Cyberscope

Audit Report

Motorcoin

March 2024

Network	BSC
Address	0x6f7A1BeAB01c90545822a43AE1f5a59b779a30d9
Audited by	© cyberscope

Analysis

Critical Medium Minor / Informative Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Passed
●	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
●	RCI	Redundant Contract Inclusion	Unresolved
●	UDO	Unnecessary Decimals Override	Unresolved
●	L09	Dead Code Elimination	Unresolved
●	L19	Stable Compiler Version	Unresolved

Table of Contents

Review

Contract Name	Motorcoin
Compiler Version	v0.8.24+commit.e11b9ed9
Optimization	200 runs
Explorer	https://bscscan.com/address/0x6f7a1beab01c90545822a43ae1f5a59b779a30d9
Address	0x6f7a1beab01c90545822a43ae1f5a59b779a30d9
Network	BSC
Symbol	MTRC
Decimals	18
Total Supply	1,000,000,000
Badge Eligibility	Yes

Audit Updates

Initial Audit	30 Jan 2024 https://github.com/cyberscope-io/audits/blob/main/mtrc/v1/audit.pdf
Corrected Phase 2	18 Mar 2024

Source Files

Filename	SHA256
Motorcoin.sol	05cd3c54dc693d78e76beb37f948c111b61d215b290c191cfb037f5e467bae14

Findings Breakdown

Critical	0
Medium	0
Minor / Informative	4

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	0	0	0
Medium	0	0	0	0
Minor / Informative	4	0	0	0

RCI - Redundant Contract Inclusion

Criticality	Minor / Informative
Location	Motorcoin.sol#L209
Status	Unresolved

Description

The contract incorporates the Ownable abstract contract within its code. However, the contract does not inherit from the Ownable contract, rendering its inclusion redundant. This redundant inclusion could increase gas costs and storage without providing any tangible benefits or functionality to the contract.

```
abstract contract Ownable is Context {
    ...
}
```

Recommendation

To optimize gas usage and streamline contract code, the team is advised to remove the redundant inclusion of the Ownable abstract contract from the contract. Since the contract does not utilize it, there is no need to include the Ownable contract within its code. By removing unused code, the contract becomes more efficient and reduces unnecessary overhead.

UDO - Unnecessary Decimals Override

Criticality	Minor / Informative
Location	Motorcoin.sol#L726
Status	Unresolved

Description

The contract is currently implementing an override of the decimals function, which simply returns the value 18. This override is redundant since the extending token contract already specifies 18 decimals as its standard. In the context of ERC-20 tokens, 18 decimals is a common default, and overriding this function to return the same value adds unnecessary complexity to the contract. This redundancy does not contribute to the functionality of the contract and could potentially lead to confusion about the necessity of this override.

```
// Override the decimals function to set it to 18
function decimals() public view virtual override returns (uint8) {
    return 18;
}
```

Recommendation

Since the inherited ERC-20 contract already defines the decimals number, maintaining an overriding function that merely repeats this value does not contribute to the contract's effectiveness. As a result, it is recommended to remove the redundant `decimals` function from the contract. Removing this function will simplify the contract, making it more straightforward to maintain without impacting its operational capabilities.

L09 - Dead Code Elimination

Criticality	Minor / Informative
Location	Motorcoin.sol#L191,622
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;
}
function _burn(address account, uint256 value) internal {
    if (account == address(0)) {
        revert ERC20InvalidSender(address(0));
    }
    _update(account, address(0), value);
}
```

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.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	Motorcoin.sol#L6
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;
```

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.

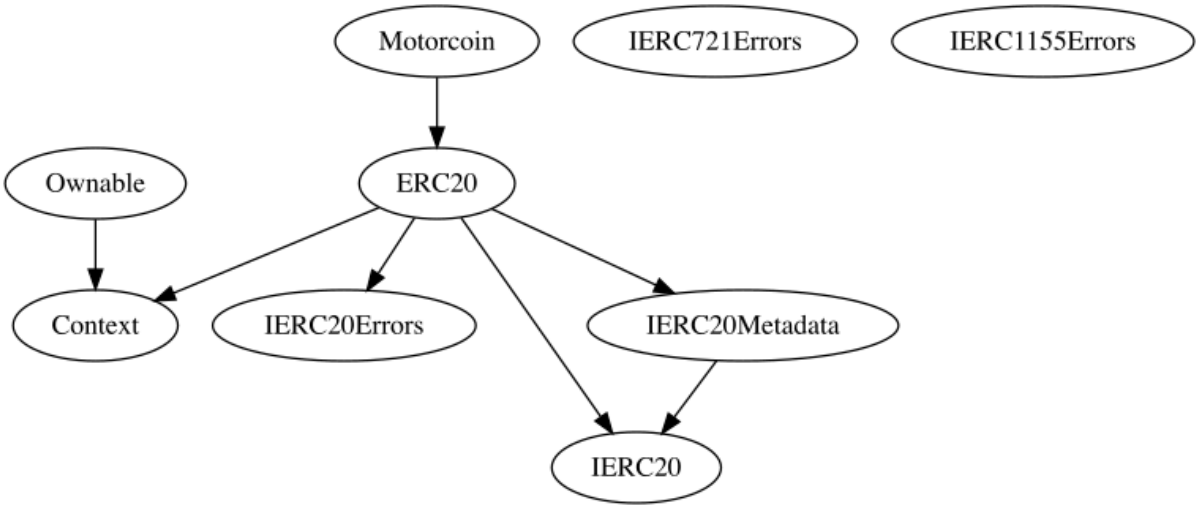
Functions Analysis

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	✓	

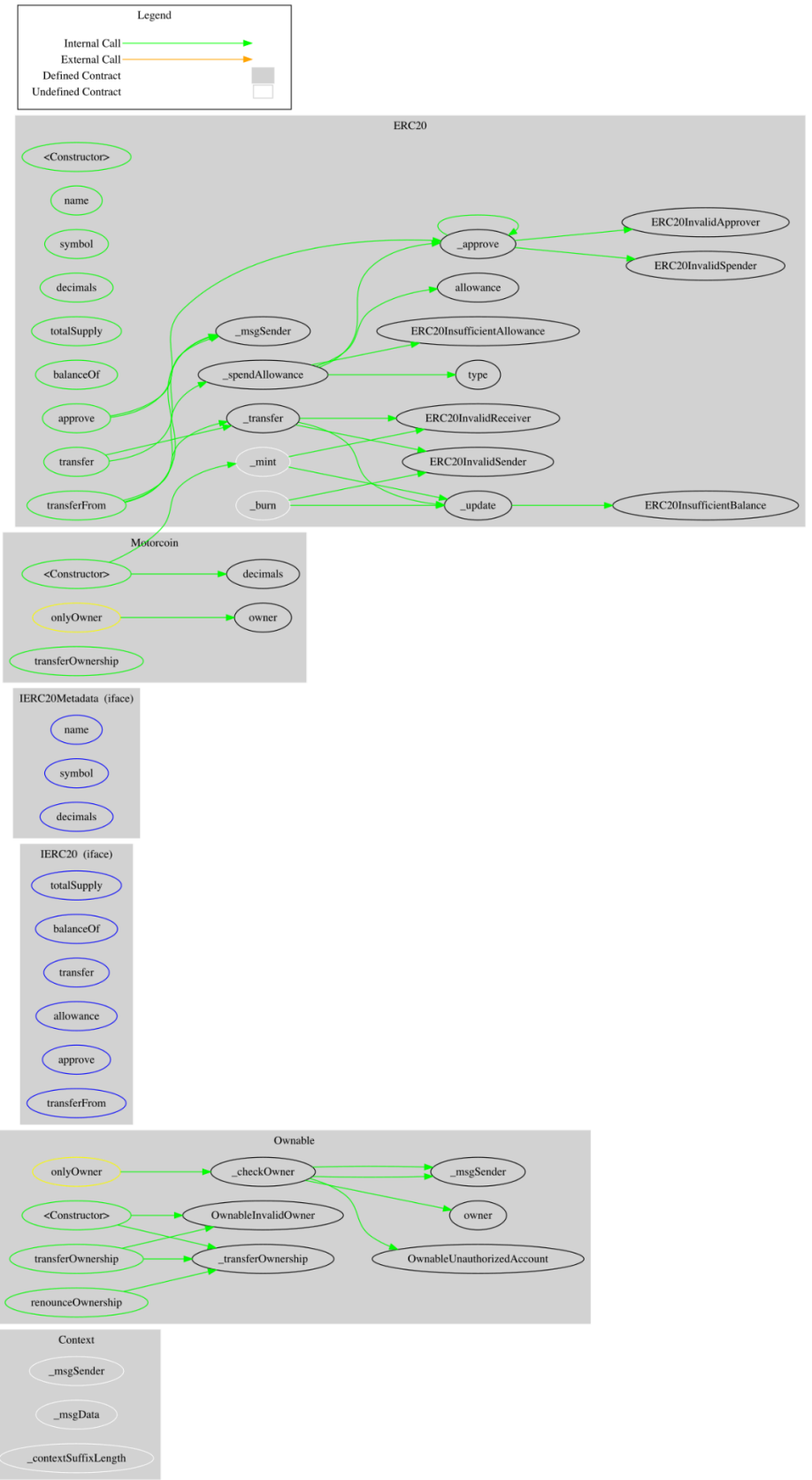
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
IERC20Metadata	Interface	IERC20		
	name	External		-
	symbol	External		-
	decimals	External		-
ERC20	Implementation	Context, IERC20, IERC20Metadata,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

Inheritance Graph



Flow Graph


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

Summary

Motorcoin contract implements a token mechanism. This audit investigates security issues, business logic concerns, and potential improvements. Motorcoin 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

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