Project: cavelarink

0xc4a413c0a32f7731f503528ca4c0009c44388f21

21/03/2024

AUDIT REPORT

SAFETY SCORE: 85

1 - Arbitrary Jump/Storage Write

Result: Pass

2 - Centralization of Control

Result: Medium

Details: The contract has an onlyOwner modifier which centralizes control over certain functions such as renounceOwnership,

initialize, launch, toggleLimits, and setTax. This centralization could pose a risk if the owner's account is compromised or if the

owner acts maliciously.

3 - Compiler Issues

Result: Pass

4 - Delegate Call to Untrusted Contract

Result: Pass

5 - Dependence on Predictable Variables

Result: Pass

6 - Ether/Token Theft

Result: Pass

7 - Flash Loans

Result: Pass

8 - Front Running

Result: Medium

Details: The contract does not appear to have specific protections against front-running attacks, such as using a commit-reveal scheme or

similar mechanisms. This could potentially allow miners or other users to observe pending transactions and place their own transactions first

with higher gas fees.

9 - Improper Events

Result: Pass

10 - Improper Authorization Scheme

Result: Pass

11 - Integer Over/Underflow

Result: Pass

Details: The contract uses SafeMath library for all arithmetic operations, which protects against integer overflow and underflow.

12 - Logical Issues

Result: Medium

Details: The contract has a function initialize that can only be called once, but it does not have a state variable to ensure that the

liquidity is only added once. This could lead to issues if the function is called again, even though there is a check for tradingOpen.

13 - Oracle Issues

Result: Pass

14 - Outdated Compiler Version

Result: Pass

15 - Race Conditions

Result: Pass

16 - Reentrancy

Result: Pass

17 - Signature Issues

Result: Pass

18 - Sybil Attack

Result: Pass

19 - Unbounded Loops

Result: Pass

20 - Unused Code

Result: Low

Details: The contract contains constants that are not used anywhere in the code, such as _preventSwapBefore. This is not a

security risk but is an example of poor code cleanliness and can lead to confusion.

Code:

```
uint256 private constant _preventSwapBefore=25;
```

Correction:

```
// This constant is not used and can be removed.
```

To address the issues found:

For Centralization of Control, consider implementing a multi-signature scheme or a decentralized governance model for critical

functions.

For Front Running, consider adding mechanisms to prevent this type of attack, such as using a commit-reveal scheme.

For Logical Issues in the initialize function, ensure that the liquidity can only be added once by using a state variable.

For Unused Code, remove any constants and variables that are not used to clean up the contract and reduce confusion.