



**SOLIDProof**  
*Bring trust into your projects*

**Blockchain Security | Smart Contract Audits | KYC**

MADE IN GERMANY

**ArtPool**

**Audit**

**Security Assessment**

**16. April, 2022**

**For**

**ARTPOOL**

Disclaimer	3
Description	5
Project Engagement	5
Logo	5
Contract Link	5
Methodology	7
Used Code from other Frameworks/Smart Contracts (direct imports)	8
Tested Contract Files	9
Source Lines	10
Risk Level	10
Capabilities	11
Inheritance Graph	12
CallGraph	13
Scope of Work/Verify Claims	14
Modifiers and public functions	18
Source Units in Scope	20
Critical issues	21
High issues	21
Medium issues	21
Low issues	21
Informational issues	22
Audit Comments	22
SWC Attacks	23

# 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	25. March 2022	<ul style="list-style-type: none"><li>• Layout project</li><li>• Automated- /Manual-Security Testing</li><li>• Summary</li></ul>
1.1	16. April 2022	Reaudit

## **Network**

Kusama

## **Website**

<https://www.artcuratorgrid.com/>

## **Twitter**

[https://twitter.com/Artpool\\_xyz](https://twitter.com/Artpool_xyz)

## **Facebook**

<https://www.facebook.com/artcuratorgrid>

## **Telegram**

<https://t.me/artpoolxyz>

## **Instagram**

<https://www.instagram.com/artpool.xyz/?hl=en>

## **LinkedIn**

<https://www.linkedin.com/company/artpool>

## **Discord**

<https://discord.gg/VJdE6tgUtd>

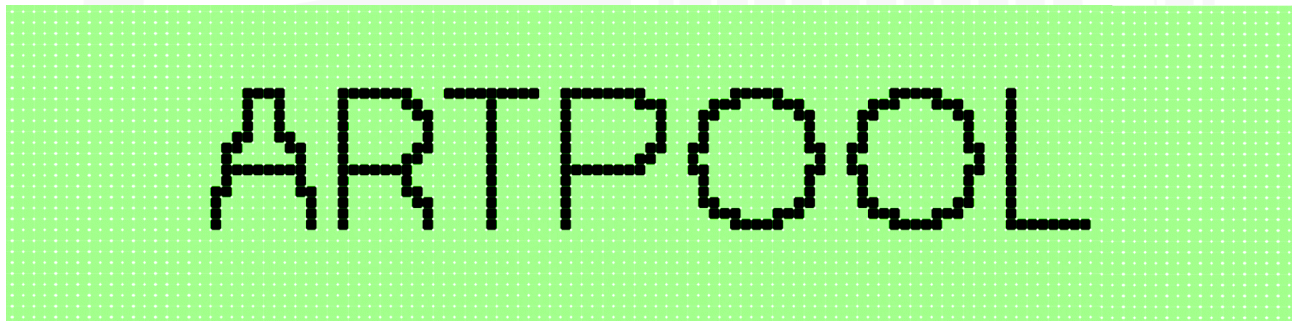
## Description

The First **Curator-Driven** NFT Platform powered by a global network of vetted art professionals

## Project Engagement

During the 24th of March 2022, **ArtPool 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

- Provided as files

### v1.1

- Provided as files

# 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)
<b>Critical</b>	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.
<b>High</b>	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.
<b>Medium</b>	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.
<b>Low</b>	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.
<b>Informational</b>	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)

Imported packages:

**v1.0**

Dependency / Import Path	Count
@openzeppelin/contracts/access/Ownable.sol	1
@openzeppelin/contracts/security/ReentrancyGuard.sol	1
@openzeppelin/contracts/token/ERC721/ERC721.sol	1
@openzeppelin/contracts/token/ERC721/extensions/ERC721Burnable.sol	1
@openzeppelin/contracts/token/ERC721/extensions/ERC721URIStorage.sol	1
@openzeppelin/contracts/utils/Counters.sol	1
@openzeppelin/contracts/utils/Strings.sol	1
@openzeppelin/contracts/utils/math/SafeMath.sol	1

**v1.1**

Dependency / Import Path	Count
@openzeppelin/contracts/access/Ownable.sol	2
@openzeppelin/contracts/interfaces/IERC2981.sol	1
@openzeppelin/contracts/security/ReentrancyGuard.sol	1
@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/token/ERC721/extensions/ERC721URIStorage.sol	1
@openzeppelin/contracts/utils/Counters.sol	1
@openzeppelin/contracts/utils/Strings.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

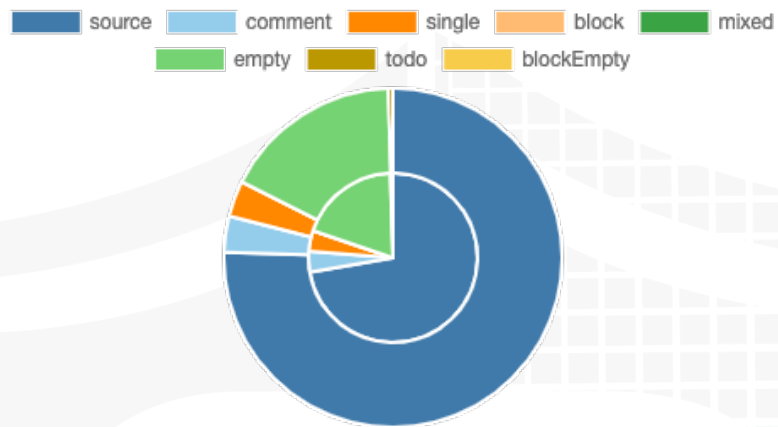
File Name	SHA-1 Hash
contracts/NftCollection/NftCollectionV1.sol	f4e67f0ea6c9fed9027448a03ed9ed6eab924686
contracts/BusinessLogic/BusinessLogicV1.sol	6f28a87585d6bcc6c88762fb7c8a966d04945a95
contracts/BusinessLogic/BusinessLogic.sol	f79341ef2a7852f38a7887d32c393640091d21c7

### v1.1

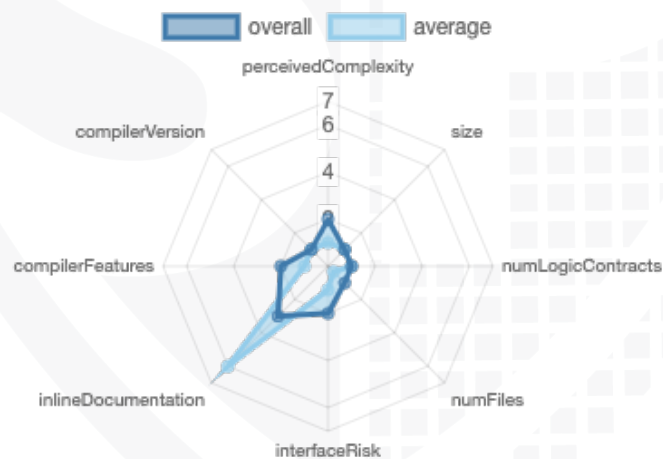
File Name	SHA-1 Hash
contracts/Membership/Membership.sol	a64995263cff812c9856d95715f0a2b7adb90ccd
contracts/Membership/IMembership.sol	a00a2e9a9ce112c1e8ded550ab87afdd6158e313
contracts/Membership/TestableMembership.sol	e8a3eff698c38b2a586239c16b8db6a135d8f356
contracts/NftCollection/NftCollectionV1.sol	e4e19210cfaa8ca9932c00941d78685166f36661
contracts/BusinessLogic/BusinessLogicV1.sol	2166f3312d96abe37c6bdf4b723bc244b8f69295
contracts/BusinessLogic/IBusinessLogic.sol	d35e17a2ae0aeb61701e5481455ab8b0b31998a7
contracts/LazyMinting/LazyMinting.sol	2d5dea79495004ce30b920c646d0e42f1959a430

# Metrics

## Source Lines v1.0



## Risk Level v1.0



## Capabilities

### Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	2	0	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	11	1

Version	External	Internal	Private	Pure	View
1.0	0	8	0	0	9

### State Variables

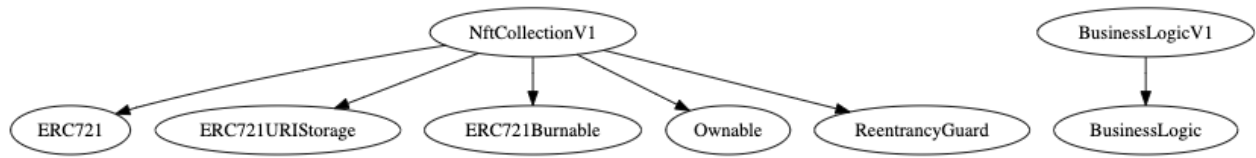
Version	Total	Public
1.0	7	0

### Capabilities

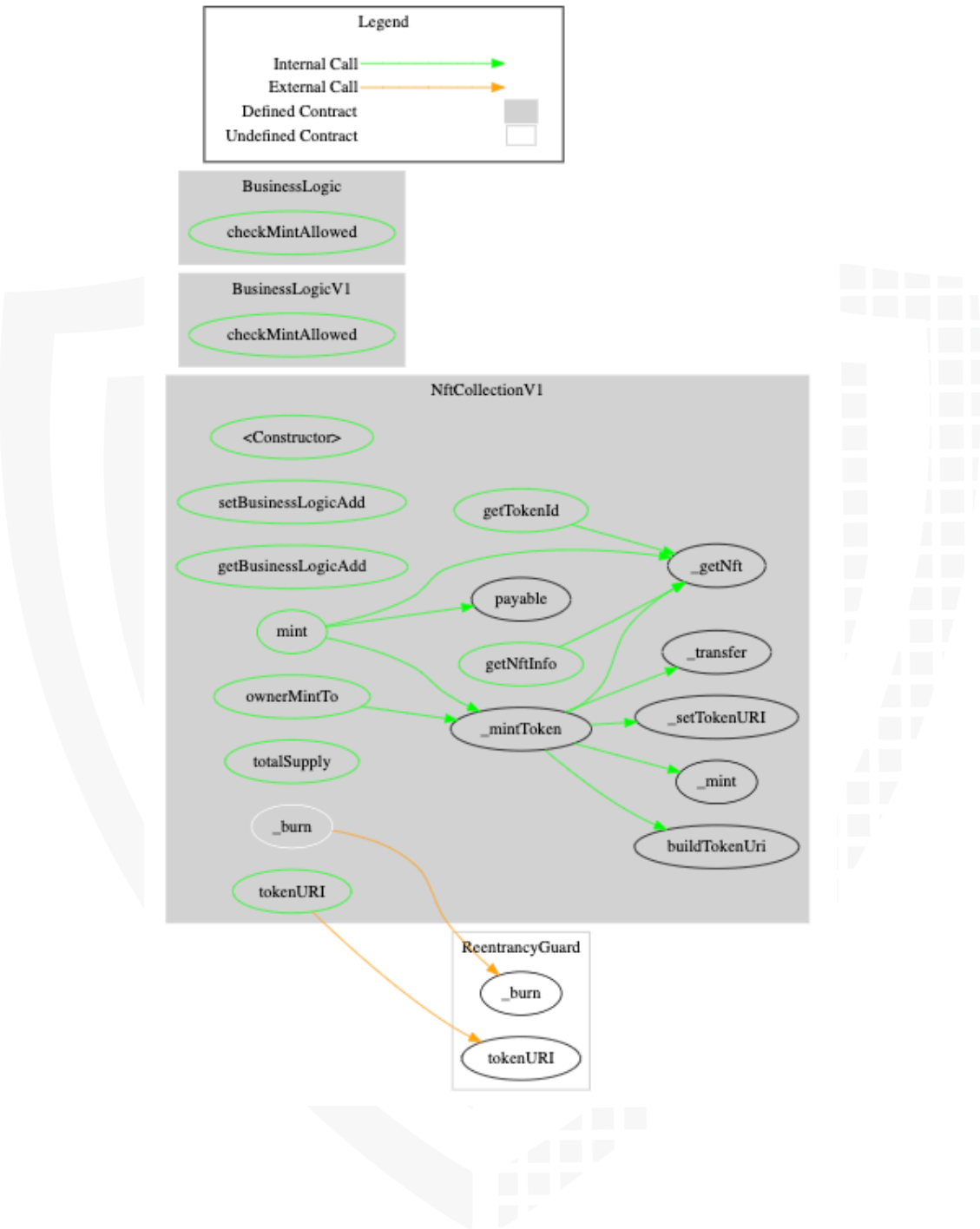
Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	^0.8.4		yes		

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

## Inheritance Graph v1.0



# CallGraph v1.0



## Scope of Work/Verify Claims

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. Overall checkup (Smart Contract Security)



## Correct implementation of Token standard

ERC721				
Function	Description	Exist	Tested	Verified
BalanceOf	Count all NFTs assigned to an owner	✓	✓	✓
OwnerOf	Find the owner of an NFT	✓	✓	✓
SafeTransferFrom	Transfers the ownership of an NFT from one address to another address	✓	✓	✓
SafeTransferFrom	See above - Difference is that this function has an extra data parameter	✓	✓	✓
TransferFrom	Transfer ownership of an NFT	✓	✓	✓
Approve	Change or reaffirm the approved address for an NFT	✓	✓	✓
SetApprovalForAll	Enable or disable approval for a third party ("operator") to manage all of `msg.sender`'s assets	✓	✓	✓
GetApproved	Get the approved address for a single NFT	✓	✓	✓
IsApprovedForAll	Query if an address is an authorized operator for another address	✓	✓	✓
SupportsInterface	Query if a contract implements an interface	✓	✓	✓
Name	Provides information about the name	✓	✓	✓
Symbol	Provides information about the symbol	✓	✓	✓
TokenURI	Provides information about the TokenUri	✓	✓	✓

## Write functions of contract v1.0

setBusinessLogicAdd  
mint 💰  
ownerMintTo

renounceOwnership  
transferOwnership  
burn  
approve  
setApprovalForAll  
transferFrom  
safeTransferFrom

## v1.1

setPresaleOffset  
setMembershipContractAddress

mint 💰  
ownerMintTo

mint 💰  
setBusinessLogicAddress



## Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

### Legend

Attribute	Symbol
Verified / Checked	✓
Partly Verified	🚩
Unverified / Not checked	✗
Not available	—

# Modifiers and public functions

## v1.0

```

  ✓ 🔹 setBusinessLogicAdd
    ☹️ onlyOwner
    🔹 mint 💰
  ✓ 🔹 ownerMintTo
    ☹️ onlyOwner

```

## Comments

- Deployer can set following addresses
  - \_businessLogicAdd
- Owner can mint tokens to everybody without paying something
- Anybody can mint new nft
- For your information, not listed functions are functions from imported packages

## v1.1

```

  🔹 setPresaleOffset
    ☹️ onlyOwner
  🔹 setMembershipContractAddress
    ☹️ onlyOwner

```

```

  🔹 mint 💰
  ✓ 🔹 ownerMintTo
    ☹️ onlyOwner
    ☹️ nonReentrant

```

```

  ✓ 🔹 mint 💰
    ☹️ nonReentrant

```

```

  ✓ 🔹 mint 💰
    ☹️ nonReentrant
  ✓ 🔹 setBusinessLogicAddress
    ☹️ onlyOwner

```

Note: Not listed functions are functions from inherited libraries







## Comments

- Deployer can set following state variables without limitations
  - BusinessLogicV1
    - \_presaleOffset
  -
- Deployer can set following addresses
  - BusinessLogicV1
    - \_membershipContractAddress
  - NftCollectionV1
    - businessLogicAddress
- We recommend you to use libraries for generating random numbers like Chainlink VRF














**Please check if an OnlyOwner or similar restrictive modifier has been forgotten.**

## Source Units in Scope

### v1.0

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
	contracts/NftCollection/NftCollectionV1.sol	1	————	188	171	134	6	76	
	contracts/BusinessLogic/BusinessLogicV1.sol	1	————	21	15	9	1	8	————
	contracts/BusinessLogic/BusinessLogic.sol	1	————	15	7	3	1	3	————
	<b>Totals</b>	<b>3</b>	————	<b>224</b>	<b>193</b>	<b>146</b>	<b>8</b>	<b>87</b>	

### v1.1

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
	contracts/Membership/Membership.sol	1	————	176	164	122	5	70	
	contracts/Membership/IMembership.sol	————	1	10	6	3	1	3	
	contracts/Membership/TestableMembership.sol	1	————	38	38	31	2	5	————
	contracts/NftCollection/NftCollectionV1.sol	1	————	112	108	83	3	53	
	contracts/BusinessLogic/BusinessLogicV1.sol	1	————	108	87	65	3	37	————
	contracts/BusinessLogic/IBusinessLogic.sol	————	1	14	6	3	1	3	————
	contracts/LazyMinting/LazyMinting.sol	1	————	327	262	203	19	91	
	<b>Totals</b>	<b>5</b>	<b>2</b>	<b>785</b>	<b>671</b>	<b>510</b>	<b>34</b>	<b>262</b>	

### 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**

## High issues

**No high issues**

## Medium issues

**No medium issues**

## Low issues

Issue	File	Type	Line	Description
#1	All	A floating pragma is set	3	The current pragma Solidity directive is „^0.8.4“.
#2	NftCollectionV1	Missing Zero Address Validation (missing-zero-check)	21, 109	Check that the address is not zero
#3	NftCollectionV1	Local variables shadowing	18, 19	Rename the local variables that shadow another component
#4	LazyMinting	Local variables shadowing	100, 101	Rename the local variables that shadow another component
#5	Membership	Local variables shadowing	26, 27	Rename the local variables that shadow another component
	BusinessLogicV1	Missing Events Arithmetic		Emit an event for critical parameter changes

## Informational issues

Issue	File	Type	Line	Description
#1	Membership	Param is not used in function	100	<p>Remove param in function if you are not going to use it.</p> <p>Change the following function L100-L116 if you're not going to use parameter:</p> <pre>function mint(string memory <b>notUsed</b>) public payable override nonReentrant {     ... }</pre> <p>to</p> <pre>function mint(string memory) public payable override nonReentrant {     ... }</pre> <p>Just remove the red marked param name.</p>

## Audit Comments

We recommend you to use the special form of comments (NatSpec Format, Follow link for more information <https://docs.soliditylang.org/en/v0.5.10/natspec-format.html>) for your contracts to provide rich documentation for functions, return variables and more. This helps investors to make clear what that variables, functions etc. do.

### 25. March 2022:

- Read whole report for more information

## SWC Attacks

ID	Title	Relationships	Status
<a href="#">SW C-1 36</a>	Unencrypted Private Data On-Chain	<a href="#">CWE-767: Access to Critical Private Variable via Public Method</a>	PASSED
<a href="#">SW C-1 35</a>	Code With No Effects	<a href="#">CWE-1164: Irrelevant Code</a>	PASSED
<a href="#">SW C-1 34</a>	Message call with hardcoded gas amount	<a href="#">CWE-655: Improper Initialization</a>	PASSED
<a href="#">SW C-1 33</a>	Hash Collisions With Multiple Variable Length Arguments	<a href="#">CWE-294: Authentication Bypass by Capture-replay</a>	PASSED
<a href="#">SW C-1 32</a>	Unexpected Ether balance	<a href="#">CWE-667: Improper Locking</a>	PASSED
<a href="#">SW C-1 31</a>	Presence of unused variables	<a href="#">CWE-1164: Irrelevant Code</a>	PASSED
<a href="#">SW C-1 30</a>	Right-To-Left-Override control character (U+202E)	<a href="#">CWE-451: User Interface (UI) Misrepresentation of Critical Information</a>	PASSED
<a href="#">SW C-1 29</a>	Typographical Error	<a href="#">CWE-480: Use of Incorrect Operator</a>	PASSED
<a href="#">SW C-1 28</a>	DoS With Block Gas Limit	<a href="#">CWE-400: Uncontrolled Resource Consumption</a>	PASSED

<a href="#">SW C-1 27</a>	Arbitrary Jump with Function Type Variable	<a href="#">CWE-695: Use of Low-Level Functionality</a>	<b>PASSED</b>
<a href="#">SW C-1 25</a>	Incorrect Inheritance Order	<a href="#">CWE-696: Incorrect Behavior Order</a>	<b>PASSED</b>
<a href="#">SW C-1 24</a>	Write to Arbitrary Storage Location	<a href="#">CWE-123: Write-what-where Condition</a>	<b>PASSED</b>
<a href="#">SW C-1 23</a>	Requirement Violation	<a href="#">CWE-573: Improper Following of Specification by Caller</a>	<b>PASSED</b>
<a href="#">SW C-1 22</a>	Lack of Proper Signature Verification	<a href="#">CWE-345: Insufficient Verification of Data Authenticity</a>	<b>PASSED</b>
<a href="#">SW C-1 21</a>	Missing Protection against Signature Replay Attacks	<a href="#">CWE-347: Improper Verification of Cryptographic Signature</a>	<b>PASSED</b>
<a href="#">SW C-1 20</a>	Weak Sources of Randomness from Chain Attributes	<a href="#">CWE-330: Use of Insufficiently Random Values</a>	<b>PASSED</b>
<a href="#">SW C-11 9</a>	Shadowing State Variables	<a href="#">CWE-710: Improper Adherence to Coding Standards</a>	<b>PASSED</b>
<a href="#">SW C-11 8</a>	Incorrect Constructor Name	<a href="#">CWE-665: Improper Initialization</a>	<b>PASSED</b>
<a href="#">SW C-11 7</a>	Signature Malleability	<a href="#">CWE-347: Improper Verification of Cryptographic Signature</a>	<b>PASSED</b>



<a href="#">SW C-11 6</a>	Timestamp Dependence	<a href="#">CWE-829: Inclusion of Functionality from Untrusted Control Sphere</a>	<b>PASSED</b>
<a href="#">SW C-11 5</a>	Authorization through tx.origin	<a href="#">CWE-477: Use of Obsolete Function</a>	<b>PASSED</b>
<a href="#">SW C-11 4</a>	Transaction Order Dependence	<a href="#">CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')</a>	<b>PASSED</b>
<a href="#">SW C-11 3</a>	DoS with Failed Call	<a href="#">CWE-703: Improper Check or Handling of Exceptional Conditions</a>	<b>PASSED</b>
<a href="#">SW C-11 2</a>	Delegatecall to Untrusted Callee	<a href="#">CWE-829: Inclusion of Functionality from Untrusted Control Sphere</a>	<b>PASSED</b>
<a href="#">SW C-11 1</a>	Use of Deprecated Solidity Functions	<a href="#">CWE-477: Use of Obsolete Function</a>	<b>PASSED</b>
<a href="#">SW C-11 0</a>	Assert Violation	<a href="#">CWE-670: Always-Incorrect Control Flow Implementation</a>	<b>PASSED</b>
<a href="#">SW C-1 09</a>	Uninitialized Storage Pointer	<a href="#">CWE-824: Access of Uninitialized Pointer</a>	<b>PASSED</b>
<a href="#">SW C-1 08</a>	State Variable Default Visibility	<a href="#">CWE-710: Improper Adherence to Coding Standards</a>	<b>PASSED</b>
<a href="#">SW C-1 07</a>	Reentrancy	<a href="#">CWE-841: Improper Enforcement of Behavioral Workflow</a>	<b>PASSED</b>
<a href="#">SW C-1 06</a>	Unprotected SELFDESTRUCT Instruction	<a href="#">CWE-284: Improper Access Control</a>	<b>PASSED</b>

<a href="#">SW</a> <a href="#">C-1</a> <a href="#">05</a>	Unprotected Ether Withdrawal	<a href="#">CWE-284: Improper Access Control</a>	<b>PASSED</b>
<a href="#">SW</a> <a href="#">C-1</a> <a href="#">04</a>	Unchecked Call Return Value	<a href="#">CWE-252: Unchecked Return Value</a>	<b>PASSED</b>
<a href="#">SW</a> <a href="#">C-1</a> <a href="#">03</a>	Floating Pragma	<a href="#">CWE-664: Improper Control of a Resource Through its Lifetime</a>	<b>NOT PASSED</b>
<a href="#">SW</a> <a href="#">C-1</a> <a href="#">02</a>	Outdated Compiler Version	<a href="#">CWE-937: Using Components with Known Vulnerabilities</a>	<b>PASSED</b>
<a href="#">SW</a> <a href="#">C-1</a> <a href="#">01</a>	Integer Overflow and Underflow	<a href="#">CWE-682: Incorrect Calculation</a>	<b>PASSED</b>
<a href="#">SW</a> <a href="#">C-1</a> <a href="#">00</a>	Function Default Visibility	<a href="#">CWE-710: Improper Adherence to Coding Standards</a>	<b>PASSED</b>

The logo features the words "SolidProof" in a white, handwritten-style script. The "P" is large and stylized, with a long horizontal stroke that extends to the left. The background is a solid blue color with a faint, large shield emblem. The shield has a grid-like pattern on its right side and a solid blue area on its left side.

SolidProof

**Blockchain Security | Smart Contract Audits | KYC**

A small horizontal bar representing the German flag, with black, red, and gold stripes.

MADE IN GERMANY