Cyberscope

# Audit Report

# Zerrw

August 2023

| | |
|---|---|
| Network | BSC |
| Address | 0x0073Ce6A7C6813c757e66bD57feaDf0C37cA70cF |
| 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 |
|---|---|---|---|
| ● | RMF | Redundant MasterChef Functionality | Unresolved |
| ● | RSK | Redundant Storage Keyword | Unresolved |
| ● | L04 | Conformance to Solidity Naming Conventions | Unresolved |
| ● | L09 | Dead Code Elimination | Unresolved |
| ● | L15 | Local Scope Variable Shadowing | Unresolved |
| ● | L17 | Usage of Solidity Assembly | Unresolved |

# Table of Contents

# Review

| | |
|---|---|
| **Contract Name** | ZerrwToken |
| **Compiler Version** | v0.6.12+commit.27d51765 |
| **Optimization** | 200 runs |
| **Explorer** | https://bscscan.com/address/0x0073ce6a7c6813c757e66bd57feadf0c37ca70cf |
| **Address** | 0x0073ce6a7c6813c757e66bd57feadf0c37ca70cf |
| **Network** | BSC |
| **Symbol** | ZRW |
| **Decimals** | 18 |
| **Total Supply** | 100,000,000 |

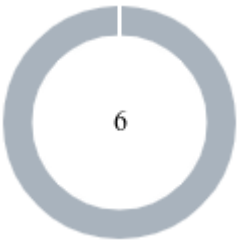
## Audit Updates

| | |
|---|---|
| **Initial Audit** | 13 Aug 2023 https://github.com/cyberscope-io/audits/blob/main/ryze/v1/audit.pdf |
| **Corrected Phase 2** | 23 Aug 2023 |

## Source Files

| | |
|---|---|
| **Filename** | SHA256 |
| **ZerrwToken.sol** | 04de9e1a23f66b8e07fb2ee861bdd38b2209010b8bce33e54ac9319c0692a6ac |

# Findings Breakdown




| | |
|---|---|
| Critical | 0 |
| Medium | 0 |
| Minor / Informative | 6 |

| Severity | Unresolved | Acknowledged | Resolved | Other |
|---|---|---|---|---|
| Critical | 0 | 0 | 0 | 0 |
| Medium | 0 | 0 | 0 | 0 |
| Minor / Informative | 6 | 0 | 0 | 0 |

## RMF - Redundant MasterChef Functionality

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | ZerrwToken.sol#L1023,1027,1258,1263,1268,1273 |
| **Status** | Unresolved |

## Description

There are code segments that could be optimized. A segment may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer operations.

The contract declares certain functions and modifiers that are not used in a meaningful way by the contract. For instance, the contract contains functionality surrounding the `_masterchef` enumerable set, such as adding and removing masterchef addresses, and the `treasury` function. However, the `treasury` function's visibility is internal and not being used by any other function in the source code. As a result, the contract contains redundant code segments and functionality.

```
EnumerableSet.AddressSet private _masterchef;

function treasury(address _to, uint256 _amount) internal onlyMasterChef
returns(bool) {
    _store(_to, _amount);
    _moveDelegates(address(0), _delegates[_to], _amount);
    return true;
}
...
```

## Recommendation

The team is advised to take these segments into consideration and rewrite them so the runtime will be more performant. That way it will improve the efficiency and performance of the source code and reduce the cost of executing it.

# RSK - Redundant Storage Keyword

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | ZerrwToken.sol#L882,889,903,937,944,958,991,998,1012 |
| **Status** | Unresolved |

## Description

The contract uses the `storage` keyword in a view function. The `storage` keyword is used to persist data on the contract's storage. View functions are functions that do not modify the state of the contract and do not perform any actions that cost gas (such as sending a transaction). As a result, the use of the `storage` keyword in view functions is redundant.

```
Set storage set
AddressSet storage set
UintSet storage set
```

## Recommendation

It is generally considered good practice to avoid using the `storage` keyword in view functions because it is unnecessary and can make the code less readable.

## L04 - Conformance to Solidity Naming Conventions

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | ZerrwToken.sol#L514,515,1023,1030,1254,1259 |
| **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 private constant _preMineSupply = 100000000 * 1e18
uint256 private constant _maxSupply = 1000000000 * 1e18
uint256 _amount
address _to
mapping (address => address) internal _delegates
address _addMasterChef
address _delMasterchef
```

## 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.

# L09 - Dead Code Elimination

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | ZerrwToken.sol#L340,345,376,405,431,441,460,474,484,753,792,889,903,944,958,984,991,998,1012,1023 |
| **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 min(uint256 x, uint256 y) internal pure returns (uint256 z) {
        z = x < y ? x : y;
    }

...
```

## 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.

## L15 - Local Scope Variable Shadowing

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | ZerrwToken.sol#L539 |
| **Status** | Unresolved |

### Description

Local scope variable shadowing occurs when a local variable with the same name as a variable in an outer scope is declared within a function or code block. When this happens, the local variable "shadows" the outer variable, meaning that it takes precedence over the outer variable within the scope in which it is declared.

```
string memory symbol
string memory name
```

### Recommendation

It's important to be aware of shadowing when working with local variables, as it can lead to confusion and unintended consequences if not used correctly. It's generally a good idea to choose unique names for local variables to avoid shadowing outer variables and causing confusion.

## L17 - Usage of Solidity Assembly

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | ZerrwToken.sol#L383,502,1249 |
| **Status** | Unresolved |

### Description

Using assembly can be useful for optimizing code, but it can also be error-prone. It's important to carefully test and debug assembly code to ensure that it is correct and does not contain any errors.

Some common types of errors that can occur when using assembly in Solidity include Syntax, Type, Out-of-bounds, Stack, and Revert.

```
assembly {
            codehash := extcodehash(account)
        }

assembly {
                    let returndata_size := mload(returndata)
                    revert(add(32, returndata), returndata_size)
                }
assembly { chainId := chainid() }
```

### Recommendation

It is recommended to use assembly sparingly and only when necessary, as it can be difficult to read and understand compared to Solidity code.

# Functions Analysis

| Contract | Type | Bases | | |
|---|---|---|---|---|
| | **Function Name** | **Visibility** | **Mutability** | **Modifiers** |
| | | | | |
| **Context** | Implementation | | | |
| | | Internal | ✓ | |
| | _msgSender | Internal | | |
| | _msgData | Internal | | |
| | | | | |
| **Ownable** | Implementation | Context | | |
| | | Internal | ✓ | |
| | owner | Public | | - |
| | renounceOwnership | Public | ✓ | onlyOwner |
| | transferOwnership | Public | ✓ | onlyOwner |
| | _transferOwnership | Internal | ✓ | |
| | | | | |
| **IBEP20** | Interface | | | |
| | totalSupply | External | | - |
| | preMineSupply | External | | - |
| | maxSupply | External | | - |
| | decimals | External | | - |
| | symbol | External | | - |

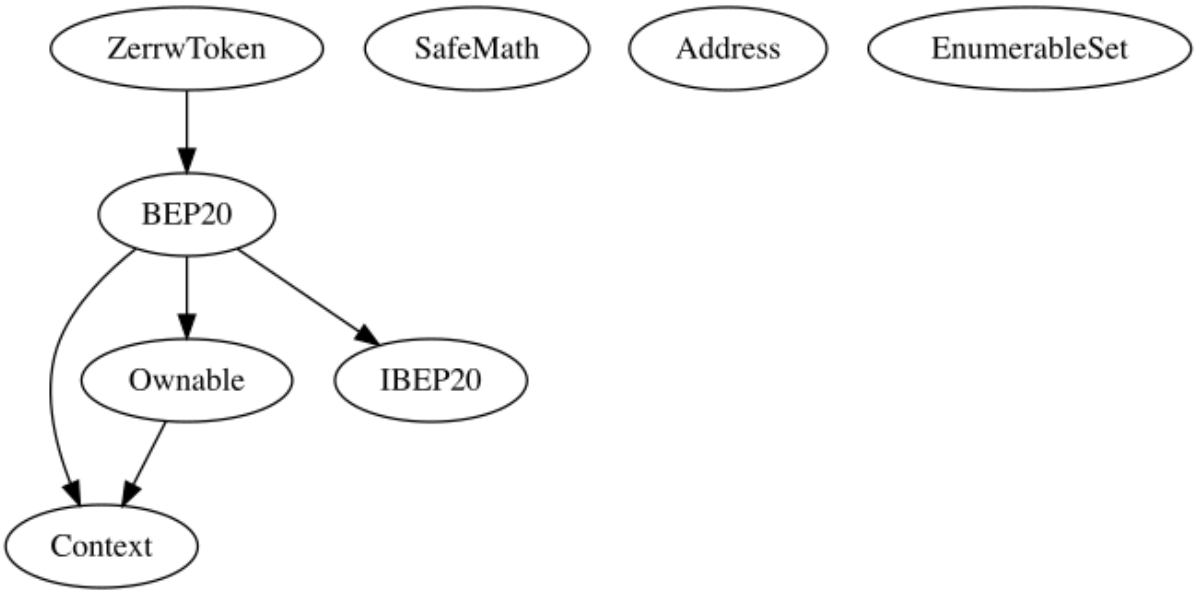
| | | | | |
|---|---|---|---|---|
| | name | External | | - |
| | getOwner | External | | - |
| | balanceOf | External | | - |
| | transfer | External | ✓ | - |
| | allowance | External | | - |
| | approve | External | ✓ | - |
| | transferFrom | External | ✓ | - |
| | | | | |
| **SafeMath** | Library | | | |
| | add | Internal | | |
| | sub | Internal | | |
| | sub | Internal | | |
| | mul | Internal | | |
| | div | Internal | | |
| | div | Internal | | |
| | mod | Internal | | |
| | mod | Internal | | |
| | min | Internal | | |
| | sqrt | Internal | | |
| | | | | |
| **Address** | Library | | | |
| | isContract | Internal | | |
| | sendValue | Internal | ✓ | |

| | | | | |
|---|---|---|---|---|
| | functionCall | Internal | ✓ | |
| | functionCall | Internal | ✓ | |
| | functionCallWithValue | Internal | ✓ | |
| | functionCallWithValue | Internal | ✓ | |
| | _functionCallWithValue | Private | ✓ | |
| | | | | |
| **BEP20** | Implementation | Context, IBEP20, Ownable | | |
| | | Public | ✓ | - |
| | getOwner | External | | - |
| | name | Public | | - |
| | decimals | Public | | - |
| | symbol | Public | | - |
| | totalSupply | Public | | - |
| | preMineSupply | Public | | - |
| | maxSupply | Public | | - |
| | balanceOf | Public | | - |
| | transfer | Public | ✓ | - |
| | allowance | Public | | - |
| | approve | Public | ✓ | - |
| | transferFrom | Public | ✓ | - |
| | increaseAllowance | Public | ✓ | - |
| | decreaseAllowance | Public | ✓ | - |
| | _transfer | Internal | ✓ | |

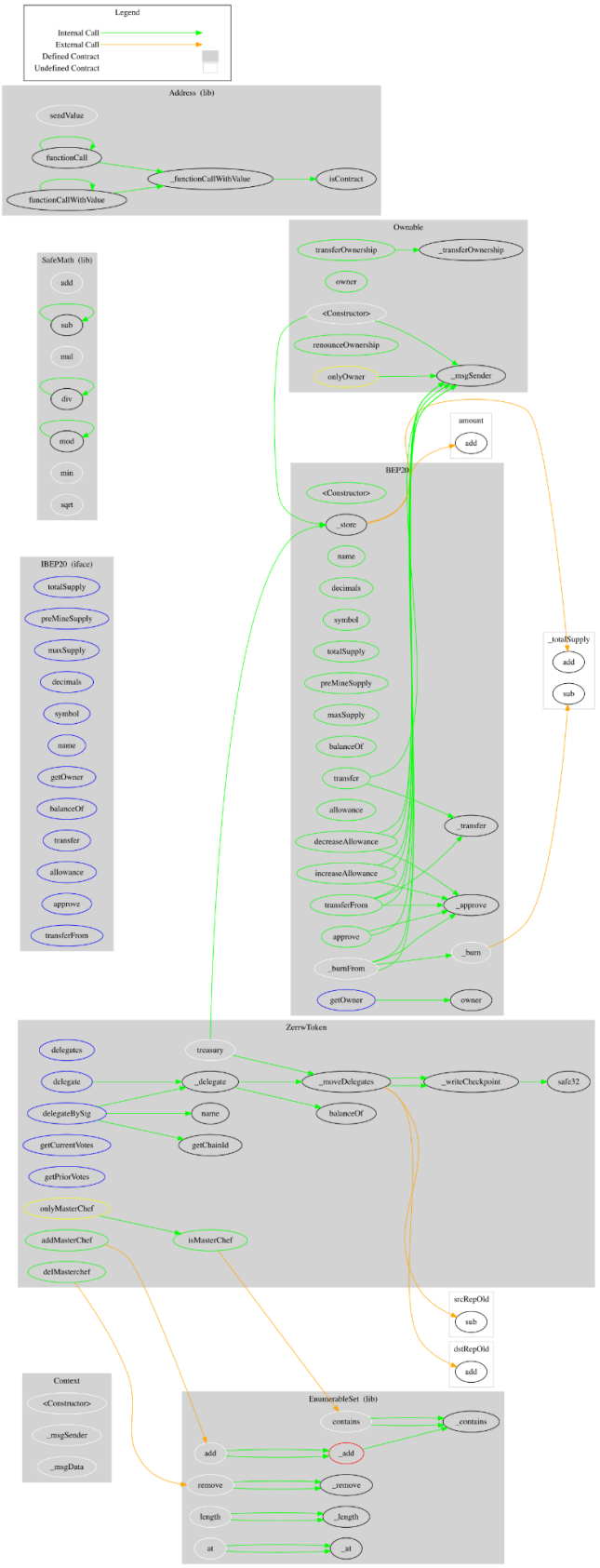
| | | | | |
|---|---|---|---|---|
| | _store | Internal | ✓ | |
| | _burn | Internal | ✓ | |
| | _approve | Internal | ✓ | |
| | _burnFrom | Internal | ✓ | |
| | | | | |
| **EnumerableSet** | Library | | | |
| | _add | Private | ✓ | |
| | _remove | Private | ✓ | |
| | _contains | Private | | |
| | _length | Private | | |
| | _at | Private | | |
| | add | Internal | ✓ | |
| | remove | Internal | ✓ | |
| | contains | Internal | | |
| | length | Internal | | |
| | at | Internal | | |
| | add | Internal | ✓ | |
| | remove | Internal | ✓ | |
| | contains | Internal | | |
| | length | Internal | | |
| | at | Internal | | |
| | | | | |
| **RYZEToken** | Implementation | BEP20 | | |

| | | | | |
|---|---|---|---|---|
| | treasury | Internal | ✓ | onlyMasterChef |
| | delegates | External | | - |
| | delegate | External | ✓ | - |
| | delegateBySig | External | ✓ | - |
| | getCurrentVotes | External | | - |
| | getPriorVotes | External | | - |
| | _delegate | Internal | ✓ | |
| | _moveDelegates | Internal | ✓ | |
| | _writeCheckpoint | Internal | ✓ | |
| | safe32 | Internal | | |
| | getChainId | Internal | | |
| | addMasterChef | Public | ✓ | onlyOwner |
| | delMasterchef | Public | ✓ | onlyOwner |
| | isMasterChef | Public | | - |

# Inheritance Graph



# Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
Address (lib)
sendValue
functionCall
_functionCallWithValue
isContract
functionCallWithValue
Ownable
transferOwnership
_transferOwnership
owner
<Constructor>
renounceOwnership
onlyOwner
_msgSender
SafeMath (lib)
add
sub
mul
div
mod
min
sqrt
amount
add
BEP20
<Constructor>
_store
name
decimals
symbol
totalSupply
preMineSupply
maxSupply
balanceOf
transfer
allowance
decreaseAllowance
increaseAllowance
transferFrom
approve
_burnFrom
getOwner
owner
_transfer
_approve
_burn
IBEP20 (iface)
totalSupply
preMineSupply
maxSupply
decimals
symbol
name
getOwner
balanceOf
transfer
allowance
approve
transferFrom
_totalSupply
add
sub
ZerrwToken
delegates
delegate
delegateBySig
getCurrentVotes
getPriorVotes
onlyMasterChef
addMasterChef
delMasterchef
treasury
_delegate
name
getChainId
isMasterChef
_moveDelegates
balanceOf
_writeCheckpoint
safe32
srcRepOld
sub
dstRepOld
add
Context
<Constructor>
_msgSender
_msgData
EnumerableSet (lib)
contains
add
remove
length
at
_contains
_add
_remove
_length
_at

# Summary

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