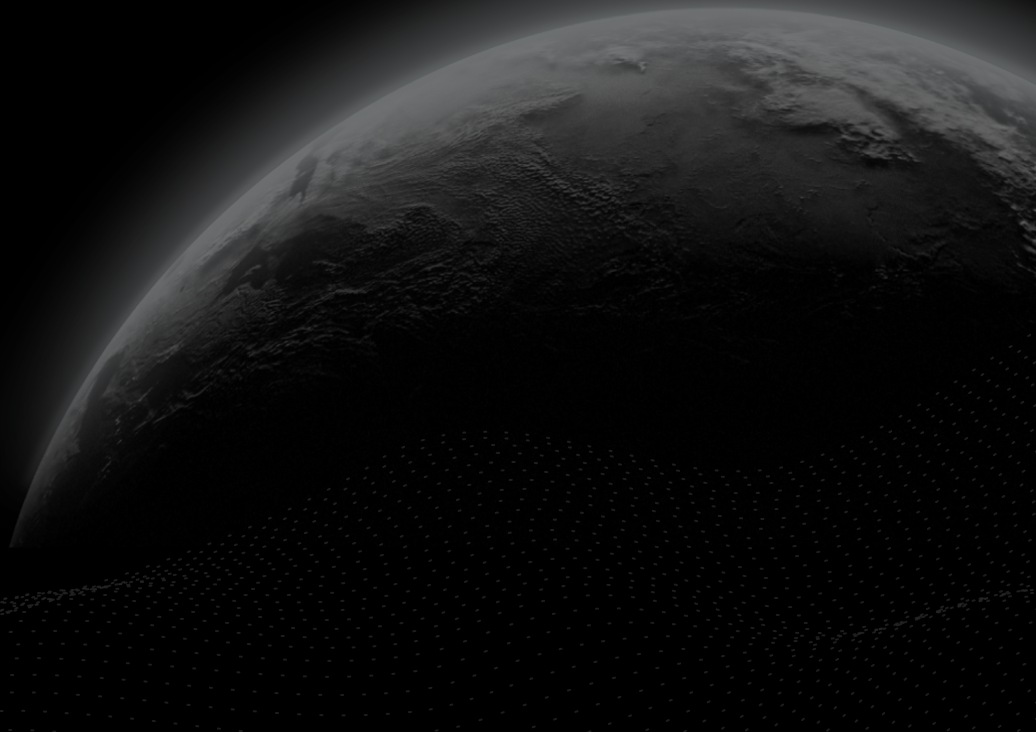




Security Assessment

OKX - Audit 3

CertiK Assessed on May 16th, 2023





CertiK Assessed on May 16th, 2023

OKX - Audit 3

The security assessment was prepared by CertiK, the leader in Web3.0 security.

Executive Summary

TYPES

Exchange

ECOSYSTEM

Ethereum (ETH)

METHODS

Manual Review, Static Analysis

LANGUAGE

Solidity

TIMELINE

Delivered on 05/16/2023

KEY COMPONENTS

N/A

CODEBASE

- Entrance.sol and UniswapV2AdapterMain.sol : <https://github.com/okx/Yield-External-Audit/>
- OkxNFTMarketAggregator.sol : <https://github.com/okx/NFT->

...View All

COMMITTS

- Entrance.sol and UniswapV2AdapterMain.sol : fc37a6284d42b98da8a5ed6b1b6d473168e167d9
- OkxNFTMarketAggregator.sol :

...View All

Vulnerability Summary



27

Total Findings

5

Resolved

0

Mitigated

0

Partially Resolved

22

Acknowledged

0

Declined

0 Critical

Critical risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.

1 Major 1 Acknowledged

Major risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.

0 Medium

Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.

10 Minor 5 Resolved, 5 Acknowledged

Minor risks can be any of the above, but on a smaller scale. They generally do not compromise the overall integrity of the project, but they may be less efficient than other solutions.

16 Informational 16 Acknowledged

Informational errors are often recommendations to improve the style of the code or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

TABLE OF CONTENTS | OKX - AUDIT 3

■ Summary

[Executive Summary](#)

[Vulnerability Summary](#)

[Codebase](#)

[Audit Scope](#)

[Approach & Methods](#)

■ Overview

[External Dependencies](#)

[Privileged Functions](#)

■ Findings

[OKX-03 : Centralization Related Risks](#)

[DRW-01 : Lack of Check on Weights](#)

[DRW-02 : Lack of Length Check on Batches](#)

[DRW-03 : Missing Validation That Sum of `batchesAmount` Is Equal to `baseRequest.fromTokenAmount`](#)

[DRW-04 : Possible Inconsistency Caused by Check of `baseRequest.fromTokenAmount`](#)

[DRW-05 : Possibly Incorrect Assignment to Variable `subIndex`](#)

[EYE-01 : Unprotected Initializer](#)

[EYE-02 : Unchecked ERC-20 `transfer\(\)`/`transferFrom\(\)` Call](#)

[EYE-03 : Conditional Check on Message Value Should Be Unconditional](#)

[ONF-01 : Variable `success` Is Overwritten](#)

[UVA-01 : Missing Validation of Token Order in Function `withdraw\(\)`](#)

[DRW-06 : Choice of Token for Parameter `fromToken` For Hops](#)

[DRW-07 : Typo in Variable Name](#)

[DRW-08 : Missing Zero Address Validation](#)

[EYE-04 : Unused Event](#)

[OKX-01 : Unnecessary Receive Function](#)

[OKX-02 : Unlocked Compiler Version](#)

[ONF-02 : Inconsistent Handling of Other Markets](#)

[ONF-03 : Overwriting Failure](#)

[ONF-04 : Allowed Markets in Function `tradeV2\(\)`](#)

[ONF-05 : Use of `delegatecall`](#)

ONF-06 : Function State Mutability Can Be Restricted to Pure and View

UVA-02 : Possible Missing Functionality From `BaseAdapter`

UVA-03 : Possibly Incorrect feeRate in Contract UniswapV2AdapterMain

UVA-04 : Reserves Are Used In Calculation of Liquidity Removal

UVA-05 : Inaccurate Require Statement

UVA-06 : Missing Validation of Tokens in Function `amountToShare()`

Appendix

Disclaimer

CODEBASE | OKX - AUDIT 3

Repository





- `Entrance.sol` and `UniswapV2AdapterMain.sol` : <https://github.com/okx/Yield-External-Audit/>
- `OkxNFTMarketAggregator.sol` : <https://github.com/okx/NFT-External-Audit-Certik>
- `DexRouter.sol` : <https://github.com/okx/Web3-DEX>

Commit

- `Entrance.sol` and `UniswapV2AdapterMain.sol` : fc37a6284d42b98da8a5ed6b1b6d473168e167d9
- `OkxNFTMarketAggregator.sol` :
 - 0d1fa3d4bc23e9b24094e8ad5432b532aa39f666
 - 401c90c5dec1fd313baa8c91125e25f8d035fd1e
- `DexRouter.sol` :
 - 18b3c82f2bcd3d7d3f3cffaf1412e9aecc409c55
 - 684a3cd4b746c8a28941bcdac878c819932ea2e0

AUDIT SCOPE | OKX - AUDIT 3

4 files audited ● 4 files with Acknowledged findings

ID	File	SHA256 Checksum
● DRW	 DexRouter.sol	9ea96090561db280d3e07b0477569100dd02fc9904d311aad5d508e3a19bef61
● ONF	 OkxNFTMarketAggregator.sol	acbf177ed1d5a61c6b5cf488776a4f421ea26d66a365b0675442cf516c3bd1ce
● EYE	 Entrance.sol	58aa49a7c1b72e69a1442bbbe745c30824b3dc562a5d819ad353c70df530d98e
● UVA	 adapters/uniswap/UniswapV2AdapterMain.sol	cf97675f513a2236a14f3a90674853711b4e456e9d08e33189dcafdce4624a43

APPROACH & METHODS | OKX - AUDIT 3

This report has been prepared for OKX to discover issues and vulnerabilities in the source code of the OKX - Audit 3 project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Manual Review and Static Analysis techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Testing the smart contracts against both common and uncommon attack vectors;
- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

OVERVIEW | OKX - AUDIT 3

OKX is a cryptocurrency and derivative exchange that adopts blockchain technology to build the financial ecosystem, including hundreds of crypto assets.

The current auditing service is provided for the following 4 contracts:

- **DexRouter**: A router built for different kinds of DEXs for the users to trade assets
- **OkxNFTMarketAggregator**: An NFT aggregator for different markets for the users to trade NFTs
- **Entrance**: An entrance to execute instructions that are allowed to be invoked by registered adapters
- **UniswapV2AdapterMain**: An adapter to the UniswapV2 pools that allows the users to stake the LP to MasterChef

External Dependencies

The scope of the audit treats third-party entities as black boxes and assumes their functional correctness. However, in the real world, third parties can be compromised and this may lead to lost or stolen assets.

There are a few dependent injection contracts or addresses in the current project:

- `approveProxy`, `adapters`, `pools`, `_WETH`, `wNativeRelayer`, and `xBridge` in the contract `DexRouter`
- `marketRegistry`, `proxy`, `CONDUIT`, and `SEAPORT` in the contract `OkxNFTMarketAggregator`
- `weth`, `approveProxy`, `adapter` registered in `registeredAdatper` and `registeredFlash` in the contract `Entrance`
- `rewardPool` and `pool` in the contract `UniswapV2AdapterMain`

We assume these contracts or addresses are valid and non-vulnerable actors and implement proper logic to collaborate with the current project.

Privileged Functions

In the **OKX** project, multiple privileged roles are adopted to ensure the dynamic runtime updates of the project, which were specified in the following findings `OKX-03 | Centralization Related Risks`.

The advantage of those privileged roles in the codebase is that the client reserves the ability to adjust the protocol according to the runtime required to best serve the community. It is also worth noting the potential drawbacks of these functions, which should be clearly stated through the client's action/plan. Additionally, if the private keys of the privileged accounts are compromised, it could lead to devastating consequences for the project.

To improve the trustworthiness of the project, dynamic runtime updates in the project should be notified to the community. Any plan to invoke the aforementioned functions should be also considered to move to the execution queue of the `TimeLock` contract.

FINDINGS | OKX - AUDIT 3



27

Total Findings

0

Critical

1

Major

0

Medium

10

Minor

16

Informational

This report has been prepared to discover issues and vulnerabilities for OKX - Audit 3. Through this audit, we have uncovered 27 issues ranging from different severity levels. Utilizing the techniques of Manual Review & Static Analysis to complement rigorous manual code reviews, we discovered the following findings:

ID	Title	Category	Severity	Status
OKX-03	Centralization Related Risks	Centralization / Privilege	Major	● Acknowledged
DRW-01	Lack Of Check On Weights	Logical Issue	Minor	● Resolved
DRW-02	Lack Of Length Check On Batches	Logical Issue	Minor	● Resolved
DRW-03	Missing Validation That Sum Of <code>batchesAmount</code> Is Equal To <code>baseRequest.fromTokenAmount</code>	Volatile Code	Minor	● Acknowledged
DRW-04	Possible Inconsistency Caused By Check Of <code>baseRequest.fromTokenAmount</code>	Logical Issue, Inconsistency	Minor	● Resolved
DRW-05	Possibly Incorrect Assignment To Variable <code>subIndex</code>	Logical Issue	Minor	● Resolved
EYE-01	Unprotected Initializer	Coding Style	Minor	● Acknowledged
EYE-02	Unchecked ERC-20 <code>transfer()</code> / <code>transferFrom()</code> Call	Volatile Code	Minor	● Acknowledged
EYE-03	Conditional Check On Message Value Should Be Unconditional	Logical Issue	Minor	● Acknowledged
ONF-01	Variable <code>success</code> Is Overwritten	Logical Issue	Minor	● Resolved

ID	Title	Category	Severity	Status
UVA-01	Missing Validation Of Token Order In Function <code>withdraw()</code>	Volatile Code	Minor	● Acknowledged
DRW-06	Choice Of Token For Parameter <code>fromToken</code> For Hops	Logical Issue	Informational	● Acknowledged
DRW-07	Typo In Variable Name	Coding Style	Informational	● Acknowledged
DRW-08	Missing Zero Address Validation	Volatile Code	Informational	● Acknowledged
EYE-04	Unused Event	Coding Style	Informational	● Acknowledged
OKX-01	Unnecessary Receive Function	Logical Issue	Informational	● Acknowledged
OKX-02	Unlocked Compiler Version	Language Specific	Informational	● Acknowledged
ONF-02	Inconsistent Handling Of Other Markets	Logical Issue	Informational	● Acknowledged
ONF-03	Overwriting Failure	Logical Issue	Informational	● Acknowledged
ONF-04	Allowed Markets In Function <code>tradeV2()</code>	Inconsistency	Informational	● Acknowledged
ONF-05	Use Of <code>delegatcall</code>	Logical Issue	Informational	● Acknowledged
ONF-06	Function State Mutability Can Be Restricted To Pure And View	Language Specific	Informational	● Acknowledged
UVA-02	Possible Missing Functionality From <code>BaseAdapter</code>	Inconsistency	Informational	● Acknowledged
UVA-03	Possibly Incorrect FeeRate In Contract UniswapV2AdapterMain	Inconsistency	Informational	● Acknowledged
UVA-04	Reserves Are Used In Calculation Of Liquidity Removal	Logical Issue	Informational	● Acknowledged

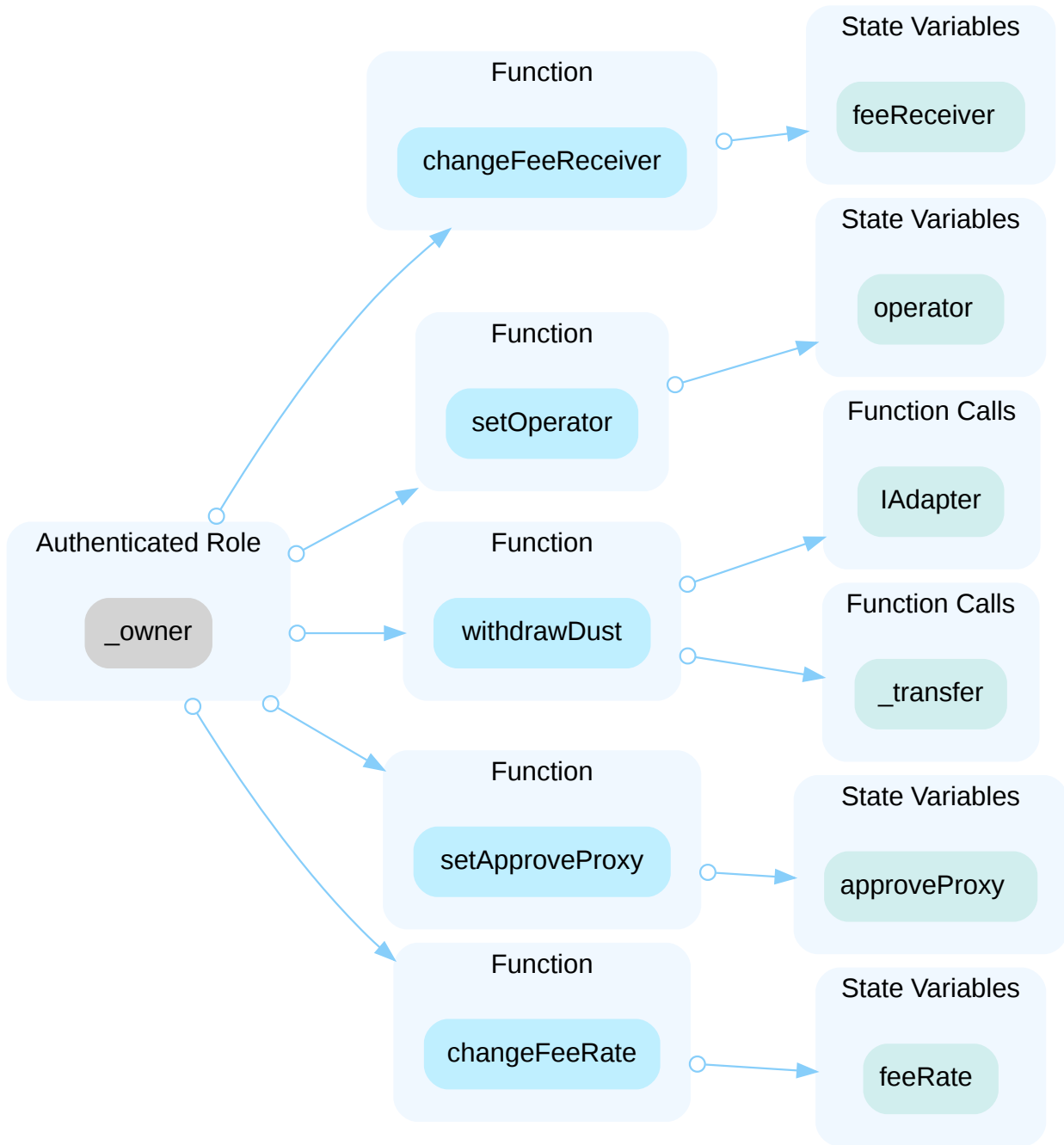
ID	Title	Category	Severity	Status
UVA-05	Inaccurate Require Statement	Logical Issue	Informational	● Acknowledged
UVA-06	Missing Validation Of Tokens In Function <code>amountToShare()</code>	Volatile Code, Inconsistency	Informational	● Acknowledged

OKX-03 | CENTRALIZATION RELATED RISKS

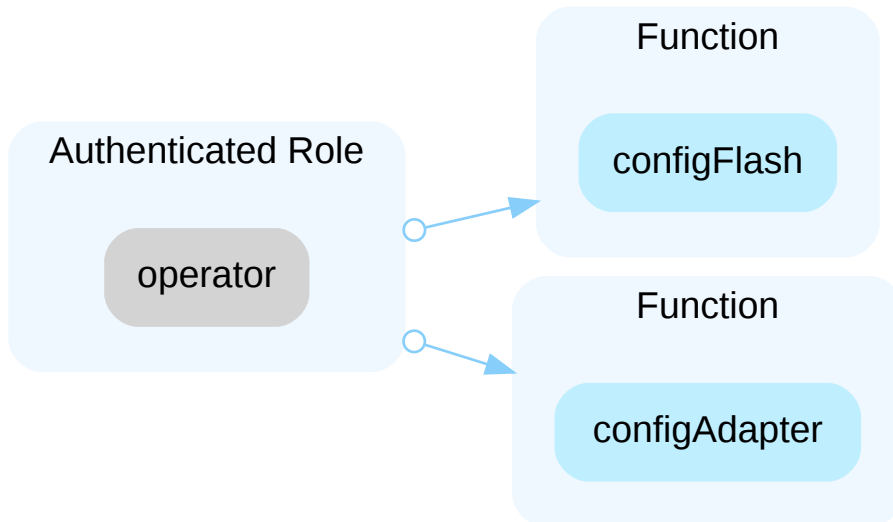
Category	Severity	Location	Status
Centralization / Privilege	● Major	OkxNFTMarketAggregator.sol (OkxNFTMarketAggregator.sol): 101~105, 109; DexRouter.sol (DexRouter.sol): 340, 346, 352, 372~378, 384~390, 396~401, 511, 515, 528~532; Entrance.sol (Entrance.sol and UniswapV2AdapterMain): 109, 118, 127, 135, 144, 164, 176, 283	● Acknowledged

Description

In the contract `Entrance` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and configure key parameters and manipulate the project.



In the contract `Entrance` the role `operator` has authority over the functions shown in the diagram below. Any compromise to the `operator` account may allow the hacker to take advantage of this authority and manipulate the project.



In addition, the contract `Entrance` is an upgradeable contract, meaning the owner can upgrade the contract without the community's commitment. If an attacker compromises the account, the attacker can change the implementation of the contract and drain tokens from the contract.

In the contract `DexRouter` the role `owner` has authority over the following functions:

- `setApproveProxy()` to set `approveProxy`
- `setWNativeRelayer()` to set the `wNativeRelayer`
- `setXBridge()` to set the `xBridge`
- `initializePMMRouter()` to initialize the PMM Router
- `setPMMFeeConfig()` to set the fee rate and receiver of the PMM Router

In addition, the role `xBridge` has authority over the following functions:

- `smartSwapByOrderIdByXBridge()` to perform the token swaps with the `smartswap`
- `unxswapByOrderIdByXBridge()` to perform the token swaps with `unxswap`
- `uniswapV3SwapToByXBridge()` to perform the token swaps with `uniswapV3`
- `PMMV2SwapByXBridge()` to perform the token swaps with `PMMV2Swap`

In addition, the contract `DexRouter` is an upgradeable contract, meaning the owner can upgrade the contract without the community's commitment. If an attacker compromises the account, the attacker can change the implementation of the contract and drain tokens from the contract.

In the contract `OkxNFTMarketAggregator` the role `owner` has authority over the following functions:

- `approveERC20()` to approve the ERC20 tokens to the operator
- `setMarketRegistry()` to set the `marketRegistry`

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multisignature wallets. Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign ($2/3$, $3/5$) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.
AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles.
OR
- Remove the risky functionality.

Alleviation

[OKX Team, 05/04/2023] :

At present, the team do have measures to reduce security risks, such as upgrading a dedicated isolated signature machine. The upgrade is the use of a professional signature machine to send upgrade transactions. We use a multi-signature mechanism to execute protocol parameters, modify important protocol parameters, and have a dedicated service for monitoring important protocol parameters, which is also about to be launched.

[CertiK, 05/04/2023] :

To mitigate this issue, the team is planning to adopt multi-signature mechanism to execute protocol parameters and the status will be updated after deployment.

Due to the company's policy, we cannot mark this issue as resolved unless the ownership is renounced or the privileged functions are removed. However, after deployment, we can check to see if the multisig wallet that has been provided is integrated and then classify this issue as Mitigated.

DRW-01 | LACK OF CHECK ON WEIGHTS

Category	Severity	Location	Status
Logical Issue	● Minor	DexRouter.sol (DexRouter.sol): 102	● Resolved

Description

In the function `_exeForks()`, multiple swaps are performed, with each swap swapping `batchAmount * weight / 10000` tokens.

```
86  function _exeForks(address payer, uint256 batchAmount, RouterPath calldata
path) private {
87      address fromToken = bytes32ToAddress(path.fromToken);
88
89      // execute multiple Adapters for a transaction pair
90      uint256 pathLength = path.mixAdapters.length;
91      for (uint256 i = 0; i < pathLength; ) {
92          bytes32 rawData = bytes32(path.rawData[i]);
93          address poolAddress;
94          bool reserves;
95          uint256 weight;
96          assembly {
97              poolAddress := and(rawData, _ADDRESS_MASK)
98              reserves := and(rawData, _REVERSE_MASK)
99              weight := shr(160, and(rawData, _WEIGHT_MASK))
100          }
101          require(weight >= 0 && weight <= 10000, "weight out of range");
102          uint256 _fromTokenAmount = (batchAmount * weight) / 10000;
103
104          _transferInternal(payer, path.assetTo[i], fromToken, _fromTokenAmount);
```

Since each swap swaps a portion of `batchAmount`, it is expected for all of `batchAmount` to be used. This would require the sum of all weights to be 10000, but no such check exists, allowing the possibility that some tokens are unused or tokens meant for a different batch are used.

Recommendation

Recommend adding a check that requires the sum of all weights to be 10000.

Alleviation

[OKX Team, 04/25/2023]:

The team heeded the advice and resolved the finding by adding an extra check that the total weight does not exceed 10000.

The change is reflected in the commit [684a3cd4b746c8a28941bcdac878c819932ea2e0](#).

DRW-02 | LACK OF LENGTH CHECK ON BATCHES

Category	Severity	Location	Status
Logical Issue	● Minor	DexRouter.sol (DexRouter.sol): 261~262	● Resolved

Description

The function `_smartSwapInternal()` simultaneously iterates over the arrays `batchesAmount` and `batches` to be used when calling `_exeHop()`.

```
316     for (uint256 i = 0; i < batches.length; ) {
317         // execute hop, if the whole swap replacing by pmm fails, the funds will
return to dexRouter
318         _exeHop(payer, batchesAmount[i], batches[i], extraData);
319         unchecked {
320             ++i;
321         }
322     }
```

This suggests that both arrays `batchesAmount` and `batches` have to be the same length, but there is no check that this is the case.

Recommendation

Recommend adding a check ensuring that `batchesAmount` and `batches` have the same length.

Alleviation

[OKX Team, 04/25/2023]:

The team heeded the advice and resolved the finding by adding a check to ensure both arrays `batchesAmount` and `batches` have the same length. The change is reflected in the commit [5033a843f434523a2b609b503eadaafe07913207](#).

DRW-03 | MISSING VALIDATION THAT SUM OF `batchesAmount` IS EQUAL TO `baseRequest.fromTokenAmount`

Category	Severity	Location	Status
Volatile Code	● Minor	DexRouter.sol (DexRouter.sol): 447-449	● Acknowledged

Description

In the function `smartSwapByInvest()`, the `fromToken` balance of the contract is redistributed proportionally with the ratio `batchesAmount[i] / baseRequest.fromTokenAmount` as shown in line 447-449.

```
446     uint256[] memory newBatchesAmount = new uint256[](batchesAmount.length);
447     for (uint256 i = 0; i < batchesAmount.length; ++i) {
448         newBatchesAmount[i] = batchesAmount[i] * amount /
baseRequest.fromTokenAmount;
449     }
```

That means the sum of `batchesAmount[i]` should be the same as `baseRequest.fromTokenAmount`, but there is no such validation in the current implementation of `smartSwapByInvest()`.

Recommendation

Recommend adding an extra check to ensure the sum of `batchesAmount[i]` is the same as `baseRequest.fromTokenAmount`.

Alleviation

[OKX Team, 04/25/2023]:

The team acknowledged the finding and decided not to make any changes to the current version.

DRW-04 | POSSIBLE INCONSISTENCY CAUSED BY CHECK OF `baseRequest.fromTokenAmount`

Category	Severity	Location	Status
Logical Issue, Inconsistency	● Minor	DexRouter.sol (DexRouter.sol): 544~547	● Resolved

Description

The function `PMMV2SwapByInvest()` invokes the `_PMMV2Swap()` to perform a token swap via the private market maker.

```
538 function PMMV2SwapByInvest(  
539     address receiver,  
540     PMMLib.PMMBaseRequest memory baseRequest,  
541     PMMLib.PMMSwapRequest calldata request  
542 ) external payable nonReentrant returns (uint256 returnAmount) {  
543     require(request.fromToken != _ETH, "Invalid source token");  
544     if (baseRequest.fromTokenAmount == 0) {  
545         baseRequest.fromTokenAmount =  
IERC20(request.fromToken).balanceOf(address(this));  
546     }  
547     return _PMMV2Swap(address(this), receiver, baseRequest, request);  
548 }
```

The argument `fromTokenPayer` is set as the contract address in line 547 of function `PMMV2SwapByInvest()`.

```
function _PMMV2Swap(  
    address fromTokenPayer,  
    address receiver,  
    PMMLib.PMMBaseRequest memory baseRequest,  
    PMMLib.PMMSwapRequest calldata request  
) internal returns (uint256 returnAmount) {  
    ...  
}
```

However, the `baseRequest.fromTokenAmount` is updated to the `IERC20(request.fromToken).balanceOf(address(this))` only if the input `baseRequest.fromTokenAmount == 0`. In the case that the `baseRequest.fromTokenAmount` is nonzero, it will not be updated. Considering the case the `baseRequest.fromTokenAmount` is larger than `IERC20(request.fromToken).balanceOf(address(this))`, the swap will fail due to insufficient funds in the contract when line 175 of function `_pmmSwapInternal()` is executed.

`_pmmSwapInternal()` in [Web3-DEX-dev/contracts/8/PMRouter.sol](#)

```
171     if (fromNative) {
172         IWETH(_WETH).deposit{ value: actualAmountRequest }();
173         IERC20(_WETH).safeTransfer(request.payer, actualAmountRequest);
174     } else if(fromTokenPayer == address(this)) {
175         IERC20(request.fromToken).safeTransfer(request.payer,
actualAmountRequest);
176     } else {
177         IApproveProxy(_APPROVE_PROXY).claimTokens(request.fromToken,
fromTokenPayer, request.payer, actualAmountRequest);
178     }
```

Recommendation

Recommend removing the check of `baseRequest.fromTokenAmount == 0` so that the `baseRequest.fromTokenAmount` will be updated under any circumstances.

Alleviation

[OKX Team, 04/25/2023]:

The team heeded the advice and resolved the finding by removing the check `baseRequest.fromTokenAmount == 0`. The change is reflected in the commit [71ad0a2c2fef629c41682822990452b5d9a4ad2e](#).

DRW-05 | POSSIBLY INCORRECT ASSIGNMENT TO VARIABLE

subIndex

Category	Severity	Location	Status
Logical Issue	● Minor	DexRouter.sol (DexRouter.sol): 203	● Resolved

Description

In the function `_tryPmmSwap()`, the variable `subIndex` is assigned the length of the `bytes` array `pmmRequest.extension`.

```
202     assembly{
203         subIndex := mload(add(extension, 0x0))
204     }
```

The operation `add(extension, 0x0)` is redundant meaning `mload(extension)` will produce the same result.

Recommendation

The auditing team would like to know if `subIndex` is supposed to be an element of `pmmRequest.extension`. For example, if `subIndex` is meant to be the first 32 bytes, then the operation should instead be `add(extension, 0x20)`.

Alleviation

[OKX Team, 04/25/2023]:

The team resolved the finding by changing the position from `0x0` to `0x20`. The change is reflected in the commit

[ea178623d4bd0f22873aa39ec0049cf6640999f5](#).

EYE-01 | UNPROTECTED INITIALIZER

Category	Severity	Location	Status
Coding Style	● Minor	Entrance.sol (Entrance.sol and UniswapV2AdapterMain): 73	● Acknowledged

Description

One or more logic contracts do not protect their initializers. An attacker can call the initializer and assume ownership of the logic contract, whereby she can perform privileged operations that trick unsuspecting users into believing that she is the owner of the upgradeable contract.

```
18 contract Entrance is InvestBase, OwnableUpgradeable, ReentrancyGuardUpgradeable
{
```

- `Entrance` is an upgradeable contract that does not protect its initializer.

```
73     function initialize(IWETH _weth) public initializer {
```

- `initialize` is an unprotected initializer function.

Recommendation

We advise calling `_disableInitializers` in the constructor or giving the constructor the `initializer` modifier to prevent the initializer from being called on the logic contract.

Reference: https://docs.openzeppelin.com/upgrades-plugins/1.x/writing-upgradeable#initializing_the_implementation_contract

Alleviation

[OKX Team, 04/24/2023]:

The team acknowledged the finding and decided not to make any changes to the current version.

EYE-02 | UNCHECKED ERC-20 `transfer()` / `transferFrom()` CALL

Category	Severity	Location	Status
Volatile Code	● Minor	Entrance.sol (Entrance.sol and UniswapV2AdapterMain): 352	● Acknowledged

Description

The return value of the `transfer()/transferFrom()` call is not checked.

```
352         weth.transfer(to, amount);
```

Recommendation

Since some ERC-20 tokens return no values and others return a `bool` value, they should be handled with care. We advise using the [OpenZeppelin's SafeERC20.sol](#) implementation to interact with the `transfer()` and `transferFrom()` functions of external ERC-20 tokens. The OpenZeppelin implementation checks for the existence of a return value and reverts if `false` is returned, making it compatible with all ERC-20 token implementations.

Alleviation

[OKX Team, 04/24/2023] :

The team acknowledged the finding and plan to fix the issue in the future.

EYE-03 | CONDITIONAL CHECK ON MESSAGE VALUE SHOULD BE UNCONDITIONAL

Category	Severity	Location	Status
Logical Issue	● Minor	Entrance.sol (Entrance.sol and UniswapV2AdapterMain): 382	● Acknowledged

Description

The function `_transferTokens()` has a check to see if `msg.value` matches the amount of native tokens that have been transferred to other addresses.

```
382         if (msg.value != 0) {require(msg.value == nativeAmount,
MetaXInvestErrors.QUANTITY_MISMATCH);}
```

However, this check is only performed if `msg.value != 0`, when it should be always done.

If `msg.value == 0` but `nativeAmount > 0`, it is possible for the contract to lose native currency that was stored within it before the current `msg.sender` interacted with it. Since this contract has a `receive()` function, it is expected that it will hold native currency, which can be lost due to the above.

Recommendation

Recommend always ensuring that `msg.value == nativeAmount`.

Alleviation

[OKX Team, 04/24/2023]:

The team acknowledged the finding and plan to fix the issue in the future.

ONF-01 | VARIABLE `success` IS OVERWRITTEN

Category	Severity	Location	Status
Logical Issue	● Minor	OkxNFTMarketAggregator.sol (OkxNFTMarketAggregator.sol): 523~528, 553, 574, 588~594, 607, 624	● Resolved

Description

In both functions `_performERC1155Transfer()` and `_performERC721Transfer()`, the variable `success` is assigned with 0 if the passed `token` is not a contract. After that, the `success` is assigned with the return value of the `call()`, which means the `success` is overwritten by the return value of `call()`.

In the case that the passed `token` is not a contract, it should return false immediately and should not continue the execution of the remaining code.

Recommendation

Recommend returning the value of `success` if it is false.

Alleviation

[OKX Team, 04/26/2023]:

The team resolved the finding by using `gt(extcodesize(token), 0)` instead of `iszero(extcodesize(token))` and removing the variable `success` from the first `if` branch to ensure the variable `success` will only be assigned with the return value of function `call()`. The change is reflected in the commit [401c90c5dec1fd313baa8c91125e25f8d035fd1e](#).

UVA-01 | MISSING VALIDATION OF TOKEN ORDER IN FUNCTION `withdraw()`

Category	Severity	Location	Status
Volatile Code	● Minor	adapters/uniswap/UniswapV2AdapterMain.sol (Entrance.sol and UniswapV2AdapterMain): 52	● Acknowledged

Description

In the implementation of Uniswap V2, the function `IUniswapV2Pair(pool).getReserves()` retrieves the reserves of tokens in which the first reserve corresponds to the token0 and the second one is the token1. The token0 and token1 are recorded according to the rule that `address(token0) < address(token1)`.

In the function `withdraw()` and `_withdraw()`, the passed `tokensOut` does not validate whether the tokens are in the order that `address(tokensOut[0].token) < address(tokensOut[1].token)`. If one of them is a zero address, it needs to be converted to the weth address first.

```
50     function withdraw(  
51         address pool,  
52         TokenOutInfo[] calldata tokensOut,  
53         bytes calldata data  
54     ) external override {  
55  
56         // Check Length  
57         require(tokensOut.length == 2, 'Length Wrong');  
58  
59         // Decode feeRate+mode from data  
60         // mod=2 for withdraw two tokens;mod=0 for withdraw token0;mod=1 for  
withdraw token1  
61         (uint256 feeRate, uint256 mode) = _getFeeRate(pool, data);  
62  
63         // Read Status  
64         uint256 tokenAmount = _balanceOf(pool, address(this));  
65  
66         // Transfer to Pool  
67         _transfer(pool, pool, tokenAmount);  
68  
69         // remove liquidity  
70         (uint256 amount0, uint256 amount1) =  
IUniswapV2Pair(pool).burn(address(this));  
71  
72         _withdraw(pool, feeRate, mode, tokensOut, amount0, amount1);  
73  
74         // Event  
75         emit Withdraw(pool, tokenAmount);  
76  
77     }
```

In this case, it will revert or transfer the wrong token amounts if the tokens are flipped.

Recommendation

Recommend adding an extra check that the token order matches the tokens in the pool.

Alleviation

[OKX Team, 04/24/2023] :

The team acknowledged the finding and plan to fix the issue in the future.

In the future, we will update with a fix, but we currently make sure that the `tokensOut` tokens are in the appropriate order.

DRW-06 | CHOICE OF TOKEN FOR PARAMETER `fromToken` FOR HOPS

Category	Severity	Location	Status
Logical Issue	● Informational	DexRouter.sol (DexRouter.sol): 124	● Acknowledged

Description

In the function `_exeHop()`, the first hop executed swaps `batchAmount` of `hops[0].fromToken`. It should be the case that `hops[0].fromToken` is the same address as `_baseRequest.fromToken` in `_smartSwapInternal()`, but there are no checks to guarantee this.

Recommendation

It is recommended to add a check ensuring that the `fromToken` for hops are the appropriate address.

Alleviation

[OKX Team, 04/25/2023]:

The team acknowledged the finding and decided not to make any changes to the current version.

DRW-07 | TYPO IN VARIABLE NAME

Category	Severity	Location	Status
Coding Style	● Informational	DexRouter.sol (DexRouter.sol): 94, 98, 106	● Acknowledged

Description

The variable `reserves` passed in the function `_exeForks()` seems to be `reverse` or `reversed` based on the following observation:

- `reserves` is assigned with the value `and(rawData, _REVERSE_MASK)`, as the constant name `_REVERSE_MASK` indicates;
- when `reserves` is true, it invokes the `IAdapter(path.mixAdapters[i]).sellQuote()` to sell the token1 in the pool pair; Otherwise, it calls `IAdapter(path.mixAdapters[i]).sellBase()` to sell the token0 in the pool pair.

The second point implies that it sells the token1 if reversed and it sells the token0 if not reversed.

Recommendation

Recommend correcting the variable name to improve the code readability.

Alleviation

[OKX Team, 04/25/2023] :

The team acknowledged the finding and decided not to make any changes to the current version.

DRW-08 | MISSING ZERO ADDRESS VALIDATION

Category	Severity	Location	Status
Volatile Code	● Informational	DexRouter.sol (DexRouter.sol): 340, 346, 352	● Acknowledged

Description

Addresses should be checked before assignment to ensure they are not zero addresses.

Recommendation

Recommend adding a check that the passed address is not a zero address.

Alleviation

[OKX Team, 04/25/2023] :

The team acknowledged the finding and decided not to make any changes to the current version.

EYE-04 | UNUSED EVENT

Category	Severity	Location	Status
Coding Style	● Informational	Entrance.sol (Entrance.sol and UniswapV2AdapterMain): 8 5	● Acknowledged

Description

The event `SetIfCheck(bool ifCheck)` is declared in the contract `Entrance`, but it has never been used in the codebase.

Recommendation

Recommend removing the unused event `SetIfCheck()`.

Alleviation

[OKX Team, 04/24/2023]:

The team acknowledged the finding and plan to fix the issue in the future.

OKX-01 | UNNECESSARY RECEIVE FUNCTION

Category	Severity	Location	Status
Logical Issue	● Informational	OkxNFTMarketAggregator.sol (OkxNFTMarketAggregator.sol): 96; DexRouter.sol (DexRouter.sol): 27; Entrance.sol (Entrance.sol and UniswapV2AdapterMain): 482; adapters/uniswap/UniswapV2AdapterMain.sol (Entrance.sol and UniswapV2AdapterMain): 469	● Acknowledged

Description

All contracts in the auditing scope contain a `receive()` function that accepts direct transfers of the native token. However, there is currently no purpose for the contracts to hold native tokens and it is possible for users to take these tokens.

Each contract contains mechanisms for a user to take any native tokens held by the contract:

- `DexRouter` performs arbitrary swaps and performs no checks on `msg.value`, allowing a user to swap native tokens inside the contract;
- `OkxNFTMarketAggregator` returns unused ETH, but performs no checks on `msg.value`, allowing a user to have native tokens inside the contract be viewed as unused ETH;
- `Entrance` transfers tokens to arbitrary addresses in `_transferToken()` and the `msg.value` check in `_transferTokens()` can be bypassed if `msg.value == 0`, allowing the transfer of native tokens held inside the contract;
- `UniswapV2AdapterMain` expects users to transfer tokens to the contract and deposit them, allowing users to deposit native tokens already held inside the contract.

If the transfer of native tokens are meant to be used with a `payable` function, then it would be better to ensure `msg.value` is correct instead of having a `receive()` function.

Recommendation

Recommend removing the `receive()` function if it is not needed by the project.

Alleviation

[OKX Team, 05/16/2023] :

The team acknowledged the finding and will not make any changes to the codebase.

OKX-02 | UNLOCKED COMPILER VERSION

Category	Severity	Location	Status
Language Specific	● Informational	OkxNFTMarketAggregator.sol (OkxNFTMarketAggregator.sol): 2; DexRouter.sol (DexRouter.sol): 2	● Acknowledged

Description

The contracts cited have an unlocked compiler version. An unlocked compiler version in the source code of the contract permits the user to compile it at or above a particular version. This, in turn, leads to differences in the generated bytecode between compilations due to differing compiler version numbers. This can lead to ambiguity when debugging, as compiler specific bugs may occur in the codebase that would be hard to identify over a span of multiple compiler versions rather than a specific one.

Recommendation

We recommend the compiler version is instead locked at the lowest version possible that the contract can be compiled at. For example, for version `v0.8.2` the contract should contain the following line:

```
pragma solidity 0.8.2;
```

Alleviation

[OKX Team, 05/16/2023] :

The team acknowledged the finding and will not make any changes to the codebase.

ONF-02 | INCONSISTENT HANDLING OF OTHER MARKETS

Category	Severity	Location	Status
Logical Issue	● Informational	OkxNFTMarketAggregator.sol (OkxNFTMarketAggregator.sol): 114, 213	● Acknowledged

Description

In the two functions labeled `trade()`, there is a check on the provided `tradeData` that the `orderToAddress` is the `msg.sender`. However, this check is done for some of the markets instead of all of them.

Recommendation

The auditing team would like to know if this check is meant to be excluded for other markets.

Alleviation

[OKX Team, 04/26/2023]:

The team acknowledged the finding and decided not to make any changes to the current version.

ONF-03 | OVERWRITING FAILURE

Category	Severity	Location	Status
Logical Issue	● Informational	OkxNFTMarketAggregator.sol (OkxNFTMarketAggregator.sol): 340, 355, 364	● Acknowledged

Description

In the function `tradeV2()`, low-level calls are performed in three places:

1. `beforeExecute()`, which transfers tokens from `msg.sender` to `address(this)`
2. During a call to `SEAPORT`
3. `afterExecute()`, which transfers tokens from `address(this)` to `msg.sender`

The only way for the process to revert is during 2) if `isAtomic == true` and the call to `SEAPORT` fails and returns data. Hence, it is possible for 1) or 2) to fail but 3) succeeds. Note that 3) returns `true` even if no low-level calls are made, for example, using an unsupported `actionType` will have 3) return `true`.

Recommendation

The auditing team would like to confirm that failed calls in 1) and 2) are allowed to be overwritten by a successful call in 3) or if no calls are performed in 3).

Alleviation

[OKX Team, 04/26/2023]:

The team acknowledged the finding. It is currently allowed and will be refactored in the next version.

ONF-04 | ALLOWED MARKETS IN FUNCTION `tradeV2()`

Category	Severity	Location	Status
Inconsistency	● Informational	OkxNFTMarketAggregator.sol (OkxNFTMarketAggregator.sol): 312~313	● Acknowledged

Description

The function `tradeV2()` allows the users to trade NFTs with specified markets. The comment in line 312 indicates only the market `wyvern` (id is 4 based on the implementation) is allowed to be used in `tradeV2()`, but the implementation in line 313:

```
313         if(tradeDetails[i].marketId != 0 && tradeDetails[i].marketId !=4 &&
tradeDetails[i].marketId < 7) {
```

also allows the market with id 0.

Recommendation

The auditing team would like to know if only the `wyvern` is allowed or if both market id 0 and 4 (`wyvern`) are allowed.

Alleviation

[OKX Team, 04/26/2023]:

The team acknowledged the finding and confirmed that both market id 0 and 4 are allowed.

ONF-05 | USE OF `delegatecall`

Category	Severity	Location	Status
Logical Issue	● Informational	OkxNFTMarketAggregator.sol (OkxNFTMarketAggregator.sol): 192, 275, 386	● Acknowledged

Description

Anyone is able to use the functions `trade()` and `tradeV2()`, which may make a `delegatecall` to `proxy` with data `tradeData`.

```
191         (bool success, ) = isLib
192         ? proxy.delegatecall(tradeData)
193         : proxy.call{value: ethValue}(tradeData);
```

As `tradeData` is user defined, if `proxy` has functions that can change state variables, state variables of `OkxNFTMarketAggregator`, such as `_owner`, may be changed.

Recommendation

The auditing team would like to know what kind of contract `proxy` will be when making a `delegatecall` to it.

Alleviation

[OKX Team, 04/26/2023]:

The team acknowledged the finding. The proxy is the adapter which is provided by the team.

ONF-06 | FUNCTION STATE MUTABILITY CAN BE RESTRICTED TO PURE AND VIEW

Category	Severity	Location	Status
Language Specific	● Informational	OkxNFTMarketAggregator.sol (OkxNFTMarketAggregator.sol): 32-35, 42-45, 424	● Acknowledged

Description

The functions `bytesToAddress()` and `bytesToBytes4()` do not read any state variable so they can be restricted to pure. In addition, the function `verifywyvern()` does not modify any state variable, so it can be changed to view.

Recommendation

Recommend changing the function state mutability of the aforementioned functions.

Alleviation

[OKX Team, 04/26/2023] :

The team acknowledged the finding and decided not to make any changes to the current version.

UVA-02 | POSSIBLE MISSING FUNCTIONALITY FROM `BaseAdapter`

Category	Severity	Location	Status
Inconsistency	● Informational	adapters/uniswap/UniswapV2AdapterMain.sol (Entrance.sol and UniswapV2AdapterMain): 10	● Acknowledged

Description

The `UniswapV2AdapterMain` contract inherits the `BaseAdapter` abstract contract which contains various functions that perform no operations. `UniswapV2AdapterMain` has overwritten some of the functions but not all of them, namely `unstake()`, `unStakeAndWithdraw()`, and `claimReward()`.

Recommendation

It is recommended to implement all functions.

Alleviation

[OKX Team, 04/24/2023]:

The team acknowledged the finding and decided not to make any changes to the current version. The `UniswapV2AdapterMain` contract does not use all of the above functions.

UVA-03 | POSSIBLY INCORRECT FEERATE IN CONTRACT UNISWAPV2ADAPTERMAIN

Category	Severity	Location	Status
Inconsistency	● Informational	adapters/uniswap/UniswapV2AdapterMain.sol (Entrance.s ol and UniswapV2AdapterMain): 165, 168, 205, 215, 324, 336, 387, 403	● Acknowledged

Description

The standard fee charged on the UniswapV2 pool is 3 out of 1000 deducted from the tokens to be swapped. In the contract `UniswapV2AdapterMain`, a custom fee rate is used to calculate `getAmountOut()`, which could result in a different value if the underlying pool uses the standard fee.

Yield-External-Audit/contracts/libraries/MetaXMath.sol

```
74     function getAmountOut(  
75         uint256 amountIn,  
76         uint256 reserveIn,  
77         uint256 reserveOut,  
78         uint256 feeInPrecision,  
79         uint256 precision  
80     ) internal pure returns (uint256 amountOut) {  
81         require(  
82             reserveIn > 0 && reserveOut > 0,  
83             "UniswapV2Library: INSUFFICIENT_LIQUIDITY"  
84         );  
85         if (amountIn > 0) {  
86             uint256 amountInWithFee = amountIn.mul(feeInPrecision);  
87             uint256 numerator = amountInWithFee.mul(reserveOut);  
88             uint256 denominator =  
reserveIn.mul(precision).add(amountInWithFee);  
89             amountOut = numerator / denominator;  
90         }  
91     }
```

Recommendation

Recommend checking if the standard fee matches the custom fee.

Alleviation

[OKX Team, 04/20/2023] :

The team acknowledged the finding and decided not to make any changes to the current version.

UVA-04 | RESERVES ARE USED IN CALCULATION OF LIQUIDITY REMOVAL

Category	Severity	Location	Status
Logical Issue	● Informational	adapters/uniswap/UniswapV2AdapterMain.sol (Entrance.sol and UniswapV2AdapterMain): 152, 158-162	● Acknowledged

Description

The function `shareToAmount()` is used to calculate the withdrawal amounts with the specified LP tokens. In the calculation of the withdrawal amounts of token0 and token1, the reserves of the pool are used instead of the balances of the pool.

```
152     (uint256 r0, uint256 r1, ) = IUniswapV2Pair(pool).getReserves();
153
154     // TotalSupply
155     uint256 totalSupply = _feeLp(r0,r1,pool);
156     require(totalSupply > amount, MetaXInvestErrors.NOT_ENOUGH);
157
158     uint256 amount0 = r0 * amount / totalSupply;
159     uint256 amount1 = r1 * amount / totalSupply;
160
161     r0 = r0 - amount0;
162     r1 = r1 - amount1;
163     ...
```

However, in the implementation of the function `burn()` in the UniswapV2, the balances in the pool are used to ensure pro-rata distribution.

```
function burn(address to) external lock returns (uint amount0, uint amount1) {
    (uint112 _reserve0, uint112 _reserve1,) = getReserves(); // gas savings
    address _token0 = token0; // gas savings
    address _token1 = token1; // gas savings
    uint balance0 = IERC20(_token0).balanceOf(address(this));
    uint balance1 = IERC20(_token1).balanceOf(address(this));
    uint liquidity = balanceOf[address(this)];

    bool feeOn = _mintFee(_reserve0, _reserve1);
    uint _totalSupply = totalSupply; // gas savings, must be defined here since
totalSupply can update in _mintFee
    amount0 = liquidity.mul(balance0) / _totalSupply; // using balances ensures
pro-rata distribution
    amount1 = liquidity.mul(balance1) / _totalSupply; // using balances ensures
pro-rata distribution
    ...
}
```

Recommendation

Recommend using the balances in the function `shareToAmount()` to obtain more accurate results.

Alleviation

[OKX Team, 04/24/2023]:

The team acknowledged the finding and plan to fix the issue in the future.

UVA-05 | INACCURATE REQUIRE STATEMENT

Category	Severity	Location	Status
Logical Issue	● Informational	adapters/uniswap/UniswapV2AdapterMain.sol (Entrance.sol and UniswapV2AdapterMain): 156	● Acknowledged

Description

The function `shareToAmount()` is used to calculate the withdrawal amounts with the specified LP tokens. After the update in the total supply of the LP, the following check is executed:

```
156     require(totalSupply > amount, MetaXInvestErrors.NOT_ENOUGH);
```

According to UniswapV2 design, `MINIMUM_LIQUIDITY = 10**3` LP tokens have been locked. Therefore, a more precise statement is as follows:

```
require(totalSupply > amount + MINIMUM_LIQUIDITY, MetaXInvestErrors.NOT_ENOUGH);
```

Recommendation

Recommend adding the `MINIMUM_LIQUIDITY` in the check if such amount of LP has been locked.

Alleviation

[OKX Team, 04/24/2023] :

The team acknowledged the finding and decided not to make any changes to the current version. This function is only used to query information.

UVA-06 | MISSING VALIDATION OF TOKENS IN FUNCTION `amountToShare()`

Category	Severity	Location	Status
Volatile Code, Inconsistency	● Informational	adapters/uniswap/UniswapV2AdapterMain.sol (Entrance.sol and UniswapV2AdapterMain): 188	● Acknowledged

Description

The function `amountToShare()` is used to calculate the liquidity given the amounts of two tokens. However, there is no check that the passed `tokenAmounts` has length 2 and the token order matches with the tokens in the underlying pool that has the requirement `address(token0) < address(token1)`. In the case that one of them is the eth, it needs to be converted to the weth first.

Recommendation

Recommend adding extra checks to ensure the `tokenAmounts` has length 2 and the token order is properly validated.

Alleviation

[OKX Team, 04/24/2023]:

The team acknowledged the finding and plan to fix the issue in the future.

This function is only used to query. We will make sure that the tokens are in order and we will add the requirement to ensure that the length is 2.

APPENDIX | OKX - AUDIT 3

Finding Categories

Categories	Description
Centralization / Privilege	Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.
Logical Issue	Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.
Volatile Code	Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.
Language Specific	Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.
Coding Style	Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.
Inconsistency	Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different require statements on the input variables than a setter function.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

DISCLAIMER | CERTIK

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK's prior written consent in each instance.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR

UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

CertiK | Securing the Web3 World

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.

