SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Blockchain Association of Ukraine
Date: March 24th, 2022

Document

Name	Smart Contract Code Review and Security Analysis Report for Blockchain Association of Ukraine.
Approved By	Evgeniy Bezuglyi | SC Department Head at Hacken OU
Type of Contracts	ERC721 token; Launchpad
Platform	EVM
Language	Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Website	https://bau.ai/
Timeline	17.03.2022 - 24.03.2022
Changelog	18.03.2022 - Initial Review 24.03.2022 - Revising

Table of contents

Introduction

Hacken OÜ (Consultant) was contracted by Blockchain Association of Ukraine (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of the Customer's smart contracts.

Scope

The scope of the project is smart contracts in the repository:
Repository:
https://github.com/museum-of-war/nft
Commit:
aa015f7db86f3bd8b55881a3adf4dd6757d5c84e
Technical Documentation: No
JS tests: Yes
Contracts:
./contracts/ERC721XYZ.sol
./contracts/FairXYZMH.sol
./contracts/FairXYZWallets.sol

We have scanned this smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that are considered:

Category	Check Item
Code review	▪ Reentrancy ▪ Ownership Takeover ▪ Timestamp Dependence ▪ Gas Limit and Loops ▪ Transaction-Ordering Dependence ▪ Style guide violation ▪ EIP standards violation ▪ Unchecked external call ▪ Unchecked math ▪ Unsafe type inference ▪ Implicit visibility level ▪ Deployment Consistency ▪ Repository Consistency

Functional review	▪ Business Logics Review ▪ Functionality Checks ▪ Access Control & Authorization ▪ Escrow manipulation ▪ Token Supply manipulation ▪ Assets integrity ▪ User Balances manipulation ▪ Data Consistency ▪ Kill-Switch Mechanism

Executive Summary

Score measurements details can be found in the corresponding section of the methodology.

Documentation quality

The Customer did not provide any documentation other than some comments in the code. The total Documentation Quality score is **1** out of **10**.

Code quality

The total CodeQuality score is **10** out of **10**. Unit tests were provided.

Architecture quality

The architecture quality score is **10** out of **10**. The project has clean and clear architecture and uses best practices.

Security score

As a result of the audit, security engineers found **3** low severity issues.

As a result of the revision, security engineers found **no** new severity issues. All previously reported issues have been fixed.

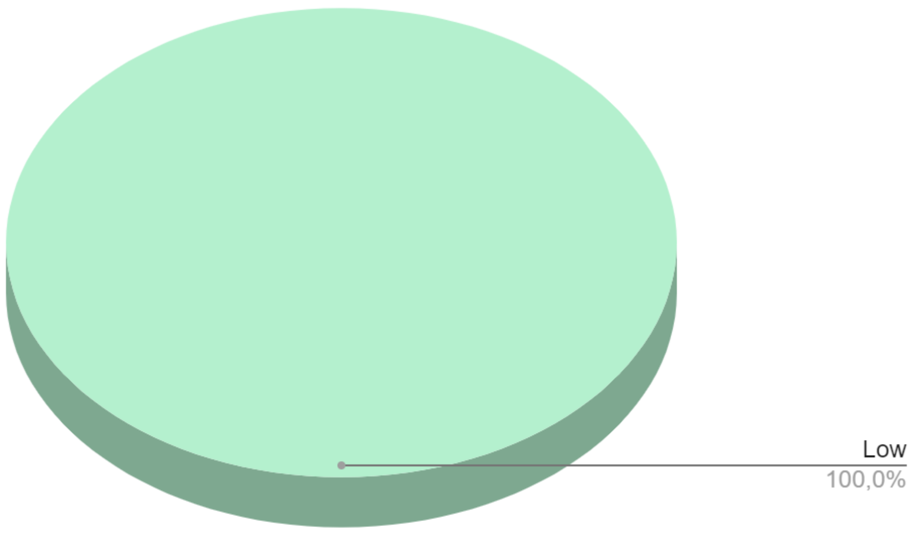
The security score is **10** out of **10**. All found issues are displayed in the "Issues overview" section.

Summary

According to the assessment, the Customer's smart contract has the following score: **9.1**

1
2
3
4
5
6
7
8
9
10
You are here

Graph 1. The distribution of vulnerabilities after the audit.



Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to assets loss or data manipulations.
High	High-level vulnerabilities are difficult to exploit; however, they also have a significant impact on smart contract execution, e.g., public access to crucial functions
Medium	Medium-level vulnerabilities are important to fix; however, they cannot lead to assets loss or data manipulations.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that cannot have a significant impact on execution

Findings

🟥🟥🟥🟥 Critical

No critical severity issues were found.

🟨🟨🟨 High

No high severity issues were found.

🟩🟩 Medium

No medium severity issues were found

🟦 Low

1. **State variables that could be declared immutable.**

   Constant state variables should be declared constant to save gas.

   Contract: FairXYZMH.sol

   Variables: ukraineAddress

   Recommendation: Add the **immutable** attribute to state variables that never change.

   Status: Fixed

2. **Different naming styles.**

   Few functions have different naming styles, for example, balanceOf and view_minted.

   Contract: ERC721xyz.sol

   Recommendation: Follow the official style guides.

   Status: Fixed

3. **Floating pragma**

   The ERC721xyz.sol, FairXYZMH.sol, and FairXYZWallets.sol contracts use floating pragma ^0.8.0 and ^0.8.7

   Contracts: ERC721xyz.sol, FairXYZMH.sol, FairXYZWallets.sol

   Recommendation: Consider locking the pragma version whenever possible and avoid using a floating pragma in the final deployment.

   Status: Fixed

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed by the best industry practices at the date 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 audit makes no statements or warranties on the security of the code. It also cannot be considered a 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 it 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.

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 audit cannot guarantee the explicit security of the audited smart contracts.