



ONYX LABS

Smart Contract Audit

Prepared For:
Avatizers

Date:
9/23/22



Table of Contents

Table of Contents	2
Summary	3
Overview	4
Project Summary	4
Audit Summary	4
Vulnerability Summary	4
Audit Scope	5
Findings	6
AVA-001 Restrict setMaxSupply	7
AVA-002 Missing Withdraw Function	8
AVA-003 Move to ERC721A	9
META-001 Restrict setSpecialImage	10
META-002 Add description to Metadata	11
Disclaimer	12
About	13
Initial Source Code	14




Summary

This report was prepared to summarize the findings of the audit performed for Avatizer NFT of their ERC-721 minting contract and associated on-chain metadata/SVG rendering contracts. The primary objective of the audit was to identify issues and vulnerabilities in the source code. This audit was performed utilizing Manual Review techniques and Static Analysis.

The audit consisted of:

- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Static Analysis utilizing Slither Static Analyzer.
- Unit testing specific functions.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Line-by-line manual review of the entire codebase by auditors.

The following report contains the results of the audit, and recommendations to guard against potential security threats and improve contract functionality.

A  next to a recommendation indicates that the recommendation was implemented prior to the completion of the audit.

All audit findings were resolved prior to the completion of this report.

We would like to thank the Avatizer team for their business and cooperation; their openness and communication made this audit a success.



Overview

Project Summary

Project Name	Avatizer
Description	Avatizer is an ERC721 NFT mint, consisting of a single WL minting phase. Minting is restricted using a merkle tree. All supply of tokens is mintable free of charge. Each token has unique on-chain dynamic art.
Blockchain	Ethereum
Language	Solidity
Codebase	AvatizersNFT.sol AvatizersMetadataManager.sol AvatizerSVGRenderer.sol
Commit	N/A

Audit Summary

Delivery Date	9/23/22
Audit Methodology	Static Analysis, Manual Review, Unit Testing

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Resolved
Critical	0	0	0	0	0
High	1	0	0	0	1
Medium	2	0	0	0	2
Low	0	0	0	0	0
Informational	2	0	0	0	2



Audit Scope

ID	File
AVA	AvatizersNFT.sol
META	AvatizersMetadataManager.sol
SVG	AvatizerSVGRenderer.sol



Findings

ID	Title	Category	Severity	Status
AVA-001	Restrict setMaxSupply	Security	Medium	
AVA-002	Missing Withdraw Function	Functional	Informational	
AVA-003	Move to ERC721A	Optimization	High	
META-001	Restrict setSpecialImage	Security	Medium	
META-002	Add Description to Metadata	Styling	Informational	



AVA-001 | Restrict setMaxSupply

Category	Severity	Location	Status
Security	Medium	N/A	Resolved

Description

The function `setMaxSupply` allows the contract owner unfettered increases of the maximum supply of tokens. This could lead to tokens beyond the intended/advertised maximum supply being minted.

Recommendation

✓ Include logic that requires `_maxSupply` to be less than `maxSupply`.

Action Taken

The `setMaxSupply` function was removed.



AVA-002 | Missing Withdraw Function

Category	Severity	Location	Status
Functional	Informational	N/A	Resolved

Description

The contract includes payable functions but does not have a method of retrieving ETH sent to the contract. The mint is intended to be free, but it would be a good practice to include a function to retrieve any incidentally deposited ETH.

Recommendation

✅ Insert a function that allows for the withdrawal of funds.

Action Taken

A function that allows for the rescue of funds was added.

Code:

```
function rescueFunds(address to) public onlyOwner {
    uint256 balance = address(this).balance;
    (bool callSuccess, ) = payable(to).call{value: balance}("");
    require(callSuccess, "Call failed");
}
```




AVA-003 | Move to ERC721A

Category	Severity	Location	Status
Optimization	High	N/A	Resolved

Description

The initial contract was built based on the standard OpenZeppelin ERC721 contract. At this time, it makes sense for most ERC721 minting contracts to inherit ERC721A, developed by Churi Labs. ERC721A offers many optimization advantages, including bulk minting with reduced gas usage.

Recommendation

✓ Replace the inheritance of OZ ERC721 with ERC721A and rework the necessary functions to fit.

Action Taken

OpenZeppelin 721 was replaced with ERC721A.

Reference: <https://chiru-labs.github.io/ERC721A/#/>



META-001 | Restrict setSpecialImage

Category	Severity	Location	Status
Security	Medium	Line 25	Resolved

Description

The function `setSpecialImage` allows the contract owner unrestricted access to alter the SVG of any token. This could lead to overriding of the current generative on-chain art for any and all tokens.

Recommendation

✅ Add logic to the function that restricts usage to only the necessary and intended uses (IE limit to use on one token).

Action Taken

The contract was modified such that `specialImage` is applicable only to `tokenId 0`.

Code:

```
function setSpecialImage(string memory svg) external onlyOwner {  
    specialImage = svg;  
}
```



META-002 | Add description to Metadata

Category	Severity	Location	Status
Styling	Informational	Line 60	Resolved

Description

The current tokenURI function in the Metadata Manager contract includes name, image, and attributes, but does not include a project description in the metadata. This is not required, but something that tends to be fairly standard across metadata sets.

Recommendation

✓ If desired, include a 'description' tag in the metadata structure.

Action Taken

Description was added to the tokenURI function.

Code:

```
function tokenURI(uint256 tokenId) public view returns (string memory) {
    string memory image;
    if (bytes(specialImages[tokenId]).length == 0) {
        image = generateImage(nftContract.getTokenGenes(tokenId));
    } else {
        image = specialImages[tokenId];
    }
    return string.concat(
        "data:application/json;base64,",
        Base64.encode(abi.encodePacked(
            '{"name":"Avatizers #', tokenId.toString(), '",',
            '"image":', image, '",',
            '"description":"Placeholder description",',
            '"attributes":[',
            '{"trait_type":"Type",',
            '"value":', (bytes(specialImages[tokenId]).length == 0)? '"Regular"' :
            '"Special"',
            '}]}'
        ))
    );
}
```



Disclaimer

The audit performed by Onyx Labs is intended to identify function weaknesses, potential security threats, and improve performance. It does not provide a guarantee of contract performance or the absence of any exploitable vulnerabilities. Any alterations to the source code provided could potentially invalidate the analysis provided in this audit report. Onyx Labs is not liable for any losses or damages incurred as a result of contract deployment and the proceeding occurrences.



About

Onyx Labs is a full-service web3 development firm founded in December 2021. Their scope of expertise ranges from smart contract development for a multitude of applications, to web development and audits. By offering best-in-class service, Onyx Labs has helped many successful projects enter the web3 space, or provided assistance with their ongoing development needs.