Blaize.Security

July 22th, 2021 / V. 1.0

SMART CONTRACT AUDIT

TABLE OF CONTENTS

Audit rating 2

Technical summary 3

The graph of vulnerabilities distribution 4

Severity definition 5

Auditing strategy and techniques applied \ Procedure 6

Executive summary 7

Complete analysis 8

Disclaimer 11

AUDIT RATING

PEAKDEFI smart contract's source code was taken from repository provided by PEAKDEFI team.

SCORE **9.8**/10

The scope of the project is **PEAKDEFI:**

1/ contracts/fund/PeakDeFiFactory.sol;

2/ contracts/fund/PeakDeFiFund.sol;

The scope of the audit is the code at the main branch with commit:

- **7a6826a1b4354c1e509244083e40cb11c66673af**

Post-audit scope for validation includes the code at the main branch with commit:

- **f4f56993833d2f99663b3b8821cced899c2751c0**

TECHNICAL SUMMARY

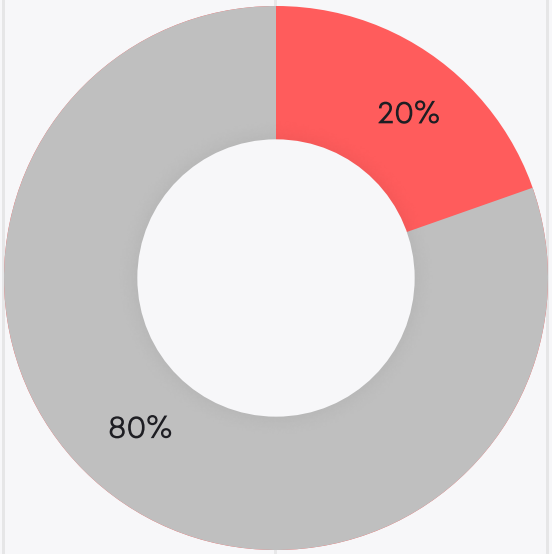
In this report, we consider the security of the contracts for **PEAKDEFI protocol**. Our task is to find and describe security issues in the smart contracts of the platform. This report presents the findings of the security audit of Customer's smart contracts conducted between - **10.07.2021-22.07.2021.**

The scope of the audit includes manual manual and automatic tools analysis, that bases on the smart contracts code, documentation and requirements presented by PEAKDEFI team. Scope includes performing additional testing strategies. Though, in order to ensure a security of the contract Blaize.Security team recommends PEAKDEFI team put in place a bug bounty program to encourage further and active analysis of the smart contracts.

THE GRAPH OF VULNERABILITIES DISTRIBUTION:

MEDIUM

LOW



The table below shows the number of found issues and their severity. A total of 5 problems were found. 3 issues were fixed or verified by the PEAKDEFI team.

	FOUND	FIXED
Critical	0	0
High	0	0
Medium	1	1
Low	4	2
Lowest	0	0

SEVERITY DEFINITION

Critical

A system contains several issues ranked as very serious and dangerous for users and the secure work of the system. Needs immediate improvements and further checking.

High

A system contains a couple of serious issues, which lead to unreliable work of the system and migh cause a huge information or financial leak. Needs immediate improvements and further checking.

Medium

A system contains issues which may lead to mediumfinancial loss or users' private information leak. Needs immediate improvements and further checking.

Low

A system contains several risks ranked as relatively small with the low impact on the users' information and financial security. Needs improvements.

Lowest

A system does not contain any issue critical to the secure work of the system, yet is relevant for best

AUDITING STRATEGYAND TECHNIQUES APPLIED \ PROCEDURE

We have scanned this smart contract for commonly known and more specific vulnerabilities:

- Unsafe type inference;
- Timestamp Dependence;
- Reentrancy;
- Implicit visibility level;
- Gas Limit and Loops;
- Transaction-Ordering Dependence;
- Unchecked external call - Unchecked math;
- DoS with Block Gas Limit;
- DoS with (unexpected) Throw;
- Byte array vulnerabilities;
- Malicious libraries;
- Style guide violation;
- ERC20 API violation;
- Uninitialized state/storage/ local variables;
- Compile version not fixed.

Procedure

In our report we checked the contract with the following parameters:

- Whether the contract is secure;
- Whether the contract corresponds to the documentation;
- Whether the contract meets best practices in efficient use of gas, code readability;

Automated analysis:

Scanning contract by several public available automated analysis tools such as Mythril, Solhint, Slither and Smartdec. Manual verification of all the issues found with tools.

Manual audit:

Manual analysis of smart contracts for security vulnerabilities. Checking smart contract logic and comparing it with the one described in the documentation.

EXECUTIVE SUMMARY

According to the assessment, the Customer's smart contracts have no critical security problems but do not follow the best coding practice in some aspects. We described issues and added Customer comments according to the conclusion of these documents. Please read the whole document to estimate the risks well.

	RATING
Security	10
Gas usage and logic optimization	9.7
Code quality	9.7
Total	9.8

COMPLETE ANALYSIS

MEDIUM	✔ Resolved

Address zero checks are missing.

At:
contracts/fund/PeakDeFiFactory.sol#44-53
contracts/fund/PeakDeFiFund.sol.initParams

Storage variables are set to passed arguments, but there could be a misuse and/or frontend mistake so zero address will be passed, which will lead to dysfunction of the contract.

Recommendation:

Add ŕequire(addr != address(0), “ZERO_ADDRESS”);`for all passed arguments before setting the internal variable

Post-audit:

Fixed by the Client in the file PeakDeFiFactory.

LOW	✔ Resolved

Code is never used.

The method CloneFactory.isClone at contracts/lib/CloneFactory.sol#48-73 is never used.

Recommendation:

Remove unused code.

LOW	Unresolved

Method should be external.

updatePeakDeFiFundAddress() should be declared external at contracts/fund/PeakDeFiProxy.sol#15-23

Recommendation:

Make methods external to save gas on deployment.

Post-audit:

Client acknowledged.

LOW	✔ Resolved

Code is not needed.

At contracts/fund/PeakDeFiFund.sol:143
proxyAddr = _proxyAddr;
proxy = IPeakDeFiProxy(_proxyAddr);
you can use address(proxy) instead of proxyAddr.

Recommendation:

Remove proxyAddr, and use address(proxy) instead.

LOW	Unresolved

Revert instead of return 0.

At:
contracts/fund/PeakDeFiFund.sol:203
contracts/fund/PeakDeFiFund.sol:224
And in a number of similar places.
It seems that returning 0 could potentially lead to misusing the method.

Recommendation:

Use `revert("...")` instead of `return 0`.

Post-audit:

Client acknowledged.

DISCLAIMER

The information presented in this report is an intellectual property of the customer including all presented documentation, code databases, labels, titles, ways of usage as well as the information about potential vulnerabilities and methods of their exploitation. This audit report does not give any warranties on the absolute security of the code. Blaize.Security is not responsible for how you use this product and does not constitute any investment advice.

Blaize.Security does not provide any warranty that the working product will be compatible with any software, system, protocol or service and operate without interruption. We do not claim the investigated product is able to meet your or anyone else requirements and be fully secure, complete, accurate and free of any errors and code inconsistency.

We are not responsible for all subsequent changes, deletions and relocations of the code within the contracts that are the subjects of this report.

You should perceive Blaize.Security as a tool which helps to investigate and detect the weaknesses and vulnerable parts that may accelerate the technology improvements and faster error elimination.