



Cyberscope

# Audit Report

## **Voucher**

December 2024

Network     BASE

Address     0xc1709720bE448D8c0C829D3Ab1A4D661E94f327a

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
●	L09	Dead Code Elimination	Unresolved
●	L13	Divide before Multiply Operation	Unresolved
●	L17	Usage of Solidity Assembly	Unresolved
●	L18	Multiple Pragma Directives	Unresolved
●	L19	Stable Compiler Version	Unresolved

# Table of Contents

<b>Analysis</b>	<b>1</b>
<b>Diagnostics</b>	<b>2</b>
<b>Table of Contents</b>	<b>3</b>
<b>Risk Classification</b>	<b>4</b>
<b>Review</b>	<b>5</b>
Audit Updates	5
Source Files	5
<b>Findings Breakdown</b>	<b>6</b>
L09 - Dead Code Elimination	7
Description	7
Recommendation	8
L13 - Divide before Multiply Operation	9
Description	9
Recommendation	9
L17 - Usage of Solidity Assembly	10
Description	10
Recommendation	10
L18 - Multiple Pragma Directives	11
Description	11
Recommendation	11
L19 - Stable Compiler Version	12
Description	12
Recommendation	12
<b>Functions Analysis</b>	<b>13</b>
<b>Inheritance Graph</b>	<b>14</b>
<b>Flow Graph</b>	<b>15</b>
<b>Summary</b>	<b>16</b>
<b>Disclaimer</b>	<b>17</b>
<b>About Cyberscope</b>	<b>18</b>

## Risk Classification

The criticality of findings in Cyberscope's smart contract audits is determined by evaluating multiple variables. The two primary variables are:

1. **Likelihood of Exploitation:** This considers how easily an attack can be executed, including the economic feasibility for an attacker.
2. **Impact of Exploitation:** This assesses the potential consequences of an attack, particularly in terms of the loss of funds or disruption to the contract's functionality.

Based on these variables, findings are categorized into the following severity levels:

1. **Critical:** Indicates a vulnerability that is both highly likely to be exploited and can result in significant fund loss or severe disruption. Immediate action is required to address these issues.
2. **Medium:** Refers to vulnerabilities that are either less likely to be exploited or would have a moderate impact if exploited. These issues should be addressed in due course to ensure overall contract security.
3. **Minor:** Involves vulnerabilities that are unlikely to be exploited and would have a minor impact. These findings should still be considered for resolution to maintain best practices in security.
4. **Informative:** Points out potential improvements or informational notes that do not pose an immediate risk. Addressing these can enhance the overall quality and robustness of the contract.

Severity	Likelihood / Impact of Exploitation
● Critical	Highly Likely / High Impact
● Medium	Less Likely / High Impact or Highly Likely/ Lower Impact
● Minor / Informative	Unlikely / Low to no Impact

## Review

Contract Name	Voucher
Compiler Version	v0.8.19+commit.7dd6d404
Optimization	1000 runs
Explorer	<a href="https://basescan.org/address/0xc1709720be448d8c0c829d3ab1a4d661e94f327a">https://basescan.org/address/0xc1709720be448d8c0c829d3ab1a4d661e94f327a</a>
Address	0xc1709720be448d8c0c829d3ab1a4d661e94f327a
Network	BASE
Symbol	vSCI
Decimals	18
Total Supply	18,910,000
Badge Eligibility	Yes

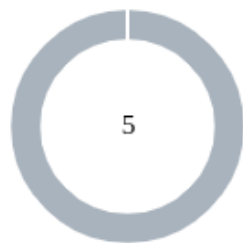
## Audit Updates

Initial Audit	30 Nov 2024
---------------	-------------

## Source Files

Filename	SHA256
Voucher.sol	73c0685eb6ff347d21588bc1e584c06c617848d245f77ca10e255f7939257fa4

## 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

## L09 - Dead Code Elimination

Criticality	Minor / Informative
Location	Voucher.sol#L209,216,224,235,245,330,348,384,395,437,448,486,499,529,555,580,814,823
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 max(uint256 a, uint256 b) internal pure returns
(uint256) {
    return a > b ? a : b;
}

function min(uint256 a, uint256 b) internal pure returns
(uint256) {
    return a < b ? a : b;
    ...
    return (a & b) + (a ^ b) / 2;
}

function ceilDiv(uint256 a, uint256 b) internal pure returns
(uint256) {
    // (a + b - 1) / b can overflow on addition, so we
    distribute.
    return a == 0 ? 0 : (a - 1) / b + 1;
}

...
```



## 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.

## L13 - Divide before Multiply Operation

<b>Criticality</b>	Minor / Informative
<b>Location</b>	Voucher.sol#L292,295,307,311,312,313,314,315,316,322
<b>Status</b>	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 precision.

```
prod0 := div(prod0, twos)
result = prod0 * inverse
```

### 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.

## L17 - Usage of Solidity Assembly

Criticality	Minor / Informative
Location	Voucher.sol#L256,561
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 {  
    let mm := mulmod(x, y, not(0))  
    prod0 := mul(x, y)  
    prod1 := sub(sub(mm, prod0), lt(mm, prod0))  
}  
  
assembly {  
    ptr := add(buffer, add(32, length))  
}
```

### 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.

## L18 - Multiple Pragma Directives

<b>Criticality</b>	Minor / Informative
<b>Location</b>	Voucher.sol#L14,106,134,163,194,543,615,864,950,980,1371,1426
<b>Status</b>	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 0.8.19;  
...
```

### 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

<b>Criticality</b>	Minor / Informative
<b>Location</b>	Voucher.sol#L14,106,134,163,194,543,615,864,950,980,1371
<b>Status</b>	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 ^0.8.0;  
...  
pragma solidity 0.8.19;  
...
```

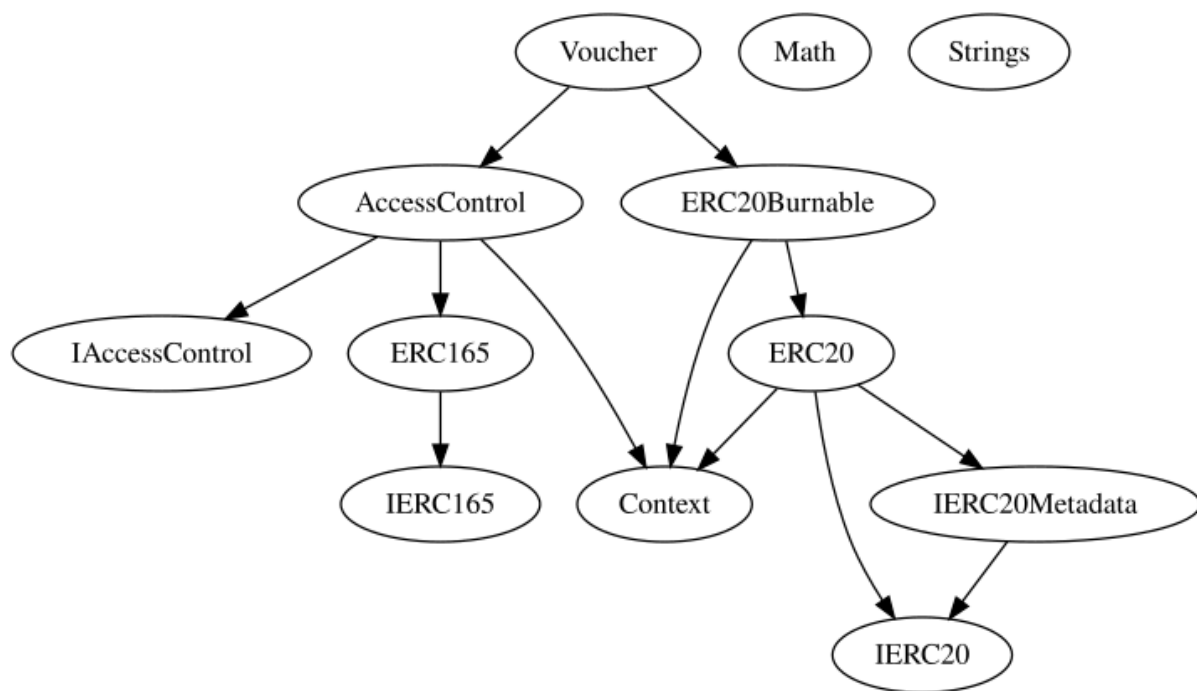
### Recommendation

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.

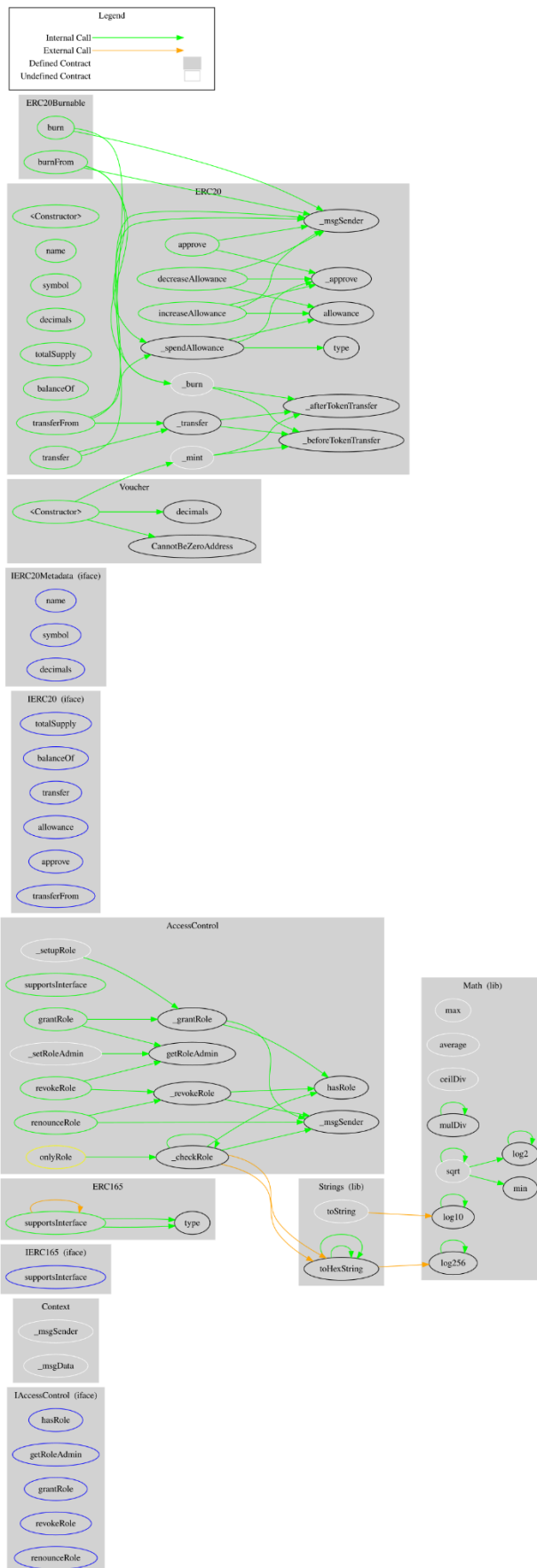
## Functions Analysis

Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
Voucher	Implementation	ERC20Burnable, AccessControl		
		Public	✓	ERC20

## Inheritance Graph



# Flow Graph





## Summary

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

[cyberscope.io](https://cyberscope.io)