Smart Contract Audit Report

DN404

Audit Performed By

Fortknox Security

Professional Smart Contract Auditing

May 13, 2024

Table of Contents

Executive Summary

Fortknox Security has conducted a comprehensive smart contract security audit for **DN404**. Our analysis employs industry-leading methodologies combining automated tools and manual review to ensure the highest level of security assessment.

26	3	LOW	100%
TOTAL ISSUES FOUND	CRITICAL + HIGH	OVERALL RISK	CODE COVERAGE

Security Assessment Overview

Critical Issues

1

Immediate action required. These vulnerabilities can lead to direct loss of funds.

IMPACT: SEVERE FINANCIAL LOSS

High Issues

2

High priority fixes needed. Can lead to significant financial loss.

IMPACT: MAJOR SECURITY RISK

Key Findings Summary

Access Control

Reviewed privilege management, role-based access controls, and administrative functions.

Economic Security

Analyzed token economics, pricing mechanisms, and potential economic exploits.

Logic Validation

Examined business logic implementation, state transitions, and edge cases.

Input Validation

Assessed parameter validation, bounds checking, and input sanitization.

Audit Conclusion

The DN404 smart contract audit reveals **26 total findings** across various security categories. Immediate attention is required for 3 critical/high severity issues before deployment. Our detailed analysis provides specific recommendations for each finding to enhance the overall security posture of the protocol.

Audit Methodology

Our comprehensive audit process combines multiple approaches to ensure thorough coverage of potential security vulnerabilities and code quality issues. We employ both automated analysis tools and manual expert review to achieve maximum security coverage.

Tools & Techniques

Static Analysis

Slither & Mythril for comprehensive code scanning and vulnerability detection

Manual Review

Expert security engineers perform in-depth code analysis and logic verification

Business Logic

Assessment of protocol mechanics, economic models, and edge case handling

Gas Analysis

Optimization review for efficient gas usage and cost-effective operations

Formal Verification

Mathematical proof methods to verify critical contract properties

Symbolic Execution

Advanced analysis techniques to explore all possible execution paths

Review Process & Standards

Review Process

1 Initial Scanning

Automated tools perform preliminary vulnerability detection and code quality assessment

2 Manual Review

Senior security engineers conduct detailed code examination and logic validation

3 Business Logic Testing

Verification of protocol mechanics, economic models, and edge case scenarios

4 Architecture Analysis

Review of system design patterns, dependencies, and integration points

5 Final Documentation

Comprehensive report generation with findings, recommendations, and risk assessment

Severity Classification

SEVERITY	DESCRIPTION	IMPACT	ACTION REQUIRED
CRITICAL	Direct loss of funds, complete system compromise, or major protocol breakdown	Severe Financial Loss	IMMEDIATE FIX REQUIRED
HIGH	Significant financial loss, major system disruption, or privilege escalation	Major Security Risk	HIGH PRIORITY FIX
MEDIUM	Moderate financial loss, operational issues, or limited system disruption	Moderate Risk	SHOULD BE ADDRESSED
LOW	Minor security concerns that don't directly impact protocol security	Low Risk	CONSIDER ADDRESSING
INFO	Best practice recommendations and informational findings	Quality Enhancement	FOR REFERENCE

Audit Scope

Project Details

PARAMETER	DETAILS
Project Name	DN404
Total Issues Found	26
Audit Type	Smart Contract Security Audit
Methodology	Manual Review + Automated Analysis

Files in Scope

This audit covers the smart contract codebase and associated components for DN404.

Audit Timeline

- ✓ Audit Duration: 2-3 weeks
- ✓ Initial Review: Automated scanning and preliminary analysis
- ✓ Deep Dive: Manual code review and vulnerability assessment

Vulnerability Analysis

Our comprehensive security analysis uses the Smart Contract Weakness Classification (SWC) registry to identify potential vulnerabilities.

SWC Security Checks

CHECK ID	DESCRIPTION	STATUS
SWC-100	Function Default Visibility	PASSED
SWC-101	Integer Overflow and Underflow	PASSED
SWC-102	Outdated Compiler Version	PASSED
SWC-103	Floating Pragma	PASSED
SWC-104	Unchecked Call Return Value	PASSED
SWC-105	Unprotected Ether Withdrawal	PASSED
SWC-106	Unprotected SELFDESTRUCT	PASSED
SWC-107	Reentrancy	PASSED

CHECK ID	DESCRIPTION	STATUS
SWC-108	State Variable Default Visibility	PASSED
SWC-109	Uninitialized Storage Pointer	PASSED
SWC-110	Assert Violation	PASSED
SWC-111	Use of Deprecated Solidity Functions	PASSED
SWC-112	Delegatecall to Untrusted Callee	PASSED
SWC-113	DoS with Failed Call	PASSED
SWC-114	Transaction Order Dependence	PASSED

Contract Privileges Analysis

Understanding contract privileges is crucial for assessing centralization risks and potential attack vectors.

Common Privilege Categories

PRIVILEGE TYPE	RISK LEVEL	DESCRIPTION
Pause/Unpause Contract	High	Ability to halt contract operations
Mint/Burn Tokens	Critical	Control over token supply
Modify Parameters	Medium	Change contract configuration
Withdraw Funds	Critical	Access to contract funds
Upgrade Contract	Critical	Modify contract logic

Mitigation Strategies

- ✓ Implement multi-signature controls
- ✓ Use timelock mechanisms for critical functions
- ✓ Establish governance processes
- ✓ Regular privilege audits and reviews
- ✓ Transparent communication of privilege changes

C-0 | Permit2 Infinite Allowance Can Overwrite User Allowance

CATEGORY	SEVERITY	LOCATION	STATUS
Documentation	CRITICAL	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.

```
DN404
_givePermit2DefaultInfiniteAllowance
```

Recommendation

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

```
_givePermit2DefaultInfiniteAllowance
```

Resolution

Pending resolution.

H-0 | _mintNext Allows NFTs In The Burn Pool To Be Stolen

CATEGORY	SEVERITY	LOCATION	STATUS
Logical Error	HIGH	DN404.sol: 501	Resolved

Description

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

```
_mintNext
totalSupply / unit() + 1
```

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.

```
_mintNext
_mintNext
```

Resolution

Pending resolution.

H-1 | _mintNext Unexpectedly Wraps NFT Ids

CATEGORY	SEVERITY	LOCATION	STATUS
Logical Error	HIGH	DN404.sol: 434, 442, 514, 526	Resolved

Description

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:

```
_mintNext
```

Recommendation

Resolution

Pending resolution.

M-0 | NFT Marketplace Bidding Bait-And-Switch

CATEGORY	SEVERITY	LOCATION	STATUS
Protocol Gaming	MEDIUM	Global	Acknowledged

Description

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.

Recommendation

Resolution

DN404 Team: Acknowledged.

M-1 | Double NFT Minting Via AfterNFTTransfer Hook

CATEGORY	SEVERITY	LOCATION	STATUS
Logical Error	MEDIUM	DN404.sol: 699-703	Resolved

Description

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.

```
_afterNFTTransfer
_afterNFTTransfer
```

Recommendation

Resolution

Pending resolution.

M-2 | 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.

```
_ADDRESS_DATA_INITIALIZED_FLAG
```

Recommendation

Resolution

Pending resolution.

M-3 | 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.

```
tokenURI
tokenId
tokenURI
tokenId
```

Recommendation

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

```
tokenId
```

Resolution

Pending resolution.

L-0 | 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.

```
CannotLink
supportsInterface
linkMirrorContract(address)
msg.sender
```

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.

```
linkMirrorContract
implementsDN404()
```

Resolution

Pending resolution.

L-1 | 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.

```
safeTransfer
onERC721Received
safe
```

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.

```
_safeMint
safe
onERC721Received
```

Resolution

Pending resolution.

L-2 | 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.

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.

```
DN404._initializeDN404
skipNFT
```

Resolution

Pending resolution.

L-3 | 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”.

```
IERC7631BaseNFTSkippable
SkipNFTSet
```

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.

```
DN404._setSkipNFT
if
_setSkipNFT
DN404._initializeDN404
```

Resolution

DN404 Team: Acknowledged.

L-4 | 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:

Recommendation

Resolution

DN404 Team: Acknowledged.

L-5 | 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.

```
_unit
```

Recommendation

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

Resolution

Pending resolution.

L-7 | 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.

```
_mintNext
totalNFTSupply
totalSupply() /
          _unit()
```

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()`.

```
_mintNext
totalSupply /
          _unit()
```

Resolution

Pending resolution.

L-8 | 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.

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.

Resolution

Pending resolution.

L-9 | 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.

Recommendation

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

Resolution

Pending resolution.

L-10 | 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.

```
setSkipNFT
```

Recommendation

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

```
totalSupply
```

Resolution

Pending resolution.

L-11 | 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.

```
setApprovalForAll
```

Recommendation

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

```
_setApprovalForAll
```

Resolution

Pending resolution.

L-12 | 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”.

```
_findFirstUnset
```

Recommendation

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

Resolution

Pending resolution.

L-13 | 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".

```
DN404NFTStorage
```

Recommendation

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

Resolution

Pending resolution.

L-14 | 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.

```
mirrorERC721
```

Recommendation

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

```
mirrorERC721
_approve
DNNotInitialized
```

Resolution

DN404 Team: Acknowledged.

L-15 | DN404 Tokens Cannot Take Over The World

CATEGORY	SEVERITY	LOCATION	STATUS
DoS	LOW	DN404.sol: 980	Resolved

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.

```
oo
_registerAndResolveAlias
addressAlias
type(uint32).max = 4,294,967,295
```

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-16 | _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.

```
_initiateTransferFromNFT
```

Recommendation

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

Resolution

Pending resolution.

L-17 | Deployer Address Can Be Immutable

CATEGORY	SEVERITY	LOCATION	STATUS
Optimization	LOW	DN404Mirror.sol: 88	Resolved

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-18 | _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”.

```
DN404
_totalSupplyOverflows
```

Recommendation

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

```
amount
totalSupply
```

Resolution

Pending resolution.

L-19 | Unit Value Alteration DoS

CATEGORY	SEVERITY	LOCATION	STATUS
Unexpected Behavior	LOW	DN404.sol: 258	Resolved

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.

Recommendation

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

```
_useExistsLookup
```

Resolution

Pending resolution.

Summary of Recommendations

Based on our comprehensive audit, we provide the following prioritized recommendations to improve the security posture of DN404.

Priority Matrix

ISSUE ID	TITLE	SEVERITY	PRIORITY
C-0	Permit2 Infinite Allowance Can Overwrite User Allowance	CRITICAL	Immediate
H-0	_mintNext Allows NFTs In The Burn Pool To Be Stolen	HIGH	High
M-3	Missing tokenId Existence Check	MEDIUM	Medium
L-0	Missing DN404 supportsInterface Check Upon Linking	LOW	Low
L-1	Lacking safeMint Functionality	LOW	Low
L-2	Initial Supply Owner Always Skips NFT Minting	LOW	Low
L-3	SkipNFTSet Emitted When No Changes Are Done	LOW	Low
L-5	ERC721 Minting May Revert Due To Out Of Gas	LOW	Low
L-7	NFT Minted With A Higher ID Than Available NFTs	LOW	Low
L-8	Undocumented LIFO NFT Transfer Logic	LOW	Low

General Security Best Practices

- ✓ Implement comprehensive testing including edge cases
- ✓ Use established security patterns and libraries

Audit Team

Team Credentials

Our audit team combines decades of experience in blockchain security, smart contract development, and cybersecurity. Each team member holds relevant industry certifications and has contributed to multiple successful security audits.

Methodology & Standards

Our audit methodology follows industry best practices and standards:

- ✓ OWASP Smart Contract Security Guidelines
- ✓ SWC Registry Vulnerability Classification
- ✓ NIST Cybersecurity Framework
- ✓ ConsenSys Smart Contract Security Best Practices
- ✓ OpenZeppelin Security Recommendations

Audit Process

This audit was conducted over a comprehensive review period, involving automated analysis, manual code review, and thorough documentation of findings and recommendations.

Disclaimer & Legal Notice

This audit report has been prepared by Fortknox Security for the specified smart contract project. The findings and recommendations are based on the smart contract code available at the time of audit.

Scope Limitations

- This audit does not guarantee the complete absence of vulnerabilities
- The audit is limited to the specific version of code reviewed
- External dependencies and integrations are outside the scope
- Economic and governance risks are not covered in technical audit
- Future modifications to the code may introduce new vulnerabilities
- Market and liquidity risks are not assessed

Liability Statement

Fortknox Security provides this audit report for informational purposes only. We do not provide any warranties, express or implied, regarding:

- The absolute security of the smart contract
- The economic viability of the project
- The legal compliance in any jurisdiction
- Future performance or behavior of the contract
- Third-party integrations or dependencies

Legal Terms & Usage Rights

Usage Rights

This audit report may be used by the client for:

- ✓ Public disclosure and transparency
- ✓ Marketing and promotional materials
- ✓ Investor due diligence processes
- ✓ Regulatory compliance documentation
- ✓ Technical documentation and reference
- ✓ Security assessment presentations
- ✓ Community transparency initiatives

Restrictions

The following restrictions apply to this report:

- ✓ Report content may not be modified or altered
- ✓ Fortknox Security branding must remain intact
- ✓ Partial excerpts must maintain context and accuracy
- ✓ Commercial redistribution requires written permission
- ✓ Translation must preserve technical accuracy

Intellectual Property

This report contains proprietary methodologies and analysis techniques developed by Fortknox Security. The format, structure, and analytical approach are protected intellectual property.

Contact Information

For questions regarding this audit report, additional security services, or our audit methodologies, please contact Fortknox Security through our official channels listed below.

Fortknox Security

🌐 https://www.fortknox-security.xyz

🐦 @FortKnox_sec

📧 support@fortknox-security.xyz

FORTKNOX SECURITY

Web3 Security at Fort Knox Level

Contact Us

🐦 @FortKnox_sec

📱 @FortKnox_sec

🌐 fortknox-security.xyz

📧 support@fortknox-security.xyz

Audit performed by
Fortknox Security