



SOLIDProof
Bring trust into your projects

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

MADE IN GERMANY

Vountain

AUDIT

SECURITY ASSESSMENT

22. July, 2024

FOR



SolidProof.io



@solidproof_io



Introduction	3
Disclaimer	3
Project Overview	4
Summary	4
Social Medias	4
Audit Summary	5
File Overview	6
Imported packages	7
Components	8
Exposed Functions	8
Capabilities	9
Inheritance Graph	10
Audit Information	11
Vulnerability & Risk Level	11
Auditing Strategy and Techniques Applied	12
Methodology	12
Overall Security	13
Upgradeability	13
Ownership	14
Ownership Privileges	15
Minting tokens	15
Burning tokens	16
Blacklist addresses	17
Fees and Tax	18
Lock User Funds	19
Centralization Privileges	20
Audit Results	22



Introduction

SolidProof.io is a brand of the officially registered company FutureVisions Deutschland, based in Germany. We're mainly focused on Blockchain Security such as Smart Contract Audits and KYC verification for project teams.

Solidproof.io assess potential security issues in the smart contracts implementations, review for potential inconsistencies between the code base and the whitepaper/documentation, and provide suggestions for improvement.

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 the security or functionality of the technology we agree to analyze.



Project Overview

Summary

Project Name	Vountain
Website	https://www.vountain.io/
About the project	The tokenization of real world assets and the mapping of all associated data records on a blockchain leads to an increase in transparency and authenticity, and creates seamless provenance, legal and data security, and decentralization.
Chain	TBA
Language	Solidity
Codebase	Provided as file.
Commit	N/A
Unit Tests	Not Provided

Social Medias

Telegram	N/A
Twitter	N/A
Facebook	N/A
Instagram	N/A
GitHub	N/A
Reddit	N/A
Medium	N/A
Discord	N/A
YouTube	N/A
TikTok	N/A
LinkedIn	https://www.linkedin.com/company/vountain/



Audit Summary

Version	Delivery Date	Change Log
v1.0	24. July 2024	<ul style="list-style-type: none">· Layout Project· Automated/ Manual-Security Testing· Summary

Note – The following audit report presents a comprehensive security analysis of the smart contract utilized in the project that includes outside manipulation of the contract's functions in a malicious way. This analysis did not include functional testing (or unit testing) of the contract/s logic. We cannot guarantee 100% logical correctness of the contract as we did not functionally test it. This includes internal calculations in the formulae used in the contract.



File Overview

The Team provided us with the files that should be tested in the security assessment. This audit covered the following files listed below with an SHA-1 Hash.

File Name	SHA-1 Hash
contracts/AccessControl.sol	56e66710125e4c39dd4232761865d6b39b935264
contracts/ConnectContract.sol	aa6bee927460181cc3817ff0288e57024d150bee
contracts/Configuration.sol	9795b4a2712a210ef3f245a9cce724ac4d484e9f
contracts/CheckRights.sol	636bdd815e5ec59e1af17c043ce63d619d26c8bb
contracts/ApproveRequest.sol	eba3c80b7dc0fd9534edc262f67ba3ca83ec8006
contracts/ViolinMetadata.sol	67a42c86117f32fc7073731145620b880fc6bd55
contracts/Violin.sol	36b3e094e4a54654a2665feb8f26b9f090b32507
contracts/TypeLibrary.sol	65d249989263657f8dced639ce607872bb772767
contracts/RequestHandling.sol	932aea045eacf15e359e3f8a7b76a8e15a7e0a79
contracts/RequestCreation.sol	2b8cab2436ef44d907403db60ead8d8d9f7c06cf
contracts/MoveRoleOwnership.sol	e2234f217361e714221d6ab54a905db56396f8d5

Please note: Files with a different hash value than in this table have been modified after the security check, either intentionally or unintentionally. A different hash value may (but need not) be an indication of a changed state or potential vulnerability that was not the subject of this scan.



Imported packages.

Used code from other Frameworks/Smart Contracts.

Dependency / Import Path	Count
@openzeppelin/contracts/access/Ownable.sol	8
@openzeppelin/contracts/token/ERC721/extensions/ERC721Enumerable.sol	2
@openzeppelin/contracts/utils/Base64.sol	2
@openzeppelin/contracts/utils/Counters.sol	1
@openzeppelin/contracts/utils/Strings.sol	2

Note for Investors: We only audited contracts mentioned in the scope above. All contracts related to the project apart from that are not a part of the audit, and we cannot comment on its security and are not responsible for it in any way.



External/Public functions

External/public functions are functions that can be called from outside of a contract, i.e., they can be accessed by other contracts or external accounts on the blockchain. These functions are specified using the function declaration's external or public visibility modifier.

State variables

State variables are variables that are stored on the blockchain as part of the contract's state. They are declared at the contract level and can be accessed and modified by any function within the contract. State variables can be needed within visibility modifier, such as public, private or internal, which determines the access level of the variable.

Components

 Contracts	 Libraries	 Interfaces	 Abstract
10	1	0	7

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			
97	0			
External	Internal	Private	Pure	View
33	77	0	4	42

StateVariables

Total	 Public
29	16

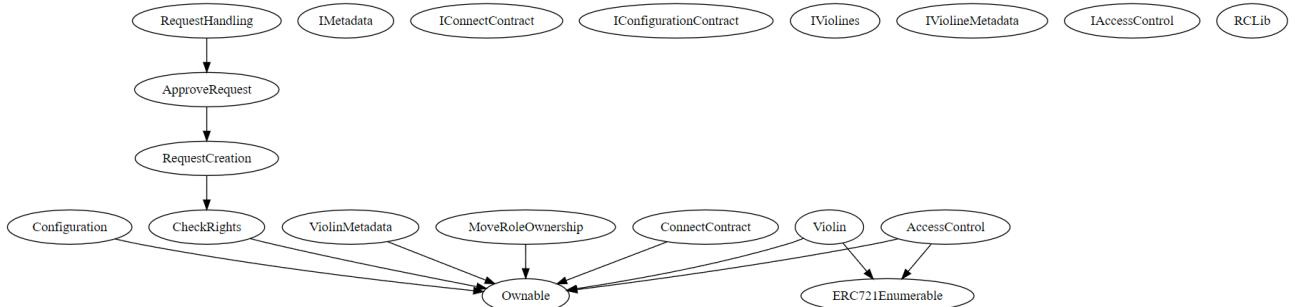


Capabilities

Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
0.8.17	-----			-----

Inheritance Graph

An inheritance graph is a graphical representation of the inheritance hierarchy among contracts. In object-oriented programming, inheritance is a mechanism that allows one class (or contract, in the case of Solidity) to inherit properties and methods from another class. It shows the relationships between different contracts and how they are related to each other through inheritance.



Audit Information

Vulnerability & Risk Level

Risk represents the probability that a certain source threat will exploit the vulnerability and the impact of that event on the organization or system. The 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 has informational character but is not affecting 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 check the repository for security-related issues, code quality, and compliance with specifications and best practices. To this end, our team of experienced pen-testers and smart contract developers reviewed the code line by line and documented any issues discovered.

We check every file manually. We use automated tools only so that they help us achieve faster and better results.

Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
 - a. Reviewing the specifications, sources, and instructions provided to SolidProof to ensure we understand the size, scope, and functionality of the smart contract.
 - b. Manual review of the code, i.e., reading the source code line by line to identify potential vulnerabilities.
 - c. Comparison to the specification, i.e., verifying that the code does what is described in the specifications, sources, and instructions provided to SolidProof.
2. Testing and automated analysis that includes the following:
 - a. Test coverage analysis determines whether test cases cover code and how much code is executed when those test cases are executed.
 - b. Symbolic execution, which is analysing a program to determine what inputs cause each part of a program to execute.
3. Review best practices, i.e., review smart contracts to improve efficiency, effectiveness, clarity, maintainability, security, and control based on best practices, recommendations, and research from industry and academia.
4. Concrete, itemized and actionable recommendations to help you secure your smart contracts.



Overall Security Upgradeability

Contract is not an upgradable

 Deployer cannot update the contract with new functionalities.

Description	The contract is not an upgradeable contract. The Deployer is not able to change or add any functionalities to the contract after deploying.
Comment	N/A





Ownership

Contract ownership is not renounced.

 The ownership is not renounced.

Description

The owner has not renounced the ownership that means that the owner retains control over the contract's operations, including the ability to execute functions that may impact the contract's users or stakeholders. This can lead to several potential issues, including:

- Centralizations
- The owner has significant control over contract's operations.

Comment

N/A

Note – *The contract cannot be considered as renounced till it is not deployed or having some functionality that can change the state of the contract.*



Ownership Privileges

These functions can be dangerous. Please note that abuse can lead to financial loss. We have a guide where you can learn more about these Functions.

Minting tokens

Minting tokens refer to the process of creating new tokens in a cryptocurrency or blockchain network. This process is typically performed by the project's owner or designated authority, who has the ability to add new tokens to the network's total supply.

Contract owner cannot mint new tokens.

 **The owner cannot mint new tokens.**

Description	The owner is not able to mint new tokens once the contract is deployed.
Comment	The latest controller contract can mint new violin NFTs in the contract.



Burning tokens

Burning tokens is the process of permanently destroying a certain number of tokens, reducing the total supply of a cryptocurrency or token. This is usually done to increase the value of the remaining tokens, as the reduced supply can create scarcity and potentially drive up demand.

Contract owner cannot burn tokens	 The owner cannot burn tokens.
Description	The owner is not able burn tokens without any allowances.
Comment	N/A



Blacklist addresses

Blacklisting addresses in smart contracts is the process of adding a certain address to a blacklist, effectively preventing them from accessing or participating in certain functionalities or transactions within the contract. This can be useful in preventing fraudulent or malicious activities, such as hacking attempts or money laundering.

Contract owner cannot blacklist addresses.

 **The owner cannot blacklist wallets.**

Description	The owner cannot blacklist wallets from transferring of tokens.
Comment	N/A



Fees and Tax

In some smart contracts, the owner or creator of the contract can set fees for certain actions or operations within the contract. These fees can be used to cover the cost of running the contract, such as paying for gas fees or compensating the contract's owner for their time and effort in developing and maintaining the contract.

Contract owner cannot set fees more than 25%.



The owner cannot set fees more than 25%.

Description	The owner cannot set fees of more than 25%.
Comment	N/A



Lock User Funds

In a smart contract, locking refers to the process of restricting access to certain tokens or assets for a specified period of time. When token or assets are locked in a smart contract, they cannot be transferred or used until the lock-up period has expired or certain conditions have been met.

Contract owner can lock function.

 The owner can lock function.

Description	The owner can lock the contract.
Comment	The contract contains the functionality in which only the allowed contract or wallet can transfer the tokens which is not recommended as this can stuck the user tokens for an indefinite period of time. The transfer functionality should be present for the user so that they can transfer their tokens to other wallets.



Centralization Privileges

Centralization can arise when one or more parties have privileged access or control over the contract's functionality, data, or decision-making. This can occur, for example, if the contract is controlled by a single entity or if certain participants have special permissions or abilities that others do not.

In the project, there are authorities that have access to the following functions:

File	Privileges
ConnectContract	<ul style="list-style-type: none">➤ The owner can update the violin contract address only once.➤ The owner can update the contact combination in the contract only when the version is not active.➤ The owner can activate the version.➤ The owner can deactivate any version.
Configuration	<ul style="list-style-type: none">➤ The owner can update the configuration only when the current configuration is not live.➤ The owner can activate/deactivate any version in the contract.
AccessControl	<ul style="list-style-type: none">➤ The controller contract can change the metadata in the contract.➤ The controller contract can update the validity of the token.➤ The controller contract can provide the role of minting NFTs.➤ The controller contract can burn the tokens from the other wallets.➤ The controller contract can burn all the tokens from the contract.
ViolinMetadata	<ul style="list-style-type: none">➤ The owner can update the base URL.➤ The controller contract can update the owner of the token.➤ The controller contract can update the manager of the token.➤ The controller contract can update the exhibitor of the token.➤ The controller contract can update the new artist of the token.➤ The controller contract can update the new violin maker address in the contract.



Violin

- The controller contract can update the new violin location in the contract.
- The controller contract can create a new document in the contract.
- The controller contract can create a new event.
- The controller contract can update the metadata.
- The latest version of the controller contract can mint new NFTs in the contract.

Recommendations

To avoid potential hacking risks, it is advisable for the client to manage the private key of the privileged account with care. Additionally, we recommend enhancing the security practices of centralized privileges or roles in the protocol through a decentralized mechanism or smart-contract-based accounts, such as multi-signature wallets.

Here are some suggestions of what the client can do:

- Consider using multi-signature wallets: Multi-signature wallets require multiple parties to sign off on a transaction before it can be executed, providing an extra layer of security e.g. Gnosis Safe
- Use of a timelock at least with a latency of e.g. 48-72 hours for awareness of privileged operations
- Introduce a DAO/Governance/Voting module to increase transparency and user involvement
- Consider Renouncing the ownership so that the owner cannot modify any state variables of the contract anymore. Make sure to set up everything before renouncing.



Audit Result

Critical Issues

No critical issues

High Issues

No high issues

Medium Issue

#1 | User cannot transfer tokens.

File	Severity	Location	Status
AccessControl	Medium	L365-386	Open

Description – The contract contains the functionality in which only the allowed contract or wallet can transfer the tokens which is not recommended as this can stuck the user tokens for an indefinite period of time. The transfer functionality should be present for the user so that they can transfer their tokens to other wallets.

#2 | The controller can burn tokens.

File	Severity	Location	Status
AccessControl	Medium	L297-310	Open

Description – The controller contract can burn tokens from the target account which is not recommended as this can cause the loss of token for the user if the token is burned from their wallet without allowance. There must be an allowance so that the tokens cannot be burned form other wallets without any allowance.



Low Issue

#1 | Missing events.

File	Severity	Location	Status
ConnectContract	Low	L25-29, L81-83	Open
Configuration	Low	L517-520, L526-528	Open

Description – It is recommended to emit all the critical parameter changes.

#2 | Local variable shadowing.

File	Severity	Location	Status
RequestCreation	Low	L26	Open
RequestHandling	Low	L10	Open
ApproveRequest	Low	L11	Open

Description – Rename the components that shadow another component.

Informational Issue

No informational issues

Legend for the Issue Status

Attribute or Symbol	Meaning
Open	The issue is not fixed by the project team.
Fixed	The issue is fixed by the project team.
Acknowledged(ACK)	The issue has been acknowledged or declared as part of business logic.



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

MADE IN GERMANY