Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Customer:	Port Network
Website:	portnetwork.io
Platform:	Ethereum
Language:	Solidity
Date:	June 25th, 2021

Table of contents

Email: audit@EtherAuthority.io

THIS IS SECURITY AUDIT REPORT DOCUMENT AND WHICH MAY CONTAIN INFORMATION WHICH IS CONFIDENTIAL. WHICH INCLUDES ANY POTENTIAL VULNERABILITIES AND MALICIOUS CODES WHICH CAN BE USED TO EXPLOIT THE SOFTWARE. THIS MUST BE REFERRED INTERNALLY AND ONLY SHOULD BE MADE AVAILABLE TO PUBLIC AFTER ISSUES ARE RESOLVED.

Email: audit@EtherAuthority.io

Introduction

We were contracted by the Port Network team to perform the Security audit of the Port Network smart contract code. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on June 25th, 2021.

The purpose of this audit was to address the following:

- Ensure that all claimed functions exist and function correctly.
- Identify any security vulnerabilities that may be present in the smart contract.

Project Background

Port network is a new innovative, high-performance, globally distributed system for cloud and volunteer computing based on a blockchain. It is a viable alternative to expensive cloud services, as well as a cost-effective choice when compared to expensive hardware and their maintenance requirements.

Audit scope

Name	**Code Review and Security Analysis Report for Port Network Smart Contract**
Platform	**Ethereum / Solidity**
File	CMErc20MintBurnPauseSnap.sol
Smart Contract Online Code	https://etherscan.io/address/0xf2c23a2a25ee6c9597b7b70b050871e34eee3385#code
File MD5 Hash	CE42717F15CCDB7B9EA3F2CD7F145D81
Audit Date	June 25th, 2021

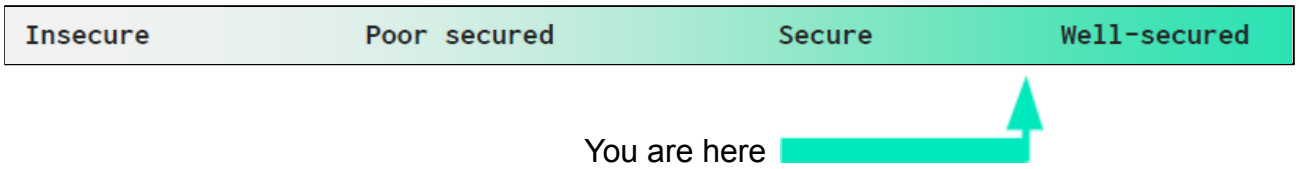
Email: audit@EtherAuthority.io

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
Name: Port Network	**YES, This is valid.**
Symbol: PORT	**YES, This is valid.**
Decimal: 18	**YES, This is valid.**
Token Burning	**YES, This is valid.**
Token Minting	**Admin can set up a minter wallet, who can mint unlimited tokens.**
Token Pause	**Admin can set up a pauser wallet, who can stop/start token transfer.**
Token Snapshot	**Admin can set up a snapshot wallet, who can take balance snapshots.**

Audit Summary

According to the standard audit assessment, Customer`s solidity smart contract is **Technically Secured**. These contracts also have owner functions (described in the centralization section below), which does not make everything 100% decentralized. Thus, the owner must execute those smart contract functions as per the business plan.



We found 0 critical, 0 high, 0 medium and 1 low and some very low level technical issues.

Technical Quick Stats

Main Category	Subcategory	Result
Contract Programming	Solidity version not specified	Passed
	Solidity version too old	Moderated
	Integer overflow/underflow	Passed
	Function input parameters lack of check	Passed
	Function input parameters check bypass	Passed
	Function access control lacks management	Passed
	Critical operation lacks event log	Passed
	Human/contract checks bypass	Passed
	Random number generation/use vulnerability	Passed
	Fallback function misuse	Passed
	Race condition	Passed
	Logical vulnerability	Passed
	Other programming issues	Passed
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Passed
	Other code specification issues	Passed
Gas Optimization	“Out of Gas” Issue	Passed
	High consumption ‘for/while’ loop	Passed
	High consumption ‘storage’ storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage not set	Not Set
	“Short Address” Attack	Passed
	“Double Spend” Attack	Passed

Overall Audit Result: **PASSED**

Code Quality

This audit scope has 1 smart contract. This smart contract also contains Libraries, Smart contracts inherits and Interfaces. This is a compact and well written contract.

The libraries in the Port Network are part of its logical algorithm. A library is a different type of smart contract that contains reusable code. Once deployed on the blockchain (only once), it is assigned a specific address and its properties / methods can be reused many times by other contracts in the Port Network .

The Port Network team has **not** provided scenario and unit test scripts, which would have helped to determine the integrity of the code in an automated way.

Some code parts are **not well** commented on smart contracts.

Documentation

We were given Port Network smart contract code in the form of an Etherscan web link. The hashes of that code are mentioned above in the table.

As mentioned above, some code parts are **not well** commented. So it is difficult to quickly understand the programming flow as well as complex code logic. Comments are very helpful in understanding the overall architecture of the protocol.

Another source of information was its official website https://portnetwork.io/ which provided rich information about the project architecture and tokenomics.

Use of Dependencies

As per our observation, the libraries are used in this smart contract infrastructure that are based on well known industry standard open source projects. And their core code blocks are written well.

Apart from libraries, its functions are not used in external smart contract calls.

AS-IS overview

Port Network is a smart contract, having functionality like mint, burn, etc.

CMErc20MintBurnPauseSnap.sol

(1) Interface

(a) IERC20

(2) Inherited contracts

(a) Context

(b) AccessControl

(c) ERC20Burnable

(d) Ownable

(e) ERC20

(f) Pausable

(g) ERC20Pausable

(h) CMERC20Mintable

(i) CMERC20Pausable

(j) ERC20Snapshot

(k) CMERC20Snapshot

(l) CMErc20MintBurnPauseSnap

(m) CM

(3) Struct

(a) RoleData

(b) Snapshots

(4) Usages

(a) using SafeMath for uint256;

(b) using EnumerableSet for EnumerableSet.AddressSet;

(c) using Address for address;

(d) using SafeMath for uint256;

(e) using Arrays for uint256[];

(f) using Counters for Counters.Counter;

(5) Events

(a) event OwnershipTransferred(address indexed previousOwner, address indexed newOwner);

(b) event Transfer(address indexed from, address indexed to, uint256 value);

(c) event Approval(address indexed owner, address indexed spender, uint256 value);

(d) event RoleAdminChanged(bytes32 indexed role, bytes32 indexed previousAdminRole, bytes32 indexed newAdminRole);

(e) event RoleGranted(bytes32 indexed role, address indexed account, address indexed sender);

(f) event RoleRevoked(bytes32 indexed role, address indexed account, address indexed sender);

(g) event Paused(address account);

(h) event Unpaused(address account);

(i) event Snapshot(uint256 id);

(6) Functions

Sl.	Functions	Type	Observation	Conclusion
1	_beforeTokenTransfer	internal	Passed	No Issue
2	_burn	internal	Passed	No Issue
3	_mint	internal	No Max Minting	Refer Audit Finding Section
4	_transfer	internal	Passed	No Issue
5	snapshot	write	Passed	No Issue
6	_snapshot	internal	Passed	No Issue
7	balanceOfAt	read	Passed	No Issue
8	totalSupplyAt	read	Passed	No Issue
9	_transfer	internal	Passed	No Issue
10	_mint	internal	Passed	No Issue
11	_burn	internal	Passed	No Issue
12	_valueAt	read	Passed	No Issue
13	_updateAccountSnapshot	write	Passed	No Issue
14	_updateTotalSupplySnapshot	write	Passed	No Issue
15	_updateSnapshot	write	Passed	No Issue
16	_lastSnapshotId	read	Passed	No Issue
17	pause	write	Passed	No Issue
18	unpause	write	Passed	No Issue
19	mint	write	Passed	No Issue

20	_beforeTokenTransfer	internal	Passed	No Issue
21	paused	read	Passed	No Issue
22	whenNotPaused	modifier	Passed	No Issue
23	whenPaused	modifier	Passed	No Issue
24	_pause	internal	Passed	No Issue
25	_unpause	internal	Passed	No Issue
26	burn	write	Passed	No Issue
27	burnFrom	write	Passed	No Issue
28	hasRole	read	Passed	No Issue
29	getRoleMemberCount	read	Passed	No Issue
30	getRoleMember	read	Passed	No Issue
31	getRoleAdmin	read	Passed	No Issue
32	grantRole	write	Passed	No Issue
33	revokeRole	write	Passed	No Issue
34	renounceRole	write	Passed	No Issue
35	_setupRole	internal	Passed	No Issue
36	_setRoleAdmin	internal	Passed	No Issue
37	_grantRole	write	Passed	No Issue
38	_revokeRole	write	Passed	No Issue
39	name	read	Passed	No Issue
40	symbol	read	Passed	No Issue
41	decimals	read	Passed	No Issue
42	totalSupply	read	Passed	No Issue
43	balanceOf	read	Passed	No Issue
44	transfer	write	Passed	No Issue
45	allowance	read	Passed	No Issue
46	approve	write	Passed	No Issue
47	transferFrom	write	Passed	No Issue
48	increaseAllowance	write	Passed	No Issue
49	decreaseAllowance	write	Passed	No Issue
50	_transfer	internal	Passed	No Issue
51	_mint	internal	Passed	No Issue
52	_burn	internal	Passed	No Issue
53	_approve	internal	Passed	No Issue
54	_beforeTokenTransfer	internal	Passed	No Issue
55	_setupDecimals	internal	Passed	No Issue
56	_setCMImage	write	access only Owner	No Issue
57	_setCMURL	write	access only Owner	No Issue
58	_supportCM	internal	Hardcoded address	Refer Audit Findings
59	owner	read	Passed	No Issue
60	onlyOwner	modifier	Passed	No Issue
61	transferOwnership	write	access only Owner	No Issue
62	renounceOwnership	write	access only Owner	No Issue

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to token loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial functions
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to tokens lose
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have significant impact on execution
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can be ignored.

Audit Findings

Critical

No critical severity vulnerabilities were found.

High

No high severity vulnerabilities were found.

Medium

No Medium severity vulnerabilities were found.

Low

(1) Unlimited token minting:

```
function _mint(address account, uint256 amount) internal virtual {
    require(account != address(0), "ERC20: mint to the zero address");

    _beforeTokenTransfer(address(0), account, amount);

    _totalSupply = _totalSupply.add(amount);
    _balances[account] = _balances[account].add(amount);
    emit Transfer(address(0), account, amount);
}
```

Owner can mint unlimited tokens. This is not good for tokenomics.

Resolution: place some kind of limit on token minting to prevent any possibility of deflation of token value.

Very Low / Discussion / Best practices:

(1) Use latest solidity version:

```
pragma solidity ^0.6.0;
```

Use the latest solidity version while contract deployment to prevent any compiler version level bugs.

Resolution: Please use 0.8.5 which is the latest version.

(2) Hardcoded address:

```
function _supportCM() internal {
    require(msg.value > 45000000000000000 wei);
    payable(0x98035297b70Cc88fbC064340Fa52344308eC8910).transfer(45000000000000000 wei);
    // Thanks for supporting coinmechanics development!
}
```

Fund transfers to the hardcoded address.

Resolution: Owner has to double confirm.

Centralization

This smart contract has some functions which can be executed by Admin (Owner) only. If the admin wallet private key would be compromised, then it would create trouble. Following are Admin functions:

- _setCMImage: Owner can set image.
- transferOwnership: Transfers ownership of the contract to a new account (`newOwner`).
- renounceOwnership: Renouncing ownership will leave the contract without an owner.
- _setCMURL: Owner can set URL.
- Owner can mint unlimited tokens.
- Owner can pause/unpause token transfer.
- Owner can take snapshots of token balances.

Conclusion

We were given a contract code. And we have used all possible tests based on given objects as files. We observed some issues in the smart contracts and those are fixed/acknowledged in the smart contracts. **So it is good to go for the production.**

Since possible test cases can be unlimited for such smart contracts protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan everything.

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. Smart Contract's high level description of functionality was presented in As-is overview section of the report.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

Security state of the reviewed contract, based on standard audit procedure scope, is **"Technically Secured".**

Our Methodology

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our security audits are to improve the quality of systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

Manual Code Review:

In manually reviewing all of the code, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

Vulnerability Analysis:

Our audit techniques included manual code analysis, user interface interaction, and whitebox penetration testing. We look at the project's web site to get a high level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. While we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation.

Documenting Results:

We follow a conservative, transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even though we have not yet verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this we analyze the feasibility of an attack in a live system.

Suggested Solutions:

We search for immediate mitigations that live deployments can take, and finally we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinized by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our report, and before the details are made public.

Disclaimers

EtherAuthority.io Disclaimer

EtherAuthority team has analyzed this smart contract in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment and functionality (performing the intended functions).

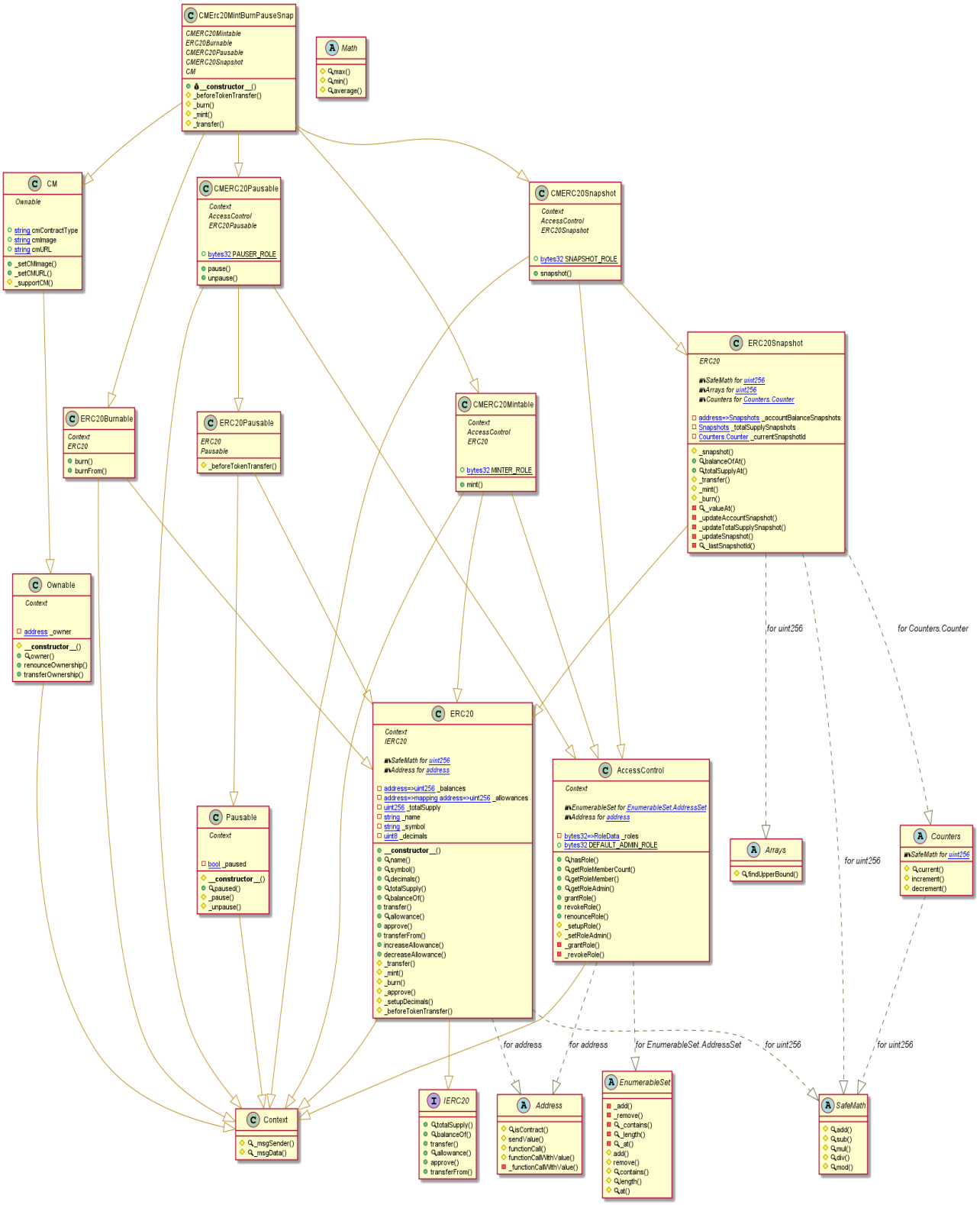
Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only. We also suggest conducting a bug bounty program to confirm the high level of security of this smart contract.

Technical Disclaimer

Smart contracts are deployed and executed on the blockchain platform. The platform, its programming language, and other software related to the smart contract can have their own vulnerabilities that can lead to hacks. Thus, the audit can’t guarantee explicit security of the audited smart contracts.

Appendix

Code Flow Diagram - Port Network



Slither Results Log

Slither log >> CMErc20MintBurnPauseSnap.sol

INFO:Detectors:
ERC20.constructor(string,string).name (CMErc20MintBurnPauseSnap.sol#485) shadows:
- ERC20.name() (CMErc20MintBurnPauseSnap.sol#494-496) (function)
ERC20.constructor(string,string).symbol (CMErc20MintBurnPauseSnap.sol#485) shadows:
- ERC20.symbol() (CMErc20MintBurnPauseSnap.sol#502-504) (function)
CMErc20MintBurnPauseSnap.constructor(string,string,uint256,uint8).name (CMErc20MintBurnPauseSnap.sol#1617) shadows:
- ERC20.name() (CMErc20MintBurnPauseSnap.sol#494-496) (function)
CMErc20MintBurnPauseSnap.constructor(string,string,uint256,uint8).symbol (CMErc20MintBurnPauseSnap.sol#1617) shadows:
- ERC20.symbol() (CMErc20MintBurnPauseSnap.sol#502-504) (function)
CMErc20MintBurnPauseSnap.constructor(string,string,uint256,uint8).decimals (CMErc20MintBurnPauseSnap.sol#1617) shadows:
- ERC20.decimals() (CMErc20MintBurnPauseSnap.sol#519-521) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Address.isContract(address) (CMErc20MintBurnPauseSnap.sol#321-330) uses assembly
- INLINE ASM (CMErc20MintBurnPauseSnap.sol#328)
Address._functionCallWithValue(address,bytes,uint256,string) (CMErc20MintBurnPauseSnap.sol#414-435) uses assembly
- INLINE ASM (CMErc20MintBurnPauseSnap.sol#427-430)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
AccessControl._setRoleAdmin(bytes32,bytes32) (CMErc20MintBurnPauseSnap.sol#1169-1172) is never used and should be removed
Address._functionCallWithValue(address,bytes,uint256,string) (CMErc20MintBurnPauseSnap.sol#414-435) is never used and should be removed
Address.functionCall(address,bytes) (CMErc20MintBurnPauseSnap.sol#374-376) is never used and should be removed
Address.functionCall(address,bytes,string) (CMErc20MintBurnPauseSnap.sol#384-386) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (CMErc20MintBurnPauseSnap.sol#399-401) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (CMErc20MintBurnPauseSnap.sol#409-412) is never used and should be removed
Address.isContract(address) (CMErc20MintBurnPauseSnap.sol#321-330) is never used and should be removed
Address.sendValue(address,uint256) (CMErc20MintBurnPauseSnap.sol#348-354) is never used and should be removed
Context._msgData() (CMErc20MintBurnPauseSnap.sol#10-13) is never used and should be removed
Counters.decrement(Counters.Counter) (CMErc20MintBurnPauseSnap.sol#1448-1450) is never used and should be removed
EnumerableSet.add(EnumerableSet.UintSet,uint256) (CMErc20MintBurnPauseSnap.sol#934-936) is never used and should be removed
EnumerableSet.at(EnumerableSet.UintSet,uint256) (CMErc20MintBurnPauseSnap.sol#972-974) is never used and should be removed
EnumerableSet.contains(EnumerableSet.UintSet,uint256) (CMErc20MintBurnPauseSnap.sol#951-953) is never used and should be removed
EnumerableSet.length(EnumerableSet.UintSet) (CMErc20MintBurnPauseSnap.sol#958-960) is never used and should be removed
EnumerableSet.remove(EnumerableSet.UintSet,uint256) (CMErc20MintBurnPauseSnap.sol#944-946) is never used and should be removed
Math.max(uint256,uint256) (CMErc20MintBurnPauseSnap.sol#1358-1360) is never used and should be removed
Math.min(uint256,uint256) (CMErc20MintBurnPauseSnap.sol#1365-1367) is never used and should be removed

SafeMath.div(uint256,uint256) (CMErc20MintBurnPauseSnap.sol#245-247) is never used and should be removed
SafeMath.div(uint256,uint256,string) (CMErc20MintBurnPauseSnap.sol#261-267) is never used and should be removed
SafeMath.mod(uint256,uint256) (CMErc20MintBurnPauseSnap.sol#281-283) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (CMErc20MintBurnPauseSnap.sol#297-300) is never used and should be removed
SafeMath.mul(uint256,uint256) (CMErc20MintBurnPauseSnap.sol#219-231) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (CMErc20MintBurnPauseSnap.sol#348-354):
- (success) = recipient.call{value: amount}() (CMErc20MintBurnPauseSnap.sol#352)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string) (CMErc20MintBurnPauseSnap.sol#414-435):
- (success,returndata) = target.call{value: weiValue}(data) (CMErc20MintBurnPauseSnap.sol#418)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function CM._setCMImage(string) (CMErc20MintBurnPauseSnap.sol#74-76) is not in mixedCase
Function CM._setCMURL(string) (CMErc20MintBurnPauseSnap.sol#78-80) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (CMErc20MintBurnPauseSnap.sol#11)" inContext (CMErc20MintBurnPauseSnap.sol#5-14)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
Reentrancy in CMErc20MintBurnPauseSnap.constructor(string,string,uint256,uint8) (CMErc20MintBurnPauseSnap.sol#1617-1630):
External calls:
- _supportCM() (CMErc20MintBurnPauseSnap.sol#1618)
- address(0x98035297b70Cc88fbC064340Fa52344308eC8910).transfer(45000000000000000) (CMErc20MintBurnPauseSnap.sol#84)
State variables written after the call(s):
- _mint(msg.sender,amount) (CMErc20MintBurnPauseSnap.sol#1621)
- _balances[account] = _balances[account].add(amount) (CMErc20MintBurnPauseSnap.sol#663)
- _setupDecimals(decimals) (CMErc20MintBurnPauseSnap.sol#1620)
- _decimals = decimals_ (CMErc20MintBurnPauseSnap.sol#717)
- _mint(msg.sender,amount) (CMErc20MintBurnPauseSnap.sol#1621)
- _totalSupply = _totalSupply.add(amount) (CMErc20MintBurnPauseSnap.sol#662)
- cmContractType = CMErc20MintBurnPauseSnap (CMErc20MintBurnPauseSnap.sol#1619)
Event emitted after the call(s):
- RoleGranted(role,account,_msgSender()) (CMErc20MintBurnPauseSnap.sol#1176)
- _setupRole(DEFAULT_ADMIN_ROLE,_msgSender()) (CMErc20MintBurnPauseSnap.sol#1624)
- RoleGranted(role,account,_msgSender()) (CMErc20MintBurnPauseSnap.sol#1176)
- _setupRole(MINTER_ROLE,_msgSender()) (CMErc20MintBurnPauseSnap.sol#1625)
- RoleGranted(role,account,_msgSender()) (CMErc20MintBurnPauseSnap.sol#1176)
- _setupRole(PAUSER_ROLE,_msgSender()) (CMErc20MintBurnPauseSnap.sol#1626)
- RoleGranted(role,account,_msgSender()) (CMErc20MintBurnPauseSnap.sol#1176)
- _setupRole(SNAPSHOT_ROLE,_msgSender()) (CMErc20MintBurnPauseSnap.sol#1627)
- Transfer(address(0),account,amount) (CMErc20MintBurnPauseSnap.sol#664)
- _mint(msg.sender,amount) (CMErc20MintBurnPauseSnap.sol#1621)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4
INFO:Detectors:
CM._supportCM() (CMErc20MintBurnPauseSnap.sol#82-86) uses literals with too many digits:
- require(bool)(msg.value > 45000000000000000) (CMErc20MintBurnPauseSnap.sol#83)
CM._supportCM() (CMErc20MintBurnPauseSnap.sol#82-86) uses literals with too many digits:
- address(0x98035297b70Cc88fbC064340Fa52344308eC8910).transfer(45000000000000000) (CMErc20MintBurnPauseSnap.sol#84)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
owner() should be declared external:
- Ownable.owner() (CMErc20MintBurnPauseSnap.sol#34-36)
renounceOwnership() should be declared external:

- Ownable.renounceOwnership() (CMErc20MintBurnPauseSnap.sol#53-56)
transferOwnership(address) should be declared external:
- Ownable.transferOwnership(address) (CMErc20MintBurnPauseSnap.sol#62-66)
_setCMImage(string) should be declared external:
- CM._setCMImage(string) (CMErc20MintBurnPauseSnap.sol#74-76)
_setCMURL(string) should be declared external:
- CM._setCMURL(string) (CMErc20MintBurnPauseSnap.sol#78-80)
name() should be declared external:
- ERC20.name() (CMErc20MintBurnPauseSnap.sol#494-496)
symbol() should be declared external:
- ERC20.symbol() (CMErc20MintBurnPauseSnap.sol#502-504)
decimals() should be declared external:
- ERC20.decimals() (CMErc20MintBurnPauseSnap.sol#519-521)
transfer(address,uint256) should be declared external:
- ERC20.transfer(address,uint256) (CMErc20MintBurnPauseSnap.sol#545-548)
approve(address,uint256) should be declared external:
- ERC20.approve(address,uint256) (CMErc20MintBurnPauseSnap.sol#564-567)
transferFrom(address,address,uint256) should be declared external:
- ERC20.transferFrom(address,address,uint256) (CMErc20MintBurnPauseSnap.sol#581-585)
increaseAllowance(address,uint256) should be declared external:
- ERC20.increaseAllowance(address,uint256) (CMErc20MintBurnPauseSnap.sol#599-602)
decreaseAllowance(address,uint256) should be declared external:
- ERC20.decreaseAllowance(address,uint256) (CMErc20MintBurnPauseSnap.sol#618-621)
getRoleMemberCount(bytes32) should be declared external:
- AccessControl.getRoleMemberCount(bytes32) (CMErc20MintBurnPauseSnap.sol#1063-1065)
getRoleMember(bytes32,uint256) should be declared external:
- AccessControl.getRoleMember(bytes32,uint256) (CMErc20MintBurnPauseSnap.sol#1079-1081)
getRoleAdmin(bytes32) should be declared external:
- AccessControl.getRoleAdmin(bytes32) (CMErc20MintBurnPauseSnap.sol#1089-1091)
grantRole(bytes32,address) should be declared external:
- AccessControl.grantRole(bytes32,address) (CMErc20MintBurnPauseSnap.sol#1103-1107)
revokeRole(bytes32,address) should be declared external:
- AccessControl.revokeRole(bytes32,address) (CMErc20MintBurnPauseSnap.sol#1118-1122)
renounceRole(bytes32,address) should be declared external:
- AccessControl.renounceRole(bytes32,address) (CMErc20MintBurnPauseSnap.sol#1138-1142)
burn(uint256) should be declared external:
- ERC20Burnable.burn(uint256) (CMErc20MintBurnPauseSnap.sol#1198-1200)
burnFrom(address,uint256) should be declared external:
- ERC20Burnable.burnFrom(address,uint256) (CMErc20MintBurnPauseSnap.sol#1213-1218)
mint(address,uint256) should be declared external:
- CMERC20Mintable.mint(address,uint256) (CMErc20MintBurnPauseSnap.sol#1333-1336)
pause() should be declared external:
- CMERC20Pausable.pause() (CMErc20MintBurnPauseSnap.sol#1343-1346)
unpause() should be declared external:
- CMERC20Pausable.unpause() (CMErc20MintBurnPauseSnap.sol#1348-1351)
balanceOfAt(address,uint256) should be declared external:
- ERC20Snapshot.balanceOfAt(address,uint256) (CMErc20MintBurnPauseSnap.sol#1510-1514)
totalSupplyAt(uint256) should be declared external:
- ERC20Snapshot.totalSupplyAt(uint256) (CMErc20MintBurnPauseSnap.sol#1519-1523)
snapshot() should be declared external:
- CMERC20Snapshot.snapshot() (CMErc20MintBurnPauseSnap.sol#1608-1611)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:CMErc20MintBurnPauseSnap.sol analyzed (20 contracts with 75 detectors), 64 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration

Email: audit@EtherAuthority.io