Cyberscope

Audit Report

Blue Brilliant AI

January 2023

Type	BEP20
Network	BSC
Address	0x7b99409F607857F4dbf1980Ab2C272d5369E4ad5
Audited by	© cyberscope

Table of Contents

Review

Contract Name	BlueBrilliantAI
Compiler Version	v0.8.8+commit.dddeac2f
Optimization	200 runs
Explorer	https://bscscan.com/address/0x7b99409f607857f4dbf1980ab2c272d5369e4ad5
Address	0x7b99409f607857f4dbf1980ab2c272d5369e4ad5
Network	BSC
Symbol	BRILL
Decimals	18
Total Supply	100,000,000

Audit Updates

Initial Audit	24 Jan 2023

Source Files

Filename	SHA256
BlueBrilliantAI.sol	2f46dc075ee37d6878d2cbe4972265acdcb282f95020351ee83774cb58a07757

Analysis

● Critical ● Medium ● Minor / Informative ● Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Unresolved
●	OCTD	Transfers Contract's Tokens	Passed
●	OTUT	Transfers User's Tokens	Unresolved
●	ELFM	Exceeds Fees Limit	Passed
●	ULTW	Transfers Liquidity to Team Wallet	Passed
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Passed

ST - Stops Transactions

Criticality	Minor / Informative
Location	BlueBrilliantAI.sol#L528
Status	Unresolved

Description

The contract owner has the authority to stop the sales for all users except the owner. The owner may take advantage of it by setting the coolDownTime to a value of up to 5 minutes and prevent users from selling again until that time passed.

```
if (coolDownEnabled) {
  uint256 timePassed = block.timestamp - _lastSell[sender];
  require(timePassed >= coolDownTime, "Cooldown enabled");
  _lastSell[sender] = block.timestamp;
}
```

Recommendation

The team should carefully manage the private keys of the owner’s account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. That risk can be prevented by temporarily locking the contract or renouncing ownership.

OTUT - Transfers User's Tokens

Criticality	Minor / Informative
Location	BlueBrilliantAI.sol#L758
Status	Unresolved

Description

The contract owner has the authority to transfer the balance of a user's contract to the owner's contract. The owner may take advantage of it by calling the `rescueBSC20` function.

```
function rescueBSC20(address tokenAdd, uint256 amount) external onlyOwner {
  require(tokenAdd != address(this), "Owner can't claim contract's balance of its own tokens");
  IBEP20(tokenAdd).transfer(owner(), amount);
}
```

Recommendation

The team should carefully manage the private keys of the owner’s account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. That risk can be prevented by temporarily locking the contract or renouncing ownership.

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
●	L02	State Variables could be Declared Constant	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L07	Missing Events Arithmetic	Unresolved
●	L13	Divide before Multiply Operation	Unresolved
●	L14	Uninitialized Variables in Local Scope	Unresolved
●	L19	Stable Compiler Version	Unresolved
●	L20	Succeeded Transfer Check	Unresolved

L02 - State Variables could be Declared Constant

Criticality	Minor / Informative
Location	BlueBrilliantAI.sol#L404
Status	Unresolved

Description

State variables can be declared as constant using the constant keyword. This means that the value of the state variable cannot be changed after it has been set. Additionally, the constant variables decrease gas consumption of the corresponding transaction.

```
uint256 private launchtax = 99
```

Recommendation

Constant state variables can be useful when the contract wants to ensure that the value of a state variable cannot be changed by any function in the contract. This can be useful for storing values that are important to the contract's behavior, such as the contract's address or the maximum number of times a certain function can be called. The team is advised to add the constant keyword to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	BlueBrilliantAI.sol#L57,59,360,402,409,588,675,681,682,683,684,685,691,692,693,694,695,701,708,735
Status	Unresolved

Description

The Solidity style guide is a set of guidelines for writing clean and consistent Solidity code. Adhering to a style guide can help improve the readability and maintainability of the Solidity code, making it easier for others to understand and work with.

The followings are a few key points from the Solidity style guide:

1. Use camelCase for function and variable names, with the first letter in lowercase (e.g., myVariable, updateCounter).
2. Use PascalCase for contract, struct, and enum names, with the first letter in uppercase (e.g., MyContract, UserStruct, ErrorEnum).
3. Use uppercase for constant variables and enums (e.g., MAX_VALUE, ERROR_CODE).
4. Use indentation to improve readability and structure.
5. Use spaces between operators and after commas.
6. Use comments to explain the purpose and behavior of the code.
7. Keep lines short (around 120 characters) to improve readability.

```
mapping(address => uint256) internal _balances
mapping(address => mapping(address => uint256)) internal _allowances
function WETH() external pure returns (address);
uint256 public genesis_block
address public constant deadWallet = 0x000000000000000000000000000000000000dEaD

...
```

Recommendation

By following the Solidity naming convention guidelines, the codebase increased the readability, maintainability, and makes it easier to work with.

Find more information on the Solidity documentation https://docs.soliditylang.org/en/v0.8.17/style-guide.html#naming-convention.

L07 - Missing Events Arithmetic

Criticality	Minor / Informative
Location	BlueBrilliantAI.sol#L678,711,730,749
Status	Unresolved

Description

Events are a way to record and log information about changes or actions that occur within a contract. They are often used to notify external parties or clients about events that have occurred within the contract, such as the transfer of tokens or the completion of a task.

It's important to carefully design and implement the events in a contract, and to ensure that all required events are included. It's also a good idea to test the contract to ensure that all events are being properly triggered and logged.

```
tokenLiquidityThreshold = new_amount * 10**decimals()
deadline = _deadline
coolDownTime = time * 1 seconds
maxBuyLimit = maxBuy * 10**decimals()
```

Recommendation

By including all required events in the contract and thoroughly testing the contract's functionality, the contract ensures that it performs as intended and does not have any missing events that could cause issues with its arithmetic.

L13 - Divide before Multiply Operation

Criticality	Minor / Informative
Location	BlueBrilliantAI.sol#L611,612,619,624,629
Status	Unresolved

Description

It is important to be aware of the order of operations when performing arithmetic calculations. This is especially important when working with large numbers, as the order of operations can affect the final result of the calculation. Performing divisions before multiplications may cause loss of prediction.

```
uint256 unitBalance = deltaBalance / (denominator - swapTaxes.liquidity)
uint256 bbAmt = unitBalance * 2 * swapTaxes.bb
```

Recommendation

To avoid this issue, it is recommended to carefully consider the order of operations when performing arithmetic calculations in Solidity. It's generally a good idea to use parentheses to specify the order of operations. The basic rule is that the multiplications should be prior to the divisions.

L14 - Uninitialized Variables in Local Scope

Criticality	Minor / Informative
Location	BlueBrilliantAI.sol#L535,536,538
Status	Unresolved

Description

Using an uninitialized local variable can lead to unpredictable behavior and potentially cause errors in the contract. It's important to always initialize local variables with appropriate values before using them.

```
uint256 feeswap
uint256 feesum
Taxes memory currentTaxes
```

Recommendation

By initializing local variables before using them, the contract ensures that the functions behave as expected and avoid potential issues.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	BlueBrilliantAI.sol#L4
Status	Unresolved

Description

The `^` symbol indicates that any version of Solidity that is compatible with the specified version (i.e., any version that is a higher minor or patch version) can be used to compile the contract. The version lock is a mechanism that allows the author to specify a minimum version of the Solidity compiler that must be used to compile the contract code. This is useful because it ensures that the contract will be compiled using a version of the compiler that is known to be compatible with the code.

```
pragma solidity ^0.8.8;
```

Recommendation

The team is advised to lock the pragma to ensure the stability of the codebase. The locked pragma version ensures that the contract will not be deployed with an unexpected version. An unexpected version may produce vulnerabilities and undiscovered bugs. The compiler should be configured to the lowest version that provides all the required functionality for the codebase. As a result, the project will be compiled in a well-tested LTS (Long Term Support) environment.

L20 - Succeeded Transfer Check

Criticality	Minor / Informative
Location	BlueBrilliantAI.sol#L760
Status	Unresolved

Description

According to the ERC20 specification, the transfer methods should be checked if the result is successful. Otherwise, the contract may wrongly assume that the transfer has been established.

```
IBEP20(tokenAdd).transfer(owner(), amount)
```

Recommendation

The contract should check if the result of the transfer methods is successful. The team is advised to check the SafeERC20 library from the Openzeppelin library.

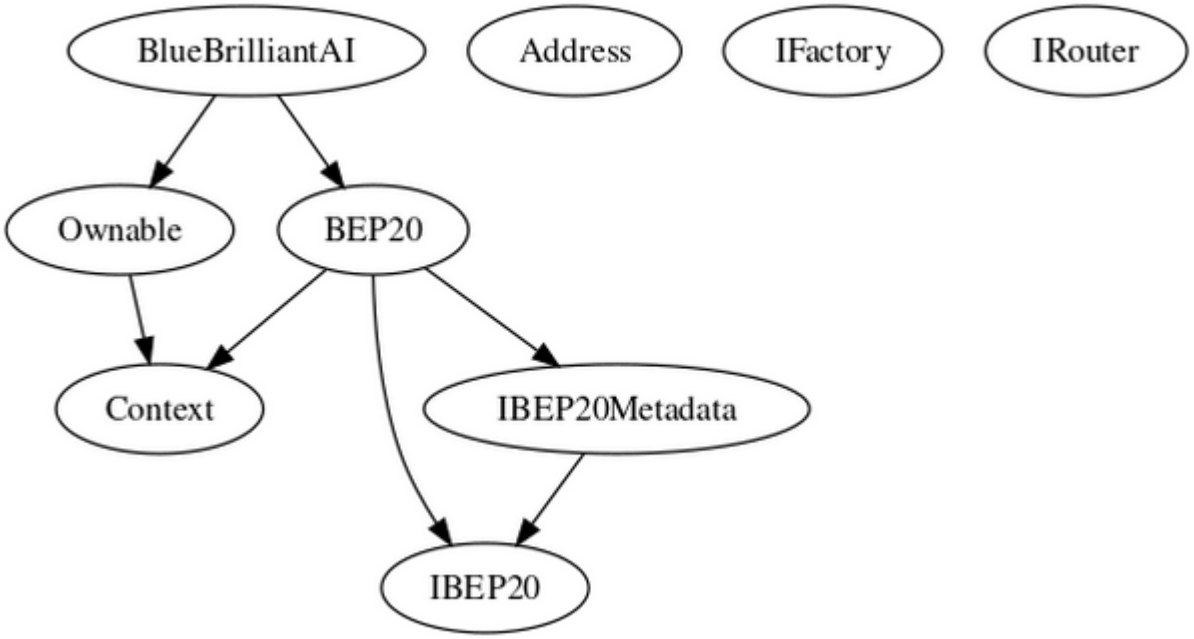
Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
IBEP20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
IBEP20Metadata	Interface	IBEP20		
	name	External		-
	symbol	External		-
	decimals	External		-
BEP20	Implementation	Context, IBEP20, IBEP20Metadata		
		Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-

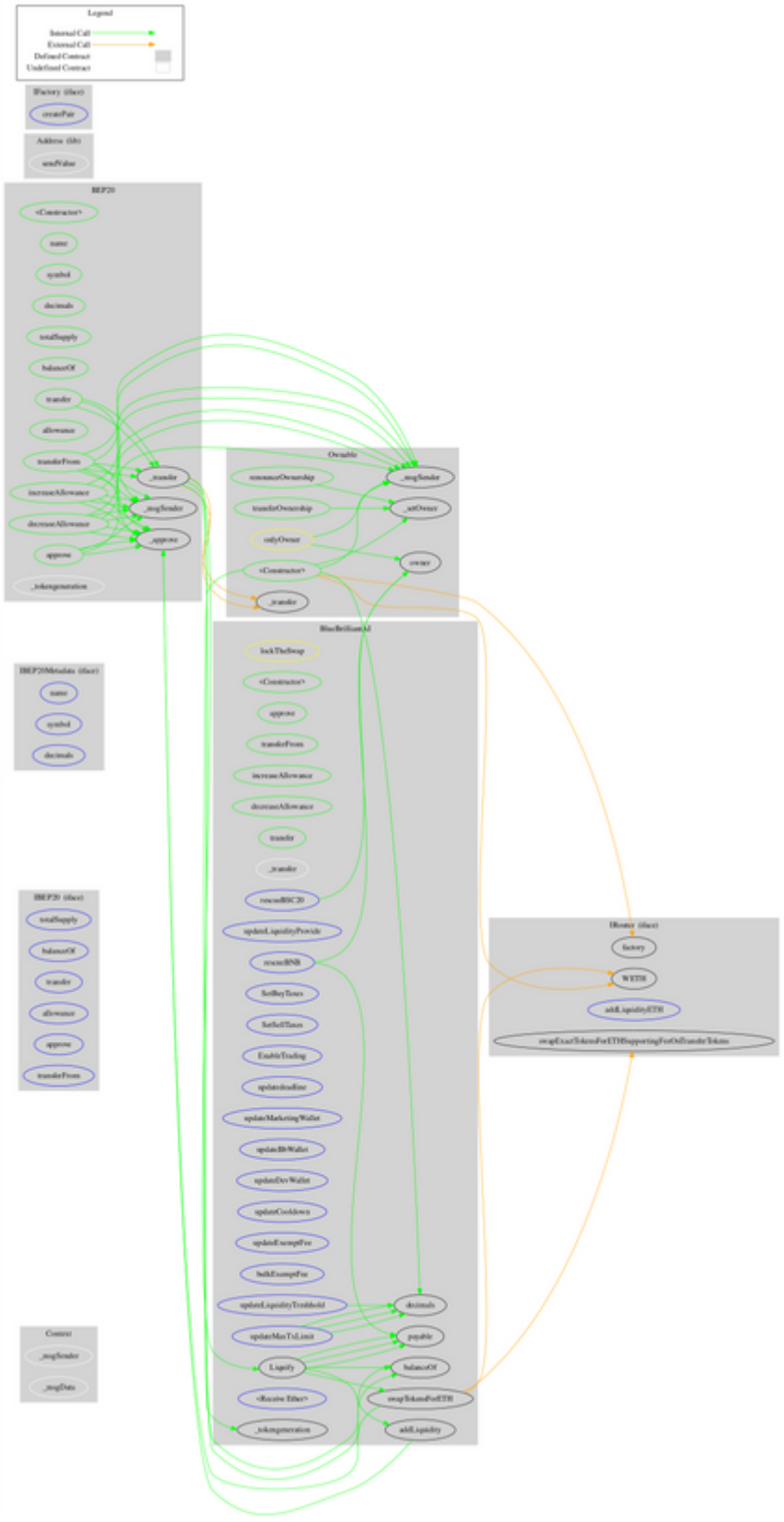
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	_transfer	Internal	✓	
	_tokengeneration	Internal	✓	
	_approve	Internal	✓	
Address	Library			
	sendValue	Internal	✓	
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_setOwner	Private	✓	
IFactory	Interface			
	createPair	External	✓	-
IRouter	Interface			
	factory	External		-
	WETH	External		-
	addLiquidityETH	External	Payable	-

	swapExactTokensForETHSupporting FeeOnTransferTokens	External	✓	-
BlueBrilliantAI	Implementation	BEP20, Ownable		
		Public	✓	BEP20
	approve	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	transfer	Public	✓	-
	_transfer	Internal	✓	
	Liquify	Private	✓	lockTheSwap
	swapTokensForETH	Private	✓	
	addLiquidity	Private	✓	
	updateLiquidityProvide	External	✓	onlyOwner
	updateLiquidityTreshhold	External	✓	onlyOwner
	SetBuyTaxes	External	✓	onlyOwner
	SetSellTaxes	External	✓	onlyOwner
	EnableTrading	External	✓	onlyOwner
	updatedeadline	External	✓	onlyOwner
	updateMarketingWallet	External	✓	onlyOwner
	updateBbWallet	External	✓	onlyOwner
	updateDevWallet	External	✓	onlyOwner
	updateCooldown	External	✓	onlyOwner
	updateExemptFee	External	✓	onlyOwner
	bulkExemptFee	External	✓	onlyOwner
	updateMaxTxLimit	External	✓	onlyOwner
	rescueBNB	External	✓	onlyOwner
	rescueBSC20	External	✓	onlyOwner
		External	Payable	-

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
IFactory (iface)
createPair
Address (lib)
sendValue
BEP20
<Constructor>
name
symbol
decimals
totalSupply
balanceOf
transfer
allowance
transferFrom
increaseAllowance
decreaseAllowance
approve
_tokengeneration
_transfer
_msgSender
_approve
Ownable
renounceOwnership
transferOwnership
onlyOwner
<Constructor>
_transfer
_msgSender
_setOwner
owner
BlueBrilliantAI
IBEP20Metadata (iface)
name
symbol
decimals
IBEP20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
Context
_msgSender
_msgData
IRouter (iface)
factory
WETH
addLiquidityETH
swapExactTokensForETHSupportingFeeOnTransferTokens
rescueBSC20
updateLiquidityProvide
rescueBNB
SetBuyTaxes
SetSellTaxes
EnableTrading
updateMarketingWallet
updateBrWallet
updateDevWallet
updateCooldown
updateExemptFee
bulkExemptFee
updateLiquidityTreshhold
updateMaxTxLimit
Liquify
<Receive Ether>
_tokengeneration
decimals
payable
balanceOf
swapTokensForETH
addLiquidity

Summary

There are some functions that can be abused by the owner like stop sales and transfer the user's tokens. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract or renouncing ownership will eliminate all the contract threats. Additionally, after the trading is enabled and until the block.number in the blockchain changes four times, the total fees are locked to 99%.

Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Cyberscope’s prior written consent. This report is not nor should be considered an “endorsement” or “disapproval” of any particular project or team. This report is not nor should be regarded as an indication of the economics or value of any “product” or “asset” created by any team or project that contracts Cyberscope to perform a security assessment. This document does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors' business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk Cyberscope’s position is that each company and individual are responsible for their own due diligence and continuous security Cyberscope’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. You agree that your access and/or use including but not limited to any services reports and materials will be at your sole risk on an as-is where-is and as-available basis Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.

The Cyberscope team

https://www.cyberscope.io