HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Merlin
Date: May 15th, 2021

Document

Name	Smart Contract Code Review and Security Analysis Report for Merlin - Second Review
Approved by	Andrew Matiukhin	CTO Hacken OU
Type	BEP20 Token, Governance, Minter, Pool and Vaults
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Zip archive	Merlin-2021-05-16.tar.gz (md5: a0af96e683684b17ff1577707deb1ddf)
Timeline	06 MAY 2021 – 15 MAY 2021
Changelog	08 MAY 2021 – INITIAL AUDIT 15 MAY 2021 – SECOND REVIEW

Table of contents

Introduction

Hacken OÜ (Consultant) was contracted by Merlin (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 on May 15th, 2021.

Scope

The scope of the project is the smart contracts provided in the zip archive:

Merlin-2021-05-16.tar.gz (**md5:** a0af96e683684b17ff1577707deb1ddf)
/contracts/MerlinPool.sol
/contracts/VaultLpToCake.sol
/contracts/VaultLp2Lp.sol
/contracts/MerlinMinter.sol
/contracts/VaultCakeToCake.sol
/contracts/MerlinToken.sol
/contracts/MerlinChef.sol
/contracts/VaultMerlin.sol
/contracts/interface/IPancakeRouter01.sol
/contracts/PriceCalculatorBSC.sol
/contracts/library/VaultController.sol
/contracts/interface/IStrategy.sol
/contracts/interface/IPancakePair.sol
/contracts/interface/IPancakeRouter02.sol
/contracts/library/PoolConstant.sol
/contracts/library/WhitelistUpgradeable.sol
/contracts/interface/AggregatorV3Interface.sol
/contracts/library/PausableUpgradeable.sol
/contracts/library/Pausable.sol
/contracts/library/SafeDecimal.sol
/contracts/interface/IMerlinChef.sol
/contracts/library/RewardsDistributionRecipientUpgradeable.sol
/contracts/Migrations.sol
/contracts/interface/IMerlinMinter.sol
/contracts/interface/IMasterChef.sol
/contracts/library/RewardsDistributionRecipient.sol
/contracts/interface/IPancakeFactory.sol
/contracts/interface/IStrategyHelper.sol
/contracts/interface/IZapBSC.sol
/contracts/interface/IStakingRewards.sol
/contracts/interface/IVaultController.sol

We have scanned these smart contracts 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
Functional review	▪ Business Logics Review ▪ Functionality Checks ▪ Access Control & Authorization ▪ Escrow manipulation ▪ Token Supply manipulation ▪ Asset's 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
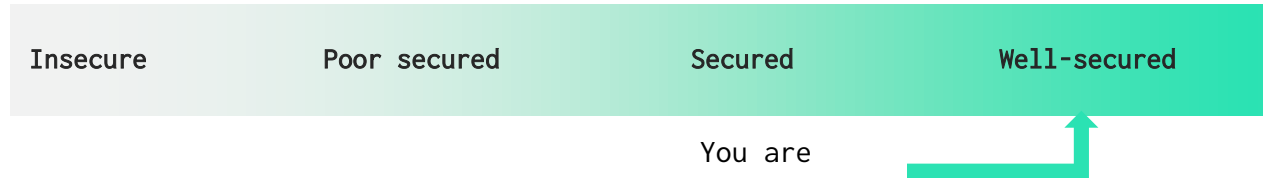
Our team performed an analysis of code functionality, manual audit, and automated checks with Mythril and Slither. All issues found during automated analysis were manually reviewed, and important vulnerabilities are presented in the Audit overview section. All found issues can be found in the Audit overview section.

Security engineers found **1** low and **3** informational issues during the first review.

Security engineers found **no issues** during the second review.

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

25.0%

75.0%

● Informational ● Low
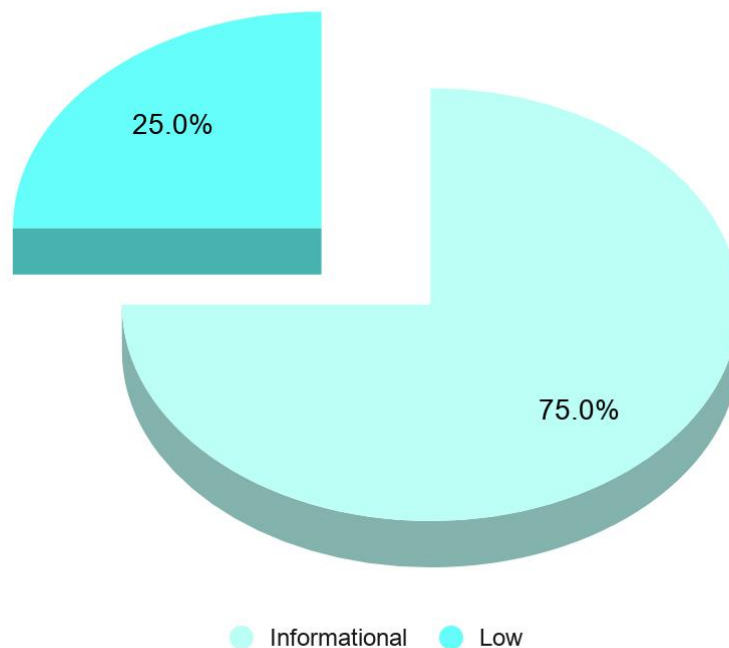
Severity Definitions

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.

www.hacken.io

Audit overview

■ ■ ■ ■ Critical

No Critical severity issues were found.

■ ■ ■ High

No High severity issues were found.

■ ■ Medium

No Medium severity issues were found.

■ Low

1. **Vulnerability:** Missing events access control

   Missing events for critical access control parameters

   Fixed before second review

■ Lowest / Code style / Best Practice

1. **Vulnerability:** Boolean equality

   Boolean constants can be used directly and do not need to be compared to true or false.

   Fixed before second review

2. **Vulnerability:** State variable that should be constant

   State variables that don't change their value should be declared constant to save gas.

   Fixed before second review

3. **Vulnerability:** Public function that could be declared external

   public functions that are never called by the contract should be declared **external** to save gas.

   Fixed before second review

Conclusion

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools.

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

Security engineers found **1** low and **3** informational issues during the first review.

Security engineers found **no issues** during the second review.

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 the 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.

www.hacken.io