



# SOLIDProof

*Bring trust into your projects*

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

MADE IN GERMANY

# Sheertopia

# AUDIT

SECURITY ASSESSMENT

## 19. March, 2024

FOR



SHEERTOPIA



**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        | 22 |
| Low issues           | 23 |
| Informational issues | 24 |





## 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      | Sheertopia  |
|-------------------|---|
| Website           | <a href="https://sheertopia.io/">https://sheertopia.io/</a>   |
| About the project | Sheertopia is a play-to-earn NFT game, blending engaging gameplay with blockchain technology for a unique, immersive experience.  |
| Chain             | Polygon   |
| Language          | Solidity  |
| Codebase Link     | <a href="https://polygonscan.com/token/0x9B8B6A1298d34B3c4bBDDce8a7fF0149121592C1#code">https://polygonscan.com/token/0x9B8B6A1298d34B3c4bBDDce8a7fF0149121592C1#code</a> |
| Commit            | N/A   |
| Unit Tests        | Not Provided  |

## Social Medias

| Telegram  | <a href="https://t.me/sheertopia">https://t.me/sheertopia</a>                             |
|-----------|---|
| Twitter   | <a href="https://twitter.com/sheertopia">https://twitter.com/sheertopia</a>               |
| Facebook  | N/A   |
| Instagram | <a href="https://www.instagram.com/sheertopia/">https://www.instagram.com/sheertopia/</a> |
| Github    | N/A   |
| Reddit    | N/A   |
| Medium    | <a href="https://medium.com/feed/@sheertopia">https://medium.com/feed/@sheertopia</a>     |
| Discord   | <a href="https://discord.com/invite/CQACamZERw">https://discord.com/invite/CQACamZERw</a> |
| Youtube   | N/A   |
| TikTok    | <a href="https://www.tiktok.com/@sheertopia">https://www.tiktok.com/@sheertopia</a>       |
| LinkedIn  | N/A   |

## Audit Summary

| Version | Delivery Date    | Changelog  |
|---------|------------------|--|
| v1.0    | 19. October 2023 | <ul style="list-style-type: none"> <li>• Layout Project</li> <li>• Automated- /Manual-Security Testing</li> <li>• Summary</li> </ul> |
| v1.1    | 19. March 2024   | <ul style="list-style-type: none"> <li>• Reaudit</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/Ambo.sol | bdd5ce172f905694e6bf2613b9609d72f9e08edf |

*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) indicate 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/access/Ownable.sol                       | 1     |
| @openzeppelin/contracts/token/ERC20/ERC20.sol                    | 1     |
| @openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.sol | 1     |
| @openzeppelin/contracts/token/ERC20/extensions/ERC20Permit.sol   | 1     |
| @openzeppelin/contracts/token/ERC20/extensions/ERC20Votes.sol    | 1     |
| @openzeppelin/contracts/utils/Nonces.sol                         | 1     |

**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 not an upgradeable**



**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

**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

The contract's owner is set to the zero address but the moderator address has the same privileges as the contract's owner and the moderator address can make state changes in the smart contract.

**Note** - If the contract is not deployed, we would consider the ownership not renounced. Moreover, if there are no ownership functionalities, 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

 **The owner cannot mint new tokens**

|             |   |
|-------------|---|
| Description | The owner is not able to mint new tokens once the contract is deployed. |
|-------------|---|

|         |     |
|---------|-----|
| Comment | N/A |
|---------|-----|



## 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 can blacklist addresses

**✗ The owner able to blacklist users**

#### Description

If the owner or developers of the smart contract abuse their power to blacklist addresses without proper justification or transparency, it can lead to a decrease in trust and credibility in the project. For example, suppose an owner or developer blacklists an address without proper explanation or communication to stakeholders. In that case, it can create speculation and uncertainty among investors, potentially causing them to sell their tokens and decreasing the token's value.

Furthermore, if the owner or developers have a significant number of tokens themselves and use their power to blacklist competitors or manipulate the market, it can lead to an unfair advantage and concentration of power, potentially harming the interests of other stakeholders.

Therefore, it is important for projects and platforms to be transparent about their blacklisting practices and ensure that they are justified and in the best interest of their stakeholders. Investors should carefully research the project and its blacklisting practices and exercise caution before investing in any cryptocurrency or DeFi project to avoid potential losses.

#### Comment

N/A

**Codebase:** Ambo.sol, L148

```

148     function addToBlackList(address _addr↑)
149         external
150         _validAddress(_addr↑)
151         _moderatorOrOwner
152     {
153         blackList[_addr↑] = true;
154     }

```



## 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 restricting access to certain tokens or assets for a specified period. 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.*

| Contract owner can lock the user funds |  | ✗ The owner is able to lock funds |
|--|--|-----------------------------------|
| Description                            | Locking the contract means that the owner is able to lock any funds of addresses that they are not able to transfer bought tokens anymore.                           |                                   |
| Example                                | An example of locking is by pausing the contract or blacklisting any addresses. That causes that the blacklisted address is not able to transfer (buy/sell) anymore. |                                   |
| Comment                                | The owner can manually add/remove wallets from the blacklist.  |                                   |

## 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> |
|--|--|---|--|
| 1  | 0  | 0   | 0  |


## 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> |
|---|--|
| 13  | 1  |

| <b>External</b> | <b>Internal</b> | <b>Private</b> | <b>Pure</b> | <b>View</b> |
|-----------------|-----------------|----------------|-------------|-------------|
| 10              | 11              | 0              | 1           | 3           |

## StateVariables

| <b>Total</b> |  <b>Public</b> |
|--------------|---|
| 7            | 7   |



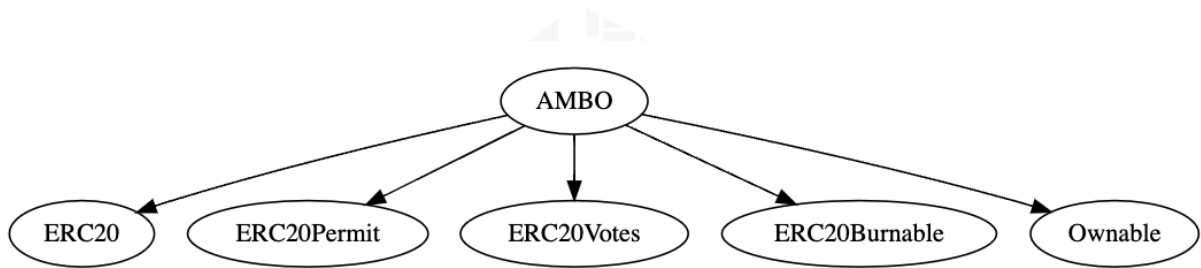
## Capabilities

| Solidity Versions observed | Transfers ETH | Can Receive Funds | Uses Assembly | Has Destroyable Contracts |
|----------------------------|---------------|-------------------|---------------|---------------------------|
| 0.8.23                     | 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   |
|------|--|
| Ambo | <ul style="list-style-type: none"> <li>• Moderator and Owner address               <ul style="list-style-type: none"> <li>• Add/Remove wallets from the blacklist</li> <li>• Enable/Disable limits and blacklist</li> <li>• Set Moderator Address</li> <li>• Set Max Transaction and Holding Amount</li> </ul> </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 can no longer modify any state variables of the contract. Make sure to set up everything before renouncing.



## Audit Results

### Critical issues

**No critical issues**

### High issues

**No high issues**

### Medium issues

**No medium issues**

## Low issues

---

### #1 | Missing Events

| File | Severity | Location | Status |
|------|----------|----------|--------|
| Ambo | Low      | L123—168 | ACK    |

**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.





## Informational issues

### No Informational issues

#### Legend for the Issue Status

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





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

MADE IN GERMANY