SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: TBOT
Date: August 23rd, 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 TBOT.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	ERC20 token; Transfer controller
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Deployed contract	https://etherscan.io/address/0xa4f2fdb0a5842d62bbaa5b903f09687b85e4bf59#code
Technical Documentation	NO
JS tests	NO
Timeline	18 AUGUST 2021 - 23 AUGUST 2021
Changelog	23 AUGUST 2021 - Initial Audit

Table of contents

Introduction

Hacken OÜ (Consultant) was contracted by TBOT (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 August 18th, 2021 - August 23rd, 2021.

Scope

The scope of the project is smart contracts in the repository:
Deployed Smart Contract:
https://etherscan.io/address/0xa4f2fdb0a5842d62bbaa5b903f09687b85e4bf59#code
Technical Documentation: No
JS tests: No
Contracts:
TokenMintERC20MintableToken

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 are secured.

Insecure	Poor secured	Secured	Well-secured
		You are here ↑	

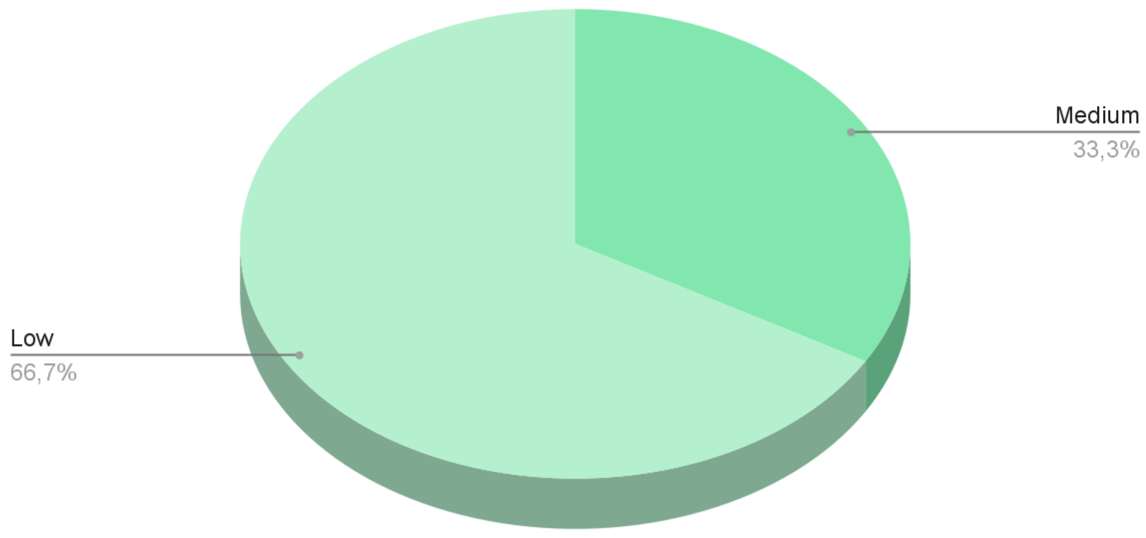
Our team performed an analysis of code functionality, manual audit, and automated checks with Mythril and Slither. All issues found during automated analysis were manually reviewed, and important vulnerabilities are presented in the Audit overview section. All found issues can be found in the Audit overview section.

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

Notice:

This contract has an unrestricted ability for an owner to mint any amount of tokens at any time.

Graph 1. The distribution of vulnerabilities after the audit.



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

No critical issues were found.

🟨 🟨 🟨 High

No high severity issues were found.

🟩 🟩 Medium

1. Unrestricted mining available

   The contract has a minter role which is able to mint an unrestricted amount of tokens at any time.

   Recommendation: Please consider moving the minter role to the community-driven governance contract (DAO) or at least Timelock so the community can react timely if the minting happens.

🟦 Low

1. Missing zero-address validation

   There is a missing zero-address verification for the feeReceiver address. If it is specified as zero, the contract wouldn't be deployed.

   Recommendation: Please add a zero-check for addresses.

2. A public function that could be declared external

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

   Recommendation: Use the **external** attribute for functions never called from the contract.

Lines: 235

```
function totalSupply() public view returns (uint256) {
```

Lines: 242

```
function balanceOf(address account) public view returns (uint256) {
```

Lines: 262

```
function allowance(address owner, address spender) public view returns
(uint256) {
```

Lines: 273

```
function approve(address spender, uint256 value) public returns (bool) {
```

Lines: 290

```
function transferFrom(address sender, address recipient, uint256 amount)
public returns (bool) {
```

Lines: 308

```
function increaseAllowance(address spender, uint256 addedValue) public
returns (bool) {
```

Lines: 327

```
function decreaseAllowance(address spender, uint256 subtractedValue)
public returns (bool) {
```

Lines: 528

```
function mint(address account, uint256 amount) public onlyMinter returns
(bool) {
```

Lines: 577

```
function transferMinterRole(address newMinter) public {
```

Lines: 586

```
function burn(uint256 value) public {
```

Lines: 595

```
function name() public view returns (string memory) {
```

Lines: 602

```
function symbol() public view returns (string memory) {
```

Lines: 609

```
function decimals() public view returns (uint8) {
```

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** medium and **2** low severity issue.

Notice:

This contract has an unrestricted ability for an owner to mint any amount of tokens at any time.

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.