Monocerus (CERUS) token

Smart Contract Security Audit

Prepared by ShellBoxes

October 23rd, 2023 - October 24th, 2023

Shellboxes.com

contact@shellboxes.com

Document Properties

Client	Monocerus
Version	1.0
Classification	Public

Scope

Contract Name	Contract Address
CerusToken	0x6F0167Dca1fAB2AaBe20cf56B7a7E1FdB47AC388

Re-Audit

Contract Name	Contract Address
CerusToken	0x6F0167Dca1fAB2AaBe20cf56B7a7E1FdB47AC388

Contacts

COMPANY	EMAIL
ShellBoxes	contact@shellboxes.com

Contents

1 Introduction

Monocerus engaged ShellBoxes to conduct a security assessment on the Monocerus (CERUS) token beginning on October 23rd, 2023 and ending October 24th, 2023. 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 Monocerus

Monocerus is a platform that combines the concepts of decentralized finance (DeFi) and non-fungible tokens (NFTs) to create a unique gaming experience. The platform offers a variety of features and utilities, including: high-yield GameFi,Dynamic NFTs and more!

Issuer	Monocerus
Website	`https://monocerus.world`
Type	Solidity Smart Contracts
Documentation	Monocerus 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 Monocerus (CERUS) token 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 , 1 medium-severity, 2 low-severity vulnerabilities.

Vulnerabilities	Severity	Status
SHB.1. Potential for Disruption of Transfers by PAUSER_ROLE	MEDIUM	Mitigated
SHB.2. Unrestricted Control Over Supply Minting by MINTER_ROLE	LOW	Mitigated
SHB.3. Floating Pragma	LOW	Acknowledged

3 Finding Details

SHB.1 Potential for Disruption of Transfers by PAUSER_ROLE

- Severity : MEDIUM
- Status : Mitigated
- Likelihood : 1
- Impact : 3

Description:

The smart contract includes a PAUSER_ROLE that has the ability to pause and unpause the contract. When the contract is paused, all token transfers are halted. This could potentially disrupt the token's economy if the PAUSER_ROLE is misused or falls into the wrong hands. An attacker with control of this role could halt all token transfers, causing significant disruption to users and potentially damaging trust in the token.

Files Affected:

SHB.1.1: CerusToken.sol

```
4365    function pause() public onlyRole(PAUSER_ROLE) {
4366        _pause();
4367    }
```

SHB.1.2: CerusToken.sol

```
4369    function unpause() public onlyRole(PAUSER_ROLE) {
4370        _unpause();
4371    }
```

SHB.1.3: CerusToken.sol

```
4377    function _beforeTokenTransfer(address from, address to, uint256
            ↪ amount)
4378        internal
```

```
4379            whenNotPaused
4380            override(ERC20, ERC20Snapshot)
4381        {
4382            super._beforeTokenTransfer(from, to, amount);
4383        }
```

Recommendation:

Consider implementing additional safeguards around the use of the PAUSER_ROLE. This could include requiring multiple signatures to pause the contract, or implementing a time delay before the pause function takes effect, giving users a chance to react.

Updates

The team mitigated the risk by revoking the PAUSER_ROLE from the previous address and granting it to a multi-sig safe.

- revokeRole transaction
- grantRole transaction

SHB.2 Unrestricted Control Over Supply Minting by MINTER_ROLE

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

Description:

In the provided smart contract, the MINTER_ROLE has unrestricted control over the minting of the token supply. This role is granted the ability to mint the whole supply of tokens. While this may be intended functionality, it poses a risk if the account with the MINTER_ROLE is compromised or misused. An attacker gaining control of this role could inflate the token

supply, potentially devaluing the token and disrupting the token economy. Therefore, there are no guarantees that the token distribution will be performed as stated in the tokenomics' documentation.

Files Affected:

SHB.2.1: CerusToken.sol

```
4373    function mint(address to, uint256 amount) public onlyRole(
            ↪ MINTER_ROLE) {
4374        _mint(to, amount);
4375    }
```

Recommendation:

Consider implementing a mechanism to ensure that the token will be distributed as stated in the tokenomics.

Updates

The team mitigated the risk by revoking the MINTER_ROLE from the previous address and granting it to a multi-sig safe.

- revokeRole transaction
- grantRole transaction

SHB.3 Floating Pragma

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

Description:

The contract uses a floating solidity pragma of 0.8.19;. This means that the contract can be compiled with any compiler version from 0.8.19 (inclusive) up to, but not including, version 0.9.0. This could potentially introduce unexpected behavior if the contract is compiled with a newer compiler version that includes breaking changes.

Files Affected:

SHB.3.1: CerusToken.sol

```
4347  pragma solidity ^0.8.19;
```

Recommendation:

It is generally recommended to lock the pragma statement to a specific compiler version to ensure that the contract behaves as expected. This can be done by removing the caret from the pragma statement.

Updates

The team acknowledged the issue, stating that there will be no recompilation, as set in Developer's Standard Operational Procedure to always use the 0.8.19 version.

4 Best Practices

BP.1 Unused Inheritance of the Ownable Contract

Description:

Upon analyzing the provided CerusToken contract, it is observed that the contract inherits from the Ownable contract. However, the onlyOwner modifier, which is typically provided by the Ownable contract to restrict certain functions to be callable only by the contract's owner, is not utilized anywhere within the CerusToken contract. The CerusTokencontract has multiple functions such as snapshot, pause, unpause, and mint that are restricted using the onlyRole modifier. This modifier checks if the caller has a specific role (e.g., SNAPSHOT_ROLE, PAUSER_ROLE, MINTER_ROLE) before allowing them to execute the function. There is no function in the CerusToken contract that uses the onlyOwner modifier to restrict its execution to the contract's owner. If there is no requirement for ownership-based restrictions in the contract, it is advisable to remove the Ownable inheritance to reduce unnecessary code and potential gas costs on deployment.

Files Affected:

BP.1.1: CerusToken.sol

```
4349 contract CerusToken is ERC20, ERC20Burnable, ERC20Snapshot, ERC20Capped,
       ↪ AccessControl, Ownable, Pausable, ERC20Permit, ERC20Votes {
```

Status - Acknowledged

5 Conclusion

In this audit, we examined the design and implementation of Monocerus (CERUS) token contract and discovered several issues of varying severity. Monocerus team mitigated 2 issues 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 Monocerus 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
CerusToken.sol	afa3d98526665d46815d545c22e4b5a0

6.2 Re-Audit

Files	MD5 Hash
CerusToken.sol	afa3d98526665d46815d545c22e4b5a0

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.

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