Smart Contract

Security Audit Report

Table Of Contents

1 Executive Summary

On 2023.04.23, the SlowMist security team received the UXUY Protocol team's security audit application for UXUY Protocol Contracts, 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:

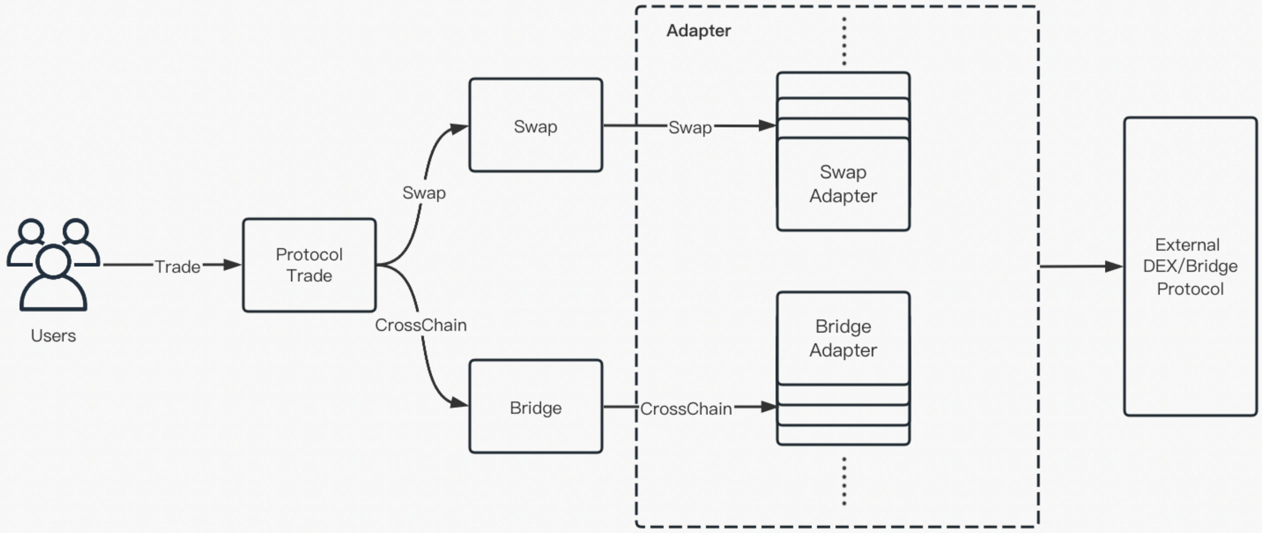
Serial Number	Audit Class	Audit Subclass
1	Overflow Audit	-
2	Reentrancy Attack Audit	-
3	Replay Attack Audit	-
4	Flashloan Attack Audit	-
5	Race Conditions Audit	Reordering Attack Audit
6	Permission Vulnerability Audit	Access Control Audit
		Excessive Authority Audit
7	Security Design Audit	External 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

UXUY Protocol is a cross-chain interoperability solution that empowers Web3 projects and users to seamlessly swap tokens across multiple chains. The UXUY protocol is mainly composed of three parts: Protocol, Swap/Bridge and Swap/Bridge Adapter. Users can only perform token swap and cross-chain operations through the Protocol contract. The Protocol contract will call Swap/Bridge according to the external incoming path to select the required Adapter contract for swap or cross-chain operations. The following is a brief architecture diagram:



3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Maximum approval issue	Design Logic Audit	Suggestion	Acknowledged
N2	Redundant payable tag	Others	Suggestion	Fixed
N3	Parameters are not strictly checked	Others	Suggestion	Fixed
N4	Bypass the fee charge for token swap	Design Logic Audit	Low	Fixed
N5	Risk of over-privilege	Authority Control Vulnerability Audit	Medium	Fixed
N6	The AnySwap router is not configured	Design Logic Audit	Low	Fixed
N7	The check of balanceBefore in swap is flawed	Design Logic Audit	Low	Fixed
N8	`try-catch` does not check the cause of the error	Others	Suggestion	Acknowledged

NO	Title	Category	Level	Status
N9	Curve pool selection issue	Design Logic Audit	Low	Fixed
N10	Curve pool index issue	Design Logic Audit	Low	Fixed
N11	Arbitrary function call issue	Design Logic Audit	Suggestion	Fixed
N12	Potential slippage decimal issue	Design Logic Audit	Suggestion	Fixed

4 Code Overview

4.1 Contracts Description

Audit Version:

https://github.com/uxuycom/uxuy-protocol-contracts

commit: 731ee66fc9d07eb91d27667def7d1e392fd84bd8

Fixed Version:

https://github.com/uxuycom/uxuy-protocol-contracts

commit: 73bca091e47aab0978137dfc82223bd675fa1097

The main network address of the contract is as follows:

Contract Name	Contract Address	Chain	Open Source or Not
UxuyProtocol	0x2f682320925a04387ed1F8Dd82f67FCF86DD4D0e	Ethereum	✓
UxuyBridge	0xEB4c5A9AB90dD1E2d8c70d8c7Bf77aC960db94AB	Ethereum	✓
UxuySwap	0x164B9A2564904c3eAe101315e3136290694604c3	Ethereum	✓
Timelock	0xe4EFB6e54755A3d570FEBc784aa4751fcB5e3ab9	Ethereum	✓

Contract Name	Contract Address	Chain	Open Source or Not
UniswapV3SwapAdapter	0xD3aED63Dc4184CE8B1Ac07B02FAc0e86A51aeef9	Ethereum	✓
UniswapV2SwapAdapter	0x9cF78431d335084e0D3c0A37dAeA1BD97ca048Ad	Ethereum	✓
UniswapV2SwapAdapter	0x0Da4c8052C2D35584A35f7754F6191575eb54d96	Ethereum	✓
CurveSwapAdapter	0x45031Ba922Ef7f9775186FC9cEe210fB6CBB31AE	Ethereum	✓
XYBridgeAdapter	0x5a93012824a8a47637165A55315252B8164A6c21	Ethereum	✓
AnySwapBridgeAdapter	0x70860C133221126650ca40BDfE80cFB29D1C974a	Ethereum	✓
UxuyProtocol	0x5e2899Be3DB7223eE87D061236410369A4bc68a3	Optimism	✓
UxuySwap	0x70860C133221126650ca40BDfE80cFB29D1C974a	Optimism	✓
UxuyBridge	0x137DFa803359229F8546ADF469eeD668786e5E28	Optimism	✓
Timelock	0xf15B31A7d7064905006CC765C427537F05530090	Optimism	✓
UniswapV3SwapAdapter	0x796aDf731aE9BE2f22C36b7F2C76741408B17595	Optimism	✓
CurveSwapAdapter	0x3019b29825A860D2B66281700d209a7303145dD6	Optimism	✓
XYBridgeAdapter	0x11165C01A9B203594018Dd6dfa7C58FD1082962c	Optimism	✓
AnySwapBridgeAdapter	0x03e6A4e5908CC91821E1A64E07c8b17F99DAE9A1	Optimism	✓
UxuyProtocol	0x3d8fd8EBC5530342B77E0172E7EF0627417a1CBA	Polygon	✓
UxuySwap	0x8a2728f15380cb90076a1a165fbd28848271d270	Polygon	✓
UxuyBridge	0x44F7cE718c0AC0088763AF2feb5488EE9896F7da	Polygon	✓

Contract Name	Contract Address	Chain	Open Source or Not
Timelock	0x83B4efeC5A51D69C66563Deba24a6377b906197e	Polygon	✓
UniswapV2SwapAdapter	0x7A483fAb45df29D26e69854771d04f23642F0aa5	Polygon	✓
UniswapV2SwapAdapter	0x84aDA12d9e4b7991bE3f127Bc7720f93F8473FD7	Polygon	✓
OneInchSwapAdapter	0xd068CCbbC0bc54984b1654777bEc7EC28a4aB8EA	Polygon	✓
UniswapV3SwapAdapter	0x4f34fd6de52373F8393C137EFc87a58Ed36bBfdD	Polygon	✓
CurveSwapAdapter	0xF8b92b1224C01afD64845809f5aFe59806dff571	Polygon	✓
UniswapV2SwapAdapter	0xEEb24183819F5c36475736e4815d172E826D197b	Polygon	✓
UniswapV2SwapAdapter	0x84CE440919a003599c02C3f452F20d5AB0E8ad8C	Polygon	✓
UniswapV2SwapAdapter	0xe6Ba4747eA495Df2a21633f1d37182fe6B810150	Polygon	✓
UniswapV2SwapAdapter	0x99bD31d6c7c2cd8d801176e3612aFD665Acc44a0	Polygon	✓
UniswapV2SwapAdapter	0x5B657f905C1887706c44A408580fb46E1A315BF2	Polygon	✓
UniswapV2SwapAdapter	0xCF021F98AA14F9f9608541806d71d9f0145598Bd	Polygon	✓
XYBridgeAdapter	0x905F0f32007dfD9833aeA796d22D181b206a5dB6	Polygon	✓
AnySwapBridgeAdapter	0xdd0bba4959862022168E5aA5A321f71196ba684d	Polygon	✓
UxuyProtocol	0x76d2112cA038a9b13977c5E9007464e687747D6C	Fantom Opera	✓
UxuySwap	0xC7c29E0bD443AFf2c5AE18e7a54B95487F09A1d8	Fantom Opera	✓
UxuyBridge	0x031E5274FE6A6143B6Aec081783769D54Fe004ee	Fantom Opera	✓

Contract Name	Contract Address	Chain	Open Source or Not
Timelock	0xa0B24AFa81E77e65185d2a96d8D14AC33d12d98A	Fantom Opera	✓
UniswapV2SwapAdapter	0xBbc38E6e3352eb6F6dF8b0240b0aC709dA1DB00c	Fantom Opera	✓
UniswapV2SwapAdapter	0xec7Db1A2e39b532b84f2f5549330eb1485953743	Fantom Opera	✓
UniswapV2SwapAdapter	0x4A6789d5DF69a5EbD953b1A13f0DE7f458094081	Fantom Opera	✓
UniswapV2SwapAdapter	0xE3B55a5f28b0B4886743392194A463452179c28d	Fantom Opera	✓
UniswapV2SwapAdapter	0xc4f6d317163eF2A0C7a9Da1eACE3Eb73eA7C4Af8	Fantom Opera	✓
UniswapV2SwapAdapter	0x6C6BF364202bfbEcA2977Cc8f7de194B45D1A74c	Fantom Opera	✓
UniswapV2SwapAdapter	0xaC33cc2869cf22d1F70B746e1c0A7659D2349FFB	Fantom Opera	✓
FantomUniswapV2Adapter	0x173123d38f24aba859f1271f1ee2B995bE99aF20	Fantom Opera	✓
XYBridgeAdapter	0x8A0961681A48B64E95A5467600Ffb782D9F7C07c	Fantom Opera	✓
AnySwapBridgeAdapter	0xCF021F98AA14F9f9608541806d71d9f0145598Bd	Fantom Opera	✓
UxuyProtocol	0x19D14E7c50f49FFc1E1709ed473a2d6bd7AB0C96	Avalanche C-Chain	✓
UxuySwap	0xDfDc1d9562e7b38C1C04Ca949C704Ab78687fb5c	Avalanche C-Chain	✓
UxuyBridge	0x31e3abe5d149A5c6aD2Ad8e4b2779fDfAef05Dc7	Avalanche C-Chain	✓
Timelock	0x3d8fd8EBC5530342B77E0172E7EF0627417a1CBA	Avalanche C-Chain	✓
UniswapV2SwapAdapter	0xf15B31A7d7064905006CC765C427537F05530090	Avalanche C-Chain	✓
TraderJoeV1SwapAdapter	0x9800F9EE815a6a4063BC0297435dB64abC5bDf6B	Avalanche C-Chain	✓

Contract Name	Contract Address	Chain	Open Source or Not
TraderJoeV1SwapAdapter	0xFb042c97263E5fcC951Ab88eD7bB28d997CE7A60	Avalanche C-Chain	✓
UniswapV2SwapAdapter	0x15d1ff81455D40e7ABcB0Ca521724f76ABaddB0a	Avalanche C-Chain	✓
CurveSwapAdapter	0x7954480Caa4ff4601b1e0964e164884b14Ad7910	Avalanche C-Chain	✓
XYBridgeAdapter	0xC7c29E0bD443AFf2c5AE18e7a54B95487F09A1d8	Avalanche C-Chain	✓
AnySwapBridgeAdapter	0x031E5274FE6A6143B6Aec081783769D54Fe004ee	Avalanche C-Chain	✓
UxuyProtocol	0x3d8fd8EBC5530342B77E0172E7EF0627417a1CBA	Binance Smart Chain	✓
UxuySwap	0x8a2728f15380cB90076A1a165FbD28848271d270	Binance Smart Chain	✓
UxuyBridge	0x44F7cE718c0AC0088763AF2feb5488EE9896F7da	Binance Smart Chain	✓
Timelock	0xF00a05F73cc002255Dae7408D52f3865f59A39dd	Binance Smart Chain	✓
UniswapV2SwapAdapter	0x4f34fd6de52373F8393C137EFc87a58Ed36bBfdD	Binance Smart Chain	✓
OneInchSwapAdapter	0x1ec64dECd438537D1CE5061fE8aEb9d5078788de	Binance Smart Chain	✓
UniswapV2SwapAdapter	0x7F24d0C188620bca036B82B057c699FeD99390d3	Binance Smart Chain	✓
UniswapV2SwapAdapter	0x468A838f95866DC7558239C2b62304E7c90cb27d	Binance Smart Chain	✓
UniswapV2SwapAdapter	0x7A483fAb45df29D26e69854771d04f23642F0aa5	Binance Smart Chain	✓
BakerySwapAdapter	0x84aDA12d9e4b7991bE3f127Bc7720f93F8473FD7	Binance Smart Chain	✓
BakerySwapAdapter	0xd068CCbbC0bc54984b1654777bEc7EC28a4aB8EA	Binance Smart Chain	✓
UniswapV2SwapAdapter	0xF8b92b1224C01afD64845809f5aFe59806dff571	Binance Smart Chain	✓

Contract Name	Contract Address	Chain	Open Source or Not
UniswapV2SwapAdapter	0xEEb24183819F5c36475736e4815d172E826D197b	Binance Smart Chain	✓
UniswapV2SwapAdapter	0x84CE440919a003599c02C3f452F20d5AB0E8ad8C	Binance Smart Chain	✓
UniswapV2SwapAdapter	0xe6Ba4747eA495Df2a21633f1d37182fe6B810150	Binance Smart Chain	✓
BakerySwapAdapter	0x99bD31d6c7c2cd8d801176e3612aFD665Acc44a0	Binance Smart Chain	✓
UxuyProtocol	0x848b5a142dcaC871EB859bf55C9dcD525549e6bD	Arbitrum One	✓
UxuySwap	0xE2f055eb653b721d71EA11dacE990c46ACBA950B	Arbitrum One	✓
UxuyBridge	0xF4338E0D7eFF4472648b6a563A5f9ea21d5D57EB	Arbitrum One	✓
Timelock	0xbFc78b288854908b2d54622aabeF5A5323CE30f6	Arbitrum One	✓
UniswapV3SwapAdapter	0x6c2d5CFD51d83A7E64cEAfcFcbd2A8e3bE699955	Arbitrum One	✓
UniswapV2SwapAdapter	0xa3B758179Cb05fD50E7Ac1D7A1828b7b15226Ff5	Arbitrum One	✓
UniswapV2SwapAdapter	0xf57572A5D203260bC735BE05798F5EDF3C1C6f8D	Arbitrum One	✓
UniswapV2SwapAdapter	0xB1efD7676116A5Af62296d1bAd5243C4F4099371	Arbitrum One	✓
UniswapV2SwapAdapter	0xDfDc1d9562e7b38C1C04Ca949C704Ab78687fb5c	Arbitrum One	✓
XYBridgeAdapter	0x31e3abe5d149A5c6aD2Ad8e4b2779fDfAef05Dc7	Arbitrum One	✓
AnySwapBridgeAdapter	0x19D14E7c50f49FFc1E1709ed473a2d6bd7AB0C96	Arbitrum One	✓
UxuyProtocol	TRm7q3ncanvx2jfZoNrb2WVYP9KBxhnMxt	TRON	✓
UxuySwap	TL43hVxXxmvBynrW1tF1cCzu3SFgcXWBmk	TRON	✓

Contract Name	Contract Address	Chain	Open Source or Not
UxuyBridge	TRdTaieyXM4fxTnhdMgRj4GE5XaZtd2Bfc	TRON	✓
Timelock	TMsn19oosUoZoZA58MQypUA6yDWY4GhAsq	TRON	✓
UniswapV2SwapAdapter	TGfJPirsWWibLPKJnoaqw7XVbiZ7Qhc34o	TRON	✓

4.2 Visibility Description

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

AnySwapBridgeAdapter			
Function Name	Visibility	Mutability	Modifiers
supportSwap	External	-	-
bridge	External	Payable	whenNotPaused onlyAllowedCaller noDelegateCall

ConnexBridgeAdapter			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
setUnwrappers	External	Can Modify State	onlyOwner
_setUnwrappers	Internal	Can Modify State	-
setDelegate	External	Can Modify State	onlyOwner
supportSwap	External	-	-
bridge	External	Payable	whenNotPaused onlyAllowedCaller noDelegateCall

XYBridgeAdapter			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
supportSwap	External	-	-
bridge	External	Payable	whenNotPaused onlyAllowedCaller noDelegateCall

CallerControl			
Function Name	Visibility	Mutability	Modifiers
_updateAllowedCaller	Internal	Can Modify State	-

CommonBase			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
<Receive Ether>	External	Payable	-
pause	External	Can Modify State	onlyOwner
unpause	External	Can Modify State	onlyOwner
withdrawNativeAsset	External	Can Modify State	onlyOwner
withdrawToken	External	Can Modify State	onlyOwner
updateAllowedCaller	External	Can Modify State	onlyOwner
_checkNotDelegateCall	Private	-	-

ProviderRegistry			
Function Name	Visibility	Mutability	Modifiers
setProvider	External	Can Modify State	onlyOwner
setProviders	External	Can Modify State	onlyOwner

ProviderRegistry			
_setProvider	Internal	Can Modify State	-
removeProvider	External	Can Modify State	onlyOwner
removeProviders	External	Can Modify State	onlyOwner
_removeProvider	Internal	Can Modify State	-
getProvider	External	-	-
getProviders	External	-	-
_getProvider	Internal	-	-

SafeNativeAsset			
Function Name	Visibility	Mutability	Modifiers
nativeAsset	Internal	-	-
isNativeAsset	Internal	-	-
safeTransfer	Internal	Can Modify State	-

SwapAdapterBase			
Function Name	Visibility	Mutability	Modifiers
getAmountIn	External	Can Modify State	-
getAmountOut	External	Can Modify State	-
getAmountInView	Public	-	-
getAmountOutView	Public	-	-
WrappedNativeAsset	Public	-	-
_wrapNativeAsset	Internal	Can Modify State	-
_unwrapNativeAsset	Internal	Can Modify State	-

SwapAdapterBase			
_setWrappedNativeAsset	Internal	Can Modify State	-
_convertPath	Internal	-	-

BakerySwapAdapter			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
getAmountInView	Public	-	-
getAmountOutView	Public	-	-
swap	External	Payable	whenNotPaused onlyAllowedCaller noDelegateCall handleWrap
_swapExactBNBForTokens	Internal	Can Modify State	-
_swapExactTokensForOthers	Internal	Can Modify State	-
_swapExactTokensForOthersSupportingFeeOnTransferTokens	Internal	Can Modify State	-

CurveSwapAdapter			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
getAmountOutView	Public	-	-
swap	External	Payable	whenNotPaused onlyAllowedCaller noDelegateCall
_getTokensIndex	Internal	Can Modify State	-

FantomUniswapV2Adapter			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
getAmountInView	Public	-	-
getAmountOutView	Public	-	-
swap	External	Payable	whenNotPaused onlyAllowedCaller noDelegateCall handleWrap
_swapExactFTMForTokens	Internal	Can Modify State	-
_swapExactTokensForOthers	Internal	Can Modify State	-
_swapExactTokensForOthersSupportingFeeOnTransferTokens	Internal	Can Modify State	-

IZumiSwapAdapter			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
getAmountOut	External	Can Modify State	-
swap	External	Payable	whenNotPaused onlyAllowedCaller noDelegateCall handleWrap

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

KyberSwapAdapter			
_getERC20Path	Internal	-	-
_getPoolPath	Internal	-	-
_getPool	Internal	-	-
swap	External	Payable	whenNotPaused onlyAllowedCaller noDelegateCall handleWrap

MDEXSwapAdapter			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
getAmountInView	Public	-	-
getAmountOutView	Public	-	-
swap	External	Payable	whenNotPaused onlyAllowedCaller noDelegateCall handleWrap
_swapExactETHForTokens	Internal	Can Modify State	-
_swapExactTokensForOthers	Internal	Can Modify State	-
_swapExactTokensForOthersSupportingFeeOnTransferTokens	Internal	Can Modify State	-

OneInchSwapAdapter			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
swap	External	Payable	whenNotPaused onlyAllowedCaller noDelegateCall

TraderJoeV1SwapAdapter			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
getAmountInView	Public	-	-
getAmountOutView	Public	-	-
swap	External	Payable	whenNotPaused onlyAllowedCaller noDelegateCall handleWrap
_swapExactAVAXForTokens	Internal	Can Modify State	-
_swapExactTokensForOthers	Internal	Can Modify State	-
_swapExactTokensForOthersSupportingFeeOnTransferTokens	Internal	Can Modify State	-

UniswapV2SwapAdapter			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
getAmountInView	Public	-	-
getAmountOutView	Public	-	-
swap	External	Payable	whenNotPaused onlyAllowedCaller noDelegateCall handleWrap
_swapExactETHForTokens	Internal	Can Modify State	-
_swapExactTokensForOthers	Internal	Can Modify State	-
_swapExactTokensForOthersSupportingFeeOnTransferTokens	Internal	Can Modify State	-

UniswapV3SwapAdapter			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
getAmountIn	External	Can Modify State	-
getAmountOut	External	Can Modify State	-
swap	External	Payable	whenNotPaused onlyAllowedCaller noDelegateCall handleWrap

UxuySwap			
Function Name	Visibility	Mutability	Modifiers
getAmountIn	External	Can Modify State	-
getAmountOut	External	Can Modify State	-
getAmountInView	Public	-	-
getAmountOutView	Public	-	-
swap	External	Payable	whenNotPaused onlyAllowedCaller noDelegateCall
_getAdapter	Internal	-	-

UxuyBridge			
Function Name	Visibility	Mutability	Modifiers
supportSwap	External	-	-
bridge	External	Payable	whenNotPaused onlyAllowedCaller noDelegateCall
_getAdapter	Internal	-	-

UxuyProtocol			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
swapContract	External	-	-
bridgeContract	External	-	-
feeDenominator	External	-	-
feeRate	External	-	-
feeShareRate	External	-	-
isFOCAccount	External	-	-
setContract	External	Can Modify State	onlyOwner
setFeeRate	External	Can Modify State	onlyOwner
setFeeRecipient	External	Can Modify State	onlyOwner
updateFOCAccounts	External	Can Modify State	onlyOwner
updateFeeTokens	External	Can Modify State	onlyOwner
trade	External	Payable	whenNotPaused noDelegateCall nonReentrant checkDeadline
_setContract	Internal	Can Modify State	-
_setFeeRate	Internal	Can Modify State	-
_setFeeRecipient	Internal	Can Modify State	-
_findFeeToken	Internal	-	-
_payExtraFee	Internal	Can Modify State	-
_payFee	Internal	Can Modify	-

UxuyProtocol			
		State	
_needPayFee	Internal	-	-
_safeTransfer	Internal	Can Modify State	-
_tokenOut	Internal	-	-

4.3 Vulnerability Summary

[N1] [Suggestion] Maximum approval issue

Category: Design Logic Audit

Content

In the bridges and swaps modules of the protocol, the adapter will approve the maximum allowance to the external protocol through the safeApproveToMax function. But in fact, the external protocol does not need to use so much allowance. So this would violate the principle of least authorization.

Code location:

contracts/bridges/*.sol

```
    function bridge(
        BridgeParams calldata params
    ) external payable whenNotPaused onlyAllowedCaller noDelegateCall returns
(uint256, uint256) {
        ...
            IERC20(params.tokenIn).safeApproveToMax(...);
        ...
    }
```

contracts/swaps/*.sol

```
    function swap(
        SwapParams calldata params
    ) external payable whenNotPaused onlyAllowedCaller noDelegateCall
handleWrap(params) returns (uint256 amountOut) {
        ...
            IERC20(tokenIn).safeApproveToMax(...);
        ...
    }
```

Solution

It is recommended to approve as much allowance as the external protocol uses, instead of directly approving the maximum amount of allowance, so as to avoid the unpredictable impact of future external protocol on UXUY.

Status

Acknowledged; After communicating with the project team, the project team stated that it will not modify it in order to save gas.

[N2] [Suggestion] Redundant payable tag

Category: Others

Content

In the UxuyProtocol contract, users can perform cross-chain or swap operations through the trade function. If the user performs swap/bridge operations on native tokens, UxuyProtocol will directly send the native tokens to the adapter contract. Therefore, the payable flags of the swap/bridge functions in UxuyBridge, UxuySwap, and adapter contracts are redundant.

Code location:

contracts/UxuyBridge.sol

contracts/bridges/*.sol

```
    function bridge(
        BridgeParams calldata params
    ) external payable whenNotPaused onlyAllowedCaller noDelegateCall returns
 (uint256, uint256) {
        ...
    }
```

contracts/UxuySwap.sol

contracts/swaps/*.sol

```
    function swap(
        SwapParams calldata params
    ) external payable whenNotPaused onlyAllowedCaller noDelegateCall
handleWrap(params) returns (uint256 amountOut) {
        ...
            IERC20(tokenIn).safeApproveToMax(...);
        ...
    }
```

Solution

Since the protocol does not need to pass in native tokens when calling swap/bridge, it is recommended to remove these redundant payable tags.

Status

Fixed

[N3] [Suggestion] Parameters are not strictly checked

Category: Others

Content

In the UxuyProtocol contract, when swaps.length is greater than 0 and bridge.providerID is not 0, the user can use the trade function to perform swap first and then perform bridge operations. However, the trade function does not strictly check whether the tokenOut after the swap is consistent with the bridge.tokenIn passed in by the user.

Code location:

contracts/UxuyProtocol.sol

```
    function trade(
        TradeParams calldata params
    )
        external
        payable
        whenNotPaused
        noDelegateCall
        nonReentrant
        checkDeadline(params.deadline)
        returns (uint256 amountOut, uint256 bridgeTxnID)
    {
        TradeState memory state;
        if (params.swaps.length > 0) {
            ...
            state.tokenIn = params.swaps[0].path[0];
            state.nextRecipient =
_swapContract.getProvider(params.swaps[0].providerID);
        } else {
            state.tokenIn = params.bridge.tokenIn;
            state.nextRecipient =
_bridgeContract.getProvider(params.bridge.providerID);
        }
        if (params.bridge.providerID == 0) {
            state.tokenOut = _tokenOut(params.swaps[params.swaps.length - 1].path);
        } else {
            require(
                _bridgeContract.getProvider(params.bridge.providerID) !=
NULL_ADDRESS,
                "UxuyProtocol: invalid bridge provider"
            );
            state.tokenOut = params.bridge.tokenOut;
        }
        ...
    }
```

Solution

When `params.swaps.length` is greater than 0 and `params.bridge.providerID` is not 0, check whether the out token of swap is equal to `bridge.tokenIn`.

Status

Fixed

[N4] [Low] Bypass the fee charge for token swap

Category: Design Logic Audit

Content

In the trade function of the UxuyProtocol contract, when the extraFeeAmountIn passed in by the user is greater than 0, the `_payExtraFee` function will be used to exchange a part of the transaction fee for the user first. However, since the `_payExtraFee` function does not check whether tokenOut is a native token, and the amount of extraFeeAmountIn will be deducted from amountOut after the swap is completed. So malicious users can use this function to perform actual token swap instead of `params.swaps`.

Code location: contracts/UxuyProtocol.sol

```
function trade(
        TradeParams calldata params
    )
        external
        payable
        whenNotPaused
        noDelegateCall
        nonReentrant
        checkDeadline(params.deadline)
        returns (uint256 amountOut, uint256 bridgeTxnID)
    {
        ...
        if (params.extraFeeAmountIn > 0) {
            require(amountOut > params.extraFeeAmountIn, "UxuyProtocol: not enough
amount for extra fee");
            state.extraFeeAmount = _payExtraFee(state.tokenIn,
params.extraFeeAmountIn, params.extraFeeSwaps);
            amountOut -= params.extraFeeAmountIn;
        }
        ...
}
```

Solution

Make sure `params.extraFeeSwaps` final swap-out token is a native token, and check that the ratio of `params.extraFeeAmountIn` to `params.amountIn` should not be too large.

Status

Fixed; After communicating with the project team, the project team said that it will mitigate this risk by checking the ratio of `params.extraFeeAmountIn` in `params.amountIn`.

[N5] [Medium] Risk of over-privilege

Category: Authority Control Vulnerability Audit

Content

In the UxuyProtocol contract, the owner role can modify `_swapContract`, `_bridgeContract`, and `_feeRate` parameters respectively through the setContract and setFeeRate functions. Because the swap and bridge of user funds depend on `_swapContract` and `_bridgeContract` contracts. And the collection of user fees in the `_payFee` function depends on the `_feeRate` parameter, if the `_feeRate` parameter is much larger than expected, the funds in the approved user address will be at risk.

In the ProviderRegistry contract, the owner role can arbitrarily modify the addresses of providers through setProvider/removeProvider, and providers involve the processing of user funds. Therefore, this will lead to the risk of excessive owner permissions.

Code location:

contracts/UxuyProtocol.sol

```
    function setContract(address swapContract_, address bridgeContract_) external
onlyOwner {
        _setContract(swapContract_, bridgeContract_);
    }

    function setFeeRate(uint256 feeRate_, uint256 feeShareRate_) external onlyOwner {
        _setFeeRate(feeRate_, feeShareRate_);
    }
```

contracts/libraries/ProviderRegistry.sol

```
    function setProvider(bytes4 id, address provider) external override onlyOwner {
        _setProvider(id, provider);
    }

    function setProviders(bytes4[] calldata ids, address[] calldata providers)
external override onlyOwner {
        require(ids.length == providers.length, "ProviderRegistry: ids and providers
length mismatch");
        for (uint256 i = 0; i < ids.length; i++) {
            _setProvider(ids[i], providers[i]);
        }
    }

    function removeProvider(bytes4 id) external override onlyOwner {
        _removeProvider(id);
    }

    function removeProviders(bytes4[] calldata ids) external override onlyOwner {
        for (uint256 i = 0; i < ids.length; i++) {
            _removeProvider(ids[i]);
        }
    }
```

Solution

It is recommended to add a timelock contract to manage the owner role, and ensure that there is no less than 48 hours of time delay when operating sensitive parameters. At the same time, the role of suspending the protocol in emergency situations should be added to ensure that the project team can quickly respond to various special situations. And it is recommended to limit the feeRate to a range to ensure that changes in fees are expected.

Status

Fixed; After communicating with the project team, the project team stated that it will use the timelock contract to control the call of sensitive functions to mitigate this risk.

At present, the project team has transferred the ownership of the agreement to the timelock contract. The timelock currently delays by 24 hours. The following permission transfer transactions:

Contract Name	Chain	Transaction
UxuyProtocol	Ethereum	0xdde7d336ed41158530bb77bb1b02cca76833ff501676ea40033693385493a222
UxuyBridge	Ethereum	0xfce4434aebd8ef1b8f53344795e2a6a46f354c2378d383a5c1532f4be4dc0f9c
UxuySwap	Ethereum	0xc1f2b83b08a1750c722673b4e66285b6d5d6c589e02863b6a7d8a28b2295d77f
UxuyProtocol	Optimism	0x34ab8ddd7369d69aee7a7f8e24bad3e479d25936a26a1b3ad4aeea6ff4818390
UxuyBridge	Optimism	0xf9b45ea545343bd1ab595021d5f795edb2f27de50b661ec9fe061df8b6a3353b
UxuySwap	Optimism	0x8b06c5d0470a9ae7711c2f152d9c635645e6622e0d03a745ea3659e47c4aae80
UxuyProtocol	Polygon	0x4ab01c71e1c82fce998fcfbb29f1c77f697714a250b77410c44de17a813d90c7
UxuyBridge	Polygon	0xbffaf3b915bf3c2fc6a8149a10018f49842ce552a95b250b9a2e34d6a77f71cc
UxuySwap	Polygon	0xaef467ed9225f7699c2e2ee460cdfa6a21ee358ffd97b5ba5f2d3a644e5e4d35
UxuyProtocol	Fantom Opera	0xce3bc6a51e58691a6259290cd56da5236b3f0b20d42a08f1dabba8dd8a0f06e4

Contract Name	Chain	Transaction
UxuyBridge	Fantom Opera	0x3267c44ac03dc170a57891396122dd882495b85e6259751e7288d9153e229c1b
UxuySwap	Fantom Opera	0x89fbad02c7a6a30e2a8c829e457ac3f5a95f4d2a63fa8f14892a442017a7ff9f
UxuyProtocol	Avalanche C-Chain	0xa424050fb32c25db4694bb5c55646476444f2cd70461b7983b3d917859bc4abc
UxuyBridge	Avalanche C-Chain	0x0cfe08fbba49fb999ac3e780afd2a3bc0791cff1ddf9aa2a3c3aa174ed125b55
UxuySwap	Avalanche C-Chain	0xaacf84c47324fbb7f211fef967966bbda573329eab8ba3150a1329b4ec8bc3d4
UxuyProtocol	Binance Smart Chain	0x5d9b9acef996525c88c719c23a957ab14b043536fb496ff4359730ffe0b1db5d
UxuyBridge	Binance Smart Chain	0xb87168ffddfe020f9b17ca6c81e6d50154cb05addf07c1c8ffe40e102c88dc5d
UxuySwap	Binance Smart Chain	0x2cb231057ce88edd55f97269184b20ae61556cc12326133ac8040bce58f39224
UxuyProtocol	Arbitrum One	0x76f8fe1611547c345c7adaf98c2df82fd3fef81e15736ed2cace18a64defd845
UxuyBridge	Arbitrum One	0xff5613e26575160af574e33441a85d5c9f55914671a452d2002237f87089cc66
UxuySwap	Arbitrum One	0x3a355cc9824d69ec832b8ab314b1117308d1635f8089075f3a374802ee3b2124
UxuyProtocol	TRON	b7af453863b4dbf2097097034da4944e9303a7a489b16bade74ad9b9527b42dc
UxuyBridge	TRON	dc2d8b227d799d0a8438fe9433f14c672b930539dad588299df11425f808ae00
UxuySwap	TRON	6716d1278ef098d02d15866b00f3af4c9314e61bdeaa657d138c6ef5c68858a8

[N6] [Low] The AnySwap router is not configured

Category: Design Logic Audit

Content

In the agreement, the funds will call the external protocol through each bridge adapter to cross-chain. The external protocol address to be called is configured in the constructor of the ConnexBridgeAdapter and

XYBridgeAdapter contracts, but it is not configured in the AnySwapBridgeAdapter contract. If the user passes in an incorrect router address, it may result in a loss of funds.

Code location: contracts/bridges/AnySwapBridgeAdapter.sol

```
    function bridge(
        BridgeParams calldata params
    ) external payable whenNotPaused onlyAllowedCaller noDelegateCall returns
(uint256, uint256) {
        (address router, address anyToken, address tokenAddr) =
abi.decode(params.data, (address, address, address));
        ...
    }
```

Solution

It is recommended to configure the router address in the initialization function of the AnySwapBridgeAdapter contract.

Status

Fixed

[N7] [Low] The check of balanceBefore in swap is flawed

Category: Design Logic Audit

Content

In the UniswapV2SwapAdapter contract, the swap function is used to call an external DEX for token swap operations. In the case that tokenIn is not a native token, it will obtain the balance of this contract as balanceBefore and compare it with `params.amountIn`. When balanceBefore is less than `params. amountIn`, it will choose to swap through the supporting fee interface. But theoretically `params. amountIn` is also the actual token balance in this contract, which is the amountOut calculated by UxuyProtocol and UxuySwap. And malicious users may interfere with the comparison between balanceBefore and `params. amountIn` by directly transferring funds to the adapter contract. So checking that balanceBefore is less than `params. amountIn` as a condition for using the supporting fee interface is not practical.

The same is true for BakerySwapAdapter, FantomUniswapV2Adapter, MDEXSwapAdapter, and TraderJoeV1SwapAdapter contracts.

Code location:

contracts/swaps/UniswapV2SwapAdapter.sol

contracts/swaps/BakerySwapAdapter.sol

contracts/swaps/FantomUniswapV2Adapter.sol

contracts/swaps/MDEXSwapAdapter.sol

contracts/swaps/TraderJoeV1SwapAdapter.sol

```
    function swap(
        SwapParams calldata params
    ) external payable whenNotPaused onlyAllowedCaller noDelegateCall
handleWrap(params) returns (uint256 amountOut) {
        ...
        } else {
            IERC20(tokenIn).safeApproveToMax(address(_router), params.amountIn);
            uint256 balanceBefore = IERC20(tokenIn).balanceOf(address(this));
            require(balanceBefore > (params.amountIn * 95) / 100,
"UniswapV2SwapAdapter: not enough balance");
            if (balanceBefore < params.amountIn) {
                amountOut = _swapExactTokensForOthersSupportingFeeOnTransferTokens(
                    params.path,
                    params.recipient,
                    params.amountIn,
                    params.minAmountOut
                );
            } else {
            ...
        }
    }
```

Solution

Since the `_swapExactTokensForOthers` function already includes compatibility with supporting fee tokens, it is recommended to use `_swapExactTokensForOthers` directly in the swap function for token swap.

Status

Fixed

[N8] [Suggestion] `try-catch` does not check the cause of the error

Category: Others

Content

In the UniswapV2SwapAdapter contract, `_swapExactETHForTokens` and `_swapExactTokensForOthers` functions are used to call the external Router contract for token swap. It will use `try-catch` to try to execute the supporting fee token swap interface when the non-supporting fee token swap interface fails. However, for supporting fee tokens, the failure to execute through the non-supporting fee token swap interface is due to the fact that the amountOut is larger than the actual value and cannot pass the K value check during swap. Therefore, it may be more reasonable to perform swap through the supporting fee interface only when the K value check fails.

The same is true for BakerySwapAdapter, FantomUniswapV2Adapter, MDEXSwapAdapter, and TraderJoeV1SwapAdapter contracts.

Code location:

contracts/swaps/UniswapV2SwapAdapter.sol

contracts/swaps/BakerySwapAdapter.sol

contracts/swaps/FantomUniswapV2Adapter.sol

contracts/swaps/MDEXSwapAdapter.sol

contracts/swaps/TraderJoeV1SwapAdapter.sol

```
    function _swapExactETHForTokens(
        address[] memory path,
        address recipient,
        uint256 amountIn,
        uint256 minAmountOut
    ) internal returns (uint256 amountOut) {
        require(address(this).balance >= amountIn, "UniswapV2SwapAdapter: not enough
native assets in transaction");
        path[0] = WrappedNativeAsset();
        try _router.swapExactETHForTokens{value: amountIn}(minAmountOut, path,
recipient, UNEXPIRED) returns (
            ...
        } catch {
           ...
    }

    function _swapExactTokensForOthers(
        address[] memory path,
        address recipient,
        uint256 amountIn,
        uint256 minAmountOut
    ) internal returns (uint256 amountOut) {
        ...
    }
```

Solution

It is recommended to use the Error function in the catch to get the failed message and check the error message.

For example:

```
try...
...
catch Error(string memory reason) {
require(keccak256(abi.encodePacked(reason)) == keccak256(abi.encodePacked("UniswapV2:
K")), reason);
...
}
```

Status

Acknowledged; After communicating with the project team, the project team indicated that error messages are not processed as they are returned differently between protocols.

[N9] [Low] Curve pool selection issue

Category: Design Logic Audit

Content

In the CurveSwapAdapter contract, the swap function is used to call Curve Pool to exchange tokens for users. But the pool address is passed in by the user and its validity is not checked. At the same time, because the pool address is imported from the outside, the protocol cannot ensure whether the pool selected by the user is the best pool.

Code location: contracts/swaps/CurveSwapAdapter.sol

```
    function swap(
        SwapParams calldata params
    ) external payable whenNotPaused onlyAllowedCaller noDelegateCall returns
(uint256 amountOut) {
        address pool = abi.decode(params.data, (address));
        ...
    }
```

Solution

It is recommended to check whether the pool passed in by the user is registered in Curve to ensure the validity of the pool. (For example, check through functions such as get_pool_name/get_n_coins of the registration contract) If you need to ensure that the pool selected by the protocol is the best, you should directly obtain the best pool address through the get_best_rate interface in the swap function.

Status

Fixed

[N10] [Low] Curve pool index issue

Category: Design Logic Audit

Content

In the CurveSwapAdapter contract, the `_getTokensIndex` function performs a for loop check on coins to get the indexes of tokenIn and tokenOut in the pool. However, it only loops up to 5 times, while the MAX_COINS in the registered contract is fixed at 8, which means that the pools may support more than 5 tokens.

Code location: contracts/swaps/CurveSwapAdapter.sol

```
function _getTokensIndex(
    address pool,
    address tokenIn,
    address tokenOut
) internal returns (int128 indexIn, int128 indexOut) {
    indexIn = -1;
    indexOut = -1;
    for (uint256 i = 0; i < 5; i++) {
       ...
}
```

Solution

It is recommended to increase the maximum number of loops to 8, or first obtain the amount of tokens supported by the pool through the get_n_coins function as the number of loops.

Status

Fixed

[N11] [Suggestion] Arbitrary function call issue

Category: Design Logic Audit

Content

In the XYBridgeAdapter contract, the bridge function is used to call the XY Finance protocol for cross-chain operations, but the incoming call data is not checked. This will cause the user's calls not to be as expected. The same is true for the swap function in the OneInchSwapAdapter contract.

Code location:

contracts/bridges/XYBridgeAdapter.sol

```
    function bridge(
        BridgeParams calldata params
    ) external payable whenNotPaused onlyAllowedCaller noDelegateCall returns
(uint256, uint256) {
        bool success;
        bytes memory data;
        if (!params.tokenIn.isNativeAsset()) {
            IERC20(params.tokenIn).safeApproveToMax(_xyBridge, params.amountIn);
        }
        (success, data) = _xyBridge.call{value: params.tokenIn.isNativeAsset() ?
params.amountIn : 0}(params.data);
        require(success, string(abi.encodePacked("XYBridgeAdapter: call xybridge
failed: ", data)));

        return (0, 0);
    }
```

contracts/swaps/OneInchSwapAdapter.sol

```
    function swap(
        SwapParams calldata params
    ) external payable whenNotPaused onlyAllowedCaller noDelegateCall returns
(uint256 amountOut) {
        if (!params.path[0].isNativeAsset()) {
            IERC20(params.path[0]).safeApproveToMax(address(_aggregator),
params.amountIn);
        }
        bool success;
        bytes memory result;
        (success, result) = _aggregator.call{value: params.path[0].isNativeAsset() ?
params.amountIn : 0}(params.data);
        if (!success) {
            revert("OneInchSwapAdapter: call 1inch failed");
        }
        (amountOut, ) = abi.decode(result, (uint256, uint256));
    }
```

Solution

It is recommended to make explicit external calls through the interface.

Status

Fixed

[N12] [Suggestion] Potential slippage decimal issue

Category: Design Logic Audit

Content

In the bridge function of the ConnexBridgeAdapter contract, the slippage calculation will be performed through `((amountIn - params.minAmountOut) * 10000) / amountIn`. Currently, ConnexBridge only supports cross-chain tokens but not cross-chain and swap operations. So theoretically the decimal of minAmountOut and amountIn is the same. However, if ConnexBridge supports token swap in the future, the minAmountOut and the decimal of the amountIn parameter may be inconsistent when users perform cross-chain and exchange of tokens, which will lead to deviations in the slippage check.

Code location: contracts/bridges/ConnexBridgeAdapter.sol

```
    function bridge(
        BridgeParams calldata params
    ) external payable whenNotPaused onlyAllowedCaller noDelegateCall returns
(uint256, uint256) {
        ...
        uint256 slippage = ((amountIn - params.minAmountOut) * 10000) / amountIn;
        ...
    }
```

Solution

It is recommended to process the accuracy of the amountIn and minAmountOut according to the actual situation before calculating the slippage.

Status

Fixed; After communicating with the project team, the project team stated that it will no longer use ConnexBridge for cross-chain operations.

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002305080001	SlowMist Security Team	2023.04.23 - 2023.05.08	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 medium risk, 5 low risks, and 6 suggestions. All findings were fixed or acknowledged.

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