BlockApex

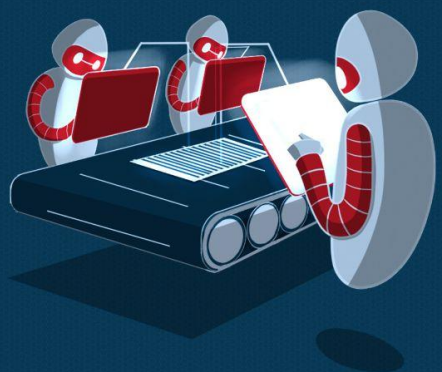
SMART CONTRACT SECURITY ANALYSIS REPORT

```
pragma solidity 0.7.0;
contract Contract {

    function hello() public returns (string) {
        return "Hello World!";
    }

    function findVulnerability() public returns (string) {
        return "Finding Vulnerability";
    }

    function solveVulnerability() public returns (string) {
        return "Solve Vulnerability";
    }
}
```



Powered by XORD

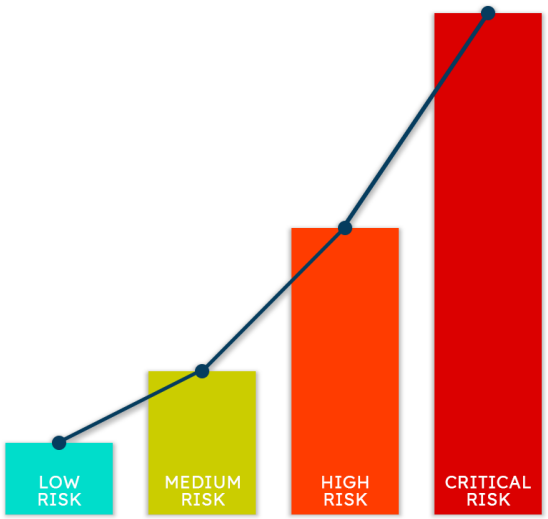
PREFACE

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



CRITICAL RISK ××××	Critical vulnerabilities are too easy to exploit and can lead to damages/loss in assets or manipulations.
HIGH RISK ×××	High-level vulnerabilities are difficult to exploit; however, they also have a significant impact on smart contract execution.
MEDIUM RISK ××	Medium-level vulnerabilities are equally imperative to fix but they tend to have minimal impact on asset loss or data manipulations.
LOW RISK ×	Lowest-level vulnerabilities, informational errors, violating code styles/practices usually can't affect smart contract execution; hence they can be ignored.

TABLE OF CONTENTS

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 _23rd October 2021___.

Name
Dafi
Auditor
Moazzam Arif | Kaif Ahmed
Platform
Ethereum/Solidity
Type of review
Dafi - Claims Contracts
Methods
Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Git repository
https://github.com/DAFIProtocol/ClaimPortal/tree/2e9b287a0707d08c8f1f98ffa9641eacf5450638
White paper/ Documentation
None
Document log
Initial review : 26-10-2021 (completed)
Final review : (pending)

Scope

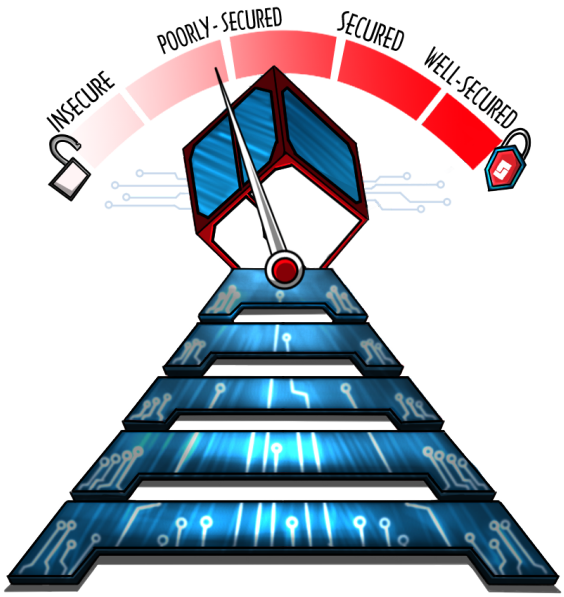
The git-repository shared was checked for common code violations along with vulnerability-specific probing to detect **major issues/vulnerabilities**. Some specific checks are as follows:

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 some of the functionalities in the contracts audited are **not working properly**.

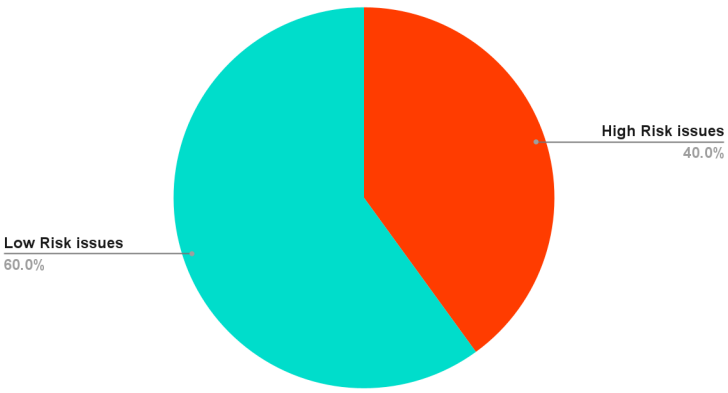


Our team performed a technique called “filtered audit”, where the contract was separately reviewed by two individuals. After their thorough and rigorous process of manual testing, an automated review was carried out using Slither. All the flags raised were manually reviewed and re-tested.

Our team found:

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

Proportion of Vulnerabilities



Findings

Critical-risk issues

No critical-risk issues were found.

High-risk issues

1. **Anyone can mint dafi tokens**

   File: dafiToken.sol

   Description:
   mint function can be called by anyone to mint supply

   Remedy:
   Add proper access controls to avoid unlimited minting

2. **Any user's token can be burnt by anyone**

   File: dafiToken.sol

   Description:
   Burn can be called by anyone and burns the token of other users

   Remedy:
   Add proper access controls to avoid unintentional burning

Medium-risk issues

No medium-risk issues were found.

Low-risk issues

1. **Allowance is subtracted twice in *transferFrom***

   File: dafiToken.sol

   Description:
   Allowance is subtracted twice. It will cause denial of service in some cases (and might cost more gas).

   Remedy:
   Remove the ***line***.

2. **Inconsistency in access controls in claiming rewards**

   File: dafiClaimContract.sol

   Description:
   claimRewardForUser has ***onlyAdmin*** modifier but claimReward doesn't have any modifier at all.

   Remedy:
   Make it consistent by adding modifiers to both or by removing them.

3. **Unfair reward amount distribution**

File: dafiClaimContract.sol

Description:
Reward is calculated for stakers as follows:

```
      uint256 _userAmount = usersStakes[_userId];
      uint256 timeDiff = block.timestamp.sub(startTime);
      /** startTime here is same for every staker **/
      uint256 time = timeDiff.sub(86400);
      uint256 timePassed = uint256(1000000000000000000).mul(time);
      uint256 perecentIncreament  = _userAmount.mul(timePassed);
      perecentIncreament  =
perecentIncreament.div(100000000000000000000);
```

startTime is the same for every staker and is updated to the latest block when deposit is called.

Remedy:
Calculate the time properly for every staker.

Informatory issues and Optimization

No informatory/optimization issues were found.

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.