Cyberscope

Audit Report

Bull Meerkat

January 2024

Network	ETH
Address	0xB12aE4921859D2A2F0363E880CC70A472880B250
Audited by	© cyberscope

Analysis

● Critical ● Medium ● Minor / Informative ● Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Unresolved
●	OTUT	Transfers User's Tokens	Passed
●	ELFM	Exceeds Fees Limit	Passed
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Passed

Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	IVU	Inefficient Variable Usage	Unresolved
●	IDI	Immutable Declaration Improvement	Unresolved
●	MEE	Missing Events Emission	Unresolved
●	RFO	Redundant Function Override	Unresolved
●	RID	Redundant Interface Declaration	Unresolved
●	RSW	Redundant Storage Writes	Unresolved
●	RC	Repetitive Calculations	Unresolved
●	L09	Dead Code Elimination	Unresolved
●	L18	Multiple Pragma Directives	Unresolved
●	L19	Stable Compiler Version	Unresolved

Table of Contents

Analysis **1**
Diagnostics **2**
Table of Contents **3**
Review **5**
Audit Updates 5
Source Files 5
Findings Breakdown **6**
ST - Stops Transactions 7
Description 7
Recommendation 7
IVU - Inefficient Variable Usage 8
Description 8
Recommendation 9
IDI - Immutable Declaration Improvement 10
Description 10
Recommendation 10
MEE - Missing Events Emission 11
Description 11
Recommendation 11
RFO - Redundant Function Override 12
Description 12
Recommendation 12
RID - Redundant Interface Declaration 13
Description 13
Recommendation 13
RSW - Redundant Storage Writes 14
Description 14
Recommendation 14
RC - Repetitive Calculations 15
Description 15
Recommendation 15
L09 - Dead Code Elimination 16
Description 16
Recommendation 16
L18 - Multiple Pragma Directives 17
Description 17
Recommendation 17
L19 - Stable Compiler Version 18
Description 18

Recommendation 18
Functions Analysis 19
Inheritance Graph 22
Flow Graph 23
Summary 24
Disclaimer 25
About Cyberscope 26

Review

Contract Name	BullMeerkatToken
Compiler Version	v0.8.22+commit.4fc1097e
Optimization	200 runs
Explorer	https://etherscan.io/address/0xb12ae4921859d2a2f0363e880cc70a472880b250
Address	0xb12ae4921859d2a2f0363e880cc70a472880b250
Network	ETH
Symbol	BULMKT
Decimals	18
Total Supply	1,000,000,000
Badge Eligibility	Yes

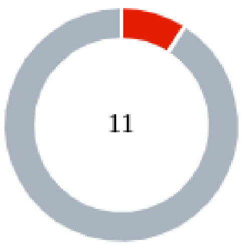
Audit Updates

Initial Audit	29 Jan 2024

Source Files

Filename	SHA256
BullMeerkatToken.sol	aacf7837f3de221c33297b3dd17dfff91d2df09fd56e2675cf99fdc511f45803

Findings Breakdown



● Critical		1
● Medium		0
● Minor / Informative		10

Severity		Unresolved	Acknowledged	Resolved	Other
●	Critical	1	0	0	0
●	Medium	0	0	0	0
●	Minor / Informative	10	0	0	0

ST - Stops Transactions

Criticality	Critical
Location	BullMeerkatToken.sol#L765,778
Status	Unresolved

Description

The transactions are initially disabled for all users excluding the authorized addresses. The owner can enable the transactions for all users. Once the transactions are enable the owner will not be able to disable them again.

```
require(
    tradingOpen || _isExcludedFromLimits[msg.sender],
    "Trading is not yet enabled."
  );
```

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. Some suggestions are:

- Introduce a multi-sign wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.

IVU - Inefficient Variable Usage

Criticality	Minor / Informative
Location	BullMeerkatToken.sol#L750,795
Status	Unresolved

Description

The contract is designed to use two separate variables, `maxTxAmount` and `maxWalletBalance`, to control transaction limits within its ecosystem. Both of these variables are set to 1% of the total supply. The `maxTxAmount` restricts the amount transferred in each transaction, while the `maxWalletBalance` limits the maximum balance a recipient can hold. However, both variables are set to the same value, making one of the conditions redundant. In the current setup, if a transaction passes the `maxTxAmount` check, it will invariably pass the `maxWalletBalance` check as well, rendering the latter check unnecessary. This redundancy not only adds unnecessary complexity to the contract but also consumes additional gas, which could be optimized for better contract efficiency.

```
maxTxAmount = totalSupply / 100; // 1% of total supply
maxWalletBalance = totalSupply / 100; // 1% of total supply
    ...
    function checkTxLimit(
        address sender,
        address recipient,
        uint256 amount
    ) internal view {
        ...
            require(
                amount <= maxTxAmount,
                "Transfer amount exceeds the maxTxAmount."
            );
            uint256 recipientBalance = balanceOf(recipient);
            require(
                recipientBalance + amount <= maxWalletBalance,
                "Exceeds maximum wallet token amount."
            );
        }
    }
```

Recommendation

It is recommended to streamline the contract by removing the redundant variable and check. Since both `maxTxAmount` and `maxWalletBalance` are equal, ensuring compliance with the `maxWalletBalance` condition inherently guarantees compliance with the `maxTxAmount` condition. Therefore, the `maxTxAmount` check can be safely removed from the `checkTxLimit` function. This simplification will reduce the contract's complexity and gas consumption, leading to a more efficient and cost-effective implementation. This will optimize and enhance the overall contract performance.

IDI - Immutable Declaration Improvement

Criticality	Minor / Informative
Location	BullMeerkatToken.sol#L751,752
Status	Unresolved

Description

The contract declares state variables that their value is initialized once in the constructor and are not modified afterwards. The `immutable` is a special declaration for this kind of state variables that saves gas when it is defined.

```
maxTxAmount
maxWalletBalance
```

Recommendation

By declaring a variable as immutable, the Solidity compiler is able to make certain optimizations. This can reduce the amount of storage and computation required by the contract, and make it more gas-efficient.

MEE - Missing Events Emission

Criticality	Minor / Informative
Location	BullMeerkatToken.sol#L815,820
Status	Unresolved

Description

The contract performs actions and state mutations from external methods that do not result in the emission of events. Emitting events for significant actions is important as it allows external parties, such as wallets or dApps, to track and monitor the activity on the contract. Without these events, it may be difficult for external parties to accurately determine the current state of the contract.

```
_isExcludedFromLimits[account] = excluded;
tradingOpen = true;
```

Recommendation

It is recommended to include events in the code that are triggered each time a significant action is taking place within the contract. These events should include relevant details such as the user's address and the nature of the action taken. By doing so, the contract will be more transparent and easily auditable by external parties. It will also help prevent potential issues or disputes that may arise in the future.

RFO - Redundant Function Override

Criticality	Minor / Informative
Location	BullMeerkatToken.sol#L824
Status	Unresolved

Description

The contract employs a custom `renounceOwnership` function, which is defined as a public function overriding the base implementation. This function delegates the inherited `super.renounceOwnership()` method. The custom implementation in this contract does not modify or extend the functionality of the inherited method. Consequently, the inclusion of this override function is redundant, as it merely replicates the existing functionality without any additional benefits or modifications.

```
function renounceOwnership() public override onlyOwner {
    super.renounceOwnership();
}
```

Recommendation

It is recommended to remove the custom `renounceOwnership` function from the contract. This action will streamline the contract by eliminating unnecessary code, reducing potential confusion, and adhering more closely to the standard implementation of the ERC20 contract. By relying on the inherited `renounceOwnership` function, the contract can maintain clarity and efficiency, ensuring that it conforms to standard practices.

RID - Redundant Interface Declaration

Criticality	Minor / Informative
Location	BullMeerkatToken.sol#L61,119
Status	Unresolved

Description

The contract includes an interface declaration that does not contain any code. While this does not affect the contract's security or functionality, it can increase complexity and make it harder to comprehend, which may result in maintenance difficulties and security risks.

```
interface IERC721Errors {}
interface IERC1155Errors {}
```

Recommendation

To ensure that contracts are efficient and easy to maintain, it's recommended to avoid creating redundant interface declarations. Developers should only define interfaces for the functions and events that need to be accessed from other contracts.

RSW - Redundant Storage Writes

Criticality	Minor / Informative
Location	BullMeerkatToken.sol#L815,819
Status	Unresolved

Description

The contract modifies the state of the following variables without checking if their current value is the same as the one given as an argument. As a result, the contract performs redundant storage writes, when the provided parameter matches the current state of the variables, leading to unnecessary gas consumption and inefficiencies in contract execution.

```
function setExcludeFromLimits(
    address account,
    bool excluded
) external onlyOwner {
    _isExcludedFromLimits[account] = excluded;
}

function enableTrading() external onlyOwner {
    tradingOpen = true;
}
```

Recommendation

The team is advised to implement additional checks within to prevent redundant storage writes when the provided argument matches the current state of the variables. By incorporating statements to compare the new values with the existing values before proceeding with any state modification, the contract can avoid unnecessary storage operations, thereby optimizing gas usage.

RC - Repetitive Calculations

Criticality	Minor / Informative
Location	BullMeerkatToken.sol#L755,756
Status	Unresolved

Description

The contract contains segments with multiple occurrences of the same calculation being performed. The calculation is repeated without utilizing a variable to store its result, which leads to redundant code, hinders code readability, and increases gas consumption. Each repetition of the calculation requires computational resources and can impact the performance of the contract, especially if the calculation is resource-intensive.

```
maxTxAmount = totalSupply / 100; // 1% of total supply
maxWalletBalance = totalSupply / 100; // 1% of total supply
```

Recommendation

To address this finding and enhance the efficiency and maintainability of the contract, it is recommended to refactor the code by assigning the calculation result to a variable once and then utilizing that variable throughout the method. By storing the calculation result in a variable, the contract eliminates the need for redundant calculations and optimizes code execution.

Refactoring the code to assign the calculation result to a variable has several benefits. It improves code readability by making the purpose and intent of the calculation explicit. It also reduces code redundancy, making the method more concise, easier to maintain, and gas effective. Additionally, by performing the calculation once and reusing the variable, the contract improves performance by avoiding unnecessary computations

L09 - Dead Code Elimination

Criticality	Minor / Informative
Location	BullMeerkatToken.sol#L191
Status	Unresolved

Description

In Solidity, dead code is code that is written in the contract, but is never executed or reached during normal contract execution. Dead code can occur for a variety of reasons, such as:

- Conditional statements that are always false.
- Functions that are never called.
- Unreachable code (e.g., code that follows a return statement).

Dead code can make a contract more difficult to understand and maintain, and can also increase the size of the contract and the cost of deploying and interacting with it.

```
function _contextSuffixLength() internal view virtual returns
(uint256) {
     return 0;
}
```

Recommendation

To avoid creating dead code, it's important to carefully consider the logic and flow of the contract and to remove any code that is not needed or that is never executed. This can help improve the clarity and efficiency of the contract.

L18 - Multiple Pragma Directives

Criticality	Minor / Informative
Location	BullMeerkatToken.sol#L5,170,201,303,385,413,736
Status	Unresolved

Description

If the contract includes multiple conflicting pragma directives, it may produce unexpected errors. To avoid this, it's important to include the correct pragma directive at the top of the contract and to ensure that it is the only pragma directive included in the contract.

```
pragma solidity ^0.8.20;
pragma solidity ^0.8.20;
pragma solidity ^0.8.22;
```

Recommendation

It is important to include only one pragma directive at the top of the contract and to ensure that it accurately reflects the version of Solidity that the contract is written in.

By including all required compiler options and flags in a single pragma directive, the potential conflicts could be avoided and ensure that the contract can be compiled correctly.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	BullMeerkatToken.sol#L5,170,201,303,385,413,736
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 ^0.8.20;
pragma solidity ^0.8.20;
pragma solidity ^0.8.20;
pragma solidity ^0.8.22;
```

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.

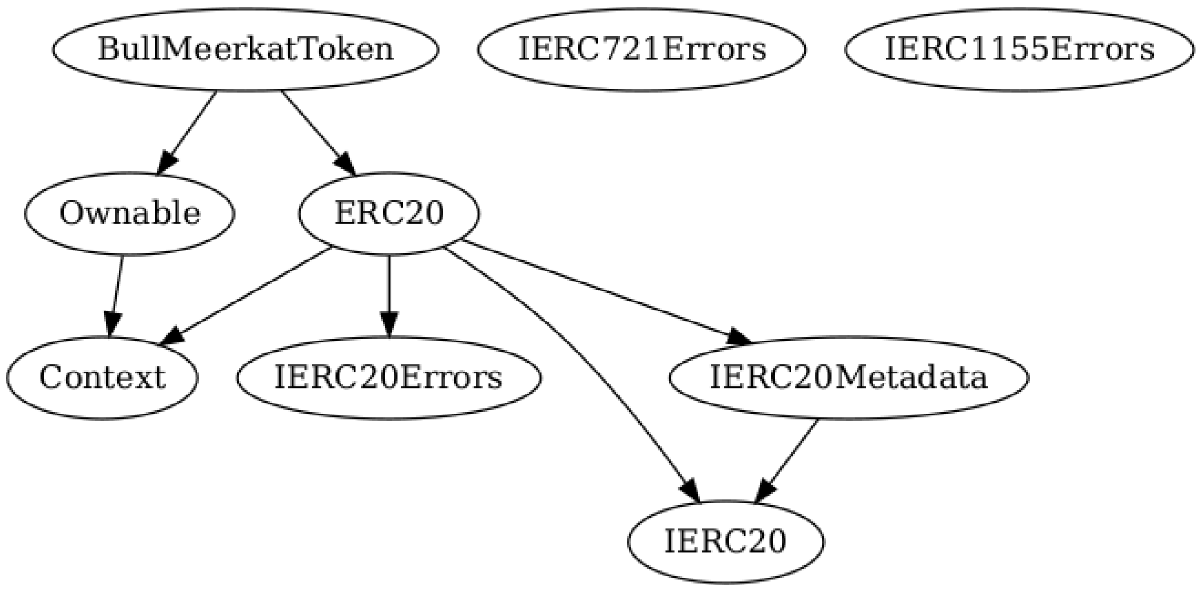
Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
IERC20Errors	Interface			
IERC721Errors	Interface			
IERC1155Errors	Interface			
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
	_contextSuffixLength	Internal		
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	_checkOwner	Internal		
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	

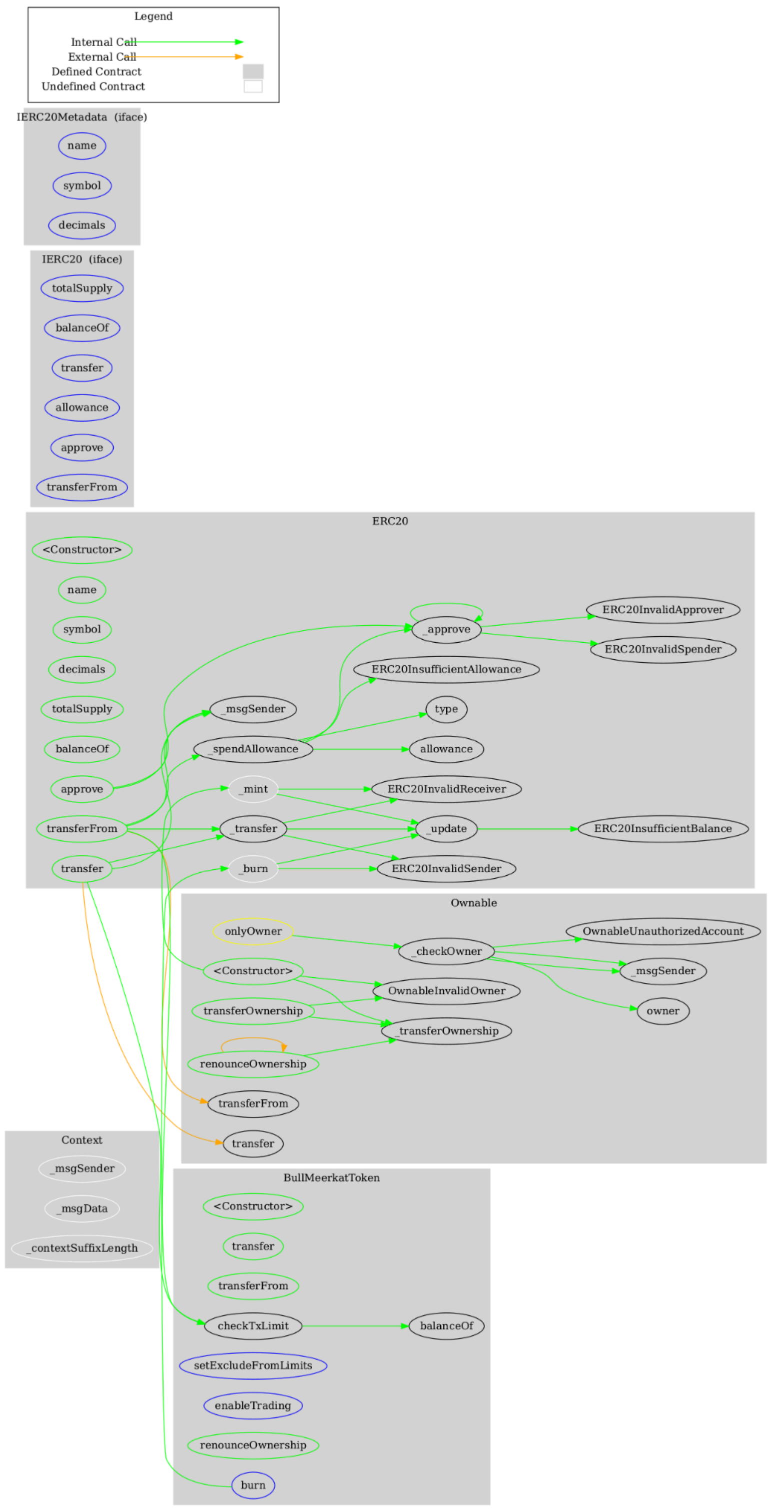
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
IERC20Metadata	Interface	IERC20		
	name	External		-
	symbol	External		-
	decimals	External		-
ERC20	Implementation	Context, IERC20, IERC20Metadata, IERC20Errors		
		Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-

	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	_transfer	Internal	✓	
	_update	Internal	✓	
	_mint	Internal	✓	
	_burn	Internal	✓	
	_approve	Internal	✓	
	_approve	Internal	✓	
	_spendAllowance	Internal	✓	
BullMeerkatToken	Implementation	ERC20, Ownable		
		Public	✓	ERC20 Ownable
	transfer	Public	✓	-
	transferFrom	Public	✓	-
	checkTxLimit	Internal		
	setExcludeFromLimits	External	✓	onlyOwner
	enableTrading	External	✓	onlyOwner
	renounceOwnership	Public	✓	onlyOwner
	burn	External	✓	-

Inheritance Graph


$$\begin{tikzcd}
BullMeerkatToken \arrow[d] \arrow[dr] & & IERC721Errors & IERC1155Errors \\
Ownable \arrow[d] & ERC20 \arrow[dl] \arrow[d] \arrow[drr] \arrow[ddr] & & \\
Context & IERC20Errors & & IERC20Metadata \arrow[dl] \\
& & IERC20 &
\end{tikzcd}$$

Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
IERC20Metadata (iface)
name
symbol
decimals
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
ERC20
<Constructor>
name
symbol
decimals
totalSupply
balanceOf
approve
transferFrom
transfer
_approve
ERC20InvalidApprover
ERC20InvalidSpender
ERC20InsufficientAllowance
_msgSender
type
_spendAllowance
allowance
_mint
ERC20InvalidReceiver
_transfer
_update
ERC20InsufficientBalance
_burn
ERC20InvalidSender
Ownable
onlyOwner
_checkOwner
OwnableUnauthorizedAccount
<Constructor>
_msgSender
OwnableInvalidOwner
transferOwnership
owner
_transferOwnership
renounceOwnership
transferFrom
transfer
Context
_msgSender
_msgData
_contextSuffixLength
BullMeerkatToken
<Constructor>
transfer
transferFrom
checkTxLimit
balanceOf
setExcludeFromLimits
enableTrading
renounceOwnership
burn

Summary

Bull Meerkat contract implements a token mechanism. This audit investigates security issues, business logic concerns, and potential improvements. Bull Meerkat is an interesting project that has a friendly and growing community. The Smart Contract analysis reported no compiler errors or critical issues. The Contract Owner can access some admin functions that can not be used in a malicious way to disturb the users' transactions.

Disclaimer

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