Cyberscope

Audit Report

Minati

May 2023

Network	BSC
Address	0xe4e11e02aa14c7f24db749421986eaec1369e8c9
Audited by	© cyberscope

Table of Contents

Table of Contents 1
Review 2
Audit Updates 2
Source Files 2
Findings Breakdown 3
Analysis 4
MT - Mints Tokens 5
Description 5
Recommendation 5
Diagnostics 6
L02 - State Variables could be Declared Constant 7
Description 7
Recommendation 7
L04 - Conformance to Solidity Naming Conventions 8
Description 8
Recommendation 8
L09 - Dead Code Elimination 9
Description 9
Recommendation 9
Functions Analysis 10
Inheritance Graph 12
Flow Graph 13
Summary 14
Disclaimer 15
About Cyberscope 16

Review

Contract Name	Minati
Compiler Version	v0.6.9+commit.3e3065ac
Optimization	200 runs
Explorer	https://bscscan.com/address/0xe4e11e02aa14c7f24db749421986eaec1369e8c9
Address	0xe4e11e02aa14c7f24db749421986eaec1369e8c9
Network	BSC
Symbol	MNTC
Decimals	18
Total Supply	12,500,000

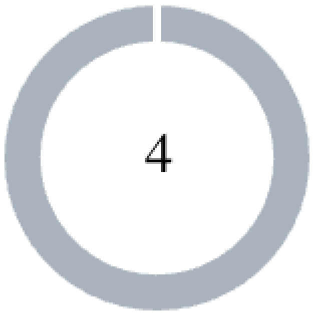
Audit Updates

Initial Audit	13 May 2023

Source Files

Filename	SHA256
Minati.sol	0fcd3225f6e38f4906d065cd4075a3adf8b9947c7a5f0edf9db81f6672eb827e

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

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	Unresolved
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Passed

MT - Mints Tokens

Criticality	Minor / Informative
Location	Minati.sol#L171
Status	Unresolved

Description

The contract owner has the authority to mint tokens. The owner may take advantage of it by calling the mint function. During the audit assessment, the max supply is capped so it cannot mint tokens, but the users are able to burn their own tokens and diverse from the max supply. As a result, the owner will be able to call the mint function up to the equivalent burn amount and the contract tokens will be highly inflated.

```
function mint(address user, uint256 value) external onlyOwner {
    require(maxSupply >= totalSupply.add(value), "Seedx:
MINT_OVERLOAD");
    balances[user] = balances[user].add(value);
    totalSupply = totalSupply.add(value);
    emit Mint(user, value);
    emit Transfer(address(0), user, value);
}
```

Recommendation

The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. Some suggestions are:

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-sign wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.
- Renouncing the ownership will eliminate the threats but it is non-reversible.

Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	L02	State Variables could be Declared Constant	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L09	Dead Code Elimination	Unresolved

L02 - State Variables could be Declared Constant

Criticality	Minor / Informative
Location	Minati.sol#L105,108
Status	Unresolved

Description

State variables can be declared as constant using the constant keyword. This means that the value of the state variable cannot be changed after it has been set. Additionally, the constant variables decrease gas consumption of the corresponding transaction.

```
uint256 public decimals = 18
uint256 public maxSupply = 125e23
```

Recommendation

Constant state variables can be useful when the contract wants to ensure that the value of a state variable cannot be changed by any function in the contract. This can be useful for storing values that are important to the contract's behavior, such as the contract's address or the maximum number of times a certain function can be called. The team is advised to add the constant keyword to state variables that never change.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	Minati.sol#L64,65
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.

```
address public _OWNER_
address public _Future_OWNER_
```

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	Minati.sol#L32,53
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 divCeil(uint256 a, uint256 b) internal pure returns (uint256) {
        uint256 quotient = div(a, b);
        uint256 remainder = a - quotient * b;
        if (remainder > 0) {
            return quotient + 1;
        } else {
            return quotient;
        }
    }

...
```

Recommendation

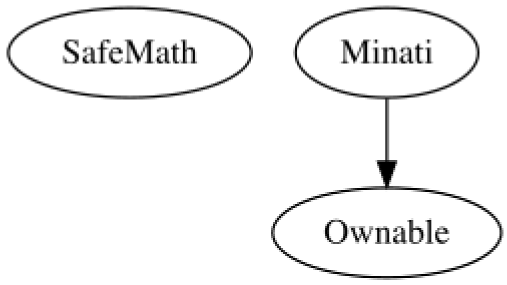
To avoid creating dead code, it's important to carefully consider the logic and flow of the contract and to remove any code that is not needed or that is never executed. This can help improve the clarity and efficiency of the contract.

Functions Analysis

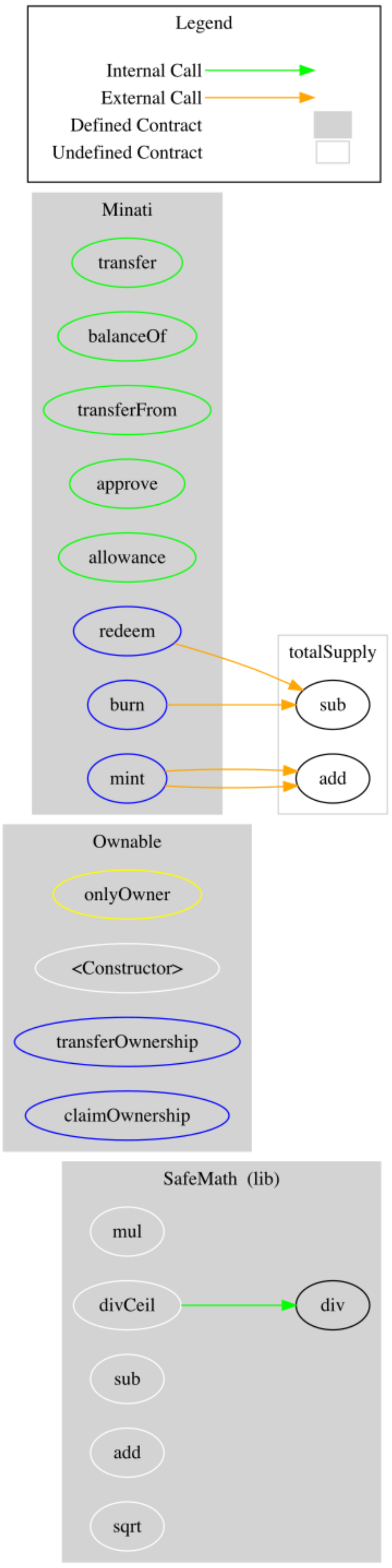
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
SafeMath	Library			
	mul	Internal		
	div	Internal		
	divCeil	Internal		
	sub	Internal		
	add	Internal		
	sqrt	Internal		
Ownable	Implementation			
		Internal	✓	
	transferOwnership	External	✓	onlyOwner
	claimOwnership	External	✓	-
Minati	Implementation	Ownable		
	transfer	Public	✓	-
	balanceOf	Public		-
	transferFrom	Public	✓	-
	approve	Public	✓	-
	allowance	Public		-

	redeem	External	✓	-
	mint	External	✓	onlyOwner
	burn	External	✓	-

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
Minati
transfer
balanceOf
transferFrom
approve
allowance
redeem
burn
mint
totalSupply
sub
add
Ownable
onlyOwner
<Constructor>
transferOwnership
claimOwnership
SafeMath (lib)
mul
divCeil
div
sub
add
sqrt

Summary

Minati contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. There are some functions that can be abused by the owner like mint tokens. if the contract owner abuses the mint functionality, then the contract will be highly inflated. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract or renouncing ownership will eliminate all the contract threats.

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