AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

AIEPH

DATED : 30 June 23'

AUDIT SUMMARY

Project name – AIEPH

Date: 30 June, 2023

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

Audit Status: Passed

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	0	0	0	0	0
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:
Contract has been tested on binance smart chain testnet which can be found in below link:
https://testnet.bscscan.com/token/0x6C3a03797378e AE216845CBc13a0b9F585f43F89

Token Information

Token Name : AIEPH EDKBLFEATCGEHGHWPJMUSK DOGE

Token Symbol: AIEPH

Decimals: 9

Token Supply: 100,000,000,000,000,000,000

Token Address:
0xF2a7C3852a150CF665221F806D08d91823C65fB6

Checksum:
ea1f0ac0f7d7a7351cce5acac5b771dc62f3df9c

Owner:
0x348e51C60e9e15C0f80b95e6a2619655b07489E4
(at time of writing the audit)

Deployer:
0x348e51C60e9e15C0f80b95e6a2619655b07489E4

TOKEN OVERVIEW

Fees:

Buy Fees: 0%

Sell Fees: 0%

Transfer Fees: 0%

Fees Privilege: No fees

Ownership: Renounced

Minting: none

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Privileges: - Initial distribution of the tokens

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

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.
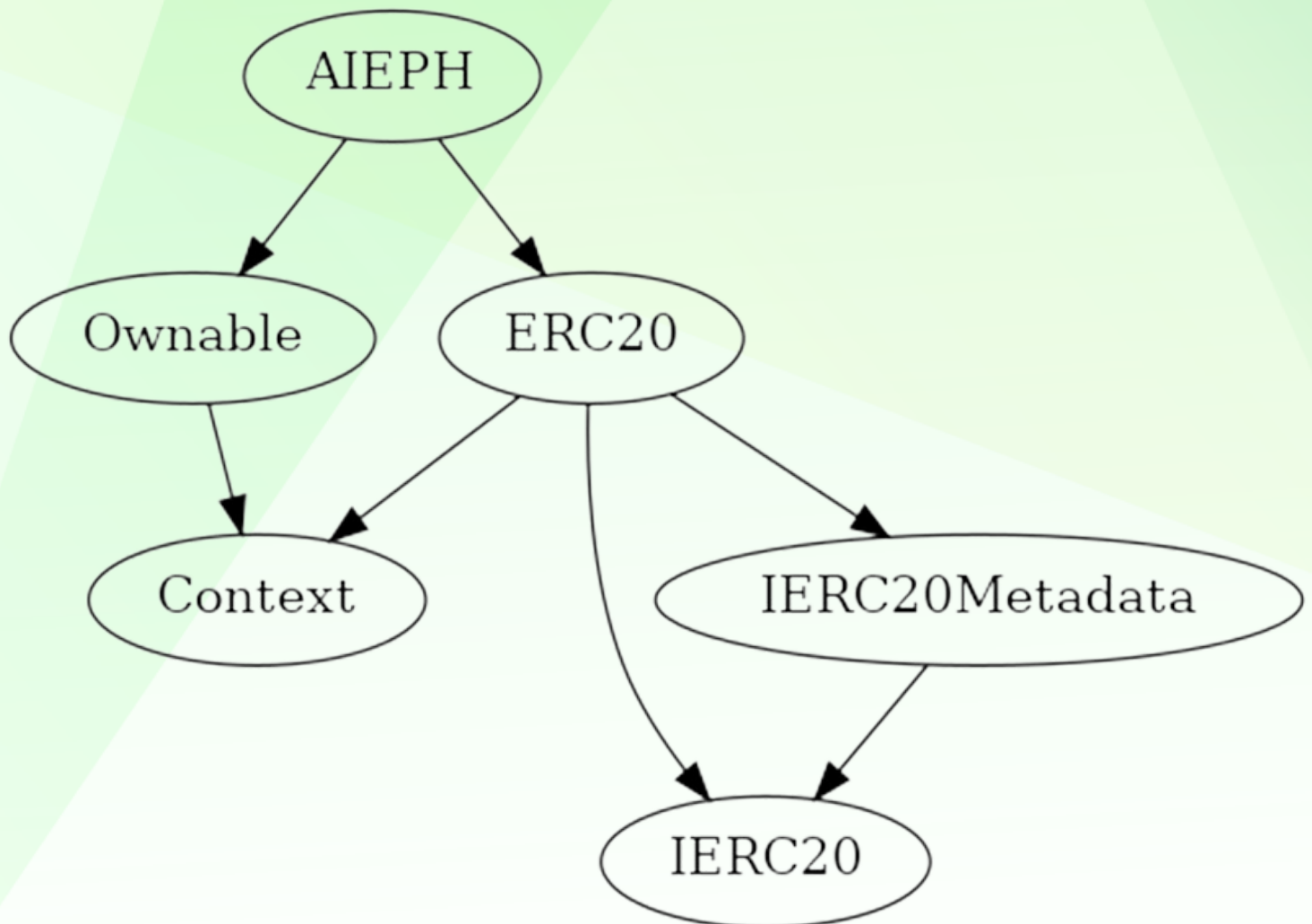
Findings

Severity	Found
◆ **Critical**	0
◆ **High-Risk**	0
◆ **Medium-Risk**	0
◆ **Low-Risk**	0
◆ **Gas Optimization / Suggestions**	0

INHERITANCE TREE

POINTS TO NOTE

- Owner is not able to set buy/sell/transfer tax
- Owner is not able to set max buy/sell/transfer/hold amount
- Owner is not able to blacklist an arbitrary wallet
- Owner is not able to disable trades
- Owner is not able to mint new tokens

CONTRACT ASSESMENT

Contract	Type	Bases			
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
Ownable	Implementation	Context		
└	<Constructor>	Public ❗	🔴	NO ❗
└	owner	Public ❗		NO ❗
└	renounceOwnership	Public ❗	🔴	onlyOwner
└	transferOwnership	Public ❗	🔴	onlyOwner
└	_transferOwnership	Internal 🔒	🔴	
IERC20	Interface			
└	totalSupply	External ❗		NO ❗
└	balanceOf	External ❗		NO ❗
└	transfer	External ❗	🔴	NO ❗
└	allowance	External ❗		NO ❗
└	approve	External ❗	🔴	NO ❗
└	transferFrom	External ❗	🔴	NO ❗
IERC20Metadata	Interface	IERC20		
└	name	External ❗		NO ❗
└	symbol	External ❗		NO ❗
└	decimals	External ❗		NO ❗
ERC20	Implementation	Context, IERC20, IERC20Metadata		
└	<Constructor>	Public ❗	🔴	NO ❗
└	name	Public ❗		NO ❗
└	symbol	Public ❗		NO ❗
└	decimals	Public ❗		NO ❗
└	totalSupply	Public ❗		NO ❗
└	balanceOf	Public ❗		NO ❗
└	transfer	Public ❗	🔴	NO ❗
└	allowance	Public ❗		NO ❗
└	approve	Public ❗	🔴	NO ❗
└	transferFrom	Public ❗	🔴	NO ❗
└	increaseAllowance	Public ❗	🔴	NO ❗
└	decreaseAllowance	Public ❗	🔴	NO ❗
└	_transfer	Internal 🔒	🔴	
└	_mint	Internal 🔒	🔴	
└	_burn	Internal 🔒	🔴	

CONTRACT ASSESMENT

└	_approve	Internal 🔒	🔴	
└	_beforeTokenTransfer	Internal 🔒	🔴	
└	_afterTokenTransfer	Internal 🔒	🔴	
AIEPH	Implementation	Ownable, ERC20		
└	<Constructor>	Public ❗	🔴	ERC20

Legend

Symbol	Meaning
🔴	Function can modify state
💲	Function is payable

STATIC ANALYSIS

```
AIEPH.constructor(uint256)._totalSupply (contracts/Token.sol#535) shadows:
        - ERC20._totalSupply (contracts/Token.sol#234) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

Different versions of Solidity are used:
        - Version used: ['^0.8.0', '^0.8.17']
        - ^0.8.0 (contracts/Token.sol#30)
        - ^0.8.0 (contracts/Token.sol#102)
        - ^0.8.0 (contracts/Token.sol#178)
        - ^0.8.0 (contracts/Token.sol#202)
        - ^0.8.0 (contracts/Token.sol#532)
        - ^0.8.17 (contracts/Token.sol#8)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Context._msgData() (contracts/Token.sol#25-27) is never used and should be removed
ERC20._burn(address,uint256) (contracts/Token.sol#461-476) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.17 (contracts/Token.sol#8) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
Pragma version^0.8.0 (contracts/Token.sol#30) allows old versions
Pragma version^0.8.0 (contracts/Token.sol#102) allows old versions
Pragma version^0.8.0 (contracts/Token.sol#178) allows old versions
Pragma version^0.8.0 (contracts/Token.sol#202) allows old versions
Pragma version^0.8.0 (contracts/Token.sol#532) allows old versions
solc-0.8.20 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

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- Adding liquidity (passed):

https://testnet.bscscan.com/tx/0x5dbb292ff818f7f1d7cdb09f5b22203082f838bb42127f69e192af03b3ccc88f

2- Buying (0% tax) (passed):

https://testnet.bscscan.com/tx/0x7ad14297657e81b214d0518128e710dfc0ed4b6dfbf8951a23c3742e04787ec6

3- Selling (0% tax) (passed):

https://testnet.bscscan.com/tx/0xabecdbfa61e76c94207b437c53766d03523ea517b85e95c5115f7b19ca66deb5

4- Transferring (0% tax) (passed):

https://testnet.bscscan.com/tx/0xd15de64a346e48adb008bfa3b9050769ad71ae79eb80b2247b49ecf14af0b36c

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