HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: SOAR
Date: January 27th, 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 SOAR.
Type	ERC-20 token with specific functionality
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Approved by	Andrew Matiukhin | CTO and co-founder Hacken
Etherscan link	https://etherscan.io/address/0xbae5f2d8a1299e5c4963eaff3312399253f27ccb
Timeline	21ST JAN 2021 – 27TH JAN 2021
Changelog	27TH JAN 2021 - Initial Audit

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

Table of contents

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

Introduction

Hacken OÜ (Consultant) was contracted by SOAR (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 January 21st, 2021 – January 27th, 2021.

Scope

The scope of the project is main net smart contract that can be found on Etherscan: https://etherscan.io/address/0xbae5f2d8a1299e5c4963eaff3312399253f27ccb#code
We have scanned this smart contract for commonly known and more specific vulnerabilities. List 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
Functional review	▪ Business Logics Review ▪ Functionality Checks ▪ Access Control & Authorization ▪ Escrow manipulation ▪ Token Supply manipulation ▪ Assets integrity ▪ User Balances manipulation ▪ Data Consistency manipulation ▪ Kill-Switch Mechanism ▪ Operation Trails & Event Generation

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

Executive Summary

According to the assessment, the Customer's smart contracts are secure.

Insecure	Poor secured	Secured	Well-secured

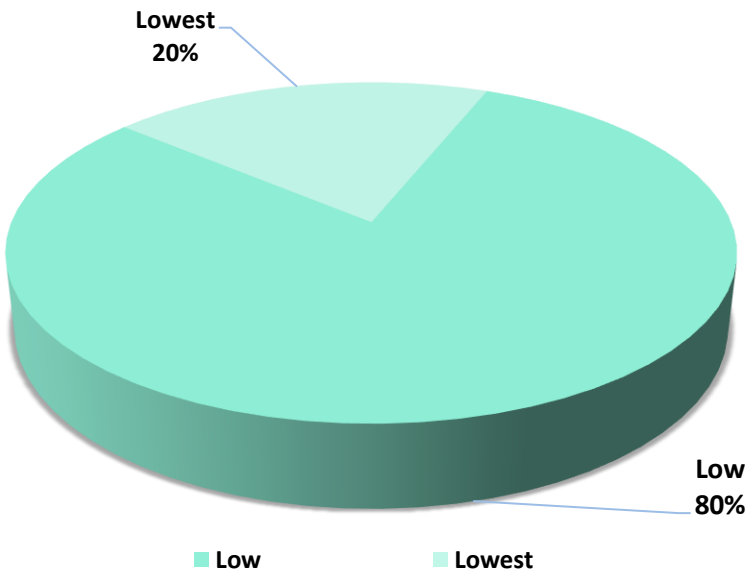
You are here

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 4 low severity issues during the audit. Overall code quality is good.

Low severity issues do not have major security impact; risks may be accepted by Customer not to redeploy the contract.

Graph 1. The distribution of vulnerabilities.



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

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to assets lose or data manipulations.
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 assets lose or data manipulations.
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.

AS-IS overview

SOAR smart contracts

SOAR smart contract consists of contract Context, interface IERC20, library SafeMath, library Address, contract Ownable, contract SOAR.

Context

Description

Context is a standard OpenZeppelin smart contract for information about execution context.

IERC20

Description

IERC20 is a standard interface for interactions with ERC20 tokens.

SafeMath

Description

SafeMath is a standard OpenZeppelin library for mathematical operations to prevent overflows.

Address

Description

Address is a standard OpenZeppelin library with different functions for address.

Ownable

Description

Ownable is a standard OpenZeppelin smart contract for basic access control with an owner role.

SOAR

Description

SOAR is a smart contract for ERC20 token with custom functions.

Imports

SOAR is audited on-chain, thus, all imports are described above.

Inheritance

SOAR contract is Context, IERC20, Ownable.

Usages

SOAR contract has following usages:

- using SafeMath for uint256;
- using Address for address;

Structs

SOAR contract has no custom structs.

Enums

SOAR contract has no custom enums.

Events

SOAR contract has no custom events.

Modifiers

SOAR contract has no custom modifiers.

Fields

SOAR contract has following parameters:

- mapping (address => uint256) private _rOwned;
- mapping (address => uint256) private _tOwned;
- mapping (address => mapping (address => uint256)) private _allowances;
- mapping (address => bool) private _isExcluded;
- address[] private _excluded;
- uint256 private constant MAX = ~uint256(0);
- uint256 private constant _tTotal = 10 * 10**6 * 10**9;
- uint256 private _rTotal = (MAX - (MAX % _tTotal));
- uint256 private _tFeeTotal;
- string private _name = 'SOAR.FI';
- string private _symbol = 'SOAR';
- uint8 private _decimals = 9;
- uint256 private startTime;

Functions

SOAR has following functions:

- ***constructor***
  Description
  initializes contract
  Visibility
  public
  Input parameters
  None
  Constraints
  None
  Events emit
  - emit Transfer(address(0), _msgSender(), _tTotal);

  Output
  None
- ***name***
  Description
  returns name
  Visibility
  public view
  Input parameters
  None
  Constraints
  None
  Events emit
  Name
  Output
  _name
- ***symbol***
  Description
  returns symbol
  Visibility
  public view
  Input parameters
  None
  Constraints
  None
  Events emit
  Name
  Output
  _symbol

- ***decimals***
  Description
  returns decimals
  Visibility
  public view
  Input parameters
  None
  Constraints
  None
  Events emit
  Name
  Output
  _decimals
- ***totalSupply***
  Description
  returns totalSupply
  Visibility
  public view
  Input parameters
  None
  Constraints
  None
  Events emit
  Name
  Output
  _tTotal
- ***balanceOf***
  Description
  returns balance for address
  Visibility
  public view
  Input parameters
  None
  Constraints
  None
  Events emit
  None
  Output
  tokenFromReflection(_rOwned[account]);
- ***transfer***
  Description
  calls internal token transfer

Visibility
public
Input parameters
- address recipient
- uint256 amount

Constraints
None
Events emit
None
Output
true

- ***allowance***
  Description
  returns allowance for owner and spender
  Visibility
  public view
  Input parameters
  - address owner
  - address spender

  Constraints
  None
  Events emit
  Name
  Output
  _allowances[owner][spender]
- ***approve***
  Description
  calls internal approve
  Visibility
  public
  Input parameters
  - address spender
  - uint256 amount

  Constraints
  None
  Events emit
  None
  Output
  true
- ***transferFrom***
  Description
  calls internal token transfer and approve

Visibility
public
Input parameters
- address sender
- address recipient
- uint256 amount

Constraints
None
Events emit
None
Output
true

- ***increaseAllowance***
Description
calls internal approve to increase allowance
Visibility
public
Input parameters
  - address spender
  - uint256 addedValue

Constraints
None
Events emit
None
Output
true
- ***decreaseAllowance***
Description
calls internal approve to decrease allowance
Visibility
public
Input parameters
  - address spender
  - uint256 subtractedValue

Constraints
None
Events emit
None
Output
true
- ***isExcluded***
Description

checks whether address is excluded
Visibility
public view
Input parameters
- address account

Constraints
None
Events emit
None
Output
_isExcluded[account]

- ***totalFees***
Description
returns total fees
Visibility
public view
Input parameters
None
Constraints
None
Events emit
None
Output
_tFeeTotal
- ***reflect***
Description
adds amount to fee
Visibility
public
Input parameters
None
Constraints
None
Events emit
None
Output
None
- ***reflectionFromToken***
Description
returns reflection amount from token
Visibility
public view

Input parameters
- uint256 tAmount
- bool deductTransferFee

Constraints
None
Events emit
None
Output
rAmount
rTransferAmount

- ***tokenFromReflection***
Description
returns token amount from reflection
Visibility
public view
Input parameters
- uint256 rAmount

Constraints
None
Events emit
None
Output
rAmount.div(currentRate)

- ***excludeAccount***
Description
adds address to excluded
Visibility
external
Input parameters
- address account

Constraints
- onlyOwner

Events emit
None
Output
None

- ***includeAccount***
Description
removes address from excluded
Visibility
external
Input parameters

- address account

Constraints

- onlyOwner

Events emit

None

Output

None

- ***_approve***

Description

approves spender amount for the owner

Visibility

private

Input parameters

- address owner
- address spender
- uint256 amount

Constraints

None

Events emit

- emit Approval(owner, spender, amount);

Output

None

- ***_transfer***

Description

transfers tokens from one account to another

Visibility

private

Input parameters

- address sender
- address recipient
- uint256 amount

Constraints

None

Events emit

None

Output

None

- ***_transferStandard***

Description

performs transfer from not excluded to not excluded

Visibility

private

Input parameters
- address sender
- address recipient
- uint256 tAmount

Constraints
None
Events emit
- emit Transfer(sender, recipient, tTransferAmount);

Output
None

- ***_transferToExcluded***
Description
performs transfer from not excluded to excluded
Visibility
private
Input parameters
  - address sender
  - address recipient
  - uint256 tAmount

Constraints
None
Events emit
  - emit Transfer(sender, recipient, tTransferAmount);

Output
None

- ***_transferBothExcluded***
Description
performs transfer from excluded to excluded
Visibility
private
Input parameters
  - address sender
  - address recipient
  - uint256 tAmount

Constraints
None
Events emit
  - emit Transfer(sender, recipient, tTransferAmount);

Output
None

- ***_transferFromExcluded***
Description

performs transfer from excluded to not excluded
Visibility
private
Input parameters
- address sender
- address recipient
- uint256 tAmount

Constraints
None
Events emit
- emit Transfer(sender, recipient, tTransferAmount);

Output
None

- ***_reflectFee***
  Description
  reflects fee to token
  Visibility
  private
  Input parameters
  - uint256 rFee
  - uint256 tFee

  Constraints
  None
  Events emit
  None
  Output
  None
- ***_getValues***
  Description
  returns transfer amount and fee values
  Visibility
  private view
  Input parameters
  - uint256 tAmount

  Constraints
  None
  Events emit
  None
  Output
  (rAmount, rTransferAmount, rFee, tTransferAmount, tFee)
- ***_getTValues***
  Description

returns token transfer amount and fee
Visibility
private pure
Input parameters
- uint256 tAmount

Constraints
None
Events emit
None
Output
(tTransferAmount, tFee)

- ***_getRValues***
Description
returns reflection transfer amount and fee
Visibility
private view
Input parameters
- uint256 tAmount

Constraints
None
Events emit
None
Output
(rAmount, rTransferAmount, rFee)

- ***_getRate***
Description
returns current reflection rate
Visibility
private view
Input parameters
None
Constraints
None
Events emit
None
Output
rSupply.div(tSupply)

- ***_getCurrentSupply***
Description
returns current token and reflection supply
Visibility
private view

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

Input parameters
None
Constraints
None
Events emit
None
Output
(rSupply, tSupply)

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

Audit overview

Critical

No critical issues were found.

High

No high issues were found.

Medium

No medium issues were found.

Low

1. Solidity version is not locked. It’s recommended to lock solidity pragma to a specific stable version.
2. Code is not covered with in-code documentations; it’s recommended to add function description for all functions.
3. No unit tests were developed for the project. It’s recommended to have 100% test coverage for code.
4. It’s highly recommended to have more events and emits for crucial functionality, for example, adding/removing to excluded, fee reflection etc.

Lowest / Code style / Best Practice

5. Default condition is unreachable for all ifs in transfer. It’s recommended to have 3 if..else checks and the default should be standard transfer. It may potentially save some gas.

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

Conclusion

Smart contracts within the scope was 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 4 low severity issues during the audit.

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

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