Security Audit
Report
05/04/2022

WrappedCENNZ

Content

Introduction

CENNZnet is a New Zealand based public blockchain network that powers DApps and the crypto currencies CENNZ and CPAY WrappedCENNZ, it is the wrapped version of the CENNZ token, a token for staking and governance of the CENNZnet blockchain.

As requested by **CENNZnet** and as part of the vulnerability review and management process, Red4Sec has been asked to perform a security code audit in order to evaluate the security of the **WrappedCENNZ** project.

The report includes specifics of the audit for all the existing vulnerabilities of **WrappedCENNZ**. The performed analysis shows that the **WrappedCENNZ** does not contain any critical vulnerability. However, Red4Sec has found issues that need to be addressed and fixed. This document outlines the results of the security audit.

Disclaimer

This document only represents the results of the code audit conducted by Red4Sec Cybersecurity and should not be used in any way to make investment decisions or as investment advice on a project.

Likewise, the report should not be considered neither "endorsement" nor "disapproval" of the guarantee of the correct business model of the analyzed project.

Scope

CENNZnet has asked Red4Sec to perform an analysis of the project's **source code and Smart Contracts.**
Red4Sec has made a thorough audit of the smart contract security level against attacks, identifying possible errors in the design, configuration or programming; therefore, guaranteeing the confidentiality, integrity and availability of the information accessed, treated, and stored.

The scope of this evaluation includes the following projects enclosed in the repository:

- https://github.com/cennznet/bridge-contracts/blob/771d1fcd7457054636de764f9bed0058bae6af7d/contracts/WrappedCENNZ.sol
- branch: `main`
- commit: `771d1fcd7457054636de764f9bed0058bae6af7d`

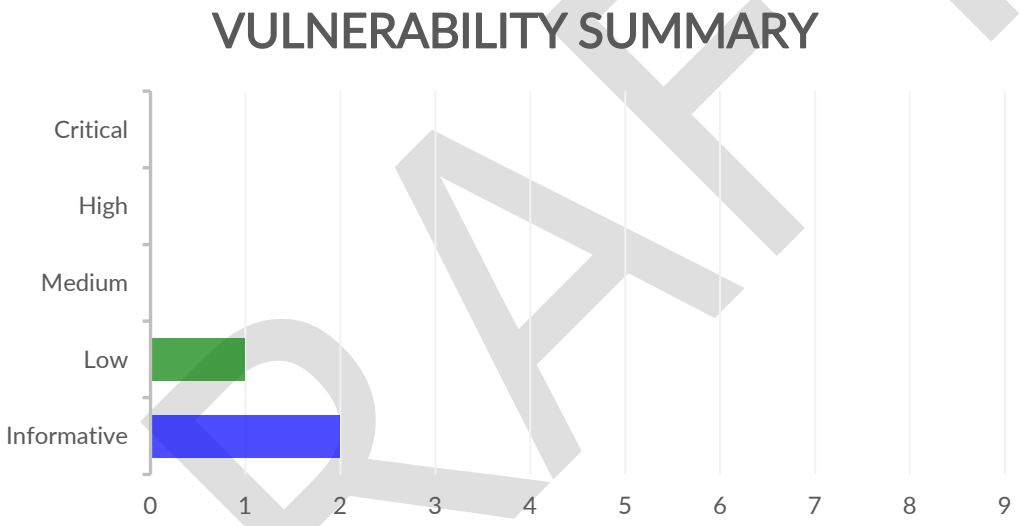
Executive Summary

As requested by **CENNZnet**, the company Red4Sec Cybersecurity has been asked to perform a security audit against the **WrappedCENNZ**, in order to assess the security of the source code and in addition to the vulnerability review and the management process.

This security audit has been conducted between the following dates: **03/31/2022** and **04/05/2022**.

Once the analysis of the technical aspects of the environment has been completed, the performed analysis shows that the audited source code contains non-critical vulnerabilities that should be mitigated as soon as possible.

During the analysis, a total of **3 vulnerabilities** were detected. These vulnerabilities have been classified in the following levels of risk according to the impact level defined by CVSS (Common Vulnerability Scoring System):



Conclusions

To this date, **05/04/2022**, the general conclusion resulting from the conducted audit, is that the **Centrality is secure**. Nevertheless, Red4Sec has found a few potential improvements, these do not pose any risk by themselves, and we have classified such issues as informative only, but they will help to continue to improve the security and quality of its developments.

The general conclusions of the performed audit are:

- **The token contract allows the `MINTER_ROLE` and `DEFAULT_ADMIN_ROLE` to arbitrarily remove balance from any account**. The Burn functionality should only allow to burn tokens that are in possession of the user that is executing a burn.
- The logic of the contract allows the owner to alter certain values of the contract at will, allowing to obtain an advantageous position in certain situations.
- A **few low impact issues** were detected and classified only as informative, but they will continue to help Centrality improve the security and quality of its developments.

Vulnerabilities

In this section, you can find a detailed analysis of the vulnerabilities encountered upon the security audit.

List of vulnerabilities

Below, we have gathered a complete list of the vulnerabilities detected by Red4Sec, presented and summarized in a way that can be used for risk management and mitigation.

All these vulnerabilities have been classified in the following levels of risk according to the impact level defined by CVSS v3 (Common Vulnerability Scoring System) by the National Institute of Standards and Technology (NIST):

Table of vulnerabilities			
ID	**Vulnerability**	**Risk**	**State**
CN-01	Contracts Management Risks	**Low**	**Open**
CN-02	Improvable Mint/Burn Logic	**Informative**	**Open**
CN-03	Use of Deprecated _setupRole	**Informative**	**Open**

Vulnerability details

In this section, we provide the details of each of the detected vulnerabilities indicating the following aspects:

- Category
- Active
- Risk
- Description
- Recommendations

Contracts Management Risks

Identifier	Category	Risk	State
CN-01	Bad Coding Practices	**Low**	**Open**

The logic design of the `WrappedCENNZ` contracts imply a few minor risks that should be reviewed and considered for their improvement.

Arbitrary Token Minting and Burning

The current implementation of the token delegates management to the admin; such as mining without a capped supply, or the possibility of burning tokens from certain accounts. Both, especially the last mentioned, are dangerous and must be properly controlled in order to not cause harm to the users.

```
/** @dev The burn function is protected by the onlyRole modifier. */
function burn(address from, uint256 amount) public onlyRole(MINTER_ROLE) {
    _burn(from, amount);
}
```

Possible frozen token

Even though this logic is intentional, it is necessary to mention that the `WrappedCENNZ` token allows to pause its functionality (`transfer`, `burn`…) and consequently revoke or give up the administrator role, which would finally leave the token permanently and irrevocably useless.

Source References

- https://github.com/cennznet/bridge-contracts/blob/3c5b63baf3a7a0202012ca9bfd5812395929fac6/contracts/WrappedCENNZ.sol#L26-L55

Improvable Mint/Burn Logic

Identifier	Category	Risk	State
CN-02	Bad Coding Practices	**Informative**	**Open**

During the security audit it has been detected that the `transfer` and `transferFrom` methods do not follow the standard defined in the EIP-20.

Carrying out the `mint` or `burn` process during the `transfer` or `transferFrom`, in addition to not complying with the defined standard, can be confusing for the users and causes the accounts with `MINTER_ROLE` to operate abnormally.

As can be seen in the following images, a user with the `MINTER_ROLE` role will not be able to normally call the `transfer` method or the `transferFrom` method. Actually, it will call the `mint` and `burn` methods instead, this action results completely unnecessary since there are already methods that do it and it will deny said user the possibility of transferring tokens.

```
function transfer(address buyer, uint256 numTokens) public override returns (bool) {
    if (hasRole(MINTER_ROLE, msg.sender)) {
        mint(buyer, numTokens);
        return true;
    }
    return super.transfer(buyer, numTokens);
}

function transferFrom(address owner, address buyer, uint256 numTokens) public override returns (bool) {
    if (hasRole(MINTER_ROLE, msg.sender)) {
        burn(owner, numTokens);
        return true;
    }
    return super.transferFrom(owner, buyer, numTokens);
}
```

This process will increase the gas costs to the rest of the user since most users will not have the role `MINTER_ROLE`, and even so they will still pay this extra cost of role verification during all the transfers throughout the life of the contract.

Recommendations

It is recommended to only use the `mint` and `burn` methods for the token creation and destruction processes.

References

- https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md#transfer
- https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md#transferfrom

Use of Deprecated _setupRole

Identifier	Category	Risk	State
CN-03	Outdated Software	**Informative**	**Open**

It has been detected that the code uses deprecated or obsolete functions, which suggests that the code has not been actively reviewed or maintained.

The `WrappedCENNZ` contract imports the `AccessControl` class of OpenZeppelin to add the access control functionality. During the constructor of `WrappedCENNZ`, the deprecated method `_setupRole` is used; it is recommended to replace it with the `_grantRole` method.

Recommendations

It is recommended not to use a deprecated function in order to maintain the code as updated as possible and to prevent potential future errors.

References

- https://docs.openzeppelin.com/contracts/4.x/api/access#AccessControl-_setupRole-bytes32-address-

Annexes

Annex A – Vulnerabilities Severity

Red4Sec determines the vulnerabilities severity in the following levels of risk according to the impact level defined by CVSS v3 (Common Vulnerability Scoring System) by the National Institute of Standards and Technology (NIST):

Vulnerability Severity

The risk classification has been made on the following 5 value scale:

Severity	Description
Critical	Vulnerabilities that possess the highest impact over the systems, services and/or sensitive information. The existence of these vulnerabilities is dangerous and should be fixed as soon as possible.
High	Vulnerabilities that could compromise severely compromise the service or the information it manages even if the vulnerability requires expertise to be exploited.
Medium	Vulnerabilities that on their own can have a limited impact and/or that combined with other vulnerabilities could have a greater impact.
Low	These vulnerabilities do not suppose a real risk for the systems. Also includes vulnerabilities which are extremely hard to exploit or whose impact on the service is low.
Informative	It covers various characteristics, information or behaviours that can be considered as inappropriate, without being considered as vulnerabilities by themselves.

Invest in Security, invest in your future