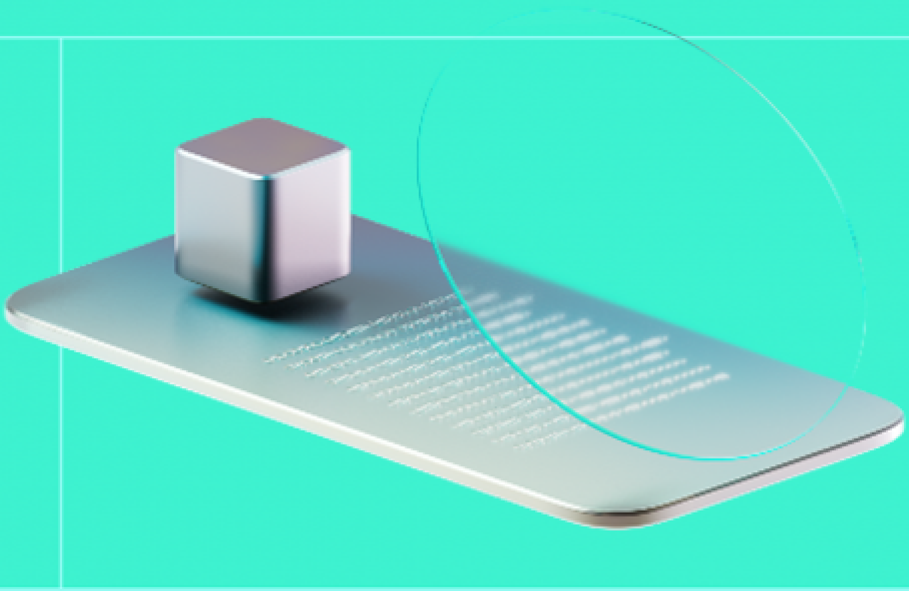
Smart Contract Code Review And Security Analysis Report



Customer: Bonuz

Date: 18.12.2023

We thank Bonuz for allowing us to conduct a Smart Contract Security Assessment. This document outlines our methodology, limitations, and results of the security assessment.

Bonuz is a groundbreaking solution, synthesizing top-tier DEX methodologies into a novel, high-performance flywheel.

Platform: EVM

Language: Solidity

Tags: ERC721

Timeline: 15.12.2023-18.12.2023

Methodology: https://hackenio.cc/sc_methodology

Last Review Scope

Repository	https://github.com/bonuz-market/BonuzSmartContracts
Commit	4f6817d

Audit Summary

10/10	10/10	100%	10/10
Security Score	Code quality score	Test coverage	Documentation quality score

Total 10/10

The system users should acknowledge all the risks summed up in the risks section of the report

2	2	0	0
Total Findings	Resolved	Accepted	Mitigated

Findings by severity

Critical	0
High	0
Medium	0
Low	2

Vulnerability	Status
F-2023-0027 - Missing zero address check	Fixed
F-2023-0034 - Missing validation in reedem voucher function	Fixed

This report may contain confidential information about IT systems and the intellectual property of the Customer, as well as information about potential vulnerabilities and methods of their exploitation.

The report can be disclosed publicly after prior consent by another Party. Any subsequent publication of this report shall be without mandatory consent.

Document

Name	Smart Contract Code Review and Security Analysis Report for Bonuz
Audited By	Carlo Parisi, Roman Tiutiun
Approved By	Przemyslaw Swiatowiec
Changelog	18.12.2023 Initial Review

Table to Contents

System Overview

The BonuzTokens Contract creates and manages, ERC721 tokens, featuring specialized roles for administrators and issuers, and capabilities for token metadata handling, redemption, and loyalty point management.

The BonuzSocialId Contract focuses on establishing and managing user profiles on the blockchain, allowing the addition and control of personal details and social links, with specific functionalities accessible only to authorized issuers and designed with upgradeability and security in mind.

Privileged roles

- The owner of the bonuzSocialId contract is able to:
  - pause;
  - unpause;
  - set issuer;
  - set allowed social link;
- The issuer of BonuzSocialId contract is able to:
  - set the user image;
  - set the user handle;
  - set the user name;
  - set the user social links;
- The Issuer of the BonuzTokens contract is able to:
  - mint tokens;
  - redeem voucher;
  - add loyalty Points;
  - remove loyalty points;
- The ower of the BonuzTokens contract is able to:
  - set an admin;
  - set pause;
- The admin of BonuzTokens contract is able to:
  - set an issuer;

Executive Summary

The score measurement details can be found in the corresponding section of the scoring methodology.

Documentation quality

The total Documentation Quality score is **10** out of **10**.

- Functional requirements are provided.
- Technical description is provided.
- Development environment is described.

Code quality

The total Code Quality score is **10** out of **10**.

Test coverage

Code coverage of the project is **100%** (branch coverage),

Security score

As a result of the audit, the code contains **0** issues. The security score is **10** out of **10**.

All found issues are displayed in the “Findings” section.

Risks

- The `mint()` function in the provided smart contract lacks proper validation for the `_expiryDate` parameter when called by an onlyIssuer. If an onlyIssuer adds `_expiryDate` as 0, it creates a vulnerability where the token becomes locked indefinitely. This oversight may allow an issuer to unintentionally or maliciously lock a token permanently, impacting the token's intended lifecycle and functionality.
- Using a version of the OpenZeppelin contracts from the version 5 and above will create problems when using the dependency `ERC721`, the project overrides `_beforeTokenTransfer()` to reject invalid transactions, the function `_beforeTokenTransfer()` has been removed in the newer version of the OpenZeppelin contracts.
- Using a version of the OpenZeppelin contracts from the version 5 and above will create problems when using the dependency `Ownable` and `OwnableUpgradeable`, the `constructor()` and `initializer()` have changed in the newer version of the OpenZeppelin contracts.
- Every function in the scope is either a view function or a function assigned to a predetermined role in the contract, as such, the contracts are extremely centralized, the admin, owners and issuers hold a lot of power over the contracts that should be handled carefully.
- The function `getUserProfileAndSocialLinks()` will return an array of social links, among other things, if any of the requested platforms are not allowed the returned array will have empty elements, this should be taken into account when interacting with this function.

Findings

Vulnerability Details

F-2023-0027 - Missing zero address check - Low

Description: `BonuzSocialId.sol` and `BonuzTokens.sol` contracts, address parameters are used without proper validation against a zero address `(0x0)` check.

This could lead to the assignment of a zero address. For instance `initialize()` in `BonuzSocialId.sol` contract might allow adding a zero address to the `issuers` mapping,

The following parameters are not checked for the zero value:

- BonuzSocialId.sol
  - `initialize()`
    - address[] memory _initialIssuers;
  - `setUserName()`
    - address _user;
  - `setSocialLink()`
    - address _user;
- BonuzTokens.sol
  - `setIssuer()`
    - address _account
  - `setAdmin()`
    - address _account - if 0×0 was set as an owner, the privileged users will be able to fix the issue in a separate transaction.

The impact of this issue was estimated as low. However, It could still disrupt the intended functionality of the smart contract for a limited amount of time.

Path: ./contracts/BonuzSocialId.sol: initialize(), setUserName(), setSocialLink();
./contracts/BonuzTokens.sol: initialize(), setUserName(), setSocialLink();

Found in: 16b25aa

Status: Fixed

Classification

Severity: Low

Impact: 2/5

Likelihood:	1/5

Recommendations

Recommendation: It is recommended to implement a zero address check when an address is passed as a parameter of a function.

Resolution

Zero address checks were introduced.

F-2023-0034 - Missing validation in redeem voucher function - Low

Description: The `redeemVoucher()` function is susceptible to a double spending vulnerability due to the absence of a check to verify whether a voucher has been previously redeemed. Without this validation, an attacker or system error could exploit the oversight, allowing the same voucher to be redeemed multiple times. Double spending, in this context, refers to the unauthorized or unintentional reuse of vouchers, leading to potential financial discrepancies and compromising the integrity of the system. Implementing a validation check is essential to mitigate the risk of double spending and ensure the proper functioning of the voucher redemption process.

Path: ./contracts/BonuzTokens.sol: redeemVoucher();

Found in: 16b25aa

Status: Fixed

Classification

Severity: Low

Impact: 3/5

Likelihood: 1/5

Recommendations

Recommendation: Add a check to verify that the voucher has not been redeemed already.

Code Example:

```
require(_token[tokenId].redeemDate == 0, “The voucher has
already been redeemed”);
```

Resolution

Check was added to the redeem voucher function.

Observation Details

F-2023-0033 - Functions That Can Be Declared External - Info

Description:

In order to save Gas, public functions that are never called in the contract should be declared as external.

`setUserProfile()`, `setSocialLink()`, `setSocialLinks()`, `getUserProfileAndSocialLinks()`, `getIssuer()` and `getAllowedSocialLinks()` can be declared external.

Path: ./contracts/BonuzSocialId.sol: setUserProfile(), setSocialLink(), setSocialLinks(), getUserProfileAndSocialLinks(), getIssuer(), getAllowedSocialLinks() ;

Found in: 16b25aa

Status: Fixed

Recommendations

Recommendation: Use the external attribute for functions never called from the contract.

Resolution

Functions aforementioned functions are declared as external.

F-2023-0079 - Floating Pragma - Info

Description:

The project uses floating pragmas `pragma solidity ^0.8.17.`
This may result in the contracts being deployed using the wrong pragma version, which is different from the one they were tested with. For example, they might be deployed using an outdated pragma version which may include bugs that affect the system negatively.

Path: ./contracts/*

Found in: 16b25aa

Status: Fixed

Recommendations

Recommendation:

Consider locking the pragma version whenever possible and avoid using a floating pragma in the final deployment. Consider known bugs for the compiler version that is chosen.

Resolution

The pragma is now locked to 0.8.17 version.

External References:

- Known Bug

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed based on best industry practices at the time of the writing of this report, with cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The report contains no statements or warranties on the identification of all vulnerabilities and security of the code. The report covers the code submitted and reviewed, so it may not be relevant after any modifications. Do not consider this report as a final and sufficient assessment regarding the utility and safety of the code, bug-free status, or any other contract statements.

While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

English is the original language of the report. The Consultant is not responsible for the correctness of the translated versions.

Technical Disclaimer

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks. Thus, the Consultant cannot guarantee the explicit security of the audited smart contracts.

Appendix 1. Severity Definitions

When auditing smart contracts Hacken is using a risk-based approach that considers the potential impact of any vulnerabilities and the likelihood of them being exploited. The matrix of impact and likelihood is a commonly used tool in risk management to help assess and prioritize risks.

The impact of a vulnerability refers to the potential harm that could result if it were to be exploited. For smart contracts, this could include the loss of funds or assets, unauthorized access or control, or reputational damage.

The likelihood of a vulnerability being exploited is determined by considering the likelihood of an attack occurring, the level of skill or resources required to exploit the vulnerability, and the presence of any mitigating controls that could reduce the likelihood of exploitation.

Severity	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to the loss of user funds or contract state manipulation.
High	High vulnerabilities are usually harder to exploit, requiring specific conditions, or have a more limited scope, but can still lead to the loss of user funds or contract state manipulation.
Medium	Medium vulnerabilities are usually limited to state manipulations and, in most cases, cannot lead to asset loss. Contradictions and requirements violations. Major deviations from best practices are also in this category.
Low	Major deviations from best practices or major Gas inefficiency. These issues will not have a significant impact on code execution, do not affect security score but can affect code quality score.

Appendix 2. Scope

The scope of the project includes the following smart contracts from the provided repository:

Scope Details

Repository	https://github.com/bonuz-market/BonuzSmartContracts
Commit	4f6817d

Contracts in Scope

./contracts/BonuzSocialId.sol

./contracts/BonuzTokens.sol