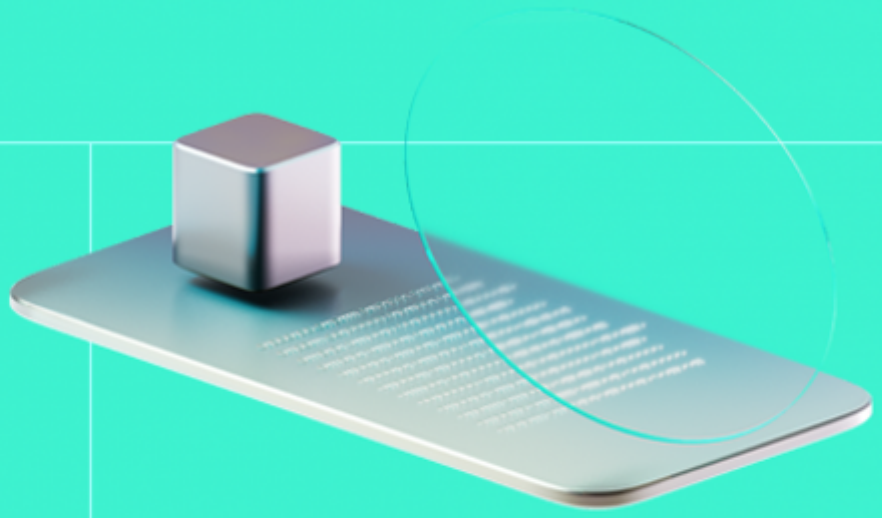
HACKEN

Smart Contract Code Review And Security Analysis Report



Customer: Wormfare

Date: 30/10/2024

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

The WOFR project introduces a fixed-supply ERC20 token with enhanced features like off-chain approval (ERC20Permit) and burn capability, designed to facilitate secure and flexible token management.

Document

Name	Smart Contract Code Review and Security Analysis Report for Wormfare
Audited By	Przemyslaw Swiatowiec
Approved By	Ataberk Yavuzer
Website	https://wormfare.com/
Changelog	30/10/2024 - Final Report
Platform	Polygon
Language	Solidity
Tags	ERC20
Methodology	https://hackenio.cc/sc_methodology

Review Scope

Repository	https://github.com/wormfare
Commit	0a580bd1f44230eb523c8d9e1f21286582551f37

Audit Summary

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

0	0	0	0
Total Findings	Resolved	Accepted	Mitigated

Findings by Severity

Severity	Count
Critical	0
High	0
Medium	0
Low	0

Documentation quality

- Functional requirements are provided.
- Technical description is provided.

Code quality

- No code quality issues were found.

Test coverage

Code coverage of the project is **100%.**

Table of Contents

System Overview **6**

Privileged Roles 6

Potential Risks **7**

Findings **8**

Disclaimers 9

Appendix 1. Definitions **10**

Severities 10

Potential Risks 10

Appendix 2. Scope **11**

Appendix 3. Additional Valuables **12**

System Overview

This contract deploys the WOFR ERC20 token with a fixed supply of 300 million tokens, minting the full supply to a specified owner address. It includes OpenZeppelin ERC20Permit for off-chain approvals and ERC20Burnable for token burning. The contract reverts if a zero address is provided as the owner.

Privileged roles

- There are no privileged roles.

Potential Risks

- **Centralized Control of Token Supply:** Minting the entire token supply to a single address grants full control to one entity, posing risks of market manipulation or instability

Findings

No issues were found.

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. Definitions

Severities

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.

Potential Risks

The "Potential Risks" section identifies issues that are not direct security vulnerabilities but could still affect the project’s performance, reliability, or user trust. These risks arise from design choices, architectural decisions, or operational practices that, while not immediately exploitable, may lead to problems under certain conditions. Additionally, potential risks can impact the quality of the audit itself, as they may involve external factors or components beyond the scope of the audit, leading to incomplete assessments or oversight of key areas. This section aims to provide a broader perspective on factors that could affect the project's long-term security, functionality, and the comprehensiveness of the audit findings.

Appendix 2. Scope

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

Scope Details	
Repository	https://github.com/wormfare
Commit	0a580bd1f44230eb523c8d9e1f21286582551f37
Whitepaper	Contract NATSpec
Requirements	Contract NATSpec
Technical Requirements	Contract NATSpec

Asset	Type
contracts/WOFR.sol [https://github.com/wormfare/contracts/blob/main/contracts/WOFR.sol]	Smart Contract

Appendix 3. Additional Valuables

Additional Recommendations

The smart contracts in the scope of this audit could benefit from the introduction of automatic emergency actions for critical activities, such as unauthorized operations like ownership changes or proxy upgrades, as well as unexpected fund manipulations, including large withdrawals or minting events. Adding such mechanisms would enable the protocol to react automatically to unusual activity, ensuring that the contract remains secure and functions as intended.

To improve functionality, these emergency actions could be designed to trigger under specific conditions, such as:

- Detecting changes to ownership or critical permissions.
- Monitoring large or unexpected transactions and minting events.
- Pausing operations when irregularities are identified.

These enhancements would provide an added layer of security, making the contract more robust and better equipped to handle unexpected situations while maintaining smooth operations.