auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

Arcstar

DATED : 26 MAY 23'

CRITICAL RISK

Centralization – Excessive fees

Severity: Critical

function: setSniperFee

Status: Not Resolved

Overview:

Owner is able to set a certain amount of tax for an arbitrary wallet (buy and sell and transfers), this tax can be within range of 0-99%. this is a critical centralization risk and can be used to disable trades for specifiec addresses.

```
function setSniperFee(
    address[] memory account,
    uint8 _sellFee,
    uint8 _buyFee
) public onlyOwner {
    for (uint256 i = 0; i < account.length; i++) {
        if (_sellFee > 0) {
            sellSniperFee[account[i]] = _sellFee;
        }
        if (_buyFee > 0) {
            buySniperFee[account[i]] = _buyFee;
        }
    }
}
```

Suggestion

To mitigate this centralization issue there are several ways:

- delete this method

- renounce ownership of the contract

- implement an automated method to blacklist sniper bots in 0-5 blocks after enablling trades for public.

CRITICAL RISK

Centralization – Claiming tokens after presale

Severity: Critical

function: setSniperFee & _transfer

Status: Not Resolved

Overview:

setting a high buy tax for presale address at time of claiming tokens, can lead to lose of tokens of contributors. This taxed tokens will be sent to marketing wallet.

```
if (
    sellSniperFee[sender] > 0 &&
    (recipient == pairAddr || sender != pairAddr)
) {
    tax = baseUnit * uint256(sellSniperFee[sender]);
} else if (buySniperFee[recipient] > 0 && sender == pairAddr) {
    tax = baseUnit * uint256(buySniperFee[recipient]);
} else if (recipient == pairAddr) {
    tax = baseUnit * uint256(sellFee);
}
```

Suggestion

to mitigate this issue there are several options:

- delete sellSniperFee function

- renounce ownership of the contract

- implement an automated method to blacklist sniper bots in 0-5 blocks after enablling trades for public.

- ensure that “sender” is not presale address

AUDIT SUMMARY

Project name – Arcstar

Date: 26 May, 2023

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

Audit Status: **Passed With Critical Risk**

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	2	0	0	0	0
Acknowledged	0	0	0	0	0
Resolved	0	0	0	0	0

USED TOOLS

Tools:

1.Manual Review: The code has undergone a line-by-line review by the **Ace** team.

2.ETH Test Network: All tests were conducted on the ETH 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:
https://testnet.bscscan.com/token/0xbc93ce86efe03f29efd83b93de9410eb6c66790b

Token Information

Name : Arcstar

Symbol : ARCSTAR

Decimals: 18

Network: Binance smart chain

Token Type:BEP20

Token Address:
0x5331Ca78BF716df553048C1d6430855540f68Cef

Owner:
0xcBd5De8b6A8e7f8a3652e3d5Ce41400c7c892b4d
(at time of writing the audit)

Deployer:0xcBd5De8b6A8e7f8a3652e3d5Ce41400c7c892b4d

Token Information

Fees:

Buy Fees: 0%

Sell Fees: 0-5%

Transfer Fees: 0%

Fees Privilige: No fees

Ownership :

0xcBd5De8b6A8e7f8a3652e3d5Ce41400c7c892b4d

Minting: None

Max Tx Amount/ Max Wallet Amount: No

Blacklist: No

Other Priviliges:- Fees modification

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	2
High-Risk	0
Medium-Risk	0
Low-Risk	0
Gas Optimization / Suggestions	0

INHERITANCE TREE

POINTS TO NOTE

- Owner is able to set 0-5% tax for sells
- Owner is not able to set max buy/sell/transfer/hold amount
- Owner is able to blacklist an arbitrary wlalet
- Owner is not able able to limit buys/transfers/sells by a max amount as limit
- Owner is not able to mint new tokens
- Owner must enabel trades manually for holders

CONTRACT ASSESMENT

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
Arcstar	Implementation	ERC20, Ownable		
└	<Constructor>	Public ❗	🛑	ERC20
└	<Receive Ether>	External ❗	💵	NO ❗
└	_transfer	Internal 🔒	🛑	
└	setMarketingWallet	External ❗	🛑	onlyOwner
└	setPairAddr	External ❗	🛑	onlyOwner
└	setExcluded	Public ❗	🛑	onlyOwner
└	removeExcluded	Public ❗	🛑	onlyOwner
└	setSniperFee	Public ❗	🛑	onlyOwner
└	removeSniperFee	Public ❗	🛑	onlyOwner
└	setSellFee	Public ❗	🛑	onlyOwner
└	isExcluded	Public ❗		NO ❗
└	isSniper	Public ❗		NO ❗

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

STATIC ANALYSIS

```
Arcstar.setSellFee(uint8) (contracts/Token.sol#1033-1036) should emit an event for:
        - sellFee = _sellFee (contracts/Token.sol#1035)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

Arcstar.constructor(address,address)._pairAddr (contracts/Token.sol#938-941) lacks a zero-check on :
                - pairAddr = _pairAddr (contracts/Token.sol#943)
Arcstar.constructor(address,address)._marketingWallet (contracts/Token.sol#932) lacks a zero-check on :
                - marketingWallet = _marketingWallet (contracts/Token.sol#944)
Arcstar.setMarketingWallet(address)._address (contracts/Token.sol#987) lacks a zero-check on :
                - marketingWallet = _address (contracts/Token.sol#988)
Arcstar.setPairAddr(address)._address (contracts/Token.sol#991) lacks a zero-check on :
                - pairAddr = _address (contracts/Token.sol#992)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Different versions of Solidity are used:
        - Version used: ['>=0.5.0', '>=0.6.2', '^0.8.17']
        - >=0.5.0 (contracts/Token.sol#359)
        - >=0.6.2 (contracts/Token.sol#157)
        - ^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#117-119) is never used and should be removed
ERC20._burn(address,uint256) (contracts/Token.sol#701-717) 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>=0.6.2 (contracts/Token.sol#157) allows old versions
Pragma version>=0.5.0 (contracts/Token.sol#359) allows old versions
solc-0.8.19 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Function IUniswapV2Router01.WETH() (contracts/Token.sol#162) is not in mixedCase
Parameter Arcstar.setMarketingWallet(address)._address (contracts/Token.sol#987) is not in mixedCase
Parameter Arcstar.setPairAddr(address)._address (contracts/Token.sol#991) is not in mixedCase
Parameter Arcstar.setSniperFee(address[],uint8,uint8)._sellFee (contracts/Token.sol#1009) is not in mixedCase
Parameter Arcstar.setSniperFee(address[],uint8,uint8)._buyFee (contracts/Token.sol#1010) is not in mixedCase
Parameter Arcstar.setSellFee(uint8)._sellFee (contracts/Token.sol#1033) is not in mixedCase
Constant Arcstar.feeLimit (contracts/Token.sol#916) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Arcstar.denominator (contracts/Token.sol#922) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/Token.sol#167) is too simila
r to IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/Token.sol#168)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar

Arcstar.initialSupply (contracts/Token.sol#914) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

Arcstar.router (contracts/Token.sol#925) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
```

Static Analysis

an static analysis of the code were performed using slither. No issues were found

FUNCTIONAL TESTING

1- Adding liquidity (passed):
https://testnet.bscscan.com/tx/0x85b94ecc70530a8d8f18d2563aa82dc992e17785aeb2f1bf9d43762d48cff9ce

2- Buying (0% tax) (passed):
https://testnet.bscscan.com/tx/0x5891fb8f8dccfbd438eb173ebd80222f9d77d1a307a00504fc2fa94ad156479e

3- Selling (0% tax) (passed):
https://testnet.bscscan.com/tx/0xec10fedf77180175267d92c395a57200e390e654defa6b25be919260050136b5

4- Transferring (0% tax) (passed):
https://testnet.bscscan.com/tx/0x586e6e0d89d1c8af996b36bd67d1ad8f8fedd3f5a95cfe298c7719312d7e1f2a

2- Buying (0% tax) (passed):
https://testnet.bscscan.com/tx/0xbcbb9be87c3ede5af8751c346f386c1a441a561243d670fab65a4f60b91d035f

3- Selling (5% tax) (passed):
https://testnet.bscscan.com/tx/0xca74d2f095119cdcd8547627f85a8c5c09324ce93227f2b21206e93face9f858

4- Transferring (0% tax) (passed):
https://testnet.bscscan.com/tx/0xc49f68b53bd02dafe2644b2bfa169ce521338c8adccd80c10c80f2ac16fefee1

FUNCTIONAL TESTING

Centralization – Excessive fees

Severity: Critical

function: setSniperFee

Status: Not Resolved

Overview:

Owner is able to set a certain amount of tax for an arbitrary wallet (buy and sell and transfers), this tax can be within range of 0-99%. this is a critical centralization risk and can be used to disable trades for specifiec addresses.

```
function setSniperFee(
    address[] memory account,
    uint8 _sellFee,
    uint8 _buyFee
) public onlyOwner {
    for (uint256 i = 0; i < account.length; i++) {
        if (_sellFee > 0) {
            sellSniperFee[account[i]] = _sellFee;
        }
        if (_buyFee > 0) {
            buySniperFee[account[i]] = _buyFee;
        }
    }
}
```

Suggestion

To mitigate this centralization issue there are several ways:

- delete this method

- renounce ownership of the contract

- implement an automated method to blacklist sniper bots in 0-5 blocks after enablling trades for public.

FUNCTIONAL TESTING

Centralization – Claiming tokens after presale

Severity: Critical

function: setSniperFee & _transfer

Status: Not Resolved

Overview:

setting a high buy tax for presale address at time of claiming tokens, can lead to lose of tokens of contributors. This taxed tokens will be sent to marketing wallet.

```
if (
    sellSniperFee[sender] > 0 &&
    (recipient == pairAddr || sender != pairAddr)
) {
    tax = baseUnit * uint256(sellSniperFee[sender]);
} else if (buySniperFee[recipient] > 0 && sender == pairAddr) {
    tax = baseUnit * uint256(buySniperFee[recipient]);
} else if (recipient == pairAddr) {
    tax = baseUnit * uint256(sellFee);
}
```

Suggestion

to mitigate this issue there are several options:

- delete sellSniperFee function

- renounce ownership of the contract

- implement an automated method to blacklist sniper bots in 0-5 blocks after enablling trades for public.

- ensure that “sender” is not presale address

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