Cyberscope

Audit Report

RECORDIAN

October 2023

Network	Pulsechain
Address	0x16d99600412cc70e17ec1c833Ff5bdB77bdbFa25
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
Medium	CSD	Circulating Supply Discrepancy	Unresolved
Minor	IDI	Immutable Declaration Improvement	Unresolved
Minor	L01	Public Function could be Declared External	Unresolved
Minor	L04	Conformance to Solidity Naming Conventions	Unresolved
Minor	L19	Stable Compiler Version	Unresolved

Table of Contents

Review

Contract Name	Recordian
Compiler Version	v0.4.26+commit.4563c3fc
Optimization	200 runs
Explorer	https://scan.v4.testnet.pulsechain.com/address/0x16d99600412cc70e17ec1c833Ff5bdB77bdbFa25
Address	0x16d99600412cc70e17ec1c833Ff5bdB77bdbFa25
Network	Pulsechain
Symbol	RECORD
Decimals	18
Total Supply	3,000,000,000

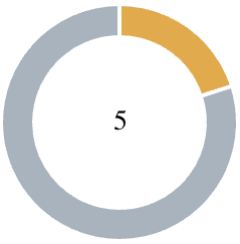
Audit Updates

Initial Audit	11 Oct 2023

Source Files

Filename	SHA256
Recordian.sol	dc21c564874613fa43d76d38e2eb490c5dd692575c2c0255de6cc576904a2e12

Findings Breakdown



Critical	0
Medium	1
Minor / Informative	4

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	0	0	0
Medium	1	0	0	0
Minor / Informative	4	0	0	0

CSD - Circulating Supply Discrepancy

Criticality	Medium
Location	Recordian.sol#L114
Status	Unresolved

Description

According to the ERC20 specification, the `totalSupply()` function should return the total supply of the token. The total supply should always equal the sum of the balances. The contract does not return the `totalSupply()`. Instead, the function returns the `totalSupply()` minus the amount that has been moved to the dead address. This amount is the circulating supply of the token. Many decentralized applications and tools are calculating many indicators like the circulating supply and market cap based on the `totalSupply()`. As a result, these applications will produce misleading results.

```
function totalSupply() public constant returns (uint) {
    return _totalSupply  - balances[address(0)];
}
```

Recommendation

The `totalSupply()` should always equal the sum of the holder's balances. The contract should comply with this convention so that the decentralized applications will produce correct results.

IDI - Immutable Declaration Improvement

Criticality	Minor / Informative
Location	Recordian.sol#L105,106,107,108
Status	Unresolved

Description

The contract declares state variables that their value is initialized once in the constructor and are not modified afterwards. The `immutable` is a special declaration for this kind of state variables that saves gas when it is defined.

```
symbol
name
decimals
_totalSupply
```

Recommendation

By declaring a variable as immutable, the Solidity compiler is able to make certain optimizations. This can reduce the amount of storage and computation required by the contract, and make it more gas-efficient.

L01 - Public Function could be Declared External

Criticality	Minor / Informative
Location	Recordian.sol#L90,146
Status	Unresolved

Description

A public function is a function that can be called from external contracts or from within the contract itself. An external function is a function that can only be called from external contracts, and cannot be called from within the contract itself.

It's generally a good idea to declare functions as external if they are only intended to be called from external contracts, as this can help make the contract's code easier to understand and maintain. Declaring a function as external can also help to improve the contract's performance and gas consumption.

```
function receiveApproval(address from, uint256 tokens, address
token, bytes data) public;

function approveAndCall(address spender, uint tokens, bytes
data) public returns (bool success) {
            allowed[msg.sender][spender] = tokens;
            emit Approval(msg.sender, spender, tokens);

ApproveAndCallFallBack(spender).receiveApproval(msg.sender,
tokens, this, data);
            return true;
        }
```

Recommendation

It's important to choose the appropriate visibility for each function based on how it is intended to be used. Declaring a function as external when it should be public, or vice versa can lead to unnecessary gas consumption.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	Recordian.sol#L99
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.

```
uint public _totalSupply
```

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.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	Recordian.sol#L43
Status	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 ^0.4.24;
```

Recommendation

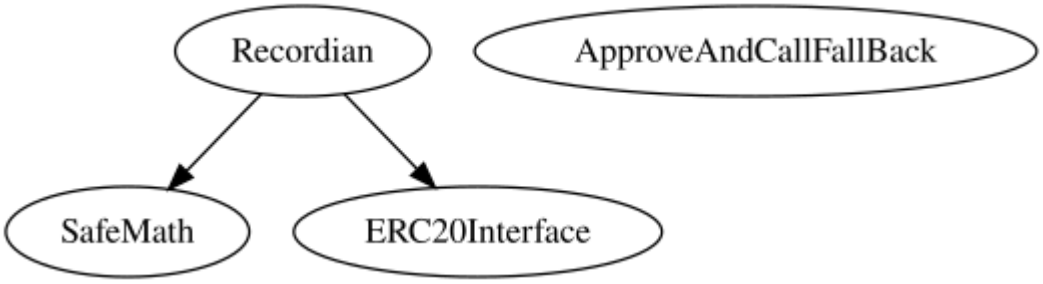
The team is advised to lock the pragma to ensure the stability of the codebase. The locked pragma version ensures that the contract will not be deployed with an unexpected version. An unexpected version may produce vulnerabilities and undiscovered bugs. The compiler should be configured to the lowest 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

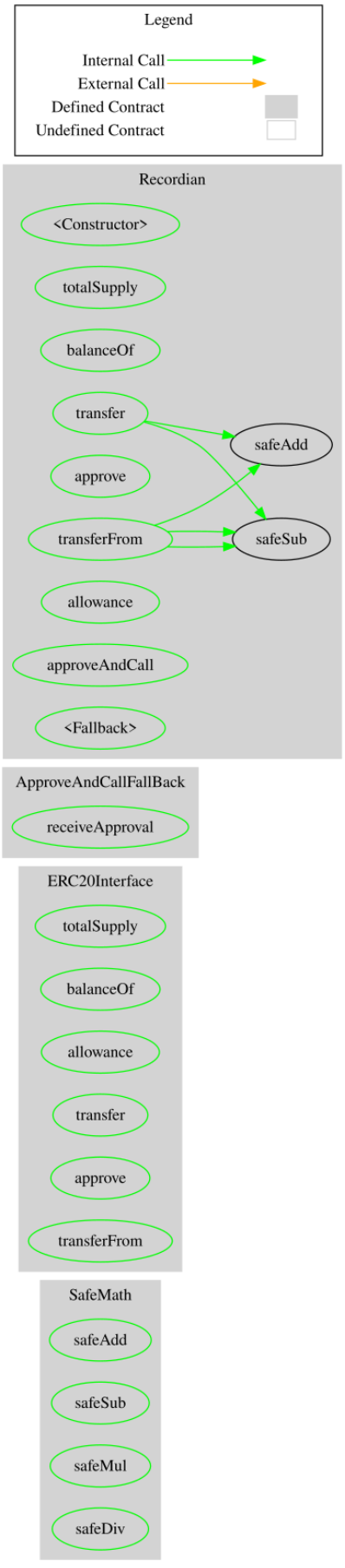
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
SafeMath	Implementation			
	safeAdd	Public		-
	safeSub	Public		-
	safeMul	Public		-
	safeDiv	Public		-
ERC20Interface	Implementation			
	totalSupply	Public		-
	balanceOf	Public		-
	allowance	Public		-
	transfer	Public	✓	-
	approve	Public	✓	-
	transferFrom	Public	✓	-
ApproveAndCallFallBack	Implementation			
	receiveApproval	Public	✓	-

Recordian	Implementation	ERC20Interface, SafeMath		
		Public	✓	-
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	allowance	Public		-
	approveAndCall	Public	✓	-
		Public	Payable	-

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
Recordian
<Constructor>
totalSupply
balanceOf
transfer
safeAdd
approve
transferFrom
safeSub
allowance
approveAndCall
<Fallback>
ApproveAndCallFallBack
receiveApproval
ERC20Interface
totalSupply
balanceOf
allowance
transfer
approve
transferFrom
SafeMath
safeAdd
safeSub
safeMul
safeDiv

Summary

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

https://www.cyberscope.io