Smart Contract
Security Audit Report

Table Of Contents

1 Executive Summary

On 2022.02.28, the SlowMist security team received the Unifi team's security audit application for Unifi 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.

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.

Level	Description
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:

- Reentrancy Vulnerability
- Replay Vulnerability
- Reordering Vulnerability
- Short Address Vulnerability
- Denial of Service Vulnerability
- Transaction Ordering Dependence Vulnerability
- Race Conditions Vulnerability
- Authority Control Vulnerability
- Integer Overflow and Underflow Vulnerability
- TimeStamp Dependence Vulnerability
- Uninitialized Storage Pointers Vulnerability
- Arithmetic Accuracy Deviation Vulnerability
- tx.origin Authentication Vulnerability

- "False top-up" Vulnerability
- Variable Coverage Vulnerability
- Gas Optimization Audit
- Malicious Event Log Audit
- Redundant Fallback Function Audit
- Unsafe External Call Audit
- Explicit Visibility of Functions State Variables Audit
- Design Logic Audit
- Scoping and Declarations Audit

3 Project Overview

3.1 Project Introduction

Project:

Unifi Protocol - uBridge

Project address:

https://github.com/unifiprotocol/ubridge

commit: 353fe317f7aea2a71b7d7cdea278a569f8511455

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Deflationary tokens are not compatible	Others	Low	Ignored

NO	Title	Category	Level	Status
N2	Excessive authority issues	Authority Control Vulnerability	Suggestion	Ignored
N3	Excessive authority issues	Authority Control Vulnerability	Suggestion	Ignored

4 Code Overview

4.1 Contracts Description

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:

BridgeToken			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ERC20

ProxyBridge			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ERC1967Proxy
upgradeTo	Public	Can Modify State	ifAdmin
getImplementationAddress	Public	-	-

UBridge			
Function Name	Visibility	Mutability	Modifiers
init	Public	Can Modify State	initializer
addVerifyAddress	Public	Can Modify State	onlyOwner
removeVerifyAddress	Public	Can Modify State	onlyOwner
getVerifyAddresses	Public	-	-
addToken	Public	Can Modify State	onlyOwner
addChainId	Public	Can Modify State	onlyOwner
setChainIdFee	Public	Can Modify State	onlyOwner
removeToken	Public	Can Modify State	onlyOwner
removeChainId	Public	Can Modify State	onlyOwner
getChainIds	Public	-	-
getOriginAddresses	Public	-	-
pause	Public	Can Modify State	onlyOwner
unpause	Public	Can Modify State	onlyOwner
changeDepositsState	Public	Can Modify State	onlyOwner
deposit	Public	Payable	nonReentrant whenNotDepositsPaused whenNotPaused

UBridge			
claimFees	Public	Can Modify State	nonReentrant
withdraw	Public	Can Modify State	nonReentrant whenNotPaused
withdrawBatch	Public	Can Modify State	-
verify	Private	-	-

4.3 Vulnerability Summary

[N1] [Low] Deflationary tokens are not compatible

Category: Others

Content

- contracts/UBridge.sol

Deflationary tokens will have compatibility issues,If it is a deflationary token, the platform will receive less tokens than the actual number, but the recorded events are based on the number before deflation, which will lead to losses for the platform

```
function deposit(
    address withdrawalAddress,
    address originTokenAddress,
    uint256 amount,
    uint256 targetChainId
 ) public payable nonReentrant whenNotDepositsPaused whenNotPaused {
    require(msg.value == chainIdFees[targetChainId], "WRONG_FEE");
    require(withdrawalAddress != address(0), "WRONG_ADDRESS");
    require(chainIdSupported[targetChainId], "CHAIN_ID_NOT_SUPPORTED");
    address destinationTokenAddress = tokensSupported[originTokenAddress]
[targetChainId];
    require(destinationTokenAddress != address(0), "UNSUPPORTED_TOKEN_ON_CHAIN_ID");
    count += 1;
```

```
    //SlowMist// If it is a deflationary token, the platform will receive less tokens
than the actual
    IERC20(originTokenAddress).safeTransferFrom(msg.sender, address(this), amount);

    emit Deposit(
      withdrawalAddress,
      originTokenAddress,
      destinationTokenAddress,
      amount,
      chainId,
      targetChainId,
      count
    );
  }
```

Solution

You can check the difference between the balance before the transfer and the balance after the transfer, as the value for accounting.

Status

Ignored; Communicate with the project party: This bridge usage is for 2 Tokens that are not deflationary.

[N2] [Suggestion] Excessive authority issues

Category: Authority Control Vulnerability

Content

- contracts/UBridge.sol

The owner permission is too large, and the key parameter settings take effect immediately. If the owner's private key is lost, the attacker can add a new verifier by setting `addVerifyAddress`, and remove the old verifier through `removeVerifyAddress`. After that, the attacker can use the `withdraw` function Transfer the token of the contract away

Solution

It is recommended to set Owner address to timelock contract, governance contract, or multi-sign contract to reduce

the risk of private key loss.

Status

Ignored; The project party will using multi sig contracts as the owner.

[N3] [Suggestion] Excessive authority issues

Category: Authority Control Vulnerability

Content

- contracts/ProxyBridge.sol

owner can modify the logical contract pointed to by the proxy contract. If the private key is lost, an attacker can modify the logical contract pointed to by the proxy contract

Solution

It is recommended to set Owner address to timelock contract, governance contract, or multi-sign contract to reduce the risk of private key loss.

Status

Ignored; The project party will using multi sig contracts as the owner.

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002203020003	SlowMist Security Team	2022.02.28 - 2022.03.03	Passed

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 low risk, 2 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