SOLIDProof

Blockchain Security | Smart Contract Audits

MADE IN GERMANY

Audit Passed

Security Assessment
9. June, 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.

Overview

Network

Binance Smart Chain (BEP20)

Website

https://www.thecollectivecoin.co/

Telegram

https://t.me/joinchat/w9TCIQ0vZRBiMmM5

Twitter

https://twitter.com/CollectiveCoin_

Discord

https://discord.gg/zY2P77y4KS

Github

https://github.com/TheCollectiveCoin/TheCollectiveCoin

Email

info@thecollectivecoin.co

Reddit

https://www.reddit.com/r/CollectiveCoin/

Description

Currently Collective Coin Team offering graphic design services in exchange for cryptocurrencies, the goal of The Collective is to create a decentralized marketplace, inspired by platforms such as Etsy and Fiverr, for other providers to also offer their products and services to clients in exchange for The Collective Coin, or virtually any established cryptocurrency of their choosing, for a small fee.

Providers will create a profile, and have the ability to establish a virtual storefront on the Marketplace, which will be accessible through a dApp from the user's wallet.

The Collective Coin is a community driven Defi coin and will be the official currency of The Collective. The coin will be the main currency on the marketplace.

Project Engagement

During the 7th of June, **Collective Coin 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. **Collective Coin Team** provided Solidproof.io with access to their code repository and whitepaper.

Logo

Contract Link

TBA

Preversion: https://github.com/TheCollectiveCoin/TheCollectiveCoin/blob/main/TheCollectiveCoin.sol

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.

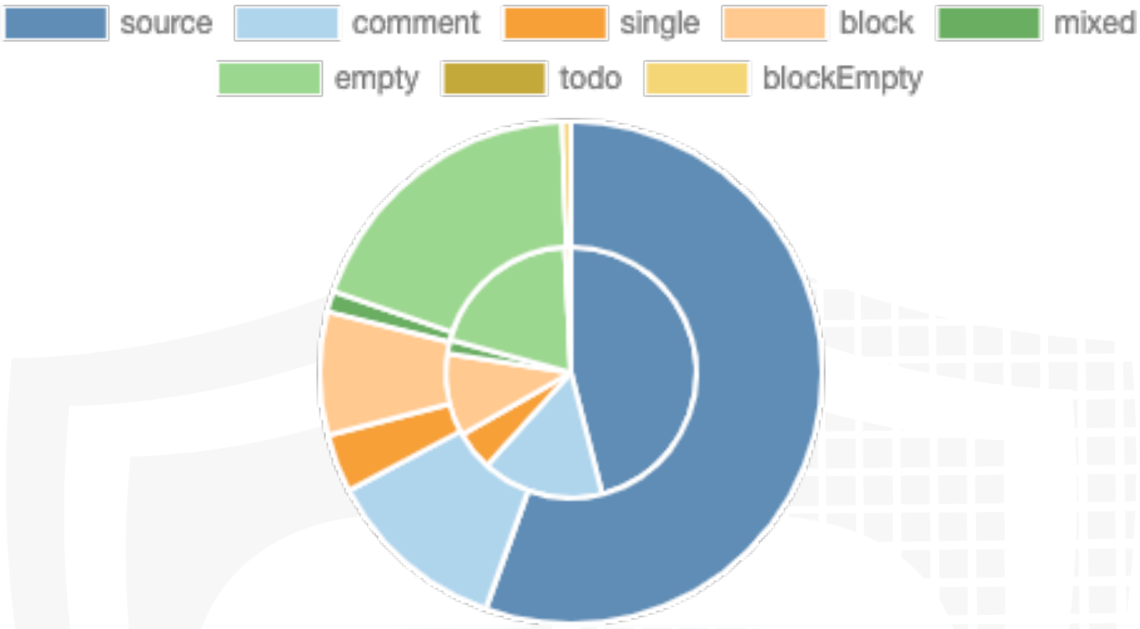
4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

Used Code from other Frameworks/Smart Contracts (direct imports)

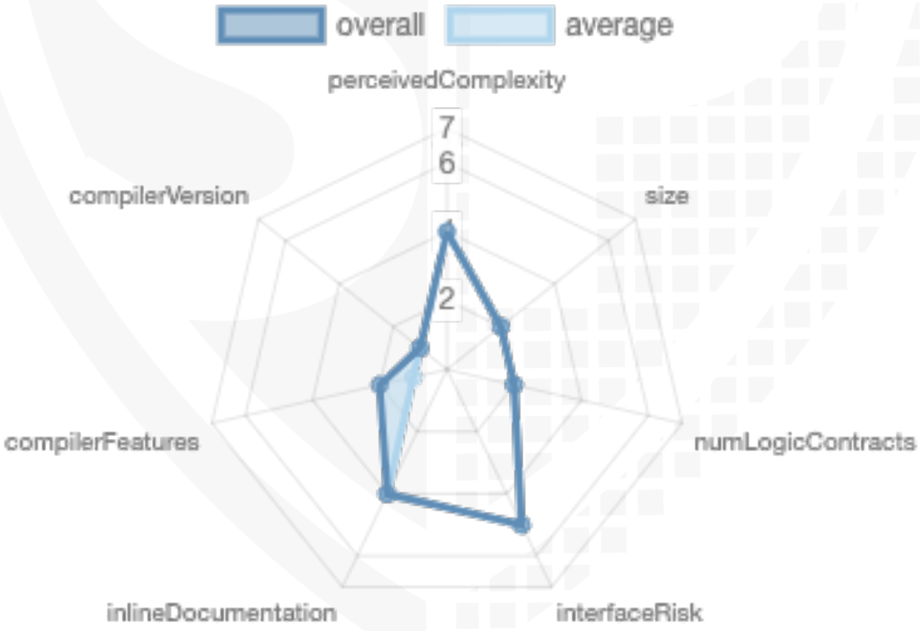
No other external frameworks imported.

Metrics

Source Lines



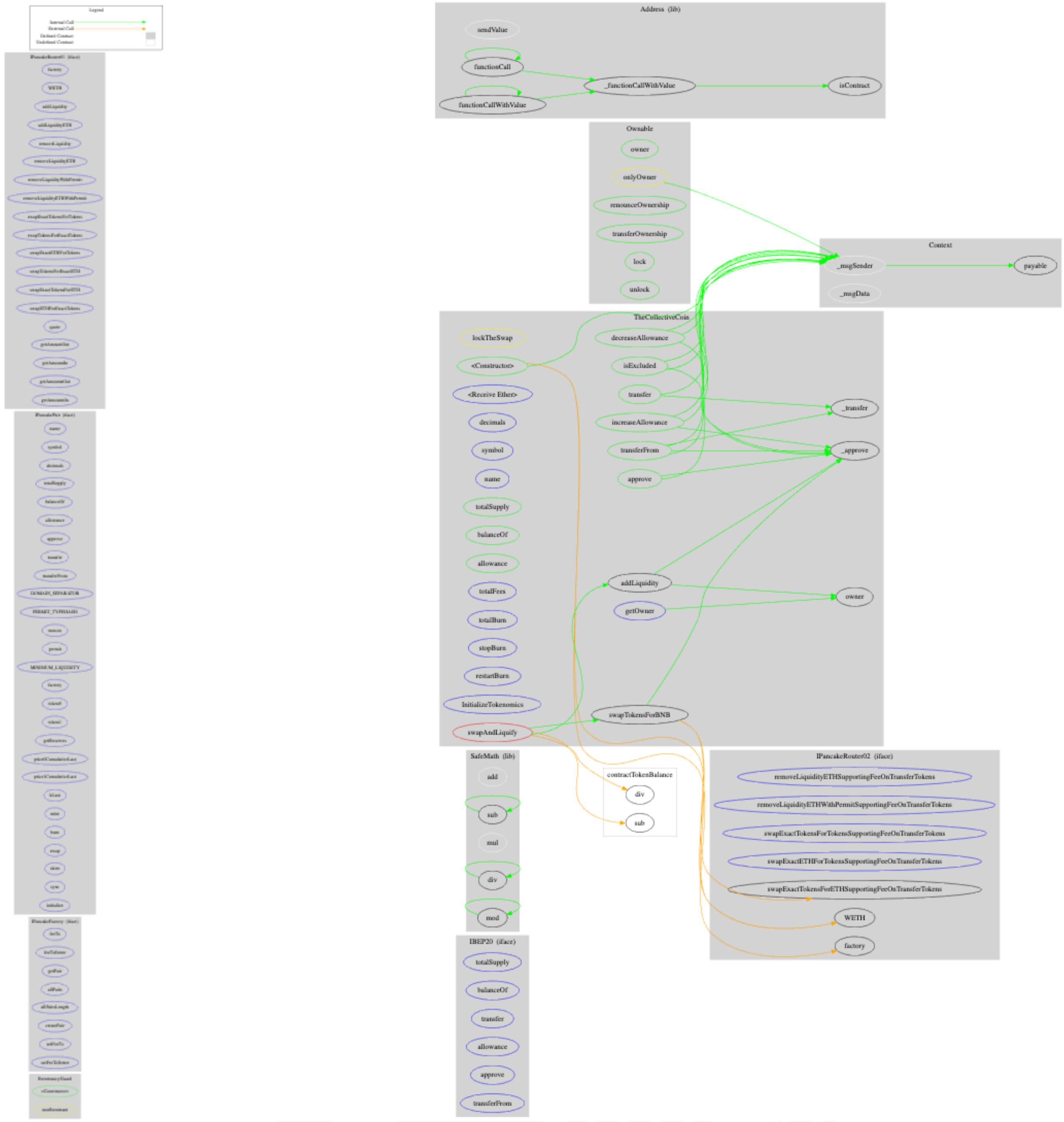
Risk Level



Capabilities

Solidity Versions observed	🧪 Experimental Features	💰 Can Receive Funds	🖥 Uses Assembly	💣 Has Destroyable Contracts
^0.8.2		Yes	yes (2 asm blocks)	

CallGraph



Source Units in Scope

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝📚🔍🎨	contracts/collective_coin.sol	6	5	766	534	310	106	389	🖥💰☀️
📝📚🔍🎨	**Totals**	**6**	**5**	**766**	**534**	**310**	**106**	**389**	🖥💰☀️

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	Line	Type	Finding
#1	Main	198-203	Dangerous usage of `block.timestamp` (timestamp)	Ownable.unlock() uses timestamp for comparisons • require(bool,string)(block.timestamp > _lockTime,Contract is still locked) (Line: 200)
#2	Main	478	Local variables shadowing (shadowing-local)	TheCollectiveCoin._approve(address,address,uint256).owner shadows: • Ownable.owner() (Line: 168-170)
#3	Main	428	Local variables shadowing (shadowing-local)	TheCollectiveCoin.allowance(address,address).owner shadows: • Ownable.owner() (Line: 168-170)
#4	Main	538-550	Unused return values (unused-return)	TheCollectiveCoin.addLiquidity(uint256,uint256) ignores return value by pancakeRouter.addLiquidityETH{value: bnbAmount}(address(this),tokenAmount,0,0,owner(),block.timestamp) (Line: 541-547)

#5	Main	331	Uninitialized state variables (uninitialized-state)	TheCollectiveCoin._tBurnTotal is never initialized. It is used in: • TheCollectiveCoin.totalBurn() (Line: 474-476)
#6	Main	330	Uninitialized state variables (uninitialized-state)	TheCollectiveCoin._tFeeTotal is never initialized. It is used in: • TheCollectiveCoin.totalFees() (Line: 471-473)

Informational issues

Issue	File	Description	Line	Finding
#1	Main	Never used	313	address[] private _excluded
#2	Main	Never used	317	mapping(address => uint256) _tOwned
#3	Main	Should be constant	318	uint256 private _tTotal = 50000000000 * 10**4 * 10**9
#4	Main	Never used	320	mapping(address => bool) private _isExcludedFromFee
#5	Main	Never used	321	mapping(address => bool) private _isExcluded
#6	Main	Never used	323	mapping(address => bool) private _isExcludedFromMaxTx
#7	Main	Never used	326	uint256 public prevTaxAmmount = taxAmmount
#8	Main	Never used	328	uint256 private prevLiqAmmount = liqAmount
#9	Main	Should be constant	335	string private _symbol = "TCC"
#10	Main	Should be constant	336	string private _name = "TheCollective"
#11	Main	Should be constant	337	uint8 private _decimals = 9
#12	Main	Should be constant	341	bool public inSwapAndLiquifyEnabled = false
#13	Main	Never used	248 - 269	function _functionCallWithValue(address target, bytes memory data, uint256 weiValue, string memory errorMessage) private returns (bytes memory)
#14	Main	Never used	244 - 247	function functionCallWithValue(address target, bytes memory data, uint256 value, string memory errorMessage) internal returns (bytes memory)
#15	Main	Is not mixedCase	596	function DOMAIN_SEPARATOR() external view returns (bytes32)
#16	Main	Is not mixedCase	597	function PERMIT_TYPEHASH() external pure returns (bytes32)

SWC Attacks

ID	Title	Relationships	Status
SWC-131	Presence of unused variables	CWE-1164: Irrelevant Code	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

MADE IN GERMANY