Cyberscope

Audit Report

DeFi Kingdoms Crystal

January 2023

Network	AVAX DFK Subnet
Address	0x04b9dA42306B023f3572e106B11D82aAd9D32EBb
Audited by	© cyberscope

Table of Contents

Review

Contract Name	CrystalToken
Symbol	CRYSTAL
Compiler Version	0.8.6+commit.11564f7e
EVM Version	berlin
Optimization	200 runs
Max Supply Cap	125,000,000
Testing Deploy	https://testnet.bscscan.com/address/0x687d7a1a6ff81974ed101557b98030d77e905a0d
Explorer	https://subnets.avax.network/defi-kingdoms/address/0x57Dec9cC7f492d6583c773e2E7ad66dcDc6940Fb

Audit Updates

Initial Audit	21 Jan 2023

Source Files

Filename	SHA256
@openzeppelin/contracts/access/Ownable.sol	9353af89436556f7ba8abb3f37a667724 9aa4df6024fbfaa94f79ab2f44f3231
@openzeppelin/contracts/token/ERC20/ERC20.sol	5031430cc2613c32736d598037d30759 85a2a09e61592a013dbd09a5bc2041b8
@openzeppelin/contracts/token/ERC20/extensions /IERC20Metadata.sol	af5c8a77965cc82c33b7ff844deb982616 6689e55dc037a7f2f790d057811990
@openzeppelin/contracts/token/ERC20/IERC20.sol	94f23e4af51a18c2269b355b8c7cf4db8 003d075c9c541019eb8dcf4122864d5
@openzeppelin/contracts/utils/Context.sol	1458c260d010a08e4c20a4a517882259 a23a4baa0b5bd9add9fb6d6a1549814a
@openzeppelin/contracts/utils/math/SafeMath.sol	0dc33698a1661b22981abad8e5c6f5ebc a0dfe5ec14916369a2935d888ff257a
contracts/testingDeploy/Authorizable.sol	50dfdcedf5b8fc6eeac08ade1adbe0823 4bc44dcb991bb01d12d15db796d6822
contracts/testingDeploy/CrystalToken.sol	6ded53e07f2dd84204df60c678907f06b 557d656b62ec4a15627957b98afba56

Introduction

The Crystal contract implements a standard ERC20 token enriched with mint and time locker functionality.

The authorized role can lock the tokens from any user. The time period is determined by the authorized role. During the lock phase, the user's tokens are transferred to the contract address. The locked tokens can be unlocked by the user proportionally to the time period that has elapsed.

The contract offers a `transferAll()` method where any of the users can transfer all their balance to another user. If the sender has locked tokens, then the tokens will be transferred to the recipient's locker balance.

Roles

There are two roles in the contract, the owner and the authorized.

Owner

- mint()

Authorized

- transferAllIntervalUpdate()
- lockFromUpdate()
- lockToUpdate()
- manualMint()
- lock()
- unlockForUser()
- updateMaxTransferAmountRate()
- setExcludedFromAntiWhale()

Public

- unlock()
- transferAll()

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
Critical	MLP	Misleading Lock Period	Unresolved
Medium	VVS	Validate Variable Setters	Unresolved
Minor / Informative	MT	Mints Tokens	Unresolved
Minor / Informative	L07	Missing Events Arithmetic	Unresolved
Minor / Informative	L19	Stable Compiler Version	Unresolved

MLP - Misleading Lock Period

Criticality	Critical
Location	contracts/testingDeploy/CrystalToken.sol#L224
Status	Unresolved

Description

The contract uses the `lastUnlockTime[address]` variable to determine when the locker will be released for each address. In the method that calculates the duration that has elapsed, the contract subtracts the `lastUnlockTime[address]` from the current timestamp. The algorithm assumes that it subtracts two timestamps.

```
block.timestamp.sub(lastUnlockTime[_holder]);
```

In the `_unlock()` method the `lastUnlockTime[address]` variable is updated with the `block.number`.

```
lastUnlockTime[holder] = block.number;
```

The `block.number`. variable returns the current block number, which is a sequential number assigned to each block in the blockchain. The "block.timestamp" variable, on the other hand, returns the timestamp of the current block, which is a value representing the number of seconds since the Unix epoch (January 1, 1970) at the time the block was mined.

The `block.number`. can be used to determine relative time between blocks, but the `block.timestamp`. can be used to determine the actual time at which a block was mined.

Recommendation

The team is advised to use the same unit across the calculations since the difference between the block number and the current timestamp may produce unexpected behavior.

VVS - Validate Variable Setters

Criticality	Medium
Status	Unresolved

Description

The contract does not ensure that the mutation of the state variables will not produce unexpected results. Assume the following scenario.

1. The user A executes the `lock()` method.

```
lockToTime = x
lockFromTime = x - 2 days
lastUnlockTime[_holder] = x - 2 days
```

2. An authorized address changes the lockFromTime.

`lockFromTime = x - 4 days`

3. At the time period `x - 1 day` The user A executes the `unlock()` method. The following expression will underflow.

```
block.timestamp.sub(lastUnlockTime[_holder]); ->
(x - 1 day) - (x - 2 days)
```

Recommendation

The team is advised to sanitize the setters variable so that it will deterministically ensure that the contract will not lead to an unexpected state.

Special handling should be considered on the mutation of the `lockToTime` and `lockFromTime` variables, so that the `lastUnlockTime` will not lead to an unexpected state.

- The `lockToTime` should be greater than `lockFromTime`.
- The `transferAllInterval` should be less than the current timestamp.

MT - Mints Tokens

Criticality	Minor / Informative
Location	contracts/testingDeploy/CrystalToken.sol#L146,151
Status	Unresolved

Description

The contract owner and the authorized addresses have the authority to mint tokens. If the minting process is abused then the contract tokens will be highly inflated.

- The owner may take advantage of it by calling the `mint` function.
- The authorized addresses may take advantage of it by calling the `manualMint` function. The manual mint can mint up to an upper threshold.

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

function manualMint(address _to, uint256 _amount) public onlyAuthorized {
    require(manualMinted < manualMintLimit, "ERC20: manualMinted greater than
manualMintLimit");
    _mint(_to, _amount);
    manualMinted = manualMinted.add(_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. That risk can be prevented by temporarily locking the contract or renouncing ownership.

L07 - Missing Events Arithmetic

Criticality	Minor / Informative
Location	contracts/testingDeploy/CrystalToken.sol#L88,93
Status	Unresolved

Description

Events are a way to record and log information about changes or actions that occur within a contract. They are often used to notify external parties or clients about events that have occurred within the contract, such as the transfer of tokens or the completion of a task.

It's important to carefully design and implement the events in a contract, and to ensure that all required events are included. It's also a good idea to test the contract to ensure that all events are being properly triggered and logged.

```
lockFromTime = _lockFromTime
lockToTime = _lockToTime
```

Recommendation

By including all required events in the contract and thoroughly testing the contract's functionality, the contract ensures that it performs as intended and does not have any missing events that could cause issues with its arithmetic.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	contracts/testingDeploy/CrystalToken.sol#L3
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 ^0.8.0;
```

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

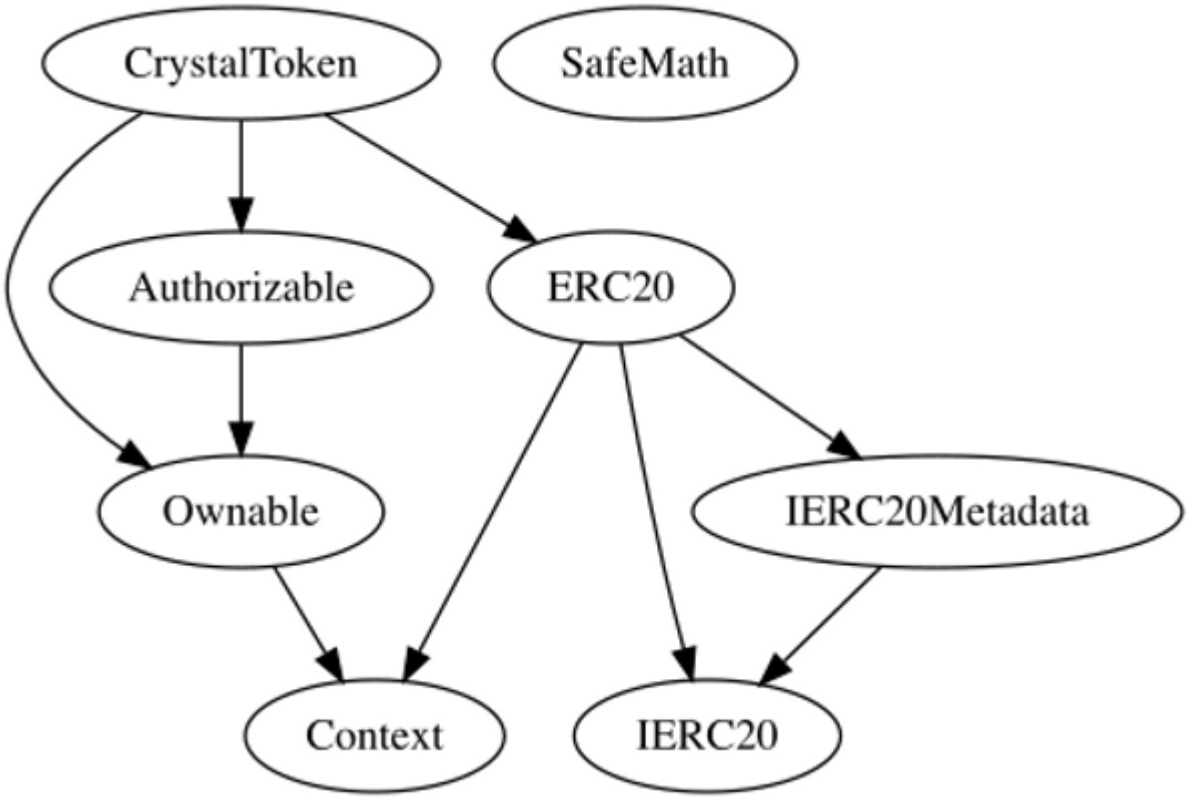
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	_checkOwner	Internal		
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	
ERC20	Implementation	Context, IERC20, IERC20Metadata		
		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	Internal	✓	
	_burn	Internal	✓	
	_approve	Internal	✓	
	_spendAllowance	Internal	✓	
	_beforeTokenTransfer	Internal	✓	
	_afterTokenTransfer	Internal	✓	
IERC20Metadata	Interface	IERC20		
	name	External		-
	symbol	External		-
	decimals	External		-
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
SafeMath	Library			
	tryAdd	Internal		
	trySub	Internal		
	tryMul	Internal		
	tryDiv	Internal		

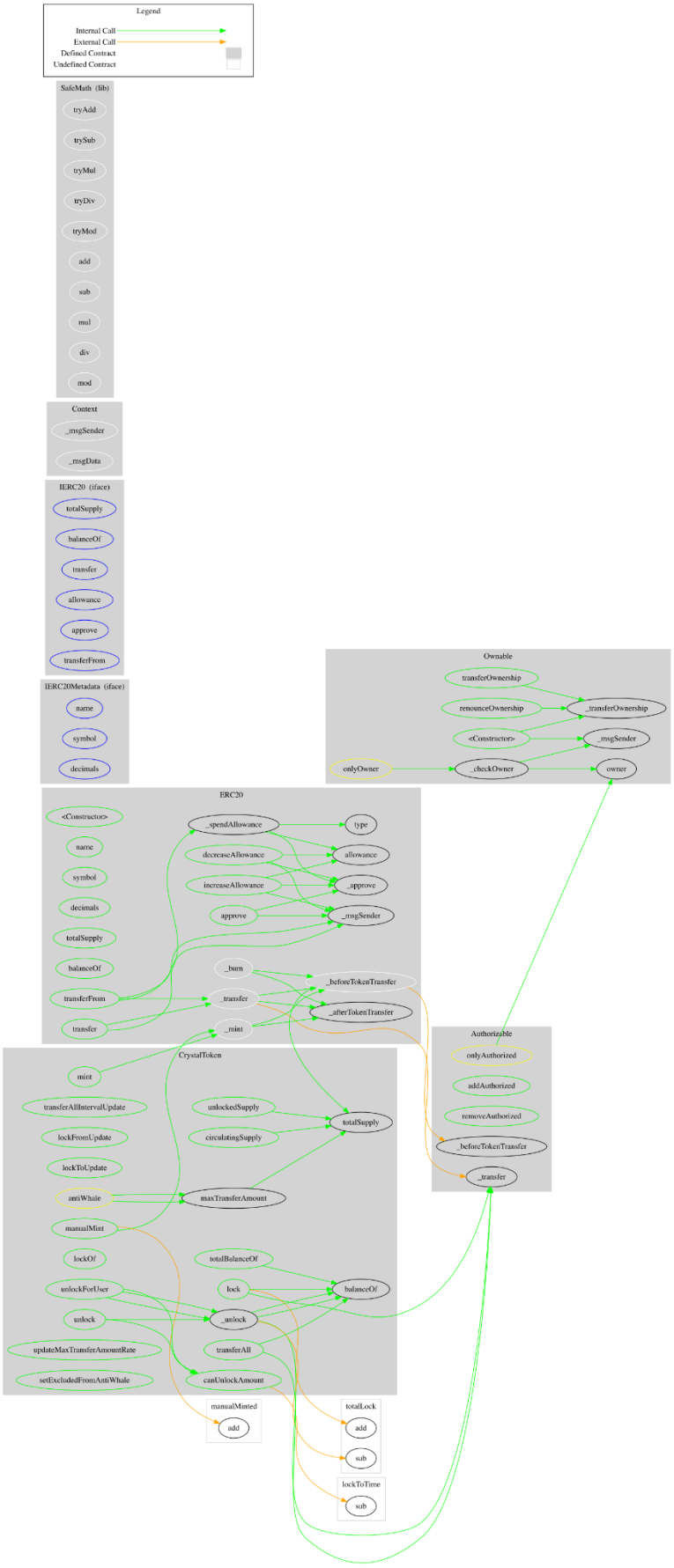
	tryMod	Internal		
	add	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	mod	Internal		
	sub	Internal		
	div	Internal		
	mod	Internal		
Authorizable	Implementation	Ownable		
	addAuthorized	Public	✓	onlyOwner
	removeAuthorized	Public	✓	onlyOwner
CrystalToken	Implementation	ERC20, Ownable, Authorizable		
		Public	✓	ERC20
	transferAllIntervalUpdate	Public	✓	onlyAuthorized
	lockFromUpdate	Public	✓	onlyAuthorized
	lockToUpdate	Public	✓	onlyAuthorized
	unlockedSupply	Public		-
	circulatingSupply	Public		-
	_beforeTokenTransfer	Internal	✓	
	_transfer	Internal	✓	antiWhale
	mint	Public	✓	onlyOwner
	manualMint	Public	✓	onlyAuthorized
	totalBalanceOf	Public		-
	lockOf	Public		-
	lock	Public	✓	onlyAuthorized
	canUnlockAmount	Public		-

	unlockForUser	Public	✓	onlyAuthorized
	unlock	Public	✓	-
	_unlock	Internal	✓	
	transferAll	Public	✓	-
	updateMaxTransferAmountRate	Public	✓	onlyAuthorized
	maxTransferAmount	Public		-
	setExcludedFromAntiWhale	Public	✓	onlyAuthorized

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
SafeMath (lib)
tryAdd
trySub
tryMul
tryDiv
tryMod
add
sub
mul
div
mod
Context
_msgSender
_msgData
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
IERC20Metadata (iface)
name
symbol
decimals
Ownable
transferOwnership
renounceOwnership
<Constructor>
onlyOwner
_transferOwnership
_msgSender
_checkOwner
owner
ERC20
<Constructor>
name
symbol
decimals
totalSupply
balanceOf
transferFrom
transfer
_spendAllowance
decreaseAllowance
increaseAllowance
approve
_burn
_transfer
_mint
type
allowance
_approve
_msgSender
_beforeTokenTransfer
_afterTokenTransfer
Authorizable
onlyAuthorized
addAuthorized
removeAuthorized
_beforeTokenTransfer
_transfer
CrystalToken
mint
transferAllIntervalUpdate
lockFromUpdate
lockToUpdate
antiWhale
manualMint
lockOf
unlockForUser
unlock
updateMaxTransferAmountRate
setExcludedFromAntiWhale
unlockedSupply
circulatingSupply
totalSupply
maxTransferAmount
totalBalanceOf
lock
_unlock
transferAll
canUnlockAmount
balanceOf
manualMinted
add
totalLock
add
sub
lockToTime
sub

Summary

DeFi Kingdoms Crystal contract implements an enriched ERC20 token mechanism. This audit investigates security issues, business logic concerns and potential improvements.

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