Smart Contract

Security Assessment

For Cartoonistic
Bidding
12 Jan 2023

Ascendant

Ascendant
@ascendantproj
www.ascendant.finance

Ascendant

Table of Contents

3	Disclaimer
4	Executive Summary
5	Overview
6	Findings Summary & Legend
8	Manual Review • Issue Checking Status • Audit Findings • Functional Test Status • Omitted Results
26	Automated Review • Unified Model Language
28	Conclusion

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	1
Medium	3
Low	11
Informational	30
Total	45

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 Cartoonistic Bidding 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	Cartoonistic Bidding
Platform	Ethereum
Language	Solidity

Contracts Assessed

Name	Location
CartoonisticBidding.sol	Goerli: 0x82F51186D5DBc2C6BAB7A1bB538699E2E44c009D
ERC721.sol	In Cartoonistic Bidding Contract
Context.sol	In Cartoonistic Bidding Contract
Ownable.sol	In Cartoonistic Bidding Contract
ERC2981.sol	In Cartoonistic Bidding Contract
OperatorFilterer.sol	In Cartoonistic Bidding Contract

Name	Location
IERC721Metadata.sol	In Cartoonistic Bidding Contract
IERC165.sol	In Cartoonistic Bidding Contract
IERC721Enumerable.sol	In Cartoonistic Bidding Contract
IERC721.sol	In Cartoonistic Bidding Contract
IERC2981.sol	In Cartoonistic Bidding Contract
IERC721Receiver.sol	In Cartoonistic Bidding Contract
Address.sol	In Cartoonistic Bidding Contract
Strings.sol	In Cartoonistic Bidding Contract
DefaultOperatorFilterer.sol	In Cartoonistic Bidding Contract
IOperatorFilterRegistry.sol	In Cartoonistic Bidding Contract

Findings Summary

Severity	Found
High	1
Medium	3
Low	11
Informational	30
Total	45

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

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	**High**
Contract	CartoonisticBidding.sol
Description	Exposed baseURI
Code Snippet	string public baseURI = [REDACTED]
Recommendation	Writing the baseURI into the smart contract allows bad actors to obtain the location of the metadata and download all NFTs, even the ones that have not yet been minted. To prevent this, the baseURI variable should be set to private, and the constructor should accepted a string argument that will allow the owner to set the baseURI on deployment so the baseURI is visible to no one.
Status	

Audit Findings

Severity	**Medium**
Contract	CartoonisticBidding.sol
Description	Checks-Effects-Interactions

Code Snippet	function mint(
	...
	
	_safeMint(msg.sender, middlePieceID);
	totalSupply++;

Recommendation	The mint functions currently all fail the checks-effects-interactions pattern, which requires that interactions (such as _safeMint) should come after state changes (updating the mapping), which exposes the function to Reentrancy attacks. totalSupply++; should either precede _safeMint or a ReentrancyGuard should be added to these functions.
Status	

Audit Findings

Severity	**Medium**
Contract	CartoonisticBidding.sol
Description	Reentrancy Vulnerability

Code Snippet	function bid(
	...
	
	if(depositedAlready[_id9_by_seller]
	[msg.sender]) {
	(bool takeback,) =
	payable(msg.sender).call{value:
	depositedAmount[_id9_by_seller]
	[msg.sender]}("");
	...
	
	depositedAlready[_id9_by_seller]
	[msg.sender] = true;

Recommendation	Function updates state after possible transfer, exposing the function to Reentrancy Attack. ReentrancyGuard should be added to this function.
Status	

Audit Findings

Severity	Medium
Contract	CartoonisticBidding.sol
Description	Hardcoded Addresses
Code Snippet	1764: address _nft_Contract = [REDACTED]
Recommendation	Hardcoding addresses should be avoided where possible to avoid copy mistakes/deployment errors. It is much easier to simply add the variables these hardcoded addresses represent as arguments in the constructor.
Status	

Audit Findings

Severity	Low
Contract	CartoonisticBidding.sol
Description	Redundant Function
Code Snippet	1933: function setbiddingClosingTime(...
Recommendation	setbiddingClosingTime is identical to setbiddingClosingTimeByOwner; consider removing one of these funcitons to optimize contract
Status	

Audit Findings

Severity	**Low**
Contract	CartoonisticBidding.sol
Description	Storage Cost
Code Snippet	1922: function setNickNameWithWallet(string memory _nickname){ ...
Recommendation	storing a nickname as a string will be expensive for users. Storing it in a mapping causes the function to be even more expensive as the size of the storage increases. Consider converting the string to bytes and storing it this way
Status	

Audit Findings

Severity	Informational(Multiple)
Contract	CartoonisticBidding.sol
Description	Public functions that are used externally and not by the contract itself should be marked external
Code Snippet	N/A
Recommendation	Public functions generally consume more gas than external functions. Any functions that are not used internally should be marked external.
Status	

Functional Test Status

Function Name	Type/Return Type	Score
IOperatorFilterRegistry		
codeHashOf	external	PASS
copyEntriesOf	external	PASS
filteredCodeHashAt	external	PASS
filteredCodeHashes	external	PASS
filteredOperatorAt	external	PASS
filteredOperators	external	PASS
isCodeHashFiltered	external	PASS
isCodeHashOfFiltered	external	PASS
isOperatorAllowed	external	PASS
isOperatorFiltered	external	PASS
isRegistered	external	PASS
register	external	PASS
registerAndCopyEntries	external	PASS
registerAndSubscribe	external	PASS
subscribe	external	PASS
subscriberAt	external	PASS
subscribers	external	PASS

Function Name	Type/Return Type	Score
subscriptionOf	external	PASS
unregister	external	PASS
unsubscribe	external	PASS
updateCodeHash	external	PASS
updateCodeHashes	external	PASS
updateOperator	external	PASS
updateOperators	external	PASS
OperatorFilterer		
_checkFilterOperator	internal	PASS
Counters		
current	internal	PASS
decrement	internal	PASS
increment	internal	PASS
reset	internal	PASS
Context		
_msgData	internal	PASS
_msgSender	internal	PASS

Function Name	Type/Return Type	Score
Ownable		
_checkOwner	internal	PASS
transferOwnership	public	PASS
owner	public	PASS
renounceOwnership	public	PASS
Address		
functionCall	internal	PASS
functionCallWithValue	internal	PASS
functionDelegateCall	internal	PASS
functionStaticCall	internal	PASS
isContract	internal	PASS
sendValue	internal	PASS
verifyCallResult	internal	PASS
IERC721Receiver		
onERC721Received	external	PASS
ERC2981		

_deleteDefaultRoyalty	internal	PASS
_feeDenominator	internal	PASS
_resetTokenRoyalty	internal	PASS
_setDefaultRoyalty	internal	PASS
royaltyInfo	public	PASS
IERC721Metadata		
approve	external	PASS
balanceOf	external	PASS
getApproved	external	PASS
isApprovedForAll	external	PASS
ownerOf	external	PASS
safeTransferFrom	external	PASS
setApprovalForAll	external	PASS
transferFrom	external	PASS
IERC721		
approve	external	PASS
balanceOf	external	PASS
getApproved	external	PASS
isApprovedForAll	external	PASS
ownerOf	external	PASS
safeTransferFrom	external	PASS

setApprovalForAll	write/external	PASS
transferFrom	write/external	PASS
ERC721		
_afterTokenTransfers	internal	PASS
_approve	private	PASS
_baseURI	internal	PASS
_beforeTokenTransfers	internal	PASS
_burn	internal	PASS
_checkOnERC721Received	private	PASS
_exists	internal	PASS
_mint	internal	PASS
_numberBurned	internal	PASS
_numberMinted	internal	PASS
_safeMint	internal	PASS
_transfer	private	PASS
approve	public	PASS
balanceOf	public	PASS
getApproved	public	PASS
isApprovedForAll	public	PASS
name	public	PASS
ownerOf	public	PASS
ownershipOf	internal	PASS
safeTransferFrom	public	PASS

setApprovalForAll	public	PASS
symbol	public	PASS
supportsInterface	public	PASS
tokenByIndex	public	PASS
tokenOfOwnerByIndex	public	PASS
tokenURI	public	PASS
totalSupply	public	PASS
transferFrom	public	PASS
Cartoonistic Bidding		
approve	public	PASS
PuzzleInput	public	PASS
mint	public	PASS
safeTransferFrom	public	PASS
setApprovalForAll	public	PASS
setBaseURI	public	PASS
setContractURI	public	PASS
calculateRoyalty	public	PASS
setNFTContract	public	PASS
name	public	PASS
ownerOf	public	PASS
symbol	public	PASS
deleteData	public	PASS
royaltyInfo	public	PASS

setBidGap	public	PASS
isApprovedForAll	public	PASS
getApproved	public	PASS
balanceOf	public	PASS
setBaseExtension	public	PASS
setMAX_Supply	public	PASS
setMax_per_wallet	public	PASS
setNotRevealedURI	public	PASS
setPublicSaleCost	public	PASS
setRoyaltyInfo	public	PASS
setRoyaltyAddress	public	PASS
toggleReveal	public	PASS
setAdditionalBidValue	public	PASS
tokenURI	public	PASS
transferFrom	public	PASS
withdraw	public	PASS
bid	public	PASS
setPublicMint_mint_status	public	PASS
setPause	public	PASS
setInitialBiddingTimeGap	public	PASS
setNickNameWithWallet	public	PASS
setBiddingClosingTime	public	PASS
setBiddingClosingTimeByOwner	public	PASS

Omitted Results

Note: Any issues that have been omitted from this report have been deemed by the reviewing team as irrelevant, inapplicable, and/or negligible to the proper functioning of this contract. Thus, any omitted issues can be safely ignored.

Automated Review

Ascendant