



Cyberscope

## Audit Report

# Gomining token

October 2024

Network    ETH/BSC

Address    0x7Ddc52c4De30e94Be3A6A0A2b259b2850f421989

Audited by    © cyberscope

# Analysis

● Critical   ● Medium   ● Minor / Informative   ● Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Unresolved
●	OTUT	Transfers User's Tokens	Passed
●	ELFM	Exceeds Fees Limit	Passed
●	MT	Mints Tokens	Unresolved
●	BT	Burns Tokens	Unresolved
●	BC	Blacklists Addresses	Passed

# Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	AOI	Arithmetic Operations Inconsistency	Unresolved
●	RSML	Redundant SafeMath Library	Unresolved
●	L02	State Variables could be Declared Constant	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>7</b>
ST - Stops Transactions	8
Description	8
Recommendation	8
MT - Mints Tokens	9
Description	9
Recommendation	9
BT - Burns Tokens	10
Description	10
Recommendation	10
AOI - Arithmetic Operations Inconsistency	11
Description	11
Recommendation	11
RSML - Redundant SafeMath Library	12
Description	12
Recommendation	12
L02 - State Variables could be Declared Constant	13
Description	13
Recommendation	13
L19 - Stable Compiler Version	14
Description	14
Recommendation	14
<b>Functions Analysis</b>	<b>15</b>
<b>Inheritance Graph</b>	<b>18</b>
<b>Flow Graph</b>	<b>19</b>
<b>Summary</b>	<b>20</b>
<b>Disclaimer</b>	<b>21</b>
<b>About Cyberscope</b>	<b>22</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	GoMiningToken
Compiler Version	v0.8.1+commit.df193b15
Optimization	200 runs
ETH Explorer	<a href="https://etherscan.io/address/0x7ddc52c4de30e94be3a6a0a2b259b2850f421989">https://etherscan.io/address/0x7ddc52c4de30e94be3a6a0a2b259b2850f421989</a>
BSC Explorer	<a href="https://bscscan.com/address/0x7ddc52c4de30e94be3a6a0a2b259b2850f421989">https://bscscan.com/address/0x7ddc52c4de30e94be3a6a0a2b259b2850f421989</a>
Address	0x7ddc52c4de30e94be3a6a0a2b259b2850f421989
Network	ETH/BSC
Symbol	GMT
Decimals	18
ETH Total Supply	352,786,976.375
BSC Total Supply	43,356,446

## Audit Updates

Initial Audit	16 Oct 2024
---------------	-------------

## Source Files

Filename	SHA256
SafeMath.sol	3bf9042f6d35f2cf0389fb8bef53b3ff29d60740a60e92d423798a62ec57cdc9
Pausable.sol	1d08116ec31b3068802b764d44eeb45357b7b4cc56b96a336e25adad718cf828

<b>Ownable.sol</b>	6fda585e8e9903204726fc7447a41a5b25e2f3c52b89a106b581cccb6e7c024e
<b>IERC20Metadata.sol</b>	1f9380710a5a86e156dc3c0feb20e432f75973345a58bee70121d8df89da7c2f
<b>IERC20.sol</b>	5f4e89bc7ee8aeb26b724218151ebe2b5787f2c73b084d3e2b54ef5716223b18
<b>GoMiningToken.sol</b>	d099462f9bd3a7103edc5201f3710230ed1f1c6727abdd3a47ac78ea31c45e98
<b>Context.sol</b>	543c46d0f81fd4e5d9d6a92beef3d2be18badb483b0b4718c819fe3dbbc37587

## Findings Breakdown



Critical	0
Medium	2
Minor / Informative	5

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	0	0	0
Medium	2	0	0	0
Minor / Informative	5	0	0	0



## ST - Stops Transactions

Criticality	Minor / Informative
Location	GoMiningToken.sol#L278,300
Status	Unresolved

### Description

The contract owner has the authority to stop the transactions for all users. The owner may take advantage of it by calling the `pause` function.

```
function pause() public virtual onlyOwner returns (bool) {
    _pause();
    return true;
}

function _beforeTokenTransfer(address from, address to, uint256
amount) internal virtual {
    require(!_paused(), "ERC20Pausable: token transfer while
paused");
}
```

### 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.

Temporary Solutions:

These measurements do not decrease the severity of the finding

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-signature wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.

Permanent Solution:

- Renouncing the ownership, which will eliminate the threats but it is non-reversible.

## MT - Mints Tokens

Criticality	Medium
Location	GoMiningToken.sol#L216
Status	Unresolved

### Description

The contract owner has the authority to mint tokens. The owner may take advantage of it by calling the `mint` function. As a result, the contract tokens will be highly inflated.

```
function mint(address account, uint256 amount) public virtual
onlyOwner {
    _mint(account, amount);
}
```

### 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.

Temporary Solutions:

These measurements do not decrease the severity of the finding

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-signature wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.

Permanent Solution:

- Renouncing the ownership, which will eliminate the threats but it is non-reversible.

## BT - Burns Tokens

Criticality	Medium
Location	GoMiningToken.sol#L240
Status	Unresolved

### Description

The contract owner has the authority to burn tokens from a specific address. The owner may take advantage of it by calling the `burn` function. As a result, the targeted address will lose the corresponding tokens.

```
function burn(address account, uint256 amount) public virtual
onlyOwner {
    _burn(account, amount);
}
```

### 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.

#### Temporary Solutions:

These measurements do not decrease the severity of the finding

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-signature wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.

#### Permanent Solution:

- Renouncing the ownership, which will eliminate the threats but it is non-reversible.

## AOI - Arithmetic Operations Inconsistency

Criticality	Minor / Informative
Location	GoMiningToken.sol#L135,201
Status	Unresolved

### Description

The contract uses both the SafeMath library and native arithmetic operations. The SafeMath library is commonly used to mitigate vulnerabilities related to integer overflow and underflow issues. However, it was observed that the contract also employs native arithmetic operators (such as +, -, \*, /) in certain sections of the code.

The combination of SafeMath library and native arithmetic operations can introduce inconsistencies and undermine the intended safety measures. This discrepancy creates an inconsistency in the contract's arithmetic operations, increasing the risk of unintended consequences such as inconsistency in error handling, or unexpected behavior.

```
_approve(sender, _msgSender(), currentAllowance - amount);  
  
_balances[sender] = senderBalance.sub(amount);  
_balances[recipient] = _balances[recipient].add(amount);
```

### Recommendation

To address this finding and ensure consistency in arithmetic operations, it is recommended to standardize the usage of arithmetic operations throughout the contract. The contract should be modified to either exclusively use SafeMath library functions or entirely rely on native arithmetic operations, depending on the specific requirements and design considerations. This consistency will help maintain the contract's integrity and mitigate potential vulnerabilities arising from inconsistent arithmetic operations.

## RSML - Redundant SafeMath Library

Criticality	Minor / Informative
Location	GoMiningToken.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 to 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 gas consumption unnecessarily in cases where the explanatory error message is not used.

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

### Recommendation

The team is advised to remove the SafeMath library in cases where the revert error message is not used. 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/stable/080-breaking-changes.html#solidity-v0-8-0-breaking-changes>.

## L02 - State Variables could be Declared Constant

<b>Criticality</b>	Minor / Informative
<b>Location</b>	GoMiningToken.sol#L21,22,23
<b>Status</b>	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.

```
string private _name = "GoMining Token"  
string private _symbol = "GMT"  
uint8 private _decimals = 18
```

### 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.

## L19 - Stable Compiler Version

<b>Criticality</b>	Minor / Informative
<b>Location</b>	GoMiningToken.sol#L3
<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;
```

### 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

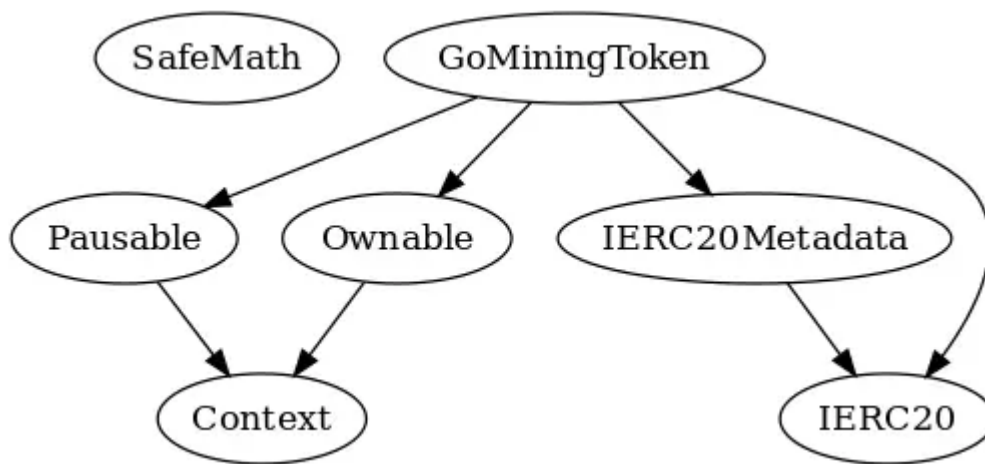
Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
<b>SafeMath</b>	Library			
	tryAdd	Internal		
	trySub	Internal		
	tryMul	Internal		
	tryDiv	Internal		
	tryMod	Internal		
	add	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	mod	Internal		
	sub	Internal		
	div	Internal		
	mod	Internal		
<b>Pausable</b>	Implementation	Context		
		Public	✓	-
	paused	Public		-
	_pause	Internal	✓	whenNotPaused
	_unpause	Internal	✓	whenPaused



<b>Ownable</b>	Implementation	Context		
		Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
<b>IERC20Metadata</b>	Interface	IERC20		
	name	External		-
	symbol	External		-
	decimals	External		-
<b>IERC20</b>	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
<b>GoMiningToken</b>	Implementation	Ownable, Pausable, IERC20, IERC20Meta data		
		Public	✓	-
	name	Public		-

	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	_transfer	Internal	✓	
	mint	Public	✓	onlyOwner
	_mint	Internal	✓	
	burn	Public	✓	onlyOwner
	_burn	Internal	✓	
	_approve	Internal	✓	
	pause	Public	✓	onlyOwner
	unpause	Public	✓	onlyOwner
	_beforeTokenTransfer	Internal	✓	
<b>Context</b>	Implementation			
	_msgSender	Internal		
	_msgData	Internal		

## Inheritance Graph



# Flow Graph



## Summary

Gomining token 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 stop transactions, mint tokens and burn tokens from any address. 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**

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