



Syndicate

ERC721M

SMART CONTRACT AUDIT

05.10.2023

Made in Germany by Softstack.io



Table of contents

1. Disclaimer.....	4
2. About the Project and Company	5
2.1 Project Overview.....	6
3. Vulnerability & Risk Level	7
4. Auditing Strategy and Techniques Applied.....	8
4.1 Methodology	8
5. Metrics	9
5.1 Tested Contract Files	9
5.2 CallGraph.....	11
5.3 Inheritance Graph.....	12
5.4 Source Lines & Risk.....	13
5.5 Capabilities	14
5.6 Dependencies / External imports.....	15
5.7 Source Unites in Scope	17
6. Scope of Work.....	19
6.1 Findings Overview	20
6.2 Manual and Automated Vulnerability Test.....	21
6.2.1 Unsafe Minting Function	21
6.2.2 Unsafe Transfer Function	22
6.2.3 Missing safeMint Function	23
6.2.4 Overpowered Admin Rights.....	24
6.2.5 Improper Input Validation.....	25



6.2.6 Improper Input Validation	26
6.2.7 Wrong Function Modifier	27
6.2.8 Repeated Existence Check	28
6.2.9 Duplicated Functions in Contract.....	29
6.2.10 Duplicated Paused Check	30
6.3 SWC Attacks	31
6.4 Verify Claims	35
6.5 Unit Tests	36
7. Executive Summary.....	47
8. About the Auditor	48



1. Disclaimer

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only.

The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of SYNDICATE INC. If you are not the intended receptor of this document, remember that any disclosure, copying or dissemination of it is forbidden.

Major Versions / Date	Description
0.1 (29.08.2023)	Layout
0.4 (01.09.2023)	Automated Security Testing Manual Security Testing
0.5 (04.09.2023)	Verify Claims and Test Deployment
0.6 (07.09.2023)	Testing SWC Checks
0.9 (08.09.2023)	Summary and Recommendation
1.0 (11.09.2023)	Final document
1.1 (05.10.2023)	Re-check a4fc81e23d8e2331a4f234601a6a664226f9cc14

2. About the Project and Company

Company address:

SYNDICATE INC.
1049 El Monte Avenue Ste C #560
Mountain View, CA 94040
USA



Website: <https://syndicate.io>

Twitter: <https://twitter.com/syndicateio>

Substack: <https://syndicateprotocol.substack.com/>

2.1 Project Overview

Syndicate is a revolutionary platform that sits at the crossroads of finance and web3, focusing on decentralizing and democratizing the investing landscape. By seamlessly integrating both financial protocols and social networking, Syndicate is poised to radically reshape the way capital and communities interact and collaborate. A cornerstone of this platform is "Collectives," an innovative web3-based tool for social networking and community building. This allows users to craft on-chain social networks and has already seen integration from notable communities like Rug Radio, BFF, Kamp Kilmer, and FWB Fellowships.

Integral to the platform's functionality is the new ERC-721M standard, an evolution of the recognized ERC-721 NFT standard. This protocol has been designed with features that allow for greater modularity, including actions such as minting, burning, transferring, and evolving metadata. One of its standout attributes is its gas-optimized design, ensuring efficiency in minting and distribution.

For those less familiar with coding, Syndicate presents a no-code tool that makes the process of setting up Collectives straightforward. This user-friendly feature ensures that customization, from member limits to transferability, is accessible to a wider audience. Importantly, it has been designed with interoperability in mind, ensuring seamless interaction with various web3 tools and platforms.

What sets Syndicate's approach to social networks apart is its emphasis on user ownership, decentralized design, interoperability, and adaptability. In a Syndicate network, the users truly own the platform, a departure from traditional centralized entities. Moreover, these networks can evolve and modify based on the specific needs and desires of the community, ensuring they remain relevant and responsive.

Looking ahead, Syndicate has ambitious plans to intertwine unique social constructs with financial mechanisms. Such integration can potentially allow founders to raise capital directly from their social networks, investors to efficiently syndicate deals, and communities to trade work in exchange for ownership stakes. Notably, the Syndicate Genesis Collective, an on-chain social network, is crafted for the broader Syndicate community, offering a unique space for collaboration. The Collective is available for a limited period and is set to introduce a plethora of dynamic features for its early members.

In essence, Syndicate is not just a platform; it is a vision of the future. A future where the boundaries between finance and community blur, creating a more inclusive, dynamic, and transparent financial ecosystem.

3. Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system. Risk Level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
Critical	9 – 10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
High	7 – 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon as possible.
Medium	4 – 6.9	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.	Implementation of corrective actions in a certain period.
Low	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

4. Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. To do so, reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as there were discovered.

4.1 Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
 - i. Review of the specifications, sources, and instructions provided to softstack to make sure we understand the size, scope, and functionality of the smart contract.
 - ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
 - iii. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to softstack describe.
2. Testing and automated analysis that includes the following:
 - i. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
 - ii. Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

5. Metrics

The metrics section should give the reader an overview on the size, quality, flows and capabilities of the codebase, without the knowledge to understand the actual code.

5.1 Tested Contract Files

The following are the MD5 hashes of the reviewed files. A file with a different MD5 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different MD5 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review

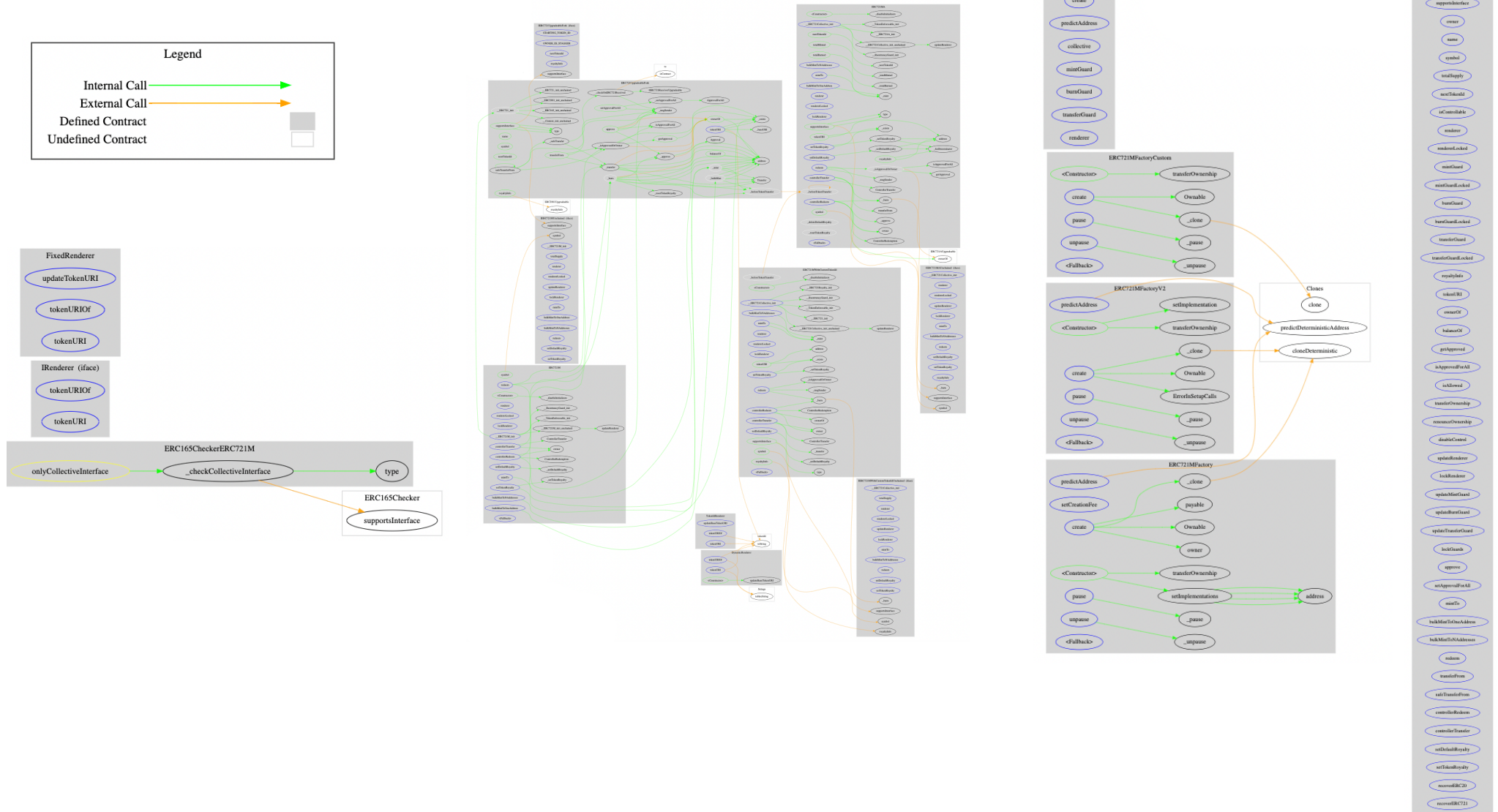
File	Fingerprint (MD5)
./contracts/ERC721M/renderer/TokenIdRenderer.sol	66ad1d372d0390aa1c84ef65ec7820e8
./contracts/ERC721M/renderer/FixedRenderer.sol	1cecd02ca9fb2738418c541a36dbc91
./contracts/ERC721M/renderer/IRenderer.sol	adc6465ad760e5c4397552998f7a6f13
./contracts/ERC721M/renderer/DynamicRenderer.sol	d58c2630cb0eeb438412db2ace42bea5
./contracts/ERC721M/IERC721M.sol	19b3ef09e5c8ad6b6732cbc778ae351f
./contracts/ERC721M/ERC165CheckerERC721M.sol	482085b24ca9b04fe97b1dcefa460228
./contracts/ERC721M/IERC721MFlat.sol	f2929ab7fb4fb1c23041533f99ec9e5f
./contracts/ERC721M/ERC721M.sol	bef4688a4917886b8016e191c696cf67
./contracts/ERC721M/ERC721UpgradeableFork.sol	05db2d0d69bce2be67c28dc09f4e75f7
./contracts/ERC721M/ERC721A/ERC721MA.sol	3cc5a81c2a03635456c2e946da2ccfd7
./contracts/ERC721M/ERC721A/IERC721MA.sol	8b1d18d888b2b81dd2bff37428681b48
./contracts/ERC721M/ERC721MWithCustomTokenId/ ERC721MWithCustomTokenId.sol	47737ebd44a7c9a161c75ef9692d8e88
./contracts/ERC721M/ERC721MWithCustomTokenId/ ERC721MWithCustomTokenId.sol	ef7e418597e56ca47f2577d58a3bb738
./contracts/ERC721M/IERC721UpgradeableFork.sol	a9b7df44f362f7c2905d146e1ce20a9c
./contracts/ERC721M/factory/ERC721MFactoryCusto m.sol	90d9038484c61a68271e19bb960a8b97

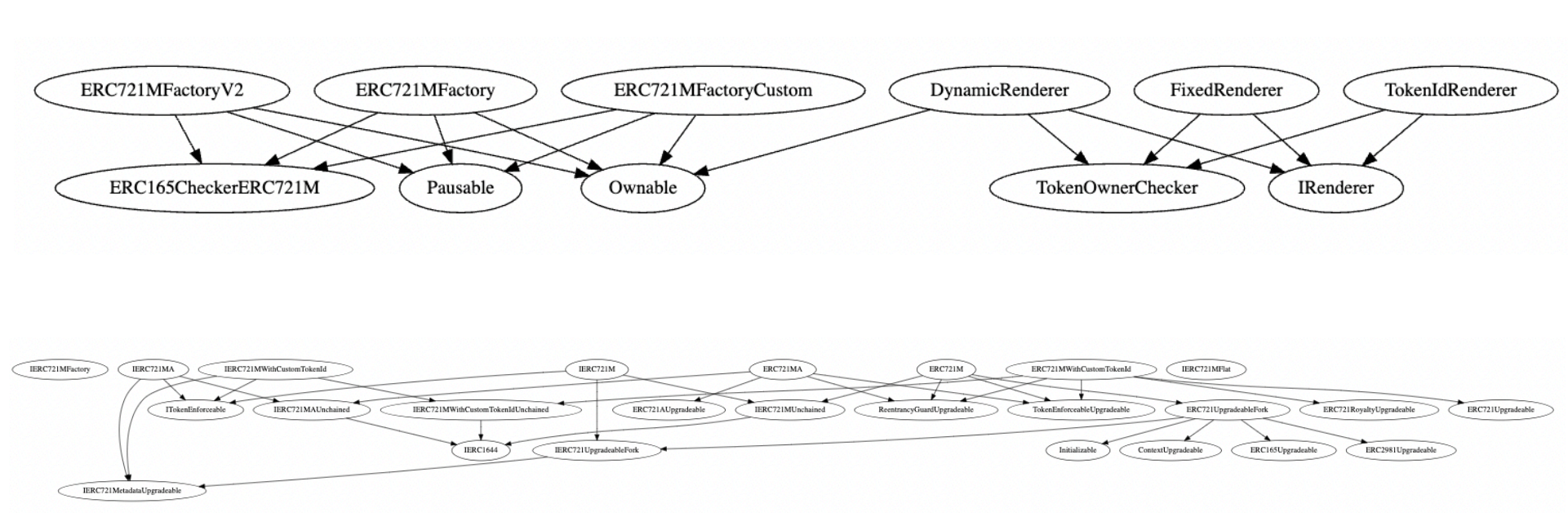
./contracts/ERC721M/factory/ERC721MFactory.sol	0fe6c802ca03ce6f460f39116d17e755
./contracts/ERC721M/factory/IERC721MFactory.sol	1c2a8471c20517c83b7d956ca670f747
./contracts/ERC721M/factory/ERC721MFactoryV2.sol	036cc575a96766ce1a8310a839133020

Updated (05.10.2023)

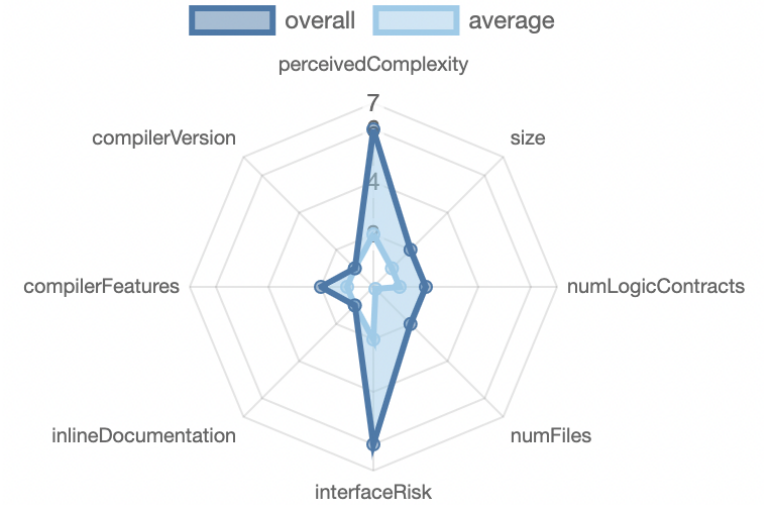
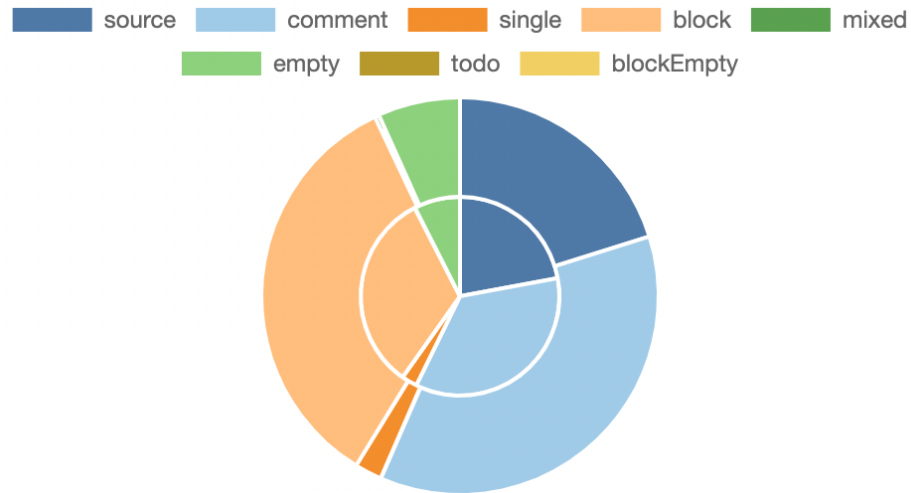
File	Fingerprint (MD5)
./contracts/ERC721M/renderer/TokenIdRenderer.sol	66ad1d372d0390aa1c84ef65ec7820e8
./contracts/ERC721M/renderer/FixedRenderer.sol	1cecd02ca9fb2738418c541a36dbc91
./contracts/ERC721M/renderer/IRenderer.sol	adc6465ad760e5c4397552998f7a6f13
./contracts/ERC721M/renderer/DynamicRenderer.sol	d58c2630cb0eeb438412db2ace42bea5
./contracts/ERC721M/IERC721M.sol	19b3ef09e5c8ad6b6732cbc778ae351f
./contracts/ERC721M/ERC165CheckerERC721M.sol	482085b24ca9b04fe97b1dcefa460228
./contracts/ERC721M/IERC721MFlat.sol	f2929ab7fb4fb1c23041533f99ec9e5f
./contracts/ERC721M/ERC721M.sol	bef4688a4917886b8016e191c696cf67
./contracts/ERC721M/ERC721UpgradeableFork.sol	f04db809d87953adc3fda9701eb89a4a
./contracts/ERC721M/ERC721A/ERC721MA.sol	972bfb3c38546a6d42682c7b61981fcd
./contracts/ERC721M/ERC721A/IERC721MA.sol	8b1d18d888b2b81dd2bff37428681b48
./contracts/ERC721M/ERC721MWithCustomTokenId/ ERC721MWithCustomTokenId.sol	bd81d8b7a81e1db89c52d6494e690731
./contracts/ERC721M/ERC721MWithCustomTokenId/ ERC721MWithCustomTokenId.sol	ef7e418597e56ca47f2577d58a3bb738
./contracts/ERC721M/IERC721UpgradeableFork.sol	a9b7df44f362f7c2905d146e1ce20a9c
./contracts/ERC721M/factory/ERC721MFactoryCusto m.sol	a2bec1fc0c39f1ba37ea2639f74b9e60
./contracts/ERC721M/factory/ERC721MFactory.sol	c6c1dfb1fa94ae2d85206cb3547ad505
./contracts/ERC721M/factory/IERC721MFactory.sol	1c2a8471c20517c83b7d956ca670f747
./contracts/ERC721M/factory/ERC721MFactoryV2.sol	f381dec24641acdc618e9f8703f26183

5.2 CallGraph





5.4 Source Lines & Risk





5.5 Capabilities


Solidity Versions observed		 Experimental Features		 Can Receive Funds		 Uses Assembly		 Has Destroyable Contracts			
0.8.21				yes		yes (1 asm blocks)					
 Transfers ETH		 Low-Level Calls		 DelegateCall		 Uses Hash Functions		 ECTrecover		 New/Create/Create2	

Exposed Functions

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

 Public	 Payable				
195	2				
External	Internal	Private	Pure	View	
162	170	1	1	92	

StateVariables










Total	 Public
27	12













5.6 Dependencies / External imports

Dependency / Import Path	Source
@openzeppelin/contracts-upgradeable/interfaces/IERC2981Upgradeable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.6.0/contracts/interfaces/IERC2981Upgradeable.sol
@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.6.0/contracts/proxy/utils/Initializable.sol
@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.6.0/contracts/security/ReentrancyGuardUpgradeable.sol
@openzeppelin/contracts-upgradeable/token/ERC721/ERC721Upgradeable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.6.0/contracts/token/ERC721/ERC721Upgradeable.sol
@openzeppelin/contracts-upgradeable/token/ERC721/IERC721ReceiverUpgradeable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.6.0/contracts/token/ERC721/IERC721ReceiverUpgradeable.sol
@openzeppelin/contracts-upgradeable/token/ERC721/IERC721Upgradeable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.6.0/contracts/token/ERC721/IERC721Upgradeable.sol
@openzeppelin/contracts-upgradeable/token/ERC721/extensions/ERC721RoyaltyUpgradeable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.6.0/contracts/token/ERC721/extensions/ERC721RoyaltyUpgradeable.sol
@openzeppelin/contracts-upgradeable/token/ERC721/extensions/IERC721MetadataUpgradeable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.6.0/contracts/token/ERC721/extensions/IERC721MetadataUpgradeable.sol
@openzeppelin/contracts-upgradeable/token/common/ERC2981Upgradeable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.6.0/contracts/token/common/ERC2981Upgradeable.sol

Dependency / Import Path	Source
@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.6.0/contracts/utils/AddressUpgradeable.sol
@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.6.0/contracts/utils/ContextUpgradeable.sol
@openzeppelin/contracts-upgradeable/utils/StringsUpgradeable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.6.0/contracts/utils/StringsUpgradeable.sol
@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.6.0/contracts/utils/introspection/ERC165Upgradeable.sol
@openzeppelin/contracts-upgradeable/utils/introspection/IERC165Upgradeable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/tree/v4.6.0/contracts/utils/introspection/IERC165Upgradeable.sol
@openzeppelin/contracts/access/Ownable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.6.0/contracts/access/Ownable.sol
@openzeppelin/contracts/proxy/Clones.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.6.0/contracts/proxy/Clones.sol
@openzeppelin/contracts/security/Pausable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.6.0/contracts/security/Pausable.sol
@openzeppelin/contracts/utils/Strings.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.6.0/contracts/utils/Strings.sol
@openzeppelin/contracts/utils/introspection/ERC165Checker.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.6.0/contracts/utils/introspection/ERC165Checker.sol

5.7 Source Unites in Scope

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
	src/contracts/ERC721M/factory/ERC721MFactoryV2.sol	1	_____	151	132	49	64	53	💰 Σ
	src/contracts/ERC721M/factory/IERC721MFactory.sol	_____	1	25	6	3	1	15	_____
	src/contracts/ERC721M/factory/ERC721MFactory.sol	1	_____	196	180	81	75	72	💰 Σ
	src/contracts/ERC721M/factory/ERC721MFactoryCustom.sol	1	_____	115	99	30	57	32	_____
	src/contracts/ERC721M/IERC721UpgradeableFork.sol	_____	1	53	22	5	35	11	_____
	src/contracts/ERC721M/ERC721MWithCustomTokenId/IERC721MWithCustomTokenId.sol	_____	2	199	38	9	141	36	_____
	src/contracts/ERC721M/ERC721MWithCustomTokenId/ERC721MWithCustomTokenId.sol	1	_____	410	372	127	214	116	Σ
	src/contracts/ERC721M/ERC721A/IERC721MA.sol	_____	2	177	38	9	127	32	_____
	src/contracts/ERC721M/ERC721A/ERC721MA.sol	1	_____	485	454	153	253	138	Σ

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
	src/contracts/ERC721M/ERC721UpgradeableFork.sol	1	_____	624	609	202	338	173	
	src/contracts/ERC721M/ERC721M.sol	1	_____	367	344	112	206	107	Σ
	src/contracts/ERC721M/IERC721MFlat.sol	_____	1	559	52	23	419	99	_____
	src/contracts/ERC721M/ERC165CheckerERC721M.sol	1	_____	31	31	15	11	5	_____
	src/contracts/ERC721M/IERC721M.sol	_____	2	178	35	8	132	34	_____
	src/contracts/ERC721M/renderer/DynamicRenderer.sol	1	_____	71	71	29	32	37	_____
	src/contracts/ERC721M/renderer/IRenderer.sol	_____	1	31	21	3	23	5	_____
	src/contracts/ERC721M/renderer/FixedRenderer.sol	1	_____	60	55	17	34	14	_____
	src/contracts/ERC721M/renderer/TokenIdRenderer.sol	1	_____	66	66	21	36	24	_____
	Totals	11	10	3798	2625	896	2198	1003	

- **Lines:** total lines of the source unit
- **nLines:** normalized lines of the source unit (e.g. normalizes functions spanning multiple lines)
- **nSLOC:** normalized source lines of code (only source-code lines; no comments, no blank lines)
- **Comment Lines:** lines containing single or block comments
- **Complexity Score:** a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...)

6. Scope of Work

The Syndicate Team provided us with the files that needs to be tested. The scope of the audit are the ERC721M contracts.

The team put forward the following assumptions regarding the security, usage of the contracts:

1. The audit should comprehensively evaluate the contracts to ascertain that it adheres to industry best practices. This includes checking for vulnerabilities like reentrancy attacks, overflow/underflow issues, and other common smart contract exploits.
2. The audit must validate the governance and administrative functions within the contract. The auditor should ensure that only authorized entities can invoke these functions and that there are adequate safeguards against unauthorized or unintended usage.
3. Given the contract's role in managing token URIs, it's vital to validate that any token-related functions, such as minting, burning, or updating URIs, are working correctly and securely. Any discrepancies could lead to significant asset mismanagement or loss.
4. The audit should thoroughly test all functionalities of the contract, ensuring each function behaves as expected. This includes verifying that each function returns the expected outputs, handles failures gracefully, and doesn't exhibit unintended side effects.
5. Given that smart contracts often interact with other contracts or external entities, the audit should assess the security implications of these interactions. This would involve verifying that the contract remains secure when interfaced with both known and unknown external contracts.

The main goal of this audit was to verify these claims. The auditors can provide additional feedback on the code upon the client's request.

6.1 Findings Overview



No	Title	Severity	Status
6.2.1	Unsafe Minting Function	MEDIUM	ACKNOWLEDGED
6.2.2	Unsafe Transfer Function	MEDIUM	ACKNOWLEDGED
6.2.3	Missing safeMint Function	MEDIUM	ACKNOWLEDGED
6.2.4	Overpowered Admin Rights	MEDIUM	FIXED
6.2.5	Improper Input Validation	LOW	FIXED



6.2.6	Improper Input Validation	LOW	FIXED
6.2.7	Wrong Function Modifier	LOW	FIXED
6.2.8	Repeated Existence Check	INFORMATIONAL	FIXED
6.2.9	Duplicated Functions in Contract	INFORMATIONAL	ACKNOWLEDGED
6.2.10	Duplicated Paused Check	INFORMATIONAL	FIXED

6.2 Manual and Automated Vulnerability Test

CRITICAL ISSUES

During the audit, softstack's experts found **no Critical issues** in the code of the smart contract.

HIGH ISSUES

During the audit, softstack's experts found **no High issues** in the code of the smart contract.

MEDIUM ISSUES

During the audit, softstack's experts found **4 Medium issues** in the code of the smart contract.

6.2.1 Unsafe Minting Function

Severity: MEDIUM

Status: ACKNOWLEDGED

Code: CWE-862

File(s) affected: ERC721MWithCustomTokenId.sol, ERC721MA.sol

Update (Project): This was a deliberate choice. We purposely used the `_mint()` function so we save on gas costs. Given the use cases our clients have, i.e. minting hundred of thousands NFTs and being a community of users, the choice was to provide gas savings in minting.

Attack / Description	The minting functions are utilizing the ERC721 <code>_mint</code> function. The <code>_mint</code> function does not check, if the token receiver can receive ERC721 tokens, which may result in the loss of funds.
Code	<pre> ERC721MWithCustomTokenId Line 207, 228 & 258 ERC721MA Line 218, 236 & 264 { ... _mint(account, tokenId); ... }</pre>
Result/Recommendation	It is recommended to use the <code>_safeMint</code> function in favor of the <code>_mint</code> function, which is checking for the ERC721Receiver interface, if the receiver is a smart contract to prevent the loss of funds.

6.2.2 Unsafe Transfer Function

Severity: MEDIUM

Status: ACKNOWLEDGED

Code: CWE-862

File(s) affected: ERC721MWithCustomTokenId.sol

Update (Project): Similar intention as issue 6.2.1

Attack / Description	The controllerTransfer function utilizes the ERC721 <code>_transfer</code> function. The <code>_transfer</code> function does not check, if the token receiver is capable of receiving ERC721 tokens, which may result in the loss of funds.
Code	<pre> ERC721MWithCustomTokenId Line 329 – 333 ERC721M Line 296 - 300</pre>

	<pre>function controllerTransfer(address from, address to, uint256 tokenId) external onlyAdmin whenControllable { // ERC721Upgradeable._transfer checks that from == ownerOf(tokenId) _transfer(from, to, tokenId); emit ControllerTransfer(owner(), from, to, tokenId); }</pre>
Result/Recommendation	It is recommended to use the <code>_safeTransfer</code> function in favor of the <code>_transfer</code> function, which is checking for the ERC721Receiver interface, if the receiver is a smart contract to prevent the loss of funds.

6.2.3 Missing safeMint Function

Severity: MEDIUM

Status: ACKNOWLEDGED

Code: NA

File(s) affected: ERC721UpgradeableFork.sol

Update (Project): Similar intention as issue 6.2.1

Attack / Description	The ERC721UpgradeableFork contract is a fork of OpenZeppelins ERC721Upgradeable but is missing the <code>_safeMint</code> function. The current implementation implements the <code>_mint</code> function, which is missing some safety checks on minting new tokens. It does not ensure that the token receiver can receive ERC721 tokens which can result in the permanent locking of minted tokens.
Code	NA
Result/Recommendation	Check whether the token receiver can receive ERC721 tokens by implementing a version of OpenZeppelins <code>_safeMint</code> function to ensure that tokens are not locked forever.

6.2.4 Overpowered Admin Rights

Severity: MEDIUM

Status: FIXED

Code: CWE-284

File(s) affected: ERC721M.sol, ERC721MA.sol, ERC721MWithCustomTokenId.sol

Update (Project): This is a feature of our product and not a bug. We provide clients APIs so they can manage their product's users via ERC721 tokens, think of content access and membership of clubs management, etc. For this we need to have admin functions so the client can control the user accessibility say through minting, burning and transferring tokens on behalf of their users.

Attack / Description	This token contracts are holding admin functions to transfer and redeem any ERC721 token of that specific contract. Therefore, the admins must not own these tokens and the contract has to be in controllable state, which is the default case. Subsequently, any admin can transfer or burn any tokens of any users by default.
Code	<pre>ERC721MA Line 322 - 331 ERC721M Line 273 – 278 ERC721MWithCustomTokenId Line 304 - 311 function controllerRedeem(address account, uint256 tokenId) external onlyAdmin whenControllable { require(account == ownerOf(tokenId), "ERC721M: wrong sender of token"); totalSupply -= 1; _burn(tokenId); emit ControllerRedemption(owner(), account, tokenId); } ERC721MA Line 348 - 353 ERC721M Line 296 – 300 ERC721MWithCustomTokenId Line 329 - 333 function controllerTransfer(address from, address to, uint256 tokenId) external onlyAdmin whenControllable {</pre>

	<pre>// ERC721Upgradeable._transfer checks that from == ownerOf(tokenId) _transfer(from, to, tokenId); emit ControllerTransfer(owner(), from, to, tokenId); }</pre>
Result/Recommendation	Remove overpowered admin functions such as controllerRedeem and controllerTransfer to eliminate central points of failure. It reduces the control of central authorities and enhances the trust and security of the contracts.

LOW ISSUES

During the audit, softstack's experts found **3 Low issues** in the code of the smart contract

6.2.5 Improper Input Validation

Severity: LOW

Status: FIXED

Code: CWE-20

File(s) affected: ERC721MFactory.sol, ERC721MFactoryV2.sol

Update (Project): fixed within commit a4fc81e23d8e2331a4f234601a6a664226f9cc14

Attack / Description	The create function accepts setupContracts and data arrays for initializing calls on these contracts but does not ensure that their lengths are the same, which may result in undesired behaviour.
Code	ERC721MFactory Line 89 – 115 ERC721MFactoryV2 Line 61 – 72 <pre>function create(string memory name, string memory symbol, bytes32 salt, address[] calldata setupContracts,</pre>

	<pre> bytes[] calldata data) external payable whenNotPaused returns (address token) { ... uint256 length = setupContracts.length; for (uint256 i; i < length;) { // solhint-disable-next-line avoid-low-level-calls (bool sent,) = setupContracts[i].call(data[i]); require(sent, "ERC721MFactory: Error making setup calls"); unchecked { ++i; } } } </pre>
Result/Recommendation	<p>Add a require statement to check if the lengths of both arrays are equal before proceeding and make sure, that the caller is aware of ordering the call data in the same way the setupContracts are ordered.</p>

6.2.6 Improper Input Validation

Severity: LOW

Status: FIXED

Code: CWE-20

File(s) affected: ERC721MA.sol, ERC721MWithCustomTokenId.sol

Update (Project): fixed within commit a4fc81e23d8e2331a4f234601a6a664226f9cc14

Attack / Description	<p>The bulkMintToNAddresses function accepts accounts and quantities arrays for minting different amounts of ERC721MA tokens, to different addresses but does not ensure that their lengths are the same, which may result in undesired behaviour.</p>
-----------------------------	--

Code	ERC721MA Line 254 – 271 ERC721MWithCustomTokenId Line 246 - 265 <pre> function bulkMintToNAddresses(address[] calldata accounts, uint256[] calldata quantities) external nonReentrant returns (bool) { require(accounts.length != 0, "ERC721MA: bulk minting zero tokens"); uint256 length = accounts.length; for (uint256 i = 0; i < length;) { _mint(accounts[i], quantities[i]); unchecked { ++i; } } return true; } </pre>
Result/Recommendation	Add a require statement to check if the lengths of accounts and quantities array are equal before proceeding and make sure that the caller is aware of ordering the quantities in the same way the accounts are ordered.

6.2.7 Wrong Function Modifier

Severity: LOW

Status: FIXED

Code: NA



File(s) affected: ERC721MFactoryV2.sol

Update (Project): fixed within commit a4fc81e23d8e2331a4f234601a6a664226f9cc14

Attack / Description	The create function is marked as payable but it does not handle any incoming funds, which may lead to permanently locking funds.
Code	Line 61 – 72 (ERC721MFactoryV2.sol) <code>function create(...) external payable whenNotPaused returns (address token) {</code>
Result/Recommendation	Remove the payable modifier to eliminate the risk of permanently locking funds by accidentally sending Ether on function call. This may be the case on migrating from V1 (where the function expects receiving funds) to V2 without changing the call parameters properly.

INFORMATIONAL ISSUES

During the audit, softstack's experts found **3 Informational issues** in the code of the smart contract.

6.2.8 Repeated Existence Check

Severity: INFORMATIONAL

Status: FIXED

Code: CWE-398

File(s) affected: ERC721UpgradeableFork.sol

Update (Project): fixed within commit a4fc81e23d8e2331a4f234601a6a664226f9cc14

Attack / Description	In the <code>_isApprovedOrOwner</code> function, the <code>_exists(tokenId)</code> check is being performed twice. Once directly inside <code>_isApprovedOrOwner</code> and again within the <code>ownerOf(tokenId)</code> .
-----------------------------	--



Code	Line 344 – 348 <pre>function _isApprovedOrOwner(address spender, uint256 tokenId) internal view virtual returns (bool) { require(!_exists(tokenId), "ERC721: operator query for nonexistent token"); address owner = ERC721UpgradeableFork.ownerOf(tokenId); return (spender == owner getApproved(tokenId) == spender isApprovedForAll(owner, spender)); }</pre> Line 186 – 187 <pre>function ownerOf(uint256 tokenId) public view virtual override returns (address) { require(!_exists(tokenId), "ERC721: owner query for nonexistent token");</pre>
Result/Recommendation	Remove the <code>_exists(tokenId)</code> check from <code>_isApprovedOrOwner</code> since the subsequent <code>ownerOf</code> call checks the same condition.

6.2.9 Duplicated Functions in Contract

Severity: INFORMATIONAL

Status: ACKNOWLEDGED

Code: CWE-398

File(s) affected: ERC721MA.sol

Update (Project): This is deliberate choice so it maintains the same function signatures as the other ERC721 contracts we have in the codebase.

Attack / Description	The functions <code>mintTo</code> and <code>bulkMintToOneAddress</code> at line numbers 216 & 235 appear to perform the exact same function, leading to redundancy.
Code	Line 216 – 220 <pre>function mintTo(address account, uint256 quantity) external nonReentrant returns (bool) {</pre>



	<pre> // calls _beforeTokenTransfer for Guard checks prior to transfer _mint(account, quantity); return true; } Line 235 – 238 function bulkMintToOneAddress(address account, uint256 amount) external nonReentrant returns (bool) { _mint(account, amount); return true; } </pre>
Result/Recommendation	Refactor the code to remove one of the redundant functions and update all references to use a single, unified function.

6.2.10 Duplicated Paused Check

Severity: INFORMATIONAL

Status: FIXED

Code: CWE-398

File(s) affected: ERC721MFactory.sol, ERC721MFactoryV2.sol, ERC721MFactoryCustom.sol

Update (Project): fixed within commit a4fc81e23d8e2331a4f234601a6a664226f9cc14

Attack / Description	The <i>create</i> function of the factory is only callable if the contract is not paused by being locked with the <i>whenNotPaused</i> modifier. The function calls the internal function <i>_clone</i> , which is also locked with the same modifier and does the same check again.
Code	ERC721MFactory Line 89 – 115 ERC721MFactoryV2 Line 61 – 86 ERC721MFactoryCustom Line 50 – 61 <pre>function create(</pre>

	<pre> ...) external payable whenNotPaused returns (address token) { ... token = _clone(name, symbol, salt); ... } </pre> <p>ERC721MFactory Line 124 – 128 ERC721MFactoryV2 Line 95 – 109 ERC721MFactoryCustom Line 75 – 89</p> <pre> function _clone(string memory name, string memory symbol, bytes32 salt) internal whenNotPaused returns (address token) </pre>
Result/Recommendation	Remove the <i>whenNotPaused</i> modifier from the internal <code>_clone</code> function to reduce the overall gas consumption.

6.3 SWC Attacks

ID	Title	Relationships	Test Result
SWC-131	Presence of unused variables	CWE-1164: Irrelevant Code	✓
SWC-130	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	✓


ID	Title	Relationships	Test Result
SWC-129	Typographical Error	CWE-480: Use of Incorrect Operator	✓
SWC-128	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	✓
SWC-127	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	✓
SWC-125	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	✓
SWC-124	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	✓
SWC-123	Requirement Violation	CWE-573: Improper Following of Specification by Caller	✓
SWC-122	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	✓
SWC-121	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	✓
SWC-120	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	✓
SWC-119	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	✓

ID	Title	Relationships	Test Result
SWC-118	Incorrect Constructor Name	CWE-665: Improper Initialization	✓
SWC-117	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	✓
SWC-116	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	✓
SWC-115	Authorization through tx.origin	CWE-477: Use of Obsolete Function	✓
SWC-114	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	✓
SWC-113	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	✓
SWC-112	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	✓
SWC-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	✓
SWC-110	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	✓
SWC-109	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	✓
SWC-108	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	✓

ID	Title	Relationships	Test Result
SWC-107	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	✓
SWC-106	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	✓
SWC-105	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	✓
SWC-104	Unchecked Call Return Value	CWE-252: Unchecked Return Value	✓
SWC-103	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	✓
SWC-102	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	✓
SWC-101	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	✓
SWC-100	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	✓

6.4 Verify Claims


6.4.1 The audit should comprehensively evaluate the contracts to ascertain that it adheres to industry best practices. This includes checking for vulnerabilities like reentrancy attacks, overflow/underflow issues, and other common smart contract exploits.

Status: tested and verified 

6.4.2 The audit must validate the governance and administrative functions within the contract. The auditor should ensure that only authorized entities can invoke these functions and that there are adequate safeguards against unauthorized or unintended usage.

Status: tested and verified 


6.4.3 Given the contract's role in managing token URIs, it's vital to validate that any token-related functions, such as minting, burning, or updating URIs, are working correctly and securely. Any discrepancies could lead to significant asset mismanagement or loss.

Status: tested and verified 

6.4.4 The audit should thoroughly test all functionalities of the contract, ensuring each function behaves as expected. This includes verifying that each function returns the expected outputs, handles failures gracefully, and doesn't exhibit unintended side effects.

Status: tested and verified 

6.4.5 Given that smart contracts often interact with other contracts or external entities, the audit should assess the security implications of these interactions. This would involve verifying that the contract remains secure when interfaced with both known and unknown external contracts.

Status: tested and verified 

6.5 Unit Tests

Running 5 tests for src/test/ERC721M/BulkMintGasTest.t.sol:BulkMintOwnerOfGasTest

[PASS] testBulkMint1000Gas() (gas: 2581638)

[PASS] testBulkMint1000OwnerOf1000Gas() (gas: 128501)

[PASS] testBulkMint1000OwnerOf1Gas() (gas: 15372)

[PASS] testBulkMint1000OwnerOf49Gas() (gas: 126279)

[PASS] testBulkMint1000OwnerOf999Gas() (gas: 126280)

Test result: ok. 5 passed; 0 failed; 0 skipped; finished in 9.04ms

Running 6 tests for src/test/modules/ERC721M/mint/AdminMintModuleTest.t.sol:AdminMintModuleTest

[PASS] testAddAdmins() (gas: 51553)

[PASS] testMintTokenWhenAdminsCall() (gas: 203096)

[PASS] testOwnerisAdmin() (gas: 18418)

[PASS] testRemoveAdmins() (gas: 38241)

[PASS] testRevertWhenAddingOrRemovingAdminsFromANonAdminAddress() (gas: 45079)

[PASS] testRevertWhenNotanAdmin() (gas: 37410)

Test result: ok. 6 passed; 0 failed; 0 skipped; finished in 4.65ms

Running 8 tests for src/test/guards/mixins/MaxTotalSupplyERC721Test.t.sol:MaxTotalSupplyERC721Test

[PASS] testBulkMintOverTotalSupply() (gas: 48942)

[PASS] testBulkMintUnderTotalSupply() (gas: 49027)

[PASS] testBurnUnderTotalSupply() (gas: 45243)

[PASS] testRequirementsNotSet() (gas: 18557)

[PASS] testRevertNonOwner() (gas: 28053)

[PASS] testRevertUnderCurrentTotalSupply() (gas: 41805)

[PASS] testSetMaxTotalSupplyERC721() (gas: 47201)

[PASS] testTransferUnderTotalSupply() (gas: 47384)

Test result: ok. 8 passed; 0 failed; 0 skipped; finished in 1.48ms

Running 4 tests for src/test/utills/AdministrableTest.t.sol:AdministrableTest

[PASS] testGrantAdmin() (gas: 163238)

[PASS] testRenounceAdmin() (gas: 60556)

[PASS] testRevertNonAdminCannotGrantAdmin() (gas: 60720)

[PASS] testRevokeAdmin() (gas: 73036)

Test result: ok. 4 passed; 0 failed; 0 skipped; finished in 1.26ms

Running 8 tests for src/test/ERC721M/renderer/TokenIdRendererTest.t.sol:TokenIdRendererTest

[PASS] testBaseURINotSetTokenURI() (gas: 335507)

[PASS] testBaseURINotSetTokenURIOf() (gas: 9437)

[PASS] testBasicTokenURI() (gas: 19183)

[PASS] testBasicTokenURIOf() (gas: 15486)

[PASS] testFuzzTokenURI(uint256) (runs: 256, μ : 24336, \sim : 21002)

[PASS] testFuzzTokenURIOf(uint256) (runs: 256, μ : 20325, \sim : 17185)

[PASS] testRevertUpdateBaseURINonOwner() (gas: 22012)

[PASS] testUpdateBaseURI() (gas: 30481)

Test result: ok. 8 passed; 0 failed; 0 skipped; finished in 93.43ms

Running 4 tests for src/test/utills/BatchableTest.t.sol:BatchableTest

[PASS] testAdminCanUpdateBatcher() (gas: 43681)

[PASS] testRevertCannotGrantAdminToBatcher() (gas: 87962)

[PASS] testRevertNonAdminCannotUpdateBatcher() (gas: 18049)

[PASS] testUpdatingBatcherToAdminRevokesAdmin() (gas: 97675)

Test result: ok. 4 passed; 0 failed; 0 skipped; finished in 1.44ms

Running 4 tests for src/test/utills/BatcherERC721Test.t.sol:ERC721BatcherTest

[PASS] testBatcher() (gas: 855911)

[PASS] testBatcherCanGrantAndRevokeAdmin() (gas: 413580)

[PASS] testBatcherCanRecoverERC721() (gas: 261676)

[PASS] testRevertBatcherNonOwner() (gas: 208027)

Test result: ok. 4 passed; 0 failed; 0 skipped; finished in 4.62ms

Running 3 tests for src/test/ERC721M/ERC721RecoverTest.t.sol:ERC721RecoverTest

[PASS] testRecover(uint256,uint256,address) (runs: 256, μ : 114141, \sim : 119248)

[PASS] testRevertFallback() (gas: 19936)

[PASS] testRevertRecover(uint256) (runs: 256, μ : 23501, \sim : 23501)

Test result: ok. 3 passed; 0 failed; 0 skipped; finished in 106.10ms

Running 6 tests for src/test/guards/ERC721M/GuardMintERC721Test.t.sol:GuardMintERC721Test

[PASS] testCollectiveInitializedAsPaused() (gas: 57235)

[PASS] testIsAllowedReturnsFalseWhenModuleIsPaused() (gas: 93622)

[PASS] testIsAllowedReturnsFalseWhenModuleNotAllowed() (gas: 21163)

[PASS] testIsAllowedReturnsTrueWhenModuleAllowed() (gas: 23321)

[PASS] testMintPausesWhenOwnerPauses() (gas: 72724)

[PASS] testRevertsWhenNonOwnerTriesToPause() (gas: 65452)

Test result: ok. 6 passed; 0 failed; 0 skipped; finished in 2.67ms

Running 11 tests for src/test/ERC721M/ERC721RoyaltyTest.t.sol:ERC721RoyaltyTest

[PASS] testDefaultRoyalty() (gas: 46943)

[PASS] testNoRoyalty() (gas: 13372)

[PASS] testRevertSetDefaultRoyaltyNonOwner() (gas: 19116)

[PASS] testRevertSetDefaultRoyaltyNullReceiver() (gas: 21404)

[PASS] testRevertSetDefaultRoyaltyTooLarge(uint96) (runs: 256, μ : 22005, \sim : 22005)

[PASS] testRevertSetTokenRoyaltyIdInvalid(uint256) (runs: 256, μ : 24232, \sim : 24232)

[PASS] testRevertSetTokenRoyaltyNonOwner() (gas: 19332)

[PASS] testRevertSetTokenRoyaltyNullReceiver() (gas: 25920)

[PASS] testRevertSetTokenRoyaltyTooLarge(uint96) (runs: 256, μ : 26565, \sim : 26565)

[PASS] testSupportERC2981Interface() (gas: 8642)

[PASS] testTokenRoyalty() (gas: 78633)

Test result: ok. 11 passed; 0 failed; 0 skipped; finished in 164.55ms

Running 9 tests for src/test/ERC721M/ControllableTest.t.sol:IsControllableTest

[PASS] testCollectiveIsInitiallyControllable() (gas: 14695)

[PASS] testControllerRedeemFromTokenOwner(uint256) (runs: 256, μ : 151474, \sim : 151745)
[PASS] testControllerTransferToOtherMember(uint256) (runs: 256, μ : 172932, \sim : 176306)
[PASS] testControllerTransferToOwner(uint256) (runs: 256, μ : 185406, \sim : 191705)
[PASS] testRevertControllerRedeemAfterControllerDisabled(uint256) (runs: 256, μ : 29477, \sim : 29477)
[PASS] testRevertControllerRedeemFromNonOwner(uint256) (runs: 256, μ : 23950, \sim : 23950)
[PASS] testRevertControllerRedeemFromWrongOwner(uint256) (runs: 256, μ : 85030, \sim : 91028)
[PASS] testRevertControllerTransferAfterControllerDisabled(uint256) (runs: 256, μ : 29700, \sim : 29700)
[PASS] testRevertNonOwnerCannotDisable() (gas: 20895)
Test result: ok. 9 passed; 0 failed; 0 skipped; finished in 603.36ms

Running 13 tests for src/test/common/TokenEnforceableTest.t.sol:TokenEnforceableUpgradeableTest

[PASS] testCannotUnlockBurnGuard(address) (runs: 256, μ : 49927, \sim : 49927)
[PASS] testCannotUnlockMintGuard(address) (runs: 256, μ : 49782, \sim : 49782)
[PASS] testCannotUnlockTransferGuardAfterLocking(address) (runs: 256, μ : 49866, \sim : 49866)
[PASS] testGuardLockedGetters() (gas: 123384)
[PASS] testNonAdminCannotUpdateGuards(address) (runs: 256, μ : 41090, \sim : 41090)
[PASS] testRevertLockAllGuardsAtOnce(address) (runs: 256, μ : 93339, \sim : 93339)
[PASS] testRevertOnlyAdminCanLockGuards() (gas: 18802)
[PASS] testRevertUpdateBurnGuardAfterLocking(address) (runs: 256, μ : 47451, \sim : 47451)
[PASS] testRevertUpdateMintGuardAfterLocking(address) (runs: 256, μ : 47304, \sim : 47304)
[PASS] testRevertUpdateTransferGuardAfterLocking(address) (runs: 256, μ : 47346, \sim : 47346)
[PASS] testUpdateBurnGuard(address) (runs: 256, μ : 45467, \sim : 45545)
[PASS] testUpdateMintGuard(address) (runs: 256, μ : 45591, \sim : 45591)
[PASS] testUpdateTransferGuard(address) (runs: 256, μ : 45653, \sim : 45653)
Test result: ok. 13 passed; 0 failed; 0 skipped; finished in 762.26ms

Running 23 tests for src/test/modules/ERC721Collective/mint/RequiredTokensMintModuleTest.t.sol:RequiredTokensMintModuleTest

[PASS] testRedeem100Gas() (gas: 6168288)
[PASS] testRedeem10Gas() (gas: 729193)
[PASS] testRedeemBalances() (gas: 194774)
[PASS] testRedeemGas() (gas: 179658)

[PASS] testRedeemMany(uint256) (runs: 256, μ : 2563811, \sim : 2233029)
[PASS] testRedeemManyBalances() (gas: 811689)
[PASS] testRedeemManyNoPrice() (gas: 6158274)
[PASS] testRedeemNoPrice() (gas: 169822)
[PASS] testRevertRedeemAndRedeemManyOverlap() (gas: 482238)
[PASS] testRevertRedeemManyMoreThanBalance(uint256) (runs: 256, μ : 58785, \sim : 58785)
[PASS] testRevertRedeemManyNoEth() (gas: 55564)
[PASS] testRevertRedeemManyNonOwner() (gas: 97221)
[PASS] testRevertRedeemManyTwiceFullOverlap() (gas: 765637)
[PASS] testRevertRedeemManyTwicePartialOverlap() (gas: 984333)
[PASS] testRevertRedeemManyWrongPrice() (gas: 71128)
[PASS] testRevertRedeemMoreThanBalance(uint16) (runs: 256, μ : 60722, \sim : 60722)
[PASS] testRevertRedeemNoEth() (gas: 44007)
[PASS] testRevertRedeemNonOwner() (gas: 57149)
[PASS] testRevertRedeemTwice() (gas: 202680)
[PASS] testRevertRedeemWrongPrice() (gas: 59666)
[PASS] testRevertRequiredTokenNotSet() (gas: 59442)
[PASS] testRevertUpdatePriceNonOwner() (gas: 29664)
[PASS] testUpdateRequirements() (gas: 101159)
Test result: ok. 23 passed; 0 failed; 0 skipped; finished in 846.03ms

Running 6 tests for src/test/ERC20Club/ControllableTest.t.sol:IsControllableTest

[PASS] testClubIsInitiallyControllable() (gas: 10337)
[PASS] testControllerRedeem(uint256) (runs: 256, μ : 57108, \sim : 57108)
[PASS] testControllerTransfer(uint256) (runs: 256, μ : 87396, \sim : 87396)
[PASS] testRevertControllerRedeemAfterControllerDisabled(uint256) (runs: 256, μ : 31153, \sim : 31153)
[PASS] testRevertControllerTransferAfterControllerDisabled(uint256) (runs: 256, μ : 31288, \sim : 31288)
[PASS] testRevertNonOwnerCannotDisable() (gas: 20840)
Test result: ok. 6 passed; 0 failed; 0 skipped; finished in 227.66ms

Running 4 tests for src/test/guards/ERC721M/GuardTransferERC721Test.t.sol:GuardTransferERC721Test

[PASS] testDisallowOwnerSenderOrRecipient(uint256) (runs: 256, μ : 53188, \sim : 53188)

[PASS] testNoUserTransfersAllowed(uint256) (runs: 256, μ : 54284, \sim : 54284)
[PASS] testOwnerControllable(uint256) (runs: 256, μ : 32569, \sim : 32569)
[PASS] testUserTransfersAllowed(uint256) (runs: 256, μ : 68865, \sim : 68865)
Test result: ok. 4 passed; 0 failed; 0 skipped; finished in 255.55ms

Running 11 tests for src/test/ERC721M/RedeemTest.t.sol:RedeemTest

[PASS] testRedeem(uint256) (runs: 256, μ : 146799, \sim : 155793)
[PASS] testRedeemGas() (gas: 50951)
[PASS] testRedeemMany() (gas: 6660616)
[PASS] testRedeemManyThenMintMany() (gas: 8948005)
[PASS] testRedeemThenNextMint(uint256) (runs: 256, μ : 233404, \sim : 244939)
[PASS] testRedeemWithApproval(uint256) (runs: 256, μ : 158394, \sim : 161610)
[PASS] testRedeemWithApprovalGas() (gas: 61147)
[PASS] testRedeemWithApprovalMany() (gas: 8831862)
[PASS] testRevertRedeem(uint256) (runs: 256, μ : 85952, \sim : 89957)
[PASS] testRevertRedeem2(uint256) (runs: 256, μ : 88039, \sim : 94651)
[PASS] testRevertRedeemWithoutApproval(uint256) (runs: 256, μ : 80508, \sim : 82412)
Test result: ok. 11 passed; 0 failed; 0 skipped; finished in 742.29ms

Running 54 tests for src/test/ERC721M/ERC721CollectiveTest.t.sol:ERC721MTest

[PASS] testBulkMint1000AddressesGas() (gas: 51800505)
[PASS] testBulkMint1000ToOneAddressGas() (gas: 2596978)
[PASS] testBulkMint100AddressesGas() (gas: 5228458)
[PASS] testBulkMint100ToOneAddressGas() (gas: 329253)
[PASS] testBulkMint10AddressesGas() (gas: 571952)
[PASS] testBulkMint10ToOneAddressGas() (gas: 120370)
[PASS] testBulkMintAndBurningAToken() (gas: 397774)
[PASS] testBulkMintAndMultipleTransfers() (gas: 352885)
[PASS] testBulkMintAndTransferTokenInMiddle1() (gas: 252179)
[PASS] testBulkMintAndTransferTokenInMiddle2() (gas: 200868)
[PASS] testBulkMintAndTransfersTokenAtEnd() (gas: 183817)
[PASS] testBulkMintNAddressesCheckOwnerOf() (gas: 155261)

[PASS] testBulkMintTo10AddressesOwnerChecks() (gas: 594533)
[PASS] testBulkMintToNAddressesAndBurn() (gas: 303121)
[PASS] testBulkMintToNAddressesAndTransfer() (gas: 277429)
[PASS] testBulkMintToNAddressesEvents() (gas: 161361)
[PASS] testBulkMintToNAddressesThenBulkMintToOne() (gas: 361424)
[PASS] testBulkMintToOneAddressEvents() (gas: 132988)
[PASS] testBulkMintTwice() (gas: 265046)
[PASS] testBulkMintTwiceAndBurnAndTransfer() (gas: 387447)
[PASS] testBulkMintTwiceAndBurnSecondOne() (gas: 321721)
[PASS] testBurnEvents() (gas: 164043)
[PASS] testEmptyRenderer() (gas: 112325)
[PASS] testLockRenderer() (gas: 25392)
[PASS] testMintDefaultGuard() (gas: 229350)
[PASS] testMintMultipleCheckOwnerOf() (gas: 257166)
[PASS] testRevertBulkMint0ToOneAddress() (gas: 24312)
[PASS] testRevertBulkMintAndBurningBasic2() (gas: 402383)
[PASS] testRevertBulkMintAndTransfersTokenAtEnd() (gas: 188286)
[PASS] testRevertBulkMintNAddressesCheckOwnerOf() (gas: 159850)
[PASS] testRevertBulkMintTo0Addresses() (gas: 22332)
[PASS] testRevertBulkMintToNAddressesAndOwnerOfBurn() (gas: 307750)
[PASS] testRevertBulkMintTwiceAndBurnAndTransfer() (gas: 392108)
[PASS] testRevertBulkMintTwiceBurnAndOwnerOf() (gas: 326514)
[PASS] testRevertCannotTransferBurnedToken() (gas: 403786)
[PASS] testRevertFallback() (gas: 20135)
[PASS] testRevertFallbackToContract() (gas: 13084)
[PASS] testRevertMintGuard() (gas: 68851)
[PASS] testRevertNonExistentTokenURI(uint256) (runs: 256, μ : 16101, \sim : 16101)
[PASS] testRevertOnlyOwnerCanLockRenderer() (gas: 21107)
[PASS] testRevertOwnerOfNonExistentTokenId() (gas: 13619)
[PASS] testRevertRendererCannotBeUpdatedAfterLocking() (gas: 385337)
[PASS] testRevertTokenIdOutOfBoundsAfterTransfers() (gas: 357357)
[PASS] testRevertTransferFrom() (gas: 134787)

[PASS] testRevertTransferFromOtherUsersTokens() (gas: 196511)
[PASS] testRevertTransferToAddressZero() (gas: 108893)
[PASS] testRevertUpdateRenderer() (gas: 379372)
[PASS] testRevertUpdateTokenURINotOwner(string) (runs: 256, μ : 27080, \sim : 27093)
[PASS] testSupplyIncrement() (gas: 228188)
[PASS] testTokenUpdateURI(string) (runs: 256, μ : 198078, \sim : 202701)
[PASS] testTransferFromDefault() (gas: 195013)
[PASS] testTransferInABulkMintGas() (gas: 115637)
[PASS] testTransferInSingleMintsGas() (gas: 66564)
[PASS] testUpdateRenderer() (gas: 389239)
Test result: ok. 54 passed; 0 failed; 0 skipped; finished in 486.47ms

Running 6 tests for src/test/ERC721M/renderer/DynamicRendererTest.t.sol:DynamicRendererTest

[PASS] testBasicTokenURIGas() (gas: 40093)
[PASS] testBasicTokenURIOf() (gas: 36032)
[PASS] testFuzzTokenURI(uint256) (runs: 256, μ : 44708, \sim : 41191)
[PASS] testFuzzTokenURIOf(uint256) (runs: 256, μ : 40302, \sim : 37130)
[PASS] testRevertUpdateBaseURINonOwner() (gas: 11208)
[PASS] testUpdateBaseURI() (gas: 49766)
Test result: ok. 6 passed; 0 failed; 0 skipped; finished in 197.09ms

Running 17 tests for src/test/ERC721M/factory/ERC721MFactoryTest.t.sol:ERC721MFactoryTest

[PASS] testCreateERC721M() (gas: 431630)
[PASS] testCreateERC721MGas()(address) (gas: 415958)
[PASS] testCreatePausedThenUnpaused() (gas: 434136)
[PASS] testCreateWithCreationFee() (gas: 475833)
[PASS] testRevertCreateWhenPaused() (gas: 44699)
[PASS] testRevertFallback() (gas: 19996)
[PASS] testRevertFallbackToContract() (gas: 10365)
[PASS] testRevertLessThanCreationFee() (gas: 70645)
[PASS] testRevertPauseNotOwner() (gas: 15821)
[PASS] testRevertSetCreationFeeNotOwner() (gas: 15911)

[PASS] testRevertSetImplementationsAllZeroAddress() (gas: 3132107)
[PASS] testRevertSetImplementationsNonCollectiveInterface() (gas: 204815)
[PASS] testRevertSetImplementationsNotOwner() (gas: 3134719)
[PASS] testRevertSetImplementationsSingleZeroAddress(uint256) (runs: 256, μ : 3138687, \sim : 3138649)
[PASS] testRevertUnpauseNotOwner() (gas: 28037)
[PASS] testSetCreationFee() (gas: 43092)
[PASS] testSetImplementations() (gas: 3164770)
Test result: ok. 17 passed; 0 failed; 0 skipped; finished in 183.94ms

Running 11 tests for src/test/ERC721M/factory/ERC721MFactoryV2Test.t.sol:ERC721MFactoryV2Test

[PASS] testCreateERC721M() (gas: 430293)
[PASS] testCreateERC721MGas():(address) (gas: 414503)
[PASS] testCreatePausedThenUnpaused() (gas: 427903)
[PASS] testRevertCreateWhenPaused() (gas: 49091)
[PASS] testRevertFallback() (gas: 15193)
[PASS] testRevertFallbackToContract() (gas: 10378)
[PASS] testRevertPauseNotOwner() (gas: 13172)
[PASS] testRevertSetImplementationNonCollectiveInterface() (gas: 194674)
[PASS] testRevertSetImplementationNotOwner() (gas: 3124518)
[PASS] testRevertUnpauseNotOwner() (gas: 20726)
[PASS] testSetImplementation() (gas: 3136449)
Test result: ok. 11 passed; 0 failed; 0 skipped; finished in 7.82ms

Running 54 tests for src/test/ERC721M/ERC721MTest.t.sol:ERC721MTest

[PASS] testBulkMint1000AddressesGas() (gas: 51800505)
[PASS] testBulkMint1000ToOneAddressGas() (gas: 2596978)
[PASS] testBulkMint100AddressesGas() (gas: 5228458)
[PASS] testBulkMint100ToOneAddressGas() (gas: 329253)
[PASS] testBulkMint10AddressesGas() (gas: 571952)
[PASS] testBulkMint10ToOneAddressGas() (gas: 120370)
[PASS] testBulkMintAndBurningAToken() (gas: 397774)
[PASS] testBulkMintAndMultipleTransfers() (gas: 352885)

[PASS] testBulkMintAndTransferTokenInMiddle1() (gas: 252179)
[PASS] testBulkMintAndTransferTokenInMiddle2() (gas: 200868)
[PASS] testBulkMintAndTransfersTokenAtEnd() (gas: 183817)
[PASS] testBulkMintNAddressesCheckOwnerOf() (gas: 155261)
[PASS] testBulkMintTo10AddressesOwnerChecks() (gas: 594533)
[PASS] testBulkMintToNAddressesAndBurn() (gas: 303121)
[PASS] testBulkMintToNAddressesAndTransfer() (gas: 277429)
[PASS] testBulkMintToNAddressesEvents() (gas: 161361)
[PASS] testBulkMintToNAddressesThenBulkMintToOne() (gas: 361424)
[PASS] testBulkMintToOneAddressEvents() (gas: 132988)
[PASS] testBulkMintTwice() (gas: 265046)
[PASS] testBulkMintTwiceAndBurnAndTransfer() (gas: 387447)
[PASS] testBulkMintTwiceAndBurnSecondOne() (gas: 321721)
[PASS] testBurnEvents() (gas: 164043)
[PASS] testEmptyRenderer() (gas: 112325)
[PASS] testLockRenderer() (gas: 25392)
[PASS] testMintDefaultGuard() (gas: 229350)
[PASS] testMintMultipleCheckOwnerOf() (gas: 257166)
[PASS] testRevertBulkMint0ToOneAddress() (gas: 24312)
[PASS] testRevertBulkMintAndBurningBasic2() (gas: 402383)
[PASS] testRevertBulkMintAndTransfersTokenAtEnd() (gas: 188286)
[PASS] testRevertBulkMintNAddressesCheckOwnerOf() (gas: 159850)
[PASS] testRevertBulkMintTo0Addresses() (gas: 22332)
[PASS] testRevertBulkMintToNAddressesAndOwnerOfBurn() (gas: 307750)
[PASS] testRevertBulkMintTwiceAndBurnAndTransfer() (gas: 392108)
[PASS] testRevertBulkMintTwiceBurnAndOwnerOf() (gas: 326514)
[PASS] testRevertCannotTransferBurnedToken() (gas: 403786)
[PASS] testRevertFallback() (gas: 20135)
[PASS] testRevertFallbackToContract() (gas: 13084)
[PASS] testRevertMintGuard() (gas: 68851)
[PASS] testRevertNonExistentTokenURI(uint256) (runs: 256, μ : 16101, \sim : 16101)
[PASS] testRevertOnlyOwnerCanLockRenderer() (gas: 21107)

[PASS] testRevertOwnerOfNonExistentTokenId() (gas: 13619)
[PASS] testRevertRendererCannotBeUpdatedAfterLocking() (gas: 385337)
[PASS] testRevertTokenIdOutOfBoundsAfterTransfers() (gas: 357357)
[PASS] testRevertTransferFrom() (gas: 134787)
[PASS] testRevertTransferFromOtherUsersTokens() (gas: 196511)
[PASS] testRevertTransferToAddressZero() (gas: 108893)
[PASS] testRevertUpdateRenderer() (gas: 379372)
[PASS] testRevertUpdateTokenURINotOwner(string) (runs: 256, μ : 27068, \sim : 27093)
[PASS] testSupplyIncrement() (gas: 228188)
[PASS] testTokenUpdateURI(string) (runs: 256, μ : 193179, \sim : 202701)
[PASS] testTransferFromDefault() (gas: 195013)
[PASS] testTransferInABulkMintGas() (gas: 115637)
[PASS] testTransferInSingleMintsGas() (gas: 66564)
[PASS] testUpdateRenderer() (gas: 389239)

Test result: ok. 54 passed; 0 failed; 0 skipped; finished in 349.22ms

7. Executive Summary

Two independent softstack experts performed an unbiased and isolated audit of the smart contract codebase provided by the Syndicate Team. The main objective of the audit was to verify the security and functionality claims of the smart contract. The audit process involved a thorough manual code review and automated security testing.

Overall, the audit identified a total of 10 issues, classified as follows:

- No critical issues were found.
- No high severity issues were found.
- Four medium severity issues were found.
- Three low severity issues were discovered
- Three informational issues were identified

The audit report provides detailed descriptions of each identified issue, including severity levels, CWE classifications, and recommendations for mitigation. It also includes code snippets, where applicable, to demonstrate the issues and suggest possible fixes. Overall, the code quality and documentation have been showing the auditor team a high grade of professionalism and have been greatly benefiting the auditing process. We advise the Syndicate team to implement the recommendations to further enhance the code's security and readability.

Update: The team has clarified that certain identified issues were deliberate design choices in alignment with the product's feature set and optimization goals, such as gas savings. All necessary issues have been mitigated and the re-check successfully done.



8. About the Auditor

Established in 2017 under the name Chainsulting, and rebranded as softstack GmbH in 2023, softstack has been a trusted name in Web3 Security space. Within the rapidly growing Web3 industry, softstack provides a comprehensive range of offerings that include software development, security, and consulting services. Softstack's competency extends across the security landscape of prominent blockchains like Solana, Tezos, Ethereum and Polygon. The company is widely recognized for conducting thorough code audits aimed at mitigating risk and promoting transparency.

The firm's proficiency lies particularly in assessing and fortifying smart contracts of leading DeFi projects, a testament to their commitment to maintaining the integrity of these innovative financial platforms. To date, softstack plays a crucial role in safeguarding over \$100 billion worth of user funds in various DeFi protocols.

Underpinned by a team of industry veterans possessing robust technical knowledge in the Web3 domain, softstack offers industry-leading smart contract audit services. Committed to evolving with their clients' ever-changing business needs, softstack's approach is as dynamic and innovative as the industry it serves.

Check our website for further information: <https://chainsulting.de>

How We Work



1 -----

PREPARATION

Supply our team with audit ready code and additional materials



2 -----

COMMUNICATION

We setup a real-time communication tool of your choice or communicate via e-mails.



3 -----

AUDIT

We conduct the audit, suggesting fixes to all vulnerabilities and help you to improve.



4 -----

FIXES

Your development team applies fixes while consulting with our auditors on their safety.



5 -----

REPORT

We check the applied fixes and deliver a full report on all steps done.

