HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Delta Theta
Date: December 6th, 2021

Document

Name	Smart Contract Code Review and Security Analysis Report for Delta Theta.
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
Repository	https://github.com/deltatheta-finance/deltatheta-protocol-v2
Commit	9d87f828b90283ffdb2a01c0d9237f32e3dfefd1
Technical Documentation	YES
JS tests	YES
Website	deltatheta.tech
Timeline	19 NOVEMBER 2021 – 06 DECEMBER 2021
Changelog	01 DECEMBER 2021 – INITIAL AUDIT 06 DECEMBER 2021 – SECOND REVIEW

Table of contents

Introduction

Hacken OÜ (Consultant) was contracted by Delta Theta (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 November 19th, 2021 – December 1st, 2021.

Second review conducted on December 6th, 2021.

Scope

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

Repository:
https://github.com/deltatheta-finance/deltatheta-protocol-v2

Commit:
9d87f828b90283ffdb2a01c0d9237f32e3dfefd1

Technical Documentation: Yes
(https://drive.google.com/drive/folders/1YnGn1K77c_OpSnuxVsUU9ZgKb67WJy2F)

JS tests: Yes (https://github.com/deltatheta-finance/deltatheta-protocol-v2/tree/main/test)

Contracts:
types/structures/MarketOrderBook.sol
interfaces/IFactory.sol
types/structures/MarketOrder.sol
types/structures/LimitOrder.sol
interfaces/IUniswapPair.sol
interfaces/IWrappedToken.sol
interfaces/IDexRouter.sol
interfaces/IPair.sol
core/PairV2.sol
tests/WETH.sol
libraries/MarketOrderLib.sol
types/structures/Requests.sol
libraries/LimitOrderBookLib.sol
libraries/LimitOrderLib.sol
interfaces/CToken.sol
core/FactoryV2.sol
types/structures/Pair.sol
libraries/UtilsLib.sol
types/Enums.sol
external/ERC2771Context.sol
libraries/MarketOrderBookLib.sol
types/structures/LimitOrderBook.sol
tests/Token.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 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 **2** low severity issues.

After the second review security engineers found that **all issues** were addressed.

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

No medium severity issues were found.

■ Low

1.   Missing zero address validation

Initialization function and constructor sets addresses that could not be changed later. Mistakenly providing zero address could make the contract misfunction.

Contracts: FactoryV2.sol, PairV2.sol

Functions: constructor, initialize, updateImplementation

Recommendation: Please add zero-address validation to make sure provided addresses are not zeros.

Status: Fixed

2.   Missing events

Changing significant contract values should emit an event to make it easy for the community to track such changes off-chain.

Contracts: PairV2.sol

Functions: updateTrustedForwarder, updateTeam, updateTrustedForwarder, updateImplementation

Recommendation: Please add zero-address validation to make sure provided addresses are not zeros.

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 **2** low severity issues.

After the second review security engineers found that **all issues** were addressed.

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.