Cyberscope

Audit Report

Eczodex

May 2024

Network SEPOLIA

Address 0x85a8897727d09dd1aa08db739be6f29e8ddf7bbf

Audited by © cyberscope

Table of Contents

Review

Contract Name	EczodexUSD
Explorer	https://sepolia.etherscan.io/address/0x85a8897727d09dd1aa08db739be6f29e8ddf7bbf
Network	Sepolia

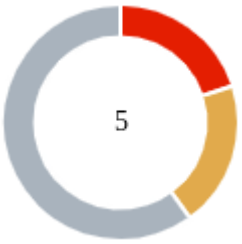
Audit Updates

Initial Audit	25 Apr 2024 https://github.com/cyberscope-io/audits/blob/main/1-usde/v1/audit.pdf
Corrected Phase 2	14 May 2024

Source Files

Filename	SHA256
EczodexUSD.sol	e3ef7e1580d2d775411f22aebbade8b37584515d460f805d2b46aecbb9b8118a

Findings Breakdown



Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	1	0	0
Medium	0	1	0	0
Minor / Informative	0	3	0	0

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
Critical	MT	Mints Tokens	Acknowledged
Medium	BC	Blacklists Addresses	Acknowledged
Minor / Informative	RMD	Redundant Minter Designation	Acknowledged
Minor / Informative	ST	Stops Transactions	Acknowledged
Minor / Informative	L19	Stable Compiler Version	Acknowledged

MT - Mints Tokens

Criticality	Critical
Location	EczodexUSD.sol#L111
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) {
    if (!isBlacklisted[account]) {
        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.

RMD - Redundant Minter Designation

Criticality	Minor / Informative
Location	EczodexUSD.sol#L105,111
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.

Team Update

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

1. Purpose and Utilization of setMinterDesignation: The setMinterDesignation function is a crucial component of our smart contract architecture. It is specifically designed to assign a designated wallet for each minter. This function is not used in regular minting operations but is intended for initial setup or occasional updates to the minter's designated wallet. This assignment method is vital in mitigating risks associated with minting to incorrect or unauthorized addresses.

2. Control and Security: By decoupling the minting process from the assignment of destination wallets, setMinterDesignation enhances control and security. It ensures that once a minter's wallet is designated, the minting process can proceed without repeatedly specifying the destination address, thus minimizing the risk of human error and unauthorized redirection of funds.

3. Regulatory Compliance and Flexibility: Our platform operates in a regulated environment, and many of our partners have legal and compliance obligations regarding the control and management of assets. The setMinterDesignation function allows these regulated entities to designate a wallet address of their choosing. This capability is critical for compliance with specific regulatory requirements and maintaining autonomy over operational wallets. It's a one-time configuration step establishing a clear, auditable trail of authority and responsibility.

4. Operational Efficiency in Context: While the auditor's suggestion to pass the destination address directly as a parameter in the mint function may seem to offer a reduction in transaction steps, in practice, the setMinterDesignation function eliminates potential errors and security risks that could arise from frequently specifying the destination address. This function is typically called only once or when there is a need to update the designated address, thus significantly mitigating the concern regarding transaction redundancy.

5. Alignment with Best Practices: Our approach aligns with best practices in software development, where a clear separation of concerns and roles enhances the system's security and maintainability. The explicit assignment of destination addresses through

setMinterDesignation ensures that roles and permissions are managed effectively, reducing the likelihood of unauthorized access or errors.

In conclusion, while we acknowledge the auditor's points on potential transaction redundancy, the setMinterDesignation function provides essential security, regulatory compliance, and operational integrity benefits. This method supports our mission to offer a secure, compliant, and user-centric stablecoin platform. We remain committed to continuous improvement and welcome further dialogue with our community and stakeholders on enhancing our systems.

ST - Stops Transactions

Criticality	Minor / Informative
Location	EczodexUSD.sol#L95,100
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.

Team Update

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

As with our MINTER and BLACKLISTER roles, we have addressed potential concerns through the implementation of the Dfns MPC wallet-as-a-service platform.

Dfns MPC Wallet Platform: As previously discussed in our responses regarding the MINTER and BLACKLISTER roles, we have implemented the Dfns MPC (Multi-Party Computation) wallet solution. This advanced technology ensures that no single individual can unilaterally pause or unpause the system, as multiple confirmations from authorized users are required to execute these critical actions. This setup enhances security by adding a consensus layer and significantly reducing the risk of misuse.

The Dfns platform is built on pioneering cryptography research and provides a bank-grade compliance framework that guarantees the security and integrity of our smart contract operations. This approach aligns with industry best practices and regulatory requirements, ensuring our system remains robust and compliant. By leveraging the Dfns MPC wallet solution, we maintain a high security and operational reliability standard, which is critical for our use by our regulated US Money Service Business (MSB) issuance partner.

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.

Team Update

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

After carefully considering and analyzing current industry standards and practices, we have retained the flexible compiler version notation. This approach allows us to take advantage of non-breaking improvements and optimizations available in newer minor and patch versions of the compiler while still ensuring compatibility with the base version we have thoroughly tested.

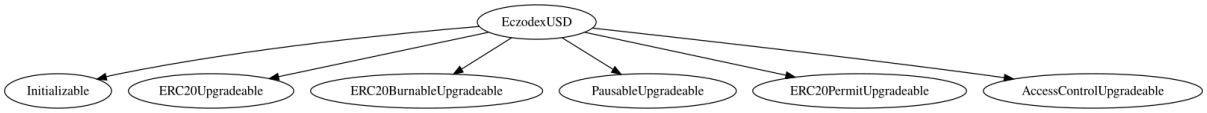
This shared industry practice reassures us that we are on the right track, maintaining flexibility while ensuring compatibility. This flexibility is crucial as it allows us to stay up-to-date with improvements in compiler efficiency, security enhancements, and new features that can benefit our contract's performance and security without necessitating frequent and extensive updates to our codebase. Furthermore, many stablecoin projects have adopted a similar approach.

We are committed to regularly reviewing our compiler strategy to ensure that it aligns with the best practices for security and stability. We will continue to monitor new releases and assess their impact to determine the appropriate time to update our compiler version or adjust our versioning strategy.

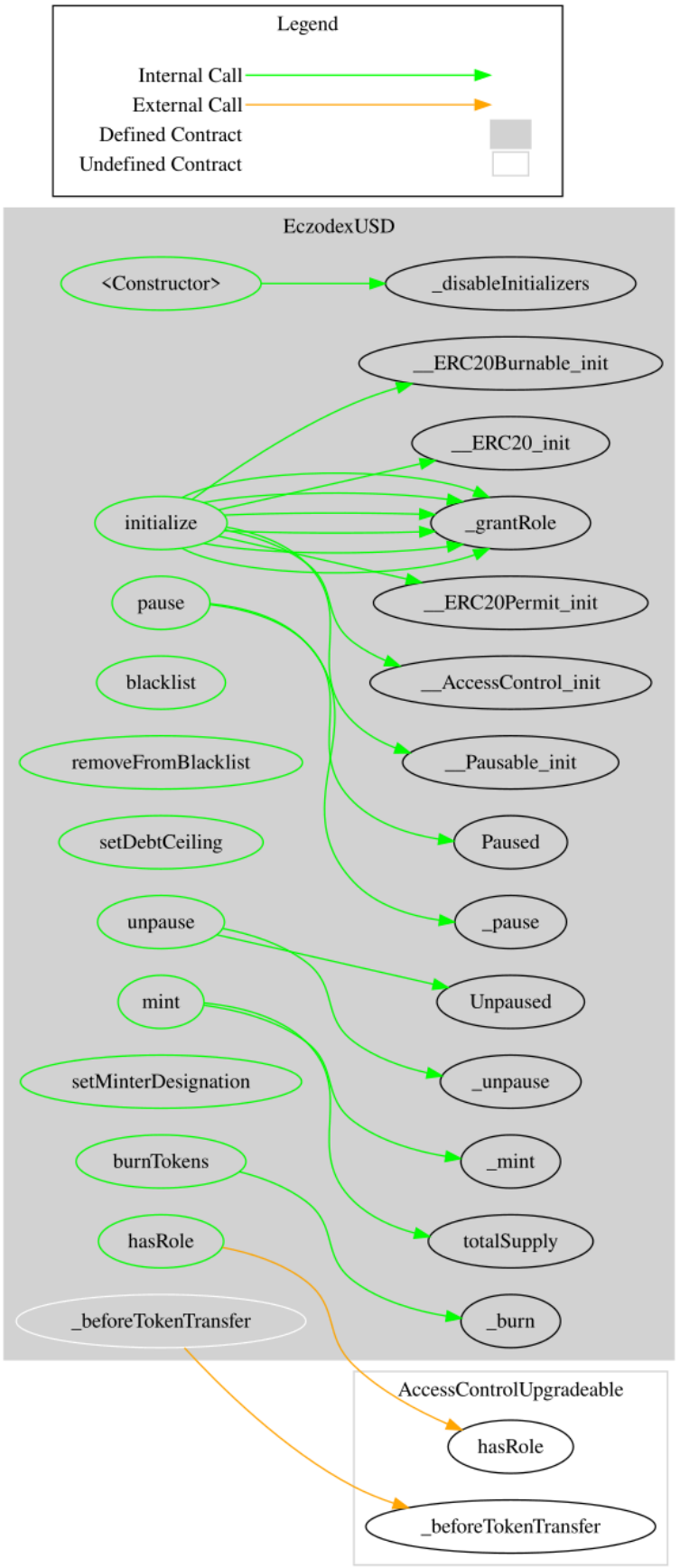
Functions Analysis

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.

Corrected Phase 2, 14 May 2024

At the time of the audit report, the contract with address 0x85A8897727d09DD1aA08DB739bE6f29E8Ddf7Bbf is pointed out by the following proxy address: 0xa6dFD766Ec553185148D8A31B597B28E1Ff862f8.

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