SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: ASI FI LTD
Date: May 14th, 2021

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

This document may contain confidential information about IT systems and the 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 fixed - upon a decision of the Customer.

Document

Name	Smart Contract Code Review and Security Analysis Report for ASI FI LTD.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	Token
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/asifinance/ASI-contracts
Commit	
Deployed contract	
Timeline	10 MAY 2021 - 14 MAY 2021
Changelog	14 MAY 2021 - INITIAL AUDIT

This document is proprietary and confidential. No part of this document may be disclosed in any manner to a third party without the prior written consent of Hacken.

Table of contents

This document is proprietary and confidential. No part of this document may be disclosed in any manner to a third party without the prior written consent of Hacken.

Introduction

Hacken OÜ (Consultant) was contracted by ASI FI LTD (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of Customer's smart contract and its code review conducted between May 10th, 2021 - May 14th, 2021.

Scope

The scope of the project is smart contracts in the repository:

```
Repository: https://github.com/asifinance/ASI-contracts
File:
      BBOTSToken.sol
      ASI.sol
```

We have scanned this smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that are considered:

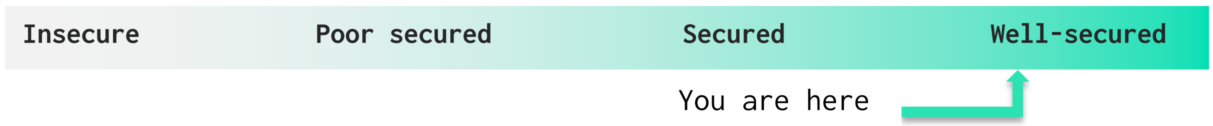
Category	Check Item
Code review	▪ Reentrancy ▪ Ownership Takeover ▪ Timestamp Dependence ▪ Gas Limit and Loops ▪ DoS with (Unexpected) Throw ▪ DoS with Block Gas Limit ▪ Transaction-Ordering Dependence ▪ Style guide violation ▪ Costly Loop ▪ ERC20 API violation ▪ Unchecked external call ▪ Unchecked math ▪ Unsafe type inference ▪ Implicit visibility level ▪ Deployment Consistency ▪ Repository Consistency ▪ Data Consistency

This document is proprietary and confidential. No part of this document may be disclosed in any manner to a third party without the prior written consent of Hacken.

Functional review	▪ Business Logics Review ▪ Functionality Checks ▪ Access Control & Authorization ▪ Escrow manipulation ▪ Token Supply manipulation ▪ Assets integrity ▪ User Balances manipulation ▪ Kill-Switch Mechanism ▪ Operation Trails & Event Generation

Executive Summary

According to the assessment, the Customer's smart contracts are well-secured.



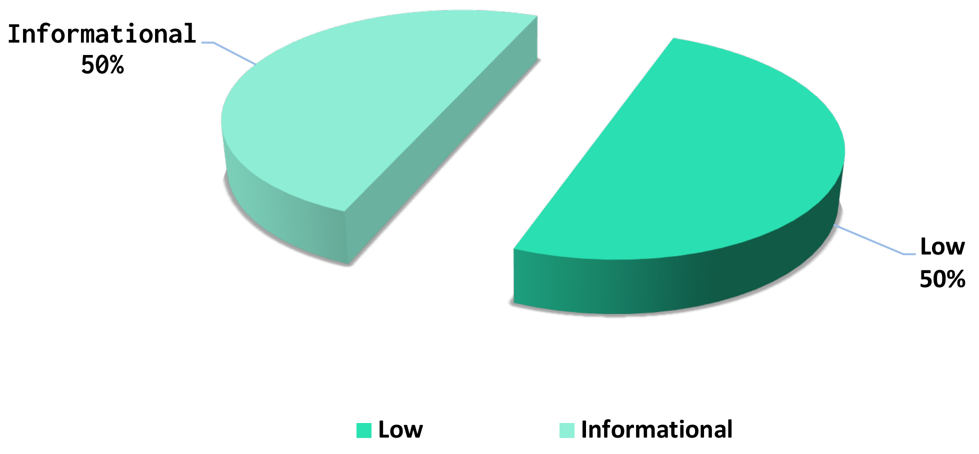
Our team performed an analysis of code functionality, manual audit, and automated checks with Mythril and Slither. All issues found during automated analysis were manually reviewed, and important vulnerabilities are presented in the Audit overview section. A general overview is presented in AS-IS section,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

Security engineers found **2** low, **2** informational issue during the audit.

Notice: the audit scope is limited and not include all files in the repository. Though, reviewed contracts are secure, we may not guarantee secureness of contracts that are not in the scope.

This document is proprietary and confidential. No part of this document may be disclosed in any manner to a third party without the prior written consent of Hacken.

Graph 1. The distribution of vulnerabilities after the first review.



This document is proprietary and confidential. No part of this document may be disclosed in any manner to a third party without the prior written consent of Hacken.

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to assets loss or data manipulations.
High	High-level vulnerabilities are difficult to exploit; however, they also have a 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 assets loss or data manipulations.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that can't have a 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.

This document is proprietary and confidential. No part of this document may be disclosed in any manner to a third party without the prior written consent of Hacken.

AS-IS overview

ASI.sol

Description

ASI is a ERC20 token.

Imports

ASI has following imports:

- SafeMath.sol - from the OpenZeppelin.
- Context.sol
- IERC20.sol
- Address.sol
- Ownable.sol

Inheritance

ASI inherit following contracts:

- Context
- IERC20
- Ownable

Usages

ASI contract has following usages:

- SafeMath for uint
- Address for address

Structs

ASI contract has no data structures.

Enums

ASI contract has no enums.

Events

ASI contract has no custom events.

This document is proprietary and confidential. No part of this document may be disclosed
in any manner to a third party without the prior written consent of Hacken.

Modifiers

ASI has no modifiers.

Fields

ASI contract has following fields and constants:

- mapping(address => uint256) private _balances;
- mapping(address => uint256) private _mock;
- mapping(address => uint256) private _scores;
- mapping(address => mapping(address => uint256)) private _allowances;
- uint256 private constant _totalSupply = 10 * 10**6 * 10**18;
- uint256 private constant _antiBotsPeriod = 45;
- uint256 private _totalFees;
- uint256 private _totalScores;
- uint256 private _rate;
- mapping(address => bool) private _exchanges;
- mapping(address => uint256) private _lastTransactionPerUser;
- string private _name = 'asi.finance';
- string private _symbol = 'ASI';
- uint8 private _decimals = 18;

Functions

ASI has following public functions:

- *constructor*
- *name*
- *symbol*
- *decimals*
- *totalSupply*
- *balanceOf*
- *transfer*
- *allowance*
- *approve*
- *transferFrom*
- *increaseAllowance*

This document is proprietary and confidential. No part of this document may be disclosed in any manner to a third party without the prior written consent of Hacken.

- *decreaseAllowance*

BBOTSToken.sol

Description

BBOTSToken is a IBEP20 token.

Imports

MasterChef has following imports:

- IBEP20.sol
- IPancakePair.sol
- Context.sol
- Address.sol
- SafeMath.sol
- Ownable.sol

Inheritance

BBOTSToken is

- Context
- IBEP20
- Ownable

Usages

BBOTSToken contract has following usages:

- SafeMath for uint256;
- Address for address;

Structs

BBOTSToken contract has no custom data structures.

Enums

BBOTSToken contract has no enums.

Events

BBOTSToken contract has no custom events.

This document is proprietary and confidential. No part of this document may be disclosed in any manner to a third party without the prior written consent of Hacken.

Modifiers

BBOTSToken has no custom modifiers.

Fields

BBOTSToken contract has following fields and constants:

- mapping (address => uint256) private _balances;
- mapping (address => mapping (address => uint256)) private _allowances;
- uint256 private _totalSupply;
- string private _name = 'asibots.finance';
- string private _symbol = 'BBOTS';
- uint8 private _decimals = 18;
- uint256 private _lastBurn;
- address private _pair;
- uint public constant DAILY_BURN = 20;
- uint public constant DEX_BURNER = 30000;

Functions

BBOTSToken has following public functions:

- *constructor*
- *name*
- *symbol*
- *decimals*
- *totalSupply*
- *balanceOf*
- *transfer*
- *allowance*
- *approve*
- *transferFrom*
- *increaseAllowance*
- *decreaseAllowance*
- *burnFrom*
- *dexBurn*
- *getLastBurn*
- *getNextBurn*
- *isPair*
- *setPair*

This document is proprietary and confidential. No part of this document may be disclosed in any manner to a third party without the prior written consent of Hacken.

Audit overview

🟥 🟥 🟥 🟥 Critical

No critical issues were found.

🟨 🟨 🟨 High

No high issues were found.

🟩 🟩 Medium

No medium severity issues were found.

🟦 Low

1. Add 0x0 address validation in function setExchange in ASI.sol
2. Default value of uint type in solidity is 0, so code construction in function ASI._calculateBalance could be simplified.

🟦 Lowest / Code style / Best Practice

1. Functions decreaseAllowance in both contracts throw an error while decreasing amount is higher than already allowed. It is not an issue, but it will be better to set allowance to 0. To simplify the user journey.

2. It is a good practice to move repeatable requires operations to modifiers due to DRY principles.

This document is proprietary and confidential. No part of this document may be disclosed in any manner to a third party without the prior written consent of Hacken.

Conclusion

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. For the contract, 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 engineers found **2** low, **2** informational issue during the audit.

Notice: the audit scope is limited and not include all files in the repository. Though, reviewed contracts are secure, we may not guarantee secureness of contracts that are not in the scope.

This document is proprietary and confidential. No part of this document may be disclosed in any manner to a third party without the prior written consent of Hacken.

Disclaimers

Hacken 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).

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 its vulnerabilities that can lead to hacks. Thus, the audit can't guarantee the explicit security of the audited smart contracts.

This document is proprietary and confidential. No part of this document may be disclosed in any manner to a third party without the prior written consent of Hacken.