HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Grizzly
Date: March 4th, 2022

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 Grizzly.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	Token
Language	Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://bitbucket.org/grizzlyfi/grizzlyficontracts
Commit	09b9339ad898957a79724b0c9871c411021bdc5e
Technical Documentation	Yes: 1. docs: https://blog.grizzly.fi/tokenomics/ 2. https://blog.grizzly.fi/grizzly-fi-introduction/ 3. WP: https://docs.google.com/document/d/1X68ROD8fJ4OGxDF81mw8bEdoxeAd6YJL0-8M_v-kUXs/edit?usp=sharing
JS tests	Yes
Website	grizzly.fi
Timeline	19 January - 24 February 2022
Changelog	24 January 2022 - Initial Audit 24 February 2022 - Second Review 04 March 2022 - Third Review

Table of contents

Introduction

Hacken OÜ (Consultant) was contracted by Grizzly(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 on January 19th - January 24th, 2022.

The second review was conducted on February 28th, 2022.

The third review was conducted on March 4th, 2022.

Scope

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

Repository:
https://bitbucket.org/grizzlyfi/grizzlyficontracts
Commit:
09b9339ad898957a79724b0c9871c411021bdc5e
Technical Documentation: Yes

- docs: https://blog.grizzly.fi/tokenomics/
- https://blog.grizzly.fi/grizzly-fi-introduction/
- WP: https://docs.google.com/document/d/1X68ROD8fJ4OGxDF81mw8bEdoxeAd6YJL0-8M_v-kUXs/edit?usp=sharing

JS tests: Yes
Contracts:
./contracts/*.sol

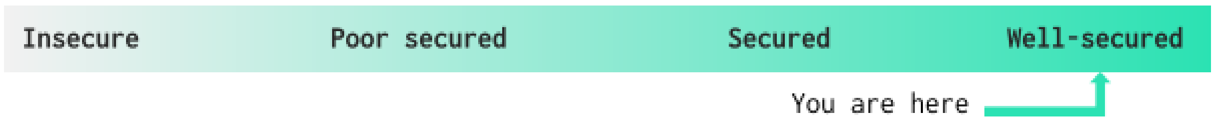
We have scanned this smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that are considered:

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

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

Executive Summary

According to the assessment, the Customer's smart contracts well-secured.



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 **3** medium, and **2** low severity issues.

After the second review, the code contains **1** critical and **2** medium issues.

After the third review, the code contains **no** issues.

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

Audit overview

■■■■ Critical

The minting rate depends on the token price that is received from a current Honey-BNB rate. This rate can be manipulated with the flashloan before deposit/withdraw/changeStrategy transactions and more rewards will be minted for all pool participants.

Contract: StakingPool.sol

Function: mintTokens

Recommendation: inline described behavior with the implemented behavior.

Status: Fixed.

■■■ High

No high severity issues were found.

■■ Medium

1. The function description states that LP tokens should be withdrawn. Though, instead of this, LPs are exchanged to underlying assets by removing liquidity from the PancakeSwap pool.

   Contract: StakingPool.sol

   Function: claimLpTokens

   Recommendation: inline described behavior with the implemented behavior.

   Status: Fixed.

2. The contract is designed to work with an LP token that represents the pair ETH-StakedToken (Honey). Though, any other pair can be passed.

   Contract: StakingPool.sol

   Function: constructor

   Recommendation: validate that one side of the pair is StakedToken.

   Status: Fixed.

3. Depositors lose their unused tokens. Though, an amount of token may not be big enough, users receive less LP tokens on their staking balance than they expected to.

   Contract: Grizzly.sol

   Function: _deposit

Status: Fixed: unused tokens are used to incentivice a user to call stakeRewardsForBounty.

Recommendation: rewrite an exchange logic to remove leftovers or send leftovers back to the caller if total amount is larger than tx expenses.

Low

1. Tokens transfer results are not validated all over the code.

   Recommendation: Though used tokens implementations are safe, validating transfer results is considered as best practice.

   Status: Fixed.

2. Allowances are set to \`2**256 - 1\`. Redundant math operation is performed. Also, some tokens implementations limit allowance that can be set to max uint96 value.

   Contract: BaseConfig.sol

   Function: constructor

   Recommendation: use type(X).max to set up max allowance.

   Status: Fixed.

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 **3** medium, and **2** low severity issues.

After the second review, the code contains **1** critical and **2** medium issues.

After the third review, the code contains **no** issues.

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.