Smart Contract Security Audit Report

Contents

1. Executive Summary

On Mar. 23, 2021, the SlowMist security team received the DePlutus team's security audit application for DePlutus Protocol, developed the audit plan according to the agreement of both parties and the characteristics of the project, and finally issued the security audit report.

The SlowMist security team adopts the strategy of "white box lead, black, grey box assists" to conduct a complete security test on the project in the way closest to the real attack.

SlowMist Smart Contract DeFi project test method:

Black box testing	Conduct security tests from an attacker's perspective externally.
Grey box testing	Conduct security testing on code module through the scripting tool, observing the internal running status, mining weaknesses.
White box testing	Based on the open source code, non-open source code, to detect whether there are vulnerabilities in programs such as nodes, SDK, etc.

SlowMist Smart Contract DeFi project risk level:

Critical vulnerabilities	Critical vulnerabilities will have a significant impact on the security of the DeFi project, and it is strongly recommended to fix the critical vulnerabilities.
High-risk vulnerabilities	High-risk vulnerabilities will affect the normal operation of DeFi project. It is strongly recommended to fix high-risk vulnerabilities.
Medium-risk vulnerabilities	Medium vulnerability will affect the operation of DeFi project. It is recommended to fix medium-risk vulnerabilities.

Low-risk vulnerabilities	Low-risk vulnerabilities may affect the operation of DeFi project in certain scenarios. It is suggested that the project party should evaluate and consider whether these vulnerabilities need to be fixed.
Weaknesses	There are safety risks theoretically, but it is extremely difficult to reproduce in engineering.
Enhancement Suggestions	There are better practices for coding or architecture.

2. Audit Methodology

Our security audit process for smart contract includes two steps:

- Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using public and in-house automated analysis tools.
- Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.

Following is the list of commonly known vulnerabilities that was considered during the audit of the smart contract:

- Reentrancy attack and other Race Conditions
- Replay attack
- Reordering attack
- Short address attack
- Denial of service attack
- Transaction Ordering Dependence attack
- Conditional Completion attack
- Authority Control attack
- Integer Overflow and Underflow attack

- TimeStamp Dependence attack
- Gas Usage, Gas Limit and Loops
- Redundant fallback function
- Unsafe type Inference
- Explicit visibility of functions state variables
- Logic Flaws
- Uninitialized Storage Pointers
- Floating Points and Numerical Precision
- tx.origin Authentication
- "False top-up" Vulnerability
- Scoping and Declarations

3. Project Background

3.1 Project Introduction

DePlutus protocol creates more liquidity and fairer trading opportunities by connecting professional investors with ordinary investors.

Helping professional investors: create more asset portfolios with their own ETFs that can be tracked and managed at the atomic level through the smart contract-based AssetPool.

Helping the average investor: to rationalize leverage and gain more liquidity and investment opportunities by following investments and making expected return asset compositions.

Audit version file information

Initial audit files:

SHA256(contracts-2021-03-23.zip)=

6e67c81dcbbeef1f6f2a6006545f59e6c4211415258356 8c079e2d35465bdf59

Final audit files:

SHA256(contracts-2021-04-09.zip)=

fda51d621c8d16374d695f6a8475357213cfeadc8e5121af90cd8823770a5258

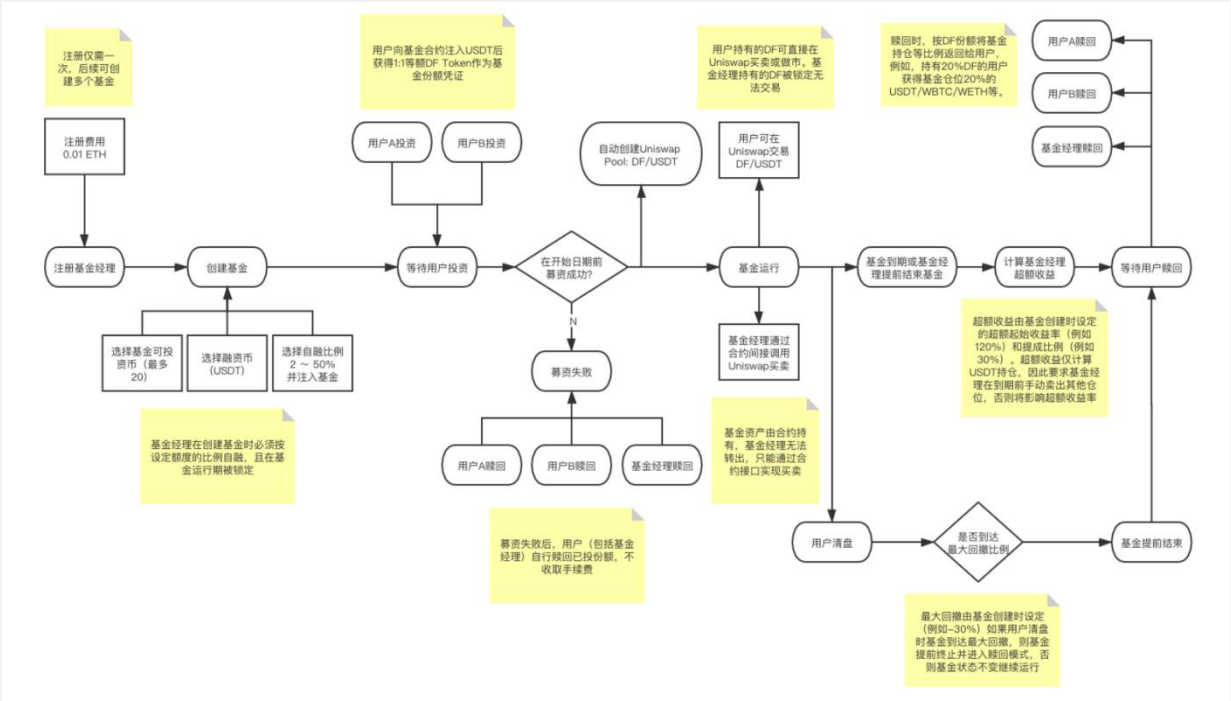
3.2 Project Structure

```
.
├── Base.sol
├── Config.sol
├── ERC20.sol
├── Fund.sol
├── FundManager.sol
├── Lock.sol
├── NameRegistry.sol
├── Ownable.sol
├── Stake.sol
├── StakeManager.sol
├── console.sol
├── interfaces
│   ├── IConfig.sol
│   ├── IERC20.sol
│   ├── IExchange.sol
│   ├── IFund.sol
│   ├── IFundManager.sol
│   ├── IStake.sol
│   ├── IStakeManager.sol
│   ├── IUniswapV2Factory.sol
│   ├── IUniswapV2Pair.sol
│   ├── IUniswapV2Router02.sol
│   └── IWETH.sol
└── libs
    ├── SafeMath.sol
    ├── TransferHelper.sol
    └── UniswapV2Library.sol
```

3.3 Contract Structure

The DePlutus Protocol project is mainly divided into 5 parts, namely fund manager registration

contract, fund creation contract, fund asset contract, pledge pool creation contract, and pledge pool contract. Users can register to become fund managers, release wealth management products, and earn income by investing on uniswap after financing. The business process is as follows:



4. Code Overview

4.1 Main Contract address

The contract has been deployed on the mainnet:

[pending]

4.2 Contracts Description

The SlowMist Security team analyzed the visibility of major contracts during the audit, the result as follows:

Fund			
Function Name	Visibility	Mutability	Modifiers
initialize	external	Can modify state	lock()
invest	external	Can modify state	ready() lock() inRaise() onlyController()
redeem	external	Can modify state	ready() lock()
swapExactTokensForTokens	external	Can modify state	inRun() ready() onlyManager()
swapTokensForExactTokens	external	Can modify state	inRun() ready() onlyManager()
transfer	public	Can modify state	-
approve	public	Can modify state	-
transferFrom	public	Can modify state	-
increaseAllowance	public	Can modify state	-
decreaseAllowance	public	Can modify state	-

FundManager			
Function Name	Visibility	Mutability	Modifiers
broadcast	external	Can modify state	-
createFund	external	Can modify state	-
invest	external	Can modify state	-
updateConfig	external	Can modify state	onlyCEO()
updateLib	external	Can modify state	onlyCEO()

NameRegistry			
Function Name	Visibility	Mutability	Modifiers
register	external	Can modify state	payable
setUrl	external	Can modify state	-
transfer	external	Can modify state	onlyCEO()
updateFee	external	Can modify state	onlyCEO()
updateConfig	external	Can modify state	onlyCEO()

Stake			
Function Name	Visibility	Mutability	Modifiers
initialize	external	Can modify state	-
stake	external	Can modify state	enable() lock() updateReward(msg.sender)
unstake	external	Can modify state	enable() lock() updateReward(msg.sender)
claim	external	Can modify state	enable() lock() updateReward(msg.sender)
quit	external	Can modify state	enable() lock() updateReward(msg.sender)
claim0	external	Can modify state	onlyController() enable() updateReward(msg.sender)

StakeManager			
Function Name	Visibility	Mutability	Modifiers
updateLib	external	Can modify state	onlyCEO()
claims	external	Can modify state	-
createStake	external	Can modify state	onlyCEO()
updateConfig	external	Can modify state	onlyCEO()

4.3 Code Audit

4.3.1 Low-risk vulnerabilities

4.3.1.1 Fund manager registration lacks access mechanism

There is no review process for fund manager registration, and there is no ability to prevent unreasonable fund parameters or other fraudulent activities.

Fix status: Ignored

4.3.1.2 gastoken attack

transfer() If _to is a malicious contract, it can trigger a gastoken attack, causing the CEO to lose a lot of gas. It is recommended to modify the call to transfer, and pay attention to the gas limit when calling

Code location: NameRegistry.sol

```
function transfer(address payable _to) external onlyCEO() {
require(_to != address(0), "Zero_Address");
require(!locker, "locked");
locker = true;
(bool sent, ) = _to.call{value: address(this).balance}("");
require(sent, "transfer failure");
locker = false;
}
```

Fix status: It has been modified to call in `transfer` mode.

4.3.2 Enhancement suggestions

4.3.2.1 `ready()` Code redundancy

Code location: FundManager.sol

```
modifier ready() {
require(lib != address(0), "lib is unset");
_;
}
```

It is recommended to delete redundant codes.

4.3.2.2 `broadcast()` Code redundancy

Code location: FundManager.sol

```
function broadcast() external {
emit FundChanged(msg.sender);
}
```

It is recommended to delete redundant codes.

4.3.3 Business risk reminder

4.3.3.1 The fund is at risk of loss

In extreme cases, the fund manager may have a `Rat Trading` and may cause the fund to lose money. Therefore, when investing funds, users should review the fund manager's self-investment ratio and the whitelist of investment currencies to make them consistent with the interests of investors, otherwise it may cause capital losses at the maximum retracement line.

5. Audit Result

5.1 Conclusion

Audit Result : Low Risk

Audit Number : 0X002103310004

Audit Date : Mar. 31, 2021

Audit Team : SlowMist Security Team

Summary conclusion: The SlowMist security team use a manual and SlowMist Team analysis tool

audit of the codes for security issues. There are 4 security issues found during the audit. There are 2 low-risk vulnerabilities, we also provide 2 enhancement suggestions. In extreme cases, the fund manager may have a `Rat Trading` and may cause the fund to lose money. Therefore, when investing funds, users should review the fund manager's self-investment ratio and the whitelist of investment currencies to make them consistent with the interests of investors, otherwise it may cause capital losses at the maximum retracement line.

6. Statement

SlowMist issues this report with reference to the facts that have occurred or existed before the issuance of this report, and only assumes corresponding responsibility base on these.

For the facts that occurred or existed after the issuance, SlowMist is not able to judge the security status of this project, and is not responsible for them. The security audit analysis and other contents of this report are based on the documents and materials provided to SlowMist by the information provider till the date of the insurance this report (referred to as "provided information"). SlowMist assumes: The information provided is not missing, tampered with, deleted or concealed. If the information provided is missing, tampered with, deleted, concealed, or inconsistent with the actual situation, the SlowMist shall not be liable for any loss or adverse effect resulting therefrom. SlowMist only conducts the agreed security audit on the security situation of the project and issues this report. SlowMist is not responsible for the background and other conditions of the project.

Official Website
www.slowmist.com

E-mail
team@slowmist.com

Twitter
@SlowMist_Team

Github
https://github.com/slowmist