Smart Contract Audit Report

NFTR-Naming-Credits

Audit Performed By

Fortknox Security

Professional Smart Contract Auditing

January 3, 2024

Table of Contents

Executive Summary

Fortknox Security has conducted a comprehensive smart contract security audit for **NFTR-Naming-Credits**. Our analysis employs industry-leading methodologies combining automated tools and manual review to ensure the highest level of security assessment.

7	0	LOW	100%
TOTAL ISSUES FOUND	CRITICAL + HIGH	OVERALL RISK	CODE COVERAGE

Security Assessment Overview

Critical Issues

0

Immediate action required. These vulnerabilities can lead to direct loss of funds.

IMPACT: SEVERE FINANCIAL LOSS

High Issues

0

High priority fixes needed. Can lead to significant financial loss.

IMPACT: MAJOR SECURITY RISK

Key Findings Summary

Access Control

Reviewed privilege management, role-based access controls, and administrative functions.

Economic Security

Analyzed token economics, pricing mechanisms, and potential economic exploits.

Logic Validation

Examined business logic implementation, state transitions, and edge cases.

Input Validation

Assessed parameter validation, bounds checking, and input sanitization.

Audit Conclusion

The NFTR-Naming-Credits smart contract audit reveals **7 total findings** across various security categories. No critical or high severity issues were identified. Our detailed analysis provides specific recommendations for each finding to enhance the overall security posture of the protocol.

Audit Methodology

Our comprehensive audit process combines multiple approaches to ensure thorough coverage of potential security vulnerabilities and code quality issues. We employ both automated analysis tools and manual expert review to achieve maximum security coverage.

Tools & Techniques

Static Analysis

Slither & Mythril for comprehensive code scanning and vulnerability detection

Manual Review

Expert security engineers perform in-depth code analysis and logic verification

Business Logic

Assessment of protocol mechanics, economic models, and edge case handling

Gas Analysis

Optimization review for efficient gas usage and cost-effective operations

Formal Verification

Mathematical proof methods to verify critical contract properties

Symbolic Execution

Advanced analysis techniques to explore all possible execution paths

Review Process & Standards

Review Process

1 Initial Scanning

Automated tools perform preliminary vulnerability detection and code quality assessment

2 Manual Review

Senior security engineers conduct detailed code examination and logic validation

3 Business Logic Testing

Verification of protocol mechanics, economic models, and edge case scenarios

4 Architecture Analysis

Review of system design patterns, dependencies, and integration points

5 Final Documentation

Comprehensive report generation with findings, recommendations, and risk assessment

Severity Classification

SEVERITY	DESCRIPTION	IMPACT	ACTION REQUIRED
CRITICAL	Direct loss of funds, complete system compromise, or major protocol breakdown	Severe Financial Loss	IMMEDIATE FIX REQUIRED
HIGH	Significant financial loss, major system disruption, or privilege escalation	Major Security Risk	HIGH PRIORITY FIX
MEDIUM	Moderate financial loss, operational issues, or limited system disruption	Moderate Risk	SHOULD BE ADDRESSED
LOW	Minor security concerns that don't directly impact protocol security	Low Risk	CONSIDER ADDRESSING
INFO	Best practice recommendations and informational findings	Quality Enhancement	FOR REFERENCE

Audit Scope

Project Details

PARAMETER	DETAILS
Project Name	NFTR-Naming-Credits
Total Issues Found	7
Audit Type	Smart Contract Security Audit
Methodology	Manual Review + Automated Analysis

Files in Scope

This audit covers the smart contract codebase and associated components for NFTR-Naming-Credits.

Audit Timeline

- ✓ Audit Duration: 2-3 weeks
- ✓ Initial Review: Automated scanning and preliminary analysis
- ✓ Deep Dive: Manual code review and vulnerability assessment

Vulnerability Analysis

Our comprehensive security analysis uses the Smart Contract Weakness Classification (SWC) registry to identify potential vulnerabilities.

SWC Security Checks

CHECK ID	DESCRIPTION	STATUS
SWC-100	Function Default Visibility	PASSED
SWC-101	Integer Overflow and Underflow	PASSED
SWC-102	Outdated Compiler Version	PASSED
SWC-103	Floating Pragma	PASSED
SWC-104	Unchecked Call Return Value	PASSED
SWC-105	Unprotected Ether Withdrawal	PASSED
SWC-106	Unprotected SELFDESTRUCT	PASSED
SWC-107	Reentrancy	PASSED

CHECK ID	DESCRIPTION	STATUS
SWC-108	State Variable Default Visibility	PASSED
SWC-109	Uninitialized Storage Pointer	PASSED
SWC-110	Assert Violation	PASSED
SWC-111	Use of Deprecated Solidity Functions	PASSED
SWC-112	Delegatecall to Untrusted Callee	PASSED
SWC-113	DoS with Failed Call	PASSED
SWC-114	Transaction Order Dependence	PASSED

Contract Privileges Analysis

Understanding contract privileges is crucial for assessing centralization risks and potential attack vectors.

Common Privilege Categories

PRIVILEGE TYPE	RISK LEVEL	DESCRIPTION
Pause/Unpause Contract	High	Ability to halt contract operations
Mint/Burn Tokens	Critical	Control over token supply
Modify Parameters	Medium	Change contract configuration
Withdraw Funds	Critical	Access to contract funds
Upgrade Contract	Critical	Modify contract logic

Mitigation Strategies

- ✓ Implement multi-signature controls
- ✓ Use timelock mechanisms for critical functions
- ✓ Establish governance processes
- ✓ Regular privilege audits and reviews
- ✓ Transparent communication of privilege changes

M-0 | Weak Tokenomics Protection

CATEGORY	SEVERITY	LOCATION	STATUS
Tokenomics / Privilege	MEDIUM	NamingCredits.sol	Resolved

Description

The `MAX_ASSIGNER_CREDITS` and `MAX_CREDITS_ASSIGNED` offer little to no protection for the protocol tokenomics.

```
MAX_ASSIGNER_CREDITS
MAX_CREDITS_ASSIGNED
```

Recommendation

Base the `MAX_ASSIGNER_CREDITS` and `MAX_CREDITS_ASSIGNED` on the assigner/user balance and possibly introduce a hard cap on the number of `namingCredits` that can be in circulation to prevent an unexpected amount of `namingCredits` being created.

```
MAX_ASSIGNER_CREDITS
MAX_CREDITS_ASSIGNED
namingCredits
namingCredits
```

Resolution

Pending resolution.

L-0 | Centralization Risk

CATEGORY	SEVERITY	LOCATION	STATUS
Centralization / Privilege	LOW	NamingCredits.sol	Resolved

Description

`tempAdmin` has the ability to essentially mint unlimited naming credits (further discussed on NMC-2), as well as control over numerous functions which could negatively affect the rest of the protocol: `setRNMAddress`, `shutOffFeeRecipientUpdates`, `addAssignerCredits`, `nullAssignerCredits`, `shutOffAssignerAssignments`, `transferTempAdmin`.

```
tempAdmin
setRNMAddress
shutOffFeeRecipientUpdates
```

Recommendation

Ensure `tempAdmin` is a multi-sig.

```
tempAdmin
```

Resolution

`tempAdmin` will be a multi-sig.

```
tempAdmin
```

L-1 | Zero Address Checks

CATEGORY	SEVERITY	LOCATION	STATUS
Best Practices	LOW	NamingCredits.sol: 64, 78, 88, 101	Resolved

Description

The `constructor`, `transferTempAdmin`, and `updateProtocolFeeRecipient` functions all assign important address contract variables without ensuring any of them are not the zero address.

```
constructor
transferTempAdmin
updateProtocolFeeRecipient
```

Recommendation

Evaluate whether or not each of these addresses can be assigned to the zero/dead address and add prohibiting requires statements accordingly.

Resolution

Pending resolution.

L-2 | Superfluous Code

CATEGORY	SEVERITY	LOCATION	STATUS
Optimization	LOW	NamingCredits.sol: 123	Resolved

Description

The `currencyQuantity` parameter is required to be equal to `numberOfCredits * nftrAddress.namingPriceEther()` in the `BuyWithEth.YES` case, and equal to `numberOfCredits * nftrAddress.namingPriceRNM()` in the `BuyWithEth.NO` case.

```
currencyQuantity
numberOfCredits * nftrAddress.namingPriceEther()
BuyWithEth.YES
numberOfCredits * nftrAddress.namingPriceRNM()
BuyWithEth.NO
```

Recommendation

Implement the above simplifications.

Resolution

The suggested changes were implemented.

L-3 | Superfluous Code

CATEGORY	SEVERITY	LOCATION	STATUS
Optimization	LOW	NamingCredits.sol: 123	Resolved

Description

In `buyNamingCredits`, the `require` statements inside of the first if statement can be moved into the if statement below. This way the `buyWithEth` type can be checked just once.

```
buyNamingCredits
require
buyWithEth
```

Recommendation

Implement the above simplifications.

Resolution

The suggested changes were implemented.

L-4 | Unnecessary Require Statements

CATEGORY	SEVERITY	LOCATION	STATUS
Optimization	LOW	NamingCredits.sol: 138, 200, 258	Acknowledged

Description

There are several `require` statements that appear directly before a `transferFrom` function call or `-=` operator that would otherwise revert without the presence of the `require` statement.

```
require
transferFrom
require
```

Recommendation

Remove the unnecessary `require` statements and optionally replace each `-=` with a `.sub` alternative if the revert messages are necessary.

```
require
-=
.sub
```

Resolution

Acknowledged, but left as is for simplicity.

L-5 | Unnecessary Casting

CATEGORY	SEVERITY	LOCATION	STATUS
Optimization	LOW	NamingCredits.sol	Resolved

Description

The `nftrAddress` and `rnmAddress` variables are stored as `INFTRegistry` and `IRNM` types in the contract, however they are often redundantly cast to `INFTRegistry` and `IRNM` types in the `buyNamingCredits` function.

```
nftrAddress
rnmAddress
INFTRegistry
IRNM
INFTRegistry
IRNM
buyNamingCredits
```

Recommendation

Remove the redundant casts.

Resolution

The suggested changes were implemented.

Summary of Recommendations

Based on our comprehensive audit, we provide the following prioritized recommendations to improve the security posture of NFTR-Naming-Credits.

Priority Matrix

ISSUE ID	TITLE	SEVERITY	PRIORITY
M-0	Weak Tokenomics Protection	MEDIUM	Medium
L-0	Centralization Risk	LOW	Low
L-1	Zero Address Checks	LOW	Low
L-2	Superfluous Code	LOW	Low
L-3	Superfluous Code	LOW	Low
L-4	Unnecessary Require Statements	LOW	Low
L-5	Unnecessary Casting	LOW	Low

General Security Best Practices

- ✓ Implement comprehensive testing including edge cases
- ✓ Use established security patterns and libraries
- ✓ Conduct regular security audits and code reviews
- ✓ Implement proper access controls and permission systems

Audit Team

Team Credentials

Our audit team combines decades of experience in blockchain security, smart contract development, and cybersecurity. Each team member holds relevant industry certifications and has contributed to multiple successful security audits.

Methodology & Standards

Our audit methodology follows industry best practices and standards:

- ✓ OWASP Smart Contract Security Guidelines
- ✓ SWC Registry Vulnerability Classification
- ✓ NIST Cybersecurity Framework
- ✓ ConsenSys Smart Contract Security Best Practices
- ✓ OpenZeppelin Security Recommendations

Audit Process

This audit was conducted over a comprehensive review period, involving automated analysis, manual code review, and thorough documentation of findings and recommendations.

Disclaimer & Legal Notice

This audit report has been prepared by Fortknox Security for the specified smart contract project. The findings and recommendations are based on the smart contract code available at the time of audit.

Scope Limitations

- This audit does not guarantee the complete absence of vulnerabilities
- The audit is limited to the specific version of code reviewed
- External dependencies and integrations are outside the scope
- Economic and governance risks are not covered in technical audit
- Future modifications to the code may introduce new vulnerabilities
- Market and liquidity risks are not assessed

Liability Statement

Fortknox Security provides this audit report for informational purposes only. We do not provide any warranties, express or implied, regarding:

- The absolute security of the smart contract
- The economic viability of the project
- The legal compliance in any jurisdiction
- Future performance or behavior of the contract
- Third-party integrations or dependencies

© 2024 Fortknox Security. All rights reserved.

Legal Terms & Usage Rights

Usage Rights

This audit report may be used by the client for:

- ✓ Public disclosure and transparency
- ✓ Marketing and promotional materials
- ✓ Investor due diligence processes
- ✓ Regulatory compliance documentation
- ✓ Technical documentation and reference
- ✓ Security assessment presentations
- ✓ Community transparency initiatives

Restrictions

The following restrictions apply to this report:

- ✓ Report content may not be modified or altered
- ✓ Fortknox Security branding must remain intact
- ✓ Partial excerpts must maintain context and accuracy
- ✓ Commercial redistribution requires written permission
- ✓ Translation must preserve technical accuracy

Intellectual Property

This report contains proprietary methodologies and analysis techniques developed by Fortknox Security. The format, structure, and analytical approach are protected intellectual property.

Contact Information

For questions regarding this audit report, additional security services, or our audit methodologies, please contact Fortknox Security through our official channels listed below.

Fortknox Security

🌐 https://www.fortknox-security.xyz

🐦 @FortKnox_sec

📧 support@fortknox-security.xyz

© 2024 Fortknox Security. All rights reserved.

FORTKNOX SECURITY

Web3 Security at Fort Knox Level

Contact Us

🐦 @FortKnox_sec

📱 @FortKnox_sec

🌐 fortknox-security.xyz

📧 support@fortknox-security.xyz

Audit performed by
Fortknox Security