



# Smart Contract Audit

FOR

## Astra Network

DATED : 20 June 23'



# AUDIT SUMMARY

---

**Project name** – Astra Network

**Date:** 20 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 done on BSC Test network, each test has its transaction has attached to it.

### 3- Slither : Static Analysis

**Testnet Link:** all tests were done using this contract, tests are done on BSC Testnet

<https://testnet.bscscan.com/address/0x6ee6a25d595c4bF7161578C4f609103007e56818>

---



# Token Information

---

**Token Name :** Astra Network

**Token Symbol:** Astra

**Decimals:** 18

**Token Supply:**1,000,000,000

**Token Address:**

0x409188D1Ca9D0ef9836688654E301c06544fcCCe

**Checksum:**

6892b4d807c7678a5031330134301a6e540d2b02

**Owner:**

0xFE9911Adb8F7a39779e47dCfF7064E2A0F4e7C  
4



# TOKEN OVERVIEW

---

## **Fees:**

Buy Fees: 0%

Sell Fees: 0%

Transfer Fees: 0%

---

**Fees Privilege: no**

---

**Ownership : no**

---

**Minting: No mint function**

---

**Max Tx Amount/ Max Wallet Amount: none**

---

**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 is exercised 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

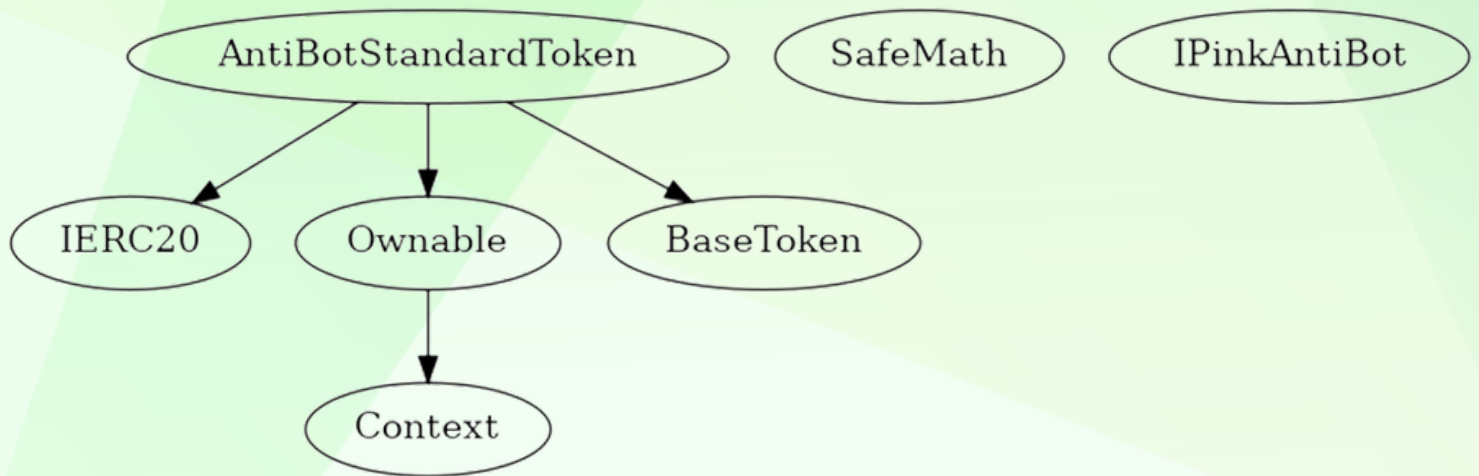
### ◆ Gas Optimization / Suggestions

0



# INHERITANCE TREE

---





# POINTS TO NOTE

---

- Owner is not able to set buy/sell/transfer taxes (0% all)
  - Owner is not able to set transfer taxes
  - 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			
:-----: :-----: :-----: :-----: :-----:					
L	**Function Name**	**Visibility**	**Mutability**	**Modifiers**	
**IERC20**   Interface					
L	totalSupply	External	!	NO	!
L	balanceOf	External	!	NO	!
L	transfer	External	!	!	NO
L	allowance	External	!	NO	!
L	approve	External	!	!	NO
L	transferFrom	External	!	!	NO
**Context**   Implementation					
L	_msgSender	Internal	!		
L	_msgData	Internal	!		
**Ownable**   Implementation   Context					
L	<Constructor>	Public	!	!	NO
L	owner	Public	!	NO	!
L	renounceOwnership	Public	!	!	onlyOwner
L	transferOwnership	Public	!	!	onlyOwner
L	_setOwner	Private	!	!	
**SafeMath**   Library					
L	tryAdd	Internal	!		
L	trySub	Internal	!		
L	tryMul	Internal	!		
L	tryDiv	Internal	!		
L	tryMod	Internal	!		
L	add	Internal	!		
L	sub	Internal	!		
L	mul	Internal	!		
L	div	Internal	!		
L	mod	Internal	!		
L	sub	Internal	!		
L	div	Internal	!		
L	mod	Internal	!		
**IPinkAntiBot**   Interface					
L	setTokenOwner	External	!	!	NO
L	onPreTransferCheck	External	!	!	NO



# CONTRACT ASSESMENT

```
||||| |
| **AntiBotStandardToken** | Implementation | IERC20, Ownable, BaseToken |||
|  | <Constructor> | Public ! |  | NO! |
|  | setEnableAntiBot | External ! |  | onlyOwner |
|  | 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  |  | |
|  | _approve | Internal  |  | |
|  | _setupDecimals | Internal  |  | |
|  | _beforeTokenTransfer | Internal  |  | |
```

## ### Legend

Symbol	Meaning
:-----: -----	
	Function can modify state
	Function is payable



# STATIC ANALYSIS

```
Reentrancy in AntiBotStandardToken.transferFrom(address,address,uint256) (contracts/Token.sol#561-569):
  External calls:
    - _transfer(sender,recipient,amount) (contracts/Token.sol#562)
    - pinkAntiBot.onPreTransferCheck(sender,recipient,amount) (contracts/Token.sol#630)
  State variables written after the call(s):
    - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20: transfer amount exceeds allowance)) (contracts/Token.sol#563-567)
    - _allowances[owner][spender] = amount (contracts/Token.sol#698)
  Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in AntiBotStandardToken._transfer(address,address,uint256) (contracts/Token.sol#625-638):
  External calls:
    - pinkAntiBot.onPreTransferCheck(sender,recipient,amount) (contracts/Token.sol#630)
  Event emitted after the call(s):
    - Transfer(sender,recipient,amount) (contracts/Token.sol#637)
  Reentrancy in AntiBotStandardToken.transferFrom(address,address,uint256) (contracts/Token.sol#561-569):
  External calls:
    - _transfer(sender,recipient,amount) (contracts/Token.sol#562)
    - pinkAntiBot.onPreTransferCheck(sender,recipient,amount) (contracts/Token.sol#630)
  Event emitted after the call(s):
    - Approval(owner,spender,amount) (contracts/Token.sol#699)
    - _approve(sender,_msgSender(),_allowances[sender][_msgSender()].sub(amount,ERC20: transfer amount exceeds allowance)) (contracts/Token.sol#563-567)
  Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

AntiBotStandardToken._burn(address,uint256) (contracts/Token.sol#671-679) is never used and should be removed
AntiBotStandardToken.setupDecimals(uint8) (contracts/Token.sol#709-711) is never used and should be removed
Context._msgData() (contracts/Token.sol#104-106) is never used and should be removed
SafeMath.div(uint256,uint256) (contracts/Token.sol#314-316) is never used and should be removed
SafeMath.div(uint256,uint256,string) (contracts/Token.sol#366-371) is never used and should be removed
SafeMath.mod(uint256,uint256) (contracts/Token.sol#330-332) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (contracts/Token.sol#388-393) is never used and should be removed
SafeMath.mul(uint256,uint256) (contracts/Token.sol#300-302) is never used and should be removed
SafeMath.sub(uint256,uint256) (contracts/Token.sol#286-288) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (contracts/Token.sol#201-207) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (contracts/Token.sol#243-248) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (contracts/Token.sol#255-260) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (contracts/Token.sol#226-236) is never used and should be removed
SafeMath.trySub(uint256,uint256) (contracts/Token.sol#214-219) 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#427) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16
solc-0.8.20 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Parameter AntiBotStandardToken.setEnableAntiBot(bool)._enable (contracts/Token.sol#466) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Variable AntiBotStandardToken.totalSupply (contracts/Token.sol#446) is too similar to AntiBotStandardToken.constructor(string,string,uint8,uint256,address).totalSupply_ (contracts/Token.sol#451)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

AntiBotStandardToken.name (contracts/Token.sol#443) should be immutable
AntiBotStandardToken.symbol (contracts/Token.sol#444) should be immutable
AntiBotStandardToken.pinkAntiBot (contracts/Token.sol#448) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
```

**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/0x4a3754e428ef71ad4d1b5aacf4d35bc4774b838a9f70d39a93e80bfb3177ea31>

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

<https://testnet.bscscan.com/tx/0x51084220947cf6c65278bb64a14ec7a94ec57bc41ceaf113970bfdea3380a340>

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

<https://testnet.bscscan.com/tx/0x2ec8821e657c6061807a94e377644da85c1c7fbaf21f6f04bcbfc4b19816f231>

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

<https://testnet.bscscan.com/tx/0x260f9c50c5a3f70a4d6685a1d981bb0b74669138335032e4d1c8c40873258b87>

---



# 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 specialize 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://t.me/Audit_Ace)**



**[https://twitter.com/auditace\\_](https://twitter.com/auditace_)**



**<https://github.com/Audit-Ace>**

---