SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

Audit

Security Assessment
20. December, 2021

For

BULLIEVERSE

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	20. December 2021	• Layout project • Automated- /Manual-Security Testing • Summary

Network
Polygon

Website
https://bullieverse.com/

Telegram
https://t.me/Bullieverse

Twitter
https://twitter.com/Bullieverse

Discord
https://discord.com/invite/bullieverse

Description

A fantasy Metaverse Island where you can play games, create experiences and earn rewards

Project Engagement

During the 15th of December 2021, **Bullieverse 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

- TBA

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:

Dependency / Import Path	Count
@openzeppelin/contracts/access/Ownable.sol	11
@openzeppelin/contracts/token/ERC20/ERC20.sol	1
@openzeppelin/contracts/token/ERC20/IERC20.sol	11
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol	11
@openzeppelin/contracts/utils/Counters.sol	1
@openzeppelin/contracts/utils/math/SafeMath.sol	22
hardhat/console.sol	7

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/AdvisorVesting.sol	6267a0a42b28834408518f36a01930678921849c
contracts/DaoVesting.sol	211e23d23db27043f0e069da14d3e2ba7bfd76d6
contracts/PrivateSaleVesting.sol	d5f45206ff6a846bf25d4695ae3315c6ddf0f67a
contracts/RewardVesting.sol	4b8b91504756d52c31d6d7949e78baf4c3c892af
contracts/SeedVesting.sol	ace728417b1605b8410e9eab9d491363f93490e8
contracts/RetroRewardVesting.sol	9794c8264ad1283d1af447656f0ee9c8a23b3e1a
contracts/TeamVesting.sol	b21ad0fd4c92a742f90d3697aebb63c12c6d2e6b
contracts/IDOVesting.sol	1175877b44ed85f1eb3cb1541e12b94c6b2b4144
contracts/PrivateVestingOp.sol	8e5975439c80173cd6b04636762c23698c067a8f
contracts/LiquidityVesting.sol	c1dfd1e0ac3ffbe55e30e8011c3d779d2d9c309d
contracts/EcoSystemDevelopmentVesting.sol	633e6e8a808d840883ade158e150adbdb53bb4ed
contracts/lib/BokkyPooBahsDateTimeLibrary.sol	8281c2415a57c619ffd711d29c4007356d42f1fa
contracts/interface/IERC2612Permit.sol	cbb65852689c5580388c4a808d69a48f00a43b0a
contracts/interface/ERC20Permit.sol	017fe499928e0905aacb2eaea97c5951d728e669

Metrics

Source Lines
v1.0
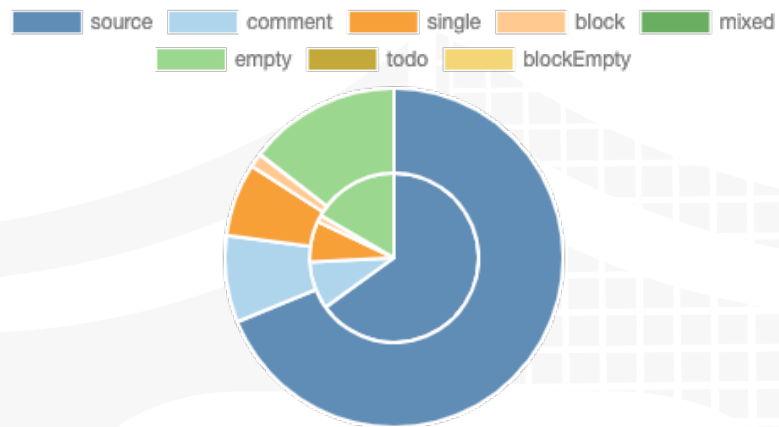
Risk Level
v1.0
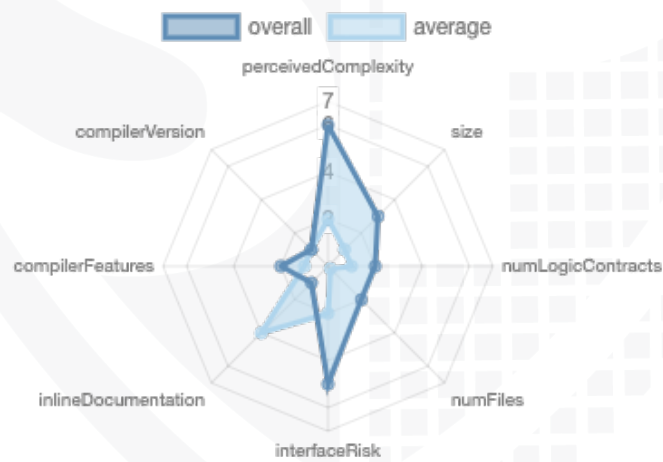
Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	11	1	1	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	70	0

Version	External	Internal	Private	Pure	View
1.0	46	119	22	50	35

State Variables

Version	Total	Public
1.0	80	35

Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	`0.8.6`			yes (1 asm blocks)	

Version	Transfers ETH	Low-Level Calls	DelegateCall	Uses Hash Functions	ECRecover	New/ Create/ Create2

1.0				yes	yes	

Scope of Work

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. Overall checkup (Smart Contract Security)

Inheritance Graph
v1.0
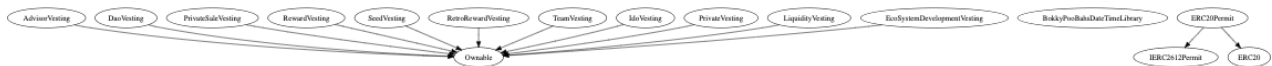
Verify Claims
Write functions of contract

addInvestors

recoverToken

renounceOwnership

setInitialTimestamp

transferOwnership

withdrawTokens

Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

Legend

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

15

Modifiers

RewardVesting

- addInvestors
  - onlyOwner
- withdrawTokens
  - onlyInvestor
  - initialized
- setInitialTimestamp
  - onlyOwner
- recoverToken
  - onlyOwner

TeamVesting

- addInvestors
  - onlyOwner
- withdrawTokens
  - onlyInvestor
  - initialized
- setInitialTimestamp
  - onlyOwner
  - notInitialized
- recoverToken
  - onlyOwner

RetroRewardVesti.

- addInvestors
  - onlyOwner
- withdrawTokens
  - onlyInvestor
  - initialized
- setInitialTimestamp
  - onlyOwner
  - notInitialized
- recoverToken
  - onlyOwner

PrivateVestingOp

- addInvestors
  - onlyOwner
- withdrawTokens
  - onlyInvestor
  - initialized
- setInitialTimestamp
  - onlyOwner
  - notInitialized
- recoverToken
  - onlyOwner

SeedVes.

- addInvestors
  - onlyOwner
- withdrawTokens
  - onlyInvestor
  - initialized
- setInitialTimestamp
  - onlyOwner
  - notInitialized
- recoverToken
  - onlyOwner

IDOVesting

- addInvestors
  - onlyOwner
- withdrawTokens
  - onlyInvestor
  - initialized
- setInitialTimestamp
  - onlyOwner
  - notInitialized
- recoverToken
  - onlyOwner

AdvisorVesting

- addInvestors
  - onlyOwner
- withdrawTokens
  - onlyInvestor
  - initialized
- setInitialTimestamp
  - onlyOwner
- recoverToken
  - onlyOwner

CallGraph

Source Units in Scope
v1.0

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝	contracts/AdvisorVesting.sol	1	—	215	193	143	13	80	—
📝	contracts/DaoVesting.sol	1	—	215	193	143	13	80	—
📝	contracts/PrivateSaleVesting.sol	1	—	215	193	137	19	82	—
📝	contracts/RewardVesting.sol	1	—	220	198	143	17	80	—
📝	contracts/SeedVesting.sol	1	—	216	194	137	19	82	—
📝	contracts/RetroRewardVesting.sol	1	—	219	197	143	16	81	—
📝	contracts/TeamVesting.sol	1	—	216	194	143	12	81	—
📝	contracts/IDOVesting.sol	1	—	213	191	138	12	82	—
📝	contracts/PrivateVestingOp.sol	1	—	238	214	155	20	87	—
📝	contracts/LiquidityVesting.sol	1	—	215	193	137	19	82	—
📝	contracts/EcoSystemDevelopmentVesting.sol	1	—	215	193	143	13	80	—
📚	contracts/lib/BokkyPooBahsDateTimeLibrary.sol	1	—	525	376	273	56	82	—
🔍	contracts/interface/IERC2612Permit.sol	—	1	54	41	8	38	5	—
🎨	contracts/interface/ERC20Permit.sol	1	—	85	77	54	10	38	🖥️🔲🖌️
📝📚🔍🎨	**Totals**	**13**	**1**	**3061**	**2647**	**1897**	**277**	**1022**	🖥️🔲🖌️

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

<div style="background-color:#5FE83A; text-align:center;">

AUDIT PASSED

</div>

Critical issues
- no critical issues found -

High issues
- no high issues found -

Medium issues
- no medium issues found -

Low issues
- no low issues found -

Informational issues

Issue	File	Type	Line	Description
#1	BokkyPooBahsDateTimeLibrary	Functions that are not used	57, 242, 219, 359, 368, 328, 377, 308, 490, 499, 508, 464, 282, 265, 231, 286, 291, 278, 300, 274, 210, 182, 195, 223, 227, 428, 437, 446, 406, 455, 386, 124, 132, 150, 162	Remove unused functions
#2	BokkyPooBahsDateTimeLibrary	Unused state variables	36, 42, 39, 37, 38	Remove unused state variables

Audit Comments

20. December 2021:

- File differences are only in the _calculateAvailablePercentage
- Token address was not provided to Solidproof

SWC Attacks

ID	Title	Relationships	Status
SWC-136	Unencrypted Private Data On-Chain	CWE-767: Access to Critical Private Variable via Public Method	PASSED
SWC-135	Code With No Effects	CWE-1164: Irrelevant Code	PASSED
SWC-134	Message call with hardcoded gas amount	CWE-655: Improper Initialization	PASSED
SWC-133	Hash Collisions With Multiple Variable Length Arguments	CWE-294: Authentication Bypass by Capture-replay	PASSED
SWC-132	Unexpected Ether balance	CWE-667: Improper Locking	PASSED
SWC-131	Presence of unused variables	CWE-1164: Irrelevant Code	NOT PASSED
SWC-130	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	PASSED
SWC-129	Typographical Error	CWE-480: Use of Incorrect Operator	PASSED
SWC-128	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	PASSED

SWC-127	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	**PASSED**
SWC-125	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	**PASSED**
SWC-124	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	**PASSED**
SWC-123	Requirement Violation	CWE-573: Improper Following of Specification by Caller	**PASSED**
SWC-122	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	**PASSED**
SWC-121	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	**PASSED**
SWC-120	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	**PASSED**
SWC-119	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	**PASSED**
SWC-118	Incorrect Constructor Name	CWE-665: Improper Initialization	**PASSED**
SWC-117	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	**PASSED**

SWC-116	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	**PASSED**
SWC-115	Authorization through tx.origin	CWE-477: Use of Obsolete Function	**PASSED**
SWC-114	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	**PASSED**
SWC-113	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	**PASSED**
SWC-112	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	**PASSED**
SWC-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	**PASSED**
SWC-110	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	**PASSED**
SWC-109	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	**PASSED**
SWC-108	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	**PASSED**
SWC-107	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	**PASSED**
SWC-106	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	**PASSED**

SWC-105	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	**PASSED**
SWC-104	Unchecked Call Return Value	CWE-252: Unchecked Return Value	**PASSED**
SWC-103	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	**PASSED**
SWC-102	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	**PASSED**
SWC-101	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	**PASSED**
SWC-100	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	**PASSED**

Solid Proofed

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY