MythX Report

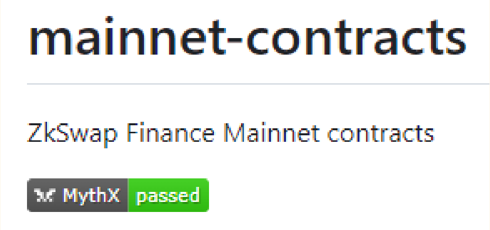
Overview

Project Name	zkSwap Finance
Auditor	MythX.io
Source Code	https://github.com/ZkSwapFinance/zf-periphery/tree/main/contracts/token
Mode	Deep
Time	Fri Dec 22nd 2023
DETECTED VULNERABILITIES	**6**

Summary

Done	Contract	High Risk Issues	Medium Risk Issues	Low Risk Issues
☑	yZFToken.sol	0	0	6

Reference

yZFToken.sol	https://github.com/ZkSwapFinance/Audit-Reports/blob/main/Original_MythX_yZFToken.pdf
MythX Passed Badge on Github  mainnet-contracts ZkSwap Finance Mainnet contracts MythX passed	https://github.com/ZkSwapFinance/zf-periphery

MythX

REPORT 658501DEF1BCF1001A61D178

Created	Fri Dec 22 2023 03:26:22 GMT+0000 (Coordinated Universal Time)
Number of analyses	1
User	648fc02af4bf584372592643

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
528c1eda-0006-4fb5-b2c9-4a6994a7b5b5	/governance/yzftoken.sol	6

Analysis 528c1eda-0006-4fb5-b2c9-4a6994a7b5b5

MythX

Started	Fri Dec 22 2023 03:26:30 GMT+0000 (Coordinated Universal Time)
Finished	Fri Dec 22 2023 04:12:04 GMT+0000 (Coordinated Universal Time)
Mode	Deep
Client Tool	Mythx-Vscode-Extension
Main Source File	/Governance/Yzftoken.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	6

ISSUES

LOW

SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

/governance/yzftoken.sol

Locations

```
// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

import "@openzeppelin/contracts/utils/math/SafeMath.sol";
```

LOW

SWC-116

A control flow decision is made based on The block.timestamp environment variable.

The block.timestamp environment variable is used to determine a control flow decision. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source file

/governance/yzftoken.sol

Locations

```
require(signatory != address(0), "Uni::delegateBySig: invalid signature");
require(nonce == nonces[signatory]++, "Uni::delegateBySig: invalid nonce");
require(block.timestamp <= expiry, "Uni::delegateBySig: signature expired");
return _delegate(signatory, delegatee);
}

/**
/**
* @notice Gets the current votes balance for `account`
* @param account The address to get votes balance
```

LOW

SWC-116

A control flow decision is made based on The block.timestamp environment variable.

The block.timestamp environment variable is used to determine a control flow decision. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source file

/governance/yzftoken.sol

Locations

```
require(signatory != address(0), "Uni::permit: invalid signature");
require(signatory == owner, "Uni::permit: unauthorized");
require(block.timestamp <= deadline, "Uni::permit: signature expired");

allowances[owner][spender] = amount;

emit Approval(owner, spender, amount);
}

```

LOW

SWC-120

Potential use of "block.number" as source of randonmness.

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source file

/governance/yzftoken.sol

Locations

```
require(blockNumber < block.number, "Uni::getPriorVotes: not yet determined");

uint32 nCheckpoints = numCheckpoints[account];
if (nCheckpoints == 0) {
return 0;
```

LOW

SWC-120

Potential use of "block.number" as source of randonmness.

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source file

/governance/yzftoken.sol

Locations

```

function _writeCheckpoint(address delegatee, uint32 nCheckpoints, uint96 oldVotes, uint96 newVotes) internal {
uint32 blockNumber = safe32(block.number, "Uni::_writeCheckpoint: block number exceeds 32 bits");

if (nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber) {
if (nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber) {
checkpoints[delegatee][nCheckpoints - 1].votes = newVotes;
} else {
```

LOW

SWC-120

A control flow decision is made based on The block.number environment variable.

The block.number environment variable is used to determine a control flow decision. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source file

`/governance/yzftoken.sol`

Locations

```
*/
function getPriorVotes(address account, uint blockNumber) public view returns (uint96) {
require(blockNumber < block.number, "Uni::getPriorVotes: not yet determined");

uint32 nCheckpoints = numCheckpoints[account];
if (nCheckpoints == 0) {
if (nCheckpoints == 0) {
return 0;
}
```