AuditAce

FROM INCEPTION TO SUCCESS

Smart Contract Audit

FOR

Blud

DATED : 20 September 23'

MANUAL TESTING

Centralization – Enabling Trades
Severity: High
function: EnableTrading
Status: Open

Overview:
The enableTrading function permits only the contract owner to activate trading capabilities. Until this function is executed, no investors can buy, sell, or transfer their tokens. This places a high degree of control and centralization in the hands of the contract owner.

```
function enableTrading() external onlyOwner {
    require(!tradingEnabled, "Trading already enabled.");
    tradingEnabled = true;
}
```

Suggestion
To reduce centralization and potential manipulation, consider one of the following approaches:
1.Automatically enable trading after a specified condition, such as the completion of a presale, is met.
2.If manual activation is still desired, consider transferring the ownership of the contract to a trustworthy, third-party entity like a certified "PinkSale Safu" developer. This can provide investors with more confidence in the eventual activation of trading capabilities, mitigating concerns of potential bad faith actions by the original owner

AUDIT SUMMARY

Project name – Blud

Date: 20 September 2023

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

Audit Status: Passed With High Risk

Issues Found

Status	Critical	High	Medium	Low	Suggestion
Open	0	1	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:
The tests were performed using the contract deployed on the BSC Testnet, which can be found at the following address:
https://testnet.bscscan.com/token/0xb51cfe399278d529c87f9dd1d160a1c0d35faca1

Token Information

Token Address :
0xC764bf59FEdBce02AB6CAa108CD55d343C9a0aa7

Name: Blud

Symbol: Blud

Decimals: 18

Network: Binance smart chain

Token Type: BEP20

Owner: 0x14Fe23906b1ea0c9E8328e0f8dBfBF3799F0FbC8

Deployer: 0x14Fe23906b1ea0c9E8328e0f8dBfBF3799F0FbC8

Token Supply: 1,000,000,000

Checksum:
b2c90d6fa070c0c68d3deae438aae07f7f200dc2

Testnet version:
The tests were performed using the contract deployed on the BSC Testnet, which can be found at the following address: https://testnet.bscscan.com/token/0xb51cfe399278d529c87f 9dd1d160a1c0d35faca1

TOKEN OVERVIEW

buy fee: 0%

Sell fee: 0-10%

transfer fee: 0%

Fee Privilege: Owner

Ownership: Owned

Minting: None

Max Tx: Yes (0.1%-100% totalsupply)

Blacklist: No

Other Privileges:

- Initial distribution of the tokens

- Modifying fees

- Enabling trades

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.
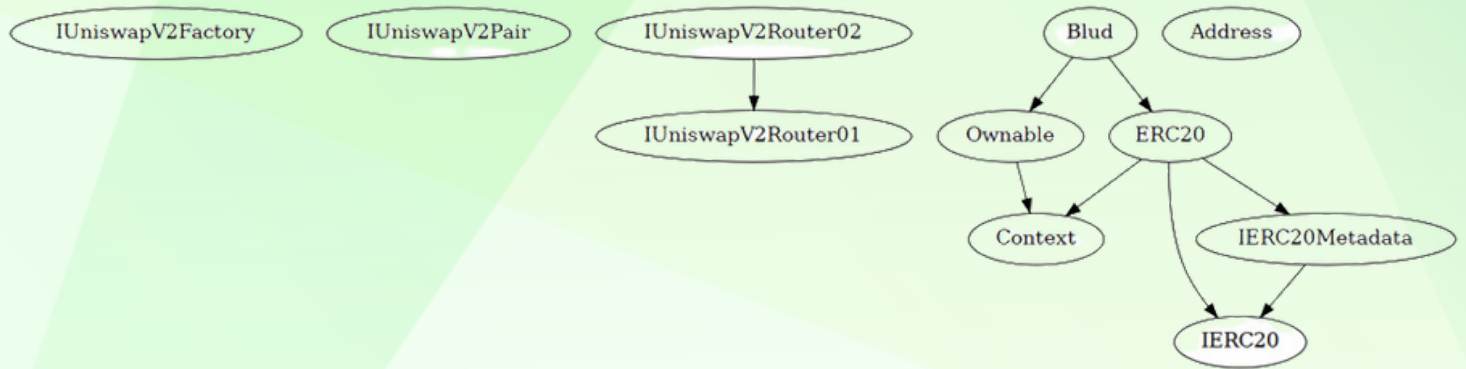
Findings

Severity	Found
◆ **Critical**	0
◆ **High-Risk**	1
◆ **Medium-Risk**	0
◆ **Low-Risk**	0
◆ **Gas Optimization / Suggestions**	0

INHERITANCE TREE

POINTS TO NOTE

- Owner is able to adjust sell fees within 0-10%

- Owner is not able to set buy/transfer fees

- Owner is not able to blacklist an arbitrary wallet

- Owner is not able to disable trades

- Owner is not able to mint new tokens

- Owner is able to limit maximum amount of buy/sell (0.1% to 100% of total supply)

- Owner must enable trades manually

STATIC ANALYSIS

```
INFO:Detectors:
Blud.constructor().router (contracts/Token.sol#645) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
Blud.claimStuckTokens(address) (contracts/Token.sol#694-706) ignores return value by address(msg.sender).sendValue(address(this).balance) (contracts/Token
.sol#700)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
Context._msgData() (contracts/Token.sol#394-397) is never used and should be removed
ERC20._burn(address,uint256) (contracts/Token.sol#584-599) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.17 (contracts/Token.sol#7) allows old versions
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (contracts/Token.sol#375-386):
        - (success) = recipient.call{value: amount}() (contracts/Token.sol#384)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (contracts/Token.sol#69) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (contracts/Token.sol#71) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (contracts/Token.sol#102) is not in mixedCase
Function IUniswapV2Router01.WETH() (contracts/Token.sol#142) is not in mixedCase
Parameter Blud.updateFees(uint256)._marketingFeeOnSell (contracts/Token.sol#725) is not in mixedCase
Parameter Blud.changeMarketingWallet(address)._marketingWallet (contracts/Token.sol#737) is not in mixedCase
Parameter Blud.setMaxTransactionAmounts(uint256,uint256)._maxTransactionAmountBuy (contracts/Token.sol#848) is not in mixedCase
Parameter Blud.setMaxTransactionAmounts(uint256,uint256)._maxTransactionAmountSell (contracts/Token.sol#849) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (contracts/Token.sol#395)" inContext (contracts/Token.sol#389-398)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (contracts/Token.sol#147) is too
similar to IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (contracts/Token.sol#148)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
Blud.marketingFeeOnBuy (contracts/Token.sol#634) should be immutable
Blud.marketingFeeOnTransfer (contracts/Token.sol#636) should be immutable
Blud.uniswapV2Pair (contracts/Token.sol#630) should be immutable
Blud.uniswapV2Router (contracts/Token.sol#629) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Slither:../contracts/Token.sol analyzed (11 contracts with 88 detectors), 22 result(s) found
```

Result => A static analysis of contract's source code has been performed using slither,

No major issues were found in the output

CONTRACT ASSESMENT

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
IUniswapV2Factory	Interface				
└	feeTo	External ❗		NO❗
└	feeToSetter	External ❗		NO❗
└	getPair	External ❗		NO❗
└	allPairs	External ❗		NO❗
└	allPairsLength	External ❗		NO❗
└	createPair	External ❗	🔴	NO❗
└	setFeeTo	External ❗	🔴	NO❗
└	setFeeToSetter	External ❗	🔴	NO❗
IUniswapV2Pair	Interface				
└	name	External ❗		NO❗
└	symbol	External ❗		NO❗
└	decimals	External ❗		NO❗
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🔴	NO❗
└	transfer	External ❗	🔴	NO❗
└	transferFrom	External ❗	🔴	NO❗
└	DOMAIN_SEPARATOR	External ❗		NO❗
└	PERMIT_TYPEHASH	External ❗		NO❗
└	nonces	External ❗		NO❗
└	permit	External ❗	🔴	NO❗
└	MINIMUM_LIQUIDITY	External ❗		NO❗
└	factory	External ❗		NO❗
└	token0	External ❗		NO❗
└	token1	External ❗		NO❗
└	getReserves	External ❗		NO❗
└	price0CumulativeLast	External ❗		NO❗
└	price1CumulativeLast	External ❗		NO❗
└	kLast	External ❗		NO❗
└	mint	External ❗	🔴	NO❗

CONTRACT ASSESMENT

└	burn	External ❗	🔴	NO❗
└	swap	External ❗	🔴	NO❗
└	skim	External ❗	🔴	NO❗
└	sync	External ❗	🔴	NO❗
└	initialize	External ❗	🔴	NO❗
IUniswapV2Router01	Interface			
└	factory	External ❗		NO❗
└	WETH	External ❗		NO❗
└	addLiquidity	External ❗	🔴	NO❗
└	addLiquidityETH	External ❗	💵	NO❗
└	removeLiquidity	External ❗	🔴	NO❗
└	removeLiquidityETH	External ❗	🔴	NO❗
└	removeLiquidityWithPermit	External ❗	🔴	NO❗
└	removeLiquidityETHWithPermit	External ❗	🔴	NO❗
└	swapExactTokensForTokens	External ❗	🔴	NO❗
└	swapTokensForExactTokens	External ❗	🔴	NO❗
└	swapExactETHForTokens	External ❗	💵	NO❗
└	swapTokensForExactETH	External ❗	🔴	NO❗
└	swapExactTokensForETH	External ❗	🔴	NO❗
└	swapETHForExactTokens	External ❗	💵	NO❗
└	quote	External ❗		NO❗
└	getAmountOut	External ❗		NO❗
└	getAmountIn	External ❗		NO❗
└	getAmountsOut	External ❗		NO❗
└	getAmountsIn	External ❗		NO❗
IUniswapV2Router02	Interface	IUniswapV2Router01		
└	removeLiquidityETHSupportingFeeOnTransferTokens	External ❗	🔴	NO❗
└	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External ❗	🔴	NO❗
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🔴	NO❗
└	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗	💵	NO❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🔴	NO❗

CONTRACT ASSESMENT

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❗
Address	Library			
└	sendValue	Internal 🔒	🔴	
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
Ownable	Implementation	Context		
└	<Constructor>	Public ❗	🔴	NO❗
└	owner	Public ❗		|NO❗
└	renounceOwnership	Public ❗	🔴	onlyOwner
└	transferOwnership	Public ❗	🔴	onlyOwner
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 🔒	🔴	
└	_approve	Internal 🔒	🔴	
└	_beforeTokenTransfer	Internal 🔒	🔴	
└	_afterTokenTransfer	Internal 🔒	🔴	

CONTRACT ASSESMENT

||||||

Blud	Implementation	ERC20, Ownable		
└	<Constructor>	Public ❗	🔴	ERC20
└	<Receive Ether>	External ❗	💵	NO❗
└	claimStuckTokens	External ❗	🔴	onlyOwner
└	excludeFromFees	Public ❗	🔴	onlyOwner
└	isExcludedFromFees	Public ❗		NO❗
└	updateFees	External ❗	🔴	onlyOwner
└	changeMarketingWallet	External ❗	🔴	onlyOwner
└	enableTrading	External ❗	🔴	onlyOwner
└	_transfer	Internal 🔒	🔴	
└	setEnableMaxTransactionLimit	External ❗	🔴	onlyOwner
└	setMaxTransactionAmounts	External ❗	🔴	onlyOwner
└	excludeFromMaxTransactionLimit	External ❗	🔴	onlyOwner
└	isExcludedFromMaxTransaction	Public ❗		NO❗

Legend

Symbol	Meaning
🔴	Function can modify state
💵	Function is payable

FUNCTIONAL TESTING

1- Adding liquidity (passed):

https://testnet.bscscan.com/tx/0xbb3ba02f1a197d54b8182ccbd86687507ec7fc3d2ec4a8740d2eec82c67adfa8

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

https://testnet.bscscan.com/tx/0xff790ae53f599c990776c78419aeceef21d5ad1bf92ae0316f6ba61a928c9135

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

https://testnet.bscscan.com/tx/0x089eefa8df4fdf31ec14b4df1ff1cc875317bba4b96162033c387d2a2d6cd9de

4- Transferring when excluded from fees (0% tax) (passed):

https://testnet.bscscan.com/tx/0xcbdaa8f4ca2291ef3c0aae58e49a36682523c597edd9f2ca9972ec1007482cf8

5- Buying when not excluded from fees (tax 0%) (passed):

https://testnet.bscscan.com/tx/0xcbdaa8f4ca2291ef3c0aae58e49a36682523c597edd9f2ca9972ec1007482cf8

6- Selling when not excluded from fees (tax 0-10%) (passed):

https://testnet.bscscan.com/tx/0x6cf03c2a18231256ae5a00b9ce466b3579556cb8660b155df0fdbc982b1f7315

7- Transferring when not excluded from fees (0% tax) (passed):

https://testnet.bscscan.com/tx/0x775ba1279db5cf9e21b59461bc148448d27fef52122041a24be464a753e72c81

MANUAL TESTING

Centralization – Enabling Trades
Severity: High
function: EnableTrading
Status: Open

Overview:
The enableTrading function permits only the contract owner to activate trading capabilities. Until this function is executed, no investors can buy, sell, or transfer their tokens. This places a high degree of control and centralization in the hands of the contract owner.

```
function enableTrading() external onlyOwner {
    require(!tradingEnabled, "Trading already enabled.");
    tradingEnabled = true;
}
```

Suggestion
To reduce centralization and potential manipulation, consider one of the following approaches:
1.Automatically enable trading after a specified condition, such as the completion of a presale, is met.
2.If manual activation is still desired, consider transferring the ownership of the contract to a trustworthy, third-party entity like a certified "PinkSale Safu" developer. This can provide investors with more confidence in the eventual activation of trading capabilities, mitigating concerns of potential bad faith actions by the original owner

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