GUARDIAN AUDITS

SMART CONTRACT SECURITY AUDIT OF

NFTR

Summary

Audit Firm: Guardian Audits

Client Firm: NFTR

Final Report Date - April 8, 2023

Audit Summary

NFTR engaged Guardian to review the security of its NFT name marketplace. From the 1st of April to the 8th of April, a team of 2 auditors reviewed the source code in scope. All findings have been recorded in the following report.

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/NFTRegistry/NameMarketplace
Commit(s)	Initial: deba81c173b944d8401bd2c665dd97ae23e9916f

Audit Summary

Delivery Date	April 8, 2023
Audit Methodology	Static Analysis, Manual Review

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	2	0	0	0	0	0
High	2	0	0	0	0	0
Medium	4	0	0	0	0	0
Low	4	0	0	0	0	0

Audit Scope & Methodology

Scope

ID	File	SHA-1 Checksum(s)
NMKT	NameMarketplace.sol	277b2bc2368dc911f0174b228c770fa6bc2b4dda
NFTR	NFTRegistry.sol	d3dfcf1e7bfd919caf3802d2e5b07c86653a8717

Methodology

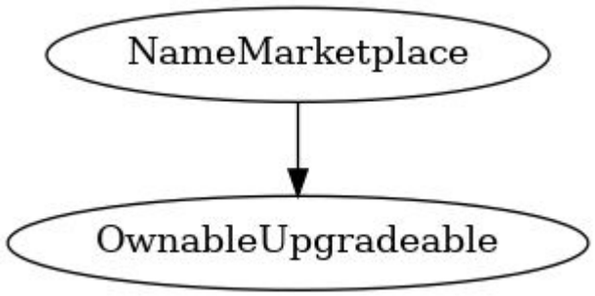
The auditing process pays special attention to the following considerations:

- 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
NFTR-1	Two Names For One Token	Logical Error	Critical	Unresolved
NMKT-1	No Bids Can Be Entered	Logical Error	Critical	Unresolved
NMKT-2	Overwriting Previous Name	Unexpected Behavior	High	Unresolved
NMKT-3	Griefing Name Sellers	Griefing	High	Unresolved
NMKT-4	Inaccurate Event Data	Events	Medium	Unresolved
NMKT-5	Inconsistent Encoded Names	Logical Error	Medium	Unresolved
NMKT-6	Reset Offers on Fee Change	Logical Error	Medium	Unresolved
NMKT-7	Unable To Withdraw Bid	Unexpected Behavior	Medium	Unresolved
NMKT-8	Unnecessary Casting	Best Practices	Low	Unresolved
NMKT-9	Using delete	Best Practices	Low	Unresolved
NMKT-10	Improper Visibility	Best Practices	Low	Unresolved
NMKT-11	Loop Optimization	Optimization	Low	Unresolved

NFTR-1 | Two Names For One Token

Category	Severity	Location	Status
Logical Error	● Critical	NFTRegistry.sol: 273	Unresolved

Description

Names could be transferred to an NFT with an existing name but the `tokenByName` mapping still contains the overwritten name pointing to the NFT. As a result, two different names may point to the same NFT.

Consider the following scenario:

1) Alice owns an NFT with name “Alice”
2) Bob owns an NFT with name “Bob”
3) Name “Alice” is transferred to Bob’s NFT
4) `tokenByName` is updated to reflect that “Alice” points to Bob’s NFT.
5) `tokenByName` still has an entry for name “Bob” also pointing to Bob’s NFT.
6) It now appears that Bob is now the owner of both names “Bob” and “Alice”.
7) A user purchases name “Bob” but ends up receiving name “Alice” upon transfer.

Recommendation

If a name may be transferred to an NFT with an existing name, dereserve the old name using `releaseTokenByName`.

NMKT-1 | No Bids Can Be Entered

Category	Severity	Location	Status
Logical Error	● Critical	NameMarketplace.sol: 376	Unresolved

Description

Initially, when no bids have been entered, the `existing` bid will have default values – `address(0)` as the collection and 0 for the `tokenId`. The zero address does not have function `ownerOf`, so the call to `getOwner` will revert. As a result, no bids can be entered.

Recommendation

Bypass the ownership check when there are no existing bids.

NMKT-2 | Overwriting Previous Name

Category	Severity	Location	Status
Unexpected Behavior	High	NameMarketplace.sol	Unresolved

Description

When a bid is accepted or through `acceptBidForName` or an offered name is purchased through `buyName`, there is no check that the NFT the name shall be transferred to is already named. As a result, a user may unexpectedly lose their old name upon transfer.

Recommendation

Consider if owned names should be overwritten upon transfer. If necessary, clearly document such behavior.

NMKT-3 | Griefing Name Sellers

Category	Severity	Location	Status
Griefing	● High	NameMarketplace.sol	Unresolved

Description

It is simple for a malicious actor to grief a seller by placing a bid higher than the previous and then withdrawing the bid or transferring the `tokenTo` to another address. As a result, any bids made with true buying intention are lost and the seller is unable to sell his name.

Recommendation

Consider allowing for a few blocks to pass before the bid can be overwritten so that a malicious actor risks losing the funds sent to make the ineffectual bid.

NMKT-4 | Inaccurate Event Data

Category	Severity	Location	Status
Events	Medium	NameMarketplace.sol: 262	Unresolved

Description

`string memory name = toLower(nftr.tokenName(collectionFrom, tokenFrom))`

is performed after the transfer is already made. As a result, the event doesn't emit the name that has been transferred but the empty string.

Recommendation

Grab the name from the `toToken` or cache the name prior to transfer.

NMKT-5 | Inconsistent Encoded Names

Category	Severity	Location	Status
Logical Error	Medium	NameMarketplace.sol	Unresolved

Description

There may be potential issues stemming from name parameters being lowered when getting the NFT with the provided name, but the `encodedName` being produced from a non-parsed input.

Consider the following scenario:

1) Bob owns an NFT with name “DIGITAL”
2) Alice calls `enterBidForName` and passes “digital” for the name parameter.
3) Bob calls `acceptBidForName` with name “DIGITAL”, but the encoded name does not match the encoded name generated by Alice’s “digital”.
4) Bob’s transaction reverts and is unable to accept Alice’s bid.

Recommendation

Lower the inputted name prior to encoding.

NMKT-6 | Reset Offers on Fee Change

Category	Severity	Location	Status
Logical Error	Medium	NameMarketplace.sol: 327	Unresolved

Description

If the owner changes the `feePerc`, even if they lower it, all offers with a different fee will need to be reset due to a fee mismatch. A seller would prefer a lower fee so it is unexpected for their offer to be rendered invalid.

Recommendation

Compare the offer's fee percentage against an upper bound rather than an exact check.

.

NMKT-7 | Unable To Withdraw Bid

Category	Severity	Location	Status
Unexpected Behavior	Medium	NameMarketplace.sol: 427	Unresolved

Description

There is potential for a user to be unable to withdraw their bid.

Consider the following scenario:

1) Bob bids for the name “Alice”.
2) He then receives the NFT named “Alice” through secondary markets or a direct transfer.
3) Bob is unable to withdraw his bid value because he now owns the NFT named “Alice”.

Recommendation

Consider whether it is necessary to check for ownership when withdrawing a bid for a name.

If necessary, properly document such behavior for users.

.

NMKT-8 | Unnecessary Casting

Category	Severity	Location	Status
Best Practices	Low	NameMarketplace.sol: 157	Unresolved

Description

`_WETH` is already an address so it is unnecessary to cast it to one.

Recommendation

Remove `address(_WETH)` cast.

NMKT-9 | Using delete

Category	Severity	Location	Status
Best Practices	Low	NameMarketplace.sol	Unresolved

Description

`delete` on a mapping entry can be used to reset to defaults rather than setting a zeroed off Offer/Bid.

Recommendation

Consider using `delete` if only default values are necessary.

NMKT-10 | Improper Visibility

Category	Severity	Location	Status
Best Practices	● Low	NameMarketplace.sol	Unresolved

Description

`toLower` has visibility public but it under the internal functions section.

Recommendation

Consider marking the function with visibility internal or move it to a different section.

NMKT-11 | Loop Optimization

Category	Severity	Location	Status
Optimization	Low	NameMarketplace.sol	Unresolved

Description

The length of `bStr` can be cached. Furthermore, because a name's length is restricted in NFTR, the index can be incremented in a `unchecked` block.

Recommendation

Consider the above gas optimizations.

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