QuillAudits

Audit Report March, 2023

For

Kresus

Table of Content

Executive Summary

Project Name	Kresus Contracts
Overview	Kresus contracts facilitate management and curation of NFT collections for the extended Kresus ecosystem. A Proxy holds the NT's and manages admins, subadmins, minting, burning, transfers, bulk transactions and migration of NFT's. An NFT contract based on ERC1155 contracts is the NFT contract that is called by the proxy.
Timeline	6 February, 2023 - 2 March, 2023
Scope of Audit	The scope of this audit was to analyse Kresus Contracts codebase for quality, security, and correctness. This included testing of smart contracts to ensure proper logic was followed, manual analysis ,checking for bugs and vulnerabilities, checks for dead code, checks for code style, security and more. The audited contracts are as follows:

- IKresusNFT.sol
- KresusProxy.sol
- KresusNFT.sol

Git Repo link: *https://github.com/Kresus-Labs-Inc/nft-marketplace-contracts*
Git Branch: main
Commit Hash: 59b35a86e2fc6821f1fb681a454b7882ba0e8cd9
Explorer Links: *Kresus Proxy* *Kresus NFT*
Fixed in: Git Branch: NA
Commit Hash: NA

6
Issues Found

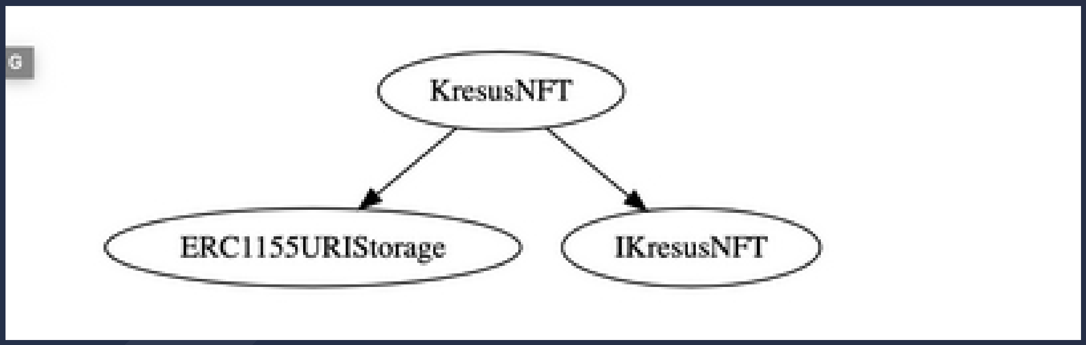
🟥 High 🟨 Medium
🟩 Low 🟪 Informational

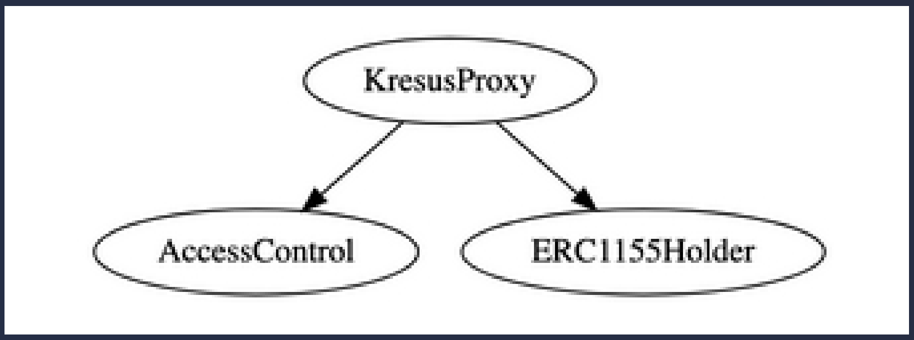
	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	**1**	**1**	**2**	**2**
Partially Resolved Issues	0	0	0	0
Resolved Issues	0	0	0	0

Contracts Information

Contract	Lines	Complexity Score	Capabilities
contracts/interfaces/IBeefyVault.sol	101	13	
contracts/KresusNFT.sol	235	55	
contracts/KresusProxy.sol	278	57	hash functions

Inheritance Graph



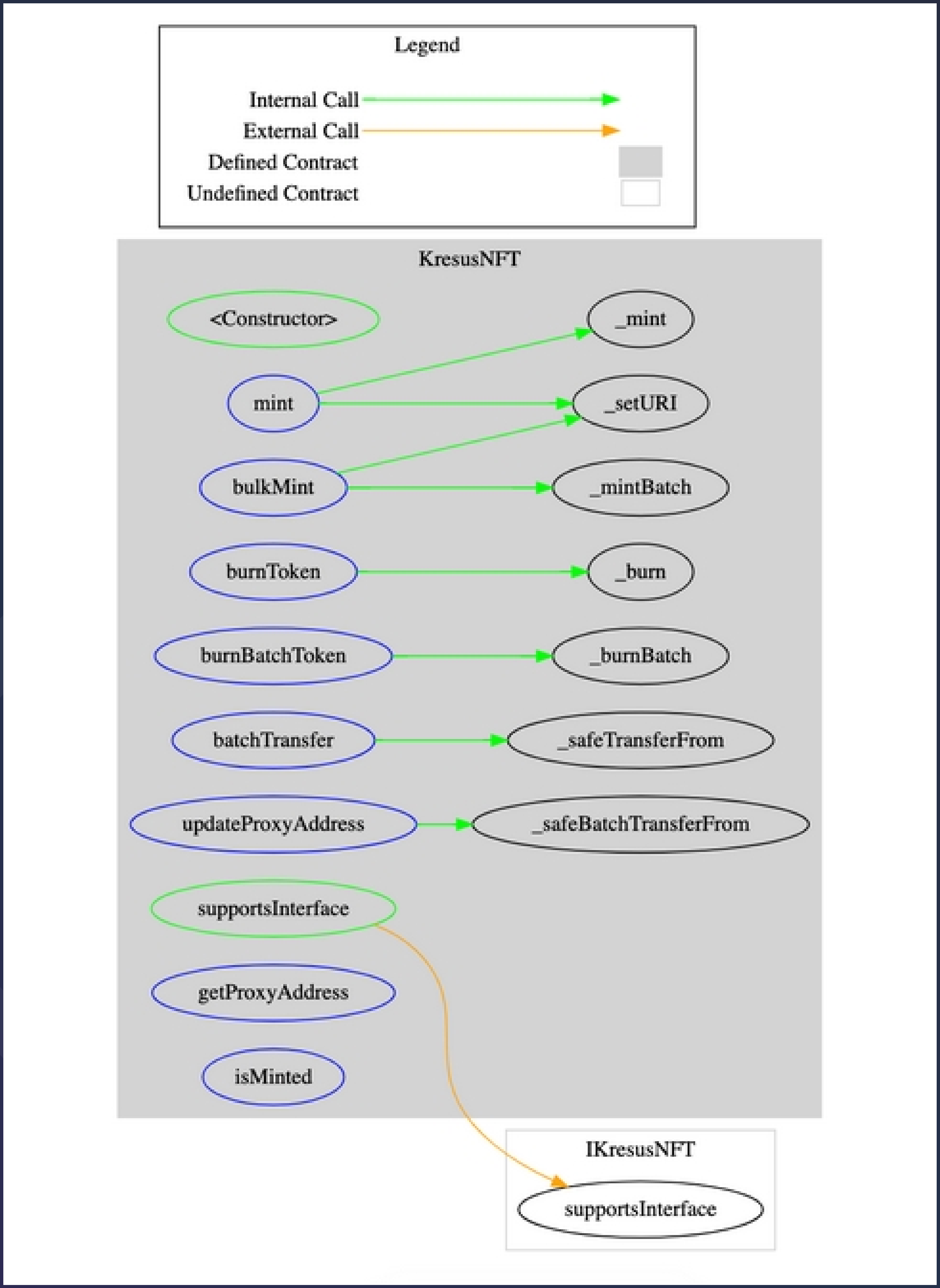


Dependencies

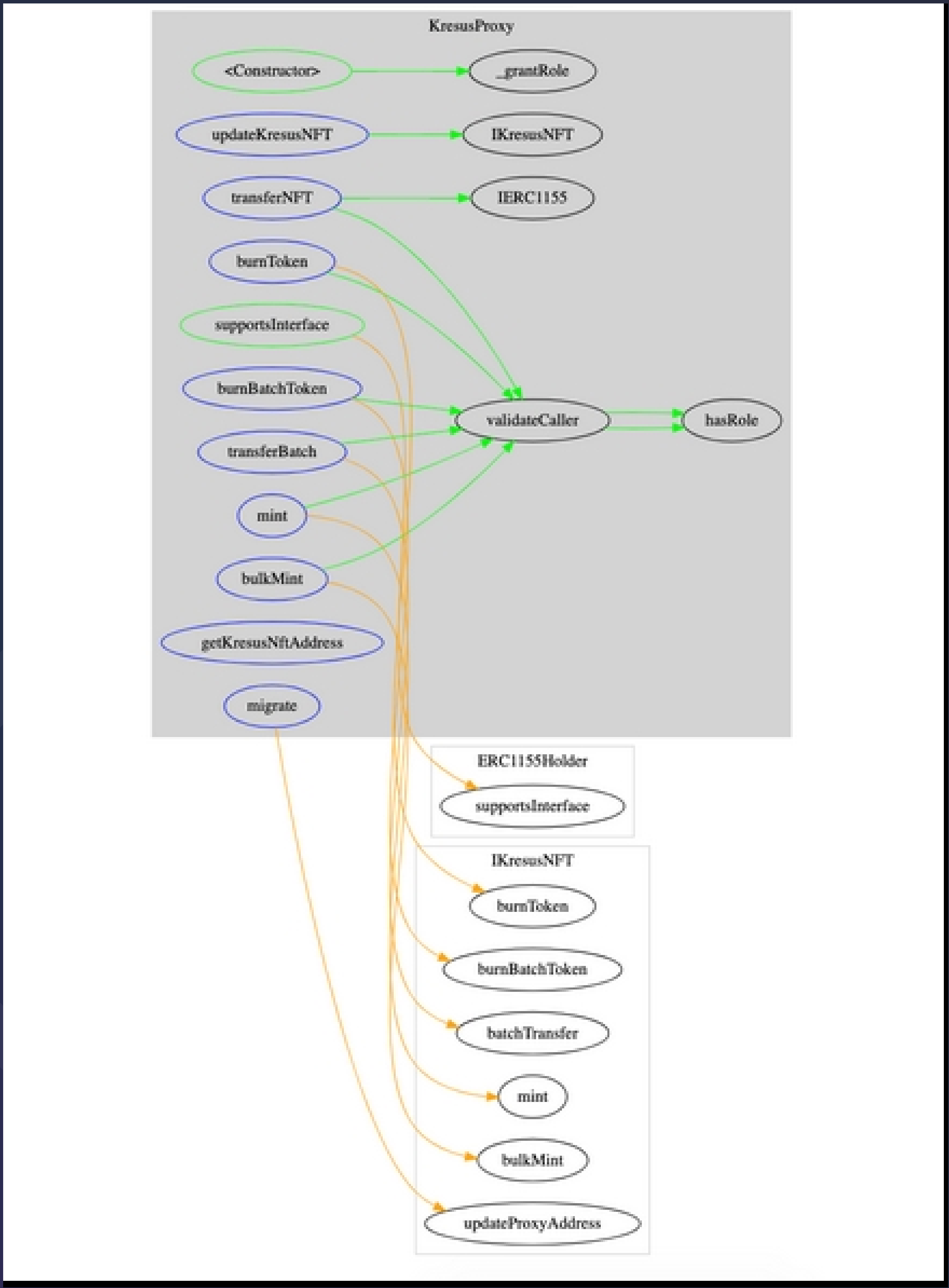
Dependency / Import Path	Count
@openzeppelin [ERC1155URIStorage.sol]	1
@openzeppelin [ERC1155Holder.sol]	1
@openzeppelin [IERC1155.sol]	1
@openzeppelin [AccessControl.sol]	1

Call Graph

contracts/KresusNFT.sol



contracts/KresusProxy.sol



Types of Severities

High

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.

Medium

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.

Low

Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Types of Issues

Open
Security vulnerabilities identified that must be resolved and are currently unresolved.

Resolved
These are the issues identified in the initial audit and have been successfully fixed.

Acknowledged
Vulnerabilities which have been acknowledged but are yet to be resolved.

Partially Resolved
Considerable efforts have been invested to reduce the risk/impact of the security issue, but are not completely resolved.

Checked Vulnerabilities

- ✓ Re-entrancy
- ✓ Timestamp Dependence
- ✓ Gas Limit and Loops
- ✓ Exception Disorder
- ✓ Gasless Send
- ✓ Use of tx.origin
- ✓ Compiler version not fixed
- ✓ Address hardcoded
- ✓ Divide before multiply
- ✓ Integer overflow/underflow
- ✓ Dangerous strict equalities
- ✓ Tautology or contradiction
- ✓ Return values of low-level calls
- ✓ Missing Zero Address Validation
- ✓ Private modifier
- ✓ Revert/require functions
- ✓ Using block.timestamp
- ✓ Multiple Sends
- ✓ Using SHA3
- ✓ Using suicide
- ✓ Using throw
- ✓ Using inline assembly

Techniques and Methods

Throughout the audit of smart contract, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

Structural Analysis

In this step, we have analysed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis

Static analysis of smart contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis

Manual analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analysed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption

In this step, we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit

Remix IDE, Truffle, Truffle Team, Solhint, Mythril, Slither, Solidity statistic analysis.

Manual Testing

High Severity Issues

1. Anyone can burn tokens

The functions burn() and burnBatchToken() in KresusNFT.sol are unprotected. KresusNFT.sol lines 148-171 this implies that anyone can call these functions directly on KresusNFT passing in the Proxy address, ids of assets held by proxy and burn the token(s) held by Proxy.

Recommendation

It is recommended to protect the burning of tokens by requiring that the caller is the Proxy if that's the intended case or create burner and or minter roles. Alternatively if intention is anyone holding the token can burn appropriate safeguards must be placed e.g require original sender _msgSender() is the _from

Status

Acknowledged

Medium Severity Issues

2. Centralization Risks / Overpowered Ownership

It is not clear in documentation if the Proxy contract will make use of decentralised control such as MultiSig. Admins can perform privileged activities. The admin has centralised control to mint, burn, migrate and update Kresus NFT contract. If the admin roles are compromised this can have an impact on working of the projects.

Recommendation

We advise the client to carefully manage the Admin's private key to avoid any potential risks of being hacked, this also applies to partners who will be sub-admins also managing their accesses well. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets

Status

Acknowledged

Low Severity Issues

3. Input array lengths not all checked for length equality

KresusNFT.sol line 115 states the following
- Lengths of `_assetIds`, `_tokenURIs` and `_amounts` must be equal.

The only check made is
require(_amounts.length == _tokenURIs.length, "Kresus NFT: Inconsistent lengths"); however _assetIds are not checked against the other input array sizes.

Recommendation

It is recommended to add an additional check to ensure all 3 arrays have equal size e.g require(_assetIds.length == _tokenURIs.length, "Kresus NFT: Inconsistent lengths")

Status

Acknowledged

4. Missing address verification/Zero address checks

Certain functions lack a safety check in the address, the address-type argument should include a zero-address test, otherwise, the contract's functionality may become inaccessible, tokens sent to invalid addresses, tokens may be burned in perpetuity or code may behave unexpectedly.

KresusNFT.sol line 59 constructor(address _proxyAddr)ERC1155("")
KresusNFT.sol line 208-213 function updateProxyAddress(address _proxyAddr,...
KresusProxy.sol line 30-32 constructor(address _defaultAdmin)
KresusProxy.sol line 264 function updateKresusNFT(address _address) external

Recommendation

It is recommended to check addresses are not zero addresses using for example require(_address != address(0), “error string”).

Status

Acknowledged

Informational Issues

5. Unnecessary use of override

Functions in KresusNFT.sol mint(), bulkMint(),burnToken(), burnBatchToken(), batchTransfer(), updateProxyAddress() make use of the override keyword. This is not necessary as contract functions are not overriding any functionality from inherited contracts.

Recommendation

It is recommended to remove the override keyword as this is used to change functionality of inherited contracts or implement its specification.

Status

Acknowledged

6. Missing events

The missing event makes it difficult to track off-chain and impact off-chain monitoring and incident response functionality. An event should be emitted for significant functionality change, admin related changes to ensure transparency and tracking.

KresusProxy.sol line 264-266 function updateKresusNFT(address _address....
KresusProxy.sol line 206-221 function migrate(address _to....
KresusNFT.sol line 208-220 function updateProxyAddress(address _proxyAddr,....

Recommendation

Critical updates may need to emit events. We recommend emitting an event to log the update of the above functionality, functions and admin related changes.

Status

Acknowledged

Functional Testing

Some of the tests performed are mentioned below:

contracts/interfaces/IKresusNFT.sol

✓	mint	function	PASS
✓	bulkMint	function	PASS
✓	burnToken	function	PASS
✓	burnBatchToken	function	PASS
✓	batchTransfer	function	PASS
✓	updateProxyAddress	function	PASS

contracts/KresusNFT.sol

✓	proxyAddress	variable	PASS
✓	mintedAssetsIds	mapping	PASS
✓	mintSuccessful	event	PASS
✓	bulkMintSuccessful	event	PASS
✓	batchTransferSuccessful	event	PASS
✓	constructor	constructor	PASS
✓	supportsInterface	function	PASS
✓	mint	function	PASS
✓	bulkMint	function	PASS
✓	burnToken	function	PASS?
✓	burnBatchToken	function	PASS?
✓	batchTransfer	function	PASS
✓	updateProxyAddress	function	PASS
✓	getProxyAddress	function	PASS
✓	isMinted	function	PASS

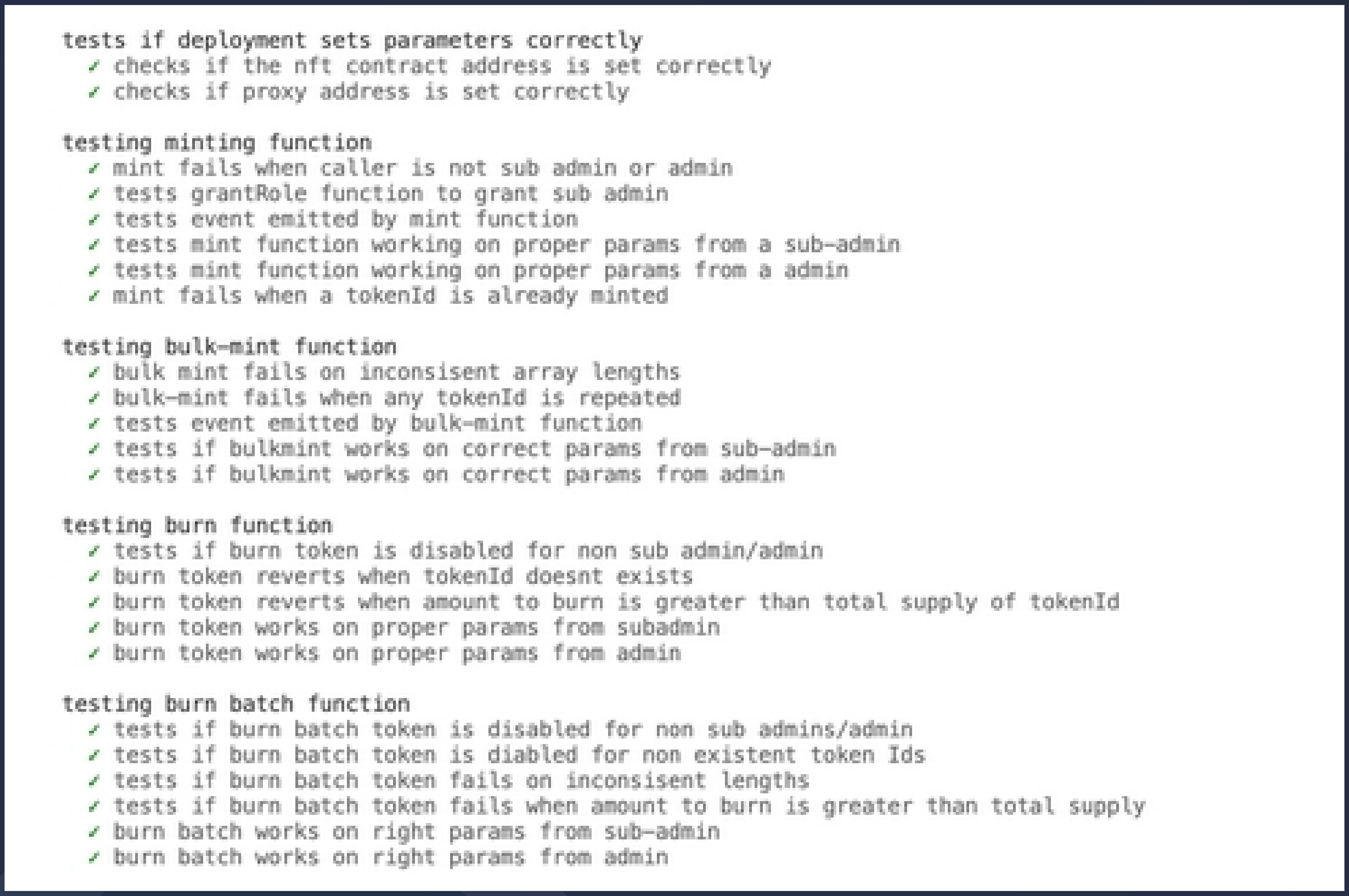
contracts/KresusProxy.sol

✓	SUB_ADMINS	variable	PASS
✓	KRESUS_NFT_CONTRACT	variable	PASS
✓	constructor	constructor	PASS
✓	supportsInterface	function	PASS
✓	mint	function	PASS
✓	bulkMint	function	PASS
✓	burnToken	function	PASS
✓	burnBatchToken	function	PASS
✓	transferNFT	function	PASS
✓	migrate	function	PASS
✓	transferBatch	function	PASS
✓	getKresusNftAddress	function	PASS
✓	updateKresusNFT	function	PASS
✓	validateCaller	function	PASS

Automated Tests

No major issues were found. Some false positive errors were reported by the tools. All the other issues have been categorized above according to their level of severity.

Test cases checked and all running and passing.



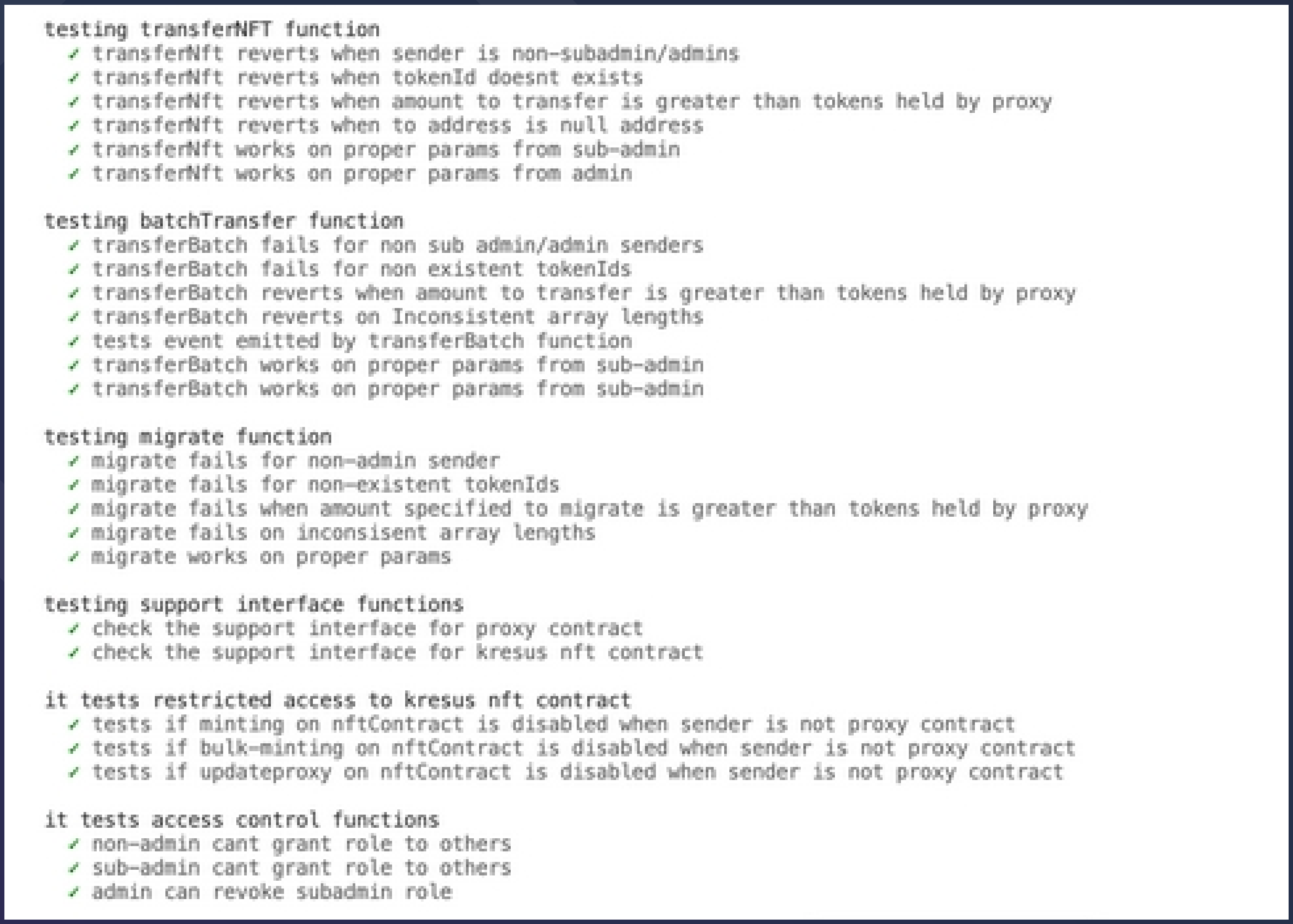
```
  tests if deployment sets parameters correctly
    ✓ checks if the nft contract address is set correctly
    ✓ checks if proxy address is set correctly

  testing minting function
    ✓ mint fails when caller is not sub admin or admin
    ✓ tests grantRole function to grant sub admin
    ✓ tests event emitted by mint function
    ✓ tests mint function working on proper params from a sub-admin
    ✓ tests mint function working on proper params from a admin
    ✓ mint fails when a tokenId is already minted

  testing bulk-mint function
    ✓ bulk mint fails on inconisent array lengths
    ✓ bulk-mint fails when any tokenId is repeated
    ✓ tests event emitted by bulk-mint function
    ✓ tests if bulkmint works on correct params from sub-admin
    ✓ tests if bulkmint works on correct params from admin

  testing burn function
    ✓ tests if burn token is disabled for non sub admin/admin
    ✓ burn token reverts when tokenId doesnt exists
    ✓ burn token reverts when amount to burn is greater than total supply of tokenId
    ✓ burn token works on proper params from subadmin
    ✓ burn token works on proper params from admin

  testing burn batch function
    ✓ tests if burn batch token is disabled for non sub admins/admin
    ✓ tests if burn batch token is diabled for non existent token Ids
    ✓ tests if burn batch token fails on inconisent lengths
    ✓ tests if burn batch token fails when amount to burn is greater than total supply
    ✓ burn batch works on right params from sub-admin
    ✓ burn batch works on right params from admin
```



```
  testing transferNFT function
    ✓ transferNft reverts when sender is non-subadmin/admins
    ✓ transferNft reverts when tokenId doesnt exists
    ✓ transferNft reverts when amount to transfer is greater than tokens held by proxy
    ✓ transferNft reverts when to address is null address
    ✓ transferNft works on proper params from sub-admin
    ✓ transferNft works on proper params from admin

  testing batchTransfer function
    ✓ transferBatch fails for non sub admin/admin senders
    ✓ transferBatch fails for non existent tokenIds
    ✓ transferBatch reverts when amount to transfer is greater than tokens held by proxy
    ✓ transferBatch reverts on Inconsistent array lengths
    ✓ tests event emitted by transferBatch function
    ✓ transferBatch works on proper params from sub-admin
    ✓ transferBatch works on proper params from sub-admin

  testing migrate function
    ✓ migrate fails for non-admin sender
    ✓ migrate fails for non-existent tokenIds
    ✓ migrate fails when amount specified to migrate is greater than tokens held by proxy
    ✓ migrate fails on inconisent array lengths
    ✓ migrate works on proper params

  testing support interface functions
    ✓ check the support interface for proxy contract
    ✓ check the support interface for kresus nft contract

  it tests restricted access to kresus nft contract
    ✓ tests if minting on nftContract is disabled when sender is not proxy contract
    ✓ tests if bulk-minting on nftContract is disabled when sender is not proxy contract
    ✓ tests if updateproxy on nftContract is disabled when sender is not proxy contract

  it tests access control functions
    ✓ non-admin cant grant role to others
    ✓ sub-admin cant grant role to others
    ✓ admin can revoke subadmin role
```

Closing Summary

Some issues of High Severity, Medium Severity, Low Severity and Informational nature were found in this audit. Some suggestions and best practices are also provided in order to improve the code quality and security posture.

Kresus Team has Acknowledged all Issues.

Disclaimer

QuillAudits smart contract audit is not a security warranty, investment advice, or an endorsement of the Kresus contracts. This audit does not provide a security or correctness guarantee of the audited smart contracts.

The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the Kresus Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

About QuillAudits

QuillAudits is a secure smart contracts audit platform designed by QuillHash Technologies. We are a team of dedicated blockchain security experts and smart contract auditors determined to ensure that Smart Contract-based Web3 projects can avail the latest and best security solutions to operate in a trustworthy and risk-free ecosystem.

700+
Audits Completed

$16B
Secured

700K
Lines of Code Audited

Follow Our Journey

Audit Report
March, 2023

For

Kresus

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com