Cyberscope

Audit Report

Eczodex

April 2024

Network SEPOLIA

Address 0xfcF54e379174736875592bc1ED48448610734Cc5e

Audited by © cyberscope

Table of Contents

Review

Explorer	https://sepolia.etherscan.io/address/0xfcf54e37917473687592bc1ed48448610734cc5e
Network	Sepolia

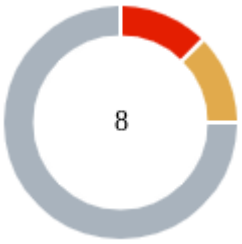
Audit Updates

Initial Audit	25 Apr 2024

Source Files

Filename	SHA256
EczodexUSD.sol	22eca9020a0b6f07db5a98b672a43d1a6f13d26011a3acaea493f44134599b8b

Findings Breakdown



Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	1	0	0
Medium	0	1	0	0
Minor / Informative	0	6	0	0

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
	MT	Mints Tokens	Acknowledged
	BC	Blacklists Addresses	Acknowledged
	RBC	Redundant Blacklist Checks	Acknowledged
	RMD	Redundant Minter Designation	Acknowledged
	RSW	Redundant Storage Writes	Acknowledged
	ST	Stops Transactions	Acknowledged
	L04	Conformance to Solidity Naming Conventions	Acknowledged
	L19	Stable Compiler Version	Acknowledged

MT - Mints Tokens

Criticality	Critical
Location	EczodexUSD.sol#L105
Status	Acknowledged

Description

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

```
function mint(uint256 amount) public onlyRole(MINTER_ROLE) {
    require(
        debtCeiling >= totalSupply() + amount,
        "Minting would exceed the debt ceiling"
    );
    address designatedMinter = _minterDesignations[msg.sender];
    _mint(designatedMinter, amount);
    emit Minted(msg.sender, amount);
}
```

Recommendation

The team should carefully manage the private keys of the MINTER'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.

Team Update

The team has acknowledged that this is not a security issue and states:

Eczodex, a Techstars-backed fintech, has deployed a stablecoin smart contract for issuance and redemption under a regulated US Money Service Business (MSB) license. In addition, the MSB has established custodian agreements with several Insured Depository Institutions providing bankruptcy remote protection to customer collateral. Consequently, our operating model offers customers the same regulatory protections as Circle's USDC and adheres to the Bank Secrecy Act. Underpinning our centralized issuance model is the Dfns MPC wallet solution.

Dfns is the leading wallet-as-a-service platform in web3. Startups, enterprises and financial institutions use Dfns to create, embed and manage programmable wallets at scale powered by the fastest, most advanced MPC technology in the world. Built by PhDs and experts in security and cryptography, their team is spread across the US and EU. Since 2020, Dfns has helped ABN AMRO, Fidelity, Zodia and many others to create over a million wallets. Their platform is SOC2 certified.

Dfns is pioneering research in state-of-the-art cryptography such as MPC (Multi-Party Computation), and has built in threshold recovery mechanisms to guarantee business continuity and fallback options. Dfns also offers a range of cryptographic protocols, wallet toolkits and authentication systems for developers building automated workflows within bank-grade compliance frameworks.

In summary, the combination of regulatory licensing and a robust MPC wallet solution maximize customer protection and provide robust guardrails for safe and compliant access to the contract admin functions.

BC - Blacklists Addresses

Criticality	Medium
Location	EczodexUSD.sol#L70
Status	Acknowledged

Description

The BLACKLISTER role has the authority to stop addresses from transactions. The BLACKLISTER address may take advantage of it by calling the `blacklist` function.

```
function blacklist(address account) public onlyRole(BLACKLISTER_ROLE) {
    isBlacklisted[account] = true;
    emit Blacklisted(account);
}
```

Recommendation

The team should carefully manage the private keys of the BLACKLISTER'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.

Team Update

The team has acknowledged that this is not a security issue and states:

The BSA requires regulated US financial institutions to implement stringent AML and CTF transaction monitoring and reporting controls. The regulatory enforcement agencies FinCEN and OFAC may, on request, instruct institutions including regulated stablecoin issuers to freeze assets or prohibit transactions with sanctioned entities or individuals. The blacklist function is an effective compliance tool and has been successfully implemented by issuers such as Circle in their smart contracts.

RBC - Redundant Blacklist Checks

Criticality	Minor / Informative
Location	EczodexUSD.sol#L120
Status	Acknowledged

Description

The contract is currently structured to emit an event when a transfer attempt involves a blacklisted address, followed by additional `require` statements that check if either the sender or the receiver is blacklisted. These subsequent checks are redundant as the initial if statement already determines the blacklist status of the involved addresses. This redundancy not only complicates the contract logic but also potentially increases gas costs due to the processing of unnecessary conditions.

```
if (isBlacklisted[from] || isBlacklisted[to]) {
    emit BlacklistedTransferAttempt(from, to, amount);
}
require(!isBlacklisted[from], "Account is blacklisted");
require(!isBlacklisted[to], "Account is blacklisted");
```

Recommendation

It is recommended to streamline the handling of blacklisted addresses by modifying the contract to revert the transaction directly within the initial `if` condition if either the sender or receiver is blacklisted. This adjustment will eliminate the need for subsequent `require` checks, thereby simplifying the contract's code and reducing transactional overhead. Simplifying these checks will enhance contract performance and ensure that operations involving blacklisted addresses are efficiently and effectively halted.

RMD - Redundant Minter Designation

Criticality	Minor / Informative
Location	EczodexUSD.sol#L99,105
Status	Acknowledged

Description

The contract is currently utilizing the `setMinterDesignation` function to assign a destination address for minting tokens, a process exclusively managed by addresses with the `MINTER_ROLE` . However, this role-based address also invokes the `mint` function, making the `setMinterDesignation` function redundant. The destination address for minting could be more efficiently passed directly as a parameter to the `mint` function itself. This redundancy in function calls complicates the contract unnecessarily and could potentially lead to inefficiencies in transaction processing.

```
    function setMinterDesignation(
        address designatedAddress
    ) public onlyRole(MINTER_ROLE) {
        _minterDesignations[msg.sender] = designatedAddress;
    }

    function mint(uint256 amount) public onlyRole(MINTER_ROLE) {
        require(
            debtCeiling >= totalSupply() + amount,
            "Minting would exceed the debt ceiling"
        );
        address designatedMinter = _minterDesignations[msg.sender];
        _mint(designatedMinter, amount);
        emit Minted(msg.sender, amount);
    }
```

Recommendation

It is recommended to refactor the minting process by eliminating the `setMinterDesignation` function and modifying the `mint` function to accept the `destination` address as a parameter. This change would streamline the minting process, reduce the number of transactions required for minting operations, and enhance the overall efficiency and clarity of the contract's functionality. Such an adjustment would

not only simplify the contract's architecture but also align its operations more closely with best practices for smart contract development.

RSW - Redundant Storage Writes

Criticality	Minor / Informative
Location	EczodexUSD.sol#L75,82
Status	Acknowledged

Description

The contract modifies the state of the following variables without checking if their current value is the same as the one given as an argument. As a result, the contract performs redundant storage writes, when the provided parameter matches the current state of the variables, leading to unnecessary gas consumption and inefficiencies in contract execution.

```
    function removeFromBlacklist(
        address account
    ) public onlyRole(BLACKLISTER_ROLE) {
        isBlacklisted[account] = false;
        emit Unblacklisted(account);
    }

    function setDebtCeiling(
        uint256 _debtCeiling
    ) public onlyRole(DEBT_CEILING_ADJUSTER_ROLE) {
        debtCeiling = _debtCeiling;
        emit DebtCeilingAdjusted(msg.sender, debtCeiling);
    }
```

Recommendation

The team is advised to implement additional checks within to prevent redundant storage writes when the provided argument matches the current state of the variables. By incorporating statements to compare the new values with the existing values before proceeding with any state modification, the contract can avoid unnecessary storage operations, thereby optimizing gas usage.

ST - Stops Transactions

Criticality	Minor / Informative
Location	EczodexUSD.sol#L89,94
Status	Acknowledged

Description

The PAUSER role has the authority to stop the sales for all users including the owner. The PAUSER may take advantage of it by calling the `pause` and `upause` functions. As a result, the contract will prevent all transactions.

```
    function pause() public onlyRole(PAUSER_ROLE) {
        _pause();
        emit Paused(msg.sender);
    }

    function unpause() public onlyRole(PAUSER_ROLE) {
        _unpause();
        emit Unpaused(msg.sender);
    }
```

Recommendation

The team should carefully manage the private keys of the PAUSER'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.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	EczodexUSD.sol#L83
Status	Acknowledged

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.

```
uint256 _debtCeiling
```

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	EczodexUSD.sol#L4
Status	Acknowledged

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 ^0.8.9;
```

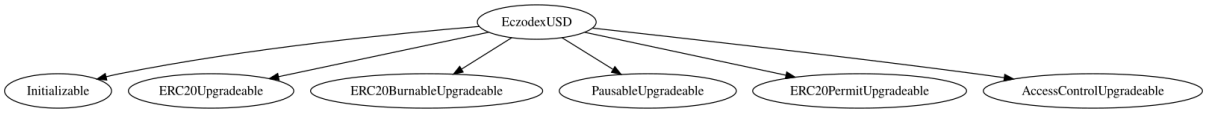
Recommendation

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.

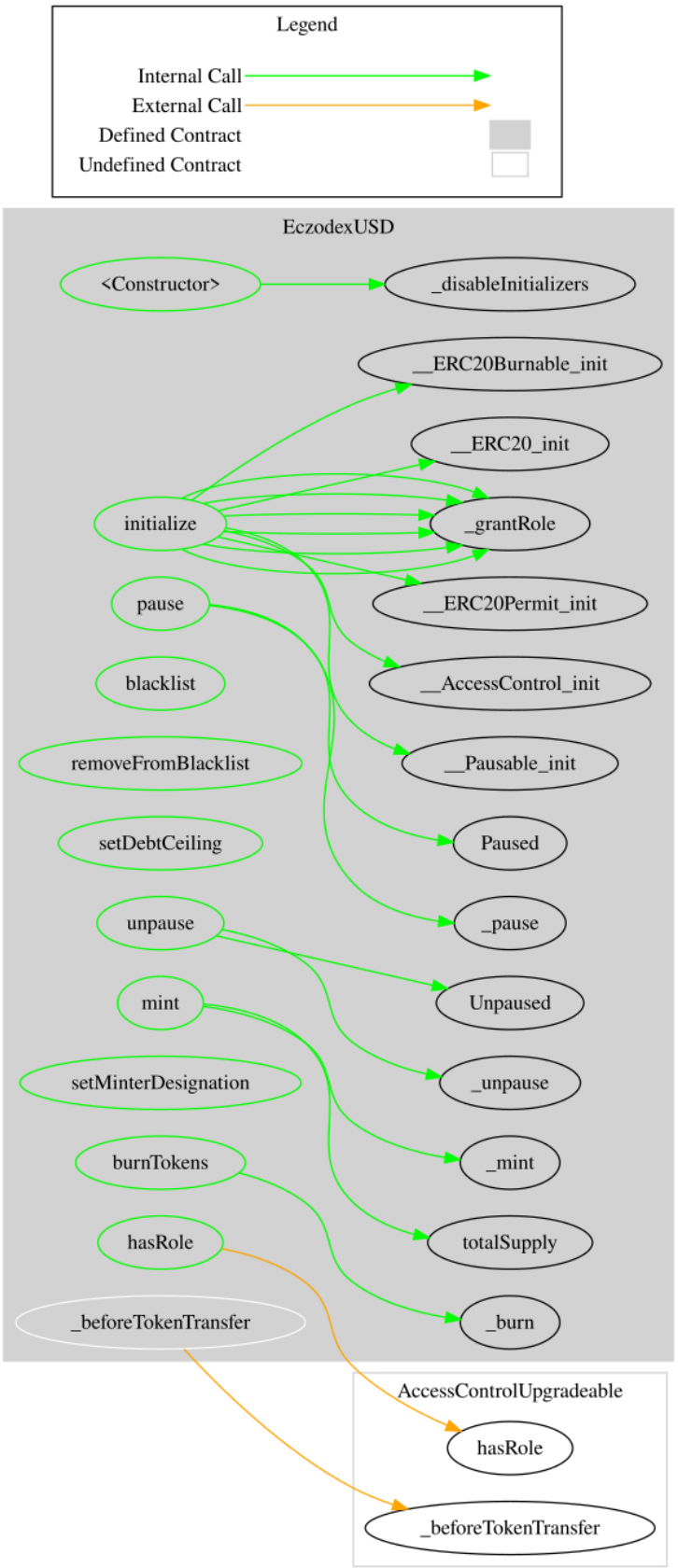
Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
EczodexUSD	Implementation	Initializable, ERC20Upgradeable, ERC20BurnableUpgradeable, PausableUpgradeable, ERC20PermitUpgradeable, AccessControlUpgradeable		
		Public	✓	-
	hasRole	Public		-
	initialize	Public	✓	initializer
	blacklist	Public	✓	onlyRole
	removeFromBlacklist	Public	✓	onlyRole
	setDebtCeiling	Public	✓	onlyRole
	pause	Public	✓	onlyRole
	unpause	Public	✓	onlyRole
	setMinterDesignation	Public	✓	onlyRole
	mint	Public	✓	onlyRole
	_beforeTokenTransfer	Internal	✓	whenNotPaused
	burnTokens	Public	✓	onlyRole

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
EczodexUSD
<Constructor>
_disableInitializers
__ERC20Burnable_init
__ERC20_init
initialize
_grantRole
pause
__ERC20Permit_init
blacklist
__AccessControl_init
removeFromBlacklist
__Pausable_init
setDebtCeiling
Paused
unpause
_pause
mint
Unpaused
setMinterDesignation
_unpause
burnTokens
_mint
hasRole
totalSupply
_beforeTokenTransfer
_burn
AccessControlUpgradeable
hasRole
_beforeTokenTransfer

Summary

Eczodex contract implements a token mechanism. This audit investigates security issues, business logic concerns, and potential improvements. The team has acknowledged the findings.

Initial Audit, 29 Apr 2024

At the time of the audit report, the contract with address 0xfcF54e37917473687592bc1ED48448610734Cc5e is pointed out by the following proxy address: 0x3D2F8AD00bA8Aeb3836ACE404071dA042cC604C6.

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