auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

INTEROP

DATED : 18 Sep 24'

MANUAL TESTING

Centralization – Enabling Trades

Severity: High

Function: EnableTrading

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 enableTrading() external onlyOwner{
    require(!tradingEnabled, "Trading already enabled.");
    tradingEnabled = true;
    swapEnabled = true;

    emit TradingEnabled(tradingEnabled);
  }
```

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.

AUDIT SUMMARY

Project name – INTEROP

Date: 18 Sep, 2024

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

Audit Status: **High Risk**

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	0	1	1	0	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/0xB68b36EA9fCfD910EE8541422dE4d856564E43a4#code

Token Information

Token Address:
0xA748F1edD376f494278c44660E002D7666C3D000

Name: INTEROP

Symbol: ITR

Decimals: 18

Network: BscScan

Token Type: BEP-20

Owner: 0x5DaEE303b2a46280fba1cEFD187AcE6333Ccc64f

Deployer: 0x5DaEE303b2a46280fba1cEFD187AcE6333Ccc64f

Token Supply: 100000000

Checksum: b17acbefe2a12642d388659dffd2f5g2

Testnet:
https://testnet.bscscan.com/address/0xB68b36EA9fCfD910EE8541422dE4d856564E43a4#code

TOKEN OVERVIEW

Buy Fee: 2-10%

Sell Fee: 2-10%

Transfer Fee: 2-10%

Fee Privilege: Owner

Ownership: Owned

Minting: None

Max Tx: None

Blacklist: No

AUDIT METHODOLOGY

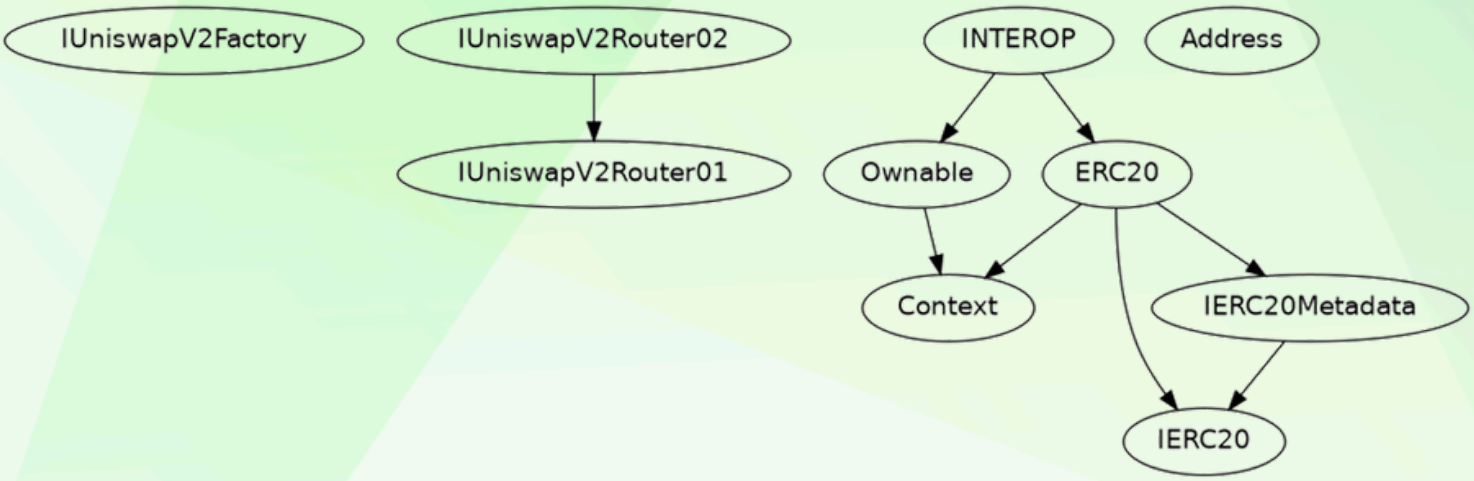
The auditing process will follow a routine as special considerations by Auditace:

- 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 transfer ownership.
- The owner can renounce ownership.
- The owner can Enable trading.
- The owner can claim stuck tokens.
- The owner can update the buy/sell/transfer fee to not more than 10%.
- The owner can exclude addresses from fees.
- The owner can update the fee receiver address.
- The owner can set swap tokens at an amount.

STATIC ANALYSIS

```
INFO:Detectors:
INTEROP.constructor().pinkLock (INTEROP.sol#270) is a local variable never initialized
INTEROP.constructor().router (INTEROP.sol#269) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
INTEROP.claimStuckTokens(address) (INTEROP.sol#318-326) ignores return value by address(msg.sender).sendValue(address(this).balance) (INTEROP.sol#321)
INTEROP.swapAndSendFee(uint256) (INTEROP.sol#429-451) ignores return value by address(feeReceiver).sendValue(newBalance) (INTEROP.sol#448)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
Reentrancy in INTEROP._transfer(address,address,uint256) (INTEROP.sol#373-419):
        External calls:
        - swapAndSendFee(contractTokenBalance) (INTEROP.sol#396)
                - (success,None) = recipient.call{value: amount}() (INTEROP.sol#62)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (INTEROP.sol#436-444)
                - address(feeReceiver).sendValue(newBalance) (INTEROP.sol#448)
        External calls sending eth:
        - swapAndSendFee(contractTokenBalance) (INTEROP.sol#396)
                - (success,None) = recipient.call{value: amount}() (INTEROP.sol#62)
        Event emitted after the call(s):
        - Transfer(sender,recipient,amount) (INTEROP.sol#206)
                - super._transfer(from,to,amount) (INTEROP.sol#418)
        - Transfer(sender,recipient,amount) (INTEROP.sol#206)
                - super._transfer(from,address(this),fees) (INTEROP.sol#415)
Reentrancy in INTEROP.swapAndSendFee(uint256) (INTEROP.sol#429-451):
        External calls:
        - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (INTEROP.sol#436-444)
        - address(feeReceiver).sendValue(newBalance) (INTEROP.sol#448)
        Event emitted after the call(s):
        - SwapAndSendFee(tokenAmount,newBalance) (INTEROP.sol#450)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Context._msgData() (INTEROP.sol#72-75) is never used and should be removed
ERC20._burn(address,uint256) (INTEROP.sol#217-228) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Version constraint 0.8.17 contains known severe issues (https://solidity.readthedocs.io/en/latest/bugs.html)
        - VerbatimInvalidDeduplication
        - FullInlinerNonExpressionSplitArgumentEvaluationOrder
        - MissingSideEffectsOnSelectorAccess.
It is used by:
        - 0.8.17 (INTEROP.sol#7)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (INTEROP.sol#59-64):
        - (success,None) = recipient.call{value: amount}() (INTEROP.sol#62)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

```
INFO:Detectors:
Redundant expression "this (INTEROP.sol#73)" inContext (INTEROP.sol#67-76)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
INTEROP.uniswapV2Pair (INTEROP.sol#247) should be immutable
INTEROP.uniswapV2Router (INTEROP.sol#246) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Slither:INTEROP.sol analyzed (10 contracts with 94 detectors), 21 result(s) found
```

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/0xeec1d73251adda2b94db11b39318a867ba35c0a880ca4a947ec08da32415be6c

2- Change Fee Receiver (passed):

https://testnet.bscscan.com/tx/0x95704fd256fa3be4debf9d092b96343b0c852e0f284fd78d56d0618981a85067

3- Update Fees (passed):

https://testnet.bscscan.com/tx/0xb3fee83d68af2ca7b276dccc57a04f036132a581882fb8f45038932a84ec8f37

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	1
Medium-Risk	1
Low-Risk	0
Gas Optimization / Suggestions	1

MANUAL TESTING

Centralization – Enabling Trades

Severity: High

Function: EnableTrading

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 enableTrading() external onlyOwner{
    require(!tradingEnabled, "Trading already enabled.");
    tradingEnabled = true;
    swapEnabled = true;

    emit TradingEnabled(tradingEnabled);
  }
```

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 – Missing Require Check.

Severity: Medium

Function: changeFeeReceiver

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 and this will lead to a potential honeypot in the contract.

```
function changeFeeReceiver(address _feeReceiver) external onlyOwner{
    require(_feeReceiver != address(0), "Fee receiver cannot be the zero address");
    feeReceiver = _feeReceiver;

    emit FeeReceiverChanged(feeReceiver);
  }
```

Suggestion:

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

MANUAL TESTING

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.

```
function _msgData() internal view virtual returns (bytes calldata) {
    this; // silence state mutability warning without generating bytecode - see https://github.com/ethereum/solidity/issues/2691
    return msg.data;
  }
```

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