SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

Audit

Security Assessment
30. August, 2021

For

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	30. August 2021	• Layout project • Automated- /Manual-Security Testing • Summary

Network
Binance Smart Chain (BEP20)

Website
https://rewardeum.com/

Telegram
https://t.me/rewardeum

Twitter
https://twitter.com/rewardeum

Description

Prior to Rewardeum the development team launched iBNB, their first project which aimed to be the **first dynamic DeFi token** with a reward pool generating protocol built on the Binance Smart Chain.

Rewardeum takes all of the best mechanics of iBNB, and improves on them in a significant way. They have also added new mechanics, such as the Smart Pool and their Multicoin Reward System to ensure they can offer much more consistent rewards than ever before across a variety of different coins.

Rewardeum's main aim is to ensure that **daily rewards remain consistent across fluctuating trading volume.** To achieve this, (1) their Dynamic Tax Protocol will prioritise and maximise the replenishment of the reward pools whilst only diverting taxes to the liquidity pool when required, and (2) their Smart Pool mechanism would store all excessive trading volume as reserve for the days of lower trading volume. This allows for consistent reward pool growth whilst simultaneously protecting the token from low liquidity price impact.

Project Engagement

During the 27th of August 2021, **Rewardeum 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

<u>Auditing Strategy and Techniques Applied</u>

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:
- OpenZeppelin
  - Ownable
  - ERC721Enumerable
  - IERC20
  - ERC721Holer
- Uniswap
  - UniswapV2Factory
  - UniswapV2Pair
  - UniswapV2Router01
  - UniswapV2Router02

Dependency / Import Path	Count
@openzeppelin/contracts/access/Ownable.sol	5
@openzeppelin/contracts/token/ERC20/ERC20.sol	1
@openzeppelin/contracts/token/ERC20/IERC20.sol	4
@openzeppelin/contracts/token/ERC721/Extensions/IERC721Enumerable.sol	1
@openzeppelin/contracts/token/ERC721/IERC721.sol	1
@openzeppelin/contracts/token/ERC721/extensions/ERC721Enumerable.sol	1
@openzeppelin/contracts/token/ERC721/utils/ERC721Holder.sol	1
@openzeppelin/contracts/utils/math/SafeMath.sol	1
@uniswap/v2-core/contracts/interfaces/IUniswapV2Factory.sol	2
@uniswap/v2-core/contracts/interfaces/IUniswapV2Pair.sol	2
@uniswap/v2-periphery/contracts/interfaces/IUniswapV2Router02.sol	2

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/Airdrops.sol	267141890a394ab6f7532e3bcf99401c55344d56
contracts/REUM_ticket.sol	b6ca10c123e35ff366bf722947158bea31d37314
contracts/Presale.sol	82f33231c58510c9da821ecdaa85c2721abe155a
contracts/vault.sol	b4f7f980cd49063332b2be368ec8830928c5daaf
contracts/Vault_01.sol	f4a25eeb4d94b13ca69a05b29c0a7d440e947e66
contracts/Rewardeum.sol	fb26feb28a13f3cc7740c7eb0dc671e92d0947a7

Metrics

Source Lines
v1.0
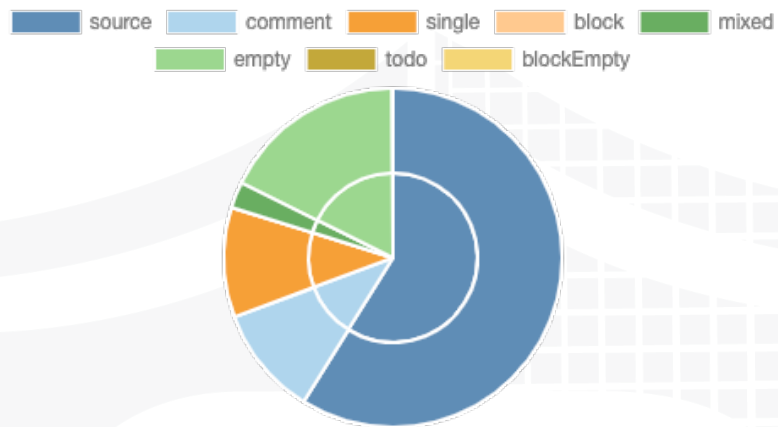
Risk Level
v1.0
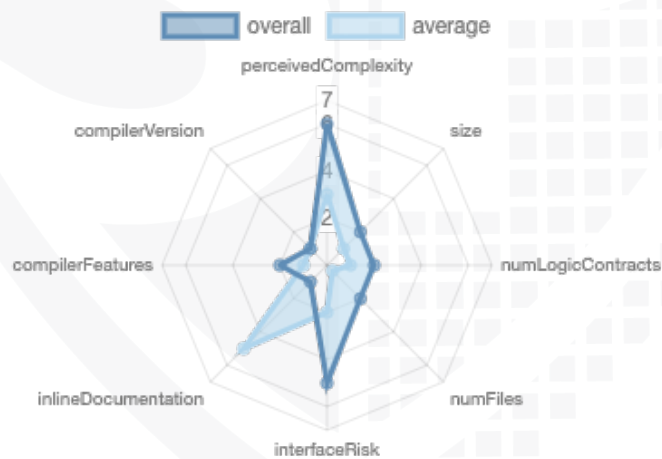
Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	6	0	2	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.

Version	Public	Payable
1.0	82	6

Version	External	Internal	Private	Pure	View
1.0	62	75	0	1	26

State Variables

Version	Total	Public
1.0	70	43

Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	`0.8.0` `^0.8.0`		yes	**** (0 asm blocks)	yes

| Version	Transfers ETH	Low-Level Calls	DelegateCall	Uses Hash Functions	ECRecover	New/ Create/ Create 2
1.0	yes			yes		yes → New Contract:R EUM_t icket

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
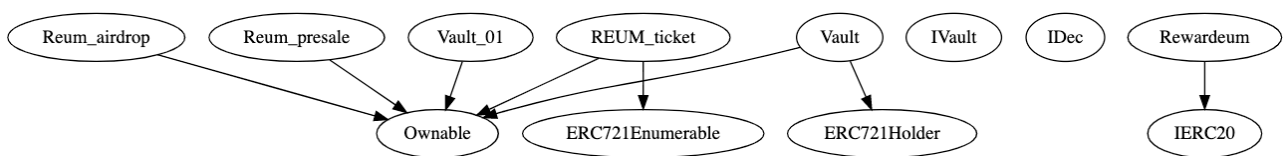
Verify Claims
Correct implementation of Token standard

Tested	Verified
✓	✓

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	✓	✓	✓

Optional implementations

Function	Description	Exist	Tested	Verified
renounceOwnership	Owner renounce ownership for more trust	–	–	–

Comment
- Owner can be set with addOwner
- Owner can be removed with removeOwner
- Multiownership is possible in contract

Deployer cannot mint any new tokens

	Name	Exist	Tested	Verified	File
Rewardeum.sol	Deployer cannot mint	–	–	–	Main
	Comment	Line: -			
REUM_Ticket.sol	Deployer cannot mint	✓	✓	✗	Main
	Comment	Line: -			

Max / Total Supply: 1.000.000.000.000.000

Comments:
v1.0
- Rewardeum.sol
    - Deployer cannot mint
- REUM_Ticket.sol
    - Deployer can mint
    - mintTicket function is called by Vault_01.sol file in line 32

Deployer cannot burn or lock user funds

Name	Exist	Tested	Verified
Deployer cannot lock	✓	✓	✓
Deployer cannot burn	✓	✓	✓

Comments:
v1.0

- Burn tokens
    - When the function concludeAndAddLiquidity function is called, liquidity tokens will burned
    - File: Presale

```
205    router.addLiquidityETH{value: balance_BNB}(
206        address(token_interface),
207        balance_token,
208        balance_token - (balance_token * emergency_slippage / 100),
209        balance_BNB,
210        address(0x000000000000000000000000000000000000dEaD), //liquidity tokens are burned
211        block.timestamp
212    );
```

.

Deployer cannot pause the contract

Name	Exist	Tested	Verified
Deployer cannot pause	✓	✓	✓

Comments:

v1.0

.

# Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

Legend

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

CallGraph

Source Units in Scope
v1.0

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝	contracts/Airdrops.sol	1	———	25	25	16	3	17	⚫
📝	contracts/REUM_ticket.sol	1	———	67	67	42	9	42	💰
📝	contracts/Presale.sol	1	———	242	242	159	42	151	💰⚫🔁
📝	contracts/vault.sol	1	———	73	73	50	9	62	💰🔁
📝	contracts/Vault_01.sol	1	———	52	52	31	11	38	💰🌀
📝🔍	contracts/Rewardeum.sol	1	2	863	856	619	90	524	💰🎟️☀️
📝🔍	**Totals**	**6**	**2**	**1322**	**1315**	**917**	**164**	**834**	💰⚫🔁🎟️🌀☀️

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

AUDIT PASSED

Critical issues
- no critical issues found -

High issues
- no high issues found -

Medium issues
- no medium issues found -

Low issues

Issue	File	Type	Line	Description
#1	Main	A floating pragma is set		The current pragma Solidity directive is „"^0.""".
#2	Presale	Missing Zero Address Validation (missing-zero-check)	218	Check that the address is not zero
#3	Reward eum	Missing Zero Address Validation (missing-zero-check)	276, 752, 748, 756	Check that the address is not zero
#4	Vault	Missing Zero Address Validation (missing-zero-check)	20	Check that the address is not zero
#5	Reward eum	State variable visibility is not set	95	It is best practice to set the visibility of state variables explicitly
#6	Airdrop s	Unused state variables (unused-state)	12	Remove unused state variables

Informational issues

Issue	File	Type	Line	Description
#1	Vault	Missing inheritance (missing-inheritance Should inherit from IVault	15-74	Inherit from the missing interface or contract
#2	REUM_ticket	Costly operations in a loop (costly-loop)	27-40	Use a local variable to hold the loop computation result and set after loop is finished • _tokenIds

Audit Comments
30. August 2021:
- Vault.sol
  - Receive function marked as TODO
- Deployer can set claim periodic time
- Deployer can set claiming taxes rates
- Deployer can set multiple ownerships

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](#)	**NOT 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	**NOT 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