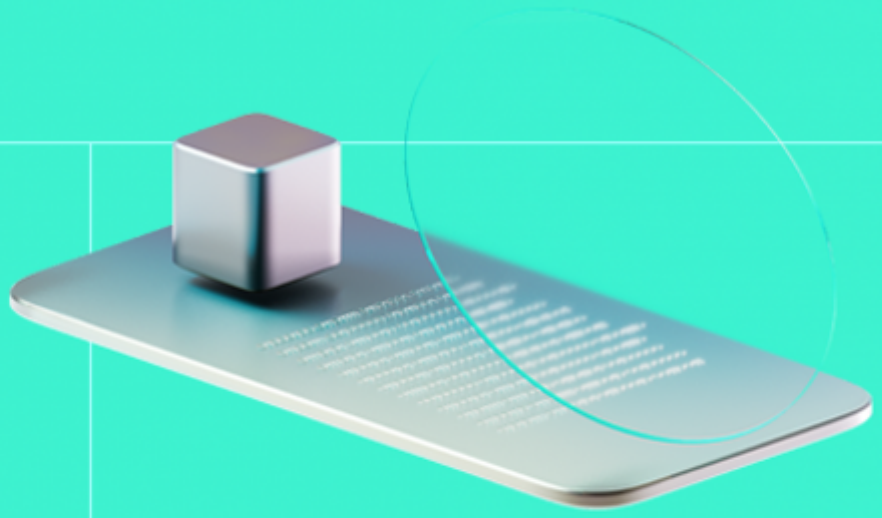
HACKEN

Smart Contract Code Review And Security Analysis Report



Customer: Ethereum Towers

Date: 04/07/2022

We express our gratitude to the Ethereum Towers team for the collaborative engagement that enabled the execution of this Smart Contract Security Assessment.

Ethereum Towers is a community-centric, vertical megastructure consisting of 4,388 resident-owned apartments and a variety of communal areas, set in the forthcoming Ethereum Worlds Metaverse.

Platform: EVM

Language: Solidity

Tags: Staking

Timeline: 16/06/2022 - 04/07/2022

Methodology: https://hackenio.cc/sc_methodology

Review Scope

Repository	https://github.com/ethereumtowers/contracts
Commit	94eb48031a02455bb3c48285ffe41fbbe3498079

Audit Summary

10/10	9/10	10/10	10/10
Security score	Code quality score	Architecture quality score	Documentation quality score

Total 9.9/10

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

6	5	0	0
Total Findings	Resolved	Accepted	Mitigated

Findings by severity

Critical	0
High	1
Medium	1
Low	4

Vulnerability	Status
F-2022-1589 - Inconsistent contract state	Fixed
F-2022-1590 - Unused renting logic	Fixed
F-2022-1591 - Variable Shadowing	Fixed
F-2022-1592 - Missing events arithmetic	Fixed
F-2022-1593 - Zero address is allowed	Fixed
F-2022-1594 - Redundant SafeCast library	Pending Fix

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 Ethereum Towers
Audited By	Hacken
Website	https://ethereumtowers.com
Changelog	17/06/2022 - Initial Review 04/07/2022 - Second Review

Table of Contents

System Overview

Ethereum Towers is a community-centric, vertical megastructure consisting of 4,388 resident-owned apartments and a variety of communal areas, set in the forthcoming Ethereum Worlds Metaverse with the following contracts:

- EthereumWorldsNFTStaking — a contract that rewards users for staking their NFTs. Rewards are calculated off-chain.

Privileged roles

- The owner of the EthereumWorldsNFTStaking contract can pause staking, unstacking, and claiming in case of emergency.
- The owner of the EthereumWorldsNFTStaking contract can update the service signer address.
- The owner of the EthereumWorldsNFTStaking contract can update the max amount of tokens in staking.
- The owner of the EthereumWorldsNFTStaking contract can withdraw ERC20 tokens sent to the contract.

Executive Summary

This report presents an in-depth analysis and scoring of the customer's smart contract project. Detailed scoring criteria can be referenced in the scoring methodology.

Documentation quality

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

- Functional and technical requirements are provided.
- Whitepaper describes the project adequately.

Code quality

The total Code quality score is **9** out of **10**.

- Code style and naming conventions are respected.
- The “should restrict calling stake for non-existing token” test is failing and should be fixed.

Architecture quality

The Architecture quality score is **10** out of **10**.

- Contracts follow single responsibility principles.
- Code is well-formatted.

Security score

Upon auditing, the code was found to contain **0** critical, **1** high, **1** medium, and **4** low severity issues, leading to a security score of **10** out of **10**.

All identified issues are detailed in the “Findings” section of this report.

Summary

The comprehensive audit of the customer's smart contract yields an overall score of **9.9**. This score reflects the combined evaluation of documentation, code quality, architecture quality, and security aspects of the project.

Risks

- In case of an admin keys leak, an attacker can lock contract functionality.
- In case the contract will not be funded with world tokens - it would not be possible to claim rewards.

Findings

Vulnerability Details

F-2022-1589 - Inconsistent contract state - High

Description:

`tokensInStake` variable was calculated incorrectly, which could lead to staking more tokens to the contract than expected.

During `emergencyUnstake` contract decides which amount to unstake, requested by the user, or `maxTokensPerUnstake`, to prevent the out of Gas exception. Contract ignores which value was selected to unstake and always decreases `tokensInStake` value by `ids.length`.

Assets:

- ./contracts/staking/EthereumWorldsNFTStaking.sol [https://github.com/ethereumtowers/contracts]

Status: Fixed

Classification

Severity: High

Recommendations

Remediation:

Use `unstakeAmount` instead of `ids_._length` when decreasing `tokensInStake`.

F-2022-1590 - Unused renting logic - Medium

Description: The contract contains unused renting logic. Considering that the contract is not upgradable - it is impossible to be sure that this logic would not be broken (for example, a user can unstake a token at any time, and it is unclear how it will affect renting logic).

Assets:

- ./contracts/staking/EthereumWorldsNFTStaking.sol [https://github.com/ethereumtowers/contracts]

Status: Fixed

Classification

Severity: Medium

Recommendations

Remediation: Consider deleting this functionality or provide documentation on how it will be implemented.

F-2022-1591 - Variable Shadowing - Low

Description: Solidity allows for ambiguous naming of state variables when inheritance is used. Contract A with a variable x could inherit contract B, which has a state variable x defined. This would result in two separate versions of x, accessed from contract A and the other from contract B. In more complex contract systems, this condition could go unnoticed and subsequently lead to security issues.

Assets:

- ./contracts/staking/EthereumWorldsNFTStaking.sol [https://github.com/ethereumtowers/contracts]

Status: Fixed

Classification

Severity: Low

Recommendations

Remediation: Consider renaming the function argument.

F-2022-1592 - Missing events arithmetic - Low

Description: To simplify off-chain changes tracking, it is recommended to emit events when a crucial part of the contract changes.

Assets:

- ./contracts/staking/EthereumWorldsNFTStaking.sol [https://github.com/ethereumtowers/contracts]

Status: Fixed

Classification

Severity: Low

Recommendations

Remediation: Emit an event for critical parameter changes.

F-2022-1593 - Zero address is allowed - Low

Description: Zero address is allowed.

Assets:

- ./contracts/staking/EthereumWorldsNFTStaking.sol [https://github.com/ethereumtowers/contracts]

Status: Fixed

Classification

Severity: Low

Recommendations

Remediation: Add a check for zero address for `_serviceSigner`.

F-2022-1594 - Redundant SafeCast library - Low

Description: The contract is using the SafeCast library to cast uint256 to uint32 and uint224. The usage of the library is redundant in this contract, as all variables could not be overflowed. To save Gas and simplify the code, it is recommended to remove the library.

Assets:

- ./contracts/staking/EthereumWorldsNFTStaking.sol [https://github.com/ethereumtowers/contracts]

Status: Pending Fix

Classification

Severity: Low

Recommendations

Remediation: Remove SafeCast library.

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 **Likelihood**, **Impact**, **Exploitability** and **Complexity** metrics to evaluate findings and score severities.

Reference on how risk scoring is done is available through the repository in our Github organization:

hknio/severity-formula

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/ethereumtowers/contracts
Commit	94eb48031a02455bb3c48285ffe41fbbe3498079
Whitepaper	Provided
Requirements	Provided
Technical Requirements	Provided

Contracts in Scope

./contracts/staking/EthereumWorldsNFTStaking.sol