ERC-Cool

Smart Contract Audit Report

AUDIT SUMMARY

ERC-Cool utilizes an improved implementation of the ERC721 standard smart contract. The project team intends to weave automated, carbon removal into any NFT use case that inherits it in an effort to increase the fundamental value and deploying NFTs as an entirely new standard in collective climate action.

For this audit, we reviewed the project team's ERC721Cool contract at 0x035A74245EFe2BEf9a428883cA8267D8839dA06A on the Ethereum Mainnet.

AUDIT FINDINGS

> *No findings were identified.*
>
> *Date: November 23rd, 2022.*
>
> *Updated: December 14th, 2022 to reflect the project team's ERCCool contract.*

CONTRACT OVERVIEW

> - *This contract is intended to be used as a template for users to enhance to fit their own desired functionality.*
> - *In the contract's current state, any user can mint any number of NFTs by specifying a desired quantity.*

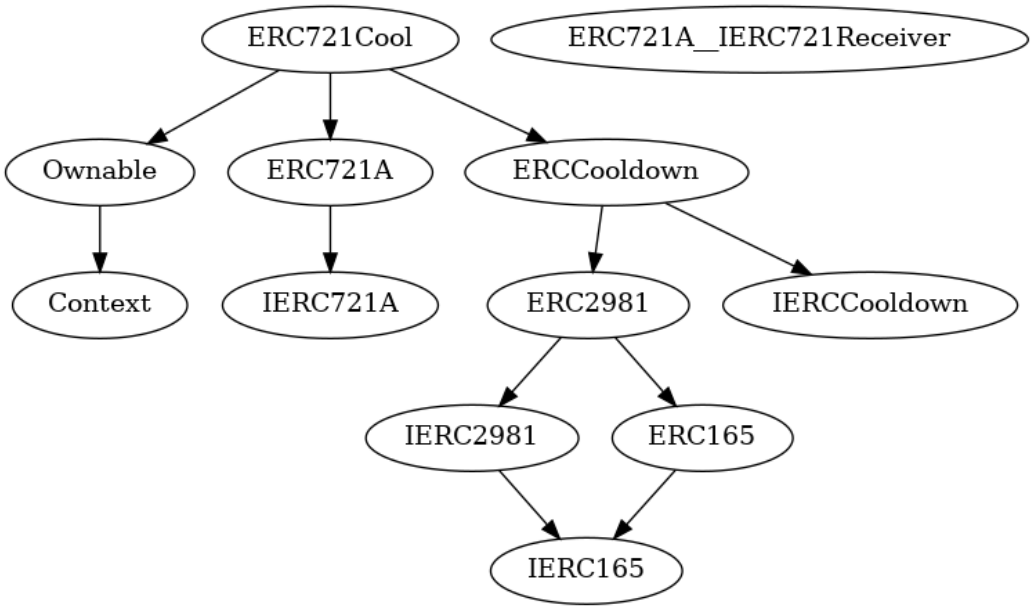
- *The minting functionality accepts ETH as payment though it is not required.*
- *A percentage of the supplied ETH is directly transferred to the TCL address based on the Mint Cool rate set by the team.*
- *A percentage of any ETH directly transferred to the contract is transferred to the TCL address based on the Transfer Cool Rate and Royalty rate set by the team.*
- *The contract implements ERC-2981 functionality that supports the use of Royalties that can be used with NFT marketplaces.*
- *As the contract is implemented with Solidity v0.8.x, it is safe from any possible overflows/underflows.*
- *The owner can withdraw all of the ETH from the contract at any time.*
- *The owner can set the Mint cool rate and Transfer cool rate to any percentages up to 100% at any time.*
- *The owner can set the Royalty rate to any value greater than the Transfer cool rate at any time.*

AUDIT RESULTS

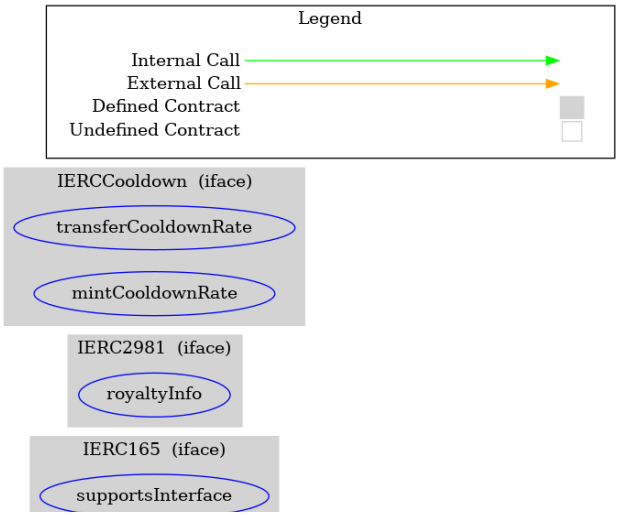
Vulnerability Category	Notes	Result
Arbitrary Jump/Storage Write	N/A	PASS
Centralization of Control	N/A	PASS
Compiler Issues	N/A	PASS
Delegate Call to Untrusted Contract	N/A	PASS
Dependence on Predictable Variables	N/A	PASS
Ether/Token Theft	N/A	PASS
Flash Loans	N/A	PASS

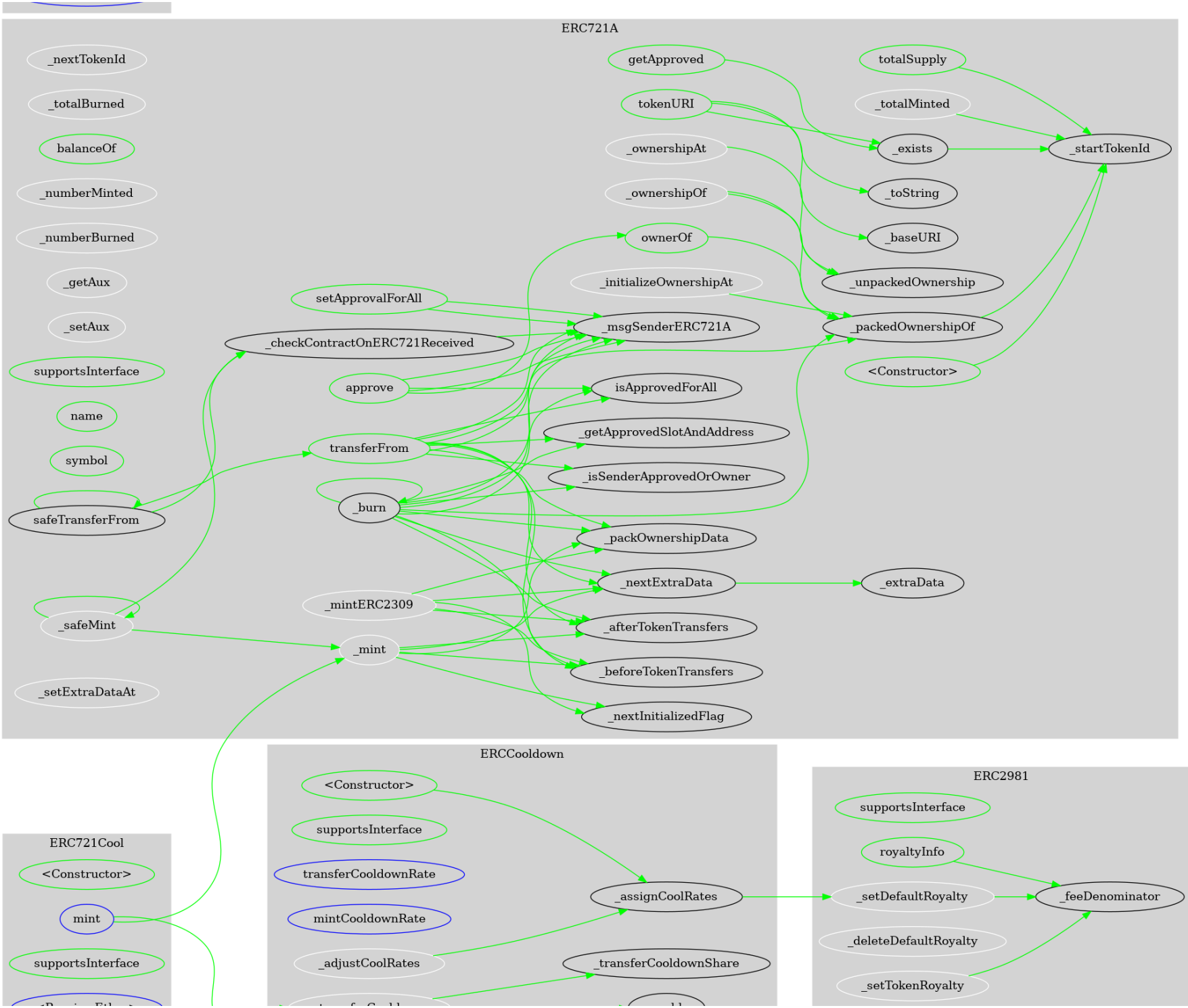
Vulnerability Category	Notes	Result
Front Running	N/A	PASS
Improper Events	N/A	PASS
Improper Authorization Scheme	N/A	PASS
Integer Over/Underflow	N/A	PASS
Logical Issues	N/A	PASS
Oracle Issues	N/A	PASS
Outdated Compiler Version	N/A	PASS
Race Conditions	N/A	PASS
Reentrancy	N/A	PASS
Signature Issues	N/A	PASS
Sybil Attack	N/A	PASS
Unbounded Loops	N/A	PASS
Unused Code	N/A	PASS
Overall Contract Safety		PASS

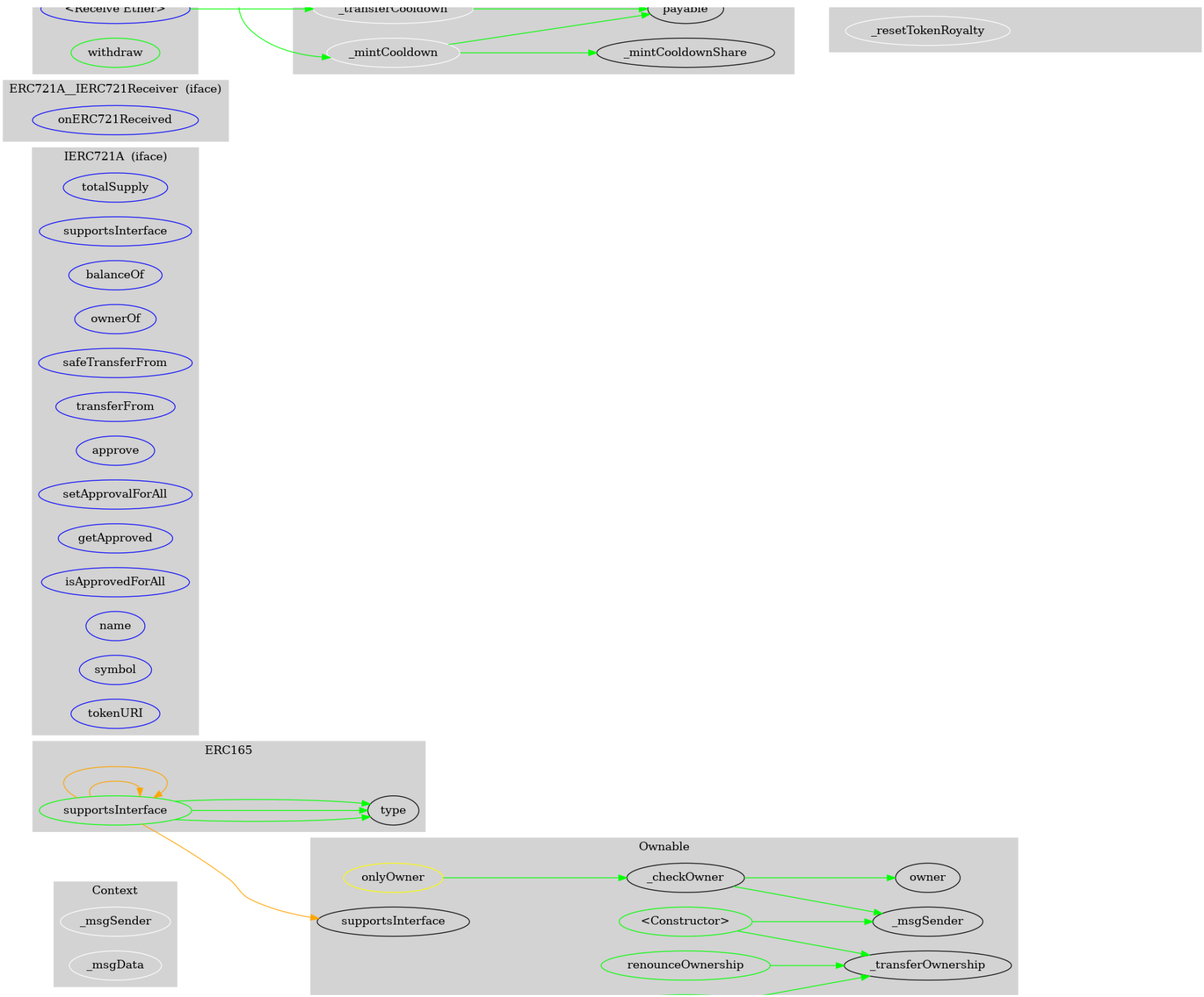
INHERITANCE CHART



FUNCTION GRAPH




ERC721A
_nextTokenId
_totalBurned
balanceOf
_numberMinted
_numberBurned
_getAux
_setAux
supportsInterface
name
symbol
safeTransferFrom
_safeMint
_setExtraDataAt
getApproved
tokenURI
_ownershipAt
_ownershipOf
ownerOf
_initializeOwnershipAt
setApprovalForAll
_checkContractOnERC721Received
approve
transferFrom
_burn
_mintERC2309
_mint
totalSupply
_totalMinted
_exists
_startTokenId
_toString
_baseURI
_unpackedOwnership
_packedOwnershipOf
<Constructor>
_msgSenderERC721A
isApprovedForAll
_getApprovedSlotAndAddress
_isSenderApprovedOrOwner
_packOwnershipData
_nextExtraData
_extraData
_afterTokenTransfers
_beforeTokenTransfers
_nextInitializedFlag
ERCCooldown
<Constructor>
supportsInterface
transferCooldownRate
mintCooldownRate
_adjustCoolRates
_assignCoolRates
_transferCooldownShare
ERC2981
supportsInterface
royaltyInfo
_setDefaultRoyalty
_deleteDefaultRoyalty
_setTokenRoyalty
_feeDenominator
ERC721Cool
<Constructor>
mint
supportsInterface


<Receive Ether>
withdraw
_transferCooldown
payable
_mintCooldown
_mintCooldownShare
_resetTokenRoyalty
ERC721A__IERC721Receiver (iface)
onERC721Received
IERC721A (iface)
totalSupply
supportsInterface
balanceOf
ownerOf
safeTransferFrom
transferFrom
approve
setApprovalForAll
getApproved
isApprovedForAll
name
symbol
tokenURI
ERC165
supportsInterface
type
Context
_msgSender
_msgData
Ownable
onlyOwner
_checkOwner
owner
supportsInterface
<Constructor>
_msgSender
renounceOwnership
_transferOwnership

FUNCTIONS OVERVIEW

```
($) = payable function
# = non-constant function

Int = Internal
Ext = External
Pub = Public

+  Context
   - [Int] _msgSender
   - [Int] _msgData

+  Ownable (Context)
   - [Pub]  #
   - [Pub] owner
   - [Int] _checkOwner
   - [Pub] renounceOwnership #
      - modifiers: onlyOwner
   - [Pub] transferOwnership #
      - modifiers: onlyOwner
   - [Int] _transferOwnership #

+ [Int] IERC721A
   - [Ext] totalSupply
   - [Ext] supportsInterface
   - [Ext] balanceOf
   - [Ext] ownerOf
   - [Ext] safeTransferFrom ($)
```

```
   - [Ext] safeTransferFrom ($)
   - [Ext] transferFrom ($)
   - [Ext] approve ($)
   - [Ext] setApprovalForAll #
   - [Ext] getApproved
   - [Ext] isApprovedForAll
   - [Ext] name
   - [Ext] symbol
   - [Ext] tokenURI

+ [Int] ERC721A__IERC721Receiver
   - [Ext] onERC721Received #

+  ERC721A (IERC721A)
   - [Pub]  #
   - [Int] _startTokenId
   - [Int] _nextTokenId
   - [Pub] totalSupply
   - [Int] _totalMinted
   - [Int] _totalBurned
   - [Pub] balanceOf
   - [Int] _numberMinted
   - [Int] _numberBurned
   - [Int] _getAux
   - [Int] _setAux #
   - [Pub] supportsInterface
   - [Pub] name
   - [Pub] symbol
   - [Pub] tokenURI
   - [Int] _baseURI
   - [Pub] ownerOf
   - [Int] _ownershipOf
   - [Int] _ownershipAt
   - [Int] _initializeOwnershipAt #
   - [Prv] _packedOwnershipOf
```

```
   - [Prv] _unpackedOwnership
   - [Prv] _packOwnershipData
   - [Prv] _nextInitializedFlag
   - [Pub] approve ($)
   - [Pub] getApproved
   - [Pub] setApprovalForAll #
   - [Pub] isApprovedForAll
   - [Int] _exists
   - [Prv] _isSenderApprovedOrOwner
   - [Prv] _getApprovedSlotAndAddress
   - [Pub] transferFrom ($)
   - [Pub] safeTransferFrom ($)
   - [Pub] safeTransferFrom ($)
   - [Int] _beforeTokenTransfers #
   - [Int] _afterTokenTransfers #
   - [Prv] _checkContractOnERC721Received #
   - [Int] _mint #
   - [Int] _mintERC2309 #
   - [Int] _safeMint #
   - [Int] _safeMint #
   - [Int] _burn #
   - [Int] _burn #
   - [Int] _setExtraDataAt #
   - [Int] _extraData
   - [Prv] _nextExtraData
   - [Int] _msgSenderERC721A
   - [Int] _toString

+ [Int] IERC165
   - [Ext] supportsInterface

+ [Int] IERC2981 (IERC165)
   - [Ext] royaltyInfo

+  ERC165 (IERC165)
```

```
   - [Pub] supportsInterface

+  ERC2981 (IERC2981, ERC165)
   - [Pub] supportsInterface
   - [Pub] royaltyInfo
   - [Int] _feeDenominator
   - [Int] _setDefaultRoyalty #
   - [Int] _deleteDefaultRoyalty #
   - [Int] _setTokenRoyalty #
   - [Int] _resetTokenRoyalty #

+ [Int] IERCCooldown
   - [Ext] transferCooldownRate
   - [Ext] mintCooldownRate

+  ERCCooldown (IERCCooldown, ERC2981)
   - [Pub]  #
   - [Pub] supportsInterface
   - [Ext] transferCooldownRate
   - [Ext] mintCooldownRate
   - [Int] _adjustCoolRates #
   - [Prv] _assignCoolRates #
   - [Prv] _transferCooldownShare
   - [Prv] _mintCooldownShare
   - [Int] _transferCooldown #
   - [Int] _mintCooldown #

+  ERC721Cool (ERCCooldown, ERC721A, Ownable)
   - [Pub]  #
      - modifiers: ERC721A,ERCCooldown
   - [Ext] mint ($)
   - [Pub] supportsInterface
   - [Ext]  ($)
   - [Pub] withdraw #
      - modifiers: onlyOwner
```

ABOUT SOLIDITY FINANCE

Solidity Finance was founded in 2020 and quickly grew to have one of the most experienced and well-equipped smart contract auditing teams in the industry. Our team has conducted 1500+ solidity smart contract audits covering all major project types and protocols, securing a total of over $50 billion U.S. dollars in on-chain value!
Our firm is well-reputed in the community and is trusted as a top smart contract auditing company for the review of solidity code, no matter how complex. Our team of experienced solidity smart contract auditors performs audits for tokens, NFTs, crowdsales, marketplaces, gambling games, financial protocols, and more!

Contact us today to get a free quote for a smart contract audit of your project!

WHAT IS A SOLIDITY AUDIT?

Typically, a smart contract audit is a comprehensive review process designed to discover logical errors, security vulnerabilities, and optimization opportunities within code. A *Solidity Audit* takes this a step further by verifying economic logic to ensure the stability of smart contracts and highlighting privileged functionality to create a report that is easy to understand for developers and community members alike.

HOW DO I INTERPRET THE FINDINGS?

Each of our Findings will be labeled with a Severity level. We always recommend the team resolve High, Medium, and Low severity findings prior to deploying the code to the mainnet. Here is a breakdown on what each Severity level means for the project:

- **High** severity indicates that the issue puts a large number of users' funds at risk and has a high probability of exploitation, or the smart contract contains serious logical issues which can prevent the code from operating as intended.
- **Medium** severity issues are those which place at least some users' funds at risk and has a medium to high probability of exploitation.
- **Low** severity issues have a relatively minor risk association; these issues have a low probability of occurring or may have a minimal impact.
- **Informational** issues pose no immediate risk, but inform the project team of opportunities for gas optimizations and following smart contract security best practices.