Project: VaultTech

0x7F9b09f4717072CF4DC18b95D1b09E2B30C76790

10/03/2024

AUDIT REPORT

SAFETY SCORE: 84

1 - Arbitrary Jump/Storage Write

Result: Pass

2 - Centralization of Control

Result: High

Details: The contract contains functions that allow the owner to exert a high level of control over the contract, which can be a centralization

risk. The owner can set fees, exclude addresses from fees, enable/disable trading, and more. This centralization could be abused by the

owner.

Code:

```
function setExcludedFromFees(address account, bool enabled) public onlyOwner { ... }

function setNewRouter(address newRouter) external onlyOwner { ... }

function setLpPair(address pair, bool enabled) external onlyOwner { ... }

function setInitializer(address init) public onlyOwner { ... }

function setProtectionSettings(bool _antiSnipe, bool _antiBlock) external onlyOwner { ... }

function lockTaxes() external onlyOwner { ... }

function setTaxes(uint16 buyFee, uint16 sellFee, uint16 transferFee) external onlyOwner { ... }

function setRatios(uint16 marketing, uint16 team, uint16 revShare, uint16 development, uint16 lottery)
```

```
external onlyOwner { ... }

function setWallets(address payable marketing, address payable
development, address payable team,

address payable revShare, address payable lottery) external
onlyOwner { ... }

function setMaxTxPercent(uint256 percent, uint256 divisor)
external onlyOwner { ... }

function setMaxWalletSize(uint256 percent, uint256 divisor)
external onlyOwner { ... }

function setSwapSettings(uint256 thresholdPercent, uint256
thresholdDivisor, uint256 amountPercent,

uint256 amountDivisor) external onlyOwner { ... }

function setContractSwapEnabled(bool swapEnabled, bool
priceImpactSwapEnabled) external onlyOwner { ...

}

function excludePresaleAddresses(address router, address
presale) external onlyOwner { ... }

function transferOwner(address newOwner) external onlyOwner {
... }

function renounceOwnership() external onlyOwner { ... }
```

Correction:

```
// To mitigate centralization risks, consider implementing a
multi-signature scheme or a decentralized

governance model for critical functions.
```

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: Pass

9 - Improper Events

Result: Pass

10 - Improper Authorization Scheme

Result: High

Details: The contract uses a simple ownership model which gives the owner unrestricted access to critical functions that can alter the

contract's behavior. This could lead to unauthorized actions if the owner's account is compromised.

Code:

```
modifier onlyOwner() { require(_owner == msg.sender, "Caller =/= owner."); _; }
```

Correction:

```
// Implement a more robust authorization scheme, such as role-based access control (RBAC) or

multi-signature verification.
```

11 - Integer Over/Underflow

Result: Pass

12 - Logical Issues

Result: Medium

Details: The contract has a function `setNewRouter` that allows the owner to change the router address after liquidity has been added, which

could lead to disruption in trading if misused.

Code:

```
function setNewRouter(address newRouter) external onlyOwner {
... }
```

Correction:

```
// Ensure that changing the router address is a well-governed process, possibly requiring a time delay

or community vote.
```

13 - Oracle Issues

Result: Pass

14 - Outdated Compiler Version

Result: Informational

Details: The contract is compiled with a range of compiler versions from 0.6.0 to 0.9.0. It is recommended to use the latest stable version of

the Solidity compiler to ensure all known bugs and security issues are addressed.

Code:

```
pragma solidity >=0.6.0 <0.9.0;
```

Correction:

```
pragma solidity 0.8.11;
```

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: Informational

Details: There are several functions and modifiers in the contract that are not used or are redundant. Removing unused code can reduce the

contract size and gas costs.

Code:

```
function approveContractContingency() external onlyOwner
returns (bool) { ... }

function setPriceImpactSwapAmount(uint256
priceImpactSwapPercent) external onlyOwner { ... }

function sweepContingency() external onlyOwner { ... }
```

```
function sweepExternalTokens(address token) external onlyOwner
{ ... }

function multiSendTokens(address[] memory accounts, uint256[]
memory amounts) external onlyOwner { ...

}
```

Correction:

```
// Remove unused functions to optimize contract size and gas
usage.
```