



SOLIDProof

Blockchain Security | Smart Contract Audits

MADE IN GERMANY

Audit Passed

Security Assessment
7. July, 2021

For



Disclaimer	3
Description	5
Project Engagement	5
Logo	5
Contract Link	5
Methodology	7
Used Code from other Frameworks/Smart Contracts (direct imports)	8
Source Lines	9
Risk Level	9
Capabilities	9
CallGraph	10
Source Units in Scope	10
Critical issues	11
High issues	11
Medium issues	11
Low issues	11
Informational issues	11
SWC Attacks	12

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.

Overview

Network

Binance Smart Chain (BEP20)

Website

<https://www.cashaperise.io/>

Telegram

<https://t.me/CashApeRise>

<https://t.me/CashApeRiseChat>

Twitter

<https://twitter.com/CashApeRise?s=09>

Facebook

<https://facebook.com/CashApeRise>

E-Mail

contact@cashaperise.io

Description

CashApeRise (P2P Cash Game Utility Fuel Token) is a decentralized and fair Bep20-based Token supported by holders, community, and players.

Users will have a chance to stake their tokens and get a portion of games fees, tax from transactions, and a 5% of total contribution to every ApeGame, CashApeRise stakers will not just depends on volume for their profits, keeping the platform well-advertised and maximizing the volume of games too will provide the stakers the maximum income they can have, a token that doesn't rely on hype and will just keep active as long as the p2p cash games active.

To play the game the user connect their wallet and make sure they have enough usdt or busd (BSC) on their wallet to deposit and enough CashApeRise tokens as fee for whatever they want to play. The player must select the game he wants to play and the game will start when reached certain players or certain time depends on kind of game.

The private game room is designed for those who want to play each other who are either close friends, business associates, relatives, or family members. The platform's ecosystem casino games depends purely on random cards or numbers generation, giving the players an entirely fair playing experience.

The design of the dapp with the feature of random number generation offers an entertaining, enjoyable, fairness, and potentially lucrative gaming experience.

Project Engagement

During the 28th of June, **CAR 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. **CAR Team** provided Solidproof.io with access to their code repository and whitepaper.

Logo



Contract Link

<https://bscscan.com/address/0xbdeB850cA842C3119d39d9FcA114Ba2c191C14Ef#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 as possible.
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)

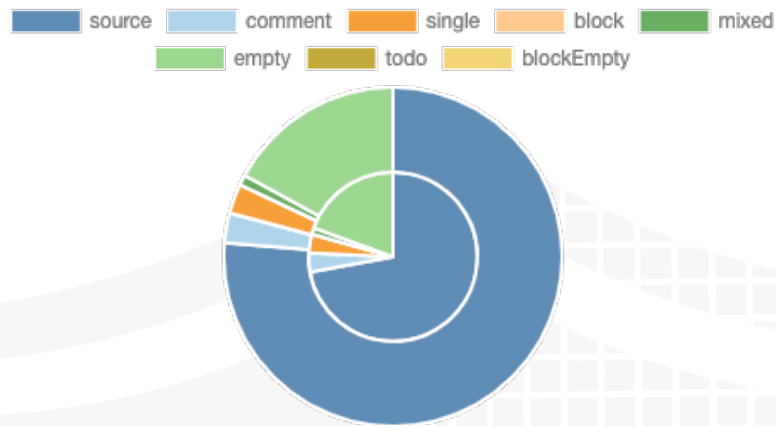
Imported packages:

- OpenZeppelin
 - Address
 - Ownable
 - SafeMatch
- Uniswap
 - UniswapV2Factory
 - UniswapV2Pair
 - UniswapV2Router01
 - UniswapV2Router02

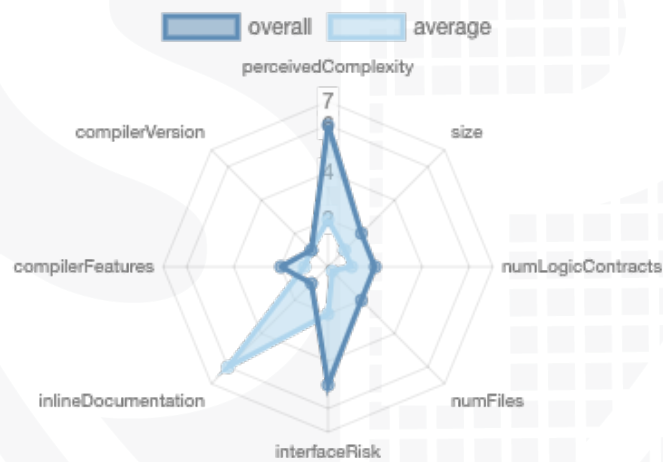


Metrics





Source Lines



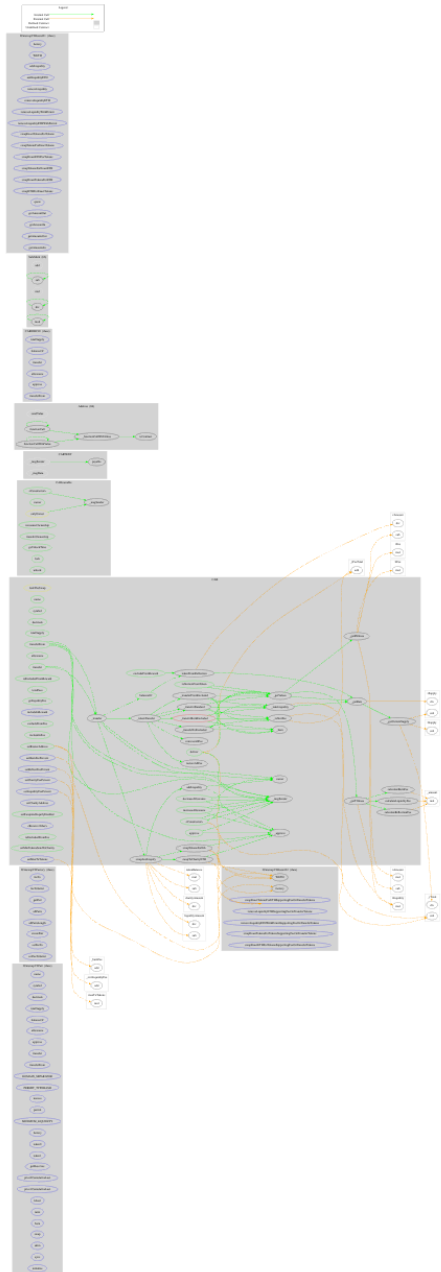
Risk Level



Capabilities

Solidity Versions observed	 Experimental Features	 Can Receive Funds	 Uses Assembly	 Has Destroyable Contracts
>=0.5.0 =0.8.6 ^0.8.4 >=0.6.2		Yes	yes (2 asm blocks)	

CallGraph



Source Units in Scope

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
	contracts/IUniswapV2Pair.sol	_____	1	53	9	5	1	55	_____
	contracts/IUniswapV2Factory.sol	_____	1	19	8	4	1	17	_____
	contracts/CAR.sol	1	_____	570	557	433	22	378	
	contracts/CARownable.sol	1	_____	57	57	44	1	38	_____
	contracts/CARTEXT.sol	2	_____	65	53	44	1	39	
	contracts/IUniswapV2Router02.sol	_____	1	48	10	6	1	16	
	contracts/CARIERC20.sol	1	1	64	54	40	1	23	
	contracts/IUniswapV2Router01.sol	_____	1	97	6	3	1	48	
	Totals	5	5	973	754	579	29	614	

Audit Results

AUDIT PASSED

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

- no informational issues found -

SWC Attacks

ID	Title	Relationships	Status
SW C-13 6	Message call with hardcoded gas amount	CWE-767: Access to Critical Private Variable via Public Method	PASSED
SW C-13 5	Message call with hardcoded gas amount	CWE-1164: Irrelevant Code	PASSED
SW C-13 4	Message call with hardcoded gas amount	CWE-655: Improper Initialization	PASSED
SW C-13 3	Presence of unused variables	CWE-294: Authentication Bypass by Capture-replay	PASSED
SW C-13 2	Presence of unused variables	CWE-667: Improper Locking	PASSED
SW C-13 1	Presence of unused variables	CWE-1164: Irrelevant Code	PASSED
SW C-13 0	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	PASSED
SW C-12 9	Typographical Error	CWE-480: Use of Incorrect Operator	PASSED
SW C-12 8	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	PASSED

SW C-12 7	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	PASSED
SW C-12 5	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	PASSED
SW C-12 4	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	PASSED
SW C-12 3	Requirement Violation	CWE-573: Improper Following of Specification by Caller	PASSED
SW C-12 2	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	PASSED
SW C-12 1	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	PASSED
SW C-12 0	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	PASSED
SW C-11 9	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	PASSED
SW C-11 8	Incorrect Constructor Name	CWE-665: Improper Initialization	PASSED
SW C-11 7	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	PASSED

SW C-11 6	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SW C-11 5	Authorization through tx.origin	CWE-477: Use of Obsolete Function	PASSED
SW C-11 4	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	PASSED
SW C-11 3	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	PASSED
SW C-11 2	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SW C-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	PASSED
SW C-11 0	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	PASSED
SW C-10 9	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	PASSED
SW C-10 8	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED
SW C-10 7	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	PASSED
SW C-10 6	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	PASSED

SW C-10 5	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	PASSED
SW C-10 4	Unchecked Call Return Value	CWE-252: Unchecked Return Value	PASSED
SW C-10 3	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	PASSED
SW C-10 2	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	PASSED
SW C-10 1	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	PASSED
SW C-10 0	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED

The logo features the words "Solid Proofed" in a white, elegant script font. The word "Solid" is positioned above "Proofed". Behind the text is a faint, stylized shield emblem. The shield is composed of a grid of squares, with the left side being a solid blue and the right side being a lighter blue with a grid pattern. The entire logo is set against a solid blue background.

Solid
Proofed

Blockchain Security | Smart Contract Audits



MADE IN GERMANY