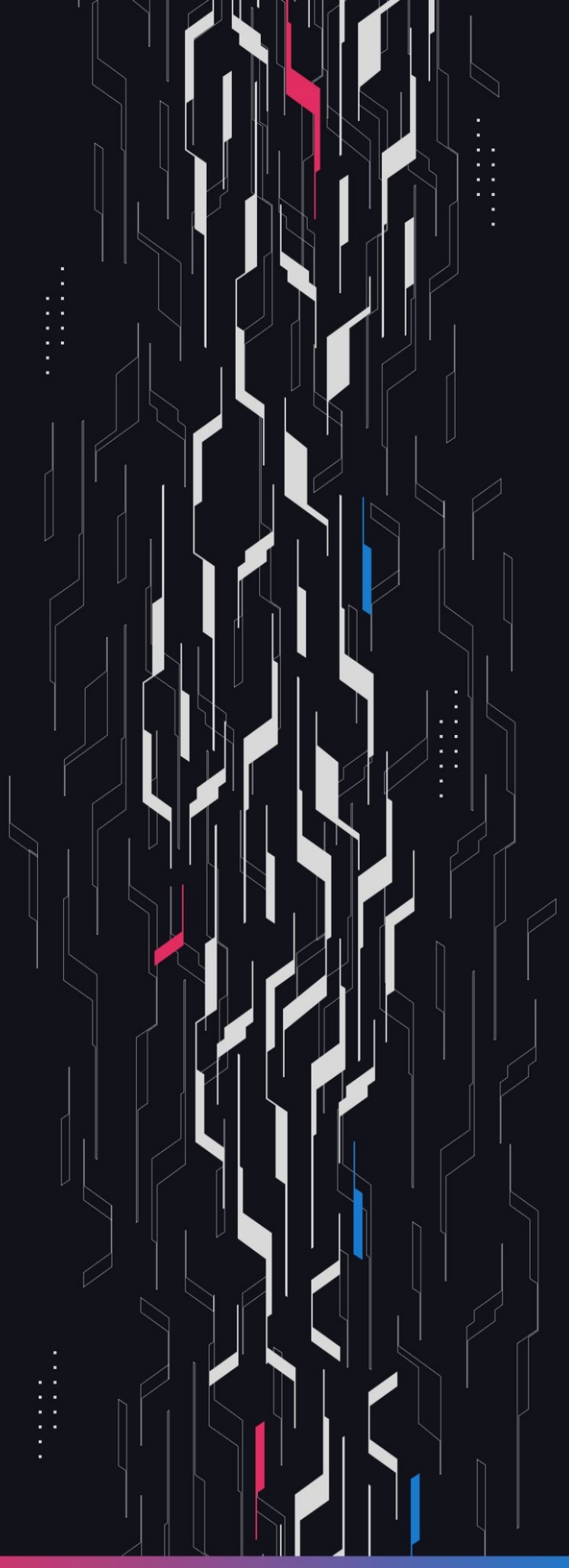
GA GUARDIAN

DN404

Security Assessment

May 13th, 2024



Summary

Audit Firm Guardian

Prepared By Daniel Gelfand, Owen Thurm, Alin Mihai Barbatei (ABA), gkrastenov, ggggtttt

Client Firm DN404

Final Report Date May 13, 2024

Audit Summary

DN404 engaged Guardian to review the security of its implementation of the ERC7631 co-joined ERC20 and ERC721 standard. From the 23rd of April to the 3rd of May a team of 5 auditors reviewed the source code in scope. All findings and resolutions have been recorded in the following report.

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

📊 Code coverage & PoC test suite: https://github.com/GuardianAudits/DN404PoCs

Table of Contents

Project Overview

Project Summary

Project Name	DN404
Language	Solidity
Codebase	https://github.com/Vectorized/dn404
Commit(s)	Initial Commit: 851d1979d687e3e0e28a3041ae5db52a7c92a451 Final Commit: 912222d62c18526504efedc4178e1fadc2c51cef

Audit Summary

Delivery Date	May 3, 2024
Audit Methodology	Static Analysis, Manual Review, Test Suite, Contract Fuzzing

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
High	2	0	0	0	0	2
Medium	4	0	0	1	0	3
Low	20	0	0	6	3	11

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 DN404, fuzz-testing with Foundry 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 up to a depth of 500 with a prepared Foundry fuzzing suite.

ID	Description	Tested	Passed	Remediation	Run Count
DN-01	Sum of Owned NFTs == Mirror Total Supply	✅	✅	✅	1,000,000+
DN-02	Sum of Owned ERC20 == Token Total Supply	✅	✅	✅	1,000,000+
DN-03	Mirror And Base Token Are Unchanged Post-Initialization	✅	✅	✅	1,000,000+
DN-04	Burned Pool Length == Tail - Head	✅	✅	✅	1,000,000+
DN-05	No User Owns type(uint32).max NFT	✅	✅	✅	1,000,000+
DN-06	Allowance Matches Approved Amount	✅	✅	✅	1,000,000+
DN-07	Owner Auxiliary Data Is Not Modified Upon Approval	✅	✅	✅	1,000,000+
DN-08	Spender Auxiliary Data Is Not Modified Upon Approval	✅	✅	✅	1,000,000+
DN-09	Direct Transfers Do Not Overlap With Burned Pool Upon Transfer	✅	❌	✅	1,000,000+
DN-10	ERC20 Balance Changes By Amount For Sender And Receiver Upon Transfer	✅	✅	✅	1,000,000+

Invariants Assessed

ID	Description	Tested	Passed	Remediation	Run Count
DN-11	ERC20 Balance Remains The Same Upon Self-Transfer	✅	✅	✅	1,000,000+
DN-12	ERC20 Total Supply Remains The Same Upon Transfer	✅	✅	✅	1,000,000+
DN-13	Auxiliary Data Is Not Modified Upon Transfer	✅	✅	✅	1,000,000+
DN-14	Direct Transfers Do Not Overlap With Burned Pool Upon TransferFrom	✅	❌	✅	1,000,000+
DN-15	ERC20 Balance Changes By Amount For Sender And Receiver Upon TransferFrom	✅	✅	✅	1,000,000+
DN-16	ERC20 Balance Is the Same Upon Self-Transfer Upon TransferFrom	✅	✅	✅	1,000,000+
DN-17	Auxiliary Data Is Not Modified Upon TransferFrom	✅	✅	✅	1,000,000+
DN-18	User Balance Increased By Mint Amount	✅	✅	✅	1,000,000+
DN-19	Total ERC20 Supply Increased By Mint Amount	✅	✅	✅	1,000,000+
DN-20	Total NFT Supply Post-Mint Is At Least Total NFT Supply Pre-Mint	✅	✅	✅	1,000,000+
DN-21	Auxiliary Data Is Not Modified Upon Mint	✅	✅	✅	1,000,000+
DN-22	User's Owned NFT's Decreased Upon Burn	✅	✅	✅	1,000,000+
DN-23	ERC20 Total Supply Decreased By Burn Amount	✅	✅	✅	1,000,000+
DN-24	Total NFT Supply Post-Burn Is At Most Total NFT Supply Pre-Burn	✅	✅	✅	1,000,000+

Invariants Assessed

ID	Description	Tested	Passed	Remediation	Run Count
DN-25	Auxiliary Data Is Not Modified Upon Burn	✅	✅	✅	1,000,000+
DN-26	Approved NFT Spender == Requested Approval	✅	✅	✅	1,000,000+
DN-27	Owner Of NFT ID Is Not Modified Upon Approval Of NFT	✅	✅	✅	1,000,000+
DN-28	Owner Auxiliary Data Is Not Modified Upon Approval	✅	✅	✅	1,000,000+
DN-29	Spender Auxiliary Data Is Not Modified Upon Approval	✅	✅	✅	1,000,000+
DN-30	NFT Balance Of Sender and Receiver Accurately Updated Upon TransferNFT	✅	✅	✅	1,000,000+
DN-31	Sender/Receiver ERC20 Balance Decremented/Incremented By Unit	✅	✅	✅	1,000,000+
DN-32	Receiver Address Is The Owner At The Sent NFT ID	✅	✅	✅	1,000,000+
DN-33	Total NFT Supply Is Unchanged Upon NFT Transfer	✅	✅	✅	1,000,000+
DN-34	Approval Is Reset Upon NFT Transfer	✅	✅	✅	1,000,000+
DN-35	Sender Auxiliary Data Is Not Modified Upon NFT Transfer	✅	✅	✅	1,000,000+
DN-36	Receiver Auxiliary Data Is Not Modified Upon NFT Transfer	✅	✅	✅	1,000,000+
DN-37	Skip NFT Status Is Updated To Requested Status	✅	✅	✅	1,000,000+

Invariants Assessed

ID	Description	Tested	Passed	Remediation	Run Count
DN-38	Auxiliary Data Is Not Modified Upon Set Skip NFT	✅	✅	✅	1,000,000+
DN-39	Set Approval For All Updated To Requested Status	✅	✅	✅	1,000,000+
DN-40	Owner Auxiliary Data Is Not Modified Upon Set Approval For All	✅	✅	✅	1,000,000+
DN-41	Spender Auxiliary Data Is Not Modified Upon Set Approval For All	✅	✅	✅	1,000,000+
DN-42	Mint Next Does Not Overlap With Burned Pool	✅	❌	✅	1,000,000+
DN-43	Mint Next Increases User's Owned NFT's If NFT Minted	✅	✅	✅	1,000,000+
DN-44	User Balance Increased By Mint Next Amount	✅	✅	✅	1,000,000+
DN-45	Total ERC20 Supply Increased By Mint Next Amount	✅	✅	✅	1,000,000+
DN-46	_ownerAt(id) Is Always The Same As NFT Holder	✅	❌	✅	1,000,000+

Findings & Resolutions

ID	Title	Category	Severity	Status
H-01	_mintNext Allows NFTs In The Burn Pool To Be Stolen	Logical Error	High	Resolved
H-02	_mintNext Unexpectedly Wraps NFT Ids	Logical Error	High	Resolved
M-01	NFT Marketplace Bidding Bait-And-Switch	Protocol Gaming	Medium	Acknowledged
M-02	Double ERC721 Minting Via AfterNFTTransfer Hook	Logical Error	Medium	Resolved
M-03	Address Initialization Allows Pools To Accumulate NFTs	Protocol Gaming	Medium	Resolved
M-04	Missing tokenId Existence Check	Unexpected Behavior	Medium	Resolved
L-01	Missing DN404 supportsinterface Check Upon Linking	Logical Error	Low	Resolved
L-02	Lacking safeMint Functionality	Missing Feature	Low	Partially Resolved
L-03	Initial Supply Owner Always Skips NFT Minting	Logical Error	Low	Resolved
L-04	SkipNFTSet Emitted When No Changes Are Done	Logical Error	Low	Acknowledged
L-05	NextTokenID Not Updated On _mintNext	Logical Error	Low	Acknowledged
L-06	ERC721 Minting May Revert Due To Out Of Gas	Logical Error	Low	Resolved
L-07	Permit2 Infinite Allowance Can Overwrite User Allowance	Logical Error	Low	Resolved

Findings & Resolutions

ID	Title	Category	Severity	Status
L-08	NFT Minted With A Higher ID Than Available NFTs	Logical Error	Low	Resolved
L-09	Undocumented LIFO NFT Transfer Logic	Logical Error	Low	Resolved
L-10	ERC20 And ERC721 Simultaneous Allowances	Logical Error	Low	Partially Resolved
L-11	Base Token Holders Control Mirror NFT totalSupply	Logical Error	Low	Resolved
L-12	Operator In setApprovalForAll Can Be Zero Address	Validation	Low	Partially Resolved
L-13	Inaccurate _findFirstUnset Documentation	Documentation	Low	Resolved
L-14	Typographical Error	Typo	Low	Resolved
L-15	Approve Can Be Called Before Initialization	Unexpected Behavior	Low	Acknowledged
L-16	DN404 Tokens Cannot Take Over The World	DoS	Low	Acknowledged
L-17	_initiateTransferFromNFT Errant Documentation	Documentation	Low	Resolved
L-18	Deployer Address Can Be Immutable	Optimization	Low	Acknowledged
L-19	_totalSupplyOverflows Errant Documentation	Documentation	Low	Resolved
L-20	Unit Value Alteration DoS	Unexpected Behavior	Low	Acknowledged

H-01 | _mintNext Allows NFTs In The Burn Pool To Be Stolen

Category	Severity	Location	Status
Logical Error	High	DN404.sol: 501	Resolved

Description PoC

The `_mintNext` function ignores the burn pool tokenIds, and mints directly starting from the existing `totalSupply / unit() + 1` id.

However this id and the ids that follow can be within the burn pool. As a result burn pool NFTs will be minted without adjusting the burn pool head.

Therefore when attempting to mint regularly, these newly minted NFT ids will be duplicated in the `toOwned` mapping and overwritten in the `oo` entry, effectively stealing this NFT from the user it was originally minted to with the `_mintNext` function.

Recommendation

Do not allow the burn pool feature to be used in tandem with the `_mintNext` function, otherwise refactor the `_mintNext` function to account for the burn pool.

Resolution

DN404 Team: The issue was resolved in PR#136.

H-02 | _mintNext Unexpectedly Wraps NFT Ids

Category	Severity	Location	Status
Logical Error	High	DN404.sol: 434, 442, 514, 526	Resolved

Description PoC

In the _mintNext function it is possible for the minted ERC721 IDs to wrap around unexpectedly, causing several potential issues for systems inheriting the DN404 contract. Consider the following scenario:

- User A has a balance of 3.6 erc20s
- Existing totalSupply is 42.55
- We _mintNext 74.43 tokens

The _mintNext function will wrap the final minted ERC721 id with the _wrapNFTId function, this is because we are attempting to mint 75 nfts to the receiver's address, however the maxId has only increased by 74. This is because:

For User A: 3.6 + 74.43 = 78.03 => +75 ERC721s for User A
For totalSupply & maxId: 42.55 + 74.43 = 116.98 => + 74 ERC721s allowed by maxId.

As a result, the _mintNext function can unexpectedly mint ERC721 IDs that are a part of the burn pool, and burn pool NFTs will be minted without adjusting the burn pool head. Therefore when attempting to mint regularly, these newly minted NFT ids will be duplicated in the toOwned mapping and overwritten in the oo entry, effectively stealing this NFT from the user it was originally minted to with the _mintNext function.

Recommendation

In the _mintNext function revert if the toAddress would receive more ERC721s than the maxId increase would allow, e.g. _zeroFloorSub(t.toEnd, toIndex) > (totalSupply_ / _unit()) - preTotalSupply.

This behavior is also present in the _mint and _transfer functions, however there are no assumptions broken by this edge case for these functions. Therefore no code changes are necessary in these functions. Consider documenting this edge case behavior for these functions for users.

Resolution

DN404 Team: The issue was resolved in PR#136.

M-01 | NFT Marketplace Bidding Bait-And-Switch

Category	Severity	Location	Status
Protocol Gaming	Medium	Global	Acknowledged

Description PoC

Whenever a mirror ERC721 token is transferred, minted, or burned through ERC20 transfers, the order in which the mirror tokens are taken from the holder is LIFO (last in first out). This behavior can be abused in certain circumstance by a malicious actor to steal user funds when interacting with NFT marketplaces.

The exact situation happens when a user that has a bid on a mirror NFT also sells a unit or more of base tokens from the same wallet. In that case, a malicious holder of the mirror NFT can make a profit and leave the user without the rare NFT.

Attack scenario using the live Asterix collection as an example:

- Bob has one of the rarest NFTs, 1960 and waits for people to bid on it
- At this point, Alice bids on it for 1.2938 WETH
- The floor for the collection is 0.674 ETH and buying an NFT by buying the base tokens from the liquidity pool is 0.5671 ETH.
- Alice has another NFT, lowest rarity, and sells by selling a unit of base tokens
- Since the collection is on Ethereum, the base swap transaction can be seen in the mempool
- Bob sees the base unit sell and front-runs it with accepting Alice's bid
- Alice gets the rare 1960 ID and pays 1.2938 WETH, but immediately loses it as it was the last in and the base unit sell burns it, marking her a loss of 1.2938 - 0.5671 ETH
- Bob quickly initiates several cycled mints/burns to reclaim the rare NFT

Even if Bob fails to reclaim it, Alice still suffered a loss of the rare NFT. The above case can also happen unintentionally, when a user bid is accepted exactly in the same block as him selling base tokens equivalent to a NFT, having the same financial loss.

Recommendation

Consider modifying the synchronization logic to that of a FIFO (first in first out) instead of LIFO. Meaning that the first NFT to be minted to the wallet is also the first to leave it. Otherwise consider clearly documenting this risk for integrators.

Resolution

DN404 Team: Acknowledged.

M-02 | Double NFT Minting Via AfterNFTTransfer Hook

Category	Severity	Location	Status
Logical Error	Medium	DN404.sol: 699-703	Resolved

Description PoC

Contracts that extend DN404 can implement the `_afterNFTTransfer` hook to be executed after any NFT token transfers, including minting and burning. An attacker can abuse any implementations that pass access to holders from within these hooks as there are situations when the `_afterNFTTransfer` function is called before the internal storage is committed, breaking CEI.

Consider the following attack scenario in a system implementing the `_afterNFTTransfe` hook:

- An approver initiates a transfer to a user where direct transfers will be made
- In the `_afterNFTTransfer` function the malicious attacker has an existing helper contract that directly transfers several ERC721 tokens to himself
- Since this was done before the `balanceOf` equivalent was updated, after the initial execution is finalized, the malicious strategy will have a smaller ERC721 balance then the number of ERC721 it owns
- As the attacker has less NFTs then they should by the `ownedLength` variable and because this `ownedLength` is used in determining how many NFT a user will have for their base token amount, the attacker can transfer any amount, even 0, to himself and the contract will mint him extra ERC721 tokens, the exact number that was sent by the helper contract

At this point, an attacker owns double the ERC721 tokens that he received, for half the amount of ERC20 base required to own that many.

Recommendation

Completely move the `_afterNFTTransfer` into its own separate loop in all cases except the call from `_transferFromNFT` which is already singular. In the particular case of direct transfer, also move it after setting the `ownedLength`.

Resolution

DN404 Team: The issue was resolved in PR#136.

M-03 | Address Initialization Allows Pools To Accumulate NFTs

Category	Severity	Location	Status
Protocol Gaming	Medium	DN404.sol: 962	Resolved

Description

Upon the first interaction with an address the address data flags are initialized, checking to see if the address holds any bytecode. If the account does not hold any bytecode at the time of account initialization then it receives only the `_ADDRESS_DATA_INITIALIZED_FLAG` flag.

However if the address is later deployed to (e.g. a new pool is created at this address), the contract will not be excluded from ERC721 minting, as it will not automatically receive the `_ADDRESS_DATA_SKIP_NFT_FLAG` flag.

As a result malicious actors may transfer tokens to an address where a pool for the token is about to be deployed to in order to avoid getting the address marked with the `_ADDRESS_DATA_SKIP_NFT_FLAG`. NFTs will then errantly be minted to and burned from the pool address upon liquidity modification and swaps.

This creates a pool of NFTs which are not owned by end users, but instead locked up in a swap pool where a significant amount of them may not be able to move due to locked liquidity. In the case of a swap pool it would be possible for a flashloan to rescue potentially rare ERC721 tokens, however in the case of a locking contract or some other arbitrary integration these ERC721 tokens could be unintentionally locked for a significant amount of time.

Recommendation

Consider checking whether an account houses bytecode when reading the `getSkipNFT` function, regardless of if the account has been initialized or not. Otherwise be sure to document this risk to users of DN404, advising them to implement their own `_getSkipNFT` functions or adding their own functionality for trusted addresses to mark addresses with the `_ADDRESS_DATA_SKIP_NFT_FLAG` as needed.

Resolution

DN404 Team: The issue was resolved in PR#136.

M-04 | Missing tokenId Existence Check

Category	Severity	Location	Status
Unexpected Behavior	Medium	DN404.sol: 1172	Resolved

Description

When a `tokenURI` function is called in the fallback of the DN404 contract, it is never checked whether the given `tokenId` exists. According to the recommendation in EIP712, the `tokenURI` function should throw an error if the `tokenId` is not a valid NFT. This recommendation is followed in the OpenZeppelin ERC721 implementation contract as well as the ERC721A implementation.

Recommendation

Check if the given `tokenId` exists; if it is invalid, revert with a custom error.

Resolution

DN404 Team: The issue was resolved in PR#136.

L-01 | Missing DN404 supportsInterface Check Upon Linking

Category	Severity	Location	Status
Logical Error	Low	DN404Mirror.sol: 407	Resolved

Description

The documentation for the `CannotLink` error states that this error is thrown when "linking to the DN404 base contract and the DN404 `supportsInterface` check fails or the call reverts". However while linking the Mirror contract with the `linkMirrorContract(address)` function selector there is no validation that the `msg.sender` is indeed a valid DN404 implementation.

Recommendation

Implement the appropriate validation when executing the `linkMirrorContract` logic such that `implementsDN404()` function selector is invoked on the purported DN404 contract to ensure that it is a valid implementation.

Resolution

DN404 Team: The issue was resolved by removing the `CannotLink` error in PR#136.

L-02 | Lacking safeMint Functionality

Category	Severity	Location	Status
Missing Feature	Low	DN404.sol: 422, 501	Partially Resolved

Description

Currently, during the transfer of tokens, there is a `safeTransfer` mechanism where the `to` address is checked if it implements the `onERC721Received` function. However, this `safe` mechanism does not exist during minting. Therefore, it is possible to mint a token to a smart contract that does not support ERC721, resulting in the token becoming permanently stuck.

Recommendation

Implement `_safeMint` function with the same `safe` mechanism as in the safeTransferFrom functions. If the `to` address is a smart contract, check if it implements the `onERC721Received` function.

Resolution

DN404 Team: The issue was documented in PR#136.

L-03 | Initial Supply Owner Always Skips NFT Minting

Category	Severity	Location	Status
Documentation	Low	DN404.sol: 239-240	Resolved

Description

When the `DN404` contract is deployed, an initial supply amount and holder address can be passed. If they are provided, the internal logic within the contract will automatically set the corresponding address to skip NFT minting, regardless if it is an EOA or smart contract.

This behavior is not stated and cannot be intuitively considered since the skip NFT logic, as suggested by the EIP and implemented, is implemented to by default skip only smart contracts, where as here any provided address is set as so.

As a result external integrators expecting that by sending the initial `ERC20` base tokens to an EOA to have the `ERC721` minted as well to it are mislead.

Recommendation

Either clearly document this behavior or change the implementation of `DN404._initializeDN404` such that it accepts a `skipNFT` parameter and mints the mirror contract tokens accordingly.

Resolution

DN404 Team: The issue was documented in PR#136.

L-04 | SkipNFTSet Emitted When No Changes Are Done

Category	Severity	Location	Status
Logical Error	Low	DN404.sol: 949-952	Acknowledged

Description

In EIP-7631, if the ERC20 base contract implements the `IERC7631BaseNFTSkippable` interface there are certain considerations that must be upheld. As specified in the interface, the `SkipNFTSet` event must be "*Emitted when the skip NFT status of owner is changed by any mechanism*" with the addition that the *"initial skip NFT status for owner can be dynamically chosen to be true or false, but any changes to it MUST emit this event"*.

In the DN404 implementation, the `SkipNFTSet` event is incorrectly emitted both when no change is done, by calling `setSkipNFT` with an already set status, and when setting the status for the initial supply owner, when deploying the DN404 token.

Emitting the `SkipNFTSet` event in situations where no change is done is not compliant with the EIP.

Recommendation

In the `DN404._setSkipNFT` function, move the event emission assembly block within the `if` branch. By doing so, the `_setSkipNFT` call from the `DN404._initializeDN404` will also not emit the event.

If the `_setSkipNFT` alteration is not implemented, in `DN404._initializeDN404` instead of calling `_setSkipNFT`, directly set the `_ADDRESS_DATA_SKIP_NFT_FLAG` flag to true, to avoid event emission.

Resolution

DN404 Team: Acknowledged.

L-05 | NextTokenID Not Updated On _mintNext

Category	Severity	Location	Status
Documentation	Low	DN404.sol: 501-563	Acknowledged

Description

There are 2 functions that provide minting functionality for the ERC20 and ERC721 balances:

- `_mint`: mints IDs from the burn pool if using the feature; If none available, uses `nextTokenId`
- `_mintNext`: will always mint the next ID outside of the existing supply

An issue appears when a contract uses both mint functions to mint tokens and NFTs as the `_mintNext` function does not update the `nextTokenId` which indicates the next free token ID.

Consider the following scenario:

- For an implementing contract, transfers are paused until all the base tokens are sent, meaning the collection is minted
- Users mint using the `_mintNext` function, which does not update the `nextTokenId`
- After a large number of mints, the contract changes minting, for whatever reason, to using the `_mint` function
- `_mint` starts validating ownership from the `nextTokenId` token ID, and since it was never set, if a large enough amount of NFTs were minted, this operation will consume a large amount of gas or even revert with OOG in extreme cases before finding the next free ID.

Recommendation

Consider if this behavior should be allowed, if so clearly document this risk so integrating protocols can avoid this scenario.

Resolution

DN404 Team: Acknowledged.

L-06 | ERC721 Minting May Revert Due To Out Of Gas

Category	Severity	Location	Status
Documentation	Low	Global	Resolved

Description

The DN404 implementation co-joins ERC20 and ERC721 standards representing dual nature token pair. Both tokens are deployed separately and then linked. The main idea of solution is to adjust both tokens balances, in such a way that the `_unit` represents an amount of ERC20 token balance that is equal to one NFT.

Thus, the balances are being updated within every burn, mint or transfer operation. Whenever significant amount of tokens are processed, that result in multiple mints or burns, a significant amount of gas is consumed. Eventually, such transactions are prone to revert due to Out Of Gas error, preventing successful execution of aforementioned operations.

Although this is mentioned as a possibility in the EIP7631 itself, it should be mentioned that the current implementation fails at around 2,600 NFTs to be minted. If the unit is low enough, complex integrators, swaps, lenders, will also fail.

Recommendation

Consider documenting these limitations, especially when it comes to the choice of a `unit` value.

Resolution

DN404 Team: The issue was documented in PR#136.

L-07 | Permit2 Infinite Allowance Can Overwrite User Allowance

Category	Severity	Location	Status
Documentation	Low	DN404.sol: 336	Resolved

Description

`DN404` integrators have the possibility to define whether Permit2 has infinite allowances by default for all owners, by means of the `_givePermit2DefaultInfiniteAllowance` internal function.

Additionally, when a user decides to overwrite the default allowance for `Permit2` the `_ADDRESS_DATA_OVERRIDE_PERMIT2_FLAG` is set to remember the users choice of a custom allowance set.

However, the flag is only set when the `_givePermit2DefaultInfiniteAllowance` function returns `true`. Integrators may have implementations that change the value returned in the aforementioned function over time.

In such a case, the user may firstly set custom allowance for Permit2. Such action will not set on the `_ADDRESS_DATA_OVERRIDE_PERMIT2_FLAG` flag, as the `_givePermit2DefaultInfiniteAllowance` function by default returns `false`.

Subsequently, changing the behavior of the `_givePermit2DefaultInfiniteAllowance` function will overwrite user's custom allowance into maximum allowance. This behavior can be considered unexpected and may be leveraged by the attacker in further attacks.

Recommendation

Clearly document this risk that arises from a non-static `_givePermit2DefaultInfiniteAllowance` value.

Resolution

DN404 Team: The issue was documented in PR#136.

L-08 | NFT Minted With A Higher ID Than Available NFTs

Category	Severity	Location	Status
Logical Error	Low	DN404.sol: 518	Resolved

Description

When minting using the `_mintNext` function, IDs are chosen outside of the current NFT supply as distinctly new NFTs are created. However it is possible to mint NFTs with an ID that is above the computed `totalNFTSupply` retrieved by the `totalSupply() / _unit()` calculation.

This corner case appears because the index where the `_mintNext` function starts looking for available IDs is chosen as if all the possible NFTs would of been minted up to this point `startId = preTotalSupply / _unit() + 1` and the wrapping function is not triggered due to the ID being free of an owner.

This creates odd situations, for example if the unit is 100 ERC20 tokens and 90 ERC20 tokens are minted to 2 addresses each as well as 20 ERC20 tokens to a 3rd address. Now, when initiating a mint of 10 base tokens to any of the first addresses, the ID minted will be 3, however the `totalSupply() / _unit() = 210 / 100 = 2` indicates that only the IDs 1 and 2 ought to exist.

Recommendation

When implementing a fix for H-01, be sure to not allow the ID of minted NFTs with the `_mintNext` function to surpass the `totalSupply / _unit()`.

Resolution

DN404 Team: The issue was resolved in PR#136.

L-09 | Undocumented LIFO NFT Transfer Logic

Category	Severity	Location	Status
Documentation	Low	Global	Resolved

Description

Whenever a mirror ERC721 token is transferred/minted/burned as a result of transferring enough base ERC20 tokens, the order in which the mirror tokens are taken from the holder is LIFO (last in first out). This behavior has a high impact on users and is virtually unmentioned.

Additionally, as the current draft of EIP-7631 does not cover the token synchronization logic it needs to be explicitly stated.

Consider a situation where, out of on the Nth base token transfer, a user mints a rare ERC721. Since they are only interested in the rare token, they send their entire base balance, minus enough for a single unit, to another address and by doing so actually lose the rare NFT.

Recommendation

Although this is a design decision, since it has severe economical implications to market participants it must be explicitly stated. Another solution is to add a mechanism to specify which NFTs you wish to have locked-in when transferring base tokens.

Users can currently workaround this by transferring NFT directly to a different (clean) wallet. Any subsequent base `ERC20` transfer to it will leave the rare NFT within the wallet of the user provided that a minimum unit value is always kept.

Resolution

DN404 Team: The issue was documented in PR#136.

L-10 | ERC20 And ERC721 Simultaneous Allowances

Category	Severity	Location	Status
Logical Error	Low	Global	Partially Resolved

Description

The DN404 implementation co-joins ERC20 and ERC721 representing dual nature token pair. Both tokens are deployed separately and then linked. The processing of both tokens is being done simultaneously whenever a transfer, mint or burn is triggered for one of them.

However, the same does not occur for ERC20 and ERC721 token allowances. Whenever one of the token allowance is granted or revoked, the other remain unchanged. This behavior appear to be secure whenever allowance increases or grant is done.

When considering the opposite situation, whenever allowance is decreased or revoked, it can be prone to human errors. In the event of emergency, when one of the contract allowance is removed, the second will remain active, and might be leveraged by an attacker in a vulnerable scenario.

Recommendation

Consider implementing additional functionality, that sets both the ERC20 allowance to 0 and and the ERC721 operator allowance is revoked.

Resolution

DN404 Team: The issue was documented in PR#136.

L-11 | Base Token Holders Control Mirror NFT totalSupply

Category	Severity	Location	Status
Documentation	Low	Global	Resolved

Description

The `setSkipNFT` function allows the token holder to decide whenever the NFT should be minted upon increasing their base ERC20 tokens balance. While this functionality may have impact on Gas consumption and gives some flexibility, it has significant drawback.

A user can leverage this to burn owned NFTs and decrease the `totalSupply` by transferring base ERC20 tokens to secondary account which has this flag set to true. As a result the NFT tokens will be burned, but not minted for destination account.

This implementation may have impact on third party integration, e.g. if marketplace will base the NFT token value on total supply.

Recommendation

Clearly highlight the indicated behavior so that any integrating 3rd party does not unknowingly integrate expecting the `totalSupply` invariant to hold.

Resolution

DN404 Team: The issue was documented in PR#136.

L-12 | Operator In setApprovalForAll Can Be Zero Address

Category	Severity	Location	Status
Validation	Low	DN404.sol: 1070-1077	Partially Resolved

Description

A user can allow an operator to manage the tokens of their mirror ERC721 tokens by calling the standard `setApprovalForAll` function.

This function is usually gated to not allow approval to be given to the zero address. In the current `DN404Mirror` implementation, zero address operator is allowed.

Recommendation

Validate that the operator passed to the `_setApprovalForAll` function is not the zero address.

Resolution

DN404 Team: The issue was documented in PR#136.

L-13 | Inaccurate _findFirstUnset Documentation

Category	Severity	Location	Status
Documentation	Low	DN404Mirror.sol: 207	Resolved

Description

In the documentation for the `_findFirstUnset` function it is mentioned that *"If no set bit is found, returns type(uint256).max"*.

However the behavior of the `_findFirstUnset` function is such that if no *unset* bit is found, the `type(uint256).max` is returned.

Recommendation

Correct the comment to reflect the behavior: *"If no unset bit is found, returns type(uint256).max"*.

Resolution

DN404 Team: The issue was resolved in PR#136.

L-14 | Typographical Error

Category	Severity	Location	Status
Typo	Low	DN404Mirror.sol: 87	Resolved

Description

In the documentation for the `DN404NFTStorage` struct, the comment about the deployer *"...link can only be done be the deployer via the ERC20 base contract"*.

Should read *"...link can only be done by the deployer via the ERC20 base contract"*.

Recommendation

Replace the second instance of *"be"* with *"by"*.

Resolution

DN404 Team: The issue was resolved in PR#136.

L-15 | Approve Can Be Called Before Initialization

Category	Severity	Location	Status
Unexpected Behavior	Low	DN404.sol: 886	Acknowledged

Description

In the DN404 contract the public approve function can be called before the DN404 contract has been initialized and connected to a corresponding `mirrorERC721` contract.

While this poses no immediate risk it may be unexpected for users of the DN404 contract that approvals can be made and approval events can be emitted before the DN404 contract is initialized.

Recommendation

Consider adding validation that the `mirrorERC721` contract is nonzero in the `_approve` function and revert with the `DNNotInitialized` error if it is.

Resolution

DN404 Team: Acknowledged.

L-16 | DN404 Tokens Cannot Take Over The World

Category	Severity	Location	Status
DoS	Low	DN404.sol: 980	Acknowledged

Description

Address aliases are used to track ownership in bytes32 size entries in the `oo` mapping.
In the `_registerAndResolveAlias` function, the resulting `addressAlias` is a monotonically increasing number. When `type(uint32).max = 4,294,967,295` address aliases have been assigned, attempting to assign another alias will revert.

There are roughly 8,000,000,000 humans on earth, notice that `8,000,000,000 > 4,294,967,295`.
As a result DN404 tokens cannot reach 100% saturation of the human race, limiting their TAM.

Recommendation

Consider documenting this limitation for users of DN404 expecting to convert humanity into a hive-mind of DN404 token holders.

Resolution

DN404 Team: Acknowledged.

L-17 | _initiateTransferFromNFT Errant Documentation

Category	Severity	Location	Status
Documentation	Low	DN404.sol: 789	Resolved

Description

In the DN404 contract the documentation for the `_initiateTransferFromNFT` function indicates that the *"Call must originate from the mirror contract"*. However there is no in-code validation that specifically requires this to hold.

Additionally, there is no dash preceding the *"msgSender must be the owner of the token, or be approved to manage the token"* requirement, therefore this requirement does not match the format of the others.

Recommendation

Either remove the documented requirement that the *"Call must originate from the mirror contract"* or implement this validation at the Smart Contract level.

Additionally add a dash preceding the *"msgSender must be the owner of the token, or be approved to manage the token"* requirement so that it matches the formatting of other requirements in the docstring.

Resolution

DN404 Team: The issue was resolved in PR#136.

L-18 | Deployer Address Can Be Immutable

Category	Severity	Location	Status
Optimization	Low	DN404Mirror.sol: 88	Acknowledged

Description

The deployer address in the Mirror contract can be immutable as it is never changed and is only used during the linking of the DN404 contract with the Mirror contract.

Recommendation

Make the deployer address immutable.

Resolution

DN404 Team: Acknowledged.

L-19 | _totalSupplyOverflows Errant Documentation

Category	Severity	Location	Status
Documentation	Low	DN404.sol: 1361	Resolved

Description

In the `DN404` contract the documentation for the `_totalSupplyOverflows` function indicates that it: *"Returns whether amount is a valid totalSupply"*.

However, the function returns whether the amount is *not* a valid total supply, specifically it returns true if it does not satisfy the required conditions.

Recommendation

Correct the comment to reflect the behavior: *Returns whether `amount` is an invalid `totalSupply`*.

Resolution

DN404 Team: The issue was resolved in PR#136.

L-20 | Unit Value Alteration DoS

Category	Severity	Location	Status
Unexpected Behavior	Low	DN404.sol: 258	Acknowledged

Description

In the `DN404` contract users may override the `unit` function in order to specify the denomination of ERC20 tokens that constitutes the ownership of 1 ERC721 token. There is no requirement nor documentation which specifies that users ought to keep the unit value constant, however significant risks arise when the unit value is changed.

Most notably, if the unit value is increased there is a risk that all ERC721 IDs from 1 up to the newly computed `maxId` are occupied when minting a new NFT. This scenario will result in an infinite loop while the `_mint` or `_transfer` function attempts to find an unoccupied ERC721 ID within the valid range, and as a result minting and transfers which would result in a mint are DoS'd.

Recommendation

Clearly document that the unit value should remain constant, similar to how is done for the `_useExistsLookup` function.

Resolution

DN404 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