

# PREFACE

## Objectives

The purpose of this document is to highlight the identified bugs/issues in the provided codebase. This audit has been conducted in a closed and secure environment, free from influence or bias of any sort. This document may contain confidential information about IT systems/architecture and intellectual property of the client. It also contains information about potential risks and the processes involved in mitigating/exploiting the risks mentioned below.

The usage of information provided in this report is limited, internally, to the client. However, this report can be disclosed publicly with the intention to aid our growing blockchain community; under the discretion of the client.

## Key Understandings

**TABLE OF CONTENTS**

[**PREFACE**](#_g7yg189t6uur) **2**

[Objectives](#_tyd6muuhl71r) 2

[Key Understandings](#_9nybeqrulq0) 2

[**INTRODUCTION**](#_dx7u2pq06djj) **4**

[Scope](#_en195ocdmohn) 5

[**AUDIT REPORT**](#_gch5146ssdkj) **6**

[Executive Summary](#_cn28ac3o2r98) 6

[Findings](#_kajpl4p84xe) 7

[Critical-risk issues](#_hjk235pgl1wm) 7

[High-risk issues](#_s04ujznhf0hp) 7

[Medium-risk issues](#_v2003w71ul0s) 8

[Low-risk issues](#_c6hnp9l5jz9k) 9

[**DISCLAIMER**](#_mlul2n3y97ld) **11**

# 

# INTRODUCTION

BlockApex (Auditor) was contracted by Dafi (Client) for the purpose of conducting a Smart Contract Audit/Code Review. This document presents the findings of our analysis which took place on 5th July 2021 .

| Name |
| --- |
| Dafi |
| Auditor |
| Moazzam Arif | Shakeib Shaida |
| Platform |
| Ethereum/Solidity |
| Type of review |
| Super-staking with Rebasing |
| Methods |
| Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review |
| Git Repository |
| *(zip files provided)* |
| White paper/Documentation |
| <https://docs.dafiprotocol.io/> |
| Document log |
| Initial Audit: 05th July 2021 |
| Final Audit: 4th August 2021 |

## Scope

The git-repository shared was checked for common code violations along with vulnerability-specific probing to detect major issues/vulnerabilities. Some specific checks are as follows:

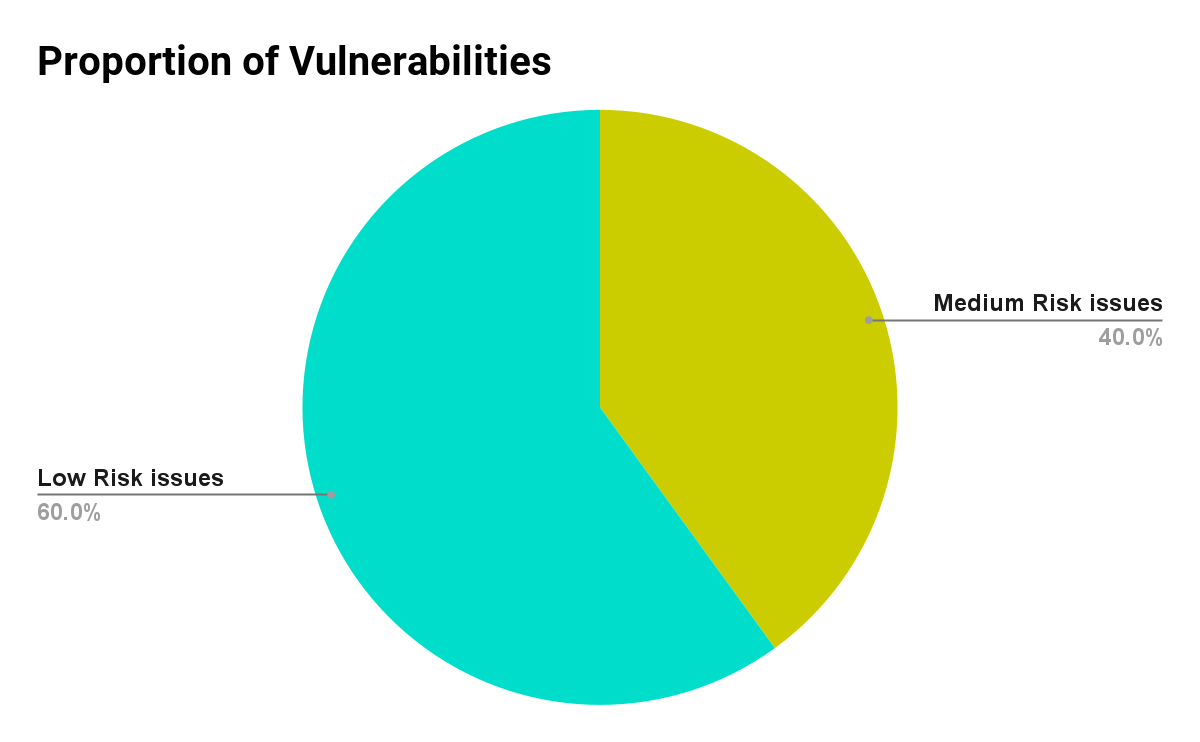
| **Code review** |  | **Functional review** |
| --- | --- | --- |
| Reentrancy | Unchecked external call | Business Logics Review |
| Ownership Takeover | ERC20 API violation | Functionality Checks |
| Timestamp Dependence | Unchecked math | Access Control & Authorization |
| Gas Limit and Loops | Unsafe type inference | Escrow manipulation |
| DoS with (Unexpected) Throw | Implicit visibility level | Token Supply manipulation |
| DoS with Block Gas Limit | Deployment Consistency | Asset’s integrity |
| Transaction-Ordering Dependence | Repository Consistency | User Balances manipulation |
| Style guide violation | Data Consistency | Kill-Switch Mechanism |
| Costly Loop |  | Operation Trails & Event Generation |

# AUDIT REPORT

## Executive Summary

The analysis indicates that the contracts audited are **secured (with a few financial scalability issues).**

Our team performed a technique called **“Filtered Audit”**, where the contract was separately audited by two individuals. After their thorough and rigorous process of manual testing, an automated review was carried out using Mythril, MythX and Slither. All the flags raised were manually reviewed and re-tested.

**Our team found:** 

| # of issues | Severity of the risk |
| --- | --- |
| 0 | Critical Risk issue(s) |
| 0 | High Risk issue(s) |
| 2 | Medium Risk issue(s) |
| 3 | Low Risk issue(s) |
| 0 | Informatory issues(s) |

## Findings

### Critical-risk issues

No issues found.

### High-risk issues

No High risk issues were found.

### 

### Medium-risk issues

1. **Rebasing and Reward Distribution Formula is flawed**

**File:** RebaseEngine.sol

**Distribution per second is calculated abnormally** In the following lines:-

uint poolCurrent = ((MAX\_DAFI - dDAFIDistributed + feesDeposited) \* demandFactorNew);

uint distributePerSecond = poolCurrent / database.getProgramDuration(); // the remaining pools is divided by the whole program duration

**Remedy:**There needs to be an extensive discussion on how the current formula and financial model works, only then a revised version can be created.

**Update (Aug 4th 2021):**We are working out the formula with the Dafi team. It’s not finalized yet. Once ready, we will share that too.

1. **Malicious whitelisted account can withdraw all staked token**

**File:** TokenPool.sol  
In function transfer(address to, uint256 value)malicious whitelisted accounts can withdraw all tokens in the contract

**Remedy:**As transfer function is only used in staking, only StakingManger should have access to this control. This can be done by creating OnlyStakingManager modifier.

**Devs Response:**As TokenPool contract is deployed by stakingManger contract, and there is no way to whitelist users from stakingManger contract, so the only whitelisted account will be StakingManger contract’s Address. There is no way to withdraw staked tokens

**Remarks:**

As TokenPool smart contract is deployed from staking Manger so the funds are safe.

### 

### Low-risk issues

1. **Reward Balance calculated incorrectly**

**File:** StakingManagerv1.sol

**function** rewardBalance(address user) **external** **view** returns (uint) {

return (userStake.totalUnclaimed \* networkDemand.calculateNetworkDemand()) / (userStake.lastDemandFactor);

}

When a user claims/unstake tokens, the reward fee is applied. But in the above function implementation no reward fee is adjusted. This will create a bad-user experience.

**Remedy:** Adjust the reward fee while calculating reward balance.

1. **Gas Cost Optimization While staking**

**File:** RebaseEngine.sol  
When a user stakes, the rebase function is called, which in turns calls the \_rebase(address user) internal function which calculates the reward for the user. As the rewards are also calculated on claimRewards and Unstake. When calling the stake method, Only user’s stake metadata like lastStakingAccumulatedWeight and other params should be updated. Instead of calling the whole \_rebase(address user), add another function which just updates the necessary parameters.

**Devs Response:**With the previous formula, explained in High Risk Issue 1, it needs to be done in that way.

**Update (Aug 4th, 2021):**   
RebaseEngine smart contract is being refactored.

1. **Changing Owner should be in two steps**

**Files:** Any file that is derived from Openzeppelin’s Ownable.sol.

Ownable files should implement two step ownership transfer. This eliminates the risk if the owner is mistakenly set to some address, which doesn't have access to private keys.

**Devs Response:**They are satisfied with the current approach and want to continue doing so.

# 

# 

# DISCLAIMER

The smart contracts provided by the client for audit purposes have been thoroughly analyzed in compliance with the global best practices till date w.r.t cybersecurity vulnerabilities and issues in smart contract code, the details of which are enclosed in this report.

This report is not an endorsement or indictment of the project or team, and they do not in any way guarantee the security of the particular object in context. This report is not considered, and should not be interpreted as an influence, on the potential economics of the token, its sale or any other aspect of the project.

Crypto assets/tokens are results of the emerging blockchain technology in the domain of decentralized finance and they carry with them high levels of technical risk and uncertainty. No report provides any warranty or representation to any third-Party in any respect, including regarding the bug-free nature of code, the business model or proprietors of any such business model, and the legal compliance of any such business. No third-party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell any token, product, service or other asset. Specifically, for the avoidance of doubt, this report does not constitute investment advice, is not intended to be relied upon as investment advice, is not an endorsement of this project or team, and it is not a guarantee as to the absolute security of the project.

Smart contracts are deployed and executed on a blockchain. The platform, its programming language, and other software related to the smart contract can have its vulnerabilities that can lead to hacks. The scope of our review is limited to a review of the Solidity code and only the Solidity code we note as being within the scope of our review within this report. The Solidity language itself remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond Solidity that could present security risks.

This audit 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 security of smart contracts.