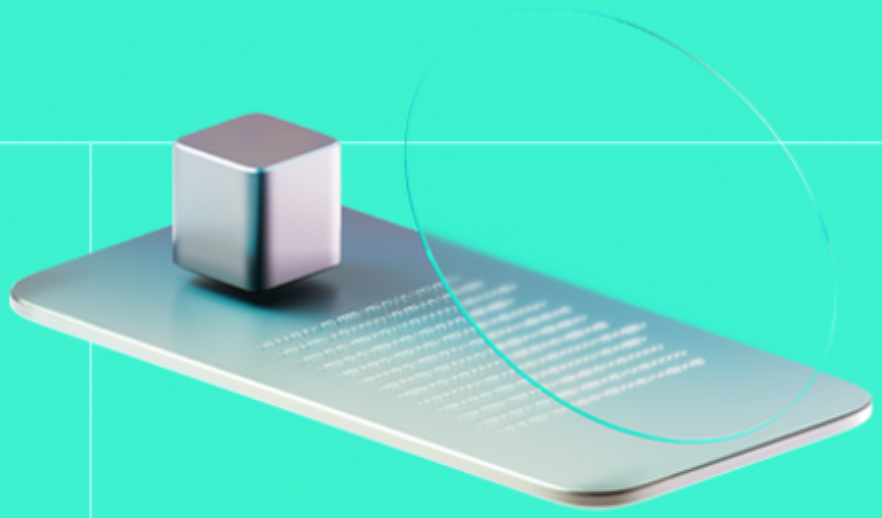
HACKEN

Smart Contract Code Review And Security Analysis Report



Customer: Vatra Inu

Date: 04/03/2024

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

Vatra INU is a ERC20 based community/meme token project. It lets token holders to become a community and grow together.

Platform: EVM

Language: Solidity

Tags: ERC20

Timeline: 20/02/2024 - 21/02/2024

Methodology: https://hackenio.cc/sc_methodology

Review

Scope

Repository	https://github.com/VATRA-INU-Project/vatra-inu-contract
Commit	8f8c3be8b2c95297f523b42b99d17f0c742363d4
Deployed address	https://etherscan.io/address/0×3d234A9d23F01c5556AD3dfA88F470f8982ab1b4

Audit Summary

10/10	10/10	100%	9/10
Security Score	Code quality score	Test coverage	Documentation quality score

Total 9.9/10

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

1	0	0	0
Total Findings	Resolved	Accepted	Mitigated

Findings by severity

Critical	0
High	0
Medium	0
Low	0

Vulnerability	Status
F-2024-0956 - Floating Solidity Pragma Version	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 Vatra Inu
Audited By	Bohdan Pukhno
Approved By	Yves Toiser
Website	https://vatrainu.com/en
Changelog	21/02/2024 - Preliminary Report

Table of Contents

System Overview

Vatra Inu — simple ERC-20 token that mints all initial supply to a deployer. Additional minting is not allowed. Burnable functionality is available.

It has the following attributes:

- Name: VATRA INU
- Symbol: VATR
- Decimals: 18
- Total supply: 250 000 000 tokens.

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 **9** out of **10**.

- Functional requirements are provided.
- Technical requirements are provided.
  - NatSpec comments are not fully provided.

Code quality

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

- The development environment is configured for the Remix compiler.

Test coverage

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

Security score

Upon auditing, the code was found to contain **0** critical, **0** high, **0** medium, and **0** 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, test coverage, and security aspects of the project.

Risks

- The version of Solidity used in this project might not work on all chains, due to the opcode push0, however it is supported by the chain it is intended to be used on.

Findings

Vulnerability Details

F-2024-0956 - Floating Solidity Pragma Version - Info

Description: The project uses floating pragmas ^0.8.20.

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.

Assets:

- vatrainu.sol [https://github.com/VATRA-INU-Project/vatra-inu-contract]

Status: Pending Fix

Classification

Severity: Info

Impact: 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.

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.

External References:

- Known bugs

Observation Details

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/VATRA-INU-Project/vatra-inu-contract
Commit	8f8c3be8b2c95297f523b42b99d17f0c742363d4
Whitepaper	https://github.com/VATRA-INU-Project/vatra-inu-contract/blob/main/documentation.pdf
Requirements	https://github.com/VATRA-INU-Project/vatra-inu-contract/blob/main/documentation.pdf
Technical Requirements	https://github.com/VATRA-INU-Project/vatra-inu-contract/blob/main/documentation.pdf

Contracts in Scope

./vatrainu.sol