Smart Contract Security Audit Report

SLOWMIST

Contents

1. Executive Summary

On Mar. 22, 2021, the SlowMist security team received the Cyclone team's security audit application for Cyclone Protocol, 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.

SlowMist Smart Contract DeFi project test method:

Black box testing	Conduct security tests from an attacker's perspective externally.
Grey box testing	Conduct security testing on code module 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.

SlowMist Smart Contract DeFi project risk level:

Critical vulnerabilities	Critical vulnerabilities will have a significant impact on the security of the DeFi project, and it is strongly recommended to fix the critical vulnerabilities.
High-risk vulnerabilities	High-risk vulnerabilities will affect the normal operation of DeFi project. It is strongly recommended to fix high-risk vulnerabilities.
Medium-risk vulnerabilities	Medium vulnerability will affect the operation of DeFi project. It is recommended to fix medium-risk vulnerabilities.

Low-risk vulnerabilities	Low-risk vulnerabilities may affect the operation of DeFi project in certain scenarios. It is suggested that the project party should evaluate and consider whether these vulnerabilities need to be fixed.
Weaknesses	There are safety risks theoretically, but it is extremely difficult to reproduce in engineering.
Enhancement Suggestions	There are better practices for coding or architecture.

2. Audit Methodology

Our security audit process for smart contract includes two steps:

- Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using public and in-house 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 attack and other Race Conditions
- Replay attack
- Reordering attack
- Short address attack
- Denial of service attack
- Transaction Ordering Dependence attack
- Conditional Completion attack
- Authority Control attack
- Integer Overflow and Underflow attack

- TimeStamp Dependence attack
- Gas Usage, Gas Limit and Loops
- Redundant fallback function
- Unsafe type Inference
- Explicit visibility of functions state variables
- Logic Flaws
- Uninitialized Storage Pointers
- Floating Points and Numerical Precision
- tx.origin Authentication
- "False top-up" Vulnerability
- Scoping and Declarations

3. Project Background

3.1 Project Introduction

Cyclone is a cross-chain, non-custodial, universal privacy-preserving protocol with the decentralized governance for all DeFi apps. Cyclone applies zkSNARKs to enable transactional privacy for all DeFi components by breaking the on-chain link between depositor and recipient addresses. It uses a smart contract that accepts coins/tokens deposits, which can be withdrawn by a different address.

Audit code:

https://github.com/cycloneprotocol/cyclone-contracts

Audit version file information

Initial audit files:

Commit: 349b135b60acb7148aebf23d6df8320c3b3a884f

Final audit files:

Commit: b4e7f8a884763054309decb79a40dd73ebf569b8

3.2 Project Structure

```
.
├── LICENSE
├── README.md
├── abi
│   ├── Aeolus.json
│   ├── CoinCyclone.json
│   ├── CycloneToken.json
│   ├── ERC20Cyclone.json
│   ├── GovernorAlpha.json
│   ├── IMimoFactory.json
│   ├── Timelock.json
│   └── Verifier.json
├── circuits
│   ├── merkleTree.circom
│   └── withdraw.circom
├── compile-hasher.js
├── contracts
│   ├── Aeolus.sol
│   ├── AeolusV2.sol
│   ├── Cyclone.sol
│   ├── CycloneV2.sol
│   ├── CycloneV2dot1.sol
│   ├── ICycloneV2.sol
│   ├── Migrations.sol
│   ├── governance
│   │   ├── GovernorAlpha.sol
│   │   ├── ITimelock.sol
│   │   └── Timelock.sol
│   ├── lifecycle
│   │   └── Pausable.sol
│   ├── math
│   │   └── SafeMath.sol
│   ├── mimo
│   │   ├── CoinCyclone.sol
│   │   ├── ERC20Cyclone.sol
│   │   ├── IMimoExchange.sol
│   │   └── IMimoFactory.sol
│   ├── mock
│   │   ├── ERC20.sol
│   │   ├── MimoExchange.sol
```

```
|   |   ├── MimoFactory.sol
|   |   ├── MockUniswapV2Router.sol
|   |   └── TestCycloneDelegate.sol
|   ├── ownership
|   |   ├── Ownable.sol
|   |   └── Whitelist.sol
|   ├── token
|   |   ├── BasicToken.sol
|   |   ├── CycloneToken.sol
|   |   ├── IERC20.sol
|   |   ├── IERC20Basic.sol
|   |   ├── IMintableToken.sol
|   |   ├── SafeERC20.sol
|   |   ├── ShadowToken.sol
|   |   └── StandardToken.sol
|   ├── uniswapv2
|   |   ├── IRouter.sol
|   |   └── UniswapV2CycloneRouter.sol
|   ├── utils
|   |   └── Address.sol
|   └── zksnarklib
|       ├── IVerifier.sol
|       ├── MerkleTreeWithHistory.sol
|       └── Verifier.sol
├── lib
|   ├── MerkleTree.js
|   ├── MiMC.js
|   └── Storage.js
├── linker.js
├── package-lock.json
├── package.json
├── production
|   ├── Verifier.sol
|   ├── withdraw.json
|   ├── withdraw_proving_key.bin
|   └── withdraw_verification_key.json
└── test
    ├── Aeolus.test.js
    ├── CoinCyclone.test.js
    ├── CycloneToken.test.js
    ├── ERC20Cyclone.test.js
    └── GovernorAlpha.test.js
```

3.3 Contract Structure

The Cyclone Protocol project is mainly divided into 4 parts, namely governance contract, token contract, liquidity mining contract, and zkSNARKs privacy pool mining contract. The governance contract (governance) uses the same code as the well-known project Compound to provide GovDAO authority governance for the contract; the token contract is CYC ERC20 token; the liquid mining contract is used to mortgage CYC-BNB LP for mining; the privacy mining contract is passed Deposit a fixed amount of tokens for mining, and provide privacy transfer services through zero-knowledge proof technology.

4. Code Overview

4.1 Main Contract address

The contract has been deployed on the mainnet:

CYC token: https://bscscan.com/token/0x810ee35443639348adbbc467b33310d2ab43c168

AeolusV2: https://bscscan.com/address/0x567da514637cfd7f9e1f185ae4aa163b3ebb5363

bnb pool: https://bscscan.com/address/0x66b5e322dc31f8c7a33ffd23975163795f8d16c7

cyc pool: https://bscscan.com/address/0xd90a6bf8439ef7214cf00da83e926068b6a507ec

busd pool: https://bscscan.com/address/0xbe19d541389c9d3e03efc08f3d5008e8c9cc42a5

iotx pool: https://bscscan.com/address/0x79459751f6882868d1299bfa412428488b434541

CycloneV2dot1:https://bscscan.com/address/0xD90a6BF8439EF7214cF00Da83E926068b6a507eC

4.2 Contracts Description

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

AeolusV2			
Function Name	Visibility	Mutability	Modifiers
setEntranceFeeRate	public	Can modify state	onlyOwner
setRewardPerBlock	public	Can modify state	onlyOwner
updateBlockReward	public	Can modify state	-
deposit	public	Can modify state	-
withdraw	public	Can modify state	-
emergencyWithdraw	public	Can modify state	-
transferOwnership	public	Can modify state	-

CycloneV2

Function Name	Visibility	Mutability	Modifiers
updateBlockReward	public	Can modify state	-
deposit	external payable	Can modify state	nonReentrant
withdraw	external payable	Can modify state	nonReentrant
updateVerifier	external	Can modify state	onlyGovDAO
changeGovDAO	external	Can modify state	onlyGovDAO
setRewardPerBlock	public	Can modify state	onlyGovDAO
setAnonymityRate	public	Can modify state	onlyGovDAO

CycloneV2dot1			
Function Name	Visibility	Mutability	Modifiers
updateBlockReward	public	Can modify state	-
deposit	external payable	Can modify state	nonReentrant
withdraw	external	Can modify state	nonReentrant

	payable		
updateVerifier	external	Can modify state	onlyGovDAO
changeGovDAO	external	Can modify state	onlyGovDAO
setRewardPerBlock	public	Can modify state	onlyGovDAO
setAnonymityRate	public	Can modify state	onlyGovDAO

CycloneToken			
Function Name	Visibility	Mutability	Modifiers
addMinter	external	Can modify state	onlyOperator
removeMinter	external	Can modify state	onlyOperator
updateOperator	external	Can modify state	onlyOperator
mint	public	Can modify state	onlyMinters whenNotPaused
burn	public	Can modify state	-
transfer	public	Can modify state	-

transferFrom	public	Can modify state	-
delegate	public	Can modify state	-
delegateBySig	public	Can modify state	-
pause	public	Can modify state	whenNotPaused
unpause	public	Can modify state	onlyOwner whenPaused
transferOwnerShip	public	Can modify state	onlyOwner
increaseApproval	public	Can modify state	-
decreaseApproval	public	Can modify state	-

4.3 Code Audit

4.3.1 Medium-risk vulnerabilities

4.3.1.1 `safeCYCTransfer` did not check the return value

`transfer` has a bool type return value, which may cause logic errors if it is not checked.

Code location: AeolusV2.sol

```
// Safe CYC transfer function, just in case if rounding error causes pool to not have enough CYCs.
function safeCYCTransfer(address _to, uint256 _amount) internal {
uint256 cycBalance = cycToken.balanceOf(address(this));
if (_amount > cycBalance) {
```

```
_amount = cycBalance;
}
rewardToDistribute -= _amount;
cycToken.transfer(_to, _amount);
}
```

Fix status: In the repair version, `require` is used to determine the return value.

4.3.1.2 Reentrancy attack

The deposit/withdraw/emergencyWithdraw function does not follow `Checks-Effects-Interactions`

It can be re-entered by the contract, and there is a risk of being attacked.

Fix status: Not fixed.

4.3.2 Low-risk vulnerabilities

4.3.2.1 The operator authority of the mainnet CycloneToken contract is too large

The operator authority has the authority to issue additional tokens. At present, the operator on the mainnet is not set as a governance contract and has greater authority.

Code location: CycloneToken.sol

```
function addMinter(address _minter) external onlyOperator {
minters[_minter] = true;
emit MinterAdded(_minter);
}

function removeMinter(address _minter) external onlyOperator {
minters[_minter] = false;
emit MinterRemoved(_minter);
```

```
}

function updateOperator(address _operator) external onlyOperator {
require (_operator != address(0), "invalid operator address");
operator = _operator;
}

function mint(address _to, uint256 _amount) public onlyMinters whenNotPaused returns (bool) {
require (_to != address(0), "invalid address for mint");
require (_amount != 0, "mint amount should not be zero");
totalSupply_ = totalSupply_.add(_amount);
balances[_to] = balances[_to].add(_amount);
emit Minted(_to, _amount);
emit Transfer(address(0), _to, _amount);
_moveDelegates(address(0), delegates[_to], _amount);
return true;
}
```

Fix status: Not fixed.

5. Audit Result

5.1 Conclusion

Audit Result : Some Risks

Audit Number : 0X002104060004

Audit Date : Apr. 06, 2021

Audit Team : SlowMist Security Team

Summary conclusion: The SlowMist security team use a manual and SlowMist Team analysis tool audit of the codes for security issues. There are 3 security issues found during the audit. There are 2

medium-risk vulnerabilities, 1 low-risk vulnerabilities. After communicating with the project party and confirming that the risks found in the audit process have been fixed or are within the acceptable range.

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 base 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 this 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