HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Blank

Date: March 5th, 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
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	Token
Platform	Ethereum / Solidity
Methods	Manual Review
Repository	https://github.com/Blank-Wallet/Blank-token/blob/master/BLANK.sol
Commit	cff1a5088a6b9c45caa597080e769f5921e9daf1
Deployed contract	
Timeline	4 MARCH 2021 – 5 MARCH 2021
Changelog	5 MARCH 2021 – INITIAL AUDIT

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

Table of contents

Introduction 4
Scope 4
Executive Summary 5
Severity Definitions 6
AS-IS overview 7
Conclusion 14
Disclaimers 15

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

Introduction

Hacken OÜ (Consultant) was contracted by Blank (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.

Scope

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

Contract deployment address:
Repository: https://github.com/Blank-Wallet/Blank-token/blob/master/BLANK.sol
Commit: cff1a5088a6b9c45caa597080e769f5921e9daf1
Files:
BLANK.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

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

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 has issues that should be fixed. The code quality should be increased.

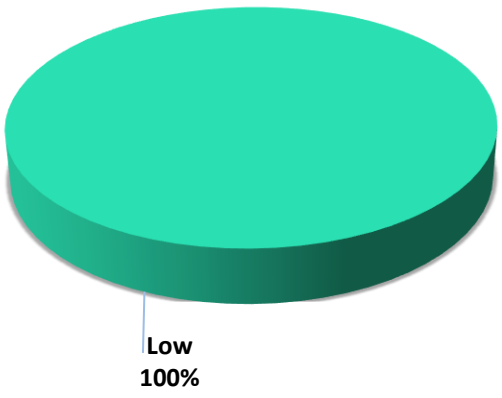
Insecure	Poor secured	Secured	Well-secured

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 **3** low issues during the audit.

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 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.

AS-IS overview

BLANK.sol

Description

Token interfaces and implementations are inherited from OpenZeppelin Contracts.

Imports

BLANK contract has the following imports:

- abstract contract Context
- interface IERC20

Usages

BLANK contract has no custom usages.

Structs

BLANK contract has no data structures.

Enums

BLANK contract has no custom enums.

Events

BLANK contract has no events.

Modifiers

BLANK has no custom modifiers.

Fields

BLANK contract has following constants:

- mapping (address => uint256) private _balances;
- mapping (address => mapping (address => uint256)) private _allowances;
- uint256 private _totalSupply;
- string private _name = "Blank Token";

- string private _symbol = "BLANK";
- constructor (uint256 totalSupply_)

Functions

BlankToken has following public functions:

- ***name***
  Visibility
  public view virtual
  Input parameters
  None
  Constraints
  None
  Events emit
  None
  Output
  - string memory
- ***symbol***
  Visibility
  public view virtual
  Input parameters
  None
  Constraints
  None
  Events emit
  None
  Output
  - String memory
- ***decimals***
  Visibility
  public view virtual
  Input parameters
  None
  Constraints
  None
  Events emit
  None
  Output
  - uint8

- ***totalySupply***
  Visibility
  public view virtual override
  Input parameters
  None
  Constraints
  None
  Events emit
  None
  Output
  - uint256
- ***balansOf***
  Visibility
  public view virtual override
  Input parameters
  - address account

  Constraints
  None
  Events emit
  None
  Output
  - uint256
- ***transfer***
  Visibility
  public view virtual override
  Input parameters
  - address recipient
  - uint256 amount

  Constraints
  None
  Events emit
  None
  Output
  - bool
- ***allowance***
  Visibility
  public view virtual override
  Input parameters
  - address owner

- address spender

Constraints
None
Events emit
None
Output

- uint256

- ***approve***

Visibility
public view virtual override
Input parameters

- address spender
- uint256 amount

Constraints
None
Events emit
None
Output

- bool

- ***transferFrom***

Visibility
public view virtual override
Input parameters

- address sender
- address recipient
- uint256 amount

Constraints
None
Events emit
None
Output

- bool

- ***burn***

Visibility
public virtual
Input parameters

- uint256 amount

Constraints
None

Events emit
None
Output
None

- ***increaseAllowance***
  Visibility
  public virtual
  Input parameters
  - address spender
  - uint256 addedValue

  Constraints
  None
  Events emit
  None
  Output
  - bool

- ***decreaseAllowance***
  Visibility
  public virtual
  Input parameters
  - address spender
  - uint256 subtractedValue

  Constraints
  None
  Events emit
  None
  Output
  - bool

- ***_transfer***
  Visibility
  internal virtual
  Input parameters
  - address sender
  - address recipient
  - uint256 amount

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

Output
None

- ***_burn***
  Visibility
  internal virtual
  Input parameters
  - address account
  - uint256 amount

  Constraints
  None
  Events emit
  - Transfer(account, address(0), amount);

  Output
  None
- ***_approve***
  Visibility
  internal virtual
  Input parameters
  - address owner
  - address spender
  - uint256 amount

  Constraints
  None
  Events emit
  - Approval(owner, spender, amount)

  Output
  None

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 severity issues were found.

Medium

No medium severity issues were found.

Low

1. BLANK.sol:49, :53, :57, :61, :65, :69, :74, :78, :83, :93, :97, :102 - wrong function visibility. Prefer external to public visibility level. A function with public visibility modifier that is not called internally. Changing visibility level to external increases code readability. Moreover, in many cases functions with external visibility modifier spend less gas comparing to functions with public visibility modifier.
2. BLANK.sol:110, :122, :133 - wrong function visibility. Those functions are declared as internal, but there is nothing inheritance of this contract. If functions will live only inside the contract, they could be declared as private.
3. BLANK.sol:78 - Using approve function of the ERC-20 token standard. The approve function of ERC-20 is vulnerable. Using front-running attack one can spend approved tokens before change of allowance value. Only use the approve function of the ERC-20 standard to change allowed amount to 0 or from 0 (wait till transaction is mined and approved). Notice: this is OpenZeppelin standard.

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

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