Building the Futuristic Blockchain Ecosystem

SECURITY AUDIT REPORT

MAGA Barron

TOKEN OVERVIEW

Risk Findings

Severity	Found
High	1
Medium	1
Low	1
Informational	2

Centralization Risks

Owner Privileges	Description
Can Owner Set Taxes >25% ?	Not Detected
Owner needs to enable trading ?	Detected
Can Owner Disable Trades ?	Not Detected
Can Owner Mint ?	Not Detected
Can Owner Blacklist ?	Not Detected
Can Owner set Max Wallet amount ?	Not Detected
Can Owner Set Max TX amount ?	Not Detected

TABLE OF CONTENTS

OVERVIEW

The Expelee team has performed a line-by-line manual analysis and automated review of the smart contract. The smart contract was analysed mainly for common smart contract vulnerabilities, exploits, and manipulation hacks. According to the smart contract audit:

Audit Result	**Passed with high risk**
Audit Date	**11 June 2024**

CONTRACT DETAILS

Token Address: 0xE6026D951aC374568cbb0342b7d1335eBa86da08

Name: MAGA Barron

Symbol: MAGABARRON

Decimals: 18

Network: BscScan

Token Type: BEP-20

Owner: 0xCAAc78CfC999bCc67f044349eC902E5F14AEfEA2

Deployer: 0xCAAc78CfC999bCc67f044349eC902E5F14AEfEA2

Token Supply: 9000000000

Checksum: B17acbefe2a12642d388659dffd20211

Testnet:
https://testnet.bscscan.com/address/0xa46E634aC783627409B0945740540D96aA5809b5#code

AUDIT METHODOLOGY

Audit Details

Our comprehensive audit report provides a full overview of the audited system's architecture, smart contract codebase, and details on any vulnerabilities found within the system.

Audit Goals

The audit goal is to ensure that the project is built to protect investors and users, preventing potentially catastrophic vulnerabilities after launch , that lead to scams and rugpulls.

Code Quality

Our analysis includes both automatic tests and manual code analysis for the following aspects:

- Exploits
- Back-doors
- Vulnerability
- Accuracy
- Readability

Tools

- Manual Review: The code has undergone a line-by-line review by the Ace team.
- 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.
- Slither: The code has undergone static analysis using Slither.

VULNERABILITY CHECKS

Design Logic	Passed
Compiler warnings	Passed
Private user data leaks	Passed
Timestamps dependence	Passed
Integer overflow and underflow	Passed
Race conditions & reentrancy. Cross-function race conditions	Passed
Possible delays in data delivery	Passed
Oracle calls	Passed
Front Running	Passed
DoS with Revert	Passed
DoS with block gas limit	Passed
Methods execution permissions	Passed
Economy model	Passed
Impact of the exchange rate on the logic	Passed
Malicious event log	Passed
Scoping and declarations	Passed
Uninitialized storage pointers	Passed
Arithmetic accuracy	Passed
Cross-function race conditions	Passed
Safe Zepplin module	Passed

RISK CLASSIFICATION

When performing smart contract audits, our specialists look for known vulnerabilities as well as logical and acces control issues within the code. The exploitation of these issues by malicious actors may cause serious financial damage to projects that failed to get an audit in time. We categorize these vulnerabilities by the following levels:

High Risk

Issues on this level are critical to the smart contract's performance/functionality and should be fixed before moving to a live environment.

Medium Risk

Issues on this level are critical to the smart contract's performance/functionality and should be fixed before moving to a live environment.

Low Risk

Issues on this level are minor details and warning that can remain unfixed.

Informational

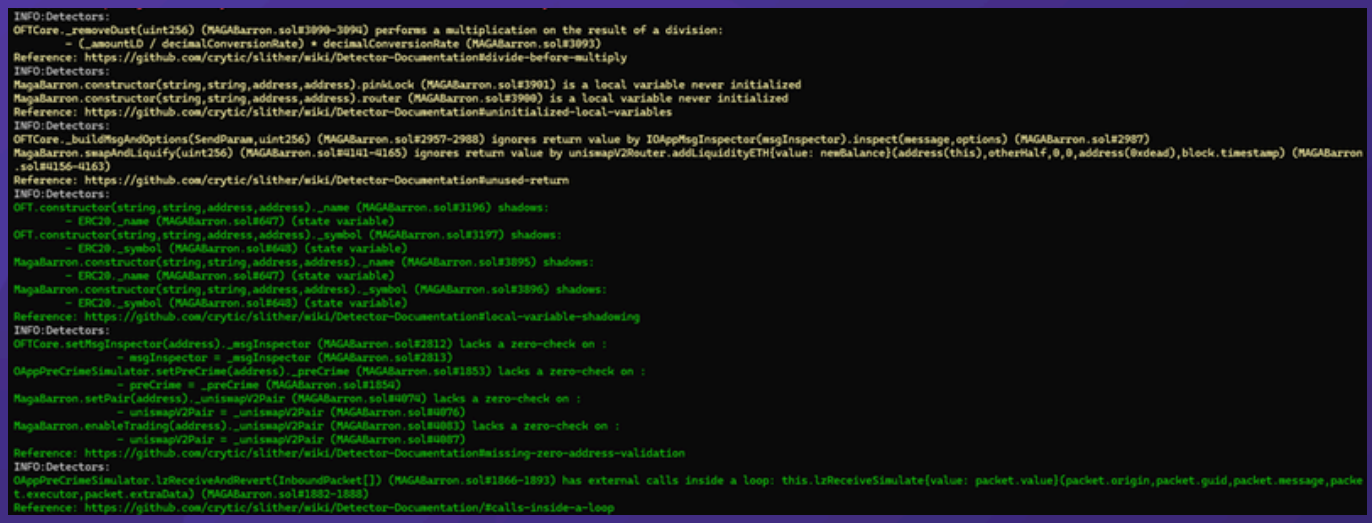
Issues on this level are minor details and warning that can remain unfixed.

expelee

INHERITANCE TREE

expelee

STATIC ANALYSIS

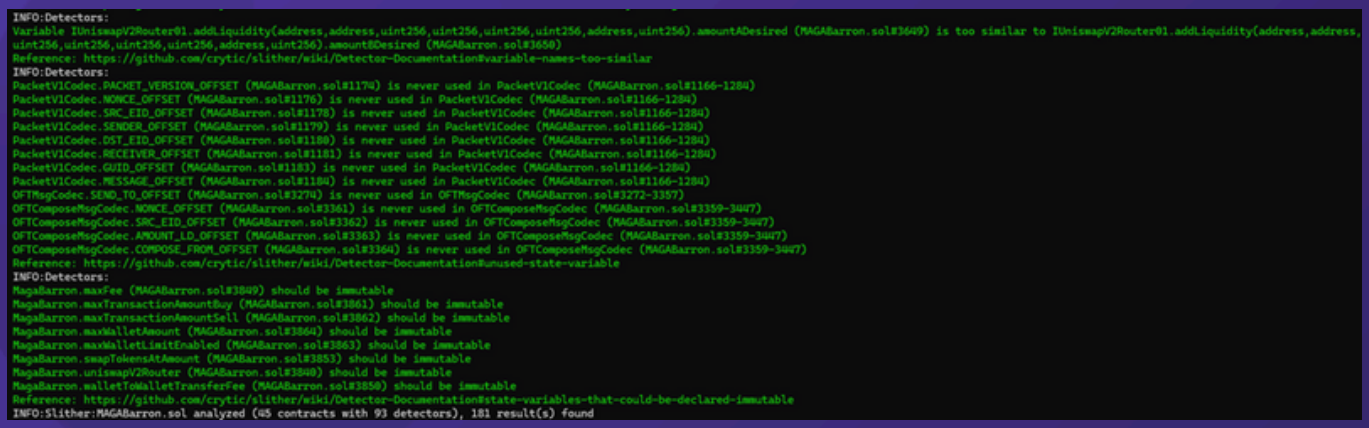


```
INFO:Detectors:
OFTCore._removeDust(uint256) (MAGABarron.sol#3090-3094) performs a multiplication on the result of a division:
        - (_amountLD / decimalConversionRate) * decimalConversionRate (MAGABarron.sol#3093)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
MagaBarron.constructor(string,string,address,address).pinkLock (MAGABarron.sol#3901) is a local variable never initialized
MagaBarron.constructor(string,string,address,address).router (MAGABarron.sol#3900) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
OFTCore._buildMsgAndOptions(SendParam,uint256) (MAGABarron.sol#2957-2988) ignores return value by IOAppMsgInspector(msgInspector).inspect(message,options) (MAGABarron.sol#2987)
MagaBarron.swapAndLiquify(uint256) (MAGABarron.sol#4141-4165) ignores return value by uniswapV2Router.addLiquidityETH{value: newBalance}(address(this),otherHalf,0,0,address(0xdead),block.timestamp) (MAGABarron
.sol#4156-4163)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
OFT.constructor(string,string,address,address)._name (MAGABarron.sol#3196) shadows:
        - ERC20._name (MAGABarron.sol#647) (state variable)
OFT.constructor(string,string,address,address)._symbol (MAGABarron.sol#3197) shadows:
        - ERC20._symbol (MAGABarron.sol#648) (state variable)
MagaBarron.constructor(string,string,address,address)._name (MAGABarron.sol#3895) shadows:
        - ERC20._name (MAGABarron.sol#647) (state variable)
MagaBarron.constructor(string,string,address,address)._symbol (MAGABarron.sol#3896) shadows:
        - ERC20._symbol (MAGABarron.sol#648) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
OFTCore.setMsgInspector(address)._msgInspector (MAGABarron.sol#2812) lacks a zero-check on :
                - msgInspector = _msgInspector (MAGABarron.sol#2813)
OAppPreCrimeSimulator.setPreCrime(address)._preCrime (MAGABarron.sol#1853) lacks a zero-check on :
                - preCrime = _preCrime (MAGABarron.sol#1854)
MagaBarron.setPair(address)._uniswapV2Pair (MAGABarron.sol#4074) lacks a zero-check on :
                - uniswapV2Pair = _uniswapV2Pair (MAGABarron.sol#4076)
MagaBarron.enableTrading(address)._uniswapV2Pair (MAGABarron.sol#4083) lacks a zero-check on :
                - uniswapV2Pair = _uniswapV2Pair (MAGABarron.sol#4087)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
OAppPreCrimeSimulator.lzReceiveAndRevert(InboundPacket[]) (MAGABarron.sol#1866-1893) has external calls inside a loop: this.lzReceiveSimulate{value: packet.value}(packet.origin,packet.guid,packet.message,packe
t.executor,packet.extraData) (MAGABarron.sol#1882-1888)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
```

```
INFO:Detectors:
Address.functionDelegateCall(address,bytes) (MAGABarron.sol#161-167) is never used and should be removed
Address.functionStaticCall(address,bytes) (MAGABarron.sol#149-155) is never used and should be removed
Address.verifyCallResult(bool,bytes) (MAGABarron.sol#195-204) is never used and should be removed
AddressCast.toAddress(bytes) (MAGABarron.sol#1328-1333) is never used and should be removed
AddressCast.toAddress(bytes32) (MAGABarron.sol#1322-1326) is never used and should be removed
AddressCast.toBytes(bytes32,uint256) (MAGABarron.sol#1307-1320) is never used and should be removed
AddressCast.toBytes32(address) (MAGABarron.sol#1301-1305) is never used and should be removed
AddressCast.toBytes32(bytes) (MAGABarron.sol#1290-1299) is never used and should be removed
Context._contextSuffixLength() (MAGABarron.sol#46-48) is never used and should be removed
Context._msgData() (MAGABarron.sol#42-44) is never used and should be removed
OFTComposeMsgCodec.addressToBytes32(address) (MAGABarron.sol#3435-3437) is never used and should be removed
OFTComposeMsgCodec.amountLD(bytes) (MAGABarron.sol#3406-3408) is never used and should be removed
OFTComposeMsgCodec.bytes32ToAddress(bytes32) (MAGABarron.sol#3444-3446) is never used and should be removed
OFTComposeMsgCodec.composeFrom(bytes) (MAGABarron.sol#3415-3417) is never used and should be removed
OFTComposeMsgCodec.composeMsg(bytes) (MAGABarron.sol#3424-3428) is never used and should be removed
OFTComposeMsgCodec.nonce(bytes) (MAGABarron.sol#3388-3390) is never used and should be removed
OFTComposeMsgCodec.srcEid(bytes) (MAGABarron.sol#3397-3399) is never used and should be removed
PacketDecoder.decode(bytes) (MAGABarron.sol#1980-1992) is never used and should be removed
PacketDecoder.decode(bytes[],uint256[]) (MAGABarron.sol#2000-2011) is never used and should be removed
PacketV1Codec.dstEid(bytes) (MAGABarron.sol#1249-1251) is never used and should be removed
PacketV1Codec.encode(Packet) (MAGABarron.sol#1186-1199) is never used and should be removed
PacketV1Codec.encodePacketHeader(Packet) (MAGABarron.sol#1201-1213) is never used and should be removed
PacketV1Codec.encodePayload(Packet) (MAGABarron.sol#1215-1219) is never used and should be removed
PacketV1Codec.guid(bytes) (MAGABarron.sol#1263-1265) is never used and should be removed
PacketV1Codec.header(bytes) (MAGABarron.sol#1221-1225) is never used and should be removed
PacketV1Codec.message(bytes) (MAGABarron.sol#1267-1271) is never used and should be removed
PacketV1Codec.nonce(bytes) (MAGABarron.sol#1231-1233) is never used and should be removed
PacketV1Codec.payload(bytes) (MAGABarron.sol#1273-1277) is never used and should be removed
PacketV1Codec.payloadHash(bytes) (MAGABarron.sol#1279-1283) is never used and should be removed
PacketV1Codec.receiver(bytes) (MAGABarron.sol#1253-1255) is never used and should be removed
PacketV1Codec.receiverB20(bytes) (MAGABarron.sol#1257-1261) is never used and should be removed
PacketV1Codec.sender(bytes) (MAGABarron.sol#1239-1241) is never used and should be removed
PacketV1Codec.senderAddressB20(bytes) (MAGABarron.sol#1243-1247) is never used and should be removed
PacketV1Codec.srcEid(bytes) (MAGABarron.sol#1235-1237) is never used and should be removed
PacketV1Codec.version(bytes) (MAGABarron.sol#1227-1229) is never used and should be removed
SafeERC20._callOptionalReturnBool(IERC20,bytes) (MAGABarron.sol#358-371) is never used and should be removed
SafeERC20.forceApprove(IERC20,address,uint256) (MAGABarron.sol#314-331) is never used and should be removed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (MAGABarron.sol#291-307) is never used and should be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (MAGABarron.sol#278-285) is never used and should be removed
SafeERC20.safeTransfer(IERC20,address,uint256) (MAGABarron.sol#254-256) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.20 (MAGABarron.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.8.18.
solc-0.8.20 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

expelee

STATIC ANALYSIS



```
INFO:Detectors:
Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (MAGABarron.sol#3649) is too similar to IUniswapV2Router01.addLiquidity(address,address,
uint256,uint256,uint256,uint256,address,uint256).amountBDesired (MAGABarron.sol#3650)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
PacketV1Codec.PACKET_VERSION_OFFSET (MAGABarron.sol#1174) is never used in PacketV1Codec (MAGABarron.sol#1166-1284)
PacketV1Codec.NONCE_OFFSET (MAGABarron.sol#1176) is never used in PacketV1Codec (MAGABarron.sol#1166-1284)
PacketV1Codec.SRC_EID_OFFSET (MAGABarron.sol#1178) is never used in PacketV1Codec (MAGABarron.sol#1166-1284)
PacketV1Codec.SENDER_OFFSET (MAGABarron.sol#1179) is never used in PacketV1Codec (MAGABarron.sol#1166-1284)
PacketV1Codec.DST_EID_OFFSET (MAGABarron.sol#1180) is never used in PacketV1Codec (MAGABarron.sol#1166-1284)
PacketV1Codec.RECEIVER_OFFSET (MAGABarron.sol#1181) is never used in PacketV1Codec (MAGABarron.sol#1166-1284)
PacketV1Codec.GUID_OFFSET (MAGABarron.sol#1183) is never used in PacketV1Codec (MAGABarron.sol#1166-1284)
PacketV1Codec.MESSAGE_OFFSET (MAGABarron.sol#1184) is never used in PacketV1Codec (MAGABarron.sol#1166-1284)
OFTMsgCodec.SEND_TO_OFFSET (MAGABarron.sol#3274) is never used in OFTMsgCodec (MAGABarron.sol#3272-3357)
OFTComposeMsgCodec.NONCE_OFFSET (MAGABarron.sol#3361) is never used in OFTComposeMsgCodec (MAGABarron.sol#3359-3447)
OFTComposeMsgCodec.SRC_EID_OFFSET (MAGABarron.sol#3362) is never used in OFTComposeMsgCodec (MAGABarron.sol#3359-3447)
OFTComposeMsgCodec.AMOUNT_LD_OFFSET (MAGABarron.sol#3363) is never used in OFTComposeMsgCodec (MAGABarron.sol#3359-3447)
OFTComposeMsgCodec.COMPOSE_FROM_OFFSET (MAGABarron.sol#3364) is never used in OFTComposeMsgCodec (MAGABarron.sol#3359-3447)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable
INFO:Detectors:
MagaBarron.maxFee (MAGABarron.sol#3849) should be immutable
MagaBarron.maxTransactionAmountBuy (MAGABarron.sol#3861) should be immutable
MagaBarron.maxTransactionAmountSell (MAGABarron.sol#3862) should be immutable
MagaBarron.maxWalletAmount (MAGABarron.sol#3864) should be immutable
MagaBarron.maxWalletLimitEnabled (MAGABarron.sol#3863) should be immutable
MagaBarron.swapTokensAtAmount (MAGABarron.sol#3853) should be immutable
MagaBarron.uniswapV2Router (MAGABarron.sol#3840) should be immutable
MagaBarron.walletToWalletTransferFee (MAGABarron.sol#3850) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Slither:MAGABarron.sol analyzed (45 contracts with 93 detectors), 181 result(s) found
```

TESTNET VERSION

1- Enable Trading (passed):
https://testnet.bscscan.com/tx/0x6dd9703d7ac3928ca02d272c4184faf5e223e03b4804b39724af085c3b0d0aa9

2- Change Marketing Wallet (passed):
https://testnet.bscscan.com/tx/0xa7128632f328748643599ea20fdc137b843f936feb4ba0bad3bf806228e79363

3- Set Delegate (passed):
https://testnet.bscscan.com/tx/0x50dc96c5f5d815de01b209c58ffc811d470e091e216c5f6e20844cd0c53ae7bd

4- Update Buy Fees (passed):
https://testnet.bscscan.com/tx/0x52bbc238e12d2c5a702b0b4a5298db00a0a203c3328557320452f3627a72fa02

5- Update Sell Fees (passed):
https://testnet.bscscan.com/tx/0x676d40a23f228464c6f808657b3234e5cf17eeffde7792c0f5fd984039ab4aec

6- Set Pair (passed):
https://testnet.bscscan.com/tx/0xb67adb1c4f85f19a966ecb387cdabe25f1ed6fed0b60b0a6e4444d6f53aa4919

7- Pair Is Set (passed):
https://testnet.bscscan.com/tx/0x38a739ef6f2f5e4033ce71f5ab2f77f4c6a0e8b5c8f59ba6b46ebde61cf83ef3

MANUAL REVIEW

Severity Criteria

Expelee assesses the severity of disclosed vulnerabilities according to methodology based on OWASP standarts.

Vulnerabilities are dividend into three primary risk categroies:
High
Medium
Low

High-level considerations for vulnerabilities span the following key areas when conducting assessments:

- Malicious input handling
- Escalation of privileges
- Arithmetic
- Gas use

Overall Risk Severity				
Impact	HIGH	Medium	High	Critical
	MEDIUM	Low	Medium	High
	LOW	Note	Low	Medium
		LOW	MEDIUM	HIGH
	Likelihood			

HIGH RISK FINDING

Centralization – Enabling Trades

Severity: High

Function: OpenTrading

Status: Open

Overview:

The OpenTrading 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 enableTrading(address _uniswapV2Pair) external onlyOwner {
    require(!tradingEnabled, "Trading already enabled.");
    tradingEnabled = true;
    swapEnabled = true;
    uniswapV2Pair = _uniswapV2Pair;
  }
```

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.

MEDIUM RISK FINDING

Centralization – Missing Require Check.

Severity: Medium

Function: changeMarketingWallet/swapAndSendMarketing

Status: Open

Overview:

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

```
function changeMarketingWallet(
    address _marketingWallet
  ) external onlyOwner {
    require(
      _marketingWallet != marketingWallet,
      "Marketing wallet is already that address"
    );
    require(
      _marketingWallet != address(0),
      "Marketing wallet cannot be the zero address"
    );
    marketingWallet = _marketingWallet;

    emit MarketingWalletChanged(marketingWallet);
  }
uint256 newBalance = address(this).balance - initialBalance;
    payable(marketingWallet).sendValue(newBalance)
```

Suggestion:

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

LOW RISK FINDING

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 setPair(address _uniswapV2Pair) external onlyOwner {
require(!pairSet, "Cannot change the pair");
    uniswapV2Pair = _uniswapV2Pair;
  }
function enableTrading(address _uniswapV2Pair) external onlyOwner {
require(!tradingEnabled, "Trading already enabled.");
    tradingEnabled = true;
    swapEnabled = true;
    uniswapV2Pair = _uniswapV2Pair;
  }
function setPreCrime(address _preCrime) external;
function setMsgInspector(address _msgInspector) public virtual onlyOwner {
    msgInspector = _msgInspector;
emit MsgInspectorSet(_msgInspector);
  }
```

INFORMATIONAL & OPTIMIZATIONS

Optimization

Severity: Informational

Subject: Floating Pragma Solidity version

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 ^0.8.20;

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.

INFORMATIONAL & OPTIMIZATIONS

Optimization

Severity: Optimization

Subject: Remove unused code.

Status: Open

Overview:

Unused variables are allowed in Solidity, and they do. not pose a direct security issue. It is the best practice though to avoid them.

```
event MaxWalletLimitStateChanged(bool maxWalletLimit);
event MaxWalletLimitAmountChanged(uint256 maxWalletAmount);
event MaxTransactionLimitAmountChanged(
    uint256 maxTransactionAmountBuy,
    uint256 maxTransactionAmountSell
  );
event SwapTokensAtAmountUpdated(uint256 swapTokensAtAmount);
function _msgData() internal view virtual returns (bytes calldata) {
    return msg.data;
  }
function _contextSuffixLength() internal view virtual returns (uint256) {
    return 0;
  }
function safeTransfer(IERC20 token, address to, uint256 value) internal {
    _callOptionalReturn(token, abi.encodeCall(token.transfer, (to,
value)));
  }
function safeIncreaseAllowance(
    IERC20 token,
    address spender,
    uint256 value
```

INFORMATIONAL & OPTIMIZATIONS

```
) internal {
    uint256 oldAllowance = token.allowance(address(this), spender);
    forceApprove(token, spender, oldAllowance + value);
  }
function safeDecreaseAllowance(
    IERC20 token,
    address spender,
    uint256 requestedDecrease
  ) internal {
    unchecked {
      uint256 currentAllowance = token.allowance(address(this),
spender);
      if (currentAllowance < requestedDecrease) {
revert SafeERC20FailedDecreaseAllowance(
          spender,
          currentAllowance,
          requestedDecrease
        );
      }
      forceApprove(token, spender, currentAllowance -
requestedDecrease);
    }
interface IERC20Permit
interface IERC721Errors
interface IERC1155Errors
function encode(
    Packet memory _packet
  ) internal pure returns (bytes memory encodedPacket) {
    encodedPacket = abi.encodePacked(
      PACKET_VERSION,
      _packet.nonce,
      _packet.srcEid,
```

INFORMATIONAL & OPTIMIZATIONS

```
      _packet.sender.toBytes32(),
      _packet.dstEid,
      _packet.receiver,
      _packet.guid,
      _packet.message
    );
  }
function encodePacketHeader(
    Packet memory _packet
  ) internal pure returns (bytes memory) {
    return
      abi.encodePacked(
        PACKET_VERSION,
        _packet.nonce,
        _packet.srcEid,
        _packet.sender.toBytes32(),
        _packet.dstEid,
        _packet.receiver
      );
  }
function encodePayload(
    Packet memory _packet
  ) internal pure returns (bytes memory) {
    return abi.encodePacked(_packet.guid, _packet.message);
  }
function header(
    bytes calldata _packet
  ) internal pure returns (bytes calldata) {
    return _packet[0:GUID_OFFSET];
  }
function version(bytes calldata _packet) internal pure returns (uint8) {
```

INFORMATIONAL & OPTIMIZATIONS

```
    return
uint8(bytes1(_packet[PACKET_VERSION_OFFSET:NONCE_OFFSET]));
  }
function senderAddressB20(
    bytes calldata _packet
  ) internal pure returns (address) {
    return sender(_packet).toAddress();
  }
function toAddress(
    bytes calldata _addressBytes
  ) internal pure returns (address result) {
    if (_addressBytes.length != 20) revert AddressCast_InvalidAddress();
    result = address(bytes20(_addressBytes));
  }
}
```

ABOUT EXPELEE

Expelee is a product-based aspirational Web3 start-up. Coping up with numerous solutions for blockchain security and constructing a Web3 ecosystem from deal making platform to developer hosting open platform, while also developing our own commercial and sustainable blockchain.

www.expelee.com

expeleeofficial

expelee

Expelee

expelee

expelee_official

expelee-co

expelee

Building the Futuristic Blockchain Ecosystem

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 guarantess against the sale of team tokens or the removal of liquidity by the project audited in this document.

Always do your own research and project yourselves from being scammed. The Expelee 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 Expelee receive a payment to manipulate those results or change the awarding badge that we will be adding in our website. Alway 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 Expelee team disclaims any liability for the resulting losses.