Smart Contract Security Audit Report

Table Of Contents

1 Executive Summary

On 2024.12.16, the SlowMist security team received the Avalon Financa team's security audit application for USDa Pool, 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.

The test method information:

Test method	Description
Black box testing	Conduct security tests from an attacker's perspective externally.
Grey box testing	Conduct security testing on code modules 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.

The vulnerability severity level information:

Level	Description
Critical	Critical severity vulnerabilities will have a significant impact on the security of the DeFi project, and it is strongly recommended to fix the critical vulnerabilities.
High	High severity vulnerabilities will affect the normal operation of the DeFi project. It is strongly recommended to fix high-risk vulnerabilities.
Medium	Medium severity vulnerability will affect the operation of the DeFi project. It is recommended to fix medium-risk vulnerabilities.
Low	Low severity vulnerabilities may affect the operation of the DeFi project in certain scenarios. It is suggested that the project team should evaluate and consider whether these vulnerabilities need to be fixed.
Weakness	There are safety risks theoretically, but it is extremely difficult to reproduce in engineering.
Suggestion	There are better practices for coding or architecture.

2 Audit Methodology

The security audit process of SlowMist security team for smart contract includes two steps:

- Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using 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:

Serial Number	Audit Class	Audit Subclass
1	Overflow Audit	-
2	Reentrancy Attack Audit	-
3	Replay Attack Audit	-
4	Flashloan Attack Audit	-
5	Race Conditions Audit	Reordering Attack Audit
6	Permission Vulnerability Audit	Access Control Audit
		Excessive Authority Audit
7	Security Design Audit	External Module Safe Use Audit
		Compiler Version Security Audit
		Hard-coded Address Security Audit
		Fallback Function Safe Use Audit
		Show Coding Security Audit
		Function Return Value Security Audit
		External Call Function Security Audit

Serial Number	Audit Class	Audit Subclass
7	Security Design Audit	Block data Dependence Security Audit
		tx.origin Authentication Security Audit
8	Denial of Service Audit	-
9	Gas Optimization Audit	-
10	Design Logic Audit	-
11	Variable Coverage Vulnerability Audit	-
12	"False Top-up" Vulnerability Audit	-
13	Scoping and Declarations Audit	-
14	Malicious Event Log Audit	-
15	Arithmetic Accuracy Deviation Audit	-
16	Uninitialized Storage Pointer Audit	-

3 Project Overview

3.1 Project Introduction

This is an asset swap agreement.

3.2 Vulnerability Information

The following is the status of the vulnerabilities found in this audit:

NO	Title	Category	Level	Status
N1	Preemptive Initialization	Race Conditions Vulnerability	Suggestion	Acknowledged
N2	Risk of excessive authority	Authority Control Vulnerability Audit	Medium	Acknowledged

NO	Title	Category	Level	Status
N3	There is a risk of denial of service for getPendingTransactionsCount	Denial of Service Vulnerability	Low	Fixed

4 Code Overview

4.1 Contracts Description

https://github.com/avalonfinancexyz/USDAPool/tree/V2.0

Audit Commit: 090d00e2e22e6d6a996f27ab6e00db7a31f1d1a8

Review Commit: 9032a3ae93cccf3906702875cc8956aa4171c69a

The main network address of the contract is as follows:

The code was not deployed to the mainnet.

4.2 Visibility Description

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

USDaPool			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
initialize	Public	Can Modify State	initializer
convertToUSDa	Public	-	-
convertToUSDT	Public	-	-
depositUSDa	External	Can Modify State	whenNotPaused onlyWhitelisted
claimUSDT	External	Can Modify State	whenNotPaused onlyWhitelisted
repayUSDT	External	Can Modify State	whenNotPaused onlyWhitelisted

USDaPool			
depositUSDT	External	Can Modify State	onlyAA whenNotPaused
setAA	External	Can Modify State	onlyRole
addWhitelist	External	Can Modify State	onlyRole
managerRepayUSDT	External	Can Modify State	onlyRole whenNotPaused
getPendingTransactionsCount	Public	-	-
pause	External	Can Modify State	onlyRole
unpause	External	Can Modify State	onlyRole
getUserInfo	External	-	-
setCap	External	Can Modify State	onlyRole
rescueUSDT	External	Can Modify State	onlyRole

4.3 Vulnerability Summary

[N1] [Suggestion] Preemptive Initialization

Category: Race Conditions Vulnerability

Content

By calling the initialize function to initialize the contract, there is a potential issue that malicious attackers preemptively call the initialize function to initialize.

- src/USDaPool.sol

```
function initialize(address _AA) public initializer {
        __AccessControl_init();
        __Pausable_init();

        _setRoleAdmin(MANAGER_ROLE, DEFAULT_ADMIN_ROLE);
        _setRoleAdmin(PAUSE_ROLE, DEFAULT_ADMIN_ROLE);

        _grantRole(DEFAULT_ADMIN_ROLE, msg.sender);
```

```
        _grantRole(MANAGER_ROLE, msg.sender);
        _grantRole(PAUSE_ROLE, msg.sender);

        AA = _AA;
    }
```

Solution

It is suggested that the initialization operation can be called in the same transaction immediately after the contract is created to avoid being maliciously called by the attacker.

Status

Acknowledged

[N2] [Medium] Risk of excessive authority

Category: Authority Control Vulnerability Audit

Content

DEFAULT_ADMIN_ROLE has the authority to set AA and rescue USDT, MANAGER_ROLE can manage whitelists and USDT repayment, and PAUSE_ROLE can control contract pause and unpause functions. If the private keys of these roles are compromised, it would pose serious security risks.

```
DEFAULT_ADMIN_ROLE can setAA
DEFAULT_ADMIN_ROLE can rescueUSDT
MANAGER_ROLE can addWhitelist
MANAGER_ROLE can managerRepayUSDT
PAUSE_ROLE can pause
PAUSE_ROLE can unpause
```

Solution

In the short term, during the early stages of the project, the protocol may need to frequently set various parameters to ensure the stable operation of the protocol. Therefore, transferring the ownership of core roles to a multisig management can effectively solve the single-point risk, but it cannot mitigate the excessive privilege risk. In the long run, after the protocol stabilizes, transferring the owner ownership to community governance and executing through a timelock can effectively mitigate the excessive privilege risk and increase the community users' trust in the protocol.

Status

Acknowledged; Will use multi-signature to manage the contract.

[N3] [Low] There is a risk of denial of service for getPendingTransactionsCount

Category: Denial of Service Vulnerability

Content

Both repayUSDT and managerRepayUSDT will use getPendingTransactionsCount(user) to determine if repayment is possible. When the number of orders is large, there may be situations where repayUSDT and managerRepayUSDT cannot be executed.

- src/USDaPool.sol

```
function getPendingTransactionsCount(address user) public view returns (uint256
count) {
        Transaction[] storage txns = userInfo[user].pendingTxns;
        for (uint i = 0; i < txns.length; i++) {
            if (!txns[i].finished) count++;
        }
    }
```

Solution

To optimize gas consumption can maintain a separate counter for completed transactions instead of iterating through the entire transaction list. This approach would significantly reduce gas costs by replacing array iteration with a simple counter lookup.

Status

Fixed

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002412170002	SlowMist Security Team	2024.12.16 - 2024.12.17	Medium Risk

Summary conclusion: The SlowMist security team use a manual and SlowMist team's analysis tool to audit the project, during the audit work we found 1 medium risk, 1 low risk, 1 suggestion vulnerabilities.

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 based 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 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