HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: DAFI Protocol
Date: October 28th, 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 DAFI Protocol.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	Staking
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/DAFIProtocol/dDAFI
Commit	780899034ef4966df8752a6030dbe7b3bcbd4bb1
Technical Documentation	YES
JS tests	NO
Website	dafiprotocol.io
Timeline	24 OCTOBER 2021 - 28 OCTOBER 2021
Changelog	26 OCTOBER 2021 - INITIAL AUDIT 28 OCTOBER 2021 - SECOND REVIEW

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

Table of contents

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

Introduction

Hacken OÜ (Consultant) was contracted by DAFI Protocol (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 October 24th, 2021 - October 26th, 2021.

Second review conducted on October 28th, 2021.

Scope

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

Repository:
https://github.com/DAFIProtocol/dDAFI/tree/main/contracts/V2
Commit:
780899034ef4966df8752a6030dbe7b3bcbd4bb1
Technical Documentation: Yes (https://docs.dafiprotocol.io/super-staking/super-staking-v2)
JS tests: No
Contracts:
interfaces\INetworkDemand.sol
interfaces\IPriceFeeds.sol
interfaces\IRebaseEngine.sol
interfaces\IStakingManager.sol
interfaces\ITVLFeeds.sol
network demand\NetworkDemand.sol
network demand\PriceFeeds.sol
network demand\TVLFeeds.sol
rebase engine\RebaseEngine.sol
StakingDatabase.sol
StakingManagerV2.sol
TokenPool.sol

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

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

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

Executive Summary

According to the assessment, the Customer's smart contracts are well-secured.

Insecure	Poor secured	Secured	Well-secured

You are here 

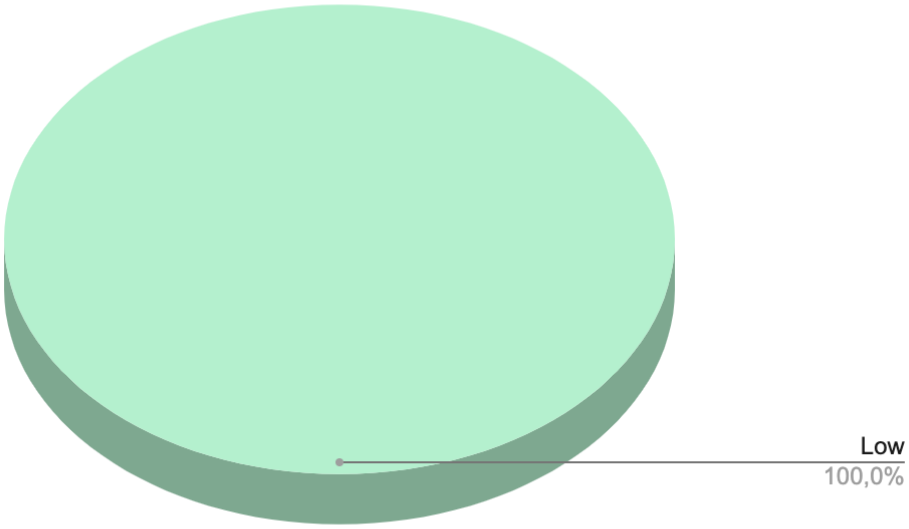
Our team performed an analysis of code functionality, manual audit, and automated checks with Mythril and Slither. All issues found during automated analysis were manually reviewed, and important vulnerabilities are presented in the Audit overview section. All found issues can be found in the Audit overview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

As a result of the audit, security engineers found **7** low severity issues.

As a result of the second review, security engineers found **6** low severity issues.

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

No medium severity issues were found.

Low

1. Too many digits

Literals with many digits are difficult to read and review.

Contracts: network demand/NetworkDemand.sol

Constants: SIX_DECIMALS, EIGHT_DECIMALS, SEVEN_DECIMALS, TWO_DECIMALS

Recommendation: Please use scientific notation and ether units suffix when it's possible. (ex.: *uint32 constant EIGHT_DECIMALS = 1e8;*)

2. Implicit visibility declaration

State variables that don't have explicitly declared visibility are implicitly set as internal.

Contracts: StakingManagerV2.sol

Variables: STAKING_ON, UNSTAKING_ON, INITIALIZED

Recommendation: Please always declare variables visibility explicitly to avoid misunderstandings.

3. Conformance to Solidity naming conventions

Solidity defines a naming convention that should be followed.

Contracts: StakingManagerV2.sol

Events: STAKED, UNSTAKED, REWARD_DISBURSED

Variables: STAKING_ON, UNSTAKING_ON, INITIALIZED

Recommendation: Follow the Solidity naming convention.

4. Incorrect contract name

The contract in the file StakingManagerV2.sol is named StakingManagerV1, which could lead to many confusions. Especially

while there is a StakingManagerV1.sol file which is actually out of the scope of the current audit.

Contracts: StakingManagerV2.sol

Recommendation: Please name the contract correctly.

Status: Fixed.

5. Not finished code

Some contracts have TODOs in the code which means some parts of the business logic are not implemented yet.

Contracts: TVLFeeds.sol

Recommendation: Please finish all TODOs.

6. No way to see rewards

There is no function for users to see their current rewards. It's more clear, open and informative for users to have the ability to see the current unclaimed earned rewards.

Contracts: StakingManagerV2.sol

Recommendation: Please add a function for users to see their current rewards balance.

7. Magic numbers

StakingManagerV1 contract is using a magic number in the code on line #212 (100000000) to calculate the Demand Factor.

Contracts: StakingManagerV2.sol

Function: _computeAndDisburseRewards

Recommendation: Please use constants defined in the NetworkDemand contract.

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 **7** low severity issues.

As a result of the second review, security engineers found **6** low severity issues.

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.