SOOHO

13770620c15d13d56b15e765854ddcec47e519f722eb7e72bbe81fe090f8801a

File: BerryValut.sol | Language:solidity | Size:32877 bytes | Date:2021-08-01T16:08:13.759Z

Critical	High	Medium	Low	Note
0	0	1	8	16

Issues

Severity	Issue	Analyzer	Code Lines
Medium	SWC-102	Achilles	13
Low	SWC-103	Achilles	13
Low	SOOHO-SOL-001	Achilles	465, 470, 471, 554, 555, 556, 557
Note	SWC-116	Achilles	649, 652, 670, 750, 756, 759, 766, 767, 773, 776, 783, 784, 791

Code

1. SWC-102 / lines: 13 Medium Achilles

```
A security vulnerability has been detected.

12
13  pragma solidity ^0.5.16;
14
```

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Using an outdated compiler version can be problematic especially if there are publicly disclosed bugs and issues that affect the current compiler version.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-103 / lines: 13 Low Achilles

```
A security vulnerability has been detected.

12
13  pragma solidity ^0.5.16;
14
```

In detail

Contracts should be deployed with the same compiler version and flags that they have been tested with thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.

3. SOOHO-SOL-001 / lines: 465 Low Achilles

```
A security vulnerability has been detected.

464     function getPair(address tokenA, address tokenB) external view returns (address pair);
465     function allPairs(uint) external view returns (address pair);
466     function allPairsLength() external view returns (uint);
```

In detail

Empty function parameter name

The name of function parameter is empty.

4. SOOHO-SOL-001 / lines: 470 Low Achilles

A security vulnerability has been detected.

```
469
470     function setFeeTo(address) external;
471     function setFeeToSetter(address) external;
```

In detail

Empty function parameter name
The name of function parameter is empty.

5. SOOHO-SOL-001 / lines: 471 Low Achilles

A security vulnerability has been detected.

```
470     function setFeeTo(address) external;
471     function setFeeToSetter(address) external;
472 }
```

In detail

Empty function parameter name
The name of function parameter is empty.

6. SOOHO-SOL-001 / lines: 554 Low Achilles

A security vulnerability has been detected.

```
553     function balanceOf(address owner) external view returns (uint);
554     function deposit(uint) external;
555     function withdraw(uint) external;
```

In detail

Empty function parameter name
The name of function parameter is empty.

7. SOOHO-SOL-001 / lines: 555 Low Achilles

A security vulnerability has been detected.

```
554     function deposit(uint) external;
555     function withdraw(uint) external;
556     function depositCherry(uint) external;
```

In detail

Empty function parameter name
The name of function parameter is empty.

8. SOOHO-SOL-001 / lines: 556 Low Achilles

A security vulnerability has been detected.

```
555     function withdraw(uint) external;
556     function depositCherry(uint) external;
557     function withdrawCherry(uint) external;
```

In detail

Empty function parameter name
The name of function parameter is empty.

9. SOOHO-SOL-001 / lines: 557 Low Achilles

A security vulnerability has been detected.

```
556     function depositCherry(uint) external;
557     function withdrawCherry(uint) external;
558     function withdrawCherryAll() external;
```

In detail

Empty function parameter name
The name of function parameter is empty.

10. SWC-116 / lines: 649 Note Achilles

A security vulnerability has been detected.

```
648     function lastTimeRewardApplicable() public view returns (uint256) {
649         return Math.min(block.timestamp, periodFinish);
650     }
```

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.

11. SWC-116 / lines: 652 Note Achilles

A security vulnerability has been detected.

```
651     function lastTimeRewardApplicable2() public view returns (uint256) {
652         return Math.min(block.timestamp, periodFinish2);
653     }
```

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.

12. SWC-116 / lines: 670 Note Achilles

A security vulnerability has been detected.

```
669     function newDuration2(uint _duration) external onlyRewardsDistribution() {
670         require(block.timestamp >= periodFinish2, "period not finish");
671         rewardsDuration2 = _duration;
```

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.

13. SWC-116 / lines: 750 Note Achilles

A security vulnerability has been detected.

```
749     function newDuration(uint _duration) external onlyRewardsDistribution() {
750         require(block.timestamp >= periodFinish, "period not finish");
751         rewardsDuration = _duration;
```

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.

14. SWC-116 / lines: 756 Note Achilles

A security vulnerability has been detected.

```
755     function notifyRewardAmount(uint256 reward) external onlyRewardsDistribution updateReward(address(0)) {
756         if (block.timestamp >= periodFinish) {
757             rewardRate = reward.div(rewardsDuration);
```

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.

15. SWC-116 / lines: 759 Note Achilles

A security vulnerability has been detected.

```
758         } else {
759             uint256 remaining = periodFinish.sub(block.timestamp);
760             uint256 leftover = remaining.mul(rewardRate);
```

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.

16. SWC-116 / lines: 766 Note Achilles

A security vulnerability has been detected.

```
765
766         lastUpdateTime = block.timestamp;
767         periodFinish = block.timestamp.add(rewardsDuration);
```

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.

17. SWC-116 / lines: 767 Note Achilles

A security vulnerability has been detected.

```
766         lastUpdateTime = block.timestamp;
767         periodFinish = block.timestamp.add(rewardsDuration);
768         emit RewardAdded(reward);
```

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.

18. SWC-116 / lines: 773 Note Achilles

A security vulnerability has been detected.

```
772     function notifyRewardAmount2(uint256 reward) external onlyRewardsDistribution updateReward2(address(0)) {
773         if (block.timestamp >= periodFinish2) {
774             rewardRate2 = reward.div(rewardsDuration2);
```

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.

19. SWC-116 / lines: 776 Note Achilles

A security vulnerability has been detected.

```
775         } else {
776             uint256 remaining = periodFinish2.sub(block.timestamp);
777             uint256 leftover = remaining.mul(rewardRate2);
```

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.

20. SWC-116 / lines: 783 Note Achilles

A security vulnerability has been detected.

```
782
783         lastUpdateTime2 = block.timestamp;
784         periodFinish2 = block.timestamp.add(rewardsDuration2);
```

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.

21. SWC-116 / lines: 784 Note Achilles

A security vulnerability has been detected.

```
783         lastUpdateTime2 = block.timestamp;
784         periodFinish2 = block.timestamp.add(rewardsDuration2);
785         emit RewardAdded(reward);
```

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.

22. SWC-116 / lines: 791 Note Achilles

A security vulnerability has been detected.

```
790         rewardsToken2.safeTransferFrom(msg.sender, address(this), reward);
```

```
791         if (block.timestamp >= periodFinish2) {
792             rewardRate2 = reward.div(rewardsDuration2);
```

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.

23. SWC-116 / lines: 794 Note Achilles

A security vulnerability has been detected.

```
793         } else {
794             uint256 remaining = periodFinish2.sub(block.timestamp);
795             uint256 leftover = remaining.mul(rewardRate2);
```

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.

24. SWC-116 / lines: 801 Note Achilles

A security vulnerability has been detected.

```
800
801         lastUpdateTime2 = block.timestamp;
802         periodFinish2 = block.timestamp.add(rewardsDuration2);
```

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.

25. SWC-116 / lines: 802 Note Achilles

A security vulnerability has been detected.

```
801         lastUpdateTime2 = block.timestamp;
802         periodFinish2 = block.timestamp.add(rewardsDuration2);
803         emit RewardAdded(reward);
```

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.