTIAL VULNERABILITIES AND METHODS OF THEIR EXPLOITATION.
THE REPORT CONTAINING CONFIDENTIAL INFORMATION CAN BE USED INTERNALLY BY THE CUSTOMER OR IT CAN BE DISCLOSED PUBLICLY AFTER ALL VULNERABILITIES ARE FIXED - UPON DECISION OF CUSTOMER.

## Document

| | |
|---|---|
| **Name** | Smart Contract Code Review and Security Analysis Report for BooBank |
| **Platform** | Ethereum / Solidity |
| **MD5 hash** | 2ac4cec836790b21787ad510865d5823 |
| **File name** | booBank.sol |
| **SHA512 hash** | f2418c62d5bcf65629482d48dc55f979fe108047888674eaa8a61bbc3f1fc08f3cfed4493ab393a8f210db3d2614f078aa0d660aa9dfe677b02304d819eced1f |
| **Date** | 11/11/2020 |

# Introduction

EtherAuthority.io (Consultant) was contracted by BooBanker Team (Customer) to conduct a Smart Contracts Code Review and Security Analysis. This report presents the findings of the security assessment of Customer`s smart contracts and its code review conducted between Oct, 31st 2020 – Nov 11th, 2020.

# Project Scope

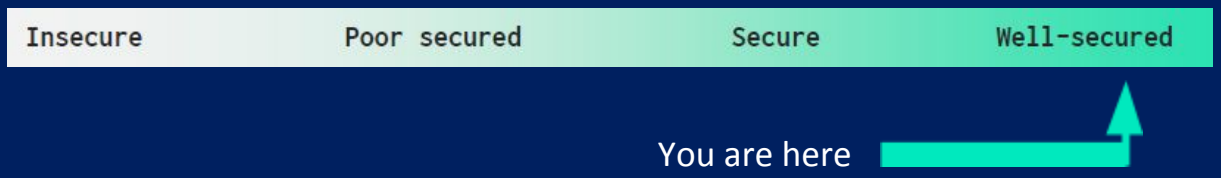
The scope of the project is **BooBank** smart contract.

We have scanned this smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that are 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 Block Gas Limit
- Transaction-Ordering Dependence
- Byte array vulnerabilities
- Style guide violation
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Unchecked external call - Unchecked math
- Unsafe type inference
- Implicit visibility level

## Executive Summary

According to the assessment, Customer`s solidity smart contract is well secured.




Our team performed analysis of code functionality, manual audit and automated checks with smartDec, Mythril, Slither and remix IDE. All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the Audit overview section. General overview is presented in AS-IS section and all found issues can be found in the Audit overview section.

We found 0 high, 0 medium and 0 low and some very low level issues.

## Code Quality

BooBanker protocol consists of modular smart contracts, which interact with each other . This modularization increases the efficiency of the code execution flow. This type of structure of smart contracts and their various libraries, create complexity while code deployment. However on another hand, it also increases ease of code writing in the development process.

The libraries in the BooBanker protocol are part of its logical algorithm. A library is a different type of smart contract that contains reusable code. Once deployed on the blockchain (only once), it is assigned a specific address and its properties / methods can be reused many times by other contracts in the BooBanker protocol.

BooBank team has **not** provided scenario and unit test scripts, which would help to determine the integrity of the code in an automated way.

Overall, the code is well commented (except some function blocks). Commenting can provide rich documentation for functions, return variables and more. Use of Ethereum Natural Language Specification Format (NatSpec) for commenting is recommended.

## Documentation

As mentioned above, our team found the code which was well commented except for a few functions. It's recommended to write comments in the smart contract code, so anyone can quickly understand the programming flow as well as complex code logic.

We were given BooBanker tokenomics online version at:
https://boobank.medium.com/boobank-tokenomics-129e772b2006
which was very helpful in understanding the overall architecture of the protocol. It also provided a clear overview of the system components, including helpful details, like the lifetime of the background script.

## Use of Dependencies

As per our observation, the libraries are used in this smart contract infrastructure. Those were based on well known industry standard open source projects. And even core code blocks are also modeled from the running source of production with little changes as mentioned in the comment section also which reflects its production ready nature. In Spite of that we conducted many test cases to find its vulnerabilities and found nothing serious.

# AS-IS overview

## BooBanker contract overview

**BooBanker** is a collection of smart contract functions that collectively provides decentralized financial activities to the users. Its file format is described below:

**booBank.sol**

**Import:** null

**Contract:** context

**Inherit:** null

**About:** This contract provides context of execution in the form of msg.sender and msg.value.

**Observation:** Since this contract was not used in GSN. So no need for this, and this can be safely removed. On another hand, the presence and use of this also does not create any vulnerability.

**Test Report:** All passed including security check.

**Score:** Passed

**Conclusion:** Remove it from all four inheritance and import lines

| Sl. | Function | Type | Observation | Test Report | Conclusion | Score |
|---|---|---|---|---|---|---|
| 1 | _msgsender | read | Passed | All Passed | remove it | Passed |
| 2 | _msgData | read | Passed | All Passed | remove it | Passed |

**Contract:** ERC20

**Inherit:** IERC20

**About:** ERC20 standard function signatures.

**Observation:** No issues identified

**Test Report:** All passed including security check.

**Score:** Passed

| Sl. | Function | Type | Observation | Test Report | Conclusion | Score |
|---|---|---|---|---|---|---|
| 1 | name | read | Passed | All Passed | No Issue | Passed |
| 2 | symbol | read | token view | All Passed | No Issue | Passed |
| 3 | decimals | read | token view | All Passed | No Issue | Passed |
| 4 | totalsupply | read | token view | All Passed | No Issue | Passed |
| 5 | balanceof | read | token view | All Passed | No Issue | Passed |
| 6 | transfer | write | Passed | All Passed | No Issue | Passed |
| 7 | allowance | read | token view | All Passed | No Issue | Passed |
| 8 | approve | write | Passed | All Passed | No Issue | Passed |
| 9 | transferFrom | write | Passed | All Passed | No Issue | Passed |
| 10 | IncreaseAllowance | write | Passed | All Passed | No Issue | Passed |
| 11 | DecreaseAllowance | write | Passed | All Passed | No Issue | Passed |
| 12 | _transfer | internal | Passed | All Passed | No Issue | Passed |
| 13 | _mint | internal | Passed | All Passed | No Issue | Passed |
| 14 | _burn | internal | Passed | All Passed | No Issue | Passed |
| 15 | _approve | Inernal | Passed | All Passed | No Issue | Passed |
| 16 | _SetupDecimals | Internal | unused | All Passed | remove it | Passed |
| 17 | _beforeTokenTransfer | Internal | Passed | All Passed | No Issue | Passed |

**Contract:** ownable

**Inherit:** context

**About:** Collectively global constant defined here

**Observation:** No issue found.

**Test Report:** All passed including security check.

**Score:** Passed

**Conclusion:** Production ready

| Sl. | Function | Type | Observation | Test Report | Conclusion | Score |
|---|---|---|---|---|---|---|
| 1 | Owner | read | Passed | All Passed | No Issue | Passed |
| 2 | renounceownership | write | Passed | All Passed | No Issue | Passed |
| 3 | transferownership | read | Passed | All Passed | No Issue | Passed |

**Contract:** Ectoplasma

**Inherit:** ERC20

**About:** Basic pool helper file

**Observation:** No issue found.

**Test Report:** All passed including security check.

**Score:** Passed

**Conclusion:** Production ready

| Sl. | Function | Type | Observation | Test Report | Conclusion | Score |
|---|---|---|---|---|---|---|
| 1 | mint | write | Passed | All Passed | No Issue | Passed |
| 2 | transfer | write | Passed | All Passed | No Issue | Passed |
| 3 | transferFrom | write | Passed | All Passed | No Issue | Passed |
| 4 | SetDevAddr | write | Passed | All Passed | No Issue | Passed |
| 5 | SetBurnRate | write | Passed | All Passed | No Issue | Passed |

**Contract:** BooBank

**Inherit:** ERC20, Ownable

**About:** Dedicated method to calculate numerical figures as per plan.

**Observation:** No issue found.

**Test Report:** All passed including security check.

**Score:** Passed

**Conclusion:** Production ready

| Sl. | Function | Type | Observation | Test Report | Conclusion | Score |
|---|---|---|---|---|---|---|
| 1 | transfer | write | Passed | All Passed | No Issue | Passed |
| 2 | transferFrom | write | Passed | All Passed | No Issue | Passed |
| 3 | SetBurnRate | write | Passed | All Passed | No Issue | Passed |
| 4 | mint | write | Passed | All Passed | No Issue | Passed |
| 5 | SetDevAddr | write | Passed | All Passed | No Issue | Passed |
| 6 | SetPauseBlock | write | Passed | All Passed | No Issue | Passed |
| 7 | delegates | read | Passed | All Passed | No Issue | Passed |
| 8 | delegate | write | Passed | All Passed | No Issue | Passed |
| 9 | delegateBysig | write | Passed | All Passed | No Issue | Passed |
| 10 | getcurrentVotes | read | Passed | All Passed | No Issue | Passed |
| 11 | getpriorVotes | read | Passed | All Passed | No Issue | Passed |
| 12 | _delegate | read | Passed | All Passed | No Issue | Passed |
| 13 | _moveDelegates | write | Passed | All Passed | No Issue | Passed |
| 14 | _writecheckpoint | write | Passed | All Passed | No Issue | Passed |
| 15 | Safe32 | read | Passed | All Passed | No Issue | Passed |
| 16 | getChainId | read | Passed | All Passed | No Issue | Passed |

## Severity Definitions

| Risk Level | Description |
|---|---|
| **Critical** | Critical vulnerabilities are usually straightforward to exploit and can lead to tokens lose etc. |
| **High** | High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial functions |
| **Medium** | Medium-level vulnerabilities are important to fix; however, they can't lead to tokens lose |
| **Low** | Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have significant impact on execution |
| **Lowest / Code Style / Best Practice** | Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can be ignored. |

## Audit Findings

### Critical

No critical severity vulnerabilities were found.

### High

No high severity vulnerabilities were found.

### Medium

No medium severity vulnerabilities were found.

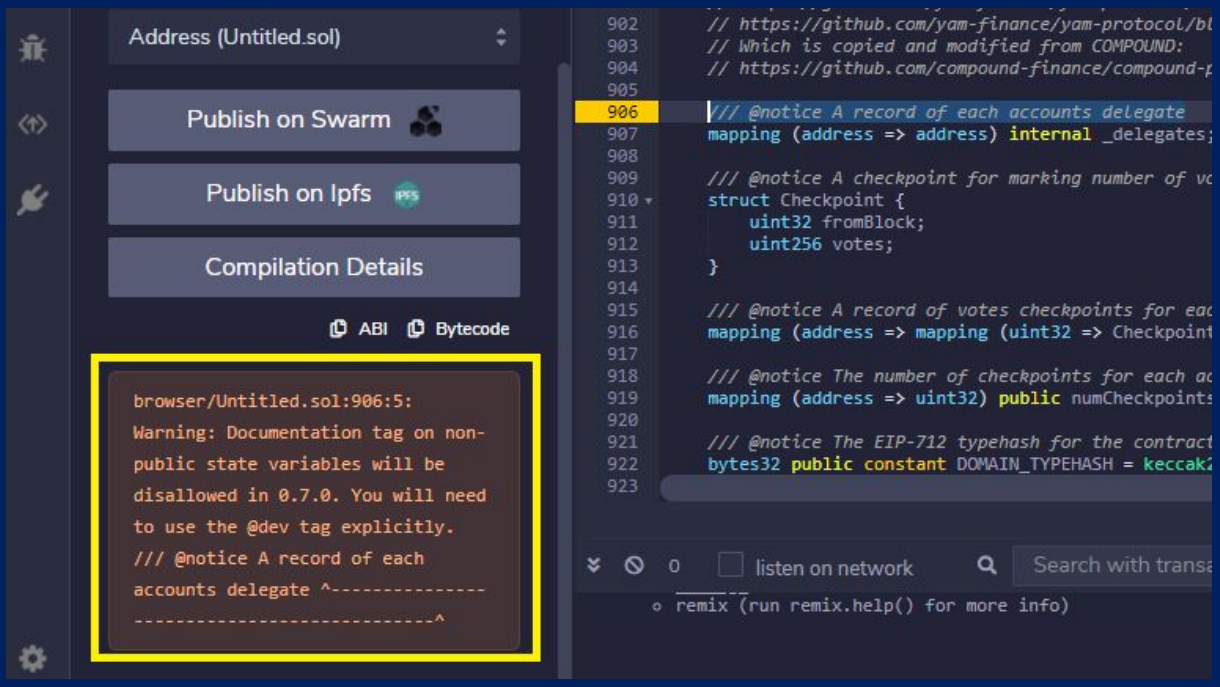
### Low

No low severity vulnerabilities were found.

### Very Low

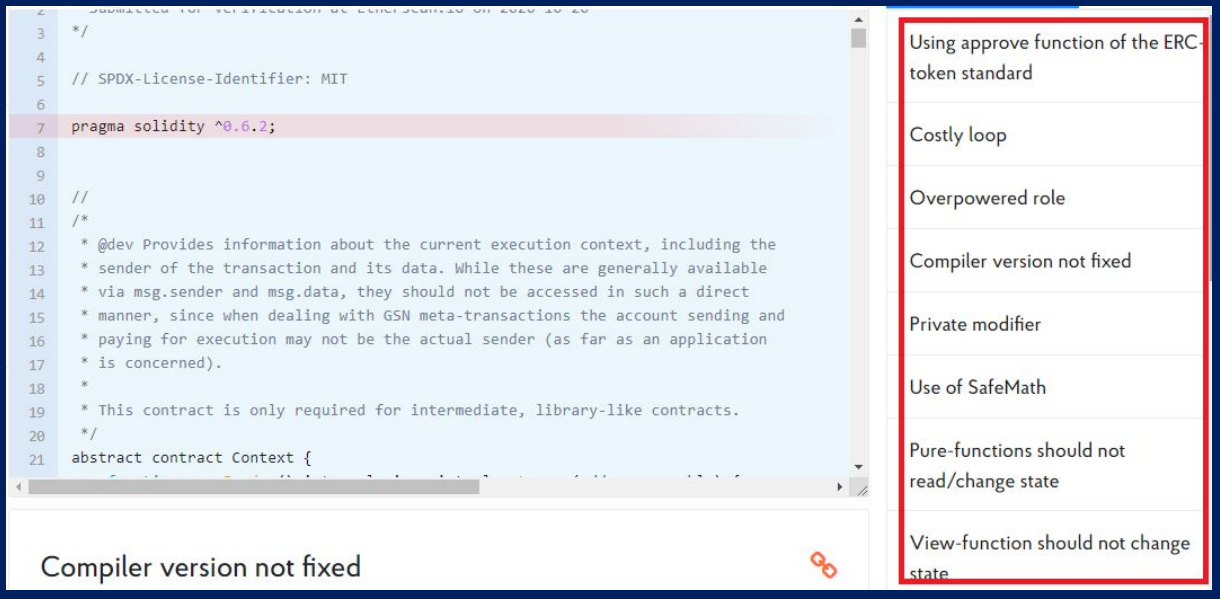
(1) Apart from some unUsed functions no such issues were found.

(2) Remix displays this warning (yellow box) is not need any attention because is is for future version alert




(3) Line 1053 contains a costly loop if limited by plan then its OK

(4) Some reports indicated by automated static tools are below, none of them is critical hence can be ignored in terms of production.




(5) There is some deep nesting of msg.sender. So it is better to just use msg.sender to save some gas consumption.

(6) By controlling byte packaging, gas consumption can be improved up to a good noticeable margin.

## Conclusion

We were given contract files. And we have used all possible tests based on given objects as files. The contracts are written so systematic and well commented (except few functions), that we found almost no critical things. So it is good to go for production.

Since possible test cases can be unlimited for such extensive smart contract protocol, so we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan everything.

Smart contracts within the scope, were manually reviewed and analyzed with static analysis tools. Smart Contract's high level description of functionality was presented in As-is overview section of the report.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

Security state of reviewed contracts is "Well Secured".

# Disclaimers

## EtherAuthority.io Disclaimer

The smart contracts given for audit have been analyzed in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment and functionality (performing the intended functions).

Because the total number of test cases are unlimited, so the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only - we recommend proceeding with several independent audits and a public bug bounty program to ensure security of smart contracts.

## Technical Disclaimer

Smart contracts are deployed and executed on blockchain platform. The platform, its programming language, and other software related to the smart contract can have their own vulnerabilities that can lead to hacks. Thus, the audit can’t guarantee explicit security of the audited smart contracts.


Ether Authority