Global Rockstar

Token (ERC1155)

SMART CONTRACT AUDIT

25.11.2021

<u>Made in Germany by Chainsulting.de</u>

Table of contents

1. Disclaimer

The audit makes no statements or warrantees 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 Global Rockstar GmbH. 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 (02.11.2021)	Layout
0.5 (02.11.2021)	Verify Claims and Test Deployment
0.6 (03.11.2021)	Testing SWC Checks
0.8 (04.11.2021)	Automated Security Testing Manual Security Testing
0.9 (04.11.2021)	Summary and Recommendation
1.0 (07.11.2021)	Final Document

2. About the Project and Company

Company address:

Global Rockstar GmbH
Company ID: FN 413241 h
Rechte Wienzeile 37/3/3
1040 Vienna
Austria

Website: https://www.globalrockstar.com

Twitter: https://twitter.com/Global_Rockstar

Facebook: https://www.facebook.com/TheGlobalRockstar

Youtube: https://www.youtube.com/user/TheGlobalRockstar

Instagram: https://www.instagram.com/globalrockstarmusic

2.1 Project Overview

Global Rockstar was founded in 2014 to support independent players of the music business. They connect Artists with the best Producers and Writers. They help to get their productions financed, distributed and promoted worldwide.

Global Rockstar is not just “the artist friendly music label” but also an online platform enabling music fans to become shareholders of promising new hit songs!

Global Rockstar is a digital label and a music publishing company created to support artists and the best up-and-coming talents from all around the world.

Global Rockstar was founded by Christof Straub, a multi-platinum-awarded songwriter, music producer and serial entrepreneur from Vienna, Austria.

Global Rockstar presents investment opportunities on the platform globalrockstar.com to enable music fans to co-finance new music productions and their marketing. Music fans who invest in new songs purchase the right to receive future proceeds from the commercial exploitation of the recording (master rights) and/or the song (publishing rights) for the term of copyright (average 70 years in Europe).

Proceeds from master rights are generated every time the song is streamed, downloaded, featured in commercials or as a soundtrack of a movie. Proceeds from publishing rights are generated every time the song is played in the radio or live.

The Global Rockstar music label

- ensures music productions on the highest quality level
- distributes them worldwide in all relevant streaming- and download-stores
- promotes and markets them with a focus on streaming- and radio-promotion
- distributes the proceeds to all rights holders digitally on the platform

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 Chainsulting 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 Chainsulting 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.

4.2 Used Code from other Frameworks/Smart Contracts (direct imports)

Dependency / Import Path	Source
@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/blob/master/contracts/access/AccessControlUpgradeable.sol
@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/blob/master/contracts/proxy/utils/Initializable.sol
@openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/blob/master/contracts/proxy/utils/UUPSUpgradeable.sol
@openzeppelin/contracts-upgradeable/token/ERC1155/IERC1155ReceiverUpgradeable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/blob/master/contracts/token/ERC1155/IERC1155ReceiverUpgradeable.sol
@openzeppelin/contracts-upgradeable/token/ERC1155/IERC1155Upgradeable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/blob/master/contracts/token/ERC1155/IERC1155Upgradeable.sol
@openzeppelin/contracts-upgradeable/token/ERC1155/extensions/IERC1155MetadataURIUpgradeable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/blob/master/contracts/token/ERC1155/extensions/IERC1155MetadataURIUpgradeable.sol
@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/blob/master/contracts/utils/AddressUpgradeable.sol
@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/blob/master/contracts/utils/ContextUpgradeable.sol
@openzeppelin/contracts-upgradeable/utils/cryptography/draft-EIP712Upgradeable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/blob/master/contracts/utils/cryptography/draft-EIP712Upgradeable.sol

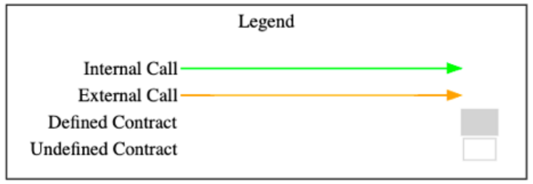
Dependency / Import Path	Source
@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/blob/master/contracts/utils/introspection/ERC165Upgradeable.sol
@openzeppelin/contracts/access/Ownable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/access/Ownable.sol
@openzeppelin/contracts/utils/Counters.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/utils/Counters.sol
@openzeppelin/contracts/utils/cryptography/ECDSA.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/utils/cryptography/ECDSA.sol

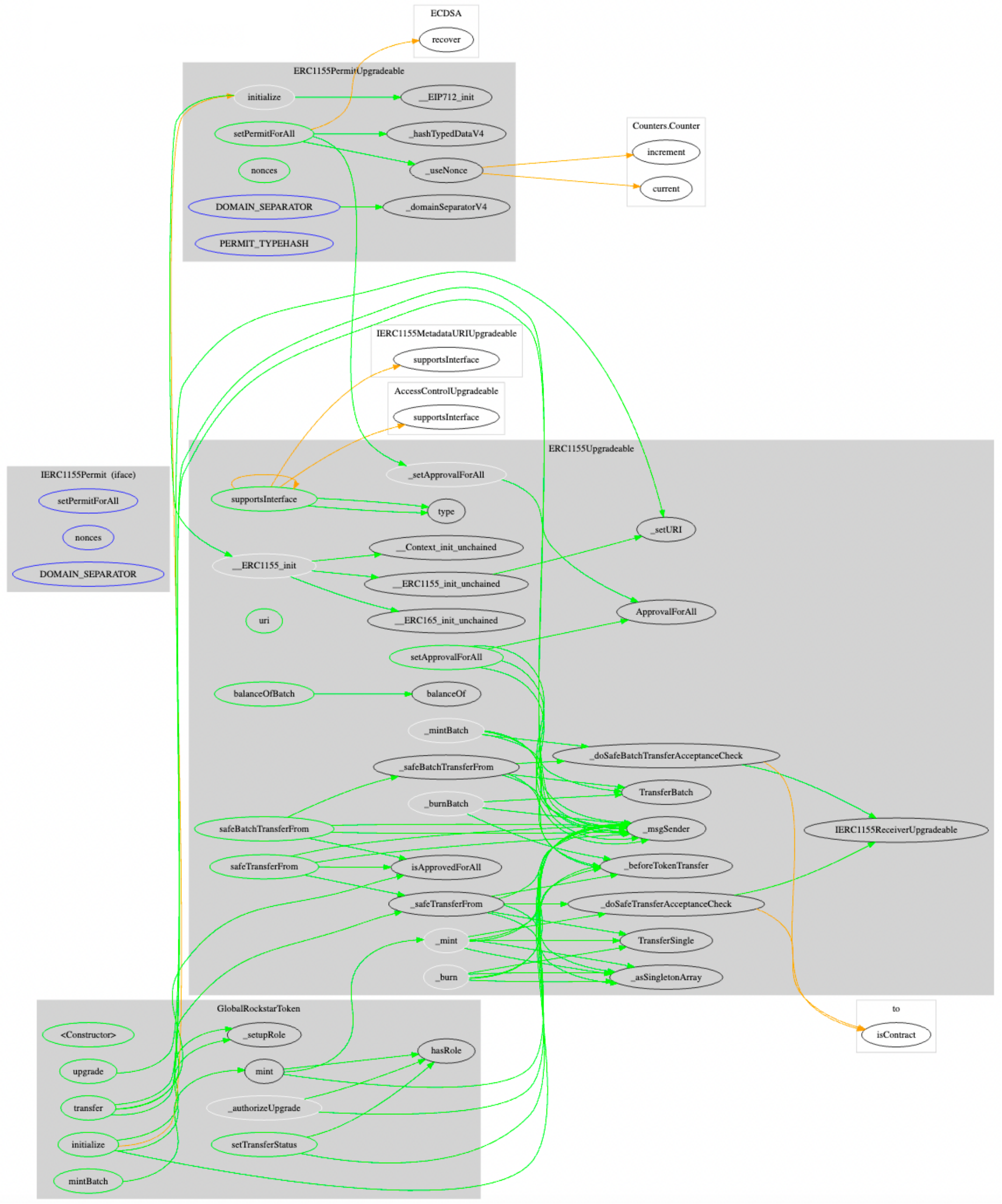
4.3 Tested Contract Files

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

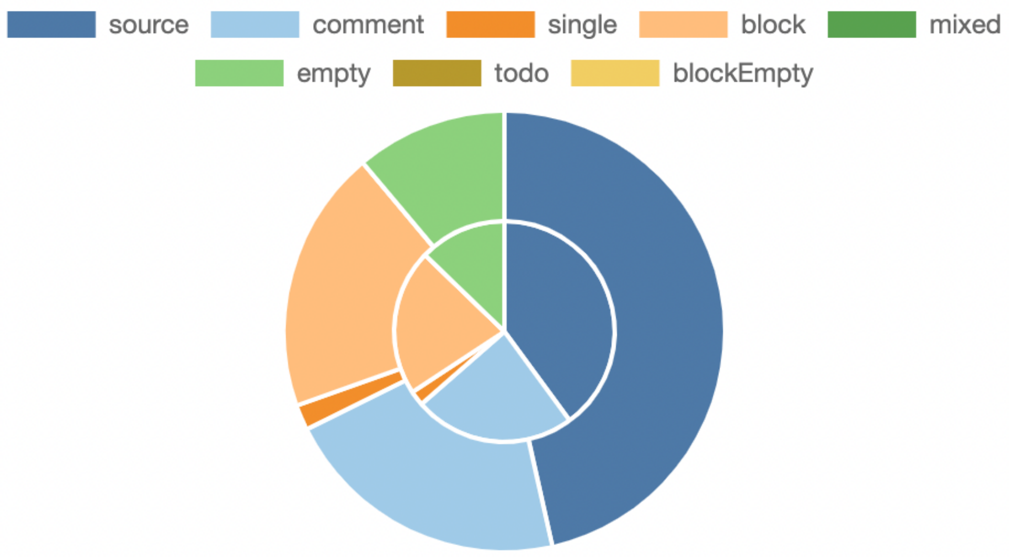
File	Fingerprint (MD5)
GlobalRockstarToken.sol	73f8ddfecd2f2d0f328671762b0dd7ce
ERC1155PermitUpgradable.sol	b7ceaf97291e9f41a6e48fd10ea5d1b5
ERC1155 Upgradable.sol	5e04e085c76f7b808930e13b035ee93c
IERC1155Permit.sol	3ffeb4e0e9a56a7670d83082109226c7

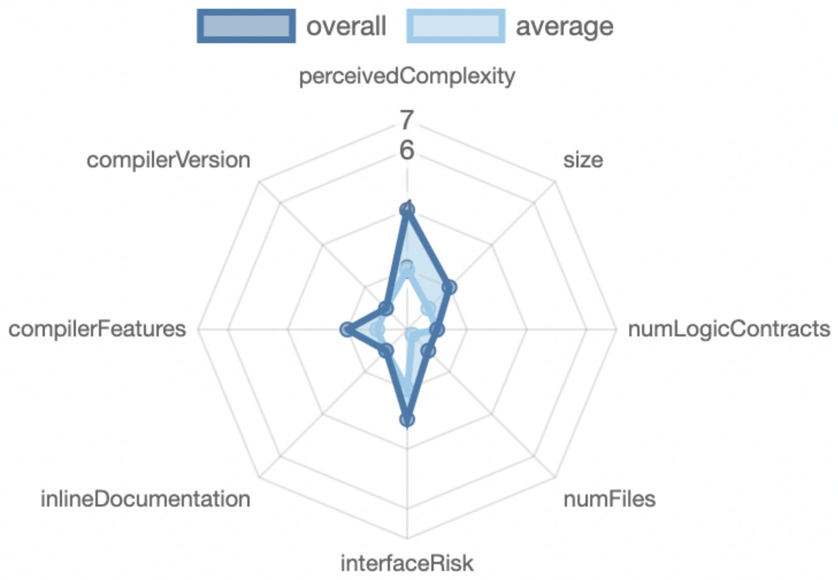
4.4 Metrics / CallGraph





4.5 Metrics / Source Lines & Risk





4.6 Metrics / Capabilities

Solidity Versions observed	🧪 Experimental Features	💰 Can Receive Funds	🖥 Uses Assembly	💣 Has Destroyable Contracts
`^0.8.0`			**** (0 asm blocks)	

📤 Transfers ETH	⚡ Low-Level Calls	👥 DelegateCall	🧮 Uses Hash Functions	🔖 ECRecover	🌀 New/Create/Create2
			`yes`		

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
22	0

External	Internal	Private	Pure	View
5	41	3	2	11

StateVariables

Total	🌐Public
10	1

4.7 Metrics / Source Unites in Scope

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝	ERC1155Upgradeable.sol	1		472	399	187	149	180	
📝	GlobalRockstarToken.sol	1		139	112	87	3	67	🧮
🔍	IERC1155Permit.sol		1	43	28	10	26	7	
🎨	ERC1155PermitUpgradeable.sol	1		112	94	47	28	34	🧮
📝🔍🎨	**Totals**	**3**	**1**	**766**	**633**	**331**	**206**	**288**	🧮

Legend: [➖]

- **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, ...)

5. Scope of Work

The Global Rockstar Team provided us with the files that needs to be tested. The scope of the audit is the ERC1155 Token contract.

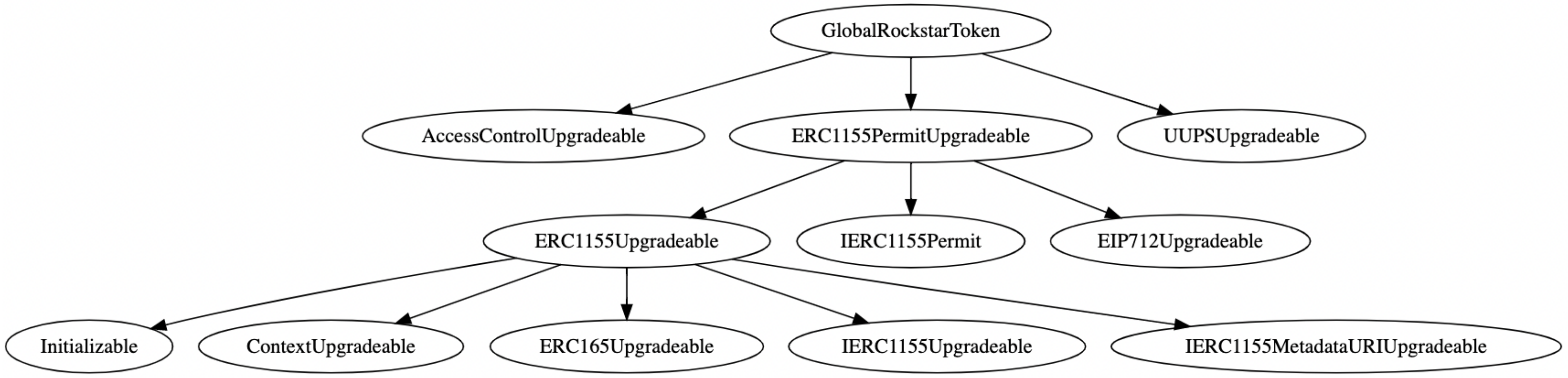
Following contracts with the direct imports has been tested:

- GlobalRockstarToken.sol

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

- The smart contract is coded according to the newest standards and in a secure way.
- The contract is using the ERC-1155 token standard
- Only the owner can mint new NFTs
- Owner cannot burn or lock user funds

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.



5.1 Manual and Automated Vulnerability Test

CRITICAL ISSUES

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

HIGH ISSUES

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

MEDIUM ISSUES

During the audit, Chainsulting’s experts found **no Medium issues** in the code of the smart contract.

LOW ISSUES

5.1.1 Missing natspec documentation

Severity: LOW
Status: ACKNOWLEDGED
Code: NA
File(s) affected: GlobalRockstarToken.sol

Attack / Description	Code Snippet	Result/Recommendation
Solidity contracts can use a special form of comments to provide rich documentation for functions, return variables and more. This special form is named the Ethereum Natural	NA	It is recommended to include natspec documentation and follow the doxygen style including @author, @title, @notice, @dev, @param, @return and make it easier to review and understand your smart contract.

Language Specification Format (NatSpec).		

INFORMATIONAL ISSUES

5.1.2 A floating pragma is set.

Severity: INFORMATIONAL
Status: ACKNOWLEDGED
Code: SWC-103
File(s) affected: ALL

Attack / Description	Code Snippet	Result/Recommendation
The current pragma Solidity directive is "^0.8.0". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.	Line 1: `pragma solidity ^0.8.0;`	It is recommended to follow the latter example, as future compiler versions may handle certain language constructions in a way the developer did not foresee. i.e. Pragma solidity 0.8.0 See SWC-103: https://swcregistry.io/docs/SWC-103

5.1.3 Storing data via URI

Severity: INFORMATIONAL
Status: ACKNOWLEDGED
Code: NA
File(s) affected: GlobalRockstarToken.sol

Attack / Description	Code Snippet	Result/Recommendation

In the current implementation the baseURI is not hardcoded, means the owner/creator is free to choose the way how the metadata file is stored.	Line 42 - 44
	`function upgrade(string memory _baseURI) public {`
	`    _setURI(_baseURI);`
	`}`
		We recommend using IPFS and pinning services to make the metadata behind the baseURI permanently stored.

5.2. 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	✓
SWC-129	Typographical Error	CWE-480: Use of Incorrect Operator	✓

ID	Title	Relationships	Test Result
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	✅

ID	Title	Relationships	Test Result
SWC-108	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	✅
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	X
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	✅

5.3. Verify Claims

5.3.1 The smart contract is coded according to the newest standards and in a secure way.
Status: tested and verified ✅

5.3.2 The contract is using the ERC-1155 token standard
Status: tested and verified ✅

5.3.3 Only the owner can mint new NFTs
Status: tested and verified ✅
Only MINTER_ROLE is able to mint new token until the hardcoded set cap.

```
function mint(
  address _to,
  uint256 _id,
  uint256 _amount
) public {
  require(
      hasRole(MINTER_ROLE, _msgSender()),
      "ERC1155: must have minter role to mint"
  );
  uint256 currentTokenSupply = _tokensSupply[_id];
  uint256 newTokenSupply = currentTokenSupply + _amount;
  require(
      newTokenSupply <= _supplyLimitForSingleToken,
      "ERC1155: cannot mint more than the set cap"
  );

  _tokensSupply[_id] = newTokenSupply;
```

```
        _mint(_to, _id, _amount, "");
    }
```

5.3.4 Owner cannot burn or lock user funds

Status: tested and verified ✅

Funds cannot be burned or locked but the transfer can be disabled

```
bool private _isTransferEnabled;
event TransferEnabledStatusChanged(bool indexed isTransferEnabled);
```

6. Executive Summary

Two (2) independent Chainsulting experts performed an unbiased and isolated audit of the smart contract codebase. The final debriefs took place on the November 07, 2021.

The main goal of the audit was to verify the claims regarding the security of the smart contract and the functions. During the audit, no critical issues were found after the manual and automated security testing and the claims been successfully verified.