GUARDIAN AUDITS

SMART CONTRACT SECURITY AUDIT OF

nfty.finance

Summary

Audit Firm Guardian

Prepared By Daniel Gelfand, Owen Thurm, Wafflemakr, Priyam Soni

Client Firm NFTY Finance

Final Report Date March 14, 2024

Audit Summary

NFTY Finance engaged Guardian to review the security of its NFT lending protocol. From the 19th of February to the 26th of February, a team of 4 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

📊 Code coverage & PoC test suite: https://github.com/GuardianAudits/NFTY-PoCs

Table of Contents

Project Overview

Project Summary

Project Name	NFTY Finance
Language	Solidity
Codebase	https://github.com/NFTYLabs/nftyfinance-monorepo
Commit(s)	Initial: e542089a05058befd917c509b071746b656fb9da Final: b1ab436778748e24d12585d379c2c04a8671cfe6

Audit Summary

Delivery Date	March 14, 2024
Audit Methodology	Static Analysis, Manual Review, Test Suite, Contract Fuzzing

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	2	0	0	0	0	2
High	2	0	0	0	0	2
Medium	6	0	0	0	0	6
Low	9	0	0	5	0	4

Audit Scope & Methodology

Vulnerability Classifications

Severity	Impact: *High*	Impact: *Medium*	Impact: *Low*
Likelihood: *High*	🔴 Critical	🟠 High	🟡 Medium
Likelihood: *Medium*	🟠 High	🟡 Medium	🟢 Low
Likelihood: *Low*	🟡 Medium	🟢 Low	🟢 Low

Impact

High Significant loss of assets in the protocol, significant harm to a group of users, or a core functionality of the protocol is disrupted.

Medium A small amount of funds can be lost or ancillary functionality of the protocol is affected. The user or protocol may experience reduced or delayed receipt of intended funds.

Low Can lead to any unexpected behavior with some of the protocol's functionalities that is notable but does not meet the criteria for a higher severity.

Likelihood

High The attack is possible with reasonable assumptions that mimic on-chain conditions, and the cost of the attack is relatively low compared to the amount gained or the disruption to the protocol.

Medium An attack vector that is only possible in uncommon cases or requires a large amount of capital to exercise relative to the amount gained or the disruption to the protocol.

Low Unlikely to ever occur in production.

Audit Scope & Methodology

Methodology

Guardian is the ultimate standard for Smart Contract security. An engagement with Guardian entails the following:

- Two competing teams of Guardian security researchers performing an independent review.
- A dedicated fuzzing engineer to construct a comprehensive stateful fuzzing suite for the project.
- An engagement lead security researcher coordinating the 2 teams, performing their own analysis, relaying findings to the client, and orchestrating the testing/verification efforts.

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. Comprehensive written tests as a part of a code coverage testing suite.
- Contract fuzzing for increased attack resilience.

Invariants Assessed

During Guardian's review of NFTY's lending contract, fuzz-testing with Echidna was performed on the protocol's main functions. Given the dynamic interactions and the potential for unforeseen edge cases in the protocol, fuzz-testing was imperative to verify the integrity of several system invariants.

Throughout the engagement the following invariants were assessed for a total of 1,000,000+ runs with a prepared Echidna fuzzing suite.

ID	Description	Initial	Remediated	Run Count
DESK-01	Desk NFT sent to msg.sender	✅	✅	1,000,000+
DESK-02	Desk balance increased with token deposits	✅	✅	1,000,000+
DESK-03	Desk balance decreased with token withdrawals	✅	✅	1,000,000+
DESK-04	Owner can't withdraw more than desk balance	✅	✅	1,000,000+
DESK-05	Only desk owner is able to update configs	✅	✅	1,000,000+
DESK-06	Owner should only freeze desk when status is Active	✅	✅	1,000,000+
DESK-07	Owner should only unfreeze desk when status is Frozen	✅	✅	1,000,000+
LOAN-01	Interest is correctly calculated during loan initialization	❌	✅	1,000,000+
LOAN-02	Loan status is Active after initialized	✅	✅	1,000,000+

Invariants Assessed

ID	Description	Initial	Remediated	Run Count
LOAN-03	Borrower token balance increased by loan amount	✅	✅	1,000,000+
LOAN-04	Platform wallet balance increased by initialization fees	✅	✅	1,000,000+
LOAN-05	Loans can't be initialized when desk status is Frozen	✅	✅	1,000,000+
LOAN-06	Loan paid back completely when resolve flag is set	✅	✅	1,000,000+
LOAN-07	Borrower should get NFT back when loan is fully repaid	✅	✅	1,000,000+
LOAN-08	Loan status is set to Resolved when fully repaid	✅	✅	1,000,000+
LOAN-09	Loan should never be repaid after duration	❌	✅	1,000,000+
LOAN-10	Only desk owner can liquidated loans	✅	✅	1,000,000+
LOAN-11	Loans should only be liquidated after loan end time	✅	✅	1,000,000+
LOAN-12	Loans should not be liquidated if not in Active status	✅	✅	1,000,000+
GLOBAL-01	Token contract balance should always be greater of equal than all desk balances	✅	✅	1,000,000+
GLOBAL-02	Total borrowed amount of all loans should be less or equal to the total deposited amount in desk	✅	N/A	1,000,000+

Findings & Resolutions

ID	Title	Category	Severity	Status
C-01	Defaults Forced By Removing lendingDeskLoanConfigs	DoS	Critical	Resolved
C-02	Frontrunning Loan Creations	Frontrunning	Critical	Resolved
H-01	Blacklisted Lenders Force Defaults	DoS	High	Resolved
H-02	Interest Calculation Set To Min Interest	Rounding	High	Resolved
M-01	Overlap Between Payment And Default Periods	Logical Error	Medium	Resolved
M-02	Errant Origination Fee Validation	Logical Error	Medium	Resolved
M-03	Paused State Leads To Forced Defaults	Logical Error	Medium	Resolved
M-04	Zero Platform Fee Can DoS New Loans	DoS	Medium	Resolved
M-05	Borrowers Exposed To Gas Griefing	Gas Griefing	Medium	Resolved
M-06	Block Stuffing Risk	Block Stuffing	Medium	Resolved
L-01	Misleading Comment	Documentation	Low	Resolved
L-02	Inaccurate NatSpec	Documentation	Low	Resolved
L-03	Empty Loan Config Check Missing	Validation	Low	Acknowledged

Findings & Resolutions

ID	Title	Category	Severity	Status
L-04	Updating NFTY Finance Address Can DoS	Centralization Risk	Low	Acknowledged
L-05	Lacking SafeCast Usage	Best Practices	Low	Resolved
L-06	Interest Charged On Repaid Principle	Unexpected Behavior	Low	Acknowledged
L-07	External Call Safety	External Calls	Low	Resolved
L-08	PUSH0 Warning	Warning	Low	Acknowledged
L-09	System Incompatible With Fee-on-transfer Tokens	Documentation	Low	Acknowledged

C-01 | Defaults Forced By Removing lendingDeskLoanConfigs

Category	Severity	Location	Status
DoS	● Critical	NFTYFinanceV1.sol	Resolved

Description

With the `removeLendingDeskLoanConfig` function, a lending desk owner is able to remove the `lendingDeskLoanConfig` for loans that are still active. As a result the lending owner is able to prevent ERC1155 loans from being closed as `nftCollectionIsErc1155` would be false for that lending desk and collection.

Consequently, the `makeLoanPayment` function errantly attempts to treat ERC1155 tokens as ERC721 tokens and ultimately reverts.

The lending desk owner can then subsequently add the correct `lendingDeskLoanConfig` back with the `setLendingDeskLoanConfigs` function only after the loan has expired and the owner can now claim the borrower's collateral.

Recommendation

Do not read from the `lendingDeskLoanConfigs` mapping in the `makeLoanPayment` function, instead add an additional `nftCollectionIsErc1155` boolean on the `Loan` struct and rely on that cached value to determine how to handle the transferring of collateral.

Similarly, do not rely on the `lendingDeskLoanConfigs` mapping in the `liquidateDefaultedLoan` function, as the config may no longer be present. Instead rely on the new `nftCollectionIsErc1155` boolean that will be stored on the `Loan` struct.

Resolution

NFTY Team: The issue was resolved in PR#230.

C-02 | Frontrunning Loan Creations

Category	Severity	Location	Status
Frontrunning	● Critical	NFTYFinanceV1.sol: 579	Resolved

Description

Each lending desk has a `LoanConfig` per `nftCollection` address, which contains the details about the minimum and maximum interest charged to the borrower.

A malicious desk owner can front-run the `initializeNewLoan` call from the borrower, and change the loan configuration with a large interest rate and a small duration, making the user pay more interest than they expected to pay when the new loan transaction was originated.

Currently, there is no validation for `maxInterest` besides `maxInterest>=minInterest`. If the desk owner sets the interest the max allowed interest `type(uint32).max = 4294967295` and configures the other Loan params to have constant interest, the borrower Loan interest will be set to `minInterest` chosen by desk owner. This means that the borrower will pay around 4900% per interest per hour and there is a minimum of 1 hour wait to repay the loan.

In summary, a malicious lender can set a small interest rate to honeypot borrowers, update the loan configuration before their transaction is initialized, and lock the borrower in at the max interest rate for at least an hour which they must pay.

Recommendation

Add an extra parameter to the `initializedNewLoan` function, where the borrower can set a `maxInterestAllowed`, which will act as a limit on what they are willing to pay.

Resolution

NFTY Team: The issue was resolved in PR#238.

H-01 | Blacklisted Lenders Force Defaults

Category	Severity	Location	Status
DoS	High	NFTYFinanceV1.sol: 822	Resolved

Description

In the `makeLoanPayment` function, the `lendingDesk.erc20` token is transferred directly to the lender address: `IERC20(lendingDesk.erc20).safeTransferFrom(msg.sender, lender, _amount);`

Therefore any lender that is blacklisted for the payment token will prevent the user from making loan payments. The lender will then force the user to be liquidated as they cannot pay back their loan in time.

Furthermore, in the case of tokens with hooks after transfers, like ERC777, the receiver can make the transfer revert, preventing the borrower to make any payments to the loan as well.

Recommendation

Do not push the `lendingDesk.erc20` tokens directly to the lender address, instead increment a uint256 value in a mapping for an individual erc20 token and allow lenders to claim this amount with a separate function (pull-over-push pattern).

Resolution

NFTY Team: The issue was resolved in PR#258.

H-02 | Interest Calculation Set To Min Interest

Category	Severity	Location	Status
Rounding	High	NFTYFinanceV1.sol: 645-649	Resolved

Description

When initializing a new loan, the user will pass both `_duration` and `_amount` parameters. If both amount and duration are variable, the interest should be calculated based on scaling both duration and amount.

The issue arises due to the inherent rounding down behavior in Solidity. When calculating the average amount and duration, they are rounded down to 0. After multiplying with `(loanConfig.maxInterest - loanConfig.minInterest)`, the result is 0. Consequently, the `interest` is always set as `loanConfig.minInterest`, resulting in the lender missing out on potential additional interest due to rounding down.

• Example: If `minAmount` = 100, `maxAmount` = 200, and `_amount`=150
• `(_amount - loanConfig.minAmount) / (loanConfig.maxAmount - loanConfig.minAmount)`
• `(150 - 100) / (200 - 100)` = `50 / 100 = 0`
• The same happens with duration.

Recommendation

Calculate in the following manner to prevent rounding down:

```
interest =
  loanConfig.minInterest +
  uint32(
      (// Take average of amount and duration factors
      (((_amount - loanConfig.minAmount) * (loanConfig.maxInterest - loanConfig.minInterest)) /
(loanConfig.maxAmount - loanConfig.minAmount)) +
       (((_duration - loanConfig.minDuration) * (loanConfig.maxInterest - loanConfig.minInterest)) /
(loanConfig.maxDuration - loanConfig.minDuration))
       ) / 2
  );
```

Resolution

NFTY Team: The issue was resolved in PR#239.

M-01 | Overlap Between Payment And Default Periods

Category	Severity	Location	Status
Logical Error	Medium	NFTYFinanceV1.sol: 741	Resolved

Description

Since the `hoursElapsed` is rounded down in the `getLoanAmountDue` function and the `loan.duration` check uses strictly greater than, loan payments are only disabled an entire hour after the end date of a loan.

Therefore during this time a borrower may still pay back their loans and may frontrun the liquidation tx to do so.

Recommendation

Alter the `hoursElapsed > loan.duration` check to be `hoursElapsed >= loan.duration`

Resolution

NFTY Team: The issue was resolved in PR#232.

M-02 | Errant Origination Fee Validation

Category	Severity	Location	Status
Logical Error	Medium	NFTYFinanceV1.sol: 888	Resolved

Description

In the `setLoanOriginationFee` function the `_loanOriginationFee` basis points value is intended to be capped at a maximum of 10%, however the validation asserts that the `_loanOriginationFee` is less than 10_000, which represents 100% in basis points.

Recommendation

Validate that the `_loanOriginationFee` value is less than 1_000, rather than less than 10_000.

Resolution

NFTY Team: The issue was resolved in PR#240.

.

M-03 | Paused State Leads To Forced Defaults

Category	Severity	Location	Status
Logical Error	Medium	NFTYFinanceV1.sol	Resolved

Description

When the owner pauses the NFTYFinanceV1 contract, borrowers cannot repay their loans and therefore may be forced to default and lose their NFT collateral.

Recommendation

Consider allowing the `makeLoanPayment` function to be called when the protocol is paused.

Resolution

NFTY Team: The issue was resolved in PR#257.

M-04 | Zero Platform Fee Can DoS New Loans

Category	Severity	Location	Status
DoS	Medium	NFTYFinanceV1.sol: 721	Resolved

Description

In the `setLoanOriginationFee` function there is not validation that the `_loanOriginationFee` is not 0, therefore the `platformFee` that is taken from new loans can be 0 when the `loanOriginationFee` is set to 0.

Some ERC20 tokens choose to revert upon transferring a 0 amount, however there is no check that the `platformFee` is nonzero before attempting to transfer this amount to the platformWallet.

Recommendation

In the `initializeNewLoan` function, only attempt to transfer the `platformFee` to the `platformWallet` if the `platformFee` is nonzero.

Resolution

NFTY Team: The issue was resolved in PR#241.

M-05 | Borrowers Exposed To Gas Griefing

Category	Severity	Location	Status
Gas Griefing	Medium	NFTYFinanceV1.sol: 688	Resolved

Description

In the `initializeNewLoan` function a `promissoryNote` is minted to the lender using the `INFTYERC721V1.mint` function, which relies on safeMint. As a result if the lender address is home to a contract, the `onERC721Received` function will be invoked at that address.

The contract at the lender address may have malicious logic implemented for the `onERC721Received` function to waste the borrower's gas, causing a loss of native tokens for the user.

Recommendation

Consider using `_mint` rather than `_safeMint` for the mint implementation in the `NFTYERC721V1` contract so that borrowers cannot be exposed to gas griefing.

Resolution

NFTY Team: The issue was resolved in PR#253.

M-06 | Block Stuffing Risk

Category	Severity	Location	Status
Block Stuffing	Medium	NFTYFinanceV1.sol	Resolved

Description

If the `block.timestamp` is a few seconds before the beginning of a new hour and User A sends a tx to pay back their full loan amount, the lender may stuff blocks on the network until the next hour begins. As a result, the borrowers debt to be repaid will increase and the borrowers tx will no longer close the loan.

Immediately the borrower will have to pay an additional period of interest. More insidiously, the borrower may not realize that the loan remains open and as a result may be unexpectedly liquidated.

Recommendation

Consider allowing users to pass a boolean indicating whether they would like to repay the full amount, rather than always specifying a particular amount to pay back. This way the transaction will close the loan regardless of when the transaction is recorded.

Resolution

NFTY Team: The issue was resolved in PR#242.

L-01 | Misleading Comment

Category	Severity	Location	Status
Documentation	Low	NFTYFinanceV1.sol: 843	Resolved

Description

In the `liquidateDefaultedLoan` function the loan status is assigned to `Defaulted` upon liquidation, however the comment on line 843 suggests that the loan state is assigned to resolved.

Recommendation

Update the comment to indicate that the `loan.status` will be assigned to `LoanStatus.Defaulted` rather than `LoanStatus.Resolved`.

Resolution

NFTY Team: The issue was resolved in PR#229.

L-02 | Inaccurate NatSpec

Category	Severity	Location	Status
Documentation	Low	NFTYERC721V1.sol: 114	Resolved

Description

The function `burn` has the same NatSpec from the `mint` function.

Recommendation

Update the documentation to reflect the `burn` function accurately.

Resolution

NFTY Team: The issue was resolved in PR#243.

L-03 | Empty Loan Config Check Missing

Category	Severity	Location	Status
Validation	Low	NFTYFinanceV1.sol: 299	Acknowledged

Description

The function `setLendingDeskLoanConfigs` is missing a check for empty loan config. Therefore it is possible to pass an empty `_loanConfigs` array but the `LendingDeskLoanConfigsSet` event will still be emitted.

Recommendation

Add a check to ensure `_loanConfigs.length > 0`

Resolution

NFTY Team: Acknowledged.

L-04 | Updating NFTY Finance Address Can DoS

Category	Severity	Location	Status
Centralization Risk	Low	NFTYERC721V1.sol: 89	Acknowledged

Description

The NFTYERC721V1 contract, which is the base contract for NFTYLendingKeysV1, NFTYObligationNotesV1 and NFTYPromissoryNotesV1, has a function `setNftyFinance` which allows the owner to update the `nftyFinance` address.

The main goal is to have the correct `nftyFinance` address set is to prevent unauthorized access to `mint` and `burn` function, meaning, only that contract is authorized to call them. In the scenario where the owner updates this address, existing loan desks and loanIds in the NFTFinanceV1 contract will be stuck, as the every call to burn will now fail, due to the `onlyNftyFinance` modifier.

Recommendation

Avoid updating `nftyFinance` when there are active loans.

Resolution

NFTY Team: Acknowledged.

L-05 | Lacking SafeCast Usage

Category	Severity	Location	Status
Best Practices	Low	NFTYFInanceV1.sol: 624, 642	Resolved

Description

In the `initializeNewLoan` function the interest calculations include casting a `uint256` to a `uint32`. These calculations should always be safe and avoid overflow, however as a best practice it would be prudent to use OpenZeppelin's `SafeCast` library to perform these casts.

Recommendation

Consider implementing `SafeCast` for these `uint32` casts.

Resolution

NFTY Team: Resolved.

L-06 | Interest Charged On Repaid Principle

Category	Severity	Location	Status
Unexpected Behavior	Low	NFTYFinanceV1.sol	Acknowledged

Description

Interest on loans is charged on the original loan amount even if some of the loan principle has been repaid.

Recommendation

Be sure to clearly document this behavior to users.

Resolution

NFTY Team: Acknowledged.

L-07 | External Call Safety

Category	Severity	Location	Status
External Calls	Low	NFTYFinanceV1.sol	Resolved

Description

Throughout the NFTYFinanceV1 contract external calls are made without regard to state updates, the following rules ought to be followed:

• `safeTransferFrom` should occur first in functions to avoid making an external call via callback tokens after accounting has been updated but before funds have been received.

• Other external calls should occur after all state updates have occurred.

Recommendation

Implemented the above suggestions, as seen in this PR:
https://github.com/GuardianAudits/NFTY-PoCs/pull/12/files

Resolution

NFTY Team: The issue was resolved in PR#259.

L-08 | PUSH0 Warning

Category	Severity	Location	Status
Warning	Low	Global	Acknowledged

Description

The NFTY Finance contracts are configured to user solidity 0.8.22 and higher, these versions of the EVM compiler make use of the PUSH0 opcode which is not supported by all EVM compatible chains.

Recommendation

The immediate deployment target of Ethereum Mainnet is safe as this network supports the PUSH0 opcode, however the team should be wary of PUSH0 support as they deploy to new EVM compatible networks.

Before deploying to a new target chain, be sure to check whether the chain supports the PUSH0 opcode, and if it does not consider reducing the compiler version to < 0.8.20.

Resolution

NFTY Team: Acknowledged.

L-09 | System Incompatible With Fee-on-transfer Tokens

Category	Severity	Location	Status
Documentation	Low	Global	Acknowledged

Description

Throughout the NFTYFinanceV1 contract the token transfer accounting assumes that the transferred amount is received, however this may not be the case for fee-on-transfer or rebase tokens.

Recommendation

Be sure to clearly document that the system is not compatible with fee-on-transfer tokens. Otherwise if they are intended to be supported then the amount of tokens actually received should be measured by a before and after balance check.

Resolution

NFTY Team: Acknowledged.

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