HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: DYNXT
Date: August 18th, 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 DYNXT.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	ERC20 token with Fees and SwapAndLiquify
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/zippa713/dynxt/blob/main/DynastyxContract.sol
Commit	d1593e45d341267e6cf4f843e416e69f52843670
Deployed contract	https://bscscan.com/token/0x9128d0a29c89d4ed520a36f8a4154b0bc64b6396
Technical Documentation	NO
JS tests	NO
Timeline	16 AUGUST 2021 - 18 AUGUST 2021
Changelog	18 AUGUST 2021 - INITIAL AUDIT

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

Table of contents

Introduction 4

Scope 4

Executive Summary 5

Severity Definitions 7

Audit overview 8

Conclusion 9

Disclaimers 11

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

Introduction

Hacken OÜ (Consultant) was contracted by DYNXT (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 16th, 2021 - August 18th, 2021.

Scope

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

Repository:
https://github.com/zippa713/dynxt/blob/main/DynastyxContract.sol
Commit:
d1593e45d341267e6cf4f843e416e69f52843670
Technical Documentation: No
JS tests: No
Contracts:
DynastyxContract.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

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

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 contract is secured but has the ability for the owner to change critical values without emitting events.

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.

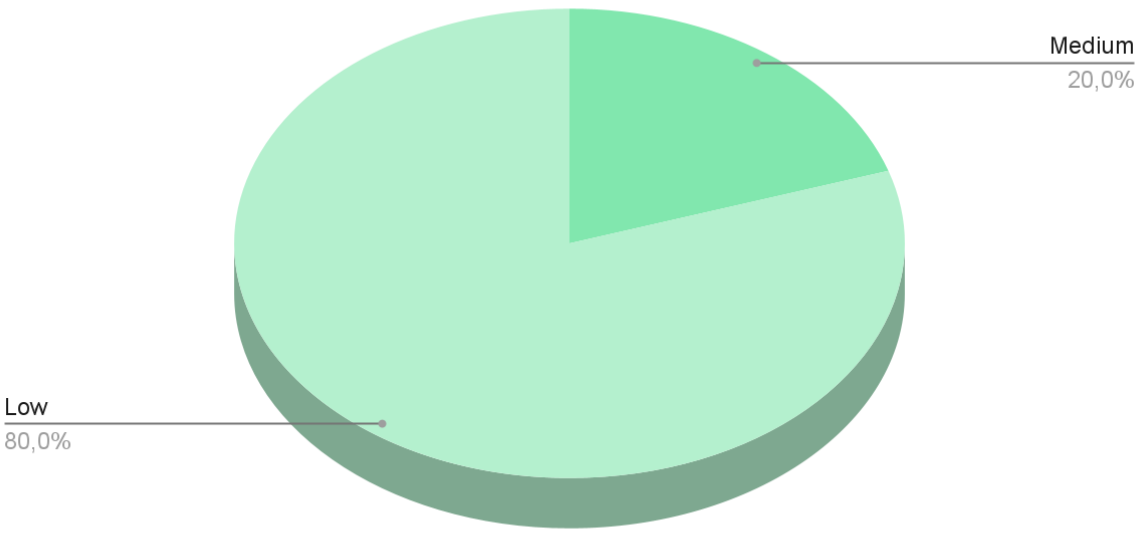
As a result of the audit, security engineers found **1** medium and **4** low severity issues.

Notice:

The **Dynastyx** contract contains one medium issue which is about not emitting events when changing critical values like thresholds, including/excluding from fees, max transaction amount, operation and reserved wallet. Counting on that contract is already deployed and those values could be a game-changer for the community so we'd recommend monitoring those from time to time manually.

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

No high severity issues were found.

Medium

1. No event on critical state changes

   There are multiple critical state variables changing which we'd recommend to emit an event for a better off-chain track.

   Recommendation: Please consider emitting events when changing operationWallet, reservedWallet, _maxTxAmount, _isExcludedFromFees, tierOneThreshold, tierTwoThreshold, tierThreeThreshold, tierFourThreshold, tierFiveThreshold.

Low

1. Using SafeMath with solidity >= 0.8.0

   Starting solidity v0.8.0 already has a built-in math checking and it's not necessary to check for overflow and underflow manually.

   Recommendation: Please do not use SafeMath library for solidity >= 0.8.0 to save gas

2. Unused state variable

   The contract defines swapping private state variable which is neither read nor write in the contract

   Recommendation: please remove the unused state variable definition to save gas.

Lines: #854

```
bool private swapping;
```

3. State variables that could be declared constant

   Constant state variables should be declared constant to save gas.

   Recommendation: Add the **constant** attributes to state variables that never change.

Lines: #863

```
uint256 public _totalSupply = 200000000000000 * (10**18);
```

Lines: #865

```
uint256 public numTokensSellToAddToLiquidity = 1000000000 * (10**18);
```

4. 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: #928

```
function isExcludedFromFee(address account) public view returns(bool) {
```

Lines: #936

```
function includeInFee(address account) public onlyOwner {
```

Lines: #940

```
function setSwapAndLiquifyEnabled(bool _enabled) public onlyOwner {
```

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** medium and **4** low severity issues.

Notice:

The **Dynastyx** contract contains one medium issue which is about not emitting events when changing critical values like thresholds, including/excluding from fees, max transaction amount, operation and reserved wallet. Counting on that contract is already deployed and those values could be a game-changer for the community so we'd recommend monitoring those from time to time manually.

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.