

Smart Contract

Security Assessment

For Sukuta

29 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

29 Automated Review

- Unified Model Language

31 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 advocacy 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	2
● Medium	2
● Low	8
● Informational	34
Total	46

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 Sukuta 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	Sukuta
Platform	Ethereum
Language	Solidity

Contracts Assessed





Name	Location
Sukuta.sol	Not published
ERC721A.sol	In Sukuta contract
Context.sol	In Sukuta contract
Ownable.sol	In Sukuta contract
Ownable2Step.sol	In Sukuta contract
OperatorFilterer.sol	In Sukuta contract

Name	Location
MerkleProof.sol	In Sukuta contract
IERC721Metadata.sol	In Sukuta contract
IERC165.sol	In Sukuta contract
IERC721Enumerable.sol	In Sukuta contract
IERC721.sol	In Sukuta contract
OwnedRegistrant.sol	In Sukuta contract
IERC721Receiver.sol	In Sukuta contract
Address.sol	In Sukuta contract
Strings.sol	In Sukuta contract
DefaultOperatorFilterer.sol	In Sukuta contract
IOperatorFilterRegistry.sol	In Sukuta contract
EnumerableSet.sol	In Sukuta contract
OperatorFilterRegistryErrorsandEvents.sol	In Sukuta contract
SafeMath.sol	In Sukuta contract
Math.sol	In Sukuta contract

Findings Summary

Severity	Found
 High	2
 Medium	2
 Low	8
 Informational	34
Total	46

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	High
Contract	Sukuta.sol
Description	Exposed baseTokenURI
Code Snippet	<pre>string public baseTokenURI;</pre>
Recommendation	Marking the baseTokenURI public 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, so when owner sets baseTokenURI after deployment the baseTokenURI is visible to no one.
Status	

Audit Findings

Severity	High
Contract	Sukuta.sol
Description	Hardcoded Authorized wallet with no set function.
Code Snippet	<pre>address public Authorized = [REDACTED] modifier onlyAuthorized() { require(msg.sender == owner() msg.sender == Authorized, "Not authorized"); _; }</pre>
Recommendation	<p>The owner will not be able to revoke/change authorized wallet after deployment. This means all functions containing the modifier 'onlyAuthorized' will be able to be called by the owner and one other wallet.</p>
Status	

Audit Findings

Severity	Medium
Contract	Sukuta.sol
Description	No set functions for withdrawal wallets; wallets cannot be changed. Owner has separate withdraw function.
Code Snippet	address private wallet1 = [REDACTED] address private wallet2 = [REDACTED] address private wallet3 = [REDACTED]
Recommendation	There are no set functions to allow the owner to change wallet1, wallet2, and wallet3. If the reason for this is to establish trust, knowing that once deployed, these values cannot be changed, that is understandable, but the owner also has a withdraw function that would enable the owner to bypass this and withdraw all funds to himself anyway.
Status	

Audit Findings

Severity	Medium
Contract	Sukuta.sol
Description	Max supply can be changed, but collection size is immutable.
Code Snippet	<pre>function setMaxSupply(uint256 _mxSupply) public onlyAuthorized { MAX_SUPPLY = _mxSupply; }</pre>
Recommendation	Once a collection is committed to IPFS with a certain number of tokenIDs, this collection cannot be changed under the same address. Furthermore, if a new baseTokenURI is set anytime during or after the initial mint, the new IPFS hash will not change what is already in a holder's wallet. If the total collection size is currently not known, it is recommended to only use this function to set the max supply once the total collection size becomes known and then never called again.
Status	

Audit Findings

Severity	Low
Contract	Sukuta.sol
Description	Hardcoded variables marked private.
Code Snippet	address private wallet1 = [REDACTED] address private wallet2 = [REDACTED] address private wallet3 = [REDACTED]
Recommendation	Do not hardcode variables that are marked private. Because these contracts are verified on etherscan, anyone can read these variables by simply going to the contract code.
Status	

Audit Findings

Severity	Low
Contract	Sukuta.sol
Description	Checks-Effects-Interactions
Code Snippet	<pre>function giveAway(uint256 numberOfTokens, address to) external onlyOwner { require(giveawayLimit.sub(numberOfT okens) >= 0, "Giveaways exhausted"); _safeMint(to, numberOfTokens); giveawayLimit = giveawayLimit.sub(numberOfTokens); }</pre>
Recommendation	<p>The giveAway function currently fails the checks-effects-interactions pattern, which requires that interactions (such as <code>_safeMint</code>) should come after state changes (updating the variable).</p> <pre>giveawayLimit = giveawayLimit.sub(numberOfTokens) should precede _safeMint.</pre>
Status	

Audit Findings

Severity	Low
Contract	Sukuta.sol
Description	Giveaway does not check if max supply has been exceeded
Code Snippet	<pre>function giveAway(uint256 numberOfTokens, address to) external onlyOwner { require(giveawayLimit.sub(numberOfT okens) >= 0, "Giveaways exhausted"); _safeMint(to, numberOfTokens); giveawayLimit = giveawayLimit.sub(numberOfTokens); }</pre>
Recommendation	<p>There is no require statement to check if the amount minted exceeds the max supply of tokens. As a result, the giveaway function will succeed, but potentially mint nonexistent tokens. Add a require statement to make sure the totalSupply + numberOfTokens is still less than the max supply.</p>
Status	

Audit Findings

Severity	Informational
Contract	Sukuta.sol
Description	whitelistMint and vipMint are functionally redundant.
Code Snippet	N/A
Recommendation	whitelistMint and vipMint are given the same mint limits, prices, etc. If this is intentional, disregard this recommendation. If not, in the interest of gas optimization, consider removing one of these sets.
Status	

Audit Findings

Severity	Informational(Multiple)
Contract	Sukuta.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
IERC20		
allowance	external	PASS
approve	external	PASS
balanceOf	external	PASS
totalSupply	external	PASS
transfer	external	PASS
transferFrom	external	PASS
Math		
average	internal	PASS

Function Name	Type/Return Type	Score
ceilDiv	internal	PASS
log10	internal	PASS
log2	internal	PASS
log256	internal	PASS
max	internal	PASS
min	internal	PASS
mulDiv	internal	PASS
sqrt	internal	PASS
EnumerableSet		
_add	private	PASS
_at	private	PASS
_contains	private	PASS
_length	private	PASS
_remove	private	PASS
_values	private	PASS
Ownable2Step		
owner	public	PASS
renounceOwnership	public	PASS

Function Name	Type/Return Type	Score
Context		
_msgData	internal	PASS
_msgSender	internal	PASS
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

OwnedRegistrant		
_transferOwnership	internal	PASS
acceptOwnership	public	PASS
pendingOwner	public	PASS
transferOwnership	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
ERC721A		
_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
MerkleProof		
verify	internal	PASS
Sukuta		
_baseURI	internal	PASS
_startTokenId	internal	PASS
flipSaleState	external	PASS
flipWhitelistSaleState	external	PASS
giveAway	external	PASS
mint	external	PASS
numberMinted	public	PASS
safeTransferFrom	public	PASS
setBaseURI	public	PASS
setMaxPurchaseLimit	public	PASS

setMaxSupply	public	PASS
setMaxTxLimit	public	PASS
setMaxTxVIP	public	PASS
setMaxTxWL	public	PASS
setMaxVIPPurchaseLimit	public	PASS
setMaxWLPurchaseLimit	public	PASS
setPrice	public	PASS
setPriceVIPWL	public	PASS
setVIPPriceWL	public	PASS
setVIPMaxSupply	public	PASS
setWLMaxSupply	public	PASS
tokenURI	public	PASS
transferFrom	public	PASS
updateMerkleRoot	external	PASS
updateVIPMerkleRoot	external	PASS
vipMint	external	PASS
whitelistMint	external	PASS
withdraw	public	PASS
withdrawAll	external	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

Conclusion

The smart contracts reviewed in this audit contain no critical 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



Ascendant