Smart Contract Security Audit Report

Table Of Contents

1 Executive Summary

2 Audit Methodology

3 Project Overview

3.1 Project Introduction

3.2 Vulnerability Information

4 Code Overview

4.1 Contracts Description

4.2 Visibility Description

4.3 Vulnerability Summary

5 Audit Result

6 Statement

1 Executive Summary

On 2025.04.28, the SlowMist security team received the DeSyn Protocol team's security audit application for DeSyn Phase9, 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 iterative audit of the DeSyn protocol, focusing mainly on the newly added Airdrop module and the ETF whitelist module. The Airdrop module allows the admin to execute designated airdrop claims via the ETF, while the ETF whitelist module enables the owner to add trusted external modules.

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Whitelist modules can only be added but not removed	Design Logic Audit	Suggestion	Acknowledged
N2	Admin can perform any operations on ETF through external modules	Others	Information	Acknowledged

4 Code Overview

4.1 Contracts Description

Audit Version:

https://github.com/Meta-DesynLab/desyn-contracts-fork/tree/white-module

commit: b7829a29b24165272d7bcbe4f2fe2284a30c69cb

https://github.com/Meta-DesynLab/desyn-modules-forge/blob/feature/module-adapt/src/move/MoveFunds.sol

commit: e25afab2ad425e00ed563e91d83f9c837f6ac75d

https://github.com/Meta-DesynLab/desyn-modules-forge/blob/feature/module-adapt/src/airdrop/Airdrop.sol

commit: e25afab2ad425e00ed563e91d83f9c837f6ac75d

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:

MoveFunds			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-

MoveFunds			
makeTransfer	External	Can Modify State	-
makeTransferPart	Public	Can Modify State	-
setReceiver	External	Can Modify State	-
_checkTx	Internal	-	-

Airdrop			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
claimAirdrop	External	Can Modify State	-
setReceiver	External	Can Modify State	-
_checkTx	Internal	-	-

ConfigurableRightsPool			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	PCToken
init	Public	Can Modify State	-
setCap	External	Can Modify State	logs lock needsBPool onlyOwner
execute	External	Can Modify State	logs lock needsBPool
addWhiteModule	External	Can Modify State	onlyOwner
couldClaimManagerFee	Public	-	-
claimManagerFee	Public	Can Modify State	logs lock onlyAdmin needsBPool
_claimManagerFee	Internal	Can Modify State	-

ConfigurableRightsPool			
createPool	External	Can Modify State	onlyOwner logs lock notPaused
joinPool	External	Can Modify State	logs lock needsBPool notPaused
exitPool	External	Can Modify State	logs lock needsBPool notPaused
whitelistLiquidityProvider	External	Can Modify State	onlyOwner lock logs
removeWhitelistedLiquidityProvider	External	Can Modify State	onlyOwner lock logs
canProvideLiquidity	Public	-	-
hasPermission	External	-	-
getRightsManagerVersion	External	-	-
getDesynSafeMathVersion	External	-	-
getSmartPoolManagerVersion	External	-	-
createPoolInternal	Internal	Can Modify State	-
addTokenToWhitelist	External	Can Modify State	onlyOwner
_verifyWhiteToken	Public	-	-
_pullUnderlying	Internal	Can Modify State	needsBPool
_pushUnderlying	Internal	Can Modify State	needsBPool
_mint	Internal	Can Modify State	-
_mintPoolShare	Internal	Can Modify State	-
_pushPoolShare	Internal	Can Modify State	-
_pullPoolShare	Internal	Can Modify State	-

ConfigurableRightsPool			
_burnPoolShare	Internal	Can Modify State	-
snapshotBeginAssets	External	Can Modify State	logs
beginFundAssets	External	-	-
endFundAssets	External	-	-
snapshotEndAssets	Public	Can Modify State	logs
snapshotAssets	Public	Can Modify State	-
_getPoolTokensInfo	Internal	-	-

4.3 Vulnerability Summary

[N1] [Suggestion] Whitelist modules can only be added but not removed

Category: Design Logic Audit

Content

In the ConfigurableRightsPool contract, the owner can add trusted external modules to the ETF via the addWhiteModule function. The ETF allows these external modules to call the execute function to perform arbitrary operations. It is important to note that once these external modules are added, they cannot be removed. If any of the external modules present risks, the ETF may be affected.

Code location: contracts/base/ConfigurableRightsPool.sol#L182

```
    function addWhiteModule(address[] calldata modules) external onlyOwner {
        require(whiteModule.limitMoudulesCount > 0, "ERR_NO_LIMIT");
        require(whiteModule.hasAddMoudulesCount < whiteModule.limitMoudulesCount,
"ERR_LIMIT_REACHED");
        for (uint i = 0; i < modules.length; i++) {
            address module = modules[i];
            require(!whiteModule.isWhiteMoudules[module],
"ERR_MODULE_ALREADY_WHITE");
            whiteModule.isWhiteMoudules[module] = true;
```

```
            emit WhiteModuleUpate(module, true);
        }
        whiteModule.hasAddMoudulesCount++;
    }
```

Solution

It is recommended to add a feature that allows the removal of external modules to mitigate the above risk.

Status

Acknowledged; After communicating with the project team, they stated that this is an intentional design choice: once external modules are added, they cannot be removed.

[N2] [Information] Admin can perform any operations on ETF through external modules

Category: Others

Content

In the Airdrop contract, the admin role of the ETF can call the execute function of the ETF via the claimAirdrop function to perform external airdrop claiming operations. However, it should be noted that both the target address and the call data for the ETF execution are provided by the admin. In theory, this allows the admin to perform arbitrary operations on the ETF, including the transfer of funds.

Code location: src/airdrop/Airdrop.sol#L33

```
    function claimAirdrop(address _etf, address claimAddr, bytes memory claimData,
address _token) external {
        _checkTx(_etf);

        ...

        IETF(_etf).invokeClaimAirdrop(claimAddr, claimData);
        ...
    }
```

Solution

It is recommended that the ETF's admin verify whether the data is as expected before execution, and it is also advised that the admin role be managed by a multi-signature wallet to avoid single point of failure risks.

Status

Acknowledged

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002504210001	SlowMist Security Team	2025.04.28 - 2025.04.29	Passed

Summary conclusion: The SlowMist security team uses a manual and SlowMist team's analysis tool to audit the project, during the audit work we found 1 suggestion and 1 information. All the findings were acknowledged. The code was not deployed to the mainnet.

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