



Cyberscope

Audit Report

ZCore

April 2023

Network BSC

Address 0x2E40565771cf2BF8622C41CDd85701519F22418A

Audited by © cyberscope

Table of Contents

Table of Contents	1
Review	2
Audit Updates	2
Source Files	3
Roles	4
Findings Breakdown	5
Analysis	6
Diagnostics	7
L18 - Multiple Pragma Directives	8
Description	8
Recommendation	8
RSML - Redundant SafeMath Library	9
Description	9
Recommendation	9
IDI - Immutable Declaration Improvement	10
Description	10
Recommendation	10
L04 - Conformance to Solidity Naming Conventions	11
Description	11
Recommendation	11
L19 - Stable Compiler Version	12
Description	12
Recommendation	12
Functions Analysis	13
Inheritance Graph	14
Flow Graph	15
Summary	16
Disclaimer	17
About Cyberscope	18

Review

Contract Name	ZCore
Compiler Version	v0.8.4+commit.c7e474f2
Optimization	2000 runs
Explorer	https://bscscan.com/address/0x2e40565771cf2bf8622c41cdd85701519f22418a
Address	0x2e40565771cf2bf8622c41cdd85701519f22418a
Network	BSC
Symbol	ZCR
Decimals	18
Total Supply	2,500,000,000

Audit Updates

Initial Audit	12 Apr 2023
---------------	-------------

Source Files

Filename	SHA256
@openzeppelin/contracts/access/Ownable.sol	9353af89436556f7ba8abb3f37a6677249a a4df6024fbfaa94f79ab2f44f3231
@openzeppelin/contracts/token/ERC20/ERC20.sol	bce14c3fd3b1a668529e375f6b70ffdf9cef 8c4e410ae99608be5964d98fa701
@openzeppelin/contracts/token/ERC20/extensions /ERC20Burnable.sol	0344809a1044e11ece2401b4f7288f414ea 41fa9d1dad24143c84b737c9fc02e
@openzeppelin/contracts/token/ERC20/extensions /IERC20Metadata.sol	af5c8a77965cc82c33b7ff844deb9826166 689e55dc037a7f2f790d057811990
@openzeppelin/contracts/token/ERC20/IERC20.sol	94f23e4af51a18c2269b355b8c7cf4db800 3d075c9c541019eb8dcf4122864d5
@openzeppelin/contracts/utils/Context.sol	1458c260d010a08e4c20a4a517882259a2 3a4baa0b5bd9add9fb6d6a1549814a
@openzeppelin/contracts/utils/math/SafeMath.sol	0dc33698a1661b22981abad8e5c6f5ebca 0dfe5ec14916369a2935d888ff257a
contracts/ZCore.sol	2793c7ef09afa043b2f1cdd9bbff29534c5a 0ce9e5040b21782b4f0471c8fea5

Roles

The contract roles consist of the owner and the bridge role.

The `owner` configured the bridge role.

The `bridge` roles have the authority to:

- Burn tokens
- Mint tokens

Findings Breakdown



● Critical	0
● Medium	0
● Minor / Informative	5

Severity	Unresolved	Acknowledged	Resolved	Other
● Critical	0	0	0	0
● Medium	0	0	0	0
● Minor / Informative	5	0	0	0

Analysis

● Critical ● Medium ● Minor / Informative ● Pass

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

Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	L18	Multiple Pragma Directives	Unresolved
●	RSML	Redundant SafeMath Library	Unresolved
●	IDI	Immutable Declaration Improvement	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L19	Stable Compiler Version	Unresolved

L18 - Multiple Pragma Directives

Criticality	Minor / Informative
Location	contracts/ZCore.sol#L2@openzeppelin/contracts/utils/math/SafeMath.sol#L4@openzeppelin/contracts/utils/Context.sol#L4@openzeppelin/contracts/token/ERC20/IERC20.sol#L4@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#L4@openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.sol#L4@openzeppelin/contracts/token/ERC20/ERC20.sol#L4@openzeppelin/contracts/access/Ownable.sol#L4
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 ^0.8.0;  
pragma solidity ^0.8.4;
```

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.

RSML - Redundant SafeMath Library

Criticality	Minor / Informative
Location	contracts/ZCore.sol
Status	Unresolved

Description

SafeMath is a popular Solidity library that provides a set of functions for performing common arithmetic operations in a way that is resistant to integer overflows and underflows.

Starting with Solidity versions that are greater than or equal to 0.8.0, the arithmetic operations revert on underflow and overflow. As a result, the native functionality of the Solidity operations replaces the SafeMath library. Hence, the usage of the SafeMath library adds complexity, overhead and increases the gas consumption unnecessarily.

```
library SafeMath {...}
```

Recommendation

The team is advised to remove the SafeMath library. Since the version of the contract is greater than `0.8.0` then the pure Solidity arithmetic operations produce the same result.

If the previous functionality is required, then the contract could exploit the `unchecked { ... }` statement.

Read more about the breaking change on

<https://docs.soliditylang.org/en/v0.8.16/080-breaking-changes.html#solidity-v0-8-0-breaking-changes>.

IDI - Immutable Declaration Improvement

Criticality	Minor / Informative
Location	contracts/ZCore.sol#L22
Status	Unresolved

Description

The contract is using variables that initialize them only in the constructor. The other functions are not mutating the variables. These variables are not defined as `immutable`.

```
maxSupply
```

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.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	contracts/ZCore.sol#L38,49,59
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.

```
uint256 _amount
address _recipient
address _account
address _bridge
```

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>.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	contracts/ZCore.sol#L2
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 ^0.8.4;
```

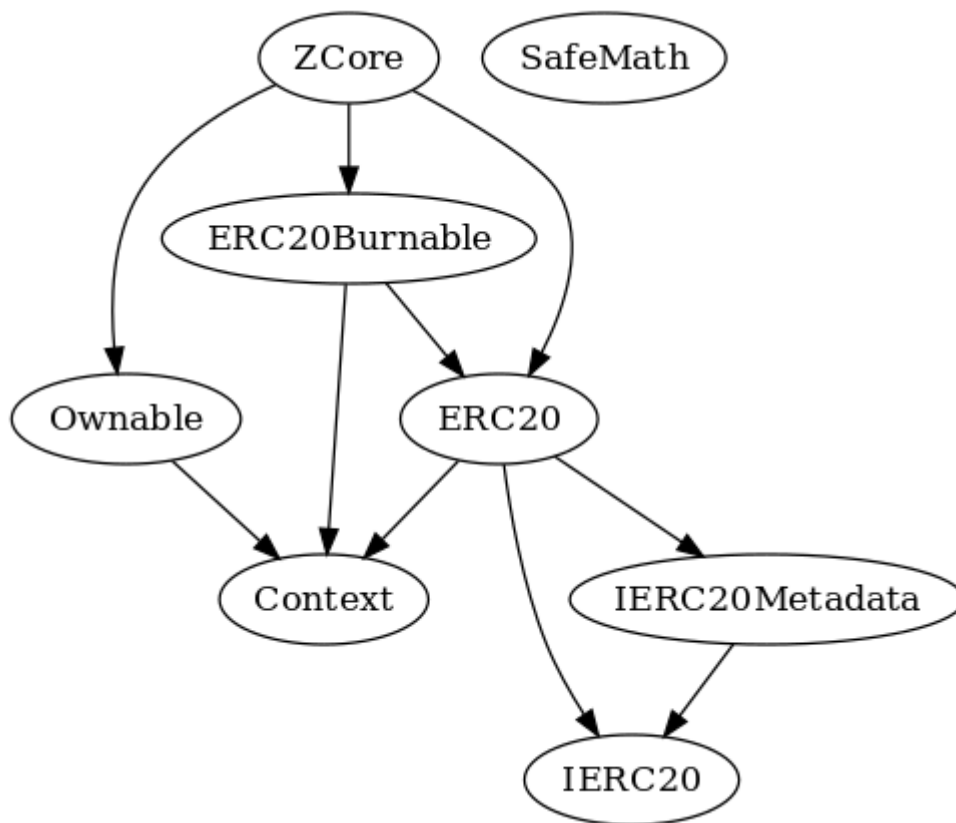
Recommendation

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

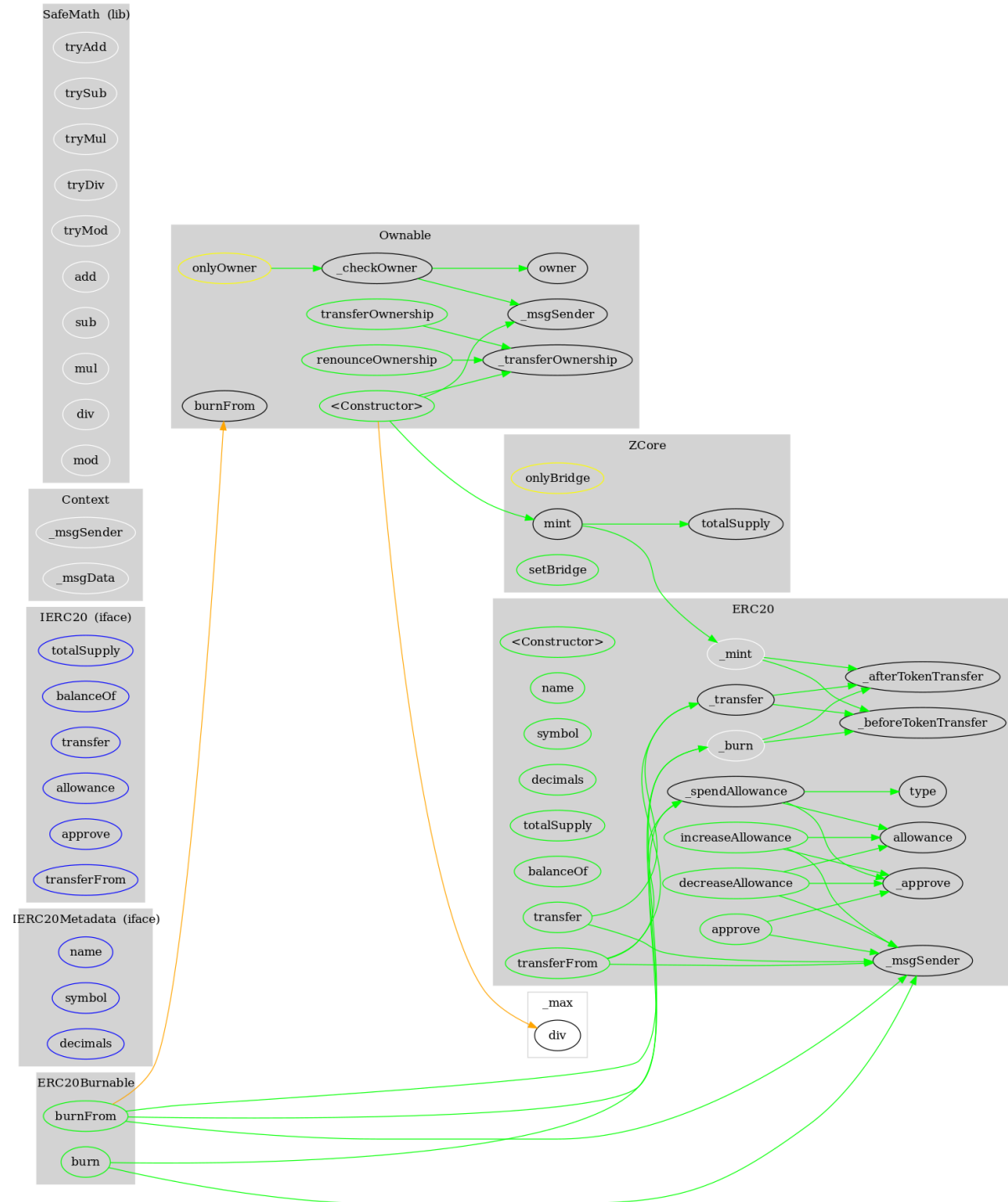
Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
ZCore	Implementation	ERC20, ERC20Burnable, Ownable		
		Public	✓	ERC20
	mint	Public	✓	onlyBridge
	burnFrom	Public	✓	onlyBridge
	setBridge	Public	✓	onlyOwner

Inheritance Graph



Legend

- Internal Call
- External Call
- Defined Contract
- Undefined Contract



Summary

ZCore contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. ZCore is an interesting project that has a friendly and growing community. The Smart Contract analysis reported no compiler error 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>