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 2024.07.31, the SlowMist security team received the DeSyn Protocol team's security audit application for DeSyn Phase8, 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

DeSyn is a web3 asset management platform that provides a decentralized asset management infrastructure for everyone around the world. This audit specifically focuses on the modifications made to the DeSyn protocol for its deployment on the zkLinkNova and Scroll networks. Users should be aware of the risks outlined in the Phase 5 report before reading this report.

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Optimizable bytecode concatenation	Gas Optimization Audit	Suggestion	Acknowledged

4 Code Overview

4.1 Contracts Description

Audit Version:

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

commit: d56bb5152a2e0bd88ac35adf7a166e6e5bbd47a0

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

commit: 63a094130ce21029897df2dbc359d9bf2d0aa0b5

The main network address of the contract is as follows:

Contract Name	Contract Address	Chain
Actions	0x96c7C102e18FC298536171277BBBCE93e00663E3	Scroll
Vault	0x301Be34Da27088f2a81F344904c5384F212b132d	Scroll
UserVault	0xb5068dA710D6Ba6D79a9E6Fd8a9e80b1bFdf9164	Scroll
Oracle	0x0B3D68F0646D0AFB2CE625B146eB99FE941ba8BC	Scroll
DesynChainlinkOracle	0x6AF58b55B4eec887Ca39946842Fb463e9Fb25Ed4	Scroll
Factory	0x09eFC8C8F08B810F1F76B0c926D6dCeb37409665	Scroll
DesynSafeMath	0xdE6b117384452b21F5a643E56952593B88110e78	Scroll
RightsManager	0x5C3027D8Cb28A712413553206A094213337E88c5	Scroll
SmartPoolManager	0x770c9d0851b21df8A84943EdE4f487D30d9741ba	Scroll
CRPFactory	0xe788511225632ffdA2c532d65ede98aF047282e8	Scroll

4.2 Visibility Description

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

CRPFactory			
Function Name	Visibility	Mutability	Modifiers
createPool	Internal	Can Modify State	-
newCrp	External	Can Modify State	-
setUserVault	External	Can Modify State	onlyBlabs
setByteCodes	External	Can Modify State	onlyBlabs _logs_
setBLabs	External	Can Modify State	onlyBlabs
concatenate	Internal	-	-
isCrp	External	-	-
addCRPFactory	External	Can Modify State	onlyBlabs
removeCRPFactory	External	Can Modify State	onlyBlabs

Factory			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
addTokenToWhitelist	External	Can Modify State	onlyBlabs
removeTokenFromWhitelist	External	Can Modify State	onlyBlabs
isTokenWhitelistedForVerify	External	-	-
isTokenWhitelistedForVerify	External	-	-
isLiquidityPool	External	-	-
createPool	Internal	Can Modify State	-

Factory

newLiquidityPool	External	Can Modify State	-
getBLabs	External	-	-
setBLabs	External	Can Modify State	onlyBlabs
getModuleStatus	External	-	-
getOracleAddress	External	-	-
setSystemModule	External	Can Modify State	onlyBlabs
registerModule	External	Can Modify State	onlyBlabs
removeModule	External	Can Modify State	onlyBlabs
setOracle	External	Can Modify State	onlyBlabs
collect	External	Can Modify State	onlyBlabs
getVault	External	-	-
setVault	External	Can Modify State	onlyBlabs
getUserVault	External	-	-
setUserVault	External	Can Modify State	onlyBlabs
setProtocolPaused	External	Can Modify State	onlyBlabs
setByteCodes	External	Can Modify State	onlyBlabs
concatenate	Internal	-	-

ConfigurableRightsPool

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

ConfigurableRightsPool			
setCap	External	Can Modify State	logs lock needsBPool onlyOwner
execute	External	Can Modify State	logs lock needsBPool
couldClaimManagerFee	Public	-	-
claimManagerFee	Public	Can Modify State	logs lock onlyAdmin needsBPool
_claimManagerFee	Internal	Can Modify State	-
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	-	-
mintPoolShareFromLib	Public	Can Modify State	-
pushPoolShareFromLib	Public	Can Modify State	-
pullPoolShareFromLib	Public	Can Modify State	-
burnPoolShareFromLib	Public	Can Modify State	-

ConfigurableRightsPool

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	-
_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	-	-

LiquidityPool

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

LiquidityPool			
isPublicSwap	External	-	-
isFinalized	External	-	-
isBound	External	-	-
getNumTokens	External	-	-
getCurrentTokens	External	-	_viewlock_
getFinalTokens	External	-	_viewlock_
getDenormalizedWeight	External	-	_viewlock_
getTotalDenormalizedWeight	External	-	_viewlock_
getNormalizedWeight	External	Can Modify State	_viewlock_
getBalance	Public	-	_viewlock_
getController	External	-	_viewlock_
setController	External	Can Modify State	_logs_ _lock_
setPublicSwap	External	Can Modify State	_logs_ _lock_
finalize	External	Can Modify State	_logs_ _lock_
bind	External	Can Modify State	_logs_
rebind	Public	Can Modify State	_logs_ _lock_
execute	External	Can Modify State	_logs_ _lock_
unbind	External	Can Modify State	_logs_ _lock_
unbindPure	External	Can Modify State	_logs_ _lock_
rebindPure	Public	Can Modify State	_logs_ _lock_
gulp	External	Can Modify State	_logs_ _lock_
joinPool	External	Can Modify State	_logs_ _lock_

LiquidityPool			
exitPool	External	Can Modify State	_logs_ _lock_
_pullUnderlying	Internal	Can Modify State	-
_pushUnderlying	Internal	Can Modify State	-
_pullPoolShare	Internal	Can Modify State	-
_pushPoolShare	Internal	Can Modify State	-
_mintPoolShare	Internal	Can Modify State	-
_burnPoolShare	Internal	Can Modify State	-
<Receive Ether>	External	Payable	-

4.3 Vulnerability Summary

[N1] [Suggestion] Optimizable bytecode concatenation

Category: Gas Optimization Audit

Content

In the setByteCodes function of the Factory and CRPFactory contracts on the Scroll chain, due to the block gasLimit restriction, it is not possible to write the complete contract bytecode into bytecodes in a single transaction. Therefore, the bytecode is concatenated using the concatenate function. The concatenate function uses a for loop to copy and concatenate the bytecode, which consumes a large amount of gas compared to using calldatacopy.

Code location:

contracts/deploy/CRPFactory.sol#L157-L196

contracts/deploy/Factory.sol#L188-L227

```
    function concatenate(bytes memory bytecode1, bytes memory bytecode2) internal
pure returns (bytes memory) {
        bytes memory concatenated;

        assembly {
            ...
            for { let i := 0 } lt(i, length1) { i := add(i, 0x20) } {
                mstore(add(memPtr, i), mload(add(add(bytecode1, 0x20), i)))
            }

            memPtr := add(memPtr, length1)

            for { let i := 0 } lt(i, length2) { i := add(i, 0x20) } {
                mstore(add(memPtr, i), mload(add(add(bytecode2, 0x20), i)))
            }

            ...
        }

        return concatenated;
    }
```

Solution

It is recommended to use calldatacopy instead of a for loop for bytecode concatenation to save gas.

Status

Acknowledged

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002407310002	SlowMist Security Team	2024.07.31 - 2024.07.31	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. All the finding was acknowledged. The code was not deployed to the mainnet. The risks identified during the comprehensive audit of this protocol have been presented in the reports of other audit phases. Users should thoroughly read all the reports to understand the overall risks of the protocol fully.

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.

SLOWMIST

Official Website

www.slowmist.com

E-mail

team@slowmist.com

Twitter

@SlowMist_Team

Github

https://github.com/slowmist