Smart Contract

Security Assessment

For HomelessFriends

09 July 2022

Ascendant

Ascendant

@ascendantproj

www.ascendant.finance.

Ascendant

Table of Contents

DISCLAIMER

Ascendant Finance ("Ascendant") has conducted an independent audit to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the codes that were provided for the scope of this audit. This audit report does not constitute agreement, acceptance or advocation for the Project that was audited, and users relying on this audit report should not consider this as having any merit for financial advice in any shape, form or nature. The contracts audited do not account for any economic developments that may be pursued by the Project in question, and that the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are completely free of exploits, bugs, vulnerabilities or deprecation of technologies. Further, this audit report shall not be disclosed nor transmitted to any persons or parties on any objective, goal or justification without due written assent, acquiescence or approval by Ascendant.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any persons reading this report should invest their funds without sufficient individual due diligence regardless of the findings presented in this report. Information is provided 'as is', and Ascendant is under no covenant to the completeness, accuracy or solidity of the contracts audited. In no event will Ascendant or its partners, employees, agents or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions and/or actions with regards to the information provided in this audit report.

Cryptocurrencies and any technologies by extension directly or indirectly related to cryptocurrencies are highly volatile and speculative by nature. All reasonable due diligence and safeguards may yet be insufficient, and users should exercise considerable caution when participating in any shape or form in this nascent industry.

The audit report has made all reasonable attempts to provide clear and articulate recommendations to the Project team with respect to the rectification, amendment and/or revision of any highlighted issues, vulnerabilities or exploits within the contracts provided. It is the sole responsibility of the Project team to sufficiently test and perform checks, ensuring that the contracts are functioning as intended, specifically that the functions therein contained within said contracts have the desired intended effects, functionalities and outcomes of the Project team.

Ascendant retains full rights over all intellectual property (including expertise and new attack or exploit vectors) discovered during the audit process. Ascendant is therefore allowed and expected to re-use this knowledge in subsequent audits and to inform existing projects that may have similar vulnerabilities. Ascendant may, at its discretion, claim bug bounties from third-parties while doing so.

Executive Summary

Severity	Found
🔴 High	0
🟠 Medium	1
🟡 Low	26
🟣 Informational	42
Total	69

We performed an independent technical audit to identify Smart Contracts uncertainties. This shall protect the code from illegitimate authorization attempts or external & internal threats of any type. This also ensures end-to-end proofing of the contract from frauds. The audit was performed semi-manually. We analyzed the Smart Contracts code line-by-line and used an automation tool to report any suspicious code.

The following tools were used:

- Truffle
- Remix IDE
- Slither

Overview

This report has been prepared for HomelessFriends on the Ethereum network. Ascendant provides a user-centered examination of the smart contracts to look for vulnerabilities, logic errors or other issues from both an internal and external perspective.

Summary

Project Name	HomelessFriends
Platform	Ethereum
Language	Solidity

Contracts Assessed

Name	**Location**
HomelessFriends.sol	Not deployed
Address.sol	In HomelessFriends.sol Contract
Base64.sol	In HomelessFriends.sol Contract
Context.sol	In HomelessFriends.sol Contract
ERC165.sol	In HomelessFriends.sol Contract
ERC721A.sol	In HomelessFriends.sol Contract
IERC165.sol	In HomelessFriends.sol Contract

IERC721.sol	In HomelessFriends.sol Contract
IERC721Enumerable.sol	In HomelessFriends.sol Contract
IERC721Metadata.sol	In HomelessFriends.sol Contract
IERC721Receiver.sol	In HomelessFriends.sol Contract
Ownable.sol	In HomelessFriends.sol Contract
ReentrancyGuard.sol	In HomelessFriends.sol Contract
Strings.sol	In HomelessFriends.sol Contract

Findings Summary

Severity	Found
🔴 High	0
🟠 Medium	1
🟡 Low	26
🟣 Informational	42
Total	69

Classification of Issues

🔴 High	Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, or impairment of the contract and its functions. Issues under this classification are recommended to be fixed with utmost urgency.
🟠 Medium	Bugs or issues that may be subject to exploit, though their impact is somewhat limited. Issues under this classification are recommended to be fixed as soon as possible.
🟡 Low	Effects are minimal in isolation and do not pose a significant danger to the project or its users. Issues under this classification are recommended to be fixed nonetheless.
🟣 Informational	Consistency, syntax or style best practices, Generally pose a negligible level of risk, if any.

Manual Review

Ascendant

Issues Checking Status

Issue Description	Checking Status
Compiler errors	PASS
Race conditions and Reentrancy. Cross-function race conditions.	PASS
Possible delays in data delivery.	PASS
Oracle calls.	PASS
Front running.	PASS
Timestamp dependence.	PASS
Integer Overflow and Underflow.	PASS
DoS with Revert.	PASS
DoS with block gas limit.	PASS
Methods execution permissions.	PASS
Economy model of the contract.	PASS
The impact of the exchange rate on the logic.	PASS
Private user data leaks.	PASS
Malicious Event log.	PASS
Scoping and Declarations.	PASS
Uninitialized storage pointers.	PASS

Arithmetic accuracy.	PASS
Design Logic.	PASS
Cross-function race conditions.	PASS
Safe Open Zeppelin contracts implementation and usage.	PASS
Fallback function security.	PASS

Audit Findings

Severity	Medium
Contract	HFriends.sol
Description	string baseURI declared private but is visible in the constructor

Code Snippet	654: constructor()
	ERC721A("HomelessFriends","HomelessFriends", MaxMintPerBatch_, TotalCollectionSize_) {
	...
	}

Recommendation	When the baseURI is explicitly written in the contract, it is visible to anyone who knows where to look. This means that even before the reveal function is called, any person can read the baseURI in the constructor, which in the best case scenario, makes the reveal pointless, but in the worse case scenario, a bad actor can use the hash to locate your stored images, download them, and sell them on their own without paying for them.
	
	Consider adding an argument in the constructor that takes the uri and sets it without having to publish it as part of the contract. Like so:
	
	constructor(string memory _uri) {
	setBaseURI(_uri)
	}

Severity	**Low**
Contract	HFriends.sol
Description	Use of hardcoded values instead of variables
Code Snippet	Throughout contract
Recommendation	Where possible, use variables that can be viewed and set instead of using values. Once the contract is deployed, these values cannot be changed, but variables can be. **Example:** require(totalSupply() < 3000, *Should be replaced with* require(totalSupply() < presaleLimit,

Severity	**Low**
Contract	HFriends.sol
Description	**Optimization:** Use of multiple mappings to store whitelisted users.
Code Snippet	Throughout Contract
Recommendation	Acknowledge. Storing multiple addresses on-chain will be very gas-costly, to call, traverse, and add new addresses. Consider using a merkletree.

Functional Test Status

Function Name	Type/Return Type	Score
TotalCollectionSize	private	PASS
MaxMintPerBatch	private	PASS
whitelistedAddressesForFreeMint	private	PASS
whitelistedAddressesForPreSale	private	PASS
_baseTokenURI	private	PASS
_uriBeforeRevel	private	PASS
MAX_PER_Transtion	read/public	PASS
PRICE	read/public	PASS
_revelNFT	read/public	PASS
status	read/public	PASS
_baseURI	internal	PASS
mint	payable/external	PASS
setURIbeforeRevel	write/external	PASS
setBaseURI	write/external	PASS
getOwnershipData	read/external	PASS
changeRevelStatus	write/external	PASS
changeMintPrice	write/external	PASS
changeMax_PER_Transtion	write/external	PASS

withdrawMoney	write/external	PASS
callerIsUser	modifier/public	PASS
tokenURI	read/public	PASS
isWhitelistedForFreeMint	read/public	PASS
addNewWhitelistUserForFreeMint	write/public	PASS
isWhitelistedForPreSale	read/public	PASS
addNewWhitelistUserForPreSale	write/public	PASS
numberMinted	read/public	PASS
reserve	write/public	PASS
getStatus	read/public	PASS

Automated Review

Ascendant

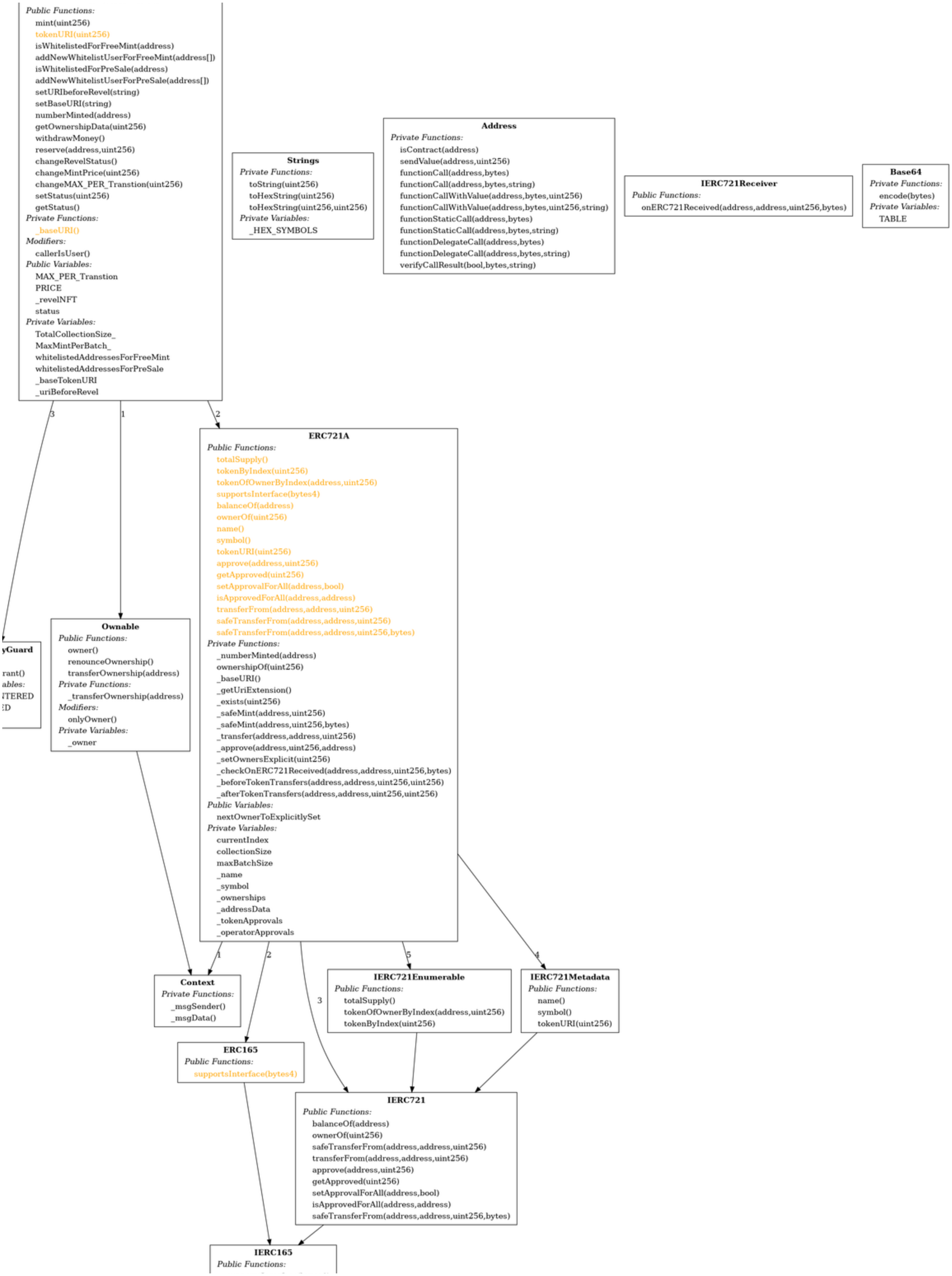
Solidity Static Analysis

Issue	Severity
Check-effects-interaction: **NOTE:** *All flags for checks-effects-interactions have been downgraded from MEDIUM to LOW due to the utilization of Reentrancy Guard.* Potential violation of Checks-Effects-Interaction pattern in Address.functionCallWithValue(address,bytes,uint256,string): Could potentially lead to re-entrancy vulnerability.uld potentially lead to re-entrancy vulnerability. Pos: 146	Low
For loop over dynamic array: Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful. Pos. 718	Informational

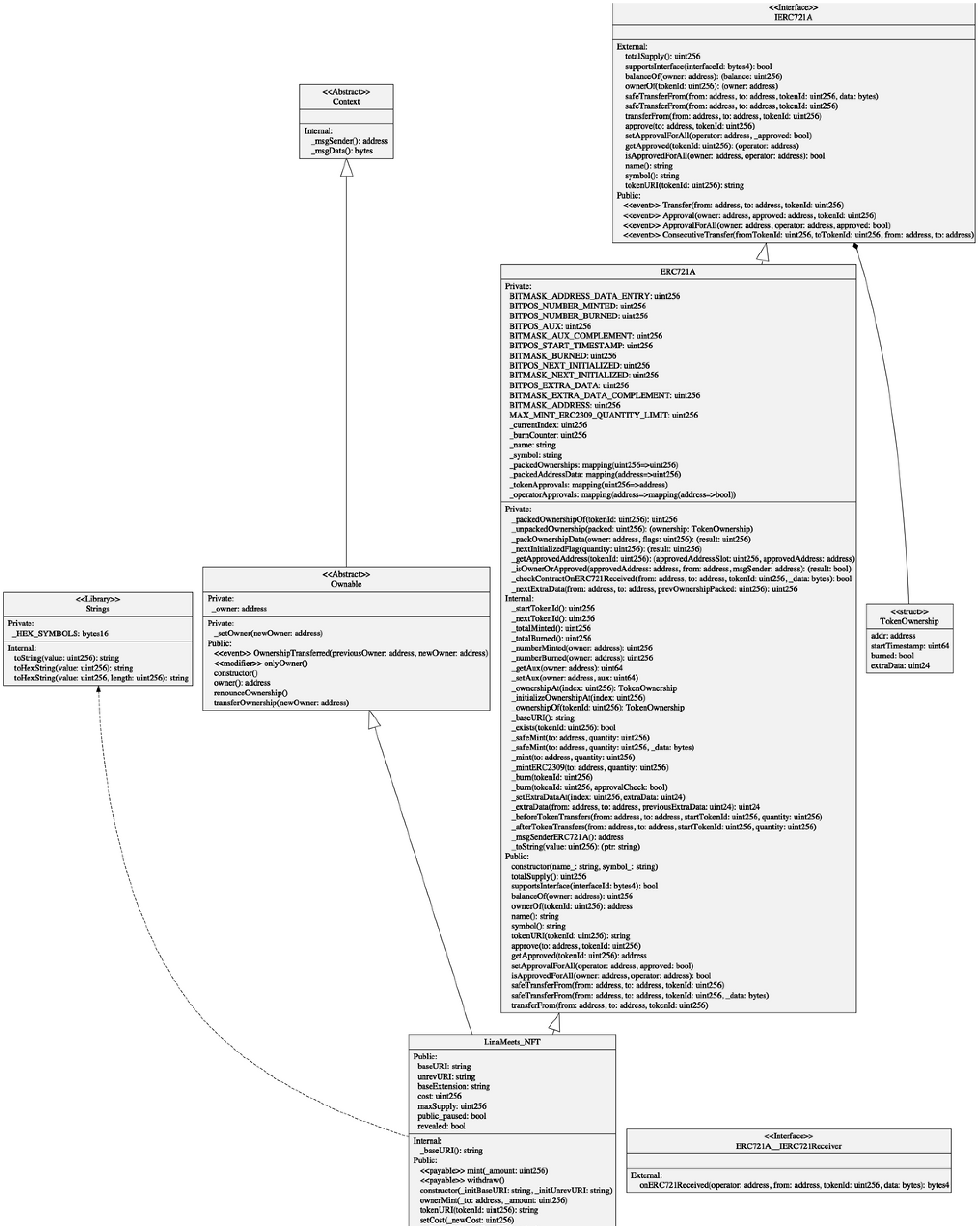
For loop over dynamic array: Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful. Pos. 718	Informational

Note: *Any issues from the automated test that have been determined by our team as not significant have been omitted from the final report.*

Inheritance Graph


Public Functions:
mint(uint256)
tokenURI(uint256)
isWhitelistedForFreeMint(address)
addNewWhitelistUserForFreeMint(address[])
isWhitelistedForPreSale(address)
addNewWhitelistUserForPreSale(address[])
setURIbeforeRevel(string)
setBaseURI(string)
numberMinted(address)
getOwnershipData(uint256)
withdrawMoney()
reserve(address,uint256)
changeRevelStatus()
changeMintPrice(uint256)
changeMAX_PER_Transtion(uint256)
setStatus(uint256)
getStatus()
Private Functions:
_baseURI()
Modifiers:
callerIsUser()
Public Variables:
MAX_PER_Transtion
PRICE
_revelNFT
status
Private Variables:
TotalCollectionSize_
MaxMintPerBatch_
whitelistedAddressesForFreeMint
whitelistedAddressesForPreSale
_baseTokenURI
_uriBeforeRevel
Strings
Private Functions:
toString(uint256)
toHexString(uint256)
toHexString(uint256,uint256)
Private Variables:
_HEX_SYMBOLS
Address
Private Functions:
isContract(address)
sendValue(address,uint256)
functionCall(address,bytes)
functionCall(address,bytes,string)
functionCallWithValue(address,bytes,uint256)
functionCallWithValue(address,bytes,uint256,string)
functionStaticCall(address,bytes)
functionStaticCall(address,bytes,string)
functionDelegateCall(address,bytes)
functionDelegateCall(address,bytes,string)
verifyCallResult(bool,bytes,string)
IERC721Receiver
Public Functions:
onERC721Received(address,address,uint256,bytes)
Base64
Private Functions:
encode(bytes)
Private Variables:
TABLE
3
1
2
ERC721A
Public Functions:
totalSupply()
tokenByIndex(uint256)
tokenOfOwnerByIndex(address,uint256)
supportsInterface(bytes4)
balanceOf(address)
ownerOf(uint256)
name()
symbol()
tokenURI(uint256)
approve(address,uint256)
getApproved(uint256)
setApprovalForAll(address,bool)
isApprovedForAll(address,address)
transferFrom(address,address,uint256)
safeTransferFrom(address,address,uint256)
safeTransferFrom(address,address,uint256,bytes)
Private Functions:
_numberMinted(address)
ownershipOf(uint256)
_baseURI()
_getUriExtension()
_exists(uint256)
_safeMint(address,uint256)
_safeMint(address,uint256,bytes)
_transfer(address,address,uint256)
_approve(address,uint256,address)
_setOwnersExplicit(uint256)
_checkOnERC721Received(address,address,uint256,bytes)
_beforeTokenTransfers(address,address,uint256,uint256)
_afterTokenTransfers(address,address,uint256,uint256)
Public Variables:
nextOwnerToExplicitlySet
Private Variables:
currentIndex
collectionSize
maxBatchSize
_name
_symbol
_ownerships
_addressData
_tokenApprovals
_operatorApprovals
yGuard
rant()
ables:
ITERED
ED
Ownable
Public Functions:
owner()
renounceOwnership()
transferOwnership(address)
Private Functions:
_transferOwnership(address)
Modifiers:
onlyOwner()
Private Variables:
_owner
1
2
3
5
4
Context
Private Functions:
_msgSender()
_msgData()
ERC165
Public Functions:
supportsInterface(bytes4)
IERC721Enumerable
Public Functions:
totalSupply()
tokenOfOwnerByIndex(address,uint256)
tokenByIndex(uint256)
IERC721Metadata
Public Functions:
name()
symbol()
tokenURI(uint256)
IERC721
Public Functions:
balanceOf(address)
ownerOf(uint256)
safeTransferFrom(address,address,uint256)
transferFrom(address,address,uint256)
approve(address,uint256)
getApproved(uint256)
setApprovalForAll(address,bool)
isApprovedForAll(address,address)
safeTransferFrom(address,address,uint256,bytes)
IERC165
Public Functions:

Unified Modeling Language(UML)



<<Interface>>
IERC721A
External:
totalSupply(): uint256
supportsInterface(interfaceId: bytes4): bool
balanceOf(owner: address): (balance: uint256)
ownerOf(tokenId: uint256): (owner: address)
safeTransferFrom(from: address, to: address, tokenId: uint256, data: bytes)
safeTransferFrom(from: address, to: address, tokenId: uint256)
transferFrom(from: address, to: address, tokenId: uint256)
approve(to: address, tokenId: uint256)
setApprovalForAll(operator: address, _approved: bool)
getApproved(tokenId: uint256): (operator: address)
isApprovedForAll(owner: address, operator: address): bool
name(): string
symbol(): string
tokenURI(tokenId: uint256): string
Public:
<<event>> Transfer(from: address, to: address, tokenId: uint256)
<<event>> Approval(owner: address, approved: address, tokenId: uint256)
<<event>> ApprovalForAll(owner: address, operator: address, approved: bool)
<<event>> ConsecutiveTransfer(fromTokenId: uint256, toTokenId: uint256, from: address, to: address)
<<Abstract>>
Context
Internal:
_msgSender(): address
_msgData(): bytes
ERC721A
Private:
BITMASK_ADDRESS_DATA_ENTRY: uint256
BITPOS_NUMBER_MINTED: uint256
BITPOS_NUMBER_BURNED: uint256
BITPOS_AUX: uint256
BITMASK_AUX_COMPLEMENT: uint256
BITPOS_START_TIMESTAMP: uint256
BITMASK_BURNED: uint256
BITPOS_NEXT_INITIALIZED: uint256
BITMASK_NEXT_INITIALIZED: uint256
BITPOS_EXTRA_DATA: uint256
BITMASK_EXTRA_DATA_COMPLEMENT: uint256
BITMASK_ADDRESS: uint256
MAX_MINT_ERC2309_QUANTITY_LIMIT: uint256
_currentIndex: uint256
_burnCounter: uint256
_name: string
_symbol: string
_packedOwnerships: mapping(uint256=>uint256)
_packedAddressData: mapping(address=>uint256)
_tokenApprovals: mapping(uint256=>address)
_operatorApprovals: mapping(address=>mapping(address=>bool))
Private:
_packedOwnershipOf(tokenId: uint256): uint256
_unpackedOwnership(packed: uint256): (ownership: TokenOwnership)
_packOwnershipData(owner: address, flags: uint256): (result: uint256)
_nextInitializedFlag(quantity: uint256): (result: uint256)
_getApprovedAddress(tokenId: uint256): (approvedAddressSlot: uint256, approvedAddress: address)
_isOwnerOrApproved(approvedAddress: address, from: address, msgSender: address): (result: bool)
_checkContractOnERC721Received(from: address, to: address, tokenId: uint256, _data: bytes): bool
_nextExtraData(from: address, to: address, prevOwnershipPacked: uint256): uint256
Internal:
_startTokenId(): uint256
_nextTokenId(): uint256
_totalMinted(): uint256
_totalBurned(): uint256
_numberMinted(owner: address): uint256
_numberBurned(owner: address): uint256
_getAux(owner: address): uint64
_setAux(owner: address, aux: uint64)
_ownershipAt(index: uint256): TokenOwnership
_initializeOwnershipAt(index: uint256)
_ownershipOf(tokenId: uint256): TokenOwnership
_baseURI(): string
_exists(tokenId: uint256): bool
_safeMint(to: address, quantity: uint256)
_safeMint(to: address, quantity: uint256, _data: bytes)
_mint(to: address, quantity: uint256)
_mintERC2309(to: address, quantity: uint256)
_burn(tokenId: uint256)
_burn(tokenId: uint256, approvalCheck: bool)
_setExtraDataAt(index: uint256, extraData: uint24)
_extraData(from: address, to: address, previousExtraData: uint24): uint24
_beforeTokenTransfers(from: address, to: address, startTokenId: uint256, quantity: uint256)
_afterTokenTransfers(from: address, to: address, startTokenId: uint256, quantity: uint256)
_msgSenderERC721A(): address
_toString(value: uint256): (ptr: string)
Public:
constructor(name_: string, symbol_: string)
totalSupply(): uint256
supportsInterface(interfaceId: bytes4): bool
balanceOf(owner: address): uint256
ownerOf(tokenId: uint256): address
name(): string
symbol(): string
tokenURI(tokenId: uint256): string
approve(to: address, tokenId: uint256)
getApproved(tokenId: uint256): address
setApprovalForAll(operator: address, approved: bool)
isApprovedForAll(owner: address, operator: address): bool
safeTransferFrom(from: address, to: address, tokenId: uint256)
safeTransferFrom(from: address, to: address, tokenId: uint256, _data: bytes)
transferFrom(from: address, to: address, tokenId: uint256)
<<Library>>
Strings
Private:
_HEX_SYMBOLS: bytes16
Internal:
toString(value: uint256): string
toHexString(value: uint256): string
toHexString(value: uint256, length: uint256): string
<<Abstract>>
Ownable
Private:
_owner: address
Private:
_setOwner(newOwner: address)
Public:
<<event>> OwnershipTransferred(previousOwner: address, newOwner: address)
<<modifier>> onlyOwner()
constructor()
owner(): address
renounceOwnership()
transferOwnership(newOwner: address)
<<struct>>
TokenOwnership
addr: address
startTimestamp: uint64
burned: bool
extraData: uint24
LinaMeets_NFT
Public:
baseURI: string
unrevURI: string
baseExtension: string
cost: uint256
maxSupply: uint256
public_paused: bool
revealed: bool
Internal:
_baseURI(): string
Public:
<<payable>> mint(_amount: uint256)
<<payable>> withdraw()
constructor(_initBaseURI: string, _initUnrevURI: string)
ownerMint(_to: address, _amount: uint256)
tokenURI(tokenId: uint256): string
setCost(_newCost: uint256)
<<Interface>>
ERC721A__IERC721Receiver
External:
onERC721Received(operator: address, from: address, tokenId: uint256, data: bytes): bytes4

Function ID Report

```
IERC721Receiver:
+-------------------------------------------------+------------+
|                       Name                      |     ID     |
+-------------------------------------------------+------------+
| onERC721Received(address,address,uint256,bytes) | 0x150b7a02 |
+-------------------------------------------------+------------+

HomelessFriends:
+-------------------------------------------------+------------+
|                       Name                      |     ID     |
+-------------------------------------------------+------------+
|    constructor(string,string,uint256,uint256)   | 0x0135f5cc |
|                  totalSupply()                  | 0x18160ddd |
|               tokenByIndex(uint256)             | 0x4f6ccce7 |
|     tokenOfOwnerByIndex(address,uint256)        | 0x2f745c59 |
|            supportsInterface(bytes4)            | 0x01ffc9a7 |
|                balanceOf(address)               | 0x70a08231 |
|                 ownerOf(uint256)                | 0x6352211e |
|                      name()                     | 0x06fdde03 |
|                     symbol()                    | 0x95d89b41 |
|                tokenURI(uint256)                | 0xc87b56dd |
|             approve(address,uint256)            | 0x095ea7b3 |
|              getApproved(uint256)               | 0x081812fc |
|         setApprovalForAll(address,bool)         | 0xa22cb465 |
|        isApprovedForAll(address,address)        | 0xe985e9c5 |
|      transferFrom(address,address,uint256)      | 0x23b872dd |
|    safeTransferFrom(address,address,uint256)    | 0x42842e0e |
| safeTransferFrom(address,address,uint256,bytes) | 0xb88d4fde |
|                  totalSupply()                  | 0x18160ddd |
|     tokenOfOwnerByIndex(address,uint256)        | 0x2f745c59 |
|               tokenByIndex(uint256)             | 0x4f6ccce7 |
|                balanceOf(address)               | 0x70a08231 |
|                 ownerOf(uint256)                | 0x6352211e |
|    safeTransferFrom(address,address,uint256)    | 0x42842e0e |
|      transferFrom(address,address,uint256)      | 0x23b872dd |
|             approve(address,uint256)            | 0x095ea7b3 |
|              getApproved(uint256)               | 0x081812fc |
|         setApprovalForAll(address,bool)         | 0xa22cb465 |
|        isApprovedForAll(address,address)        | 0xe985e9c5 |
| safeTransferFrom(address,address,uint256,bytes) | 0xb88d4fde |
|            supportsInterface(bytes4)            | 0x01ffc9a7 |
|                      name()                     | 0x06fdde03 |
|                    ----+
 symbol()                    | 0x95d89b41 |
|              tokenURI(uint256)              | 0xc87b56dd |
|          supportsInterface(bytes4)          | 0x01ffc9a7 |
|                  owner()                    | 0x8da5cb5b |
|            renounceOwnership()              | 0x715018a6 |
|          transferOwnership(address)         | 0xf2fde38b |
|               constructor()                 | 0x90fa17bb |
|               mint(uint256)                 | 0xa0712d68 |
|              tokenURI(uint256)              | 0xc87b56dd |
|      isWhitelistedForFreeMint(address)      | 0xcca3f458 |
|   addNewWhitelistUserForFreeMint(address[])  | 0x180e548e |
|       isWhitelistedForPreSale(address)      | 0xd7c701ed |
|    addNewWhitelistUserForPreSale(address[])  | 0x5404259a |
|          setURIbeforeRevel(string)          | 0x5c37809d |
|             setBaseURI(string)              | 0x55f804b3 |
|           numberMinted(address)             | 0xdc33e681 |
|          getOwnershipData(uint256)          | 0x9231ab2a |
|              withdrawMoney()                | 0xac446002 |
|          reserve(address,uint256)           | 0xcc47a40b |
|            changeRevelStatus()              | 0xbd0a8439 |
|          changeMintPrice(uint256)           | 0x3fd17366 |
|       changeMAX_PER_Transtion(uint256)      | 0xaf7b26e9 |
|             setStatus(uint256)              | 0x69ba1a75 |
|                getStatus()                  | 0x4e69d560 |
|         nextOwnerToExplicitlySet()          | 0xd7224ba0 |
|            MAX_PER_Transtion()              | 0xd04950a1 |
|                  PRICE()                    | 0x8d859f3e |
|                _revelNFT()                  | 0x62c6f7b9 |
|                 status()                    | 0x200d2ed2 |
+----------------------------------------------+--------
```

Conclusion

The smart contracts reviewed in this audit contain no High severity issues and all Medium to Low issues have either been corrected or acknowledged.

Please check the disclaimer above and note, the audit makes no statements or warranties on business model, investment attractiveness or code sustainability. The report is provided for the only contract mentioned in the report and does not include any other potential contracts deployed by Owner.

Ascendant

Ascendant

@ascendantproj

www.ascendant.finance.com

Ascendant