SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC
Development | Marketing

MADE IN GERMANY

Cruna Protocol

AUDIT

SECURITY ASSESSMENT

11. April, 2024

FOR

CRUNA PROTOCOL

SolidProof_io

@solidproof_io

Introduction 3
Disclaimer 3
Project Overview 4
Summary 4
Social Medias 4
Audit Summary 5
File Overview 6
Imported packages 8
Audit Information 9
Vulnerability & Risk Level 9
Auditing Strategy and Techniques Applied 10
Methodology 10
Overall Security 11
Upgradeability 11
Ownership 12
Ownership Privileges 13
Burning tokens 13
Lock User Funds 14
Components 15
Exposed Functions 15
StateVariables 15
Capabilities 16
Audit Results 17
Critical issues 17
High issues 17
Medium issues 17
Low issues 18
Informational issues 19

Introduction

SolidProof.io is a brand of the officially registered company MAKE Network GmbH, based in Germany. We're mainly focused on Blockchain Security such as Smart Contract Audits and KYC verification for project teams.
Solidproof.io assess potential security issues in the smart contracts implementations, review for potential inconsistencies between the code base and the whitepaper/documentation, and provide suggestions for improvement.

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 the security or functionality of the technology we agree to analyze.

Project Overview

Summary

Project Name	Cruna Protocol
Website	
About the project	N/A
Chain	EVM Compatible Chains
Language	Solidity
Codebase Link	https://github.com/crunaprotocol/cruna-protocol
Commit	9530211
Unit Tests	Provided

Social Medias

Telegram	N/A
Twitter	N/A
Facebook	N/A
Instagram	N/A
Github	N/A
Reddit	N/A
Medium	N/A
Discord	N/A
Youtube	N/A
TikTok	N/A
LinkedIn	N/A

Audit Summary

Version	Delivery Date	Changelog
v1.0	11. April 2024	• Layout Project • Automated- /Manual-Security Testing • Summary

Note - The following audit report presents a comprehensive security analysis of the smart contract utilized in the project, including malicious outside manipulation of the contract's functions. This analysis did not include functional testing (or unit testing) of the contract/s logic. We cannot guarantee 100% logical correctness of the contract as we did not functionally test it. This includes internal calculations in the formulae used in the contract.

File Overview

The Team provided us with the files that should be tested in the security assessment. This audit covered the following files listed below with an SHA-1 Hash.

File Name
contracts/erc/IERC6454.sol
contracts/erc/IERC6982.sol
contracts/erc/IERC7656Contract.sol
contracts/erc/IERC7656Registry.sol
contracts/guardian/CrunaGuardian.sol
contracts/guardian/ICrunaGuardian.sol
contracts/libs/Canonical.sol
contracts/libs/ExcessivelySafeCall.sol
contracts/libs/ManagerConstants.sol
contracts/manager/Actor.sol
contracts/manager/CrunaManager.sol
contracts/manager/CrunaManagerBase.sol
contracts/manager/CrunaManagerProxy.sol
contracts/manager/ICrunaManager.sol
contracts/plugins/CrunaPluginBase.sol
contracts/plugins/ICrunaPlugin.sol
contracts/plugins/inheritance/IInheritanceCrunaPlugin.sol
contracts/plugins/inheritance/InheritanceCrunaPlugin.sol
contracts/plugins/inheritance/InheritanceCrunaPluginProxy.sol
contracts/token/CrunaProtectedNFT.sol
contracts/token/CrunaProtectedNFTOwnable.sol
contracts/token/CrunaProtectedNFTTimeControlled.sol

contracts/token/ICrunaProtectedNFT.sol
contracts/token/IManagedNFT.sol
contracts/utils/CommonBase.sol
contracts/utils/ERC6551AccountProxy.sol
contracts/utils/ICommonBase.sol
contracts/utils/INamed.sol
contracts/utils/INamedAndVersioned.sol
contracts/utils/ISignatureValidator.sol
contracts/utils/IVersioned.sol
contracts/erc/ERC7656Registry.sol
contracts/utils/SignatureValidator.sol
contracts/utils/TokenLinkedContract.sol

Please note: Files with a different hash value than in this table have been modified after the security check, either intentionally or unintentionally. A different hash value may (but need not) indicate a changed state or potential vulnerability that was not the subject of this scan.

Imported packages

Used code from other Frameworks/Smart Contracts (direct imports).

Dependency / Import Path	Count
@openzeppelin/contracts/access/Ownable2Step.sol	1
@openzeppelin/contracts/governance/TimelockController.sol	2
@openzeppelin/contracts/proxy/ERC1967/ERC1967Utils.sol	1
@openzeppelin/contracts/proxy/Proxy.sol	1
@openzeppelin/contracts/token/ERC721/ERC721.sol	2
@openzeppelin/contracts/token/ERC721/IERC721.sol	1
@openzeppelin/contracts/utils/Address.sol	1
@openzeppelin/contracts/utils/Context.sol	2
@openzeppelin/contracts/utils/ReentrancyGuard.sol	3
@openzeppelin/contracts/utils/StorageSlot.sol	2
@openzeppelin/contracts/utils/Strings.sol	3
@openzeppelin/contracts/utils/cryptography/ECDSA.sol	4
@openzeppelin/contracts/utils/cryptography/EIP712.sol	1
erc6551/interfaces/IERC6551Registry.sol	1
erc6551/lib/ERC6551AccountLib.sol	2

Note for Investors: We only audited contracts mentioned in the scope above. All contracts related to the project apart from that are not a part of the audit, and we cannot comment on its security and are not responsible for it in any way

Audit Information

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. The 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 check the repository for security-related issues, code quality, and compliance with specifications and best practices. To this end, our team of experienced pen-testers and smart contract developers reviewed the code line by line and documented any issues discovered.

We check every file manually. We use automated tools only so that they help us achieve faster and better results.

Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
    a. Review the specifications, sources, and instructions provided to SolidProof to ensure we understand the smart contract's size, scope, and functionality.
    b. Manual review of the code, i.e., reading the source code line by line to identify potential vulnerabilities.
    c. Comparison to the specification, i.e., verifying that the code does what is described in the specifications, sources, and instructions provided to SolidProof.

2. Testing and automated analysis that includes the following:
    a. Test coverage analysis determines whether test cases cover code and how much code is executed when those test cases are executed.
    b. Symbolic execution is analysing a program to determine what inputs cause each part of a program to execute.

3. Review best practices, i.e., smart contracts to improve efficiency, effectiveness, clarity, maintainability, security, and control based on best practices, recommendations, and research from industry and academia.

4. Concrete, itemized and actionable recommendations to help you secure your smart contracts.

Overall Security

Upgradeability

Contract is not an upgradeable	✅ Deployer cannot update the contract with new functionalities
Description	The contract is not an upgradeable contract. The deployer is not able to change or add any functionalities to the contract after deploying.
Comment	N/A

Ownership

The ownership is renounced	✅ **The owner is renounced**
Description	The owner renounced the ownership that means the contract's owner will no longer have any control or authority over the contract's operations.
Comment	The smart contracts will be used as a library so the ownership will technically belong to the users who will implement this code in their project.

Ownership Privileges

These functions can be dangerous. Please note that abuse can lead to financial loss. We have a guide where you can learn more about these Functions.

Burning tokens

Burning tokens is the process of permanently destroying a certain number of tokens, reducing the total supply of a cryptocurrency or token. This is usually done to increase the value of the remaining tokens, as the reduced supply can create scarcity and potentially drive up demand.

Contract owner cannot burn tokens without allowance	✅ **The owner cannot burn tokens**
Description	The owner is not able burn tokens without any allowances.
Comment	N/A

Lock User Funds

In a smart contract, locking refers to the process of restricting access to certain tokens or assets for a specified period of time. When tokens or assets are locked in a smart contract, they cannot be transferred or used until the lock-up period has expired or certain conditions have been met.

Owner cannot lock the contract	✅ **The owner cannot lock the contract**
Description	The owner is not able to lock the contract by any functions or updating any variables.
Comment	N/A

External/Public functions

External/public functions are functions that can be called from outside of a contract, i.e., they can be accessed by other contracts or external accounts on the blockchain. These functions are specified using the function declaration's external or public visibility modifier.

State variables

State variables are variables that are stored on the blockchain as part of the contract's state. They are declared at the contract level and can be accessed and modified by any function within the contract. State variables can be defined with a visibility modifier, such as public, private, or internal, which determines the access level of the variable.

Components

📝Contracts	📚Libraries	🔍Interfaces	🎨Abstract
8	3	16	8

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
178	1

External	Internal	Private	Pure	View
169	178	0	37	132

StateVariables

Total	🌐Public
30	1

Capabilities

Solidity Versions observed	Transfers ETH	💰 Can Receive Funds	🖥 Uses Assembly	💣 Has Destroyable Contracts
`^0.8.20`		Yes	Yes	

Audit Results

Critical issues

No critical issues

High issues

No high issues

Medium issues

#1 | Unsafe Use of 'delete'

File	Severity	Location	Status
Actor.sol	Medium	L137	Fixed
CrunaProtectedNFT.sol	Medium	L151	Fixed
CrunaGuardian.sol	Medium	L74	Fixed
CrunaManager.sol	Medium	L445, 446, 506, 514, 781	Fixed

Description - Using 'delete', assigns the initial default value for the type to the variable in question. For integer variables, this means setting the value to 0. This behaviour can be particularly useful when resetting an integer variable to its default state. Otherwise, if the intention is not to reset the values, unexpected behaviour can occur when the code is live.

Remediation - We recommend avoiding the use of "delete" if the intention is not to reset the values.

Alleviation - The use case of 'delete' in the entire codebase is intended for resetting values.

Low issues

#1 | Variable Shadowing

File	Severity	Location	Status
CrunaManager.sol	Low	L377	Fixed
TokenLinkedContract.sol	Low	L45	Fixed

Description - Ensure the variable name does not shadow another component in the same or inherited codebase.

Informational issues

#1 | NatSpec documentation missing

File	Severity	Location	Status
All	Informational	N/A	ACK

Description — If you have started commenting on your code, comment on all other functions, variables, etc. Most of the code is well-documented, but certain functions refer the reader to the interface of a certain contract. Since these contracts are meant to be used as a library, we recommend improving the natspec.

Legend for the Issue Status

Attribute or Symbol	Meaning
Open	The issue is not fixed by the project team.
Fixed	The issue is fixed by the project team.
Acknowledged(ACK)	The issue has been acknowledged or declared as part of business logic.

Solid Proofed

Blockchain Security | Smart Contract Audits | KYC
Development | Marketing

MADE IN GERMANY