SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

v1.0: 06. November, 2021
v1.1: 10. November, 2021

Audit

Security Assessment
15. December, 2021

For

METAWALLS
Berlin

Disclaimer

SolidProof.io reports are not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team. SolidProof.io do not cover testing or auditing the integration with external contract or services (such as Unicrypt, Uniswap, PancakeSwap etc’...)

SolidProof.io Audits do not provide any warranty or guarantee regarding the absolute bug- free nature of the technology analyzed, nor do they provide any indication of the technology proprietors. SolidProof Audits should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

SolidProof.io Reports represent an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology. Blockchain technology and cryptographic assets present a high level of ongoing risk. SolidProof’s position is that each company and individual are responsible for their own due diligence and continuous security. SolidProof in no way claims any guarantee of security or functionality of the technology we agree to analyze.

Version	Date	Description
1.0	06. November 2021	• Layout project • Automated- /Manual-Security Testing • Summary
1.1	10. November 2021	• Reaudit
2.0	15. December 2021	• New contracts

Network
Polygon POS Chain

Website
https://www.metawalls.io/

Twitter
https://twitter.com/Metawalls_bln

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

Youtube
https://www.youtube.com/channel/UCmCq7eDzGwRy8H7H_19VakA

Discord
https://discord.gg/y6qQZhV5rv

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

Description

METAWALLS is the new NFT platform inspired by, and built for Berlin's artists. METAWALLS is bringing Berlin's vibrant Street Art culture into the Metaverse and the world of blockchain.

METAWALLS wants to help reform the art market, and this means for us facilitating more equality, participation, and empowerment for artists and art buyers alike through our revolutionary CO-NFT (Collective-Ownership NFT) platform.

The unique design and features of CO-NFT will empower a global community of art lovers to support and engage directly with Berlin's unique Street Art culture in ways not previously possible.
METAWALLS runs on Polygon, an energy-efficient 'Proof of Stake' blockchain. Polygon provides users with secure, low cost transactions 24/7.

Project Engagement

During the 03rd of November 2021, **MetaWalls Team** engaged Solidproof.io to audit smart contracts that they created. The engagement was technical in nature and focused on identifying security flaws in the design and implementation of the contracts. They provided Solidproof.io with access to their code repository and whitepaper.

Logo

Contract Link

v1.0/1.1/2.0

TBA

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 aspossible.
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

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.

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 SolidProof 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 SolidProof 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.

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

v1.0/1.1

Imported packages:

Dependency / Import Path	Count
hardhat/console.sol	1
openzeppelin-solidity/contracts/access/AccessControl.sol	1
openzeppelin-solidity/contracts/access/Ownable.sol	2
openzeppelin-solidity/contracts/token/ERC1155/ERC1155.sol	1
openzeppelin-solidity/contracts/utils/Counters.sol	1
openzeppelin-solidity/contracts/utils/Strings.sol	1
openzeppelin-solidity/contracts/utils/math/SafeMath.sol	2

v2.0

Imported packages:

Dependency / Import Path	Count
@openzeppelin/contracts/access/AccessControl.sol	1
@openzeppelin/contracts/access/Ownable.sol	2
@openzeppelin/contracts/token/ERC721/ERC721.sol	1
@openzeppelin/contracts/token/ERC721/extensions/ERC721Burnable.sol	1
@openzeppelin/contracts/token/ERC721/extensions/ERC721Enumerable.sol	1
@openzeppelin/contracts/utils/Context.sol	1
@openzeppelin/contracts/utils/cryptography/ECDSA.sol	1
@openzeppelin/contracts/utils/math/SafeMath.sol	1

Tested Contract Files

This audit covered the following files listed below with a SHA-1 Hash.

A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

v1.0

File Name	SHA-1 Hash
contracts/ERC1155Tradable.sol	573c782884d204a80b5b9f82c927d8fd2433480b
contracts/MetaWallsCoin.sol	1a1a5a673f63e64281302250b49f11579faa67af
contracts/common/grid.sol	bcb42923d48dd116fb35e5f1b9e02bb1ab5a67cd
contracts/common/meta-transactions/NativeMetaTransaction.sol	5200bcd3d499fc3c479324f952993759e8e8664e
contracts/common/meta-transactions/ContentMixin.sol	8b7d3ce9306b7dc5f48d895e25b5bee4748e1bf6
contracts/common/meta-transactions/EIP712Base.sol	5c67dc2114a8a2e9ce81e818e6c27559b0161818
contracts/common/meta-transactions/Initializable.sol	34d0f1f0845599d3146a262bff4773b57743de49

v1.1

File Name	SHA-1 Hash
contracts/ERC1155Tradable.sol	92951c56a4a58960b77e2a25c6f0a9573493f715
contracts/MetaWallsCoin.sol	809107c527266269c9888bcc2ad549941f79f55c
contracts/common/grid.sol	bcb42923d48dd116fb35e5f1b9e02bb1ab5a67cd
contracts/common/meta-transactions/NativeMetaTransaction.sol	621e8a6fe3a5c1f85b33a8a675b9632d480685ce
contracts/common/meta-transactions/ContentMixin.sol	8b7d3ce9306b7dc5f48d895e25b5bee4748e1bf6
contracts/common/meta-transactions/EIP712Base.sol	5c67dc2114a8a2e9ce81e818e6c27559b0161818
contracts/common/meta-transactions/Initializable.sol	34d0f1f0845599d3146a262bff4773b57743de49

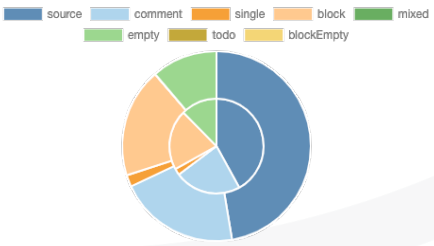
V2.0

File Name	SHA-1 Hash
contracts/GridMath.sol	0111ed761f10a2a3b61e8bd7d6654a841ee8277f
contracts/MetaWalls.sol	b94237c0da6b305d29c3b6d3cb8ab1d6715df54f
contracts/Builder.sol	89337598c3e76f1440c8d8c6108c8af5034e938c
contracts/opensea/common/grid.sol	bcb42923d48dd116fb35e5f1b9e02bb1ab5a67cd
contracts/opensea/common/meta-transactions/ContextMixin.sol	8b7d3ce9306b7dc5f48d895e25b5bee4748e1bf6
contracts/opensea/common/meta-transactions/NativeMetaTransaction.sol	43de403db3e0db3fd5b61a98b271608c44f9ab98
contracts/opensea/common/meta-transactions/EIP712Base.sol	5c67dc2114a8a2e9ce81e818e6c27559b0161818
contracts/opensea/common/meta-transactions/Initializable.sol	34d0f1f0845599d3146a262bff4773b57743de49

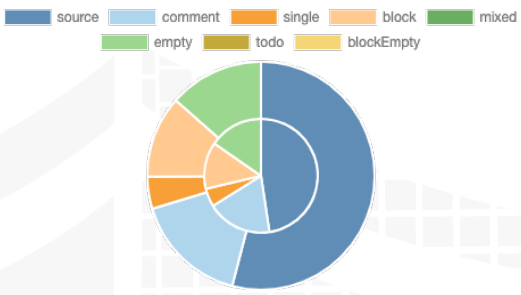
Metrics

Source Lines

v1.0

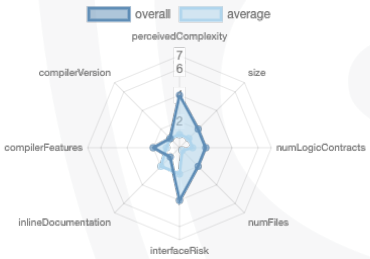


v2.0

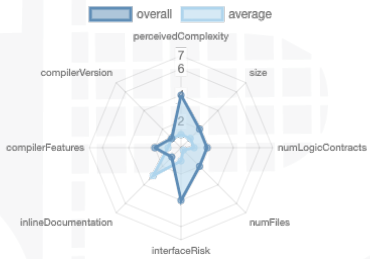


Risk Level

v1.0



v2.0



Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	7	1	0	1
2.0	6	1	0	1

Exposed Functions

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

Version	Public	Payable
1.0	33	1
1.1	39	1
2.0	31	1

Version	External	Internal	Private	Pure	View
1.0	1	25	2	5	24
1.1	1	37	5	8	26
2.0	1	40	2	10	24

State Variables

Version	Total	Public
1.0	16	6
1.1	17	6
2.0	16	4

Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	`^0.8.0` `>=0.8.0` `<0.9.0`		`yes`	`yes` (2 asm blocks)	

Version	Transfers ETH	Low-Level Calls	DelegateCall	Uses Hash Functions	ECRecover	New/Create/Create2
1.0				`yes`	`yes`	

Scope of Work

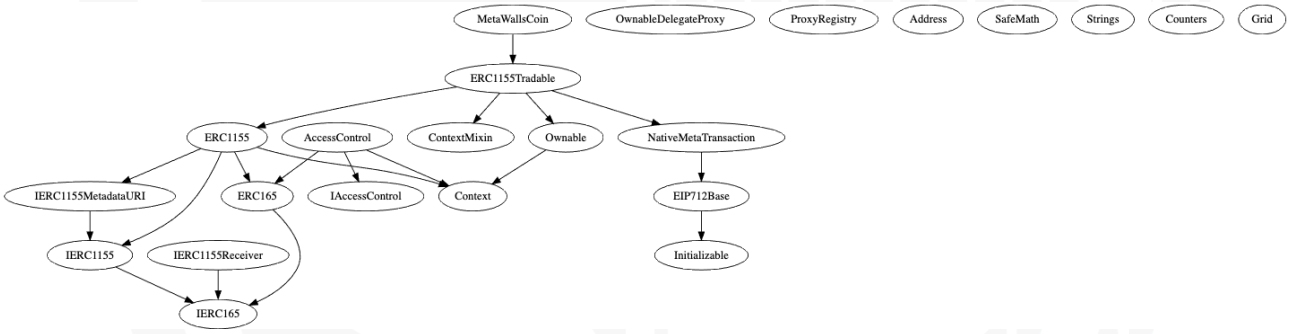
The above token Team provided us with the files that needs to be tested (Github, Bscscan, Etherscan, files, etc.). The scope of the audit is the main contract (usual the same name as team appended with .sol).

We will verify the following claims:

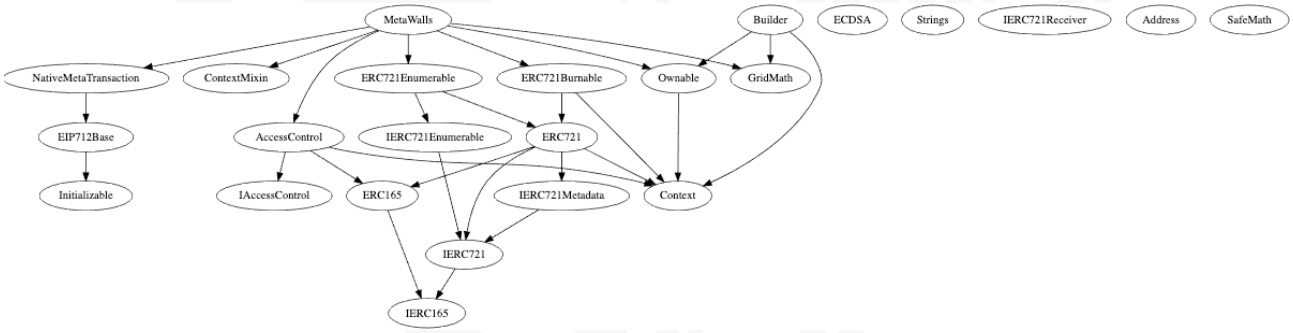
1. Correct implementation of Token standard
2. Deployer cannot mint any new tokens
3. Deployer cannot burn or lock user funds
4. Deployer cannot pause the contract
5. Overall checkup (Smart Contract Security)

Inheritance Graph

v1.0



v2.0



Verify Claims

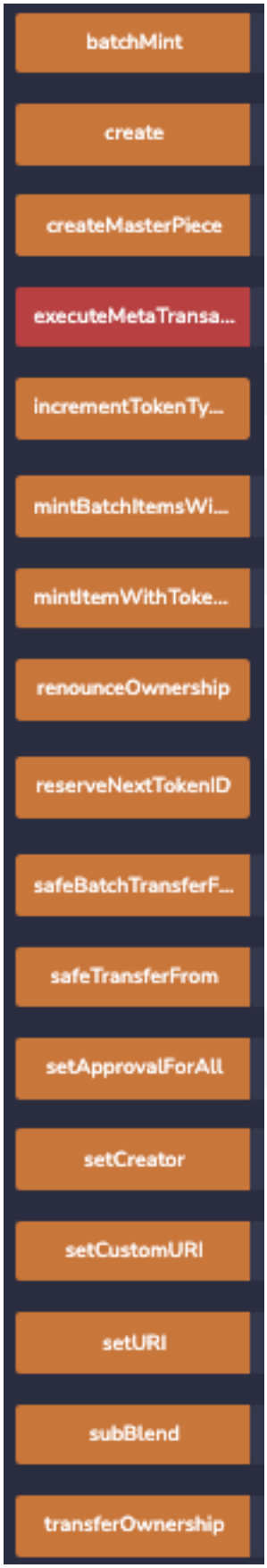
Correct implementation of Token standard

Tested	Verified
✓	✓

Function	Description	Exist	Tested	Verified
TotalSupply	provides information about the total token supply	✓	✓	✓
BalanceOf	provides account balance of the owner's account	✓	✓	✓
Transfer	executes transfers of a specified number of tokens to a specified address	✓	✓	✓
TransferFrom	executes transfers of a specified number of tokens from a specified address	✓	✓	✓
Approve	allow a spender to withdraw a set number of tokens from a specified account	✓	✓	✓
Allowance	returns a set number of tokens from a spender to the owner	✓	✓	✓

Write functions of contract

V1.0



v1.1

Changes

v2.0


METAWALLS
approve
burn
createMasterPiece
executeMetaTransaction
grantRole
mintBatchItemsWithToke...
mintItemWithTokenURI
mintItemWithTokenURIEx
renounceOwnership
renounceRole
revokeRole
safeTransferFrom
safeTransferFrom
setApprovalForAll
setNextTokenId
setProxyAddress
transferFrom
transferOwnership
BUILDER
blend
renounceOwnership
transferOwnership

Deployer cannot mint any new tokens

Name	Exist	Tested	Verified	File
Deployer cannot mint	✓	✓	✗	Main
Comment	Line: -			

Comments:

v1.0

- batchMint function
  - Only creators can mint
- create
  - onlyOwner can create/mint
- createMasterPiece
  - onlyOwner can create/mint
- mintItemWithTokenURI
  - onlyOwner can create/mint
- mintBatchItemsWithTokenURI
  - onlyOwner can create/mint

v1.1

- batchMint function
  - Removed
- onlyOwner changed to onlyOperatorOrOwner
- onlyOwner can mint

v2.0

- Only MINTER_ROLE
  - _create

Deployer cannot burn or lock user funds

Version	Name	Exist	Tested	Verified
1.0	Deployer cannot lock	✓	✓	✓
1.0	cannot burn	✓	✓	⚑
1.1	cannot burn	✓	✓	✗
2.0	cannot burn	✓	✓	✗

Comments:

v1.0

- _burn function is used in subBlend function

```
function subBlend(
    address account,
    uint256 masterPieceId,
    uint256[] memory ids,
    uint256 newItemId,
    string memory _uri,
    bytes memory data
) public masterPieceExists(masterPieceId) {
    for (uint256 x = 0; x < ids.length; x++) {
        _burn(account, ids[x], 1);
    }

    createMasterPiece(
        account,
        creators[masterPieceId],
        newItemId,
        1,
        1,
        _uri,
        data
    );
}
```

V1.1

```
233     function burn(
234         address account,
235         uint256 id,
236         uint256 value
237     ) public virtual {
238         require(
239             account == _msgSender() || isApprovedForAll(account, _msgSender()) || operator() == _msgSender(),
240             "ERC1155: caller is not owner nor approved nor operator"
241         );
242
243         _burn(account, id, value);
244     }
245
        ftrace | funcSig
246     function burnBatch(
247         address account,
248         uint256[] memory ids,
249         uint256[] memory values
250     ) public virtual {
251         require(
252             account == _msgSender() || isApprovedForAll(account, _msgSender()) || operator() == _msgSender(),
253             "ERC1155: caller is not owner nor approved nor operator"
254         );
255
256         _burnBatch(account, ids, values);
257     }
```

Burn function added

v2.0

```
        ftrace | funcSig
233     function _burn(uint256 id) internal virtual override {
234         super._burn(id);
235
236         _items[id].masterId = 0;
237     }
```

Deployer cannot pause the contract

Name	Exist	Tested	Verified
Deployer cannot pause	✓	✓	✓

Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

Legend

Attribute	Symbol
Verfified / Checked	✓
Partly Verified	⚑
Unverified / Not checked	✗
Not available	–

OnlyOwner functions

V1.0

- setURI
- Create
- createMaserPiece
- mintItemWithTokenURI
- mintBatchItemsWithTokenURI
- reserveNextTokenID
- incrementTokenTypeId

V1.1

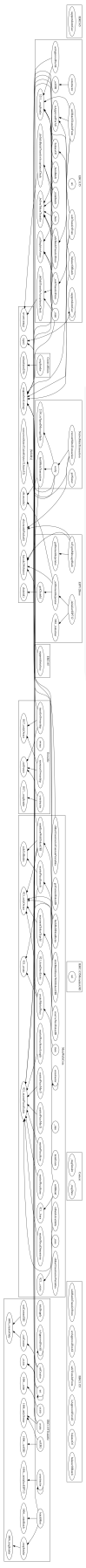
- onlyOwner
    - incrementTokenTypeId
    - reserveNextTokenID
    - createMasterPiece
    - incrementTokenTypeId

- onlyOperatorOrOwner
    - mintBatchItemsWithTokenURI
    - mintItemWithTokenURI
    - transferOperatorRole

v2.0

- onlyOwner
    - setNextTokenId
    - setProxyAddress
- onlyRole (MINTER_ROLE)
    - createMasterPiece
    - mintItemWithTokenURI
    - mintItemWithTokenURIEx
    - mintBatchItemsWithTokenURI
- masterPieceExists
    - mintItemWithTokenURI
    - mintItemWithTokenURIEx
    - mintBatchItemsWithTokenURI

CallGraph



Source Units in Scope

v1.0

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝	contracts/ERC1155Tradable.sol	3	———	230	218	105	86	85	☀️
📝	contracts/MetaWallsCoin.sol	1	———	347	282	164	67	140	———
📚	contracts/common/grid.sol	1	———	15	11	6	4	3	———
📝	contracts/common/meta-transactions/NativeMetaTransaction.sol	1	———	84	72	41	15	34	💰🧱🔮
🎨	contracts/common/meta-transactions/ContentMixin.sol	1	———	27	27	15	9	16	💻
📝	contracts/common/meta-transactions/EIP712Base.sol	1	———	70	70	41	18	32	💻🧱
📝	contracts/common/meta-transactions/Initializable.sol	1	———	21	21	9	8	3	———
📝📚🎨	**Totals**	**9**	———	**794**	**701**	**381**	**207**	**313**	💻💰🧱🔮☀️

v1.1

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝	contracts/ERC1155Tradable.sol	3	———	221	207	99	81	75	☀️
📝	contracts/MetaWallsCoin.sol	1	———	541	443	263	99	205	———
📚	contracts/common/grid.sol	1	———	15	11	6	4	3	———
📝	contracts/common/meta-transactions/NativeMetaTransaction.sol	1	———	117	101	70	15	34	💰🧱🔮
🎨	contracts/common/meta-transactions/ContentMixin.sol	1	———	27	27	15	9	16	💻
📝	contracts/common/meta-transactions/EIP712Base.sol	1	———	70	70	41	18	32	💻🧱
📝	contracts/common/meta-transactions/Initializable.sol	1	———	21	21	9	8	3	———
📝📚🎨	**Totals**	**9**	———	**1012**	**880**	**503**	**234**	**368**	💻💰🧱🔮☀️

v2.0

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝	contracts/GridMath.sol	1	———	56	52	41	3	21	———
📝	contracts/MetaWalls.sol	1	———	489	396	228	90	185	🧱
📝	contracts/Builder.sol	1	———	178	152	87	35	63	🧱
📚	contracts/opensea/common/grid.sol	1	———	15	11	6	4	3	———
🎨	contracts/opensea/common/meta-transactions/ContextMixin.sol	1	———	27	27	15	9	16	💻
📝	contracts/opensea/common/meta-transactions/NativeMetaTransaction.sol	1	———	84	72	41	15	34	💰🧱🔮
📝	contracts/opensea/common/meta-transactions/EIP712Base.sol	1	———	70	70	41	18	32	💻🧱
📝	contracts/opensea/common/meta-transactions/Initializable.sol	1	———	21	21	9	8	3	———
📝📚🎨	**Totals**	**8**	———	**940**	**801**	**468**	**182**	**357**	💻💰🧱🔮

Legend

Attribute	Description
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, ...)

Audit Results

AUDIT PASSED

Critical issues

- no critical issues found -

High issues

- no high issues found -

Medium issues

- no medium issues found -

Low issues

Version	Issue	File	Type	Line	Description
1.0	#1	Main	A floating pragma is set	5	The current pragma Solidity directive is „“>=0.8.0 <0.9.0 "".
1.1	#2	ERC1155Tradable	Missing Zero Address Validation	67	Check that the address is not zero
1.1	#3	ERC1155Tradable	State variable visibility is not set.	41, 38	It is best practice to set the visibility of state variables explicitly.
2.0	#4	MetaWalls	Missing Zero Address Validation	89, 435	Check that the address is not zero
2.0	#5	Builder	Missing Zero Address Validation	50	Check that the address is not zero
2.0	#6	MetaWalls	Local variables shadowing	442	_owner shadows Ownable._owner state variable Rename the local variables that shadow another component

2.0	#7	MetaWalls	A floating pragma is set	5	The current pragma Solidity directive is „“>=0.8.0 <0.9.0 "".
2.0	#8	GridMath	Source file does not specify required compiler version	/	Consider adding "pragma solidity ^0.8.2;"

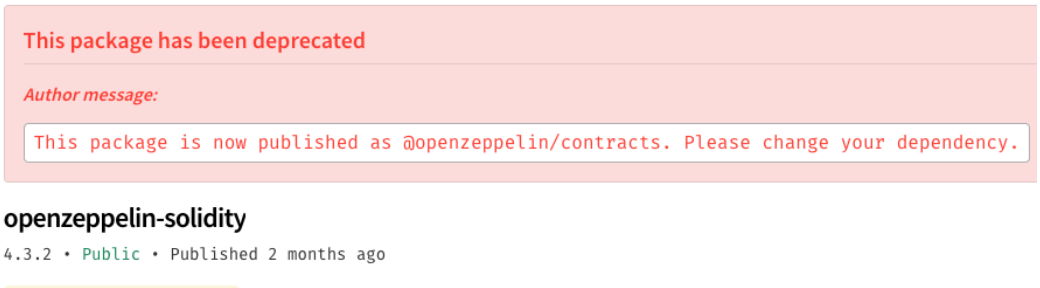
Informational issues

Version	Issue	File	Type	Line	Description
1.0	#1	MetaWallsCoin	Functions that are not used	76-79, 81-84	Remove unused functions
2.0	#2	GridMath	Functions that are not used	4, 9	Remove unused functions
2.0	#3	MetaWalls	Functions that are not used	229	Remove unused functions
2.0	#4	GridMath	SPDX license identifier not provided	/	Before publishing, consider adding a comment containing "SPDX-License-Identifier: <SPDX-License>" to each source file.

Audit Comments

V1.0: 06. November 2021:

- Openzeppelin-solidity/contracts library is deprecated
  - Used in files:
    - ERC1155Tradeable
    - MetaWallsCoin
    - NativeMetaTransaction



Source: https://www.npmjs.com/package/openzeppelin-solidity, Sat 6. Nov. 11:48 AM

- Following files were borrowed from OpeanSea (Source: https://github.com/ProjectOpenSea/opensea-creatures)
  - Meta-transactions directory
    - ContextMixin
    - EIP712Base
    - Initializable
    - NativeMetaTransaction
  - ERC1155Tradable
- Anyone can set creator

V1.1: 10. November 2021:

- MetaWalls.sol
  - New in file:
    - AccessControl removed
    - ERC1155Burnable added
    - Contract name MetaWallsCoin changed to MetaWalls
    - Struct PieceItem
      - Added:
        - Bool isToBeBlend
        - Bool isToBeCracked

```
31    address private _operator;
32
33    event OperatorRoleTransferred(address indexed previousOperator, address indexed newOperator);
34
35    /**
36     * @dev Emitted when sub blending is requested by a certain account for concrete items
37     */
38    event SubBlendingRequested(address indexed operator, address indexed account, uint256 masterPieceId, uint256[] ids);
39
40    /**
41     * @dev Emitted when cracking is requested by a certain account for a concrete item
42     */
43    event CrackingRequested(
44        address indexed operator,
45        address indexed account,
46        uint256 masterPieceId,
47        uint256 sourceItemId
48    );
```

```
45    modifier masterPieceExists(uint256 _id⬆) {
46        require(_masterPieces[_id⬆].countX > 0, "Master Piece does not exist");
47        _;
48    }
```

```
73    modifier itemDoesExists(uint256 _id⬆) {
74        require(_items[_id⬆].masterId > 0, "Item does not exist");
75        _;
76    }
```

```
81    modifier onlyOperatorOrOwner() {
82        require(operator() == _msgSender() || owner() == _msgSender(), "caller is not the operator or owner");
83        _;
84    }
```

New modifier

```
100   function operator() public view virtual returns (address) {
101       return _operator;
102   }
103
104   /**
105    * @dev Transfers operator role of the contract to a new account (`newOperator`).
106    * Can only be called by the current owner or current operator.
107    */
      ftrace | funcSig
108   function transferOperatorRole(address newOperator⬆) public virtual onlyOperatorOrOwner {
109       require(newOperator⬆ != address(0), "new operator is the zero address");
110       _setOperator(newOperator⬆);
111   }
112
      ftrace | funcSig
113   function _setOperator(address newOperator⬆) private {
114       address oldOperator = _operator;
115       _operator = newOperator⬆;
116       emit OperatorRoleTransferred(oldOperator, newOperator⬆);
117   }
```

New functions added

```
140     function _maxArray(uint32[] memory values⬆) private pure returns (uint32) {
141         require(values⬆.length > 0);
142
143         uint32 max = values⬆[0];
144         for (uint32 i = 1; i < values⬆.length; ++i) {
145             if (values⬆[i] > max) {
146                 max = values⬆[i];
147             }
148         }
149         return max;
150     }
151
        ftrace | funcSig
152     function _minArray(uint32[] memory values⬆) private pure returns (uint32) {
153         require(values⬆.length > 0);
154
155         uint32 min = values⬆[0];
156         for (uint32 i = 1; i < values⬆.length; ++i) {
157             if (values⬆[i] < min) {
158                 min = values⬆[i];
159             }
160         }
161         return min;
162     }
163
        ftrace | funcSig
164     function isRectangle(uint32[] memory xs⬆, uint32[] memory ys⬆) public pure returns (bool) {
165         require(xs⬆.length > 1 || ys⬆.length > 1, "Rectangle requires more than one piece");
166         require(xs⬆.length == ys⬆.length, "Length of xs and ys must match");
167
168         uint32 minX = _minArray(xs⬆);
169         uint32 maxX = _maxArray(xs⬆);
170
171         uint32 minY = _minArray(ys⬆);
172         uint32 maxY = _maxArray(ys⬆);
173
174         uint32 width = maxX - minX + 1;
175         uint32 height = maxY - minY + 1;
176
177         if (xs⬆.length == width * height) {
178             return true;
179         }
180
181         return false;
182     }
```

New functions added

v2.0: 15. December 2021:

- Import OpenZeppelin with @ at the start and use @openzeppelin/contracts instead of openzeppelin-solidity (see v1.0 comments above)

MetaWalls Test Protocol

Used NFT library (with modifications)
https://github.com/ProjectOpenSea/opensea-creatures

Proxy address on Rinkeby network:
0xF57B2c51dED3A29e6891aba85459d600256Cf317

Proxy address on other networks:
0xa5409ec958c83c3f309868babaca7c86dcb077c1

Testnet address (owner)
0xd07fDB68bbcA2A00f9694Ffe0A05472687C0af83

Note: This library was borrowed from OpenSea in order to get OpenSea-Compatibility
The condition in the deploy.js file to select the proxy address is always true.

Compiling successful
Compiling 22 files with 0.8.2
Generating typings for: 24 artifacts in dir: typechain for target: ethers-v5
Successfully generated 33 typings!
Compilation finished successfully

All Unit Tests successful
40 passing (7s)

Deployment to Rinkeby testnet over infura.io successful
Contract address: 0xBFA1AB6CF738CD1377Fa134d0d62cD79bc8e85EF
Transaction:
0x7abbc0497fa4678e5e26d9953039b751eaa081676277476f9a10f6bc7b9000d3

Verification of the contract files successful
Nothing to compile
No need to generate any newer typings.
Compiling 1 file with 0.8.2
Successfully submitted source code for contract
contracts/MetaWallsCoin.sol:MetaWallsCoin at
0xbfa1ab6cf738cd1377fa134d0d62cd79bc8e85ef
for verification on Etherscan. Waiting for verification result...

Successfully verified contract MetaWallsCoin on Etherscan.
https://rinkeby.etherscan.io/address/
0xbfa1ab6cf738cd1377fa134d0d62cd79bc8e85ef#code

Creation of a MasterPiece successful
Transaction: https://rinkeby.etherscan.io/tx/
0x165a31a7fb0345f177592f7cec3602d91397ced72403ef931e3130d2cad00a88

Minting of Co-NFT successful
Transaction of creation: https://rinkeby.etherscan.io/tx/0x322534d277591ef23dd6216b83ff89142fdab4dba15afe9adfed4ae0f64a8566

Transaction failed if MasterPiece doesn't exists: https://rinkeby.etherscan.io/tx/0x374a5d7592c8c585ff796d81d95bdbdcef4d01dd8e652364d9f3655f6b348884

Sub-blend of two Co-NFT's successful
Transaction: https://rinkeby.etherscan.io/tx/0xaca04c084ec34495bcbd28ea9ea37012cc614b5d341ec33971ac2b602f9fbb21

The two items will be deleted and a new MasterPiece will be created. After a sub-blend transaction the items are still visible in the itemExists function.

Checking read-only functions

balanceOf	success
exists	success
getChainId	success
itemExists	success
item	no return if item doesn't exists
masterPieceDimensions	no return if MasterPiece doesn't exists
masterPieceItems	success
masterPieceItemAt	success
masterPieceItemsLength	success
masterPieceMaxX	success
masterPieceMaxY	success

Additional due to V2

Compiling successful
Compiling 26 files with 0.8.2
Generating typings for: 26 artifacts in dir: typechain-types for target: ethers-v5
Successfully generated 41 typings!
Compilation finished successfully

Deployment
All contracts were deployed on local running blockchain node. No excessive gas usages were detected.

Conclusion
Custom claims were not specified, the basic functions that can be constructed from the unit tests were tested.

The contract is safe to deploy and basic logic errors were not detected. The code is written to the best standard and sufficiently commented. Known risks were checked by the auditor and the code was scanned for other vulnerabilities as well.

All unit tests make sense and have also been checked. Logic errors could not be found here either.

All Unit Tests successful

39 passing (7s)

```
Token
  ✓ calculates the grid correctly
  Security: everything is setup as expected
    ✓ Chantal is owner
    ✓ Axel is NOT an owner
    ✓ Ben is NOT an owner
  Item existence can be checked correctly
    ✓ itemExists for non-existing item works correctly
    ✓ item() for non-existing item reverts correctly
  Tokenization tests: creation of MasterPieces and minting of Co-NFTs
    ✓ non-owner are not allowed to mint a master piece
    ✓ Chantal as owner can create a master piece
    can not ask for data on non-existing tokens
      ✓ can' read data from non-existing tikens
    can not do things on non-existing Master-Pieces
      ✓ can' read data from non-existing master piece
      ✓ can' read countX from non-existing master piece
      ✓ can' read countY from non-existing master piece
      ✓ can' read items from non-existing master piece
      ✓ can' read item from non-existing master piece
      ✓ can' read items-length from non-existing master piece
      ✓ can' mint Co-NFT for non-existing master piece
    minting Co-NFTs of a master piece
      ✓ has the right bounds/dimensions
      ✓ all Co-NFT items are set to zero
      ✓ should mint tokens to Axel successfully
      ✓ should mint tokens with Custom URI successfully
      ✓ it can mix tokens with custom URIs and the base URI properly
      ✓ won't allow to mint a Co-NFT out of bounds of master-piece
    transfer of token should be work properly
      ✓ axel can transfer token to ben
      ✓ axel can't transfer token again to ben
    Batch Minting
      ✓ can be minted as batch
      ✓ can event be minted as batch with empty uris
    Burning on behalf of the user
      ✓ axel is allowed to burn his own tokens
      ✓ a non-burning role will NOT be able to burn other users token

Grid
  ✓ calculates the index in a grid correctly

Special Co-NFT capabilities
  tooling works properly
    ✓ isRectangle will detect rectangle
    ✓ isRectangle will detect a non-rectangle properly
    ✓ isRectangle will detect a wrong list of xs/ys properly
  Sub Blending works properly
    Sub Blending Example 1
      will detect rectangle correctly
        ✓ will detect the full rectangle correctly
        ✓ will detect the top-left corner as rectangle correctly
        ✓ will detect the right-bottom corner as rectangle correctly
        ✓ will detect the middle area as rectangle correctly
        ✓ will detect a non-rectangle correctly
        ✓ will detect non-existing items correctly
      can sub-blend a square successfully
        ✓ will emit burn and mint events properly
```

Disclaimer

We have checked and verified the code to the best of our knowledge. However, deeper logic errors cannot be excluded and Solidproof.io cannot be held liable for any damage that may occur.

SWC Attacks

ID	Title	Relationships	Status
SWC-136	Unencrypted Private Data On-Chain	CWE-767: Access to Critical Private Variable via Public Method	PASSED
SWC-135	Code With No Effects	CWE-1164: Irrelevant Code	PASSED
SWC-134	Message call with hardcoded gas amount	CWE-655: Improper Initialization	PASSED
SWC-133	Hash Collisions With Multiple Variable Length Arguments	CWE-294: Authentication Bypass by Capture-replay	PASSED
SWC-132	Unexpected Ether balance	CWE-667: Improper Locking	PASSED
SWC-131	Presence of unused variables	CWE-1164: Irrelevant Code	PASSED
SWC-130	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	PASSED
SWC-129	Typographical Error	CWE-480: Use of Incorrect Operator	PASSED
SWC-128	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	PASSED

SWC-127	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	PASSED
SWC-125	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	PASSED
SWC-124	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	PASSED
SWC-123	Requirement Violation	CWE-573: Improper Following of Specification by Caller	PASSED
SWC-122	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	PASSED
SWC-121	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	PASSED
SWC-120	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	PASSED
SWC-119	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	NOT PASSED
SWC-118	Incorrect Constructor Name	CWE-665: Improper Initialization	PASSED
SWC-117	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	PASSED

SWC-116	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SWC-115	Authorization through tx.origin	CWE-477: Use of Obsolete Function	PASSED
SWC-114	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	PASSED
SWC-113	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	PASSED
SWC-112	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SWC-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	PASSED
SWC-110	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	PASSED
SWC-109	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	PASSED
SWC-108	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	NOT PASSED
SWC-107	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	PASSED
SWC-106	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	PASSED

SWC-105	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	PASSED
SWC-104	Unchecked Call Return Value	CWE-252: Unchecked Return Value	PASSED
SWC-103	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	NOT PASSED
SWC-102	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	PASSED
SWC-101	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	PASSED
SWC-100	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY