SLOWMIST

Smart Contract Security Audit Report

Table Of Contents

1 Executive Summary

On 2022.11.14, the SlowMist security team received the team's security audit application for BlockDEX, developed the audit plan according to the agreement of both parties and the characteristics of the project, and finally issued the security audit report.

The SlowMist security team adopts the strategy of "white box lead, black, grey box assists" to conduct a complete security test on the project in the way closest to the real attack.

The test method information:

Test method	Description
Black box testing	Conduct security tests from an attacker's perspective externally.
Grey box testing	Conduct security testing on code modules through the scripting tool, observing the internal running status, mining weaknesses.
White box testing	Based on the open source code, non-open source code, to detect whether there are vulnerabilities in programs such as nodes, SDK, etc.

The vulnerability severity level information:

Level	Description
Critical	Critical severity vulnerabilities will have a significant impact on the security of the DeFi project, and it is strongly recommended to fix the critical vulnerabilities.
High	High severity vulnerabilities will affect the normal operation of the DeFi project. It is strongly recommended to fix high-risk vulnerabilities.
Medium	Medium severity vulnerability will affect the operation of the DeFi project. It is recommended to fix medium-risk vulnerabilities.
Low	Low severity vulnerabilities may affect the operation of the DeFi project in certain scenarios. It is suggested that the project team should evaluate and consider whether these vulnerabilities need to be fixed.
Weakness	There are safety risks theoretically, but it is extremely difficult to reproduce in engineering.

Level	Description
Suggestion	There are better practices for coding or architecture.

2 Audit Methodology

The security audit process of SlowMist security team for smart contract includes two steps:

Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using automated analysis tools.

Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.

Following is the list of commonly known vulnerabilities that was considered during the audit of the smart contract:

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

Serial Number	Audit Class	Audit Subclass
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
		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

Audit Version:

Project address:

https://github.com/congo86/ZGX

commit: 043941f5852219523a2cbe097197ca0d9dc821be

Fixed Version:

Project address:

https://github.com/congo86/ZGX

commit: 92ee2c24e09b20a60cb96dbe9e49ce3c7d025aa6

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Missing event record	Others	Suggestion	Fixed
N2	Risk of excessive authority	Authority Control Vulnerability	Medium	Fixed
N3	Redundant code	Others	Suggestion	Fixed
N4	Incorrect variable update	Variable Coverage Vulnerability	High	Fixed
N5	Incorrect variable acquisition	Design Logic Audit	Critical	Fixed

4 Code Overview

4.1 Contracts Description

The main network address of the contract is as follows:

The code was not deployed to the mainnet.

4.2 Visibility Description

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

ZXRouter			
Function Name	Visibility	Mutability	Modifiers
<Receive Ether>	External	Payable	-
<Constructor>	Public	Can Modify State	-
createOrder	External	Payable	ensure
createOrderGasEfficient	External	Payable	ensure
cancelOrder	External	Can Modify State	ensure
getAmountOut	External	Can Modify State	-
getAmountIn	External	Can Modify State	-
exactInput	External	Payable	ensure
exactOutput	External	Payable	ensure
swapTokenForExactTokensSupportingFeeOnTransferToken	External	Can Modify State	ensure

ZXRouter			
settleAllOrders	External	Can Modify State	ensure
settleOrder	External	Can Modify State	ensure
pay	Internal	Can Modify State	-
refundETH	Public	Payable	-
unwrapWETH	Public	Payable	-
uniswapExactInputInternal	Private	Can Modify State	-
uniswapExactOutputInternal	Private	Can Modify State	-
getPool	Private	-	-
getPair	Private	-	-
uniswapV3SwapCallback	External	Can Modify State	-
getV2AmountOut	Public	-	-
getV2AmountIn	Public	-	-

ZXVault			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ZXVaultStorage ERC721
createOrder	External	Can Modify State	nonReentrant
createOrderGasEfficient	External	Can Modify State	nonReentrant
cancelOrder	External	Can Modify State	nonReentrant

ZXVault			
swap	External	Can Modify State	nonReentrant
settleOrder	Public	Can Modify State	nonReentrant
settleOrderByOther	Public	Can Modify State	nonReentrant
_storeCreateOrder1	Private	Can Modify State	-
_storeCreateOrder2	Private	Can Modify State	-
_storeCancelOrder	Private	Can Modify State	-
_storeSwap	Private	Can Modify State	-
_settle100FillAmounts	Private	Can Modify State	-
_storeSettleOrder	Private	Can Modify State	-
_reset	Private	Can Modify State	-
getAmountOut	External	-	-
getAmountIn	External	-	-
get100SettleAmounts	External	-	-
getPrices	External	-	-
_getMaxSwapAmountOut	Private	Can Modify State	-
_get100FillAmounts	Private	-	-
_getCumulCancelAmountIn	Private	-	-
_getAmountOut	Private	-	-
_getAmountIn	Private	-	-
_getPrices	Private	-	-

ZXVault			
_getBalance	Private	-	-
_getV3MaxSwapAmountOut	Internal	Can Modify State	-
_getV2MaxSwapAmountOut	Internal	-	-
uniswapV3SwapCallback	External	-	-

ZXVaultStorage			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
<Receive Ether>	External	Payable	-
initialize	External	Can Modify State	initializer

ZXZoo			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
vaultsLength	External	-	-
createVaults	External	Can Modify State	-
_createVault	Private	Can Modify State	-
setAdmin	External	Can Modify State	onlyAdmin
setVaultImplementation	External	Can Modify State	onlyAdmin
getTokenOrder	Internal	-	-

ZXERC20			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	ERC20
mint	External	Can Modify State	-
burn	External	Can Modify State	-
burnFrom	External	Can Modify State	-
setMinter	External	Can Modify State	-

4.3 Vulnerability Summary

[N1] [Suggestion] Missing event record

Category: Others

Content

In the ZXZoo contract, the admin role can change the address of vaultImplementation and the address of admin, but the event log is missing here.

Code location: contracts/ZXZoo.sol#L68-L77

```
    function setAdmin(address _admin) external onlyAdmin {
        admin = _admin;
    }

    function setVaultImplementation(address _vaultImplementation)
        external
        onlyAdmin
    {
        vaultImplementation = _vaultImplementation;
    }
```

Solution

It is recommended to add corresponding event records.

Status

Fixed

[N2] [Medium] Risk of excessive authority

Category: Authority Control Vulnerability

Content

In the ZXZoo contract, the admin role can change the address of vaultImplementation. If admin privileges are stolen (e.g. private key compromise), this could result in vault implementation contracts being modified to malicious contracts, putting user assets at risk.

Code location: contracts/ZXZoo.sol#L72-77

```
    function setVaultImplementation(address _vaultImplementation)
        external
        onlyAdmin
    {
        vaultImplementation = _vaultImplementation;
    }
```

Solution

It is recommended to use multi-signature to manage the admin role.

Status

Fixed

[N3] [Suggestion] Redundant code

Category: Others

Content

In the ZXVault contract, the settleOrderByOther function is not used in other contracts. And if the function is called by an external user, it will fail because the nft corresponding to the order is missing in the vault contract.

Code location: contracts/ZXVault.sol#L214-219

```
    function settleOrderByOther(
        address owner,
        uint256 tokenID
    ) public virtual override nonReentrant {
        _settle100FillAmounts(owner, owner, tokenID);
    }
```

Solution

It is recommended to remove the redundant function.

Status

Fixed

[N4] [High] Incorrect variable update

Category: Variable Coverage Vulnerability

Content

In the ZXVault contract, the _storeCancelOrder function will update the value of the cancelAmountInMap variable when the order is cancelled. But here the value of cancelAmountInMap is updated using = instead of +=, causing the newly passed amountIn parameter to overwrite the old value instead of superimposing. It will eventually affect the result of _get100FillAmounts function calculation is unexpected.

Code location: contracts/ZXVault.sol#L254

```
    function _storeCancelOrder(
        uint256 tokenID,
        uint256 orderAmountIn,
        uint256 amountIn
    ) private returns (bool burned) {
        cumulCancelAmountIn += uint112(amountIn);
        cancelAmountInMap[tokenID] = uint112(amountIn);
        ...
    }
```

Solution

It is recommended to modify `cancelAmountInMap[tokenID] = uint112(amountIn);` to `cancelAmountInMap[tokenID] += uint112(amountIn);`

Status

Fixed

[N5] [Critical] Incorrect variable acquisition

Category: Design Logic Audit

Content

In the ZXRouter contract, when the exactOutput and swapTokenForExactTokensSupportingFeeOnTransferToken functions are called and isTokenInWETH && balanceInETH >= amountIn is false, the amountIn variable is calculated based on the before and after balance of the passed params.to, but since the passed params.tokenIn tokens are passed from msg.sender is passed to the vault contract, so when msg.sender is not equal to params.to, the amountIn will be calculated to zero and the swap function is not executed, but the assets of msg.sender are consumed.

Code location: contracts/ZXRouter.sol#L234

```
    function exactOutput(ExactOutputParams calldata params) external payable override
ensure(params.deadline) {
        ...
        if (isTokenInWETH && balanceInETH >= amountIn) {
            IWETH(WETH).deposit{value: amountIn}();
            IWETH(WETH).transfer(vault, amountIn);
            amountIn = balanceInETH - address(this).balance;
            balanceInETH = address(this).balance;
        } else {
            balanceInToken = IERC20(params.tokenIn).balanceOf(params.to);
            IERC20(params.tokenIn).safeTransferFrom(msg.sender, vault, amountIn);
            amountIn = balanceInToken - IERC20(params.tokenIn).balanceOf(params.to);
            balanceInToken = IERC20(params.tokenIn).balanceOf(params.to);
        }
        ...
    }
```

contracts/ZXRouter.sol#L277

```
    function swapTokenForExactTokensSupportingFeeOnTransferToken(
        ExactOutputParams calldata params
    ) external override ensure(params.deadline) {
        ...
        if (isTokenInWETH && balanceInETH >= amountIn) {
            IWETH(WETH).deposit{value: amountIn}();
            IWETH(WETH).transfer(vault, amountIn);
            amountIn = balanceInETH - address(this).balance;
            balanceInETH = address(this).balance;
        } else {
            balanceInToken = IERC20(params.tokenIn).balanceOf(params.to);
            IERC20(params.tokenIn).safeTransferFrom(msg.sender, vault, amountIn);
            amountIn = balanceInToken - IERC20(params.tokenIn).balanceOf(params.to);
            balanceInToken = IERC20(params.tokenIn).balanceOf(params.to);
        }
        ...
    }
```

Solution

It is recommended to use the balance before and after msg.sender instead of params.to to calculate the amountIn when calling the exactOutput and swapTokenForExactTokensSupportingFeeOnTransferToken functions if isTokenInWETH && balanceInETH >= amountIn is false.

Status

Fixed

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002211240001	SlowMist Security Team	2022.11.14 - 2022.11.24	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 critical risk, 1 high risk, 1 medium risk and 2 suggestion vulnerabilities. All the findings were fixed. The code was not deployed to the mainnet.

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