AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

WE WANT NO WAR

DATED : 28 Aug 24'

MANUAL TESTING

Centralization – Enabling Trades

Severity: High

Function: openTrading

Status: Open

Overview:

The EnableTrading function permits only the contract owner to activate trading capabilities. Until this function is executed, no investors can buy, sell, or transfer their tokens. This places a high degree of control and centralization in the hands of the contract owner.

```
function openTrading() external {
require(hasRole(ADMIN_ROLE, msg.sender), "NOWAR: Not an admin");
require(!tradingOpen, "NOWAR: Trading is already open");

    swapEnabled = true;
    tradingOpen = true;
```

Suggestion:

To reduce centralization and potential manipulation, consider one of the following approaches:

1.Automatically enable trading after a specified condition, such as the completion of a presale, is met.

2.If manual activation is still desired, consider transferring the ownership of the contract to a trustworthy, third-party entity like a certified "PinkSale Safu" developer. This can give investors more confidence in the eventual activation of trading capabilities, mitigating concerns of potential bad-faith actions by the original owner.

MANUAL TESTING

Centralization – Owner can blacklist wallets.

Severity: High

Function: addbot, delBots

Status: Open

Overview:
The owner can blacklist wallets from transferring of tokens for an indefinite period of time which is not recommended. Which can lock user's token.

```
function addBot(address botAddress_) public {
require(hasRole(ADMIN_ROLE, msg.sender), "NOWAR: Not a admin");
require(botAddress_ != address(0), "NOWAR: Invalid address");
    bots[botAddress_] = true;
emit BotAdded(botAddress_);
}

function delBots(address nobotAddress_) public {
require(hasRole(ADMIN_ROLE, msg.sender), "NOWAR: Not a admin");
require(nobotAddress_ != address(0), "NOWAR: Invalid address");
    bots[nobotAddress_] = false;
emit BotRemoved(nobotAddress_);
}
```

Suggestion: There should be a locking period so that the wallet cannot be locked for an indefinite. Period of time.

AUDIT SUMMARY

Project name – WE WANT NO WAR

Date: 28 Aug, 2024

Scope of Audit- Audit Ace was consulted to conduct the smart contract audit of the solidity source codes.

Audit Status: **High Risk Major Flag**

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	0	2	3	4	1
Acknowledged	0	0	0	0	0
Resolved	0	0	0	0	0

USED TOOLS

Tools:

1- Manual Review:
A line by line code review has been performed by audit ace team.

2- BSC Test Network: All tests were conducted on the BSC Test network, and each test has a corresponding transaction attached to it. These tests can be found in the "Functional Tests" section of the report.

3- Slither :
The code has undergone static analysis using Slither.

Testnet version:
The tests were performed using the contract deployed on the BSC Testnet, which can be found at the following address:

https://testnet.bscscan.com/address/0x3ee9eb85097a25874d9ef3ab901d073d00875fa8

Token Information

Token Address:
0xaE6D2712b1949314F81993b5965c0Ad7a44FF6e2

Name: WE WANT NO WAR

Symbol: NOWAR

Decimals: 18

Network: Ether Scan

Token Type: ERC-20

Owner: 0x522Dc688Fdcd93037C82AF7f74667705E71EF097

Deployer:
0x522Dc688Fdcd93037C82AF7f74667705E71EF097

Token Supply: 1,000,000,000

Checksum: b45f4a3f449246571a8b891ae4723d05

Testnet:
https://testnet.bscscan.com/address/0x3ee9eb85097a25874d9ef3ab901d073d00875fa8

TOKEN OVERVIEW

Buy Fee: 0-2%

Sell Fee: 0-2%

Transfer Fee: 0-0%

Fee Privilege: Owner

Ownership: Owned

Minting: None

Max Tx: No

Blacklist: Yes

AUDIT METHODOLOGY

The auditing process will follow a routine as special considerations by Auditace:

- Review of the specifications, sources, and instructions provided to Auditace to make sure the contract logic meets the intentions of the client without exposing the user's funds to risk.
- Manual review of the entire codebase by our experts, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
- Specification comparison is the process of checking whether the code does what the specifications, sources, and instructions provided to Auditace describe.
- Test coverage analysis determines whether the test cases are covering the code and how much code isexercised when we run the test cases.
- Symbolic execution is analysing a program to determine what inputs cause each part of a program to execute.
- Reviewing the codebase to improve maintainability, security, and control based on the established industry and academic practices.

VULNERABILITY CHECKLIST

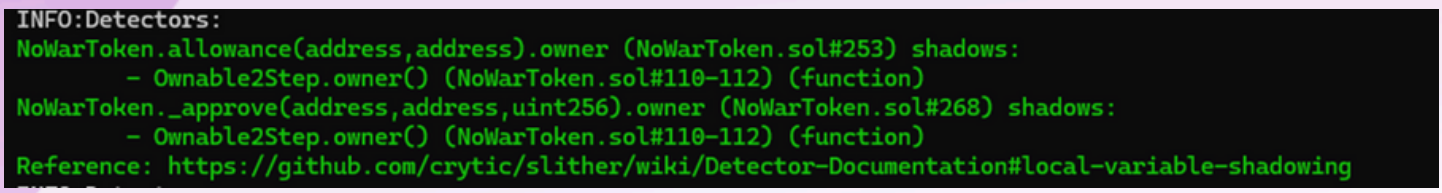
- [x] Return values of low-level calls
- [x] Private modifier
- [x] Multiple Sends
- [x] Using Suicide
- [x] Gas Limitand Loops
- [x] Address hardcoded
- [x] Exception Disorder
- [x] Using inline assembly
- [x] Divide before multiply
- [x] Missing Zero Address Validation
- [x] Compiler version not fixed
- [x] **Gasless Send**
- [x] Using block.timestamp
- [x] Re-entrancy
- [x] Tautology or contradiction
- [x] Timestamp Dependence
- [x] Revert/require functions
- [x] Use of tx.origin
- [x] Integer overflow/underflow
- [x] Dangerous strict equalities
- [x] Using SHA3
- [x] Using throw

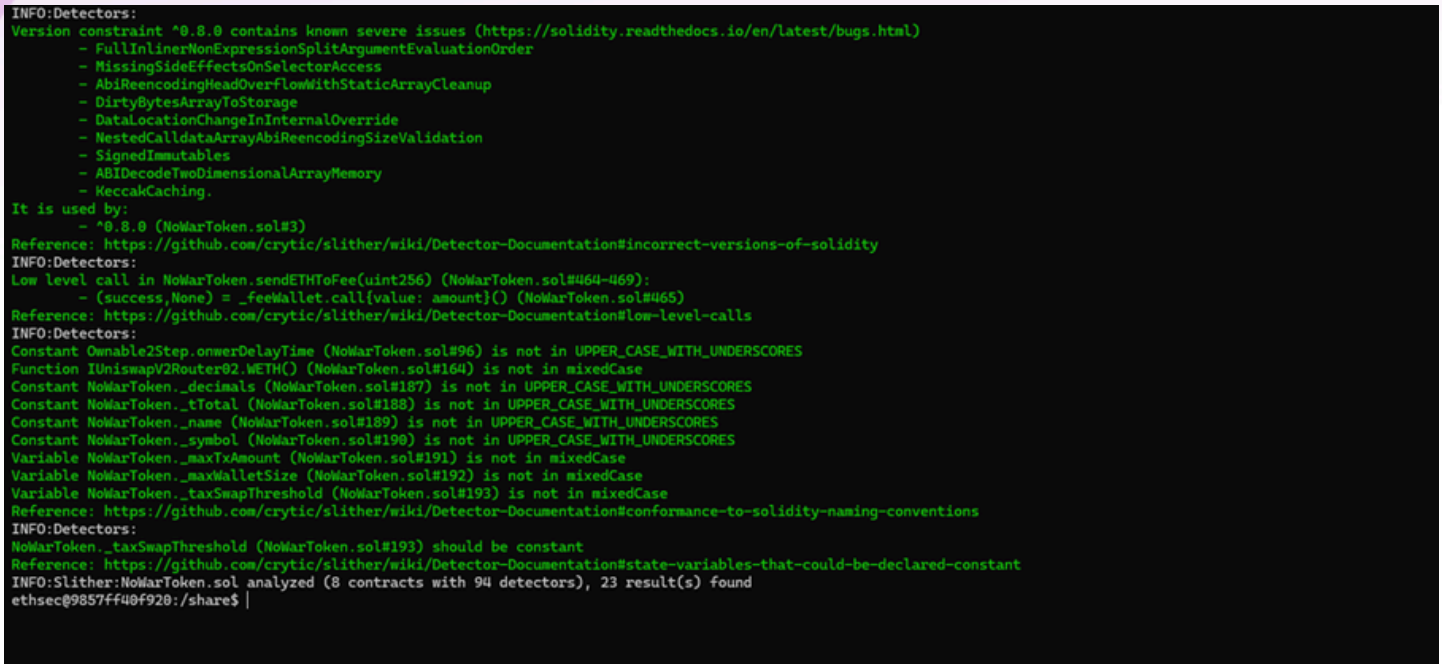
INHERITANCE TREE

POINTS TO NOTE

- The owner can setFeeWallet.
- The owner can add/remove admin.
- The owner remove limits.
- The owner can open trading..
- The owner can add/delete bots.
- The owner can manual swap.
- The owner can rescueEth and rescue Nowar.

STATIC ANALYSIS





Result => A static analysis of contract's source code has been performed using slither,

No major issues were found in the output

FUNCTIONAL TESTING

1- Approve (passed):

https://testnet.bscscan.com/tx/0xdd351b6edb3e5bed91e98a79a2cf20f0f7ac1920fc247ab603ce213fa04d506e

2- Add Bot (passed):

https://testnet.bscscan.com/tx/0x03c7f6a327b7b25fc0b996bafdc3e7b60af720d887ac7cb9adf866be818e14b4

3- Del Bots (passed):

https://testnet.bscscan.com/tx/0xd50264fc8d5ba08a4d801fc311312b2088a04e5904cf0e0a0d70237e86d59984

4- Add Admin (passed):

https://testnet.bscscan.com/tx/0x34f2ddc868a930c7209d67ea9c5e4b7a569c6bbf7963cdfdd16fc36ac127279e

5- Set Fee Wallet (passed):

https://testnet.bscscan.com/tx/0xa1ad61cbadec008547f1d505fe455733b4e3cee10ff28e5c28f154901373df91

CLASSIFICATION OF RISK

Severity	Description
Critical	These vulnerabilities could be exploited easily and can lead to asset loss, data loss, asset, or data manipulation. They should be fixed right away.
High-Risk	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.
Medium-Risk	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.
Low-Risk	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.
Gas Optimization /Suggestion	A vulnerability that has an informational character but is not affecting any of the code.

Findings

Severity	Found
Critical	0
High-Risk	2
Medium-Risk	3
Low-Risk	4
Gas Optimization / Suggestions	1

MANUAL TESTING

Centralization – Enabling Trades

Severity: High

Function: openTrading

Status: Open

Overview:

The EnableTrading function permits only the contract owner to activate trading capabilities. Until this function is executed, no investors can buy, sell, or transfer their tokens. This places a high degree of control and centralization in the hands of the contract owner.

```
function openTrading() external {
require(hasRole(ADMIN_ROLE, msg.sender), "NOWAR: Not an admin");
require(!tradingOpen, "NOWAR: Trading is already open");

    swapEnabled = true;
    tradingOpen = true;
```

Suggestion:

To reduce centralization and potential manipulation, consider one of the following approaches:

1.Automatically enable trading after a specified condition, such as the completion of a presale, is met.

2.If manual activation is still desired, consider transferring the ownership of the contract to a trustworthy, third-party entity like a certified "PinkSale Safu" developer. This can give investors more confidence in the eventual activation of trading capabilities, mitigating concerns of potential bad-faith actions by the original owner.

MANUAL TESTING

Centralization – Owner can blacklist wallets.

Severity: High

Function: addbot, delBots

Status: Open

Overview:
The owner can blacklist wallets from transferring of tokens for an indefinite period of time which is not recommended. Which can lock user's token.

```
function addBot(address botAddress_) public {
require(hasRole(ADMIN_ROLE, msg.sender), "NOWAR: Not a admin");
require(botAddress_ != address(0), "NOWAR: Invalid address");
    bots[botAddress_] = true;
emit BotAdded(botAddress_);
}

function delBots(address nobotAddress_) public {
require(hasRole(ADMIN_ROLE, msg.sender), "NOWAR: Not a admin");
require(nobotAddress_ != address(0), "NOWAR: Invalid address");
    bots[nobotAddress_] = false;
emit BotRemoved(nobotAddress_);
}
```

Suggestion: There should be a locking period so that the wallet cannot be locked for an indefinite. Period of time.

MANUAL TESTING

Centralization – The owner can regain ownership.

Severity: Medium

Function: pendingOwner

Status: Open

Overview:

The owner can regain ownership after transferring it with the following steps:

1. Call lock function to set previous owner to the own address
2. Call unlock function to get ownership back
3. Transfer/renounce ownership
4. Call unlock function to get ownership back

```
abstract contract Ownable2Step is Context, AccessControl {
address private _owner;
address private _newOwnerCandidate;
uint private changeOwnershipTimestamp;
uint256 private constant onwerDelayTime = 7 days;

event OwnershipTransferred(address indexed previousOwner, address indexed
newOwner);
event NewOwnerCandidateSet(address indexed previousCandidate, address
indexed newCandidate);
event OwnershipRecoveryInitiated(address indexed currentOwner, address
indexed newOwnerCandidate);
event OwnershipRecovered(address indexed previousOwner, address indexed
newOwner);

constructor () {
address msgSender = _msgSender();
require(msgSender != address(0), "Invalid message sender address");
    _owner = msgSender;
emit OwnershipTransferred(address(0), msgSender);
```

MANUAL TESTING

```
}

function owner() public view returns (address) {
return _owner;
}

function newOwnerCandidate() public view returns (address) {
return _newOwnerCandidate;
}

function transferOwnership(address newOwnerCandidate_) public {
require(msg.sender == _owner, "Caller not onwer");
require(newOwnerCandidate_ != address(0), "Invalid new owner candidate
address");
require(newOwnerCandidate_ != _owner, "New onwer is the same as the
current one");
   _newOwnerCandidate = newOwnerCandidate_;
   changeOwnershipTimestamp = block.timestamp + onwerDelayTime; // 7
days delay for safety
emit NewOwnerCandidateSet(_newOwnerCandidate, newOwnerCandidate_);
}

function acceptOwnership() public {
require(_newOwnerCandidate != address(0), "Invalid new owner candidate
address");
require(msg.sender == _newOwnerCandidate, "Only new owner candidate can
initiate recovery");
require(block.timestamp >= changeOwnershipTimestamp, "Ownership change
period not elapsed");

address previousOwner = _owner;
   _owner = _newOwnerCandidate;
delete _newOwnerCandidate;

emit OwnershipRecovered(previousOwner, _owner);
emit OwnershipTransferred(previousOwner, _owner);
```

MANUAL TESTING

```
_revokeRole(ADMIN_ROLE, previousOwner);
_grantRole(ADMIN_ROLE, _owner);
}

function cancelOwnershipRecovery() public {
require(msg.sender == _owner, "Caller not onwer");
require(_newOwnerCandidate != address(0), "No ownership recovery in
progress");
delete _newOwnerCandidate;
emit NewOwnerCandidateSet(_newOwnerCandidate, address(0));
}
}
```

Suggestion: Make sure to set the previous ownership back to address zero after using the unlock function.

MANUAL TESTING

Centralization – Missing Require Check.

Severity: Medium

Function: setFeeWallet

Status: Open

Overview:
The owner can set any arbitrary address excluding zero address as this is not recommended because if the owner will set the address to the contract address, then the Eth will not be sent to that address and the transaction will fail and this will lead to a potential honeypot in the contract.

```
 function setFeeWallet(address newFeeWallet_) public {
require(hasRole(ADMIN_ROLE, msg.sender), "NOWAR: Not an admin");
require(newFeeWallet_ != address(0), "Invalid address: zero address");
require(newFeeWallet_ != _feeWallet, "New fee wallet address is the same as the current one");

address payable previousFeeWallet_ = _feeWallet;
    _isExcludedFromFee[previousFeeWallet_] = false;
    _isExcludedFromFee[newFeeWallet_] = true;
    _feeWallet = payable(newFeeWallet_);
emit FeeWalletUpdated(previousFeeWallet_, newFeeWallet_);
}
```

Suggestion: It is recommended that the address should not be able to set as a contract address.

MANUAL TESTING

Centralization – Liquidity is added to EOA.

Severity: Medium

function: addLiquidityETH

Status: Open

Overview:

Liquidity is added to EOA. It may be drained by the owner.

```
IUniswapV2Router02 _uniswapV2Router =
IUniswapV2Router02(0xD99D1c33F9fC3444f8101754aBC46c52416550D1);
    uniswapV2Router = _uniswapV2Router;
_approve(address(this), address(uniswapV2Router), _tTotal);
    uniswapV2Pair =
IUniswapV2Factory(_uniswapV2Router.factory()).createPair(address(this),
_uniswapV2Router.WETH());
    uniswapV2Router.addLiquidityETH{value: address(this).balance}(
address(this),
balanceOf(address(this)),
0,
0,
owner(),
block.timestamp
    )
```

Suggestion:

It is suggested that the address should be a contract address or a dead address.

MANUAL TESTING

Centralization – Missing Events

Severity: Low

Subject: Missing Events

Status: Open

Overview:
They serve as a mechanism for emitting and recording data onto the blockchain, making it transparent and easily accessible.

```
function getFeeWallet() public view returns (address payable) {
}

function addAdmin(address newAdmin) public {
require(hasRole(ADMIN_ROLE, msg.sender), "NOWAR: not an admin");
_grantRole(ADMIN_ROLE, newAdmin);
}

function removeAdmin(address delAdmin) public {
require(hasRole(ADMIN_ROLE, msg.sender), "NOWAR: not an admin");
require(delAdmin != owner(), "NOWAR: can not remove onwer");
_revokeRole(ADMIN_ROLE, delAdmin);
}
```

Suggestion: Emit an event for critical changes.

MANUAL TESTING

Centralization – Local variable Shadowing

Severity: Low

Subject: Variable Shadowing

Status: Open

Overview:

```
function _approve(address owner, address spender, uint256 amount) private {
require(owner != address(0), "NOWAR: approve from the zero address");
require(spender != address(0), "NOWAR: approve to the zero address");
    _allowances[owner][spender] = amount;
emit Approval(owner, spender, amount);
}
 function allowance(address owner, address spender) public view override
returns (uint256) {
return _allowances[owner][spender];
}
```

Suggestion:
Rename the local variables that shadow another component.

Centralization – Local variable Shadowing
Severity: Low
Subject: Variable Shadowing
Status: Open
Overview:

```
function _approve(address owner, address spender, uint256 amount) private {
require(owner != address(0), "NOWAR: approve from the zero address");
require(spender != address(0), "NOWAR: approve to the zero address");
    _allowances[owner][spender] = amount;
emit Approval(owner, spender, amount);
}
```

MANUAL TESTING

```
function allowance(address owner, address spender) public view override
returns (uint256) {
return _allowances[owner][spender];
}
```

Suggestion:
Rename the local variables that shadow another component.

MANUAL TESTING

Centralization – Missing Zero Address

Severity: Low

Subject: Zero Check

Status: Open

Overview:
functions can take a zero address as a parameter (0x00000...). If a function parameter of address type is not properly validated by checking for zero addresses, there could be serious consequences for the contract's functionality.

```
function getFeeWallet() public view returns (address payable) {
}
```

Suggestion:
It is suggested that the address should not be zero or dead.

MANUAL TESTING

Centralization – Remove the safe math library.

Severity: Low

Status: Open

Line Number: 11-46

Overview:

The Safe Math library is no longer needed for Solidity version 0.8 and above. This is because Solidity 0.8 includes checked arithmetic operations by default. All of Safe Math's methods are now inherited into Solidity programming.

MANUAL TESTING

Optimization

Severity: Informational

Subject: Floating Pragma.

Status: Open

Overview:
It is considered best practice to pick one compiler version and stick with it. With a floating pragma, contracts may accidentally be deployed using an outdated.
pragma solidity ^0.8.0;

Suggestion:
Adding the latest constant version of solidity is recommended, as this prevents the unintentional deployment of a contract with an outdated compiler that contains unresolved bugs.

DISCLAIMER

All the content provided in this document is for general information only and should not be used as financial advice or a reason to buy any investment. Team provides no guarantees against the sale of team tokens or the removal of liquidity by the project audited in this document. Always Do your own research and protect yourselves from being scammed. The Auditace team has audited this project for general information and only expresses their opinion based on similar projects and checks from popular diagnostic tools. Under no circumstances did Auditace receive a payment to manipulate those results or change the awarding badge that we will be adding in our website. Always Do your own research and protect yourselves from scams. This document should not be presented as a reason to buy or not buy any particular token. The Auditace team disclaims any liability for the resulting losses.

ABOUT AUDITACE

We specializes in providing thorough and reliable audits for Web3 projects. With a team of experienced professionals, we use cutting-edge technology and rigorous methodologies to evaluate the security and integrity of blockchain systems. We are committed to helping our clients ensure the safety and transparency of their digital assets and transactions.

https://auditace.tech/

https://t.me/Audit_Ace

https://twitter.com/auditace_

https://github.com/Audit-Ace