

# Auditor Labs

**Curio Cards**  
**SMART CONTRACT AUDIT**

Made by [auditorlabs.com](https://auditorlabs.com)

## Table of contents

1. Disclaimer .....	3
2. About the Project and Company .....	4
2.1 Project Overview .....	5
3. Vulnerability & Risk Level.....	6
4. Auditing Strategy and Techniques Applied .....	7
4.1 Methodology.....	7
4.2 Used Code from other Frameworks/Smart Contracts .....	8
4.3 Tested Contract Files.....	9
4.4 Metrics / CallGraph .....	10
4.5 Metrics / Source Lines & Risk .....	11
4.6 Metrics / Capabilities.....	12
5. Scope of Work.....	14
5.1 Manual and Automated Vulnerability Test .....	15
5.1.1 Big number of tokens are still hold by owner.....	15
5.1.2 Wrong import of OpenZeppelin library.....	16
5.1.3 Storing metadata via IPFS.....	17
5.2. SWC Attacks .....	21
5.3. Verify Claims .....	25
6. Executive Summary .....	26
7. Deployed Smart Contract.....	26

## 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 Curio Cards. 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 (22.08.2021)	Layout
0.2 (25.08.2021)	Test Deployment
0.5 (28.08.2021)	Automated Security Testing Manual Security Testing
0.6 (10.09.2021)	Testing SWC Checks
0.7 (15.09.2021)	Verify Claims
0.9 (25.09.2021)	Summary and Recommendation
1.0 (30.09.2021)	Final document
1.1 (30.09.2021)	Adding deployed contract address

## 2. About the Project and Company

### Company address:

Founder: Thomas Hunt

<https://www.madbitcoins.com>

<https://www.linkedin.com/in/thomashuntbitcoin>

San Francisco, California

United States of America



**Website:** <https://curio.cards>

**Twitter:** <https://twitter.com/MyCurioCards>

**Discord:** <https://discord.curio.cards>

**OpenSea:** <https://opensea.io/collection/curiocardswrapper>

**Documentation:** <https://docs.curio.cards>

**Governance:** <https://governance.curio.cards/#/>

## 2.1 Project Overview

Curio Cards is the first Ethereum NFT Collectibles dating back to 2017. Thomas Hunt, aka Crypto Personality “Mad Bitcoins” created the cards.

Curio Cards is an online art show and permanent gallery that launched on May 9, 2017. The goal of the project was to use a distributed network called Ethereum to create a new model for digital artwork ownership -- a model that allows for the sale and collection of unique digital artwork without taking a cut of the artist's revenue. Today, it is the oldest known example on Ethereum of what is now called NFT Artwork (NFT = Non-Fungible Token) and features 30 unique series of NFT cards from 7 different artists.

Curio Cards behave like "digital prints", with each series of artwork being unique but containing multiple cards in that series. Curio Cards was referenced in the ERC-721 EIP, having been developed and released before the ERC-721 NFT standard was proposed. Curio Cards contains many features and concepts that were included in ERC-721 and other NFT standards, such as:

- Non-fungible tokens (between the different card series, aka 'card number')
- Non-divisible tokens with a limited supply (within each card series)
- Purchasing initial cards directly from a smart contract
- IPFS hash of the artwork embedded into the smart contract (without referencing any urls)
- Ownership of a token representing ownership of publicly viewable digital artwork

The original cards still work and are transferable, however since they pre-date modern NFT standards they do not conform to those standards. Instead, Curio Cards are a heavily modified ERC-20 token. A wrapper was developed that lets users take their ERC-20 Curio Cards and "wrap" them inside an ERC-1155 token contract, for use on modern NFT marketplaces like OpenSea. ERC-1155 is a well supported and flexible modern NFT standard that maps closely with Curio's original design.

No new cards will ever be released. It was originally planned to open Curio Cards up for anyone to make new cards sets and collections, but modern contract standards offer a better choice for new artists. Learn more about how Curio Cards will evolve over time through the Governance process.

### 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 Auditor Labs 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 Auditor Labs 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
ERC1155	<a href="https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.1.0/contracts/token/ERC1155/ERC1155.sol">https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.1.0/contracts/token/ERC1155/ERC1155.sol</a>
IERC1155	<a href="https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.1.0/contracts/token/ERC1155/IERC1155.sol">https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.1.0/contracts/token/ERC1155/IERC1155.sol</a>
ERC165	<a href="https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.1.0/contracts/introspection/ERC165.sol">https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.1.0/contracts/introspection/ERC165.sol</a>
IERC1155TokenReceiver	<a href="https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.1.0/contracts/token/ERC1155/IERC1155Receiver.sol">https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.1.0/contracts/token/ERC1155/IERC1155Receiver.sol</a>
IERC1155Metadata	<a href="https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.1.0/contracts/token/ERC1155/IERC1155MetadataURI.sol">https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.1.0/contracts/token/ERC1155/IERC1155MetadataURI.sol</a>
SafeMath	<a href="https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.1.0/contracts/math/SafeMath.sol">https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.1.0/contracts/math/SafeMath.sol</a>
Address	<a href="https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.1.0/contracts/utils/Address.sol">https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.1.0/contracts/utils/Address.sol</a>



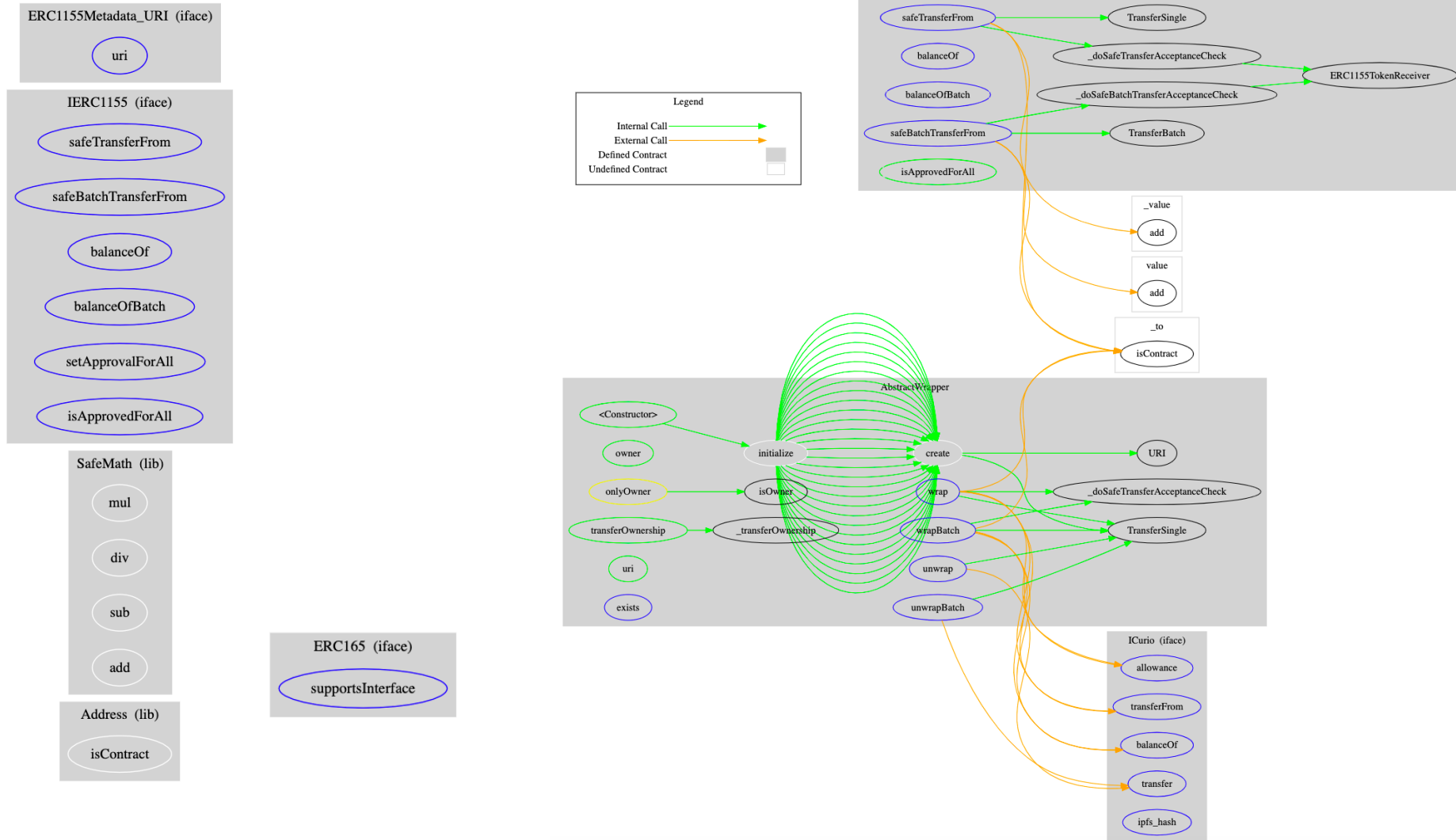
## 4.3 Tested Contract Files

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)
./CardToken.sol	9546acf6808b32821c969cd1e243723e

File	Fingerprint (MD5)
./AbstractWrapper.sol	f19be47ea320b06db1e60b585bb3c86f
./CurioERC1155Wrapper.sol	78bfa699efd524ed4f1d7ab9c1cf57a1
./ICurio.sol	90c12f204b09fe68cd86c54dccde55bd

## 4.4 Metrics / CallGraph

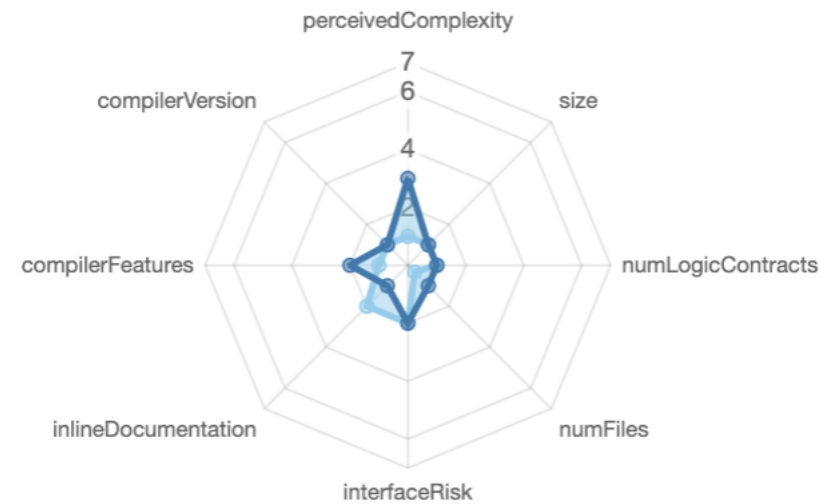


## 4.5 Metrics / Source Lines & Risk











source comment single block mixed  
empty todo blockEmpty



overall average





4.6 Metrics / Capabilities


Solidity Versions observed		 Experimental Features		 Can Receive Funds		 Uses Assembly		 Has Destroyable Contracts			
<div>^0.5.0</div>				<div></div>		**** (0 asm blocks)		<div></div>			
 Transfers ETH		 Low-Level Calls		 DelegateCall		 Uses Hash Functions		 ECTrecover		 New/Create/Create2	
<div>yes</div>		<div></div>		<div></div>		<div></div>		<div></div>			

Exposed Functions







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

 <b>Public</b>	 <b>Payable</b>			
16	0			
<b>External</b>	<b>Internal</b>	<b>Private</b>	<b>Pure</b>	<b>View</b>
10	17	0	0	7

StateVariables

<b>Total</b>	 <b>Public</b>
4	3

## 4.7 Metrics / Source Unites in Scope

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
	contracts/AbstractWrapper.sol	1	_____	208	203	106	55	103	
	contracts/ICurio.sol	_____	1	10	5	3	1	11	_____
	contracts/CurioERC1155Wrapper.sol	1	_____	43	43	38	30	36	_____
	<b>Totals</b>	<b>2</b>	<b>1</b>	<b>261</b>	<b>251</b>	<b>147</b>	<b>86</b>	<b>150</b>	

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, ...)

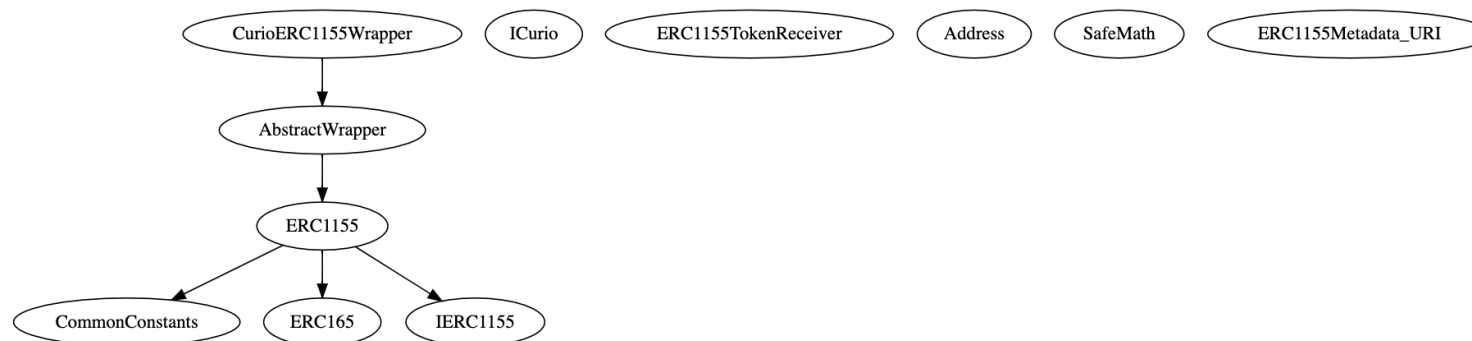
## 5. Scope of Work

Cause of the public attraction and historical value of the project, we started an audit, to make sure contracts are safe to use. The scope of the audit is the Curio Cards wrapper for ERC1155 and the original Curio Cards contract.

Auditor Labs put forward the following assumptions regarding the security, usage of the contracts:

- The wrapper contract is using the ERC1155 token standard
- Owner cannot mint any new tokens (original and wrapper contract)
- Owner cannot burn or lock user funds (original and wrapper contract)
- Owner cannot pause the contract (original and wrapper contract)
- IPFS hashes are implemented in the correct way
- The smart contract is coded according to the newest standards and in a secure way.

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, Auditor Labs experts found **no Critical issues** in the code of the smart contract.

### HIGH ISSUES

During the audit, Auditor Labs experts found **no High issues** in the code of the smart contract.

### MEDIUM ISSUES

#### 5.1.1 Big number of tokens are still hold by owner

Severity: MEDIUM

Status: **FIXED**

File(s) affected: NA

Attack / Description	Code Snippet	Result/Recommendation
The creator and owner of the contract is still holding a decent number of tokens by themselves. Which can cause by a hack or dump a decrease in price on the market.	<a href="https://etherscan.io/address/0x3cc44273a97e8fbfcbcd3d60200cc9fd33d84d66">https://etherscan.io/address/0x3cc44273a97e8fbfcbcd3d60200cc9fd33d84d66</a>  Last activity 37 days ago (30. Sep. 2021)	Seems like etherscan.io has not properly indexed that asset and checking via <a href="https://etherscan.io/address/0x72b34d637c0d14ace58359ef1bf472e4b4c57125#readContract">https://etherscan.io/address/0x72b34d637c0d14ace58359ef1bf472e4b4c57125#readContract</a>  <a href="https://etherscan.io/address/0x2fce2713a561bb019bc5a110be0a19d10581ee9e#readContract">https://etherscan.io/address/0x2fce2713a561bb019bc5a110be0a19d10581ee9e#readContract</a>

		<a href="https://etherscan.io/address/0xbf4cc966f1e726087c5c55aac374e687000d4d45#readContract">https://etherscan.io/address/0xbf4cc966f1e726087c5c55aac374e687000d4d45#readContract</a>  BalanceOf is giving back 0
--	--	---

## LOW ISSUES

### 5.1.2 Wrong import of OpenZeppelin library

Severity: LOW

Status: ACKNOWLEDGED

File(s) affected: All

Attack / Description	Code Snippet	Result/Recommendation
In the current implementation, OpenZeppelin files are added directly into the code. This violates OpenZeppelin's MIT license, which requires the license and copyright notice to be included if its code is used. Moreover, updating code manually is error-prone.	SafeMath, IERC1155, IERC1155Metadata, IERC1155TokenReceiver, IERC1155, ERC1155, ERC165, Common, Address	We highly recommend using npm (import "@openzeppelin/contracts/..") in order to guarantee that original OpenZeppelin contracts are used with no modifications. This also allows for any bug-fixes to be easily integrated into the codebase.



## INFORMATIONAL ISSUES

### 5.1.3 Storing metadata via IPFS

Severity: INFORMATIONAL

Status: ACKNOWLEDGED

Code: NA

File(s) affected: CurioERC1155Wrapper.sol

Attack / Description	Code Snippet	Result/Recommendation
In the current implementation the IPFS addresses are hardcoded into the CurioERC1155Wrapper contract. To ensure that data persists on IPFS, and is not deleted during garbage collection, data can be pinned to one or more IPFS nodes. Pinning gives you control over disk space and data retention. As such, you should use that control to pin any content you wish to keep on IPFS indefinitely.	<pre>function initialize() internal {     create(1, 0x6Aa2044C7A0f9e2758EdAE97247B03a0D7e73d6c, "ipfs://QmWHUnrdfA4w89TeepZqrvygbaf9wV48k97Wf27skL5cry ");     create(2, 0xE9A6A26598B05dB855483fF5eCc5f1d0C81140c8, "ipfs://QmVJn6B289Xt3cq9evzubdyk4f1usPAu277SmUusmdYYWU ");     create(3, 0x3f8131B6E62472CEea9cb8Aa67d87425248a3702, "ipfs://QmWBb6T4nviPWdAygGJTki7VA6fpTmcYP37U9jpYAfHzPP ");     create(4, 0x4F1694be039e447B729ab11653304232Ae143C69, "ipfs://Qmbcw8ix8xdK1reFpDEjKtk9EWuRwrBMKqvEvWkttNzXkH ");     create(5, 0x5a3D4A8575a688b53E8b270b5C1f26fd63065219, "ipfs://QmXmj9YdsvBVddzC352Xsh7bmyJtfZvbVJeetK7PXW21p8 ");</pre>	<p>We recommend using IPFS pinning services to make the metadata permanently stored.</p> <p>ipfs://QmXafwRpoJPiiQ9TZihhbSsFmgKqKMqrHSRLkp1wyQ3jUU</p> <p>Check more information here: <a href="https://docs.ipfs.io/concepts/persistence/#persistence-versus-permanence">https://docs.ipfs.io/concepts/persistence/#persistence-versus-permanence</a></p>

	<pre>         create(6, 0x1Ca6AC0Ce771094F0F8a383D46BF3acC9a5BF27f, "ipfs://Qmdf16YMPM7zG5QkSYB4HjbxQPasYazsL6d1npdJG8J7h ");          create(7, 0x2647bd8777e0C66819D74aB3479372eA690912c3, "ipfs://QmUGmWwrNR7JKBCSu3CkGnTYSFat7y2AiUzACcbAoZcj2d ");          create(8, 0x2FCE2713a561bB019BC5A110BE0A19d10581ee9e, "ipfs://QmXQfBgJRsuQbf8UkViATdpsySXzREsifegWzLvw5QsQPj ");          create(9, 0xbf4Cc966F1e726087c5C55aac374E687000d4d45, "ipfs://Qmctv89ppbYTuWCWFA9waVCeE8g6YM3Ah54bZW1WGmEHh ");          create(10, 0x72b34d637C0d14acE58359Ef1bF472E4b4c57125, "ipfs://QmaSBVrCcBsYHjVuvTsj6ev4Pua7NYX7sDNzdAYwCdAAne ");          create(11, 0xb36c87F1f1539c5FC6f6e7b1C632e1840C9B66b4, "ipfs://QmZjSs71uBYyDLx5Ju443KiSYjxQcJQLL5ZnhuzWX6nC19 ");          create(12, 0xD15af10A258432e7227367499E785C3532b50271, "ipfs://QmQqMKDMKiRhgbFBrmAJPKnzYHEKuH7VrqPZ7NS5vFoy78 ");          create(13, 0x2d922712f5e99428c65b44f09Ea389373d185bB3, "ipfs://QmeShnRPe6uiRcBy81nQXDZ9TWUpFNQfiAThf9ruAQGcRa "); </pre>	
--	---	--









	<pre> create(14, 0x0565ac44e5119a3224b897De761a46A92aA28ae8, "ipfs://Qmdi8vQuQQWksIM5HCCVXfzSzcaemzQwYkUe4Tb94DP6vK ");  create(15, 0xdb7F262237Ad8acca8922aA2c693a34D0d13e8fe, "ipfs://QmS3UF256kWHbX8Wi7CYExyCxzLNx1nsaMwpaGBN73rr31 ");  create(16, 0x1b63532CcB1FeE0595c7fe2Cb35cFD70ddF862Cd, "ipfs://Qmbj1YcmQidTzvgjLmu1b99PPdXZLSgk72YZQSt9LEe1R ");  create(17, 0xF59536290906F204C3c7918D40C1Cc5f99643d0B, "ipfs://QmbDsZABRUPMcuoFWePRH7YiGyR64udWHc4u1mQPJYmB2c ");  create(18, 0xA507D9d28bbca54cBCfFad4BB770C2EA0519F4F0, "ipfs://QmXafwRpoJPiiQ9TZihhbSsFmgKqKMqrHSRLkp1wyQ3jUU ");  create(19, 0xf26BC97Aa8AFE176e275Cf3b08c363f09De371fA, "ipfs://QmTWJR1XJ2svexE2NT3A6cCtks8rgh6TKYaLYXwfHapNDN ");  create(20, 0xD0ec99E99cE22f2487283A087614AEe37F6B1283, "ipfs://Qmd3HzUX52MmZcj1Se3ocgYWEJWSvzSceEqQFV1YL7LRWL ");  create(21, 0xB7A5a84Ff90e8Ef91250fB56c50a7bB92a6306EE, "ipfs://QmX6stsihT3SNUakiFQLWU1cjvH7rC3pqtCnToxNn2T8JS "); </pre>	
--	--	--

	<pre> create(22, 0x148fF761D16632da89F3D30eF3dFE34bc50CA765, "ipfs://Qmc1sj8LRdfbPinoqKMmAe6UvJUG33VMmSU3XzNK2GnjJB ");  create(23, 0xCDE7185B5C3Ed9eA68605a960F6653AA1a5b5C6C, "ipfs://Qmdwh3S4imtE5RxZ4ddAzy3DMqNrD11JL6SATTyREuvrtN ");  create(24, 0xE67dad99c44547B54367E3e60fc251fC45a145C6, "ipfs://QmbfTxH6XvbgGcyWWaygmPko6NQ6tKuT6dJj5WjnQGp5g8 ");  create(25, 0xC7f60C2b1DBDfd511685501EDEb05C4194D67018, "ipfs://QmXHyK19F4sMAUi6XYZ1BJJYzxsdp8koVnL4BwsFA93Q47 ");  create(26, 0x1cB5BF4Be53eb141B56f7E4Bb36345a353B5488c, "ipfs://QmYK88qy84rcL46CZGPqpKRm4fE2PQYJ931pV69ZNi4J1D ");  create(27, 0xFb9F3fa2502d01d43167A0A6E80bE03171DF407E, "ipfs://QmcUTEkPpmRPHCHiXskd9daQcEZwGzkHgybZmCWmFYha1T ");  create(28, 0x59D190e8A2583C67E62eEc8dA5EA7f050d8BF27e, "ipfs://QmTmi8j5BBE5FWhEDAg1bTqpmkkEcaPgTUeYFJ4z3PxXqN ");  create(29, 0xD3540bCD9c2819771F9D765Edc189cBD915FEAbd, "ipfs://QmVTGJtgnUgnMPttJV2VkfonCUYLRnJqX66gJLiig5QVgC "); </pre>	
--	---	--






	<pre> create(30, 0x7F5B230Dc580d1e67DF6eD30dEe82684dD113D1F, "ipfs://QmQBu8jYC3vEGzx59BUW4knBdNRyFd8aTVLLFCEprdjZ5e "); } </pre>	
--	--	--

## 5.2. SWC Attacks

ID	Title	Relationships	Test Result
<a href="#">SWC-131</a>	Presence of unused variables	<a href="#">CWE-1164: Irrelevant Code</a>	✓
<a href="#">SWC-130</a>	Right-To-Left-Override control character (U+202E)	<a href="#">CWE-451: User Interface (UI) Misrepresentation of Critical Information</a>	✓
<a href="#">SWC-129</a>	Typographical Error	<a href="#">CWE-480: Use of Incorrect Operator</a>	✓
<a href="#">SWC-128</a>	DoS With Block Gas Limit	<a href="#">CWE-400: Uncontrolled Resource Consumption</a>	✓
<a href="#">SWC-127</a>	Arbitrary Jump with Function Type Variable	<a href="#">CWE-695: Use of Low-Level Functionality</a>	✓

ID	Title	Relationships	Test Result
<a href="#">SWC-125</a>	Incorrect Inheritance Order	<a href="#">CWE-696: Incorrect Behavior Order</a>	
<a href="#">SWC-124</a>	Write to Arbitrary Storage Location	<a href="#">CWE-123: Write-what-where Condition</a>	
<a href="#">SWC-123</a>	Requirement Violation	<a href="#">CWE-573: Improper Following of Specification by Caller</a>	
<a href="#">SWC-122</a>	Lack of Proper Signature Verification	<a href="#">CWE-345: Insufficient Verification of Data Authenticity</a>	
<a href="#">SWC-121</a>	Missing Protection against Signature Replay Attacks	<a href="#">CWE-347: Improper Verification of Cryptographic Signature</a>	
<a href="#">SWC-120</a>	Weak Sources of Randomness from Chain Attributes	<a href="#">CWE-330: Use of Insufficiently Random Values</a>	
<a href="#">SWC-119</a>	Shadowing State Variables	<a href="#">CWE-710: Improper Adherence to Coding Standards</a>	
<a href="#">SWC-118</a>	Incorrect Constructor Name	<a href="#">CWE-665: Improper Initialization</a>	
<a href="#">SWC-117</a>	Signature Malleability	<a href="#">CWE-347: Improper Verification of Cryptographic Signature</a>	


ID	Title	Relationships	Test Result
<a href="#">SWC-116</a>	Timestamp Dependence	<a href="#">CWE-829: Inclusion of Functionality from Untrusted Control Sphere</a>	✓
<a href="#">SWC-115</a>	Authorization through tx.origin	<a href="#">CWE-477: Use of Obsolete Function</a>	✓
<a href="#">SWC-114</a>	Transaction Order Dependence	<a href="#">CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')</a>	✓
<a href="#">SWC-113</a>	DoS with Failed Call	<a href="#">CWE-703: Improper Check or Handling of Exceptional Conditions</a>	✓
<a href="#">SWC-112</a>	Delegatecall to Untrusted Callee	<a href="#">CWE-829: Inclusion of Functionality from Untrusted Control Sphere</a>	✓
<a href="#">SWC-111</a>	Use of Deprecated Solidity Functions	<a href="#">CWE-477: Use of Obsolete Function</a>	✓
<a href="#">SWC-110</a>	Assert Violation	<a href="#">CWE-670: Always-Incorrect Control Flow Implementation</a>	✓
<a href="#">SWC-109</a>	Uninitialized Storage Pointer	<a href="#">CWE-824: Access of Uninitialized Pointer</a>	✓
<a href="#">SWC-108</a>	State Variable Default Visibility	<a href="#">CWE-710: Improper Adherence to Coding Standards</a>	✓
<a href="#">SWC-107</a>	Reentrancy	<a href="#">CWE-841: Improper Enforcement of Behavioral Workflow</a>	✓

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




## 5.3. Verify Claims


5.3.1 The wrapper contract is using the ERC1155 token standard

**Status:** tested and verified 


5.3.2 Owner cannot mint any new tokens (original and wrapper contract)

**Status:** tested and verified 


5.3.3 Owner cannot burn or lock user funds (original and wrapper contract)

**Status:** tested and verified 


5.3.4 Owner cannot pause the contract (original and wrapper contract)

**Status:** tested and verified 

5.3.5 IPFS hashes are implemented in the correct way

**Status:** tested and verified 

5.3.6 The smart contract is coded according to the newest standards and in a secure way.

**Status:** tested and verified 

## 6. Executive Summary

Two (2) independent Auditor Labs experts performed an unbiased and isolated audit of the smart contract codebase. The final debriefs took place on the September 30, 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.

## 7. Deployed Smart Contract

VERIFIED

### Wrapper

<https://etherscan.io/address/0x73da73ef3a6982109c4d5bdb0db9dd3e3783f313#code>

### Original contracts

Card1: <https://etherscan.io/address/0x6Aa2044C7A0f9e2758EdAE97247B03a0D7e73d6c#code>

Card2: <https://etherscan.io/address/0xE9A6A26598B05dB855483fF5eCc5f1d0C81140c8#code>

Card3: <https://etherscan.io/address/0x3f8131B6E62472CEea9cb8Aa67d87425248a3702#code>

Card4: <https://etherscan.io/address/0x4F1694be039e447B729ab11653304232Ae143C69#code>

Card5: <https://etherscan.io/address/0x5a3D4A8575a688b53E8b270b5C1f26fd63065219#code>

Card6: <https://etherscan.io/address/0x1Ca6AC0Ce771094F0F8a383D46BF3acC9a5BF27f#code>

Card7: <https://etherscan.io/address/0x2647bd8777e0C66819D74aB3479372eA690912c3#code>

Card8: <https://etherscan.io/address/0x2FCE2713a561bB019BC5A110BE0A19d10581ee9e#code>

Card9: <https://etherscan.io/address/0xbf4Cc966F1e726087c5C55aac374E687000d4d45#code>

Card10: <https://etherscan.io/address/0x72b34d637C0d14acE58359Ef1bF472E4b4c57125#code>

Card11: <https://etherscan.io/address/0xb36c87F1f1539c5FC6f6e7b1C632e1840C9B66b4#code>  
Card12: <https://etherscan.io/address/0xD15af10A258432e7227367499E785C3532b50271#code>  
Card13: <https://etherscan.io/address/0x2d922712f5e99428c65b44f09Ea389373d185bB3#code>  
Card14: <https://etherscan.io/address/0x0565ac44e5119a3224b897De761a46A92aA28ae8#code>  
Card15: <https://etherscan.io/address/0xdb7F262237Ad8acca8922aA2c693a34D0d13e8fe#code>  
Card16: <https://etherscan.io/address/0x1b63532CcB1FeE0595c7fe2Cb35cFD70ddF862Cd#code>  
Card17: <https://etherscan.io/address/0xF59536290906F204C3c7918D40C1Cc5f99643d0B#code>  
Card18: <https://etherscan.io/address/0xA507D9d28bbca54cBCfFad4BB770C2EA0519F4F0#code>  
Card19: <https://etherscan.io/address/0xf26BC97Aa8AFE176e275Cf3b08c363f09De371fA#code>  
Card20: <https://etherscan.io/address/0xD0ec99E99cE22f2487283A087614AEe37F6B1283#code>  
Card21: <https://etherscan.io/address/0xB7A5a84Ff90e8Ef91250fB56c50a7bB92a6306EE#code>  
Card22: <https://etherscan.io/address/0x148fF761D16632da89F3D30eF3dFE34bc50CA765#code>  
Card23: <https://etherscan.io/address/0xCDE7185B5C3Ed9eA68605a960F6653AA1a5b5C6C#code>  
Card24: <https://etherscan.io/address/0xE67dad99c44547B54367E3e60fc251fC45a145C6#code>  
Card25: <https://etherscan.io/address/0xC7f60C2b1DBDfd511685501EDEb05C4194D67018#code>  
Card26: <https://etherscan.io/address/0x1cB5BF4Be53eb141B56f7E4Bb36345a353B5488c#code>  
Card27: <https://etherscan.io/address/0xfb9f3fa2502d01d43167a0a6e80be03171df407e#code>  
Card28: <https://etherscan.io/address/0x59D190e8A2583C67E62eEc8dA5EA7f050d8BF27e#code>  
Card29: <https://etherscan.io/address/0xD3540bCD9c2819771F9D765Edc189cBD915FEAbd#code>  
Card30: <https://etherscan.io/address/0x7f5b230dc580d1e67df6ed30dee82684dd113d1f#code>