HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Polkalokr

Date: April 1st, 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 Polkalokr.
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/Polkalokr/lkr-token/commit/26bae7bd735a0a6da3bad482a814deab5e3a350d
Commit	
Deployed contract	
Timeline	29 MAR 2021 – 1 APR 2021
Changelog	1 APR 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 Polkaokr (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 March 29th, 2021 – April 1st, 2021.

Scope

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

Contract deployment address:
Repository
File:
contracts\token\PolkalokrToken.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 contracts are well-secured.

Insecure	Poor secured	Secured	Well-secured

You are 

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 no issues during the audit.

Notice: any address can be added to the black list by the administrator. After that, sending and receiving tokens are blocked. This functionality is declared as protection against snipe bots.

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.

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.

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

AS-IS overview

PolkalokrToken.sol

Description

PolkalokrToken is an ERC20 token contract. The contract mints 100 million tokens. The contract is upgradeable.

Imports

PolkalokrToken contract has following imports:

- AccessControlUpgradeable.sol – from OpenZeppelin.
- ERC20Upgradeable.sol – from OpenZeppelin.
- PausableUpgradeable.sol – from OpenZeppelin.
- ILocker.sol – from the project files.
- ILockerUser.sol – from the project files.

Inheritance

PolkalokrToken is ERC20Upgradeable, PausableUpgradeable, AccessControlUpgradeable and ILockerUser.

Structs

PolkalokrToken contract has no data structures.

Enums

PolkalokrToken contract has no enums.

Events

PolkalokrToken contract has no custom events.

Modifiers

PolkalokrToken has following modifiers:

- onlyAdmin - checks if sender has admin role;
- onlyAdmin - checks if sender has pauser role;

Fields

PolkalokrToken contract has following fields and constants:

- string constant NAME = 'Polkalokr' - the name of the token;
- string constant SYMBOL = 'LKR' - the symbol of the token;
- uint8 constant DECIMALS = 18 - decimals;
- uint256 constant TOTAL_SUPPLY = 100_000_000 * 10**uint256(DECIMALS) - the total supply;
- bytes32 public constant PAUSER_ROLE = keccak256("PAUSER_ROLE") - the pauser role identifier;
- bytes32 public constant WHITELISTED_ROLE = keccak256("WHITELISTED_ROLE") - the pauser role identifier;
- bytes32 public constant BLACKLISTED_ROLE = keccak256("BLACKLISTED_ROLE") - the pauser role identifier;
- ILocker public override locker;

Functions

PolkalokrToken contract has following functions:

- ***initialize***

  Description

  Initializes the contract.

  Visibility

  external

  Input parameters

  None

  Constraints

  - initialized modifier.

  Events emit

  None

Output

None

- ***__PolkalokrToken_init***

  Description

  Initializes the contract. Sets name and symbol of the token.

  Visibility

  internal

  Input parameters

  None

  Constraints

  - initialized modifier.

  Events emit

  None

  Output

  None

- ***__PolkalokrToken_init_unchained***

  Description

  Sets roles and mints the total supply.

  Visibility

  internal

  Input parameters

  None

Constraints

- initialized modifier.

Events emit

None

Output

None

- ***_beforeTokenTransfer***

Description

Overrides the default _beforeTokenTransfer function with additional checks.

Visibility

internal

Input parameters

- address from
- address to
- uint256 amount

Constraints

- If paused, then only a whitelisted role can transfer tokens.
- The sender should not be blacklisted.
- The receiver should not be blacklisted.

Events emit

None

Output

None

- ***pause***

  Description

  Suspends the contract.

  Visibility

  external

  Input parameters

  None

  Constraints

  Only pauser role can call it.

  Events emit

  None

  Output

  None

- ***unpause***

  Description

  Continues the contract.

  Visibility

  external

  Input parameters

  None

  Constraints

  Only pauser role can call it.

Events emit

None

Output

None

- ***setLocker***

  Description

  Sets locker.

  Visibility

  external

  Input parameters

  None

  Constraints

  Only admin role can call it.

  Events emit

  None

  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

No low severity issues were found.

Lowest / Code style / Best Practice

No lowest severity issues were found.

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 no issues during the audit.

Notice: any address can be added to the black list by the administrator. After that, sending and receiving tokens are blocked. This functionality is declared as protection against snipe bots.

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.

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.