Audit Report

A51 Finance

December 2023

Repository	https://github.com/a51finance/A51-token
Commit	dc2087a2ae2a569b817145219db718a7eab92bd0
Audited by	© cyberscope

Table of Contents

Table of Contents 1
Review 2
Audit Updates 2
Source Files 2
Findings Breakdown 3
Diagnostics 4
ALM - Array Length Mismatch 5
Description 5
Recommendation 5
L17 - Usage of Solidity Assembly 7
Description 7
Recommendation 7
Functions Analysis 8
Flow Graph 9
Summary 10
Disclaimer 11
About Cyberscope 12

Review

Repository	https://github.com/a51finance/A51-token
Commit	dc2087a2ae2a569b817145219db718a7eab92bd0

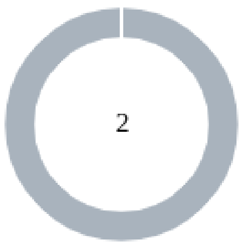
Audit Updates

Initial Audit	27 Dec 2023

Source Files

Filename	SHA256
A51.sol	665e11823c465cbc71e3c7bc7577465aa1efbcc5dd7b9bea102b8216af4ccf7d

Findings Breakdown



●	Critical	0
●	Medium	0
●	Minor / Informative	2

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

Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	ALM	Array Length Mismatch	Unresolved
●	L17	Usage of Solidity Assembly	Unresolved

ALM - Array Length Mismatch

Criticality	Minor / Informative
Location	A51.sol#L39
Status	Unresolved

Description

The contract is currently utilizing the `_mintToWallets` function to mint tokens to a list of accounts. This function is called within the constructor and takes the `accounts` and `amounts` arrays as input. However, there is no verification to ensure that the lengths of the `accounts` and `amounts` arrays are equal. This oversight can lead to potential issues. If the accounts array is longer than the amounts array, the contract will attempt to access an index in the amounts array that does not exist, resulting in a runtime error and transaction failure. Conversely, if the amounts array is longer, some amounts will not be minted to any account, leading to an inconsistency in the intended token distribution.

```
 constructor(address[] memory accounts, uint256[] memory amounts)
ERC20("A51 Finance", "A51") {
    _mintToWallets(accounts, amounts);
  }

  function _mintToWallets(address[] memory _accounts, uint256[] memory
_amounts) internal {
    for (uint256 i = 0; i < _accounts.length; i++) {
      _mint(_accounts[i], _amounts[i]);
    }
  }
```

Recommendation

It is recommended to add an additional check in the `_mintToWallets` function to ensure that the lengths of both the accounts and amounts arrays are the same. This can be implemented as a `require` statement at the beginning of the function, comparing the lengths of the two arrays and reverting the transaction if they do not match. This check will prevent the function from executing if the arrays are not of equal length, thereby avoiding potential runtime errors and ensuring that each account in the accounts array has a corresponding amount in the amounts array. This modification will enhance the robustness and reliability of the token minting process in the contract.

L17 - Usage of Solidity Assembly

Criticality	Minor / Informative
Location	A51.sol#L85
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 {
            chainId := chainid()
        }
```

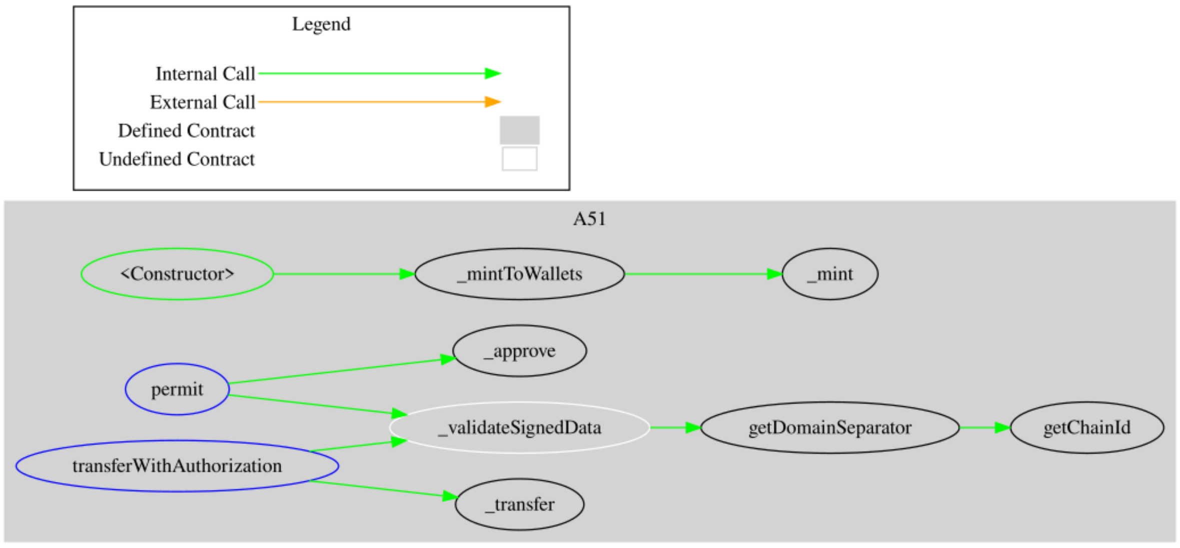
Recommendation

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**
A51	Implementation	ERC20Burnable		
		Public	✓	ERC20
	_mintToWallets	Internal	✓	
	_validateSignedData	Internal		
	permit	External	✓	-
	getDomainSeparator	Public		-
	getChainId	Public		-
	transferWithAuthorization	External	✓	-

Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
A51
<Constructor>
_mintToWallets
_mint
_approve
permit
_validateSignedData
getDomainSeparator
getChainId
transferWithAuthorization
_transfer

Summary

A51 Finance contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. A51 Finance 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

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