SOLIDProof
Bring trust into your projects

**Blockchain Security | Smart Contract Audits | KYC
Development | Marketing**

MADE IN GERMANY

Coin7

Audit

Security Assessment
06. June, 2023

For

coin7
BLOCKCHAIN TECHNOLOGY

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	01. June 2023	• Layout project • Automated- /Manual-Security Testing • Summary
1.1	06. June 2023	• Reaudit

Note - This Audit report comprises a security analysis of the **Coin7** smart contracts. This analysis did not include functional testing (or unit testing) of the contract's logic.

Network
Ethereum

Website
https://coin7.org

Twitter
https://twitter.com/coin7official

YouTube
https://www.youtube.com/@coin7-official

Instagram
https://www.instagram.com/coin7official/

Telegram
https://t.me/coin7official

Facebook
https://www.facebook.com/coin7.org

Description

Coin7 aims to revolutionize the trade of goods and services by building one of the world's first marketplaces based on cryptocurrencies. The project provides a secure, fast, and cost-effective alternative to traditional payment methods. With its MLM and advertising cashback module, users can generate extra income and actively shape the platform. The mining module allows users to mine coins directly from their mobile device, contributing to their availability in the market and earning coins. The advertising module offers app users a default 50% revenue share of all advertising income, and the MLM module allows users to create additional income streams and build a distribution structure.

Project Engagement

During the Date of 30 May 2023, **Coin7 Team** engaged Solidproof.io to audit smart contracts that they created. The engagement was technical 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

coin7
BLOCKCHAIN TECHNOLOGY

Contract Link

v1.0

- Provided as Files

v1.1

Token Contract - https://etherscan.io/address/0x5516a6e0cC74077061c86B2AEe46DcbE96582333#code

- Other contracts are same as the V1.0

Note for Investors: We only Audited a token, presale, and a crowdsale contract for the **Coin7 Team**. However, Suppose the project has other contracts (for example, a Presale contract etc.) that were not provided to us in the audit scope. In that case, we cannot comment on its security and are not responsible for it in any way.

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.

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.

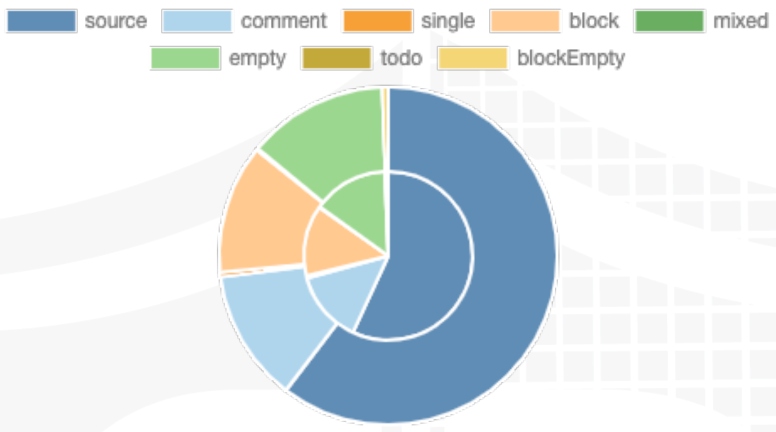
v1.0

File Name	SHA-1 Hash
contracts/ Crowdsale_flattened.sol	ac8d9fc569db815c61f222f0a917d6b 5772c32b5
contracts/ PriceConsumerV3_flattened.so l	bae743aec59a71e062e41dba2d88af 9626987716
contracts/Coin7_flattened.sol	5824544779dcee6ef8270f4c571bca2 29c2cad27
contracts/ PresaleToken_flattened.sol	3fdcbd575dccf8bcfd4f336c79342041 8b0c463c

Metrics

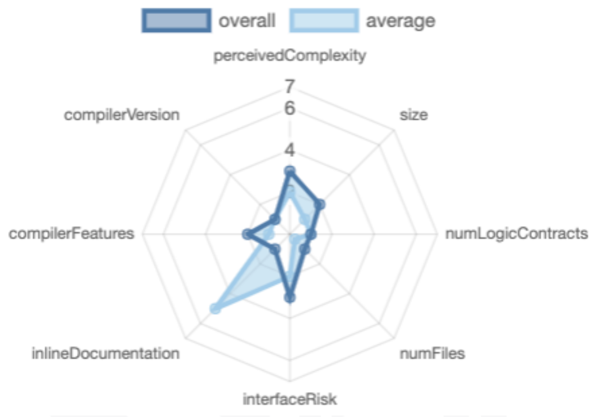
Source Lines

v1.0



Risk Level

v1.0



Capabilities

Components

📝Contracts	📚Libraries	🔍Interfaces	🎨Abstract
8	13	11	10

Exposed Functions

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

🌐Public	💰Payable
114	2

External	Internal	Private	Pure	View
55	235	12	62	81

StateVariables

Total	🌐Public
70	45

Capabilities

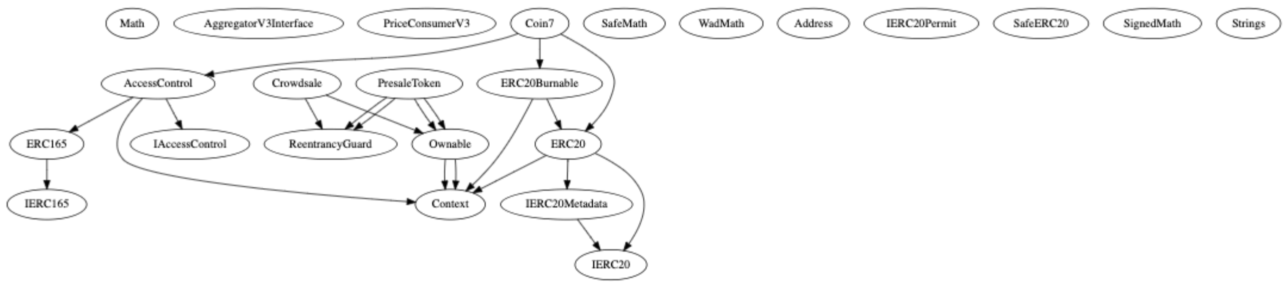
Solidity Versions observed	🧪 Experimental Features	💰 Can Receive Funds	🖥 Uses Assembly	💣 Has Destroyable Contracts
`>=0.4.22 <0.9.0` `^0.8.0` `^0.8.1`		`yes`	`yes` (7 asm blocks)	

📤 Transfers ETH	⚡ Low-Level Calls	👥 DelegateCall	🧮 Uses Hash Functions	🔖 ECRecover	🌀 New/Create/Create2
`yes`		`yes`	`yes`		

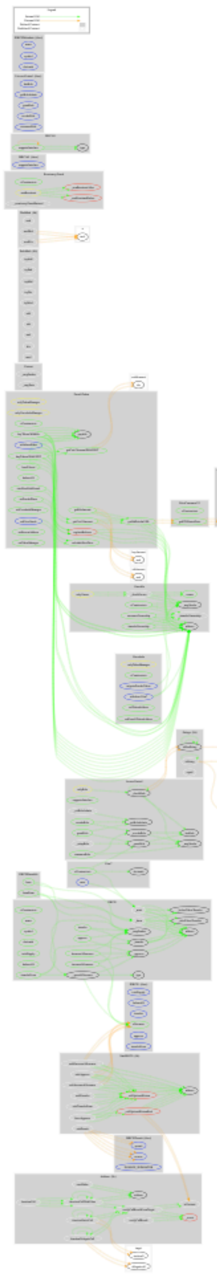
♻️ TryCatch	Σ Unchecked
	`yes`

Inheritance Graph

v1.0



CallGraph
v1.0



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. Is contract an upgradeable
2. Correct implementation of Token standard
3. Deployer cannot mint any new tokens
4. Deployer cannot burn or lock user funds
5. Deployer cannot pause the contract
6. Deployer cannot set fees
7. Deployer cannot blacklist/antisnipe addresses
8. Overall checkup (Smart Contract Security)

Is contract an upgradeable

Name	
Is contract an upgradeable?	No

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	✓	✓	✓

Deployer cannot mint any new tokens

Name	Exist	Tested	Status
Deployer cannot mint	✓	✓	✓
Max / Total Supply	10000000000		

Comments:

v1.1

- The Token contract has a modified Mint function, but it has been made internal and uncallable in version 1.1. Hence, the owner cannot create new tokens now.

Deployer cannot burn or lock user funds

Name	Exist	Tested	Status
Deployer cannot lock	–	–	–
Deployer cannot burn	–	–	–

Deployer cannot pause the contract

Name	Exist	Tested	Status
Deployer can 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)

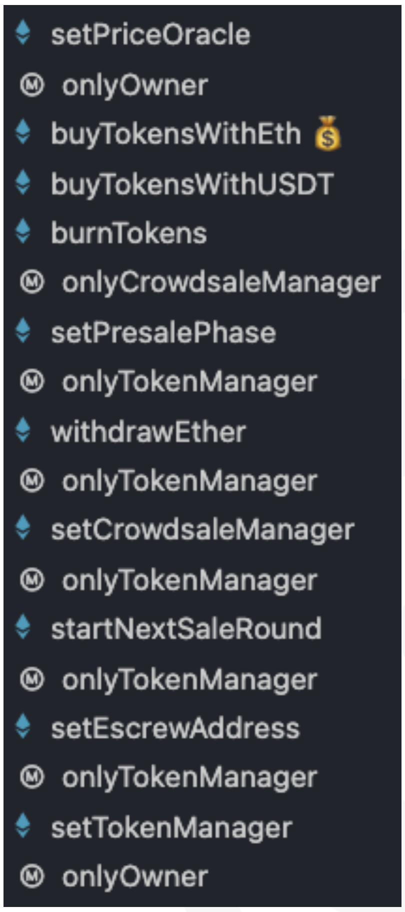
Tested	Verified
✓	✓

Legend

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

Modifiers and public functions
v1.0

Crowdsale_flattened



setPriceOracle
onlyOwner
buyTokensWithEth 💰
buyTokensWithUSDT
burnTokens
onlyCrowdsaleManager
setPresalePhase
onlyTokenManager
withdrawEther
onlyTokenManager
setCrowdsaleManager
onlyTokenManager
startNextSaleRound
onlyTokenManager
setEscrewAddress
onlyTokenManager
setTokenManager
onlyOwner

Ownership Privileges

❖ *Crowdsale_flattened.sol* -

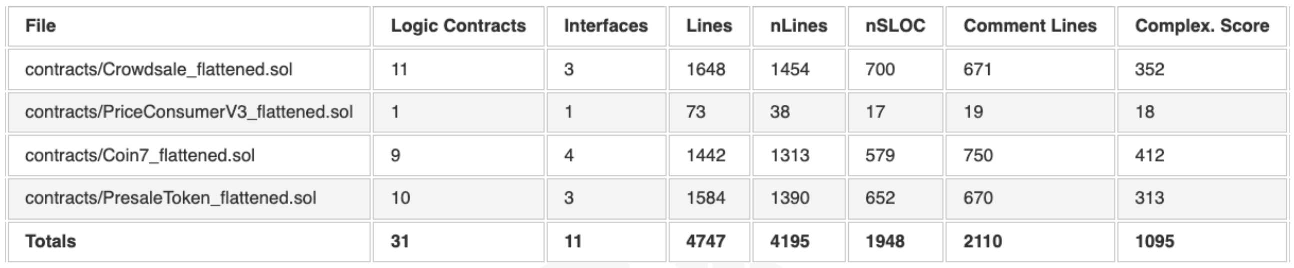
The owner, crowd sale manager, and token manager addresses have the following privileges

- Set Price Oracle address
- Crowd Sale Manager can Burn owner's tokens but only when
- Switch Presale phase
- Withdraw Ethereum from the escrow address
- Set Escrow contract address, where the funds will be stored when a user buys
- The owner can set the token manager address
- The owner can set token and presale addresses

❖ *PresaleToken_flattened.sol* -

The presale token contract has the same privileges as the Crowdsale contract.

Source Units in Scope
v1.0



File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score
contracts/Crowdsale_flattened.sol	11	3	1648	1454	700	671	352
contracts/PriceConsumerV3_flattened.sol	1	1	73	38	17	19	18
contracts/Coin7_flattened.sol	9	4	1442	1313	579	750	412
contracts/PresaleToken_flattened.sol	10	3	1584	1390	652	670	313
Totals	**31**	**11**	**4747**	**4195**	**1948**	**2110**	**1095**

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

Medium Issues Fixed

Issue	File	Type	Line	Description	Status
#1	Coin7_flattened.sol	Owner can mint tokens	1439	The owner can grant MINTER ROLE to any address and that address can mint unlimited tokens	FIXED

Low issues

Issue	File	Type	Line	Description	Status
#1	Crowdsale_flattened.sol	Missing Zero Address Validation (missing-zero-check)	1284, 1558, 1567	Check that the address is not zero	Open
#2	Crowdsale_flattened.sol	Missing Events Arithmetic	1558, 1562	Emit an event for critical parameter changes	Open
#3	PresaleToken_flattened.sol	Missing Zero Address Validation (missing-zero-check)	1558, 1562	Check that the address is not zero	Open

Informational issues

Issue	File	Type	Line	Description	Status
#1	All	NatSpec documentation missing	—	If you started to comment your code, also comment all other functions, variables etc.	Open

Audit Comments

We recommend you use the particular form of comments (NatSpec Format, Follow the link for more information https://docs.soliditylang.org/en/latest/natspec-format.html) for your contracts to provide rich documentation for functions, return variables and more. This helps investors to make clear what that variable, functions etc., do.

06. June 2023:

- There is still an owner because the contracts are yet to be deployed (The owner still has not renounced ownership)
- Unit tests with at least 95% code coverage and a Whitepaper were not provided to SolidProof, so we cannot ensure the complete functional correctness of the code's logic.
- We recommend that the **Coin7** team conduct unit and fuzz tests thoroughly to rule out the possibilities of unwanted logical and calculation errors.
- Read the whole report and modifiers section for more information

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	**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	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

SolidProof_io

@solidproof_io

Solid Proofed

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY