



# xNFP

Security Audit – xNFP Token by NFPrompt

February 13, 2024

Prepared for: NFPrompt ([nfprompt.io](https://nfprompt.io))

Presented By: Audita Security ([audita.io](https://audita.io))

# Document

The contents of this document may include confidential information pertaining to the IT systems, intellectual property, and possible vulnerabilities along with methods of exploitation that the Client may possess. The report that contains this confidential information can be utilized internally by the Client, and can be made available to the public after all vulnerabilities are addressed, depending on the decision of the Client.

---

**Approved by:** Jose Saez Lopez, Reni Dimitrova @ Audita.io

**Contracts:** `xNFP.sol`

**Network:** Binance Smart Chain

**Programming language:** Solidity

**Method:** Manual Audit by Solidity Experts

**Client Website:** <https://nfprompt.io/>

**Timeline:** 06/02/2023 - 13/02/2023

# Table of Contents

<b>Document</b>	<b>1</b>
<b>Executive Summary</b>	<b>3</b>
<b>Audita Vulnerability Classifications</b>	<b>4</b>
<b>Scope</b>	<b>5</b>
<b>Findings</b>	<b>6</b>
General Risks	6
Summary	6
Detailed Findings	8
<b>Recommendations</b>	<b>16</b>
<b>Fixes</b>	<b>18</b>

# Executive Summary

## Fixes

TBD – Information on Fixes introduced will be included here in the **Final Audit Report**.

## Manual Audit

During the manual audit conducted by our experts, we did not identify any **Critical** severity vulnerabilities.

We identified 1 **High**, 2 **Medium** and 1 **Low** severity vulnerabilities.

12 **Informational** issues were indicated, relating to:

- Code Quality
- Gas Optimization

## Overall Assessment

Severity	Count	Acknowledged	Addressed
<b>Critical</b>	0	–	Details on fixes and addressed issues will be included in the Final Audit Report.
<b>High</b>	1	TBD	
<b>Medium</b>	2	TBD	
<b>Low</b>	1	TBD	
<b>Informational</b>	12	TBD	

## Documentation

We recommend selected information from this report, as well as the contracts' README files to be included in protocol's official [Documentation](#), as soon as code is deployed.

# Audita Vulnerability Classifications

Audita follows the most recent standards for vulnerability severities, taking into consideration both the possible impact and the likelihood of an attack occurring due to a certain vulnerability.

Severity	Description
<b>Critical</b>	Critical vulnerability is one where the attack is more straightforward to execute and can lead to exposure of users' data, with catastrophic financial consequences for clients and users of the smart contracts.
<b>High</b>	The vulnerability is of high importance and impact, as it has the potential to reveal the majority of users' sensitive information and can lead to significant financial consequences for clients and users of the smart contracts.
<b>Medium</b>	The issue at hand poses a potential risk to the sensitive information of a select group of individual users. If exploited, it has the potential to cause harm to the client's reputation and could result in unpleasant financial consequences.
<b>Low</b>	The vulnerability is relatively minor and not likely to be exploited repeatedly, or is a risk that the client has indicated is not impactful or significant, given their unique business situation.
<b>Informational</b>	The issue may not pose an immediate threat to ongoing operation or utilization, but it's essential to consider implementing security and software engineering best practices, or employing backup measures as a safety net.

# Scope

The security assessment was scoped to the following smart contract in [token-generation](#) repository:

Contract name
xNFP.sol

The codebase has been audited up to and including commit

*9acfa4a1c0783ae7260db6f1cde3f9342d573bbd*

# Findings

## General Risks

- The owner role is allowed to change the `_redemptionActive` status which may lock each step of the redemption process.
- In case the owner removes the xNFP contract address from the whitelist, the contract functionality that requires transferring xNFP token to or from it may be locked.

## Summary

Code	Description	Severity	Fixes
[xNFP-01]	Insufficient NFP Tokens for Redemption Payouts	High	TDB
[xNFP-02]	( <code>_usagesWhitelist</code> ) User Tokens Locked Due to <code>usageAddress</code> Removal	Medium	TDB
[xNFP-03]	Discrepancy Between Documentation and Code on Token Burning in The Redemption Process	Medium	TDB
[xNFP-04]	Potential Loss of Expected Tokens for Users Due to <code>usagesDeallocationFee</code> Changes	Low	TDB
[xNFP-05]	Missing Event Emitting When Updating Redeem Settings	Informational	TDB
[xNFP-06]	Different variables naming styles	Informational	TDB
[xNFP-07]	( <code>redeemDividendsAdjustment</code> ) Two different scaling systems are used	Informational	TDB

[xNFP-08]	(convertWithPermit) Missing Inline Documentation	Informational	TDB
[xNFP-09]	(_convert) TransferFrom should be used above (_mint) function	Informational	TDB
[xNFP-10]	(getRedeemAmountByDuration) Make all 3333, 6666, 10000 numbers to be constants	Informational	TDB
[xNFP-11]	(startRedemption) No need of <= 0 check	Informational	TDB
[xNFP-12]	Move INFP interface in another folder	Informational	TDB
[xNFP-13]	(updateDividendsAddress, cancelRedemption, completeRedemption) Move function invocations to be before emitting events	Informational	TDB
[xNFP-14]	(_currentBlockTimestamp) Simply return the block.timestamp	Informational	TDB
[xNFP-15]	(startRedemption) Consider allocating the entire xAmount	Informational	TDB
[xNFP-16]	Treasury Spelling	Informational	TDB



## Detailed Findings

[xNFP-01]	Insufficient NFP Tokens for Redemption Payouts	High
-----------	--	------

**Functions:** startRedemption, completeRedemption, \_deallocate

**Details:**

When initiating the redemption process through the startRedemption function, the current implementation fails to check whether there are sufficient NFP tokens available to complete the process. This oversight can lead to scenarios where users are unable to receive the allocated tokens upon the completion of the redemption process due to an insufficient amount of NFP tokens.

Additionally, an issue arises with the premature burning of NFP tokens within the \_deallocate function. It burns NFP tokens that may be intended to be distributed to users at the end of the redemption process. As a result, users may not receive the tokens allocated to them after the duration of the redemption process has elapsed.

**Recommendation:**

Implement a preliminary check within startRedemption to ensure sufficient NFP tokens are available before initiating the redemption process (and burn NFP during completing the redemption). In the allocate function, implement a check to verify if there are sufficient tokens to cover the maximum possible deallocation fee, using MAX\_DEALLOCATION\_FEE as the usagesDeallocationFee can vary, plus the tokens needed for all ongoing redemptions.

[xNFP-02]	(_usagesWhitelist) User Tokens Locked Due to usageAddress Removal	Medium
-----------	---	--------

**Functions:** deallocate, updateToUsagesWhitelist

**Details:**

The deallocate function employs a validateUsage modifier, which results in failure if the usageAddress is not present in \_usagesWhitelist. In scenarios where a user allocates tokens

with a specific `usageAddress`, and later this `usageAddress` is removed from `_usagesWhitelist`, the user becomes unable to deallocate their tokens. This issue effectively locks the user's tokens, preventing them from reclaiming them.

**Recommendation:**

Ensure that users can still withdraw their tokens when their `usageAddress` is removed from `_usagesWhitelist`. This might involve modifying the `validateUsage` modifier or implementing an additional check within the `deallocate` function to allow token deallocation even if the `usageAddress` is no longer whitelisted. This change will safeguard user assets against being inadvertently locked due to administrative changes to the whitelist.

<b>[xNFP-03]</b>	Discrepancy Between Documentation and Code on Token Burning in The Redemption Process	<b>Medium</b>
------------------	---	---------------

**Functions:** `completeRedemption`

**Details:**

The documentation for The Redemption Process specifies that "if the selected redemption duration is shorter than the maximum (implying a ratio < 1:1), a penalty amount of xNFP tokens is supposed to be burned." Additionally, it mentions that "The NFPrompt Foundation may reserve up to 50% of the would-be burned xNFP tokens to support the Network." However, the actual code implementation results in the burning of NFP tokens instead of xNFP tokens. This discrepancy indicates that either the code has been implemented incorrectly, or the documentation inaccurately reflects the intended process.

**Recommendation:**

Ensure consistency between the documentation and the actual code implementation regarding the token type to be burned during the redemption process.

[xNFP-04]	Potential Loss of Expected Tokens for Users Due to usagesDeallocationFee Changes	Low
-----------	--	-----

**Functions:** deallocate, updateDeallocationFee

**Details:**

When a user deallocates tokens using the deallocate function, a deallocationFeeAmount is burned, which is calculated based on the usagesDeallocationFee. This fee can be modified through the updateDeallocationFee function. Adjusting the usagesDeallocationFee, either through standard updates or front-running tactics, can result in users receiving fewer tokens than expected. Such changes to the deallocation fee can lead to unpredictability in the token return for users, undermining trust in the token deallocation process.

**Recommendation:**

Ensure users receive the amount of tokens they expect upon deallocation by introducing a safeguard, such as a minOut parameter, in the deallocation process.

[xNFP-05]	Missing Event Emitting When Updating Redeem Settings	Informational
-----------	--	---------------

**Functions:** updateRedeemSettings

**Details:** The event is not emitted in the updateRedeemSettings. Therefore, it is not possible to track it off-chain.

**Recommendation:**

Emit the corresponding event.

<b>[xNFP-06]</b>	Different variables naming styles	Informational
------------------	-----------------------------------	---------------

**Details:**

The variables `_redemptionActive` and `usagesWhitelist` use different naming styles; they include an underscore prefix, while other variables do not. This inconsistency decreases code readability.

**Recommendation:**

Ensure that the naming style of variables is consistent across the codebase.

<b>[xNFP-07]</b>	(redeemDividendsAdjustment) Two different scaling systems are used	Informational
------------------	--	---------------

**Function:** `redeemDividendsAdjustment`

**Details:**

Two different scaling systems are used for the percentages.

**Recommendation:**

Work with only one scaling system and move the 50% value to become a constant or provide it as a constructor argument.

<b>[xNFP-08]</b>	(convertWithPermit) Missing Inline Documentation	Informational
------------------	--	---------------

**Function:** `convertWithPermit`

**Recommendation:**

Include inline documentation to make the code more understandable.

[xNFP-09]	( <code>_convert</code> ) <code>TransferFrom</code> should be used above ( <code>_mint</code> ) function	Informational
-----------	--	---------------

**Function:** `_convert`

**Recommendation:**

Use `TransferFrom` above the `_mint` function. Keep Check Effects Interactions pattern.

[xNFP-10]	( <code>getRedeemAmountByDuration</code> ) Make all 3333, 6666, 10000 numbers to be constants	Informational
-----------	---	---------------

**Recommendation:**

Make all 3333, 6666, 10000 numbers to be constants.

[xNFP-11]	( <code>startRedemption</code> ) No need of <code>&lt;= 0</code> check	Informational
-----------	--	---------------

**Recommendation:**

Uint can never become a negative number. No need for `<= 0` check.

[xNFP-12]	Move INFP interface in another folder	Informational
-----------	---------------------------------------	---------------

**Recommendation:**

Move the INFP interface in another folder for better organization of files in the repo.

[xNFP-13]	(updateDividendsAddress, cancelRedemption, completeRedemption) Move function invocations to be before emitting events	Informational
-----------	---	---------------

**Function:** updateDividendsAddress, cancelRedemption, completeRedemption

**Recommendation:**

Call the function before emitting the events, as the current implementation violates best Solidity coding practices.

This way the proper order of events can be tracked offline.

[xNFP-14]	(_currentBlockTimestamp) Simply return the block.timestamp	Informational
-----------	--	---------------

**Function:** \_currentBlockTimestamp

**Recommendation:**

No need to use \_currentBlockTimestamp, simply return the block.timestamp.

[xNFP-15]	(startRedemption) Consider allocating the entire xAmount	Informational
-----------	--	---------------

**Function:** startRedemption

**Details:**

It's difficult for us to understand why there is  $xAmount * redeemDividendsAdjustment$ ?

**Recommendation:**

Consider allocating the entire xAmount to reduce complexity, unless there is a specific need to make this calculation.

[xNFP-16]	Treasury Spelling	Informational
-----------	-------------------	---------------

**Details:**

The spelling of "treasury" is incorrect within the code. Fix spelling.

## Overall Assessment

Severity	Count	Acknowledged	Addressed
Critical	0	–	Details on fixes and addressed issues will be included in the Final Audit Report.
High	1	TBD	
Medium	2	TBD	
Low	1	TBD	
Informational	12	TBD	



# Recommendations

Audita has put forward the following recommendations for xNFP smart contract:

- Please pay attention to the [General Risks](#) outlined in the beginning of this document.
- Implement a preliminary check within startRedemption to ensure sufficient NFP tokens are available.
- In the allocate function, implement a check to verify if there are sufficient tokens to cover the maximum possible deallocation fee.
- Ensure that users can still withdraw their tokens when their usageAddress is removed from \_usagesWhitelist.
- Ensure consistency between the documentation and the actual code implementation regarding the token type to be burned during the redemption process.
- Ensure users receive the amount of tokens they expect upon deallocation by introducing a safeguard, such as a minOut parameter, in the deallocation process.
- Emit event in the updateRedeemSettings.
- Ensure that the naming style of variables is consistent across the codebase.
- Work with only one scaling system and move the 50% value to become a constant or provide it as a constructor argument.
- Include inline documentation in convertWithPermit to make the code more understandable.
- Use TransferFrom above the \_mint function. Keep Check Effects Interactions pattern.
- Make all 3333, 6666, 10000 numbers to be constants.
- Uint can never become a negative number. No need for  $\leq 0$  check.
- Move INFP interface in another folder for better organization of files in the repo.

- Call `deleteRedeemEntry` and `updateDividendsAddress` function before emitting the events. This way the proper order of events can be tracked offline.
- No need to use `_currentBlockTimestamp`, simply return the `block.timestamp`.
- Consider allocating the entire `xAmount` to reduce complexity, unless there is a specific need to make this calculation.
- Fix the spelling of the word “treasury” within the code.

## Fixes

NFPrompt's team are dedicated and responsive, cooperating to acknowledge and implement the above recommendations.

Information on specific fixes will be included here in the **Final Audit Report**.

## Disclaimer

This audit makes no statements or warranties on the security of the code. This report should not be considered a sufficient assessment on the safety of the code, quality status, or any other contract statements. **While we have conducted the analysis to our best abilities and produced this report in line with latest industry developments, it is important to not rely on this report only.** In order for contracts to be considered as safe as possible, the industry standard requires them to be checked by several independent auditing bodies. Those can be other audit firms or public bounty programs.

The contacts live on a blockchain (a smart contract platform) – Smart contract platforms, their programming languages, and other software components are not immune to vulnerabilities that can be exploited by hackers. As a result, although a smart contract audit can help identify potential security issues, it cannot provide an absolute guarantee of the audited smart contract's security.