auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

PORA AI

DATED : 7 March, 2024

MANUAL TESTING

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 openTrading() external onlyOwner {
        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 provide investors with more confidence in the eventual activation of trading capabilities, mitigating concerns of potential bad-faith actions by the original owner.

AUDIT SUMMARY

Project name – PORA AI

Date: 7 March, 2024

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

Audit Status: **Passed With High Risk**

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	0	1	1	1	3
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/0x5D653bEABbC69600f94a987e290966FE5B852fC8#code

Token Information

Token Name : PORA AI

Token Symbol: PORA

Decimals: 18

Token Supply: 100000000000

Network: EtherScan

Token Type: ERC-20

Token Address:
0xdD4cE03B97085e5023D3a5FBff6e4f2c4DffB7c3

Checksum:
A2032c616934aeb47e6039f76b20d231

Owner:
0x790ce9775965A80BA5a2f3DC2b0B7321b0465fFA
(at time of writing the audit)

Deployer:
0x790ce9775965A80BA5a2f3DC2b0B7321b0465fFA

TOKEN OVERVIEW

Fees:

Buy Fee: 0-0%

Sell Fee: 0-0%

Fees Privilege: Owner

Ownership: Owned

Minting: None

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

AUDIT METHODOLOGY

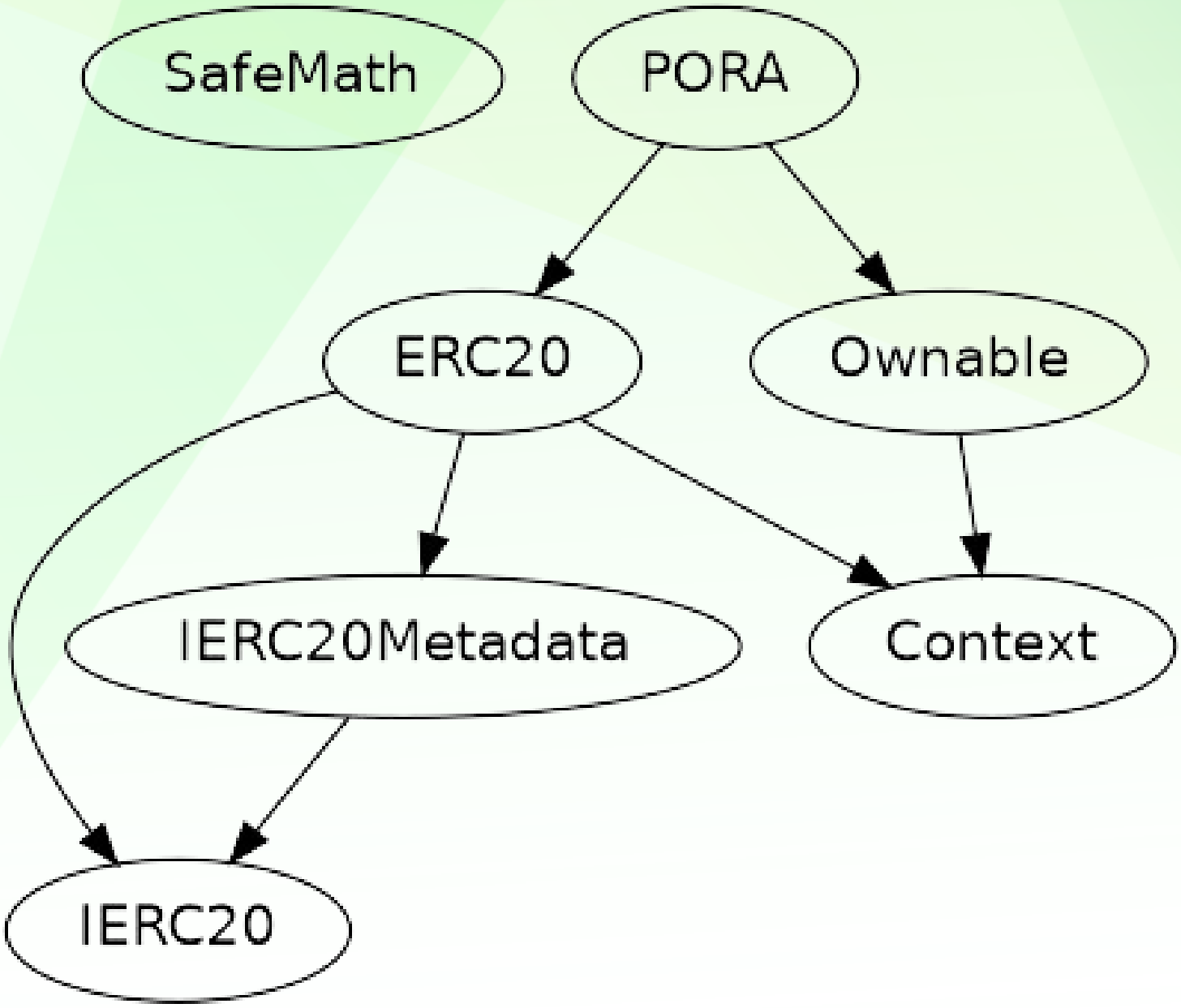
The auditing process will follow a routine as special considerations by Auditace:

- 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

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

INHERITANCE TREE



STATIC ANALYSIS

A static analysis of the code was performed using Slither.

No issues were found.

```
INFO:Detectors:
Contract locking ether found:
        Contract PORA (PORA.sol#766-822) has payable functions:
         - PORA.receive() (PORA.sol#779)
        But does not have a function to withdraw the ether
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#contracts-that-lock-ether
INFO:Detectors:
PORA.constructor()._totalSupply (PORA.sol#774) shadows:
        - ERC20._totalSupply (PORA.sol#354) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Context._msgData() (PORA.sol#344-346) is never used and should be removed
ERC20._burn(address,uint256) (PORA.sol#619-634) is never used and should be removed
SafeMath.add(uint256,uint256) (PORA.sol#102-104) is never used and should be removed
SafeMath.div(uint256,uint256) (PORA.sol#144-146) is never used and should be removed
SafeMath.div(uint256,uint256,string) (PORA.sol#199-208) is never used and should be removed
SafeMath.mod(uint256,uint256) (PORA.sol#159-161) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (PORA.sol#225-234) is never used and should be removed
SafeMath.mul(uint256,uint256) (PORA.sol#130-132) is never used and should be removed
SafeMath.sub(uint256,uint256) (PORA.sol#116-118) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (PORA.sol#176-185) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (PORA.sol#16-25) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (PORA.sol#67-75) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (PORA.sol#82-90) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (PORA.sol#47-60) is never used and should be removed
SafeMath.trySub(uint256,uint256) (PORA.sol#32-40) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version>=0.8.19 (PORA.sol#8) necessitates a version too recent to be trusted. Consider deploying with 0.8.18.
solc-0.8.19 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Parameter PORA.setPresaleAddress(address,bool)._address (PORA.sol#786) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Slither:PORA.sol analyzed (7 contracts with 93 detectors), 20 result(s) found
```

FUNCTIONAL TESTING

1- Approve (passed):
https://testnet.bscscan.com/tx/0xddc5861d82c53edc192957a96f2a9ff7da202462aeb22ecd73d13c153b99e138

2- Increase Allowance (passed):
https://testnet.bscscan.com/tx/0x17eb4d1d969ad5fdc07807bef711eba1d360b522faaefe5c003b85314c698a13

3- Decrease Allowance (passed):
https://testnet.bscscan.com/tx/0x2c2ddc2e16942de008e4eb87c5dbd8e362ec128e3ae20a3b966327c098404875

4- Open Trading (passed):
https://testnet.bscscan.com/tx/0xb9a885eadc2f43c89390f4b323fddd8ae4d3755f6ef33efc277bfd1fb5cdd8e7

5- Set Presale Address (passed):
https://testnet.bscscan.com/tx/0x08e659a17f4e7ad1d7b3429cceb0afdb04f44b0a1bdfa59957de10ad8409f887

6- Transfer (passed):
https://testnet.bscscan.com/tx/0xb4552b8cb6704dac9e61ca69089d0618462cd1447de1705899f39cf3cc85a1ad

POINTS TO NOTE

- The owner can transfer ownership.
- The owner can renounce ownership.
- The owner can set the presale address.
- The owner can open trading.

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	1
Gas Optimization / Suggestions	3

MANUAL TESTING

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 openTrading() external onlyOwner {
        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 provide investors with 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: setPresaleAddress

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 setPresaleAddress(
        address _address,
        bool state
    ) external onlyOwner {
        presaleAddress[_address] = state;
    }
```

Suggestion:

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

MANUAL TESTING

Centralization – Missing Events

Severity: Low

Function: 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 setPresaleAddress(
        address _address,
        bool state
    ) external onlyOwner {
        presaleAddress[_address] = state;
    }
```

Suggestion:
Emit an event for critical changes.

MANUAL TESTING

Optimization

Severity: Informational

Function: Remove Safe Math

Status: Open

Line: 10-235

Overview:

compiler version above 0.8.0 can control arithmetic overflow/underflow, it is recommended to remove the unwanted code to avoid high gas fees.

MANUAL TESTING

Optimization

Severity: Informational

Function: 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 >=0.8.19;
```

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.

MANUAL TESTING

Optimization

Severity: Optimization

Function: 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) {
        return msg.data;
    }
function _burn(address account, uint256 amount) internal virtual {
        require(account != address(0), "ERC20: burn from the zero address");

        _beforeTokenTransfer(account, address(0), amount);

        uint256 accountBalance = _balances[account];
        require(accountBalance >= amount, "ERC20: burn amount exceeds balance");
        unchecked {
            _balances[account] = accountBalance - amount;
        }
        _totalSupply -= amount;

        emit Transfer(account, address(0), amount);

        _afterTokenTransfer(account, address(0), amount);
    }
```

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