Project: PAALAI

0x14feE680690900BA0ccCfC76AD70Fd1b95D10e16

06/03/2024

AUDIT REPORT

SAFETY SCORE: 78

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 risk for

decentralization. The owner can change fees, exclude accounts from fees, enable/disable swaps, and transfer ownership.

Code:

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

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

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

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

Correction:

```
// To mitigate centralization risks, consider implementing a time-locked admin function or a

multi-signature requirement for sensitive functions.
```

3 - Compiler Issues

Result: Pass

4 - Delegate Call to Untrusted Contract

Result: Pass

5 - Dependence on Predictable Variables

Result: Medium

Details: The contract uses block.timestamp for various time checks, which can be slightly manipulated by miners.

Code:

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

function enableTrading() public onlyOwner {...}
```

Correction:

```
// Consider replacing block.timestamp with a more reliable
time oracle if precision is critical.
```

6 - Ether/Token Theft

Result: Pass

7 - Flash Loans

Result: Pass

8 - Front Running

Result: Medium

Details: The contract is vulnerable to front-running attacks because it uses the swapExactTokensForETHSupportingFeeOnTransferTokens

function, which can be observed and front-run on the mempool.

Code:

```
function contractSwap(uint256 contractTokenBalance) internal
inSwapFlag {...}
```

Correction:

```
// Implement a commit-reveal scheme or similar mechanism to
mitigate front-running vulnerabilities.
```

9 - Improper Events

Result: Pass

10 - Improper Authorization Scheme

Result: High

Details: The contract allows the owner to exclude accounts from fees and protection, which can be misused.

Code:

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

function setExcludedFromProtection(address account, bool
enabled) external onlyOwner {...}
```

Correction:

```
// Implement a more decentralized authorization scheme, such
as a multi-signature requirement or DAO

voting for these actions.
```

11 - Integer Over/Underflow

Result: Pass

12 - Logical Issues

Result: Pass

13 - Oracle Issues

Result: Pass

14 - Outdated Compiler Version

Result: Informational

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

and optimization improvements.

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: Low

Details: There are several instances of the `success` variable being set but not used, which is unnecessary and can be removed to save gas.

Code:

```
bool success;

...

(success,) = _taxWallets.marketing.call{value:
marketingBalance, gas: 55000}("");

...

(success,) = _taxWallets.staking.call{value: stakingBalance,
gas: 55000}("");

...

(success,) = _taxWallets.development.call{value:
developmentBalance, gas: 55000}("");

...

(success,) = _taxWallets.externalBuyback.call{value:
externalBuybackBalance, gas: 55000}("");
```

Correction:

```
_taxWallets.marketing.call{value: marketingBalance, gas:
55000}("");

...

_taxWallets.staking.call{value: stakingBalance, gas:
55000}("");

...

_taxWallets.development.call{value: developmentBalance, gas:
55000}("");
```

```
...

_taxWallets.externalBuyback.call{value:
externalBuybackBalance, gas: 55000}("");
```