Smart Contract Security Audit Report

[2021]

Table Of Contents

1 Executive Summary

On 2021.06.15, the SlowMist security team received the WaultFinance team's security audit application for WaultFinance-WexPolyMaster, 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 party should evaluate and consider whether these vulnerabilities need to be fixed.
Weakness	There are safety risks theoretically, but it is extremely difficult to reproduce in engineering.

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

2 Audit Methodology

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

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

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

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

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

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

3 Project Overview

3.1 Project Introduction

Audit version:

WexMaster :

https://github.com/WaultFinance/WAULT/blob/master/contracts/WexPolyMaster.sol

commit: 1fbc64acfb9a263f0ce6f7d94f4d258fc24b1b4a

3.2 Vulnerability Information

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

NO	Title	Category	Level	Status
N1	Excessive Authority Issue	Others	Low	Confirming

4 Code Overview

4.1 Contracts Description

The main network address of the contract is as follows:

WexPolyMaster:

https://polygonscan.com/address/0xC8Bd86E5a132Ac0bf10134e270De06A8Ba317BFe#code

4.2 Visibility Description

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

WexPolyMaster			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
poolLength	External	-	-
getMultiplier	Public	-	-
add	Public	Can Modify State	onlyOwner
set	Public	Can Modify State	onlyOwner
pendingWex	External	-	-
massUpdatePools	Public	Can Modify State	-
updatePool	Public	Can Modify State	-
deposit	Public	Can Modify State	-
withdraw	Public	Can Modify State	-
emergencyWithdraw	Public	Can Modify State	-
claim	Public	Can Modify State	-

WexPolyMaster			
safeWexTransfer	Internal	Can Modify State	-
setWexPerBlock	Public	Can Modify State	onlyOwner

4.3 Vulnerability Summary

[N1] [Low] Excessive Authority Issue

Category: Others

Content

The owner can set the value of wexPerBlock arbitrarily, which will affect the profit of wexReward, and there is no limit on the value range of wexPerBlock, and there is a issue of excessive authority.

https://polygonscan.com/address/0xC8Bd86E5a132Ac0bf10134e270De06A8Ba317BFe#code

```
function setWexPerBlock(uint256 _wexPerBlock) public onlyOwner {
    require(_wexPerBlock > 0, "!wexPerBlock-0");
    wexPerBlock = _wexPerBlock;
}
```

Owner can add pool, can set the allocPoint of pool, there is a issue of selfish mining.

```
function add(
        uint256 _allocPoint,
        IERC20 _lpToken,
        bool _withUpdate
    ) public onlyOwner {
        if (_withUpdate) {
            massUpdatePools();
        }
        uint256 lastRewardBlock = block.number > startBlock
            ? block.number
            : startBlock;
        totalAllocPoint = totalAllocPoint.add(_allocPoint);
        poolInfo.push(
```

```
            PoolInfo({
                lpToken: _lpToken,
                allocPoint: _allocPoint,
                lastRewardBlock: lastRewardBlock,
                accWexPerShare: 0
            })
        );

    function set(
        uint256 _pid,
        uint256 _allocPoint,
        bool _withUpdate
    ) public onlyOwner {
        if (_withUpdate) {
            massUpdatePools();
        }
        totalAllocPoint = totalAllocPoint.sub(poolInfo[_pid].allocPoint).add(
            _allocPoint
        );
        poolInfo[_pid].allocPoint = _allocPoint;
    }
```

Solution

It is recommended to transfer ownership to community governance or timelock contract.

Status

Confirming

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002106150005	SlowMist Security Team	2021.06.15 - 2021.06.15	Low Risk

Summary conclusion: The SlowMist security team use a manual and SlowMist team's analysis tool to audit the project, during the audit work we found 1 low risk vulnerabilities.

6 Statement

SlowMist issues this report with reference to the facts that have occurred or existed before the issuance of this report, and only assumes corresponding responsibility based on these.

For the facts that occurred or existed after the issuance, SlowMist is not able to judge the security status of this project, and is not responsible for them. The security audit analysis and other contents of this report are based on the documents and materials provided to SlowMist by the information provider till the date of the insurance report (referred to as "provided information"). SlowMist assumes: The information provided is not missing, tampered with, deleted or concealed. If the information provided is missing, tampered with, deleted, concealed, or inconsistent with the actual situation, the SlowMist shall not be liable for any loss or adverse effect resulting therefrom. SlowMist only conducts the agreed security audit on the security situation of the project and issues this report. SlowMist is not responsible for the background and other conditions of the project.

Official Website

www.slowmist.com

E-mail

team@slowmist.com

Twitter

@SlowMist_Team

Github

https://github.com/slowmist