SHELLBOXES

Creo Launchpad Staking

Smart Contract Security Audit

Prepared by ShellBoxes

April 26th, 2024 – April 30th, 2024

Shellboxes.com

contact@shellboxes.com

Document Properties

Client	Creo
Version	1.0
Classification	Public

Scope

Repository	Commit Hash
https://github.com/Kommunitas-net/staking-v3/tree/creo-audit	ddf360d2d076c4b883d27d37d0e99134362ec976

Re-Audit

Repository	Commit Hash
https://github.com/Kommunitas-net/staking-v3/tree/creo-audit	e1032dca6a2b66b12c255482032e31f82119f5f2

Contacts

COMPANY	EMAIL
ShellBoxes	contact@shellboxes.com

Contents

1 Introduction

Creo engaged ShellBoxes to conduct a security assessment on the Creo Launchpad Staking beginning on April 26th, 2024 and ending April 30th, 2024. In this report, we detail our methodical approach to evaluate potential security issues associated with the implementation of smart contracts, by exposing possible semantic discrepancies between the smart contract code and design document, and by recommending additional ideas to optimize the existing code. Our findings indicate that the current version of smart contracts can still be enhanced further due to the presence of many security and performance concerns.

This document summarizes the findings of our audit.

1.1 About Creo

Creo Engine is a web3 gaming ecosystem that connects worlds in a one-size-fits-all gaming hub, leveling up the web3 gaming experience for everyone's benefit!

Issuer	Creo
Website	https://creoengine.com/
Type	Solidity Smart Contract
Documentation	Creo Engine Docs
Audit Method	Whitebox

1.2 Approach & Methodology

ShellBoxes used a combination of manual and automated security testing to achieve a balance between efficiency, timeliness, practicability, and correctness within the audit's scope. While manual testing is advised for identifying problems in logic, procedure, and implementation, automated testing techniques help to expand the coverage of smart contracts and can quickly detect code that does not comply with security best practices.

1.2.1 Risk Methodology

Vulnerabilities or bugs identified by ShellBoxes are ranked using a risk assessment technique that considers both the LIKELIHOOD and IMPACT of a security incident. This framework is effective at conveying the features and consequences of technological vulnerabilities.

Its quantitative paradigm enables repeatable and precise measurement, while also revealing the underlying susceptibility characteristics that were used to calculate the Risk scores. A risk level will be assigned to each vulnerability on a scale of 5 to 1, with 5 indicating the greatest possibility or impact.

- Likelihood quantifies the probability of a certain vulnerability being discovered and exploited in the untamed.
- Impact quantifies the technical and economic costs of a successful attack.
- Severity indicates the risk's overall criticality.

Probability and impact are classified into three categories: H, M, and L, which correspond to high, medium, and low, respectively. Severity is determined by probability and impact and is categorized into four levels, namely Critical, High, Medium, and Low.

Impact	High	Critical	High	Medium
	Medium	High	Medium	Low
	Low	Medium	Low	Low
		High	Medium	Low

Likelihood

2 Findings Overview

2.1 Summary

The following is a synopsis of our conclusions from our analysis of the Creo Launchpad Staking implementation. During the first part of our audit, we examine the smart contract source code and run the codebase via a static code analyzer. The objective here is to find known coding problems statically and then manually check (reject or confirm) issues highlighted by the tool. Additionally, we check business logics, system processes, and DeFi-related components manually to identify potential hazards and/or defects.

2.2 Key Findings

In general, these smart contracts are well-designed and constructed, but their implementation might be improved by addressing the discovered flaws, which include , 2 medium-severity, 1 low-severity vulnerabilities.

Vulnerabilities	Severity	Status
SHB.1. Decimal Precision Mismatch for CREO Token	MEDIUM	Fixed
SHB.2. Centralization Risk	MEDIUM	Acknowledged
SHB.3. Owner Can Renounce Ownership	LOW	Acknowledged

3 Finding Details

SHB.1 Decimal Precision Mismatch for CREO Token

- Severity : MEDIUM
- Status : Fixed

- Likelihood : 2
- Impact : 2

Description:

The CreoEngineStaking implementation assumes that the CREO token has 18 decimals. However, the CreoEngineDummy contract, which represents the CREO token, specifies only 8 decimals. This inconsistency in decimal precision between the implementation and the dummy contract can lead to incorrect calculations and potential issues in the staking mechanism.

Files Affected:

SHB.1.1: CreoEngineDummy.sol

```
contract CreoEngineDummy is ERC20('CreoEngine', 'CREO'), ERC20Burnable {
    constructor() {
        _mint(0xd5a468Ca329760E0823F2Ec70EA0Aca898d24306, 1000000 * (10
            ↪ ** decimals()));
        _mint(0x5dd51918C3594324728AFf637AE12f8178F20575, 1000000 * (10
            ↪ ** decimals()));
    }

    function decimals() public view virtual override returns (uint8) {
        return 8;
    }
}
```

SHB.1.2: CreoEngineStaking.sol

```
        minStaking = 100 * 1e18; // 100 creoToken
        maxStaking = 1000000000 * 1e18; // 1B creoToken
        minGetCreoV = 5000 * 1e18; // 5K creoToken
```

Recommendation:

To address this issue, consider removing the decimals function override from the CreoEngineDummy contract. The default value for the ERC20 token decimals is 18, which aligns with the actual decimal precision of the CREO token. Removing the override ensures consistency in decimal precision and avoids potential issues in the staking mechanism.

Updates

The team has fixed the issue by removing the decimals overridden function from the CreoEngineDummy contract.

SHB.2 Centralization Risk

- Severity : MEDIUM
- Status : Acknowledged

- Likelihood : 2
- Impact : 2

Description:

The current implementation of the CreoEngineStaking contract grants the owner significant control over critical functions. The owner can manage the contract workers, set the period in days for staking, set penalty fees, set APY, adjust the minimum and maximum staking token amounts, and control the pausable feature. This centralization of control poses a risk as it concentrates power in the hands of a single entity, potentially leading to abuse or manipulation of the contract's functionality.

Files Affected:

SHB.2.1: CreoEngineStaking.sol

```solidity
    function addWorker(address _worker) external virtual onlyOwner {
```

SHB.2.2: CreoEngineStaking.sol

```solidity
    function removeWorker(address _worker) external virtual onlyOwner {
```

SHB.2.3: CreoEngineStaking.sol

```solidity
    function changeWorker(address _oldWorker, address _newWorker)
       ↪ external virtual onlyOwner {
```

SHB.2.4: CreoEngineStaking.sol

```solidity
    function toggleTrustedForwarder(address _forwarder) external virtual
       ↪  onlyOwner {
```

SHB.2.5: CreoEngineStaking.sol

```solidity
    function setMinMax(
        uint128 _minGetCreoV,
        uint128 _minStaking,
        uint128 _maxStaking
    ) external virtual whenPaused onlyOwner {
```

SHB.2.6: CreoEngineStaking.sol

```solidity
    function setPeriodInDays(uint16 _lockIndex, uint128
       ↪ _newLockPeriodInDays) external virtual onlyOwner {
```

SHB.2.7: CreoEngineStaking.sol

```solidity
    function setPenaltyFee(uint16 _lockIndex, uint64 _feeInPercent_d2)
       ↪ external virtual onlyOwner {
```

SHB.2.8: CreoEngineStaking.sol

```solidity
    function setAPY(uint16 _lockIndex, uint64 _apy_d2) external virtual
       ↪ onlyOwner {
```

SHB.2.9: CreoEngineStaking.sol

```
    function addLockNumber(
        uint128 _lockPeriodInDays,
        uint64 _apy_d2,
        uint64 _feeInPercent_d2
    ) external virtual whenPaused onlyOwner {
```

SHB.2.10: CreoEngineStaking.sol

```
    function togglePause() external virtual onlyOwner {
```

Recommendation:

To mitigate this risk, it is recommended to reduce centralization by implementing mechanisms that decentralize control over critical functions. Consider using multi-signature schemes for key actions, implementing community governance features, or utilizing decentralized autonomous organizations (DAOs) to manage the contract.

Updates

The team has acknowledged the risk, stating that they want to be able to manage the staking duration, APY, and any other future aspects. They also plan to use a multisignature wallet to manage it.

SHB.3 Owner Can Renounce Ownership

- Severity : LOW
- Status : Acknowledged
- Likelihood : 1
- Impact : 2

Description:

The CreoTokenVoting governance token contract inherits from the Ownable OpenZeppelin contract, which allows the owner to renounce ownership. Renouncing ownership leaves the contract without an owner, effectively disabling any functionality exclusively available to the owner. This poses a risk as it could lead to the contract becoming unusable or losing control over key functions.

Files Affected:

SHB.3.1: CreoTokenVoting.sol

```
// CreoTokenVoting - Governance Token
contract CreoTokenVoting is ERC20('CreoTokenVoting', 'CREOV'), Ownable {
```

Recommendation:

It is recommended to prevent the owner from invoking the renounceOwnership function or to disable its functionality by overriding it. Alternatively, consider inheriting from the OwnableUpgradeable contract instead of the Ownable OpenZeppelin contract, as it provides a safer way to manage ownership.

Updates

The team has acknowledged the issue and indicated that they would like to retain the ability for the owner to renounce ownership as a potential feature.

4 Best Practices

BP.1 Remove Unused swapPaused Variable

Description:

The CreoTokenVoting contract contains a swapPaused boolean variable that is declared but not utilized in the contract's logic. This unused variable adds unnecessary complexity to the contract and increases the potential for confusion. It is recommended to remove the swapPaused variable and related functions, such as toggleSwap, to streamline the contract and improve readability. This practice reduces the risk of accidental misuse or misunderstanding of the contract's functionality.

Files Affected:

BP.1.1: CreoTokenVoting.sol

```
    bool public swapPaused = false;
```

BP.1.2: CreoTokenVoting.sol

```
    function toggleSwap() public onlyOwner {
        swapPaused = !swapPaused;
    }
```

Status - Fixed

BP.2 Enhancing transferFrom Functionality with Additional Logic

Description:

When overriding functions, such as transferFrom in the CreoTokenVoting contract, to add extra functionality without duplicating code, it's a best practice to use super.transferFrom() to invoke the parent contract's implementation of the function and then add the extra logic.

In this case, the additional logic is _moveDelegates call. This approach ensures that the original functionality is maintained and any updates or improvements to the parent contract's logic are automatically inherited.

Files Affected:

BP.2.1: CreoTokenVoting.sol

```
    function transferFrom(
        address sender,
        address recipient,
        uint256 amount
    ) public override hasPermission returns (bool) {
        _transfer(sender, recipient, amount);
        _approve(
            sender,
            _msgSender(),
            allowance(sender, _msgSender()).sub(amount, 'ERC20: transfer amount exceeds allowance')
        );
        _moveDelegates(_delegates[sender], _delegates[recipient], amount);
        return true;
    }
```

Status - Fixed

BP.3 Improve Error Message Clarity in CreoEngineStaking Contract

Description:

Most functions in the CreoEngineStaking contract use vague and uninformative error message (bad). When implementing functions that require conditions, it's crucial to provide clear and descriptive error messages. Vague messages like bad can obscure the understanding of what went wrong during execution, making debugging difficult and decreasing the code's usability and auditability. Instead, strive to use specific error messages that provide clear information about the nature of the error.

Files Affected:

BP.3.1: CreoEngineStaking.sol

```
305         require(staked[_staker].length > _userStakedIndex, 'bad');
```

BP.3.2: CreoEngineStaking.sol

```
347         require(
348             staked[_staker].length > _userStakedIndex && // user staked
                    ↪ index validation
349                 stakeDetail.compoundType != _newCompoundType, // compound
                        ↪ type validation
350             'bad'
351         );
```

BP.3.3: CreoEngineStaking.sol

```
554         require(_worker != address(0) && !isWorker[_worker], 'bad');
```

BP.3.4: CreoEngineStaking.sol

```
566         require(
567             _oldWorker != address(0) && _newWorker != address(0) &&
                    ↪ isWorker[_oldWorker] && !isWorker[_newWorker],
568             'bad'
569         );
```

BP.3.5: CreoEngineStaking.sol

```
592         require(lockNumber > _lockIndex && _newLockPeriodInDays >= 1 &&
                ↪ _newLockPeriodInDays <= (5 * 365), 'bad');
```

BP.3.6: CreoEngineStaking.sol

```
        require(lockNumber > _lockIndex && _feeInPercent_d2 >= 100 &&
            ↪ _feeInPercent_d2 < 10000, 'bad');
```

BP.3.7: CreoEngineStaking.sol

```
        require(lockNumber > _lockIndex && _apy_d2 < 10000, 'bad');
```

Status - Fixed

BP.4 Upgrade Pragma Version for CreoTokenVoting Contract

Description:

The CreoTokenVoting contract currently uses pragma version 0.7.6, which is an older version of Solidity. It is recommended to upgrade the pragma version to 0.8.x to benefit from the improvements and optimizations introduced in newer versions. By upgrading, the contract can avoid importing the SafeMath library for arithmetic operations, as the compiler now includes built-in checks for arithmetic overflow and underflow. This upgrade can enhance the contract's consistency, readability, and efficiency.

Files Affected:

BP.4.1: CreoTokenVoting.sol

```
pragma solidity 0.7.6;
```

Status - Fixed

5 Conclusion

In this audit, we examined the design and implementation of Creo Launchpad Staking contract and discovered several issues of varying severity. Creo team addressed 1 issue raised in the initial report and implemented the necessary fixes, while classifying the rest as a risk with low-probability of occurrence. Shellboxes' auditors advised Creo Team to maintain a high level of vigilance and to keep those findings in mind in order to avoid any future complications.

6 Scope Files

6.1 Audit

Files	MD5 Hash
contracts/CreoEngineStaking.sol	dc80ff4eb56f0b251d7d8603feb437dc
contracts/util/CreoEngineDummy.sol	38b1cd1ba967d29383ef2a8f0c3dd5c3
contracts/util/CreoTokenVoting.sol	fd2d181e41d0267e1af9be1ffa108b2d

6.2 Re-Audit

Files	MD5 Hash
contracts/CreoEngineStaking.sol	6ee5049a30bcb5bb733d8a7a7d00eb39
contracts/util/CreoEngineDummy.sol	b368810c468a9b432cd5b064b2dbfc2d
contracts/util/CreoTokenVoting.sol	ccdc6ac44ecf06a327d31f4438ec3455

7 Disclaimer

Shellboxes reports should not be construed as ”endorsements” or ”disapprovals” of particular teams or projects. These reports do not reflect the economics or value of any ”product” or ”asset” produced by any team or project that engages Shellboxes to do a security evaluation, nor should they be regarded as such. Shellboxes Reports do not provide any warranty or guarantee regarding the absolute bug-free nature of the examined technology, nor do they provide any indication of the technology's proprietors, business model, business or legal compliance. Shellboxes Reports should not be used in any way to decide whether to invest in or take part in a certain project. These reports don't offer any kind of investing advice and shouldn't be used that way. Shellboxes Reports are the result of a thorough auditing process designed to assist our clients in improving the quality of their code while lowering the significant risk posed by blockchain technology. According to Shellboxes, each business and person is in charge of their own due diligence and ongoing security. Shellboxes does not guarantee the security or functionality of the technology we agree to research; instead, our purpose is to assist in limiting the attack vectors and the high degree of variation associated with using new and evolving technologies.

SHELLBOXES

For a Contract Audit, contact us at contact@shellboxes.com