Smart Contract
Security Audit Report

Table Of Contents

1 Executive Summary

On 2023.08.28, the SlowMist security team received the UXUY Protocol team's security audit application for UXUY Protocol Phase2, 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

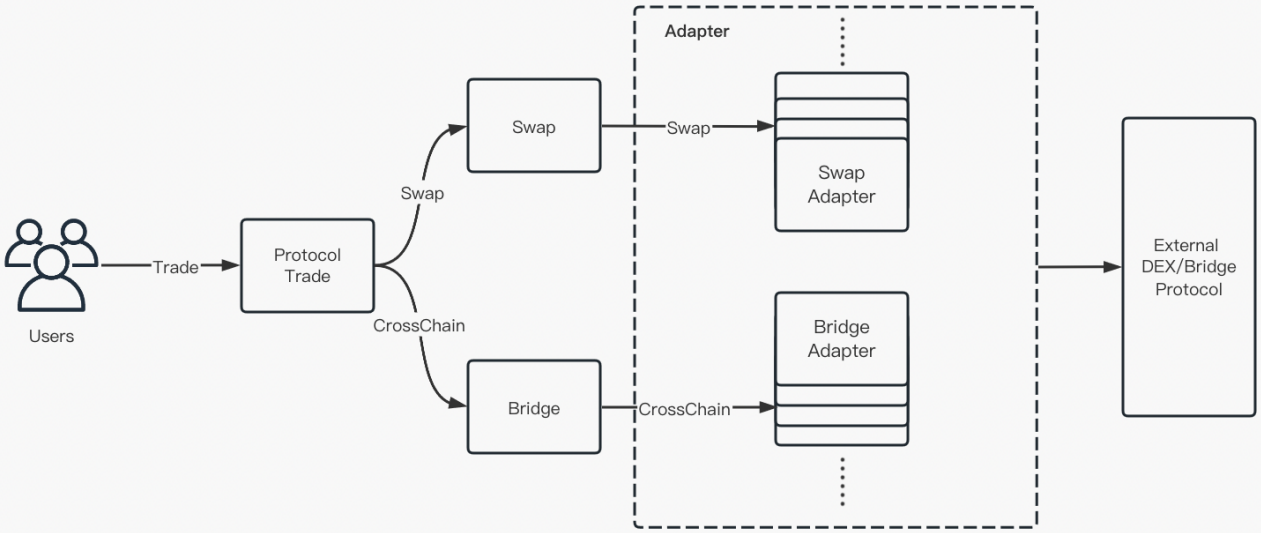
Serial Number	Audit Class	Audit Subclass
7	Security Design Audit	Block data Dependence Security Audit
		tx.origin Authentication Security Audit
8	Denial of Service Audit	-
9	Gas Optimization Audit	-
10	Design Logic Audit	-
11	Variable Coverage Vulnerability Audit	-
12	"False Top-up" Vulnerability Audit	-
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. This audit of the UXUY protocol is an iterative audit based on the previous audit.

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	Others	Suggestion	Acknowledged
N2	Risk of over-privilege	Authority Control Vulnerability Audit	Medium	Acknowledged
N3	`try-catch` does not check the cause of the error	Design Logic Audit	Suggestion	Acknowledged
N4	Potential risks of using the protocol for nefarious purposes	Design Logic Audit	Suggestion	Acknowledged
N5	Parameter status is not checked when modifying it	Design Logic Audit	Suggestion	Acknowledged
N6	Incorrect function modifier	Gas Optimization Audit	Suggestion	Acknowledged

4 Code Overview

4.1 Contracts Description

Audit Version:

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

commit: 09a3845ad89be942c734083344c920f32567aa31

The main network address of the contract is as follows:

Contract Name	Contract Address	Chain	Open Source or Not
UxuySwap	0x9800f9ee815a6a4063bc0297435db64abc5bdf6b	Ethereum	×
UxuyBridge	0xfb042c97263e5fcc951ab88ed7bb28d997ce7a60	Ethereum	×
UxuyProtocol	0x15d1ff81455D40e7ABcB0Ca521724f76ABaddB0a	Ethereum	✓
UniswapV2SwapAdapter	0xC7c29E0bD443AFf2c5AE18e7a54B95487F09A1d8	Ethereum	×
UniswapV3SwapAdapter	0x031E5274FE6A6143B6Aec081783769D54Fe004ee	Ethereum	×
OneInchSwapAdapter	0x6c6bf364202bfbeca2977cc8f7de194b45d1a74c	Ethereum	×
UPoolBridgeAdapter	0xec7Db1A2e39b532b84f2f5549330eb1485953743	Ethereum	×
Timelock	0xe4EFB6e54755A3d570FEBc784aa4751fcB5e3ab9	Ethereum	✓
UxuySwap	0x83B4efeC5A51D69C66563Deba24a6377b906197e	Binance Smart Chain	×
UxuyBridge	0x79053f0CCe8a93Ee5D3Fd0428fb8a926635b9D1A	Binance Smart Chain	×
UxuyProtocol	0x7B458DBE4E7a693BE2e691DEdDff4607104f1266	Binance Smart Chain	✓
UniswapV2SwapAdapter	0x6073D44F8eF4e5cBfc8fA8646c3296752806732c	Binance Smart Chain	×
UniswapV3SwapAdapter	0x66C2269392b193075d4B8CA0af4F7C8419ea0d8f	Binance Smart Chain	×

Contract Name	Contract Address	Chain	Open Source or Not
BakerySwapAdapter	0xc68796a03c9246925986bCfCA81B2451114ED2Af	Binance Smart Chain	✗
OneInchSwapAdapter	0xEB2456F422Bf6d7CE0b5c40A946be09Afa93F42e	Binance Smart Chain	✗
StargateBridgeAdapter	0x6050406fd854879D9D98833B4469DC2a59A8180c	Binance Smart Chain	✗
UPoolBridgeAdapter	0xd4fdD4588b4202411548C3DbDD5f4Fb763bC2E9E	Binance Smart Chain	✗
Timelock	0xF00a05F73cc002255Dae7408D52f3865f59A39dd	Binance Smart Chain	✓
UxuySwap	0x3d8fd8EBC5530342B77E0172E7EF0627417a1CBA	Arbitrum One	✗
UxuyBridge	0x4f34fd6de52373F8393C137EFc87a58Ed36bBfdD	Arbitrum One	✗
UxuyProtocol	0x1ec64dECd438537D1CE5061fE8aEb9d5078788de	Arbitrum One	✓
UniswapV2SwapAdapter	0x7A483fAb45df29D26e69854771d04f23642F0aa5	Arbitrum One	✗
UniswapV3SwapAdapter	0x84aDA12d9e4b7991bE3f127Bc7720f93F8473FD7	Arbitrum One	✗
OneInchSwapAdapter	0x5b657f905c1887706c44a408580fb46e1a315bf2	Arbitrum One	✗
UPoolBridgeAdapter	0xEEb24183819F5c36475736e4815d172E826D197b	Arbitrum One	✗
Timelock	0xbFc78b288854908b2d54622aabeF5A5323CE30f6	Arbitrum One	✓
UxuySwap	0x468A838f95866DC7558239C2b62304E7c90cb27d	Avalanche C-Chain	✗
UxuyBridge	0x7A483fAb45df29D26e69854771d04f23642F0aa5	Avalanche C-Chain	✗
UxuyProtocol	0x137a06a85e4557ef91244966d2b7f77a1b3481c3	Avalanche C-Chain	✓
UniswapV2SwapAdapter	0xEEb24183819F5c36475736e4815d172E826D197b	Avalanche C-Chain	✗

Contract Name	Contract Address	Chain	Open Source or Not
UniswapV3SwapAdapter	0x84CE440919a003599c02C3f452F20d5AB0E8ad8C	Avalanche C-Chain	✗
OneInchSwapAdapter	0xA2AC3F036E761fB2418a98f0C2648EbEc20958F6	Avalanche C-Chain	✗
UPoolBridgeAdapter	0x905F0f32007dfD9833aeA796d22D181b206a5dB6	Avalanche C-Chain	✗
Timelock	0x3d8fd8EBC5530342B77E0172E7EF0627417a1CBA	Avalanche C-Chain	✓
UxuySwap	0x8818F58784b4cDa3150cA887abD0c45bBb26B4Df	Fantom Opera	✗
UxuyBridge	0x2dF4a9076Fe14B00309635eA68F301bE2e0AFC6E	Fantom Opera	✗
UxuyProtocol	0xcF55b66034C64dD441ddB8aBBd8693B062461DFC	Fantom Opera	✓
UniswapV2SwapAdapter	0x03de74632AC9cFE86BBb3BaEc3711e6706f1189b	Fantom Opera	✗
UniswapV3SwapAdapter	0x4e5426397000174833fbfd2ece7ab7d4f12d7f01	Fantom Opera	✗
OneInchSwapAdapter	0xd07463f87e2f1af16febe782e12c8ab90e881510	Fantom Opera	✗
UPoolBridgeAdapter	0xc8e49A5e0447b99C13aad79542812858D4220b24	Fantom Opera	✗
Timelock	0xa0B24AFa81E77e65185d2a96d8D14AC33d12d98A	Fantom Opera	✓
UxuySwap	0x7954480Caa4ff4601b1e0964e164884b14Ad7910	Optimism	✗
UxuyBridge	0xBa5eb65b16D4d22288d548E0b6Ac474cd893a195	Optimism	✗
UxuyProtocol	0xC7c29E0bD443AFf2c5AE18e7a54B95487F09A1d8	Optimism	✓
UniswapV2SwapAdapter	0xBbc38E6e3352eb6F6dF8b0240b0aC709dA1DB00c	Optimism	✗
UniswapV3SwapAdapter	0xec7Db1A2e39b532b84f2f5549330eb1485953743	Optimism	✗

Contract Name	Contract Address	Chain	Open Source or Not
OneInchSwapAdapter	0x8a0961681a48b64e95a5467600ffb782d9f7c07c	Optimism	✗
UPoolBridgeAdapter	0xc4f6d317163eF2A0C7a9Da1eACE3Eb73eA7C4Af8	Optimism	✗
Timelock	0xf15B31A7d7064905006CC765C427537F05530090	Optimism	✓
UxuySwap	0xfC1586605cf97292f78FF97267d78490708fEbD0	Polygon	✗
UxuyBridge	0x49BDE1fC86922e9a2E744e0c12Db9384e6484EF0	Polygon	✗
UxuyProtocol	0x137A06a85e4557Ef91244966D2b7f77a1b3481c3	Polygon	✓
UniswapV2SwapAdapter	0x512C1AACd3D81cd22e8AC172295a15F4D53196ed	Polygon	✗
UniswapV3SwapAdapter	0x505891a5A9b81fc2078c3c441bC5166A61FE33bb	Polygon	✗
OneInchSwapAdapter	0xdf7c75b585d0878b30d0cd0258787ca688d2d904	Polygon	✗
StargateBridgeAdapter	0x013cCeD2fbeA59Aee73afb765DbffA7F0C979cD2	Polygon	✗
UPoolBridgeAdapter	0x9774ef62cF1F5985f1F2ee21d1e8c631a470f626	Polygon	✗
Timelock	0x83B4efeC5A51D69C66563Deba24a6377b906197e	Polygon	✓
UxuySwap	TSNBFBojQ2TTx22sKpGVPFnjavv4cD4R2k	TRON	✗
UxuyBridge	TKbMQrhxh5ytdzswRhArLPS4XFzUpBAsoi	TRON	✗
UxuyProtocol	TF8UrzKTXx1kMyG7oigcbuTNUNuaUET7JS	TRON	✗
UniswapV2SwapAdapter	TKAgimGwJqrjyTjp6foaQyiA9juF4oqLtA	TRON	✗
UPoolBridgeAdapter	TDuTc9J2RpD7M3vaXtSUuvxHFBQ9BRtsLe	TRON	✗

Contract Name	Contract Address	Chain	Open Source or Not
Timelock	TMsn19oosUoZoZA58MQypUA6yDWY4GhAsq	TRON	✓

4.2 Visibility Description

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

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	onlyAdmin
setFeeRate	External	Can Modify State	onlyAdmin
setFeeRecipient	External	Can Modify State	onlyAdmin
updateFOCAccounts	External	Can Modify State	onlyAdmin
updateFeeTokens	External	Can Modify State	onlyAdmin
trade	External	Payable	whenNotPaused noDelegateCall nonReentrant checkDeadline

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

UxuyBridge			
Function Name	Visibility	Mutability	Modifiers
supportSwap	External	-	-
bridge	External	Can Modify State	whenNotPaused onlyAllowedCaller noDelegateCall
_getAdapter	Internal	-	-

UxuySwap			
Function Name	Visibility	Mutability	Modifiers
swap	External	Can Modify State	whenNotPaused onlyAllowedCaller noDelegateCall
_getAdapter	Internal	-	-

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

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

BakerySwapAdapter			
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
_swapExactBNBForTokens	Internal	Can Modify State	-

BakerySwapAdapter			
_swapExactTokensForOthers	Internal	Can Modify State	-

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

UniswapV2SwapAdapter			
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
_swapExactETHForTokens	Internal	Can Modify State	-
_swapExactTokensForOthers	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	-

UniswapV3SwapAdapter			
swap	External	Payable	whenNotPaused onlyAllowedCaller noDelegateCall handleWrap

4.3 Vulnerability Summary

[N1] [Suggestion] Maximum approval issue

Category: Others

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] [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.

Code location:

contracts/UxuyProtocol.sol

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

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.

Status

Acknowledged; 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.

[N3] [Suggestion] `try-catch` does not check the cause of the error

Category: Design Logic Audit

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 contracts.

For example:

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

Code location:

contracts/swaps/UniswapV2SwapAdapter.sol

contracts/swaps/BakerySwapAdapter.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.

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.

[N4] [Suggestion] Potential risks of using the protocol for nefarious purposes

Category: Design Logic Audit

Content

In the protocol, users can pass in any specified provider/router address, which will make the protocol more flexible and easier to use and reduce some gas costs. But be aware that this design may be used for illegal purposes.

For example: malicious users can use the UXUYProtocol contract as the entry point of the phishing contract. Ordinary users will trust the UXUYProtocol contract, but malicious users can replace the provider/router address in the parameter with a malicious contract, so that the UXUYProtocol contract transfers the user's funds to the malicious user. This will not only cause users to suffer financial losses but will also damage the reputation of UXUY Protocol.

Code location: contracts/UxuyProtocol.sol

```
    function trade(
        TradeParams calldata params
```

```
    )
        external
        payable
        whenNotPaused
        noDelegateCall
        nonReentrant
        checkDeadline(params.deadline)
        returns (uint256 amountOut, uint256 bridgeTxnID)
    {
        ...
    }
```

Solution

It is recommended to use provider and router whitelists to avoid this risk.

Status

Acknowledged; After communicating with the project team, the project team stated that this was the expected design and would accept this risk.

[N5] [Suggestion] Parameter status is not checked when modifying it

Category: Design Logic Audit

Content

In the StargateBridgeAdapter contract, the owner role can modify the `_acceptedTokens` state through updateAcceptedTokens, but it does not check whether the `_acceptedTokens` state of the tokens that need to be modified is consistent with the state to be set.

The same is true for the updateAcceptedTokens and updateUAgents functions in the UPoolBridgeAdapter contract.

Code location: contracts/bridges/*.sol

```
    function updateAcceptedTokens(address[] calldata tokens, bool accepted) external
onlyOwner {
        ...
    }

    function updateUAgents(address[] calldata accounts, bool uagent) external
onlyOwner {
        ...
    }
```

Solution

It is recommended that when modifying the status, check whether the existing status is consistent with the status to be set. If they are consistent, there is no need to repeat the setting.

Status

Acknowledged

[N6] [Suggestion] Incorrect function modifier

Category: Gas Optimization Audit

Content

In the UniswapV2SwapAdapter and BakerySwapAdapter contracts, there are getAmountIn and getAmountOut functions for calculating the amount of tokens required for swap. However, there is no operation to change the storage state in its function, so the function can use the view modifier.

Code location:

contracts/swaps/BakerySwapAdapter.sol

contracts/swaps/UniswapV2SwapAdapter.sol

```
    function getAmountIn(
        address router,
        address[] memory path,
        uint256 amountOut
    ) external virtual returns (uint256 amountIn, bytes memory swapData) {
        ...
    }

    function getAmountOut(
        address router,
        address[] memory path,
        uint256 amountIn
    ) external virtual returns (uint256 amountOut, bytes memory swapData) {
        ...
    }
```

Solution

It is recommended to use the view modifier for functions that do not change state.

Status

Acknowledged

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002308300001	SlowMist Security Team	2023.08.28 - 2023.08.30	Medium Risk

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 and 5 suggestions. All the findings were Acknowledged. At present, some contracts are not open source, and the ownership of the admin role of the UxuyProtocol contract has not yet been transferred to the Timelock contract, so the protocol still has the risk of excessive privileges.

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