MythX Report - Farm

Overview

Project Name	zkSwap Finance - Farm
Auditor	MythX.io
Source Code	https://github.com/ZkSwapFinance/zf-farm
Mode	Deep
Time	Sat Aug 19th 2023
DETECTED VULNERABILITIES	**6**

Summary

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

Reference

MythX Passed Badge on Github mainnet-contracts ZkSwap Finance Mainnet contracts MythX passed	https://github.com/ZkSwapFinance/zf-farm

MythX

REPORT 64E1F074F3B80E0019369FEB

Created	Sun Aug 20 2023 10:52:36 GMT+0000 (Coordinated Universal Time)
Number of analyses	1
User	648fc02af4bf584372592643

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
29926adb-9847-46f9-944a-8a521e0cf61b	/farm/zffarm.sol	6

Analysis 29926adb-9847-46f9-944a-8a521e0cf61b MythX

Started	Sun Aug 20 2023 10:52:46 GMT+0000 (Coordinated Universal Time)
Finished	Sun Aug 20 2023 11:38:21 GMT+0000 (Coordinated Universal Time)
Mode	Deep
Client Tool	Mythx-Vscode-Extension
Main Source File	/Farm/Zffarm.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	6

ISSUES

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

/farm/zffarm.sol

Locations

```
214 startTime = _startTime;
215
216 uint256 length = poolInfo.length;
217 for (uint256 pid = 0; pid < length; ++pid) {
218 PoolInfo storage pool = poolInfo[pid];
219 pool.lastRewardTime = startTime;
```

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

/farm/zffarm.sol

Locations

```
85 uint256 lastRewardTime = block.timestamp > startTime ? block.timestamp : startTime;
86 totalAllocPoint = totalAllocPoint.add(_allocPoint);
87 poolInfo.push(PoolInfo({
88 lpToken: _lpToken,
89 allocPoint: _allocPoint,
90 lastRewardTime: lastRewardTime,
```

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

/farm/zffarm.sol

Locations

```
117 return;
118 }
119 uint256 lpSupply = pool.lpToken.balanceOf(address(this));
120 if (lpSupply == 0 || pool.allocPoint == 0) {
121 pool.lastRewardTime = block.timestamp;
122 return;
123 }
```

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

/farm/zffarm.sol

Locations

```
70 uint256 lpSupply = pool.lpToken.balanceOf(address(this));
71 if (block.timestamp > pool.lastRewardTime && lpSupply != 0) {
72 uint256 multiplier = getMultiplier(pool.lastRewardTime, block.timestamp);
73 uint256 zfReward = multiplier.mul(zfPerSecond).mul(pool.allocPoint).div(totalAllocPoint);
74 accZFPerShare = accZFPerShare.add(zfReward.mul(1e12).div(lpSupply));
75 }
```

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

/farm/zffarm.sol

Locations

```
70 uint256 lpSupply = pool.lpToken.balanceOf(address(this));
71 if (block.timestamp > pool.lastRewardTime && lpSupply != 0) {
72 uint256 multiplier = getMultiplier(pool.lastRewardTime, block.timestamp);
73 uint256 zfReward = multiplier.mul(zfPerSecond).mul(pool.allocPoint).div(totalAllocPoint);
74 accZFPerShare = accZFPerShare.add(zfReward.mul(1e12).div(lpSupply));
75 }
76 uint256 pending = user.amount.mul(accZFPerShare).div(1e12).sub(user.rewardDebt);
77 return pending;
78 }
79
80 function add(uint256 _allocPoint, IERC20 _lpToken, uint16 _depositFeeBP, bool _withUpdate) public onlyOwner {
81 require(_depositFeeBP <= MAXIMUM_DEPOSIT_FEE, "add: invalid deposit fee");
82 if (_withUpdate) {
```