HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: TGB Finance
Date: December 9th, 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 TGB Finance.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	ERC20 token
Platform	Binance Smart Chain / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Deployed contract	https://bscscan.com/address/0xffef225b4a1b5de683d53dd745664c4ef8840f61#code
Technical Documentation	NO
JS tests	NO
Website	tgb.finance
Timeline	06 DECEMBER 2021 - 09 DECEMBER 2021
Changelog	09 DECEMBER 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 TGB Finance (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 December 6th, 2021 - December 9th, 2021.

Scope

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

URL:
https://bscscan.com/address/0xffef225b4a1b5de683d53dd745664c4ef8840f61#code
Technical Documentation: No
JS tests: No
Contracts:
Context.sol
DividendPayingToken.sol
DividendPayingTokenInterface.sol
DividendPayingTokenOptionalInterface.sol
ERC20.sol
IERC20.sol
IERC20Metadata.sol
IterableMapping.sol
IUniswapV2Factory.sol
IUniswapV2Pair.sol
IUniswapV2Router.sol
Ownable.sol
SafeMath.sol
SafeMathInt.sol
SafeMathUint.sol
TGB.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 secured but could have issues calculating fees if administrator functions would be called in certain order.

Insecure	Poor secured	Secured	Well-secured

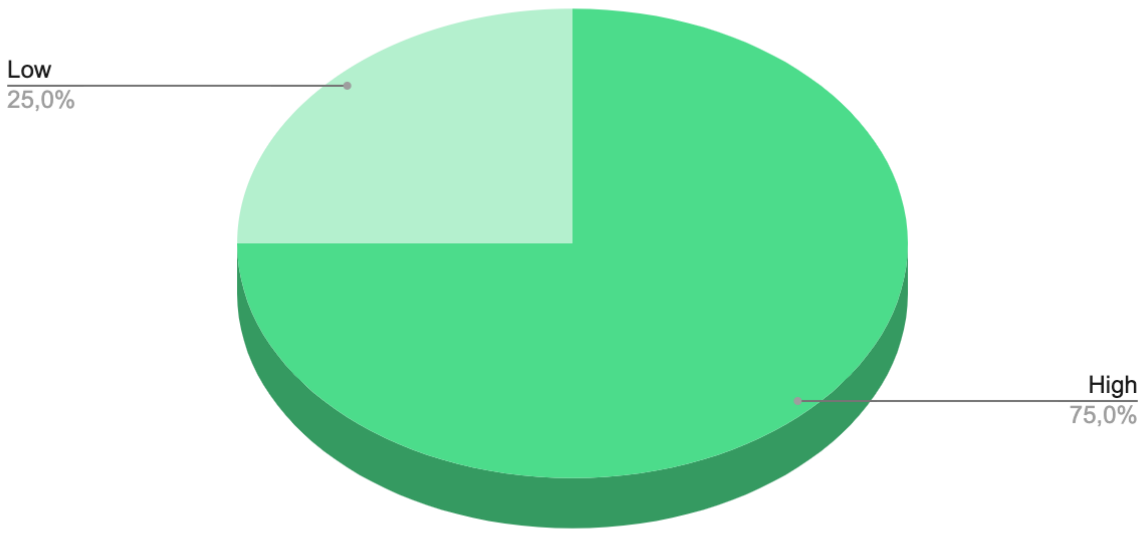
You are here

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 **3** high and **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

1. Incorrect fee calculations.

The “totalFees” amount is calculated incorrectly. While “setBurnFee” recalculates “totalFees” as the sum of: BUSDRewardsFee + liquidityFee + marketingFee + burnFee, but other fee-setter functions don’t use “burnFee” in the same calculation.

Contract: TGB.sol

Functions: setMarketingFee, setLiquiditFee, setBUSDRewardsFee

Recommendation: While the contract is already deployed the only thing what we could recommend is to call “setBurnFee” function each time after calling the functions specified above, it will fix calculations.

Customer’s notice: Burning fees are set to zero, which means this should not cause any issues, and we intend to call the nofees function after we burn 50% of the tokens and will share all the information with our community.

2. Not including in dividends back.

Calling _setAutomatedMarketMakerPair with value=true will exclude specified address from dividends. While calling with value=false should include in dividends back, but it doesn’t.

Contract: TGB.sol

Functions: _setAutomatedMarketMakerPair

Recommendation: While the contract is already deployed the only thing we could recommend is to check five times before calling the function to not exclude some address that should be included in dividends.

Customer’s notice: The function _setAutomatedMarketMakerPair is not meant to re-add an address to the dividends, since it requires the address not to be the same if calling the function another time. In this case when we need to change our market pair (ex: Pancakeswap pair) we choose if we need it to be one of the dividends or no using the bool value.

3. No functionality to include back in dividends

Calling excludeFromDividends by mistake will exclude provided address from dividends and there is no ability to include it back.

Contract: TGB.sol

Functions: excludeFromDividends

Recommendation: While the contract is already deployed the only thing we could recommend is to check five times before calling the function to not exclude some address that should be included in dividends.

Customer's notice: The dividends feature is temporary and we will stop it when the burning reach 50% of the total supply (nofees), in this case we are very careful about address to be excluded from the dividends.

Medium

No medium issues were found

Low

No events on fee changes.

While changing fees it's suggested to emit corresponding events so the community could track such changes off-chain.

Contract: TGB.sol

Functions: noFees, setBUSDRewardsFee, setLiquiditFee, setMarketingFee, setBurnFee

Recommendation: Please consider emitting events on such sufficient changes like fees.

Customer's notice: We will inform our community by any announcements regarding the slippage fees or the busd rewards stopping.

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 **3** high and **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.