SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY

Versailles Heroes

Audit

Security Assessment

16. September, 2022

For

SolidProof_io

@solidproof_io

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	16. September 2022	• Layout project • Automated- /Manual-Security Testing • Summary

Network
Ethereum (ERC20)

Website
https://versaillesheroes.com/

Telegram
https://t.me/VRHannouncement

Twitter
https://twitter.com/VersHeroes

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

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

Description

Created by experienced game developers and blockchain experts, Versailles Heroes is the play-to-earn game for everyone to enjoy that combines crypto-economics with the gaming world through fascinating storytelling, NFT game art, and blockchain technology. Start your Versailles adventure with one of our NFT heroes and earn your rewards today!

Project Engagement

During the 14th of September 2022, **Versailles Heroes 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

- https://etherscan.io/address/0xeAd482da0793B00bbAe0E34C8cfaE6DAf29a44b2#code

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)

Imported packages:
ERC20

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

Filename	SHA-1 Hash
vyper_contract.vy	6edfefba271ceaf832b451d05d8cbccd75107e55

Metrics

Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	1	0	1	0

Exposed Functions

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

Version	External	Internal	Pure	View	Payable
1.0	13	3	0	3	0

State Variables

Version	Total	Public
1.0	19	11

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. Deployer cannot mint any new tokens
3. Deployer cannot burn or lock user funds
4. Deployer cannot pause the contract
5. Deployer cannot set fees
6. Deployer cannot blacklist/antisnipe addresses
7. Overall checkup (Smart Contract Security)

Correct implementation of Token standard

ERC20				
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

1. update_mining_parameters
2. start_epoch_time_write
3. future_epoch_time_write
4. transfer
5. transferFrom
6. approve
7. set_minter
8. set_admin
9. mint
10. burn

Deployer cannot mint any new tokens

Name	Exist	Tested	Status
Deployer cannot mint	✓	✓	✗
Max / Total Supply	727_200_000		

Comments:

v1.0

- Owner can set new minter (only once) that can mint new tokens

Deployer cannot burn or lock user funds

Name	Exist	Tested	Status
Deployer cannot lock	✓	✓	✓
Deployer cannot burn	✓	✓	✗

Comments:

v1.0

- Tokens
  - can be burned by msg.sender

Deployer cannot pause the contract

Name	Exist	Tested	Status
Deployer cannot pause	–	–	–

Deployer cannot set fees

Name	Exist	Tested	Status
Deployer cannot set fees over 25%	–	–	–
Deployer cannot set fees to nearly 100% or to 100%	–	–	–

Deployer can blacklist/antisnipe addresses

Name	Exist	Tested	Status
Deployer cannot blacklist/antisnipe addresses	–	–	–

Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

Legend

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

Asserts and external functions

v1.0

- assert block.timestamp >= self.start_epoch_time + RATE_REDUCTION_TIME, L95
- assert start <= end, L199
- assert end <= current_epoch_time + RATE_REDUCTION_TIME, L209
- assert current_rate <= INITIAL_RATE, L230
- assert msg.sender == self.admin, L237 L245
- assert self.minter == ZERO_ADDRESS, L238
- assert msg.sender == self.minter, L259
- assert _to != ZERO_ADDRESS, L260 L283
- assert _total_supply <= self._available_supply(), L266

1. update_mining_parameters
2. start_epoch_time_write
3. future_epoch_time_write
4. transfer
5. transferFrom
6. approve
7. set_minter
8. set_admin
9. mint
10. burn

Comments

- *Deployer can set following addresses*
  - minter
  - admin

- Anyone can call and update the mining parameters after a certain time (block.timestamp >= self.start_epoch_time + RATE_REDUCTION_TIME)
- Only minter can mint new tokens. The owner is not able to do it

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

Source Units in Scope

v1.0

Lines	nLines	nSLOC	Comment lines
294	236	181	55

Legend

Attribute	Description
Lines	total lines of the source unit
nLines	normalised lines of the source unit (e.g. normalises functions spanning multiple lines)
nSLOC	normalised 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

Critical issues

No critical issues

High issues

No high issues

Medium issues

No medium issues

Low issues

Issue	File	Type	Line	Description
#1	Main	Missing Zero Address Validation (missing-zero-check)	140, 155, 175, 236, 244	Check that the address is not zero If you are checking for admin address is zero then we would recommend you to implement a function to renounce the ownership also

Informational issues

Issue	File	Type	Line	Description
#1	Main	Error message is missing	95, 199, 209, 230, 237, 238, 245, 259, 260, 266, 283	Provide an error message for assert statement
#2	Main	NatSpec documentation missing	-	If you started to comment your code, also comment all other functions, variables etc.

Audit Comments

We recommend you to use the special form of comments (NatSpec Format, Follow link for more information https://vyper.readthedocs.io/en/

v0.3.6/natspec.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.

16. September 2022:

- Read whole report and modifiers section for more information

SolidProof_io

@solidproof_io

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY