SOOHO

c491fcb0d08a86ee5143cd004e578c6f4b91e9fa52be7313f257387d8daa63ee

File: SimpleRewarde | Language:solidity | Size:10291 bytes | Date:2022-06-15T16:11:56.441Z

Critical	High	Medium	Low	Note
0	0	1	0	8

Issues

Severity	Issue	Analyzer	Code Lines
Medium	SWC-102	Achilles	2
Note	SWC-116	Achilles	140, 197, 198, 233, 237, 242
Note	SWC-131	Achilles	160, 162

Code

1. SWC-102 / lines: 2 Medium Achilles

```
A security vulnerability has been detected.

1  // SPDX-License-Identifier: GPL-3.0
2  pragma solidity ^0.6.12;
3  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Using an outdated compiler version can be problematic especially if there are publicly disclosed bugs and issues that affect the current compiler version.

2. SWC-116 / lines: 140 Note Achilles

```
A security vulnerability has been detected.

139          isNative = _isNative;
140          poolInfo = PoolInfo({lastRewardTimestamp: block.timestamp, accTokenPerShare: 0});
141      }
```

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.

3. SWC-116 / lines: 197 Note Achilles

```
A security vulnerability has been detected.

196
197          if (block.timestamp > pool.lastRewardTimestamp && lpSupply != 0) {
198              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.

4. SWC-116 / lines: 198 Note Achilles

A security vulnerability has been detected.

```
197         if (block.timestamp > pool.lastRewardTimestamp && lpSupply != 0) {
198             uint256 timeElapsed = block.timestamp.sub(pool.lastRewardTimestamp);
199             uint256 tokenReward = timeElapsed.mul(tokenPerSec);
```

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: 233 Note Achilles

A security vulnerability has been detected.

```
232
233         if (block.timestamp > pool.lastRewardTimestamp) {
234             uint256 lpSupply = lpToken.balanceOf(address(MCA));
```

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: 237 Note Achilles

A security vulnerability has been detected.

```
236             if (lpSupply > 0) {
237                 uint256 timeElapsed = block.timestamp.sub(pool.lastRewardTimestamp);
238                 uint256 tokenReward = timeElapsed.mul(tokenPerSec);
```

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: 242 Note Achilles

A security vulnerability has been detected.

```
241
242             pool.lastRewardTimestamp = block.timestamp;
243             poolInfo = 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.

8. SWC-131 / lines: 160 Note Achilles

A security vulnerability has been detected.

```
159             if (isNative) {
160                 uint256 balance = address(this).balance;
161                 if (pending > balance) {
```

In detail

Unused variables are allowed in Solidity and they do not pose a direct security issue. It is best practice though to avoid them as they can:
- cause an increase in computations (and unnecessary gas consumption)
- indicate bugs or malformed data structures and they are generally a sign of poor code quality
- cause code noise and decrease readability of the code

9. SWC-131 / lines: 162 Note Achilles

A security vulnerability has been detected.

```
161                 if (pending > balance) {
162                     (bool success, ) = _user.call.value(balance)("");
163                     require(success, "Transfer failed");
```

In detail

Unused variables are allowed in Solidity and they do not pose a direct security issue. It is best practice though to avoid them as they can:
- cause an increase in computations (and unnecessary gas consumption)
- indicate bugs or malformed data structures and they are generally a sign of poor code quality
- cause code noise and decrease readability of the code