



# SOLIDProof

*Bring trust into your projects*

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

MADE IN GERMANY

## Refi Hub

# AUDIT

SECURITY ASSESSMENT

**20. November, 2023**

FOR

# ReFi Hub



**SolidProof\_io**



**@solidproof\_io**

Introduction	4
Disclaimer	4
Project Overview	5
Summary	5
Social Medias	5
Audit Summary	6
File Overview	7
Imported packages	8
Audit Information	9
Vulnerability & Risk Level	9
Auditing Strategy and Techniques Applied	10
Methodology	10
Overall Security	11
Upgradeability	11
Ownership	12
Ownership Privileges	13
Minting tokens	13
Burning tokens	14
Blacklist addresses	15
Fees and Tax	16
Lock User Funds	17
Components	18
Exposed Functions	18
StateVariables	18
Capabilities	19
Inheritance Graph	20
Centralization Privileges	21
Audit Results	22
Critical issues	22
High issues	22



Medium issues	23
Low issues	24
Informational issues	25



## Introduction

[SolidProof.io](#) is a brand of the officially registered company MAKE Network GmbH, 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	Refi Hub
Website	<a href="https://www.refihub.io/">https://www.refihub.io/</a>
About the project	Reefihub is an impact investment project that operates on multiple chains using a master-slave pattern.
Chain	Moonbeam, Polygon, Celo
Language	Solidity
Codebase Link	Provided as Files in a Private Repo
Commit	N/A
Unit Tests	Provided

## Social Medias

Telegram	N/A
Twitter	<a href="https://twitter.com/CoralTribeNFT">https://twitter.com/CoralTribeNFT</a>
Facebook	N/A
Instagram	<a href="https://www.instagram.com/deus.natura_/">https://www.instagram.com/deus.natura_/</a>
Github	N/A
Reddit	N/A
Medium	<a href="https://medium.com/@coraltribe">https://medium.com/@coraltribe</a>
Discord	<a href="https://discord.com/invite/B7f8Vx2ZFK">https://discord.com/invite/B7f8Vx2ZFK</a>
Youtube	<a href="https://www.youtube.com/@coraltribe">https://www.youtube.com/@coraltribe</a>
TikTok	N/A
LinkedIn	<a href="https://www.linkedin.com/company/82225165/admin/feed/posts/">https://www.linkedin.com/company/82225165/admin/feed/posts/</a>

## Audit Summary

Version	Delivery Date	Changelog
v1.0	20. November 2023	<ul style="list-style-type: none"> <li>• Layout Project</li> <li>• Automated- /Manual-Security Testing</li> <li>• Summary</li> </ul>

**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/BaseVault.sol	f04beb5665eba9ddd0dd904ee65d46aead7aa99e
contracts/DepositCollector.sol	21e0738d0e09e437d1d373aec2a3faf27947b714
contracts/UniqueIdentity.sol	d5e454f586c423b77538ef00fa399d8beea43bc8
contracts/SideVault.sol	ffe3939d539338432cb54ba7891bb0509a9e01de
contracts/wormhole/IWormholeReceiver.sol	4b57d43d7e2be64c473289a7152e5893e25f7232
contracts/wormhole/IWormholeRelayer.sol	2d7496cbeee3a15a117bf93be2e6d9583213d614
contracts/MainVault.sol	15d5344bca88decd5c6eab256924fe4fdf3b03e4

*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 (direct imports).*

Dependency / Import Path	Count
@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol	1
@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol	1
@openzeppelin/contracts/access/Ownable.sol	1
@openzeppelin/contracts/interfaces/IERC20.sol	2
@openzeppelin/contracts/token/ERC20/ERC20.sol	1
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol	2
@openzeppelin/contracts/token/ERC721/extensions/ERC721Enumerable.sol	1
@openzeppelin/contracts/utils/cryptography/ECDSA.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



## Audit Information

### 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. The 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 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 an upgradeable**

**✗ Deployer can update the contract with new functionalities**

Description

The deployer can replace the old contract with a new one with new features. Be aware of this, because the owner can add new features that may have a negative impact on your investments.

Example

We assume that you have funds in the contract and it has been audited by any security audit firm. Now the audit has passed. After that, the deployer can upgrade the contract to allow him to transfer the funds you purchased without any approval from you. This has the consequence that your funds can be taken by the creator.

Comment

N/A

## Ownership

**The ownership is not renounced**

**✗ The owner is not renounce**

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** - If the contract is not deployed then we would consider the ownership to be not renounced. Moreover, if there are no ownership functionalities then the ownership is automatically considered renounced.




## 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	
<div>  <b>The owner cannot mint new tokens</b> </div>	
Description	The owner is not able to mint new tokens once the contract is deployed.
Comment	Owners who have the ability to mint new KYC tokens to any account without paying the minting fees

## 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 can burn tokens

**✗ The owner able to burn tokens**

Description	The owner has the ability to burn the KYC tokens of the users at any time without the need of their allowance
Comment	N/A

### File: UniquelIdentity.sol Codebase:

```

116     function burn(
117         address account↑,
118         uint256 expiresAt↑,
119         bytes calldata signature↑
120     ) public onlySigner(account↑, tokenOfOwnerByIndex(account↑, 0), expiresAt↑, signature↑) {
121         require(balanceOf(account↑) == 1, "Balance before burn must be 1");
122         _burn(tokenOfOwnerByIndex(account↑, 0));
123         require(balanceOf(account↑) == 0, "Balance after burn must be 0");
124     }
125

```

**Alleviation** - This is actually a requirement of the KYC NFT contract. KYC is done off chain via a KYC provider and then the NFT is issued for the KYC privilege. If the KYC becomes invalid (KYC provider finds a problem with the user etc.) then NFT needs to be burned to revoke access to the protocol.



## 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 addresses**

Description

The owner is not able blacklist addresses to lock funds.

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 levy unfair taxes**

Description

The owner is not able to set the fees above 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 tokens 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.*

### Owner cannot lock the contract



**The owner cannot lock the contract**

Description

The owner is not able to lock the contract by any functions or updating any variables.

Comment

N/A

## 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 defined with a visibility modifier, such as public, private, or internal, which determines the access level of the variable.

## Components

 <b>Contracts</b>	 <b>Libraries</b>	 <b>Interfaces</b>	 <b>Abstract</b>
4	0	6	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.

 <b>Public</b>	 <b>Payable</b>
53	21

External	Internal	Private	Pure	View
36	29	2	2	19

## StateVariables

Total	 <b>Public</b>
39	39



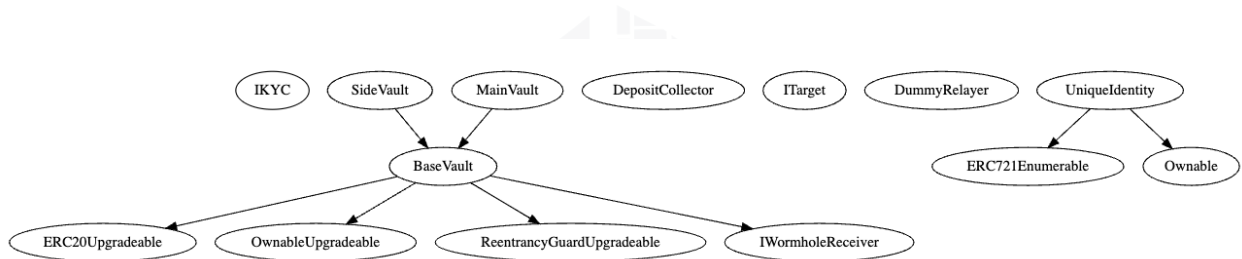
## Capabilities

Solidity Versions observed	Transfers ETH	 Can Receive Funds	 Uses Assembly	 Has Destroyable Contracts
0.8.18	Yes	Yes		



## 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.



## 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 a single entity controls the contract or if certain participants have special permissions or abilities that others do not.*

In the project, some authorities have access to the following functions:

File	Privileges
BaseVault	<ul style="list-style-type: none"> <li>onlyOwner can set the signer address</li> </ul>
MainVault	<ul style="list-style-type: none"> <li>onlyOwner can set side vault address for different chains</li> </ul>
UniqueIdentity	<ul style="list-style-type: none"> <li>onlyOwner can set the signer address and Base URI</li> <li>The signer address can mint KYC tokens without paying and also the signer address can burn KYC tokens of the users at any time</li> </ul>

## Recommendations

To avoid potential hacking risks, the client should 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.



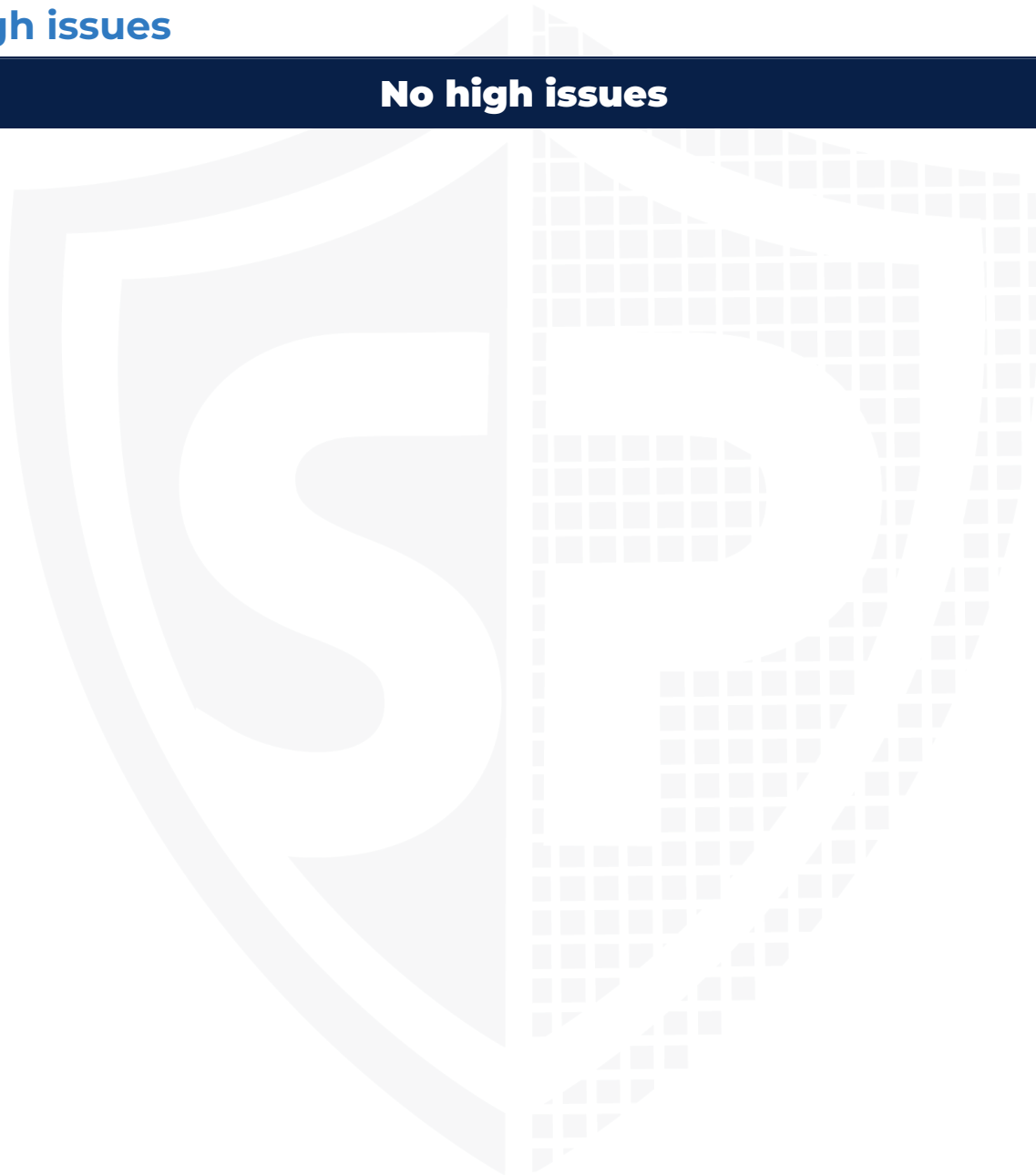
## Audit Results

### Critical issues

**No critical issues**

### High issues

**No high issues**



## Medium issues

### #1 | Owner can Burn Tokens

File	Severity	Location	Status
UniquelIdentity	Medium	L88, 116	ACK

**Description** - The contract owner can set any address as the signer address, and this signer address has the privilege to burn the KYC token of any address without approval.

**Remediation** - We recommend not burning tokens without the token owner's approval.

**Alleviation** - This is actually a requirement of the KYC NFT contract. KYC is done off chain via a KYC provider and then the NFT is issued for the KYC privilege. If the KYC becomes invalid (KYC provider finds a problem with the user etc.) then NFT needs to be burned to revoke access to the protocol.

## Low issues

### #1 | Missing Address Validation

File	Severity	Location	Status
SideVault	Low	L18,19,22	Fixed

**Description** - Make sure to validate that the address passed in the function parameters is a valid contract address.

### #2 | Missing Events

File	Severity	Location	Status
UniquelIdentity	Low	L197	Fixed

**Description** - Make sure to emit events for all the critical parameter changes in the contract to ensure the transparency and trackability of all the state variable changes.

### #3 | Missing Value Checks

File	Severity	Location	Status
SideVault	Low	L79—84	Fixed

**Description** - The contract should check that the minimum value must not exceed the maximum collected value. Similarly, the sale's start time must be less than the end time, and they should be non-zero. Moreover, The Beneficiary fees should have a hard cap. Otherwise, it could be set to any arbitrary value.



## Informational issues

### #1 | State Variable Visibility

File	Severity	Location	Status
BaseVault	Informational	L170	Fixed

**Description** - Ensure that the visibility of all the variables is marked explicitly.

### #2 | Disable initializing

File	Severity	Location	Status
SideVault MainVault	Informational	N/A	ACK

**Description** - If the owner updates the contract, a disableInitializer call in the constructor must be implemented. This prevents calling the initialize function again to set the state variables in the contract. This should be implemented only if the contract was deployed before. Otherwise, the owner cannot call the initialize function to set the variables.

**Recommendation** - If the contract hasn't been deployed, remove the disableInitializer in the constructor. Otherwise, you are not able to initialize the contract. When the contract has a deployed version already, leave it as it is.

### 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