b0d81f232b617924c346806297e1d0ea1fa49cdcb8a42be8a282cc4a32c6429f

File: MasterChef.sol | Language:solidity | Size:13759 bytes | Date:2022-06-15T16:11:39.701Z

Critical	High	Medium	Low	Note
1	0	1	0	7

Issues

Severity	Issue	Analyzer	Code Lines
Critical	SWC-107	Achilles	107 - 138
Medium	SWC-102	Achilles	3
Note	SWC-116	Achilles	124, 124, 190, 191, 220, 224, 230

Code

1. SWC-107 / lines: 107 - 138 Critical Achilles

A security vulnerability has been detected.

```
106     /// @param _withUpdate Whether call "massUpdatePools" operation.
107     function add(
108         uint256 _allocPoint,
109         IERC20 _lpToken,
110         IRewarder _rewarder,
111         bool _withUpdate
112     ) external onlyOwner {
113         require(!isPool[_lpToken], "add: LP already added");
114         // Sanity check to ensure _lpToken is an ERC20 token
115         _lpToken.balanceOf(address(this));
116         // Sanity check if we add a rewarder
117         if (address(_rewarder) != address(0)) {
118             _rewarder.onAuraReward(address(0), 0);
119         }
120         if (_withUpdate) {
121             massUpdatePools();
122         }
123
124         uint256 lastRewardTimestamp = block.timestamp > startTimestamp ? block.timestamp : startTimestamp;
125         totalAllocPoint = totalAllocPoint.add(_allocPoint);
126
127         poolInfo.push(
128             PoolInfo({
129                 lpToken: _lpToken,
130                 allocPoint: _allocPoint,
131                 lastRewardTimestamp: lastRewardTimestamp,
132                 accAuraPerShare: 0,
133                 rewarder: _rewarder
134             })
135         );
136         isPool[_lpToken] = true;
137         emit Add(poolInfo.length.sub(1), _allocPoint, _lpToken, _rewarder);
138     }
139
```

In detail

One of the major dangers of calling external contracts is that they can take over the control flow. In the reentrancy attack (a.k.a. recursive call attack), a malicious contract calls back into the calling contract before the first invocation of the function is finished. This may cause the different invocations of the function to interact in undesirable ways.

2. SWC-102 / lines: 3 Medium Achilles

A security vulnerability has been detected.

```
2
3    pragma solidity 0.6.12;
4    pragma experimental ABIEncoderV2;
```

In detail

Using an outdated compiler version can be problematic especially if there are publicly disclosed bugs and issues that affect the current compiler version.

3. SWC-116 / lines: 124 Note Achilles

A security vulnerability has been detected.

```
123
124          uint256 lastRewardTimestamp = block.timestamp > startTimestamp ? block.timestamp : startTimestamp;
125          totalAllocPoint = totalAllocPoint.add(_allocPoint);
```

In detail

Contracts often need access to the current timestamp to trigger time-dependent events. As Ethereum is decentralized, nodes can synchronize time only to some degree. Moreover, malicious miners can alter the timestamp of their blocks, especially if they can gain advantages by doing so. However, miners can't set timestamp smaller than the previous one (otherwise the block will be rejected), nor can they set the timestamp too far ahead in the future. Taking all of the above into consideration, developers can't rely on the preciseness of the provided timestamp.

4. SWC-116 / lines: 124 Note Achilles

A security vulnerability has been detected.

```
123
124          uint256 lastRewardTimestamp = block.timestamp > startTimestamp ? block.timestamp : startTimestamp;
125          totalAllocPoint = totalAllocPoint.add(_allocPoint);
```

In detail

Contracts often need access to the current timestamp to trigger time-dependent events. As Ethereum is decentralized, nodes can synchronize time only to some degree. Moreover, malicious miners can alter the timestamp of their blocks, especially if they can gain advantages by doing so. However, miners can't set timestamp smaller than the previous one (otherwise the block will be rejected), nor can they set the timestamp too far ahead in the future. Taking all of the above into consideration, developers can't rely on the preciseness of the provided timestamp.

5. SWC-116 / lines: 190 Note Achilles

A security vulnerability has been detected.

```
189          uint256 lpSupply = pool.lpToken.balanceOf(address(this));
190          if (block.timestamp > pool.lastRewardTimestamp && lpSupply != 0) {
191              uint256 timeElapsed = block.timestamp.sub(pool.lastRewardTimestamp);
```

In detail

Contracts often need access to the current timestamp to trigger time-dependent events. As Ethereum is decentralized, nodes can synchronize time only to some degree. Moreover, malicious miners can alter the timestamp of their blocks, especially if they can gain advantages by doing so. However, miners can't set timestamp smaller than the previous one (otherwise the block will be rejected), nor can they set the timestamp too far ahead in the future. Taking all of the above into consideration, developers can't rely on the preciseness of the provided timestamp.

6. SWC-116 / lines: 191 Note Achilles

A security vulnerability has been detected.

```
190          if (block.timestamp > pool.lastRewardTimestamp && lpSupply != 0) {
191              uint256 timeElapsed = block.timestamp.sub(pool.lastRewardTimestamp);
192              uint256 auraReward = timeElapsed.mul(auraPerSec).mul(pool.allocPoint).div(totalAllocPoint);
```

```
192             uint256 auraReward = timeElapsed.mul(auraPerSec).mul(pool.allocPoint).div(totalAllocPoint);
```

In detail

Contracts often need access to the current timestamp to trigger time-dependent events. As Ethereum is decentralized, nodes can synchronize time only to some degree. Moreover, malicious miners can alter the timestamp of their blocks, especially if they can gain advantages by doing so. However, miners can't set timestamp smaller than the previous one (otherwise the block will be rejected), nor can they set the timestamp too far ahead in the future. Taking all of the above into consideration, developers can't rely on the preciseness of the provided timestamp.

7. SWC-116 / lines: 220 Note Achilles

⊖ `A security vulnerability has been detected.`

```
219         PoolInfo memory pool = poolInfo[_pid];
220         if (block.timestamp > pool.lastRewardTimestamp) {
221             uint256 lpSupply = pool.lpToken.balanceOf(address(this));
```

In detail

Contracts often need access to the current timestamp to trigger time-dependent events. As Ethereum is decentralized, nodes can synchronize time only to some degree. Moreover, malicious miners can alter the timestamp of their blocks, especially if they can gain advantages by doing so. However, miners can't set timestamp smaller than the previous one (otherwise the block will be rejected), nor can they set the timestamp too far ahead in the future. Taking all of the above into consideration, developers can't rely on the preciseness of the provided timestamp.

8. SWC-116 / lines: 224 Note Achilles

⊖ `A security vulnerability has been detected.`

```
223             if (lpSupply > 0 && totalAllocPoint > 0) {
224                 uint256 timeElapsed = block.timestamp.sub(pool.lastRewardTimestamp);
225                 uint256 auraReward = timeElapsed.mul(auraPerSec).mul(pool.allocPoint).div(totalAllocPoint);
```

In detail

Contracts often need access to the current timestamp to trigger time-dependent events. As Ethereum is decentralized, nodes can synchronize time only to some degree. Moreover, malicious miners can alter the timestamp of their blocks, especially if they can gain advantages by doing so. However, miners can't set timestamp smaller than the previous one (otherwise the block will be rejected), nor can they set the timestamp too far ahead in the future. Taking all of the above into consideration, developers can't rely on the preciseness of the provided timestamp.

9. SWC-116 / lines: 230 Note Achilles

⊖ `A security vulnerability has been detected.`

```
229             }
230             pool.lastRewardTimestamp = block.timestamp;
231             poolInfo[_pid] = pool;
```

In detail

Contracts often need access to the current timestamp to trigger time-dependent events. As Ethereum is decentralized, nodes can synchronize time only to some degree. Moreover, malicious miners can alter the timestamp of their blocks, especially if they can gain advantages by doing so. However, miners can't set timestamp smaller than the previous one (otherwise the block will be rejected), nor can they set the timestamp too far ahead in the future. Taking all of the above into consideration, developers can't rely on the preciseness of the provided timestamp.