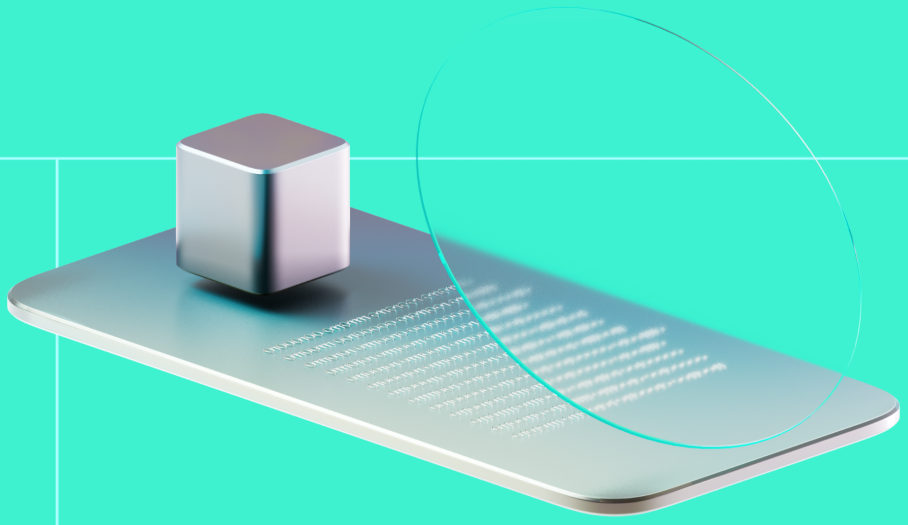
HACKEN

Smart Contract Code Review And Security Analysis Report



Customer: Zharta

Date: 11 Dec, 2023

Hacken OU
Parda 4, Kesklinn, Tallinn
10151 Harju Maakond, Eesti
Kesklinna, Estonia

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

Zharta is a platform that provides instant loans & renting services using NFTs as collateral, enabling NFT holders to get liquidity & yield without having to sell their NFTs.

Platform: EVM

Language: Vyper

Tags: NFT Renting, ERC721, Escrow, Delegation

Timeline: 30.11.2023 - 11.12.2023

Methodology: Link

Last review scope

Repository	https://github.com/Zharta/lotm-renting-protocol-v1
Commit	d455048

View full scope

Hacken OU
Parda 4, Kesklinn, Tallinn
10151 Harju Maakond, Eesti
Kesklinna, Estonia

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	Acknowledged	Mitigated

Findings by severity	Findings Number	Resolved	Mitigated	Acknowledged
Critical	0	0	0	0
High	0	0	0	0
Medium	0	0	0	0
Low	2	2	0	0

Hacken OU
Parda 4, Kesklinn, Tallinn
10151 Harju Maakond, Eesti
Kesklinna, Estonia

Document

Name	Smart Contract Code Review and Security Analysis Report for Zharta
Audited By	David Camps Novi | SC Lead Auditor at Hacken OÜ Viktor Lavrenenko | SC Auditor at Hacken OÜ
Approved By	Przemyslaw Swiatowiec | SC Audits Expert at Hacken OÜ
Website	https://www.zharta.io/
Changelog	06.12.2023 – Preliminary Report 11.12.2023 - Second Review

Hacken OU
Parda 4, Kesklinn, Tallinn
10151 Harju Maakond, Eesti
Kesklinna, Estonia

Introduction

Hacken OÜ (Consultant) was contracted by Zharta (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.

System Overview

The LOTM Renting Protocol is a trustless way for LOTM players to rent game assets from users.

The NFT renting process in this protocol is as follows:

1. A lender deposits NFT in a vault, setting the lending terms (price + rental duration).
2. A borrower gets the NFT they wish, by paying the rental price upfront.
3. The vault containing the NFT delegates it to the borrower's specific wallet for a specific duration.
4. Once the loan duration is reached, the rental finishes.
    a. The borrower can end the rental before the end date, recovering the funds corresponding to the period unused.

Fees are defined in *Renting.vy* as *protocol_fee* and *protocol_wallet*, which are used when a new rental is created through *Vault.start_rental*.

Protocol contracts:

- Vault - holds the core functionality of the NFT lending.
- Renting - entry point for users and other protocols.
    - Contains the admin protocol methods to update the protocol fee and the protocol admin.

Privileged roles

Renting.vy:

- Admin - sets the protocol fees and the protocol wallet that receives those fees.
    - Admin can be changed via *propose_admin* and *claim_ownership* functions.

Vault.py

- Owner - the owner of each NFT in the vault. Only this address can interact with the vault through the *Renting.vy* contract.

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:
    - Project overview is provided.
    - Use cases are described and detailed.
    - All the project's features are mentioned.
    - Roles and authorization are described.
- Technical description is sufficient:
    - Technical specifications are present.
    - Deployment instructions are provided.

Code quality

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

- The code maintains a high standard.
- Best practices are followed.

Test coverage

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

- Deployment and basic user interactions are covered with tests.
- Negative case coverage is present.
- Interactions by several users are tested.

Security score

As a result of the audit, the code contains no issues. The security score is **10** out of **10**.

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

Summary

According to the assessment, the Customer's smart contract has the following score: **10**. The system users should acknowledge all the risks summed up in the risks section of the report.

Risks

- The Vault contract calls ERC20’s methods *transfer* and *transferFrom*, checking the return value. However, if a token were used that does not revert or return at all, functions including the aforementioned calls would fail.

- Renting time can be set to unlimited, resulting in the loss of control over the rented NFT.

- When a *renter* sets an allowance higher than the original *listing.price*, there is a risk that the *nft_owner* can frontrun the *renter* by changing the *listing.price* via *set_listings()* function.
- Interactions with *delegation_registry_addr* (warm.xyz) are out of scope of this audit and cannot be validated (see the following risk statement).

Risk Statement

This audit report focuses exclusively on the security assessment of the contracts within the specified review scope. Interactions with out-of-scope contracts are presumed to be correct and are not examined in this audit. We want to highlight that Interactions with contracts outside the specified scope, such as:

/contracts/Renting.vy: delegation_registry_addr

/contracts/Vault.vy: delegation_registry_addr

have not been verified or assessed as part of this report.

While we have diligently identified and mitigated potential security risks within the defined scope, it is important to note that our assessment is confined to the isolated contracts within this scope. The overall security of the entire system, including external contracts and integrations beyond our audit scope, cannot be guaranteed.

Users and stakeholders are urged to exercise caution when assessing the security of the broader ecosystem and interactions with external contracts. For a comprehensive evaluation of the entire system, additional audits and assessments outside the scope of this report are necessary.

This report serves as a snapshot of the security status of the audited contracts within the specified scope at the time of the audit. We strongly recommend ongoing security evaluations and continuous monitoring to maintain and enhance the overall system's security.

Hacken OU
Parda 4, Kesklinn, Tallinn
10151 Harju Maakond, Eesti
Kesklinna, Estonia

Findings

Critical

No critical severity issues were found.

High

No high severity issues were found.

Medium

No medium severity issues were found.

Low

L01. Missing Maximum Protocol Fee Check

Impact	Low
Likelihood	Medium

The *protocol_fee* can be set higher than the maximum protocol fee on the contract deployment (*__init__* function).

Path: ./contracts/Renting.vy : __init__()

Recommendation: Implement a check in the *__init__()* function to make sure the *protocol_fee* cannot surpass *_max_protocol_fee*.

Found in: 8c001bb

Status: Fixed (Revised commit: d455048)

Remediation: A check was added into the *__init__()* function to prevent the *protocol_fee* from surpassing *_max_protocol_fee*.

L02. Missing Zero Address Checks

Impact	Medium
Likelihood	Low

The contract currently utilizes certain address parameters without implementing checks for the 0×0 address case. Given that these parameters are immutable, setting any of them to 0×0 would render the contract inoperable. This oversight not only affects the functionality but also results in wasted Gas during deployment, as the contract cannot be used once deployed with a 0×0 address parameter.

Path: ./contracts/Renting.vy : __init__() → _vault_impl_addr, _payment_token_addr, _nft_contract_addr, _delegation_registry_addr.

Recommendation: Implement zero address checks for reported parameters.

Found in: 8c001bb

Status: Fixed (Revised commit: d455048)

Remediation: Zero address checks were added into the *__init__()* function.

Informational

I01. Unused Parameter

The input parameter *sender* is not used in *_set_listing()*.

Unused code impacts code clarity and makes integration harder for other systems.

Path: ./contracts/Vault.vy : _set_listing().

Recommendation: Unused parameters should be removed from the code.

Found in: 8c001bb

Status: Fixed (Revised commit: d455048)

Remediation: The *sender* parameter was removed from *_set_listing()*.

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.

Risk Level	High Impact	Medium Impact	Low Impact
High Likelihood	Critical	High	Medium
Medium Likelihood	High	Medium	Low
Low Likelihood	Medium	Low	Low

Risk Levels

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.

Impact Levels

High Impact: Risks that have a high impact are associated with financial losses, reputational damage, or major alterations to contract state. High impact issues typically involve invalid calculations, denial of service, token supply manipulation, and data consistency, but are not limited to those categories.

Medium Impact: Risks that have a medium impact could result in financial losses, reputational damage, or minor contract state manipulation. These risks can also be associated with undocumented behavior or violations of requirements.

Low Impact: Risks that have a low impact cannot lead to financial losses or state manipulation. These risks are typically related to unscalable functionality, contradictions, inconsistent data, or major violations of best practices.

Likelihood Levels

High Likelihood: Risks that have a high likelihood are those that are expected to occur frequently or are very likely to occur. These risks could be the result of known vulnerabilities or weaknesses in the contract, or could be the result of external factors such as attacks or exploits targeting similar contracts.

Medium Likelihood: Risks that have a medium likelihood are those that are possible but not as likely to occur as those in the high likelihood category. These risks could be the result of less severe vulnerabilities or weaknesses in the contract, or could be the result of less targeted attacks or exploits.

Low Likelihood: Risks that have a low likelihood are those that are unlikely to occur, but still possible. These risks could be the result of very specific or complex vulnerabilities or weaknesses in the contract, or could be the result of highly targeted attacks or exploits.

Informational

Informational issues are mostly connected to violations of best practices, typos in code, violations of code style, and dead or redundant code.

Informational issues are not affecting the score, but addressing them will be beneficial for the project.

Appendix 2. Scope

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

Scope details

Repository	https://github.com/Zharta/lotm-renting-protocol-v1
Commit	d455048
Whitepaper	Whitepaper
Requirements	README.md
Technical Requirements	README.md

Contracts in Scope

./contracts/Renting.vy
./contracts/Vault.vy