auditace.tech

AuditAce
FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

FCUK BULL

DATED : 18 MAY 23'

AUDIT SUMMARY

Project name – BULL

Date: 18 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	1	1	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:
The tests were performed using the contract deployed on the ETHTestnet, which can be found at the following address:
https://testnet.bscscan.com/token/0xBCE6f2Ff37f7b69D8628cf604CfA325Bd7053c0F?a=0xff0ad89f982bd10e07a9fb108a655eaad4d9b2fa

Token Information

Name : FCUK BULL

Symbol : BULL

Decimals: 18

Network: ETH

Token Type:ERC20

Token Address:
0x600D4C42676bbEaB2B201f93300C6af9B1F6D163

Owner:
0x4e1BEEfc268bBeEcD74C4bB8ee834fA398bb3ef9
(at time of writing the audit)

Deployer:0x8587A1e4EA67506D44f003628770d88D95cE1da6

Token Information

Fees:

Buy Fees: 0%

Sell Fees: 0%

Transfer Fees: 0%

Fees Privilige: None

Ownership :

0x4e1BEEfc268bBeEcD74C4bB8ee834fA398bb3ef9

Minting: None

Max Tx Amount/ Max Wallet Amount: No

Blacklist: Yes

Other Priviliges:- blacklisting wallets

- enabling/disabling trades

- setting max and minimum wallet amounts

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

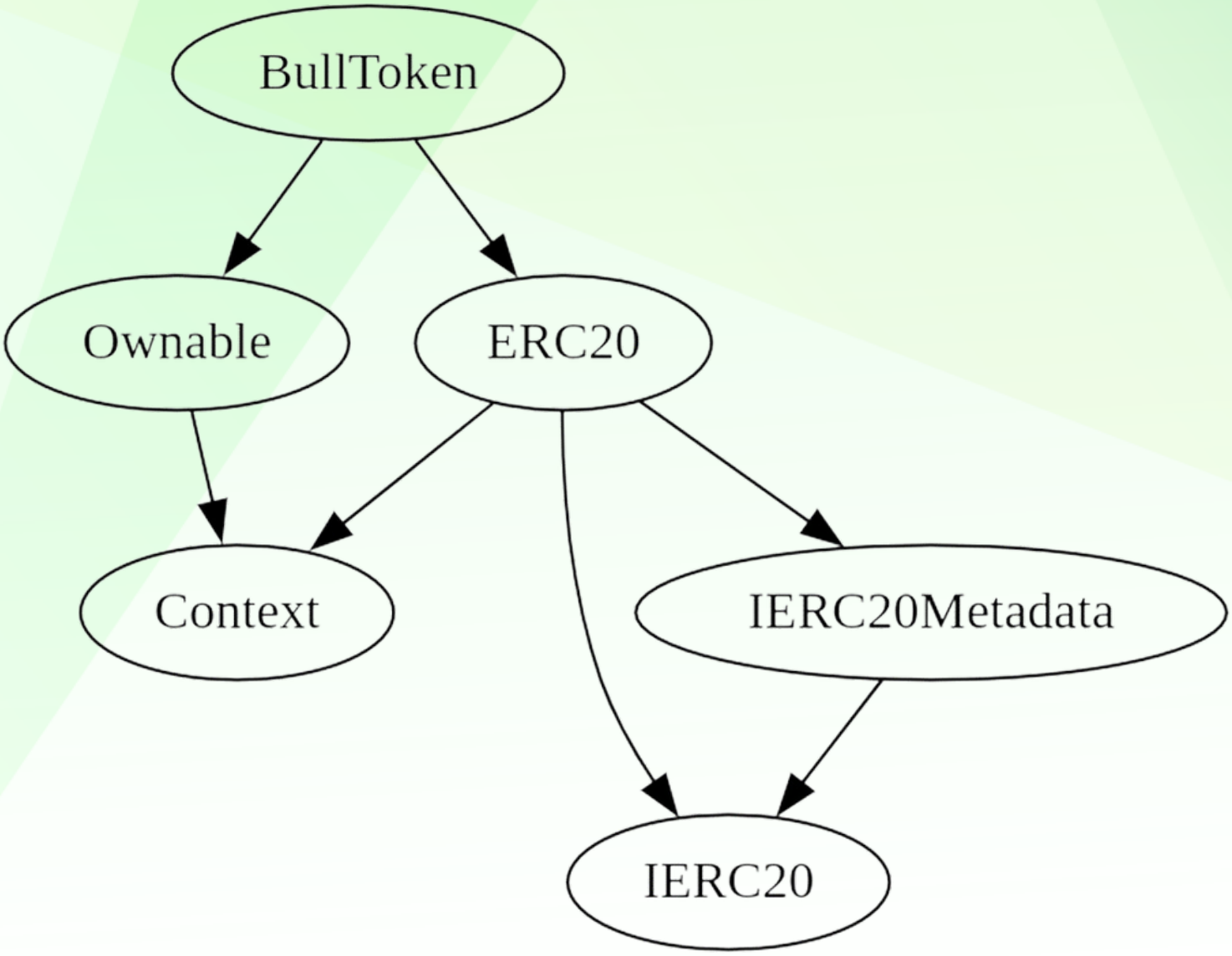
CLASSIFICATION OF RISK

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	1
Low-Risk	1
Gas Optimization / Suggestions	0

INHERITANCE TREE



POINTS TO NOTE

- The contract has an owner, and only the owner can perform certain actions such as **blacklisting** addresses, setting limitations, and transferring ownership.
- The contract owner can blacklist or unblacklist an address, preventing them from sending or receiving tokens.(buy, sell, transfers would be disabled)
- The contract owner can set trading rules, including enabling/disabling trading limits, setting the UniswapV2Pair address (if set to address zero, can disable trades), and setting maximum and minimum holding amounts.
- The contract owner cannot set buy/sell/transfer fees.
- The contract owner cannot mint new tokens after the initial supply is created.
- The contract owner can disable trades
- The contract has a burn function that allows any address to burn their tokens, reducing the total supply.
- The contract checks for blacklisted addresses before allowing token transfers.

POINTS TO NOTE

- The contract checks for trading limits before allowing token transfers from the UniswapV2Pair address.
- It's a fork from Pepe

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 🔒	🛑	

CONTRACT ASSESMENT

└	_burn	Internal 🔒	🛑	
└	_approve	Internal 🔒	🛑	
└	_beforeTokenTransfer	Internal 🔒	🛑	
└	_afterTokenTransfer	Internal 🔒	🛑	
BullToken	Implementation	Ownable, ERC20		
└	<Constructor>	Public ❗	🛑	ERC20
└	blacklist	External ❗	🛑	onlyOwner
└	setRule	External ❗	🛑	onlyOwner
└	_beforeTokenTransfer	Internal 🔒	🛑	
└	burn	External ❗	🛑	NO ❗

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

STATIC ANALYSIS

```
BullToken.constructor(uint256)._totalSupply (contracts/Token.sol#637) shadows:
        - ERC20._totalSupply (contracts/Token.sol#278) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

BullToken.setRule(bool,address,uint256,uint256) (contracts/Token.sol#648-658) should emit an event for:
        - maxHoldingAmount = _maxHoldingAmount (contracts/Token.sol#656)
        - minHoldingAmount = _minHoldingAmount (contracts/Token.sol#657)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

BullToken.setRule(bool,address,uint256,uint256)._uniswapV2Pair (contracts/Token.sol#650) lacks a zero-check on :
                - uniswapV2Pair = _uniswapV2Pair (contracts/Token.sol#655)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Different versions of Solidity are used:
        - Version used: ['^0.8.0', '^0.8.17']
        - ^0.8.0 (contracts/Token.sol#42)
        - ^0.8.0 (contracts/Token.sol#124)
        - ^0.8.0 (contracts/Token.sol#218)
        - ^0.8.0 (contracts/Token.sol#246)
        - ^0.8.0 (contracts/Token.sol#628)
        - ^0.8.17 (contracts/Token.sol#16)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Context._msgData() (contracts/Token.sol#33-35) is never used and should be removed
ERC20._beforeTokenTransfer(address,address,uint256) (contracts/Token.sol#599-603) 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#16) 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#42) allows old versions
Pragma version^0.8.0 (contracts/Token.sol#124) allows old versions
Pragma version^0.8.0 (contracts/Token.sol#218) allows old versions
Pragma version^0.8.0 (contracts/Token.sol#246) allows old versions
Pragma version^0.8.0 (contracts/Token.sol#628) 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

Parameter BullToken.blacklist(address,bool)._address (contracts/Token.sol#642) is not in mixedCase
Parameter BullToken.blacklist(address,bool)._isBlacklisting (contracts/Token.sol#643) is not in mixedCase
Parameter BullToken.setRule(bool,address,uint256,uint256)._limited (contracts/Token.sol#649) is not in mixedCase
Parameter BullToken.setRule(bool,address,uint256,uint256)._uniswapV2Pair (contracts/Token.sol#650) is not in mixedCase
Parameter BullToken.setRule(bool,address,uint256,uint256)._maxHoldingAmount (contracts/Token.sol#651) is not in mixedCase
Parameter BullToken.setRule(bool,address,uint256,uint256)._minHoldingAmount (contracts/Token.sol#652) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

Static Analysis

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

FUNCTIONAL TESTING

Router (PCS V2):
0xD99D1c33F9fC3444f8101754aBC46c52416550D1

1- Adding liquidity (passed):
https://testnet.bscscan.com/tx/0xcd182dc6f75222bd03744215587eba59c9d0a0a3d123270d26636f9f3c15b539

2- Buying when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0x6c99497930c0e37cc241ebeb127204fc682b80c6fe1e8b72b5e0e82859282ef6

3- Selling when excluded (0% tax) (passed):
https://testnet.bscscan.com/tx/0xb072be27bf6ebcfc1f5656569de29e66202d67ef25654ec5878a41d1a1aeeb71

4- Transferring when excluded from fees (0% tax) (passed):
https://testnet.bscscan.com/tx/0xd292bb172c8011fb435739382082992665862bc7045dfe503124ecbefc90bbc5

5- Buying from a regular wallet (0% tax) (passed):
https://testnet.bscscan.com/tx/0xf59522e98e1d10e130ff006360daa5e1f14737c2c0de6fd3d40ad89d1bd75a0d

6- Selling from a regular wallet (0% tax) (passed):
https://testnet.bscscan.com/tx/0xf7db27e4828477fc6eb5f443266dc0a2379b2a75749f98f73900fd3b6384bb6d

7- Transferring from a regular wallet (0% tax) (passed):
https://testnet.bscscan.com/tx/0xc431dcfbd0c3c0d4cfcc57162cbd3c0bb5feedc3319c05ccd5c1cca719b6e417

FUNCTIONAL TESTING

8- Internal swap (marketing bnb) (passed):
https://testnet.bscscan.com/tx/0xaf0baf835c9add08bd86fa8c8950e0bcdc67c7418727462da7a2d508b3af3f10

FINDINGS TABLE

Category	Subject	Severity	Suggestion/Summary
Logical	Incorrect handling of limited transfers	Medium	Enforce the limits for all types of transfers except sells
Logical	Incomplete trading start condition	Critical	Add a boolean variable tradingStarted and create a mapping to exclude specific addresses when trading is not enabled
Centralization	Limiting trading	Critical	Ensure max holding is not more than a % of total supply, and min holding amount doesn't disable buys. Make uniswapV2Pair immutable
Centralization	Burning tokens	Low	Implement a decentralized governance mechanism to decide on burning tokens

FUNCTIONAL TESTING

Category: Logical
Subject: Incorrect handling of limited transfers
Severity: Medium
Overview:
The contract has a limited transfer feature that restricts the amount of tokens a user can hold. However, the implementation only checks the limits when the sender is the UniswapV2Pair, which means that the limits are not enforced for other types of transfers. This means holders are still able to accumulate more tokens than what is allowed which could result in price manipulations

Logical issue code:

```
if (limited && from == uniswapV2Pair) {
   require( super.balanceOf(to) + amount <= maxHoldingAmount && super.balanceOf(to) + amount >=
minHoldingAmount, "Forbid" );
}
```

Suggestion:
To enforce the limits for all types of transfers except sells, the condition should be updated to check for the limited flag and if to is not equal to unispwav2Pair, and not specifically for the UniswapV2Pair as the sender. The updated code should look like this:

```
if (limited && to != uniswapV2Pair) {
   require( super.balanceOf(to) + amount <= maxHoldingAmount && super.balanceOf(to) + amount >=
minHoldingAmount, "Forbid" );
}
```

FUNCTIONAL TESTING

Category: Logical
Subject: Incomplete trading start condition can disable claims at end of presale
Severity: Critical
Overview:
The contract has a condition to check if trading has started by comparing the UniswapV2Pair address with the **zero address**. However, this condition is not sufficient to ensure that trading has actually started and also doesn't allow transfers from or to wallets except than owner's wallet (which can disable claims at end of presale).

Logical issue code:

```
if (uniswapV2Pair == address(0)) {
   require(from == owner() || to == owner(), "trading is not started");
   return;
}
```

Suggestion:
To ensure that trading has actually started and also to allow finalize of the presale, you can add a boolean variable `tradingStarted` in the contract and set it to true when the trading is initialized. Also you can create a mapping to exclude specifiec addresses from this limitations when trading is not still enabled. The updated condition should look like this:

```
if (!tradingStarted) {
   require(isWhitelisted[from] || isWhitelisted[to], "trading is not started");
   return;
}
```

And when initializing the trading, set the `tradingStarted` variable to true:

```
function startTrading() external onlyOwner {
   tradingStarted = true;
}
```

FUNCTIONAL TESTING

Category: Centralization
Subject: Limiting trading
Severity: Critical
Overview:
The contract allows the owner to set rules that limit trading, including setting a maximum and minimum holding amount.

Code:

```
bool public limited;
uint256 public maxHoldingAmount;
uint256 public minHoldingAmount;

function setRule( bool _limited, address _uniswapV2Pair, uint256 _maxHoldingAmount, uint256
_minHoldingAmount ) external onlyOwner {
   limited = _limited;
   uniswapV2Pair = _uniswapV2Pair;
   maxHoldingAmount = _maxHoldingAmount;
   minHoldingAmount = _minHoldingAmount;
}
```

Suggestion:

- max holding amount should be more than a reasonable % of total supply (0.1% suggested by pinksale)
- min holding amount should not be more than a reasonable amount which can disable buys
- uniswapV2Pair should be immutable

FUNCTIONAL TESTING

Category: Centralization
Subject: Burning tokens
Severity: Low
Overview:
The contract allows users to burn their tokens, which can lead to centralization if a few users hold a large portion of the total supply.

Code:
```
function burn(uint256 value) external {
    _burn(msg.sender, value);
}
```

Suggestion:
Implement a decentralized governance mechanism to decide on burning tokens

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