GUARDIAN AUDITS

SMART CONTRACT SECURITY AUDIT OF

NFTR

Summary

Audit Firm: Guardian Audits

Client Firm: NFTR

Final Report Date - September 30, 2022

Audit Summary

After a line by line manual analysis and automated review, Guardian has concluded that:

- NFTR's smart contracts have a LOW RISK SEVERITY
- NFTR's smart contracts have an **ACTIVE OWNERSHIP**
- Important owner/tempAdmin privileges – `updateProtocolFeeRecipient`, `shutOffAssignments`, `reduceNamingCredits`, `setRNMAddress`, `shutOffFeeRecipientUpdates`, `addAssignerCredits`, `nullAssignerCredits`, `shutOffAssignerAssignments`, `transferTempAdmin`
- NFTR's smart contract owner has multiple "write" privileges. Centralization risk correlated to the active ownership is **LOW**

Notice that the examined smart contracts are not resistant to internal exploit. For a detailed understanding of risk severity, source code vulnerability, and potential attack vectors, refer to the complete audit report below.

🔗 Blockchain network: **Ethereum**

✅ Verify the authenticity of this report on Guardian's GitHub: https://github.com/guardianaudits

Table of Contents

Project Overview

Project Summary

Project Name	NFTR
Language	Solidity
Codebase	https://github.com/greatrat00/NFTRegistryAudit
Commit(s)	Initial: 92eab9cf7a9aa22ed6eb397ef2ae8ccc7a4161be Remediated: ea0816068510db1046caa7669c9bfeddb8eca04f

Audit Summary

Delivery Date	September 30, 2022
Audit Methodology	Static Analysis, Manual Review

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
High	0	0	0	0	0	0
Medium	1	0	0	0	0	1
Low	17	0	0	3	0	14

Audit Scope & Methodology

Scope

ID	File	SHA-1 Checksum(s)
NMC	NamingCredits.sol	Initial: 92eab9cf7a9aa22ed6eb397ef2ae8ccc7a4161be Remediated: fe422f38142fe351116c5a468e3e98b26695272d

Methodology

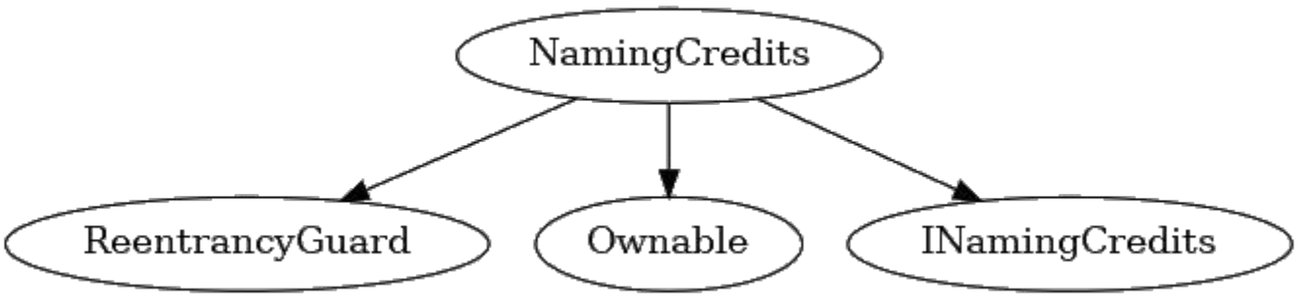
The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross-referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

Vulnerability Classifications

Vulnerability Level	Classification
🔴 Critical	Easily exploitable by anyone, causing loss/manipulation of assets or data.
🟠 High	Arduously exploitable by a subset of addresses, causing loss/manipulation of assets or data.
🟡 Medium	Inherent risk of future exploits that may or may not impact the smart contract execution.
🟢 Low	Minor deviation from best practices.

Inheritance Graph



Findings & Resolutions

ID	Title	Category	Severity	Status
NMC-1	Centralization Risk	Centralization / Privilege	Low	Resolved
NMC-2	Weak Tokenomics Protection	Tokenomics / Privilege	Medium	Resolved
NMC-3	Zero Address Checks	Best Practices	Low	Resolved
NMC-4	Superfluous Code	Optimization	Low	Resolved
NMC-5	Superfluous Code	Optimization	Low	Resolved
NMC-6	Unnecessary Require Statements	Optimization	Low	Acknowledged
NMC-7	Unnecessary Casting	Optimization	Low	Resolved
NMC-8	Typo	Typo	Low	Resolved
NMC-9	Default Value Assignment	Optimization	Low	Resolved
NMC-10	Cache Array Length	Optimization	Low	Resolved
NMC-11	Uint Comparisons	Optimization	Low	Resolved
NMC-12	Storage Modifiers	Optimization	Low	Resolved
NMC-13	Shorten Revert Strings	Optimization	Low	Acknowledged

Findings & Resolutions

ID	Title	Category	Severity	Status
NMC-14	Access Modifiers	Optimization	Low	Resolved
NMC-15	Duplicate Reads	Optimization	Low	Resolved
NMC-16	For-Loop Increment	Optimization	Low	Resolved
NMC-17	Custom Reverts	Optimization	Low	Acknowledged
NMC-18	Visibility Modifiers	Visibility Modifiers	Low	Resolved

NMC-1 | 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`.

Recommendation

Ensure `tempAdmin` is a multi-sig.

Resolution

NFTR Team:

- `tempAdmin` will be a multi-sig.

NMC-2 | 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.

The `MAX_ASSIGNER_CREDITS` can be easily circumvented by simply calling the `addAssignerCredits` multiple times.

The `MAX_CREDITS_ASSIGNED` can be easily circumvented by calling the `assignNamingCredits` function multiple times or calling `assignNamingCreditsBulk` with a `user` list that contains the same address multiple times.

Therefore it is relatively easy for the `tempAdmin` and `assigners` to manipulate the tokenomics of the project, potentially creating unlimited naming credits or allowing a particular address to accumulate more than the intended amount of naming credits.

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.

Resolution

NFTR Team:

- The suggested changes were implemented.

NMC-3 | 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.

Recommendation

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

Resolution

NFTR Team:

- The suggested changes were implemented.

NMC-4 | 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.

Therefore all subsequent computations of `numberOfCredits * nftrAddress.namingPriceEther()` or `numberOfCredits * nftrAddress.namingPriceRNM()` can be replaced with the `currencyQuantity`.

Recommendation

Implement the above simplifications.

Resolution

NFTR Team:

- The suggested changes were implemented.

NMC-5 | 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.

Recommendation

Implement the above simplifications.

Resolution

NFTR Team:

- The suggested changes were implemented.

NMC-6 | 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.

Recommendation

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

Resolution

NFTR Team:

- Acknowledged, but left as is for simplicity.

NMC-7 | 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.

Recommendation

Remove the redundant casts.

Resolution

NFTR Team:

- The suggested changes were implemented.

NMC-8 | Typo

Category	Severity	Location	Status
Typo	● Low	NamingCredits.sol: 120	Resolved

Description

On line 120, “buy” is misspelled as “by”.

Recommendation

Replace “by” with “buy” in the comment.

Resolution

NFTR Team:

- The suggested changes were implemented.

NMC-9 | Default Value Assignment

Category	Severity	Location	Status
Optimization	● Low	NamingCredits.sol: 195	Resolved

Description

On line 195 the `uint` variable i is initialized to the default value of 0.

Recommendation

Remove the unnecessary assignment.

Resolution

NFTR Team:

- The suggested changes were implemented.

NMC-10 | Cache Array Length

Category	Severity	Location	Status
Optimization	● Low	NamingCredits.sol: 195	Resolved

Description

Caching the array length outside a `for` loop saves reading it on each iteration.

Recommendation

Declare a `len` variable and use it as the upper bound in the `for` loop.

Resolution

NFTR Team:

- The suggested changes were implemented.

NMC-11 | Uint Comparisons

Category	Severity	Location	Status
Optimization	Low	NamingCredits.sol: 191	Resolved

Description

When dealing with unsigned integer types, comparisons with `!= 0` are cheaper than with `> 0`.

Recommendation

Replace the `assigners[msg.sender] > 0` check with `assigners[msg.sender] != 0`.

Resolution

NFTR Team:

- The suggested changes were implemented.

NMC-12 | Storage Modifiers

Category	Severity	Location	Status
Optimization	● Low	NamingCredits.sol: 189	Resolved

Description

In `assignNamingCreditsBulk` the `address[] memory user` and `address[] memory numberOfCredits` parameters are never altered and therefore can be declared `calldata`.

Recommendation

Declare the variables `calldata`.

Resolution

NFTR Team:

- The suggested changes were implemented.

NMC-13 | Shorten Revert Strings

Category	Severity	Location	Status
Optimization	● Low	NamingCredits.sol	Acknowledged

Description

Throughout the contract revert strings that are longer than 32 bytes are used.

Recommendation

Shorten revert strings to less than 32 bytes to save on gas.

Resolution

NFTR Team:

- Acknowledged, but left as is for simplicity.

NMC-14 | Access Modifiers

Category	Severity	Location	Status
Optimization	● Low	NamingCredits.sol	Resolved

Description

Throughout the contract there are several require statements that assert the `msg.sender` is the `tempAdmin`. These `require` statements can be deduplicated into a single `onlyTempAdmin` modifier that can be used on each of these functions.

Recommendation

Create an `onlyTempAdmin` modifier and apply it to each of these functions.

Resolution

NFTR Team:

- The suggested changes were implemented.

NMC-15 | Duplicate Reads

Category	Severity	Location	Status
Optimization	● Low	NamingCredits.sol	Resolved

Description

In `assignNamingCreditsBulk` the `numberOfCredits[i]` value is read up to five times upon each iteration. Declare a `uint creditNum` outside of the `for` loop and cache the `numberOfCredits` value in it upon each iteration.

Recommendation

Implement the above suggestion.

Resolution

NFTR Team:

- The suggested changes were implemented.

NMC-16 | For-Loop Increment

Category	Severity	Location	Status
Optimization	● Low	NamingCredits.sol: 196	Resolved

Description

Because the `user` array's length is bound by `MAX_BULK_ASSIGNMENT`, there is no risk of overflow. To reduce bytecode, use an unchecked block in the loop to increment.

Recommendation

Implement the above suggestion.

Resolution

NFTR Team:

- The suggested changes were implemented.

NMC-17 | Custom Reverts

Category	Severity	Location	Status
Optimization	● Low	NamingCredits.sol	Acknowledged

Description

Since Solidity v0.8.4, the more gas-efficient custom-errors have been introduced. They allow for passing dynamic data in the error and remove costly and repeated string error messages.

Recommendation

Consider replacing `require` statements with custom errors.

https://blog.soliditylang.org/2021/04/21/custom-errors/

Resolution

NFTR Team:

- Acknowledged, but left as is for simplicity.

NMC-18 | Visibility Modifiers

Category	Severity	Location	Status
Visibility Modifiers	● Low	NamingCredits.sol: 88, 101	Resolved

Description

The functions `setRNMAddress` and `updateProtocolFeeRecipient` are declared as `public` but are never called from within the contract.

Recommendation

Modify the visibility from `public` to `external`.

Resolution

NFTR Team:

- The suggested changes were implemented.

Auditor’s Verdict

After a line by line manual analysis and automated review, Guardian has concluded that:

- NFTR’s smart contracts have a LOW RISK SEVERITY
- NFTR’s smart contracts have an **ACTIVE OWNERSHIP**
- Important owner/tempAdmin privileges – `updateProtocolFeeRecipient`, `shutOffAssignments`, `reduceNamingCredits`, `setRNMAddress`, `shutOffFeeRecipientUpdates`, `addAssignerCredits`, `nullAssignerCredits`, `shutOffAssignerAssignments`, `transferTempAdmin`
- NFTR’s smart contract owner has multiple “write” privileges. Centralization risk correlated to the active ownership is **LOW**

Disclaimer

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts Guardian to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. Guardian’s position is that each company and individual are responsible for their own due diligence and continuous security. Guardian’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by Guardian is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

Notice that smart contracts deployed on the blockchain are not resistant from internal/external exploit. Notice that active smart contract owner privileges constitute an elevated impact to any smart contract’s safety and security. Therefore, Guardian does not guarantee the explicit security of the audited smart contract, regardless of the verdict.

About Guardian Audits

Founded in 2022 by DeFi experts, Guardian Audits is a leading audit firm in the DeFi smart contract space. With every audit report, Guardian Audits upholds best-in-class security while achieving our mission to relentlessly secure DeFi.

To learn more, visit https://guardianaudits.com

To view our audit portfolio, visit https://github.com/guardianaudits

To book an audit, message https://t.me/guardianaudits