

# 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

[**TABLE OF CONTENTS**](#_krwvvcm24vh6) **3**

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

[Scope](#_en195ocdmohn) 5

[Project Overview](#_h7d0ojd7txke) 6

[System Architecture](#_gch5146ssdkj) 6

[Methodology & Scope](#_4c7492d86iuj) 7

[**AUDIT REPORT**](#_of4oarut3cz8) **8**

[Executive Summary](#_cn28ac3o2r98) 8

[Findings](#_takyh8mfnnv0) 9

[Critical-risk issues](#_hjk235pgl1wm) 9

[High-risk issues](#_s04ujznhf0hp) 10

[Medium-risk issues](#_7gi0igqgh8pl) 11

[Low-risk issues](#_b8uzmreflwx5) 12

[Informatory issues and Optimization](#_uwkkipofwdis) 12

[**DISCLAIMER**](#_my0td8qwc60h) **13**

# 

# INTRODUCTION

BlockApex (Auditor) was contracted by Voirstudio (Client) for the purpose of conducting a Smart Contract Audit/Code Review. This document presents the findings of our analysis which took place on \_9th November 2021\_\_\_ .

| Name |
| --- |
| Unipilot Farming |
| Auditor |
| Moazzam Arif | Muhammad Jarir ul din |
| Platform |
| Ethereum/Solidity |
| Type of review |
| Staking and Farming |
| Methods |
| Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review |
| Git repository |
| <https://github.com/VoirStudio/unipilot-farming-contract/tree/dev> |
| White paper/ Documentation |
| [UnipilotFarm Contract Checklist](https://docs.google.com/document/d/17sS3pNzFGppCQGFirtT4L9lJCLYz8pJWqONy6z_7pKA/edit) |
| Document log |
| Initial audit completed on 12th November 2021 |
| Final audit completed on 15th November 2021 |

## Scope

The git-repository shared was checked for common code violations along with vulnerability-specific probing to detect[**major issues/vulnerabilities**](https://docs.google.com/document/d/1sEXMfjXRiqiEvnGKMUUoNvJ0pUMafy-7XsDEpP5_DJY/edit?usp=sharing). 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 |

## 

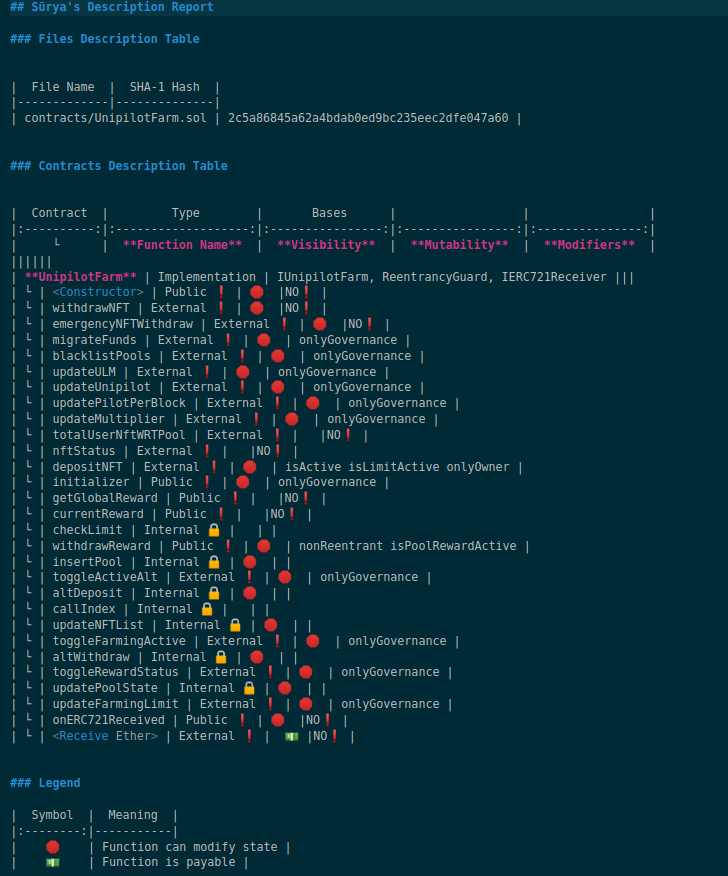
## Project Overview

Unipilot yield farming incentivizes LPs to earn $PILOT by staking their Unipilot NFT of whitelisted pools.

## System Architecture

Unipilot yield farming has 1 main smart contract.

[UnipilotFarm.sol](https://github.com/VoirStudio/unipilot-farming-contract/blob/dev/contracts/UnipilotFarm.sol): Smart contract which allows LPs to earn $PILOT by staking their Unipilot NFTs*. UnipilotFarm* linearly distributes the $PILOT according to *rewardPerBlock* and *rewardMultiplier*



## Methodology & Scope

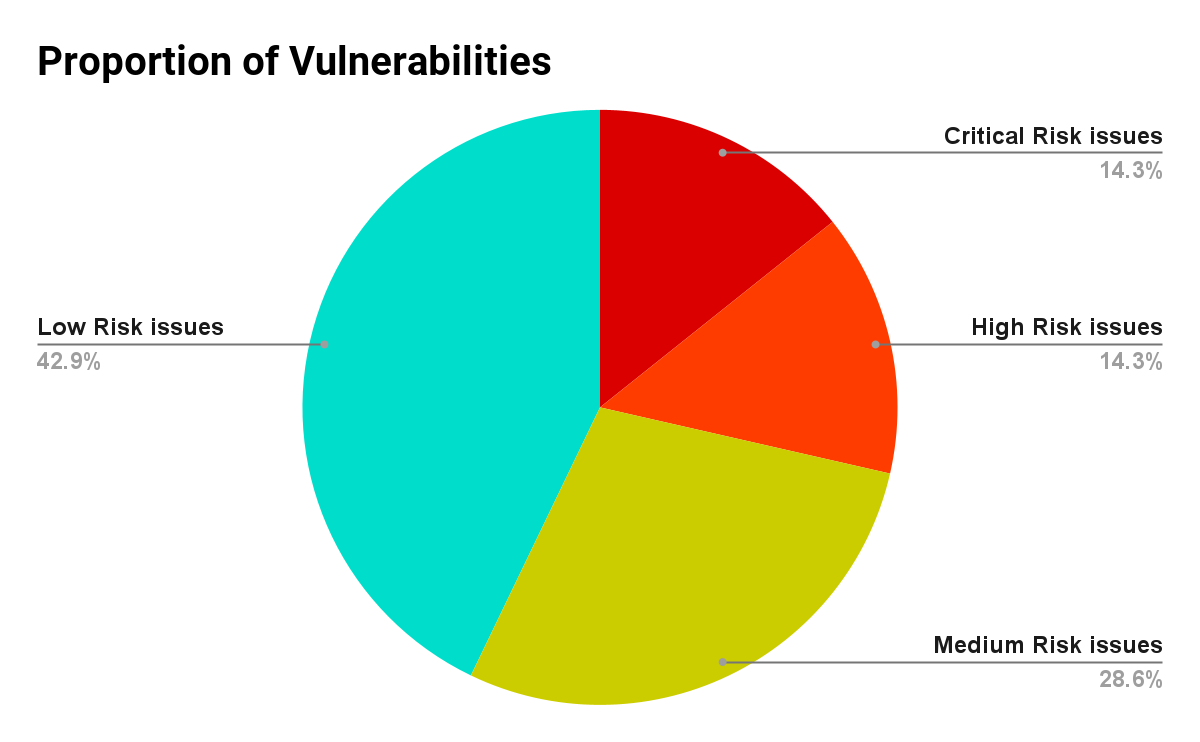
The codebase was audited in an incremental way. Fixes were applied on the way and were re-audited. We used a static analysis tool ([slither](https://github.com/crytic/slither)) which indicated the reentrancy bug in the code. We did manual reviews on the code to find logical bugs and the bugs reported by the automated tools.

# AUDIT REPORT

## Executive Summary

The analysis indicates that some of the functionalities in the contracts audited are **working properly**.

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 |
| --- | --- |
| 1 | Critical Risk issue(s) |
| 1 | High Risk issue(s) |
| 2 | Medium Risk issue(s) |
| 3 | Low Risk issue(s) |
| 0 | Informatory issue(s) |

## 

## Findings

### Critical-risk issues

1. **DOS in deposit**

**Description**The smart contract maintains a global reward per LP share as the pool's *globalReward*. It is calculated as the following formula:

globalReward = FullMath.mulDiv(blockDifference.mul(temp), 1e18,poolInfo[pool].totalLockedLiquidity).add(\_globalReward);

When the last user in the pool withdraws his Unipilot NFT, *totalLockedLiquidity* is  
set to 0. When the next user tries to deposit in the same pool, the contract throws an error (div by 0).

**Remedy:**This edge case should be handled.

**Dev’s Response:**They acknowledged and fixed by resetting the pool’s variables when the last users removes NFT

**Status:**Fixed and Verified

### High-risk issues

1. **Reentrancy when distributing Alt token Rewards**

**Description:**Unipilot also helps other tokens (specially new tokens) to gain traction by rewarding LPs with this Alt token. So the LPs will be rewarded $PILOT and $ALT\_TOKEN. Now when the user claims his reward, Unipilot transfers the shares of $ALT\_TOKEN but updates the user reward debt after transferring. Now if the $ALT\_TOKEN is ERC777, reentrancy is possible. Although it uses IERC20, but it will work with ERC777 tokens

IERC20(poolAltState.altToken).safeTransfer(userInfo[\_tokenId].user, altReward);  
poolAltState.lastRewardBlock = block.number;

**Remedy:**Update all state variables before transfer. And also use ReentrancyGuard.

**Dev’s Response:**We already have used *nonReentrant* modifier in the new commit

**Status**Fixed and verified

**Note:**  
It is good to communicate the changes earlier, so the auditor know beforehand the commit he is auditing

### Medium-risk issues

1. **Wrong assumption in require statement**

**Description:**In *depositNft* the smart contract assumes that the user shares of liquidity should be less than *totalLiquidity* in the pool and shares of liquidity should be greater than zero. But in the following line it uses ***or(* || *)*** instead of ***and (&&)*** operator

require(totalLiquidity >= liquidity || liquidity > 0, "IL");

**Remedy:**AND (&&) should be used.

**Dev’s Response**Acknowledged

**Status:**Not Fixed yet

1. **Unfair Reward distribution when *pilotPerBlock* is changed**

**Description:**There is a global variable *pilotPerBlock* which is used to calculate the $PILOT reward and act as a multiplier. This variable can be updated via governance. But When this variable is updated, it multiplies with the whole duration of staking.  
  
**Remedy:**Implement a mechanism like updating the rewardMultiplier of the pool by looping on all pools. Or just remove *pilotPerBlock* and just use *rewardMultiplier* to adjust pool rewards

**Dev’s Response:**Acknowledged and will loop on all pools

**Status:** Fixed and Verified

### Low-risk issues

1. Missing Event in *migrateFunds*
2. Wrong event in *updateUnipilot*
3. Remove unused imports like *LiquidityAmount.sol*

### Informatory issues and Optimization

No 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.