SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

Audit

Security Assessment

04. October, 2021

For

CRYPTOPUNT

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	04. October 2021	• Layout project • Automated- /Manual-Security Testing • Summary

Network
Polygon Chain (Matic)

Website
https://www.cryptopunt.com/

Telegram
https://t.me/CryptoPunt

Twitter
https://twitter.com/PuntCrypto

Description
Crypto-powered gambling and games with openly auditable algorithms. Bet and level up your holdings.

Project Engagement
During the 29th of September 2021, **CryptoPunt 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
coinflip logic: https://polygonscan.com/address/0xe91A0a62DbDB07cf6c6F580d6Cb52FDC8f639392

coinflip proxy: https://polygonscan.com/address/0x760B55F7AA89aa519a48a9acE82FA3B9e3083715

vault logic: https://polygonscan.com/address/0xE00A070bEA5235f9DF1Db78a3CdC019b4ea7D6aB

vault proxy: https://polygonscan.com/address/0x2aCcc4FA6e5361682b2141258A151B99699B1D57

jackpot logic: https://polygonscan.com/address/0x5f5aDaee7F36699671CDDD1b3326F69b3262f821

jackpot proxy: https://polygonscan.com/address/0x90b292e0393f31d031470dA5005D727D40fAe78c

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)

Imported packages:

Dependency / Import Path	Count
@chainlink/contracts/src/v0.8/VRFRequestIDBase.sol	1
@chainlink/contracts/src/v0.8/interfaces/LinkTokenInterface.sol	1
@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol	3
@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol	3
@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol	5
@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol	2
@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol	2
@openzeppelin/contracts/token/ERC20/IERC20.sol	1
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol	1

Tested Contract Files

This audit covered the following files listed below with a SHA-1 Hash.

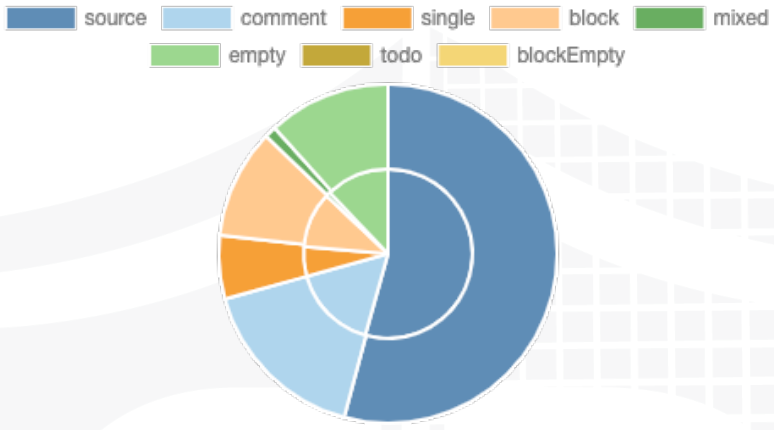
A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

v1.0

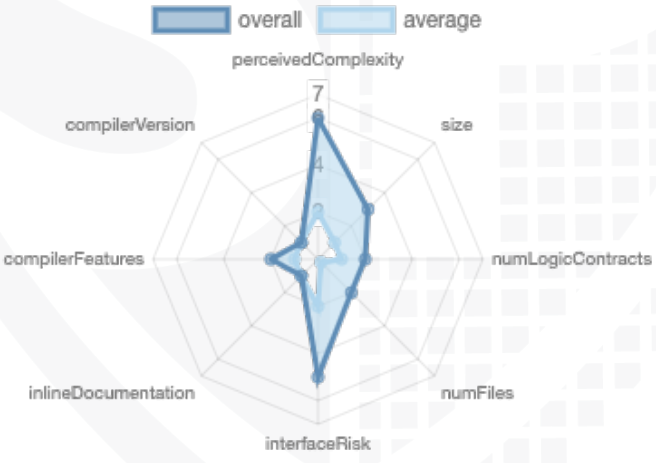
File Name	SHA-1 Hash
contracts/JackPot.sol	0362b3c3ac07f146575196a7609e8048d554979f
contracts/CoinFlip.sol	c1f99a8fd66cd0ffef64918f6d6b077193020771
contracts/Proxy.sol	e5a9350a8bc0169deaa984178c210abe483b284d
contracts/Vault.sol	d4b072162a3ae903fa46a6d5e77e69166c91a72c
contracts/TransparentUpgradeableProxy.sol	e8c2cb3e95bacd2946f82af1edc6159df276e8d8
contracts/UpgradeableProxy.sol	afa2bd90a67f8789db4c1b374516b70d69d7a22d
contracts/utils/IGame.sol	8fbd9a7850ae1e75ac1bcbdd36eb6f8509247cfe
contracts/utils/VRFConsumerBaseUpgradeable.sol	360619a84c8ed976ecd4e826d1a2107fe0fd63bf
contracts/utils/AccessController.sol	bc64148e6b91cbcafa15bf6fa1cee2796c5ff6cd
contracts/utils/Address.sol	2c2adbf3a0b52de7150e317844c55ee624d039fb
contracts/utils/IVault.sol	f45cbb04e1bfbafc092fc2c49b9576e03f58017c

Metrics

Source Lines
v1.0



Risk Level
v1.0



Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	8	1	2	2

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	72	8

Version	External	Internal	Private	Pure	View
1.0	59	107	3	0	12

State Variables

Version	Total	Public
1.0	48	37

Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	`^0.8.4` `^0.7.0`		`yes`	`yes` (7 asm blocks)	

Version	Transfers ETH	Low-Level Calls	DelegateCall	Uses Hash Functions	ECRecover	New/Create/Create2

1.0	`yes`		`yes`	`yes`		

Scope of Work

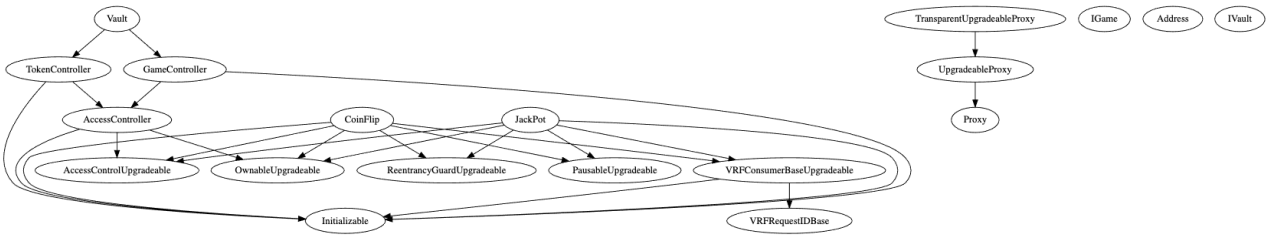
The above token Team provided us with the files that needs to be tested (Github, Bscscan, Etherscan, files, etc.). The scope of the audit is the main contract (usual the same name as team appended with .sol).

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



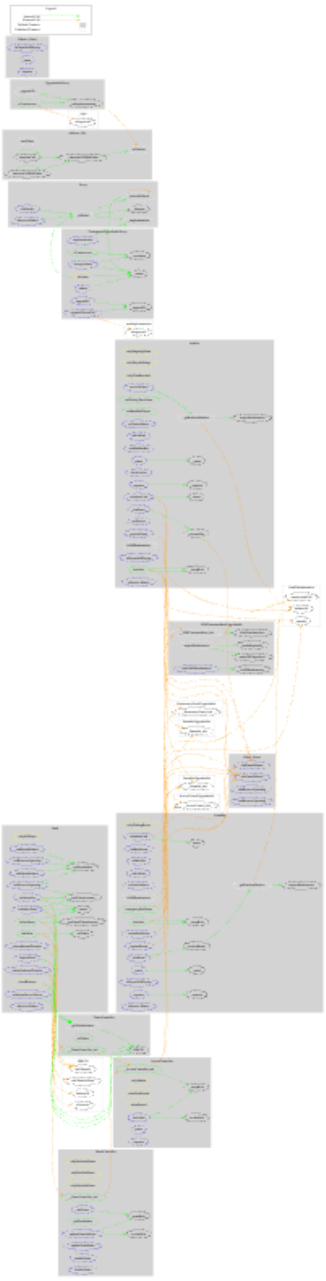
Overall checkup (Smart Contract Security)

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/JackPot.sol	1	———	529	445	331	78	180	💰📩🧱
📝	contracts/CoinFlip.sol	1	———	422	393	295	43	141	💰📩🧱
🎨	contracts/Proxy.sol	1	———	83	76	25	47	48	💻💰👥🔆
📝	contracts/Vault.sol	3	———	387	335	268	1	180	💰📩
📝	contracts/TransparentUpgradeableProxy.sol	1	———	153	153	52	86	62	💻💰👥🧱🔆
📝	contracts/UpgradeableProxy.sol	1	———	80	80	32	38	33	💻💰👥🧱🔆
🔍	contracts/utils/IGame.sol	———	1	11	6	3	1	7	———
🎨	contracts/utils/VRFConsumerBaseUpgradeable.sol	1	———	60	46	39	2	20	———
📝	contracts/utils/AccessController.sol	1	———	70	70	50	1	41	🧱
📚	contracts/utils/Address.sol	1	———	141	126	55	87	37	💻🔆
🔍	contracts/utils/IVault.sol	———	1	39	7	2	17	9	———
📝📚🔍🎨	**Totals**	**11**	**2**	**1975**	**1737**	**1152**	**401**	**758**	💻💰📩👥🧱🔆

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	3 (Jackpot.sol), 3 (CoinFlip.sol), 3 (Vault.sol)	The current pragma Solidity directive is „“^0.8.4““.
#2	TransparentUpgradeableProxy.sol	Missing Zero Address Validation (missing-zero-check)	116	Check that the address is not zero
#3	Upgradeable.sol	Missing Zero Address Validation (missing-zero-check)	24	Check that the address is not zero
#4	Vault.sol	Missing Zero Address Validation (missing-zero-check)	153, 52	Check that the address is not zero
#5	Jackpot.sol	State variable visibility is not set	49	It is best practice to set the visibility of state variables explicitly
#6	Jackpot.sol	Missing Events Arithmetic (events-maths)	123-185	Emit an event for critical parameter changes

Informational issues

Issue	File	Type	Line	Description
#7	Jackpot.sol	Tautology or contradiction (tautology)	237-240, 206-209	Fix the incorrect comparison by changing the value type or the comparison require(_feeRate < 50 && _feeRate >= 0, "Jackpot.sol: Fee Rate can not be more than 50."); require(_max <= 255, "JackPot.sol: Max players can't be more than 255 due to the Maximum of playerIndex");
#8	OwnableUpgradeable.sol	Unused state variables (unused-state)	96	Remove unused state variables
#9	PausableUpgradeable.sol	Unused state variables (unused-state)	96	Remove unused state variables
#10	CoinFlip.sol	Missing inheritance (missing-inheritance)	14-423	CoinFlip (CoinFlip.sol:14-423) should inherit from IGame (utils/IGame.sol#5-12) Recommendation: Inherit from the missing interface or contract.

#11	Jackpot.sol	Missing inheritance (missing-inheritance)	15-530	JackPot (JackPot.sol:13-530) should inherit from IGame (utils/IGame.sol#5-12) Recommendation: Inherit from the missing interface or contract.
#12	Jackpot.sol	Missing inheritance (missing-inheritance)	162-388	Vault (Vault.sol:162-388) should inherit from IVault (utils/IVault.sol#5-40) Recommendation: Inherit from the missing interface or contract.
#13	Jackpot.sol	Functions that are not used (dead-code)	425-475	Remove unused functions
#14	CoinFlip.sol	Functions that are not used (dead-code)	327-364	Remove unused functions

Commented Code exist

There are some instances of code being commented out in the following files that should be removed:

File	Line	Comment

Jackpot.sol	214-218, 231-233	//don't use the direct token transfer anymore. Use vault's function instead.
		// setup token address
		// function setPlayingToken(address _tokenAddress) external onlyOwner {
		// token = IERC20Metadata(_tokenAddress);
		// }
		
		// function setGameState(GameState _state) external onlyOwner {
		// state = _state;
		// }

IVault.sol	21-38	// function getGameIndex(address _addr) external view returns (uint8);
		// send token to player
		// function tokenOutboundTransfer(
		// address _to,
		// uint128 _amount,
		// uint128 _fee
		//) external;
		// token interfaces
		// function checkApproval(address _userAddress)
		// external
		// view
		// returns (uint256);
		// function balanceOf(address _userAddress) external view returns (uint256);
		// player ledger

Recommendation

Remove the commented code, or address them properly.

Audit Comments

04. October 2021:

- Developer can
  - set fees whenPaused
  - Set maxMinPlayers whenPaused
  - Set Vault Address whenPaused
  - setMinMaxBet whenPaused
  - setMaxRoom whenPaused
  - setMaxBet whenPaused
  - Call emergencyEndGame function

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	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

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY