HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: DefiWarrior
Date: September 29th, 2021

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

Document

Name	Smart Contract Code Review and Security Analysis Report for DefiWarrior.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	ERC20 token lockable; Lock
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/Defi-Warrior/presale-contracts
Commit	c4d0efb61a3d644a3e6aca72d141f42cd11fdae4
Technical Documentation	NO
JS tests	YES
Timeline	03 SEPTEMBER 2021 - 29 SEPTEMBER 2021
Changelog	10 SEPTEMBER 2021 - Initial Audit 21 SEPTEMBER 2021 - Second Review 29 SEPTEMBER 2021 - Third Review

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 DefiWarrior (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of the Customer's smart contract and its code review conducted between September 3rd, 2021 - September 29th, 2021.

Second review conducted on September 21st, 2021.

Third review conducted on September 29th, 2021.

Scope

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

Repository:
https://github.com/Defi-Warrior/presale-contracts
Commit:
c4d0efb61a3d644a3e6aca72d141f42cd11fdae4
Technical Documentation: No
JS tests: Yes
Contracts:
extensions/IERC20.sol
extensions/IERC20Metadata.sol
extensions/ILocker.sol
utils/Address.sol
utils/Context.sol
utils/EnumerableSet.sol
utils/Ownable.sol
utils/SafeMath.sol
utils/Strings.sol
LockV2.sol
LockableERC20Token.sol
MockLocker.sol

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

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
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 well-secured.

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. All found issues can be found in the Audit overview section.

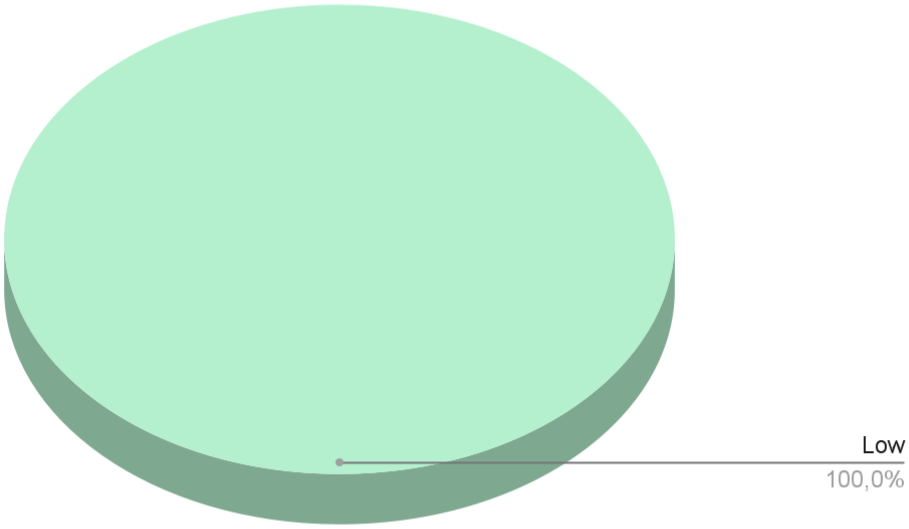
As a result of the audit, security engineers found **1** high, **2** medium and **2** low severity issues.

After the second review, security engineers found **1** high and **1** low severity issues.

After the second review, security engineers found **1** low severity issue.

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

Graph 1. The distribution of vulnerabilities after the audit.



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

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

Audit overview

Critical

No critical issues were found.

High

The owner could lock transfers anytime

The current contract logic means that the owner could lock any amount of tokens on any address at any time. It would be much clear and safe for the community to have a list of pre-defined lock durations and as well addresses.

Recommendation: Please rewrite locking logic to be more clear.

Fixed before the third review

Medium

1. Tests could not be executed

   Provided tests could not be executed because of several reasons:

   - Migrations are written incorrectly
   - Tests import artifacts that aren't present in the contracts

   Recommendation: Please double check tests and migrations to make sure everything is working smoothly.

   Fixed before the second review

2. Probably logic issues

   Looking through the logic of the **LockerV2**.getLockedAmount and **LockerV2**.lock functions it looks like we should keep locked a certain amount of tokens before the IDO and after it started, we should unlock 5% of the locked tokens.

   Right now **LockerV2**.checkLock function returns **true** for any amount and any address before the IDO is started.

   Looking through the **DefiWarriorToken**._transfer method, it looks like if the locker is set and IDO is not started, we would always revert with any token balance or amount for transfer.

   Recommendation: Please make sure the **LockerV2**.checkLock logic is correct.

   Fixed before the second review

Low

1. No interface implementation

   The **LockerV2** contract will definitely be used as a locker for the **DefiWarriorToken**, but it doesn't implement the needed **ILocker** explicitly.

Recommendation: Please declare ILocker interface implementation in the LockerV2 contract.

Fixed before the second review

2. Too many digits

   Literals with many digits are difficult to read and review.

   Recommendation: Please use the scientific notation and ether units (ie. *10e9 ether*).

   Fixed before the third review

3. Costly deployment

   Right now to deploy the LockerV2 contract will cost about **14M** of gas. Having the average gas price for 55 Gwei (for reference), it will cost about **1ETH** to deploy the LockerV2 contract to the ethereum mainnet, which is really costly.

   Recommendation: Please consider rewriting the initial lock logic if you want to save some gas fees.

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.

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

As a result of the audit, security engineers found **1** high, **2** medium and **2** low severity issues.

After the second review, security engineers found **1** high and **1** low severity issues.

After the second review, security engineers found **1** low severity issue.

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 the security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bug-free 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 the security of smart contracts.

Technical Disclaimer

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks. Thus, the audit can't guarantee the explicit security of the audited smart contracts.