AuditAce

FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

SpaceDF

DATED : 11 June 23'

AUDIT SUMMARY

Project name – SpaceDF

Date: 11 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/0x54AAb1ff7000B12
72C73cF115F2be70a6602C201

Token Information

Token Name : SpaceDoge Finance

Token Symbol: SpaceDF

Decimals: 18

Token Supply: 1,000,000

Token Address:
0xc833364eA318E897441C3eD123D525564378206D

Checksum:
6892b4d807c7678a5031330134301a6e540d2b02

Owner:
0xFeD9911Adb8F7a39779e47dCfF7064E2A0F4e7C4

Deployer:
0xFeD9911Adb8F7a39779e47dCfF7064E2A0F4e7C4

TOKEN OVERVIEW

Fees:

Buy Fees: 0%

Sell Fees: 0%

Transfer Fees: 0%

Fees Privilege: None

Ownership: Owned

Minting: None

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Privileges: - initial distribution of the token

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

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

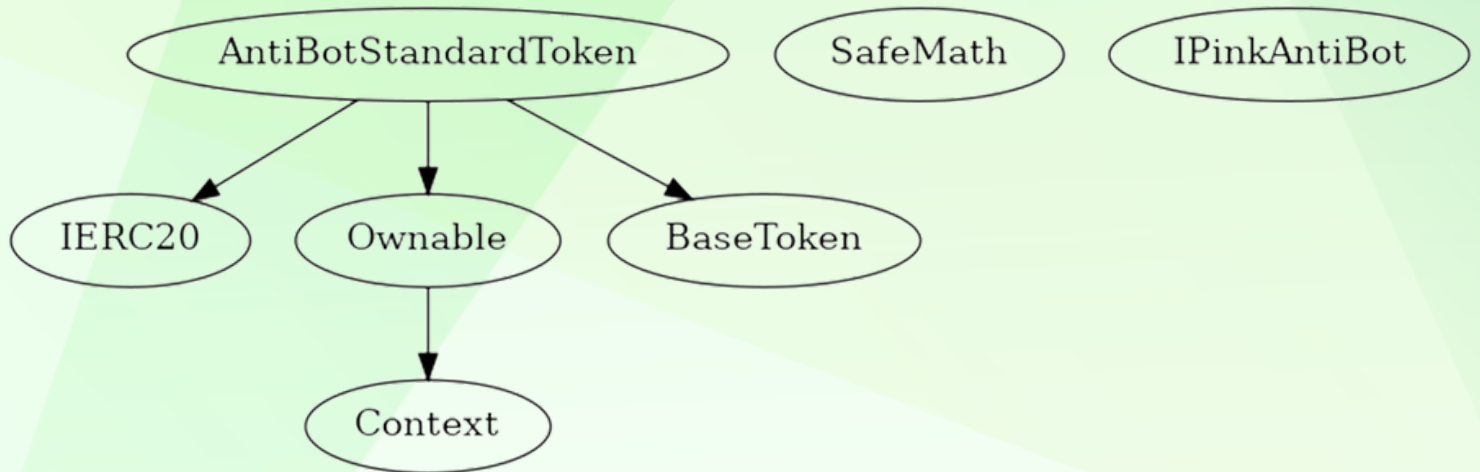| Severity	Found
◆ Critical	0
◆ High-Risk	0
◆ Medium-Risk	0
◆ Low-Risk	0
◆ Gas Optimization / Suggestions	0

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		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
IERC20	Interface				
└	totalSupply	External ❗	|NO ❗	
└	balanceOf	External ❗	|NO ❗	
└	transfer	External ❗	🔴 |NO ❗	
└	allowance	External ❗	|NO ❗	
└	approve	External ❗	🔴 |NO ❗	
└	transferFrom	External ❗	🔴 |NO ❗	
Context	Implementation				
└	_msgSender	Internal 🔒			
└	_msgData	Internal 🔒			
Ownable	Implementation	Context			
└	<Constructor>	Public ❗	🔴 |NO ❗	
└	owner	Public ❗	|NO ❗	
└	renounceOwnership	Public ❗	🔴	onlyOwner	
└	transferOwnership	Public ❗	🔴	onlyOwner	
└	_setOwner	Private 🔐	🔴		
SafeMath	Library				
└	tryAdd	Internal 🔒			
└	trySub	Internal 🔒			
└	tryMul	Internal 🔒			
└	tryDiv	Internal 🔒			
└	tryMod	Internal 🔒			
└	add	Internal 🔒			
└	sub	Internal 🔒			
└	mul	Internal 🔒			
└	div	Internal 🔒			
└	mod	Internal 🔒			
└	sub	Internal 🔒			
└	div	Internal 🔒			
└	mod	Internal 🔒			
IPinkAntiBot	Interface				
└	setTokenOwner	External ❗	🔴 |NO ❗	
└	onPreTransferCheck	External ❗	🔴 |NO ❗	
BaseToken	Implementation				

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).tot
alSupply_ (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/0x0bba6347cab108a77537f86df0888a
e06c957a97f5df2209f3b605767dd6692e

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

https://testnet.bscscan.com/tx/0x3c78ad9eaf9dad692c83072189e306
e10624a69f69099acd27a1b04a3986e201

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

https://testnet.bscscan.com/tx/0xc459d253213e0c93896d0020b1b1faa
ffd26eda21f18a98947b80a22bb9d9692

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

https://testnet.bscscan.com/tx/0x625eaccc3bc9cbf6613b376ef4d278b
fc87b8df3beeee364cfd9aa8dd328e5c3

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