



SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

IslamiCoin

Audit

**Security Assessment
08. July, 2022**

For



[SolidProof_io](#)



[@solidproof_io](#)

Disclaimer	3
Description	5
Project Engagement	5
Logo	5
Contract Link	5
Methodology	7
Used Code from other Frameworks/Smart Contracts (direct imports)	8
Tested Contract Files	9
Source Lines	10
Risk Level	10
Capabilities	11
Inheritance Graph	12
CallGraph	13
Scope of Work/Verify Claims	14
Modifiers and public functions	16
Source Units in Scope	18
Critical issues	19
High issues	19
Medium issues	19
Low issues	19
Informational issues	20
Audit Comments	20
SWC Attacks	21

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	30. June 2022	<ul style="list-style-type: none">• Layout project• Automated- /Manual-Security Testing• Summary
1.1	02. July 2022	<ul style="list-style-type: none">• Reaudit
1.2	08. July 2022	<ul style="list-style-type: none">• Reaudit

Network

Polygon

Website

<https://islamicoin.finance/>

Telegram

<https://t.me/IslamiCoinChat>

Twitter

<https://twitter.com/islamicoin>

Facebook

<https://www.facebook.com/islamicoin>

Instagram

<https://intagram.com/islamicoin>

Whatsapp

<https://wa.me/message/PJMEHT55KKN2O1>

Youtube

https://www.youtube.com/channel/UCPdg9Cx2g9DyTR_xD5S_IxA

LinkedIn

<https://linkedin.com/company/islamicoin>

Description

ISLAMICOIN: a Sharia Compliant Certified Islamic Cryptocurrency for the Global Muslim Community and it is a core to develop ISLAMIBLOCKCHAIN.

Project Engagement

During the 28th of June 2022, **IslamiCoin 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

- Github
 - <https://github.com/ISLAMIBLOCKCHAIN/ISLAMICOIN/tree/main/contracts>
 - Commit: d6b737fb968c4050a40c5b4e4af73d446944ae87

v1.1

- Github
 - <https://github.com/ISLAMIBLOCKCHAIN/ISLAMICOIN/tree/main/contracts>
 - Commit: 3cd385b3403163682a329c3a19e3da8a5e867fd

v1.2

- Github
 - <https://github.com/ISLAMIBLOCKCHAIN/ISLAMICOIN/tree/main/contracts>
 - Commit: c10691d6333000fa40cb423173ea93d6bec428da

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 they 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:

Islamicoin



Tested Contract Files

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

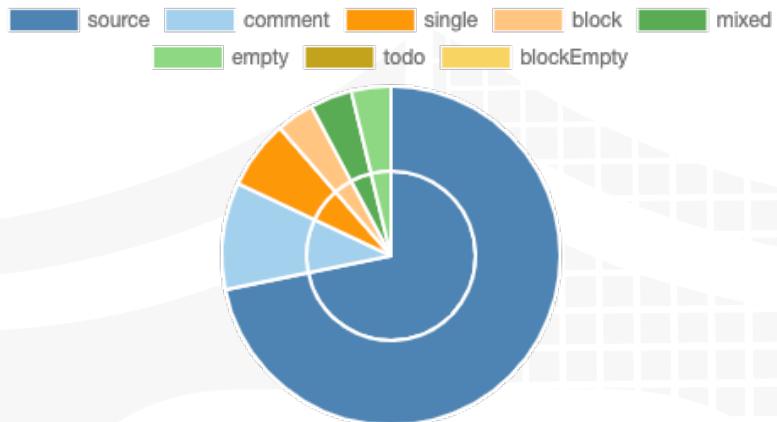
A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

v1.0

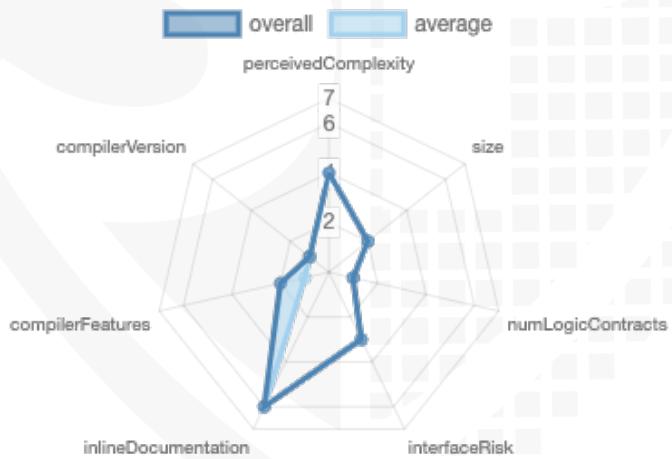
File Name	SHA-1 Hash
contracts/ISLAMInvesting_V4.sol	4d7b1807ab8eadf18e7a8daff45c94302d5ee7e6

Metrics

Source Lines v1.0



Risk Level v1.0



Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	1	0	0	0

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	25	1

Version	External	Internal	Private	Pure	View
1.0	22	29	0	0	2

State Variables

Version	Total	Public
1.0	30	19

Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	=0.8.1 3		yes		

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

1.0	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. Overall checkup (Smart Contract Security)

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

transferOwnership	
onlyOwner	
changeBaytAlMal	
onlyOwner	
setMonthlyPercentage	
onlyOwner	
setMinLock	
onlyOwner	
setEmergencyFee	
onlyOwner	
setOneVote	
onlyOwner	
addToVote	
onlyOwner	
deleteVoteProject	
onlyOwner	
addInvestor	
onlyOwner	
selfLock	
isBlackListed	
nonReentrant	
editSelfLock	
ISslInvestor	
nonReentrant	
extendSelfLock	
ISslInvestor	
nonReentrant	
recoverWallet	
ISslInvestor	
nonReentrant	
selfUnlock	
ISslInvestor	
nonReentrant	
emergencyWithdrawal	
ISslInvestor	
nonReentrant	
claimMonthlyAmount	
isInvestor	
nonReentrant	
claimRemainings	
isInvestor	
nonReentrant	
voteFor	
isBlackListed	
nonReentrant	
releaseWallet	
isInvestor	
nonReentrant	
withdrawalISLAMI	
onlyOwner	
withdrawalERC20	
onlyOwner	
withdrawalMatic	
onlyOwner	

Comments

- Deployer can set following state variables without any limitations
 - mp
 - minLock

- `ewFee`
- `OneVote`
- Deployer can enable/disable following state variables
 - `voteSystem`
 - Add a project to vote system
 - `investor`
- Deployer can set following addresses
 - `owner`
 - `BaytAlMal`
 - Fee addresse
- Existing Modifiers
 - `onlyOwner`
 - `isInvestor`
 - `ISsllInvestor`
 - `isBlackListed`
- There are several authorities which are authorized to call some functions, that means, if the owner is renounced, another address is still authorized to call functions
 - Be aware of this
- Owner can
 - withdraw matics from contract balance
 - Withdraw tokens from external contracts

Please check if an OnlyOwner or similar restrictive modifier has been forgotten.

Source Units in Scope

v1.0

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
	contracts/ISLAMInvesting_V4.sol	1	—	409	409	359	51	234	
	Totals	1	—	409	409	359	51	234	

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

High issues

No high issues

Medium issues

No medium issues

Low issues

Issue	File	Type	Line	Description
#1	Main	Contract doesn't import npm packages from source (like OpenZeppelin etc.)	-	We recommend to import all packages from npm directly without flatten the contract. Functions could be modified or can be susceptible to vulnerabilities
#2	Main	Missing Zero Address Validation (missing-zero-check)	552	Check that the address is not zero
#3	Main	Missing Events Arithmetic	199	Emit an event for critical parameter changes.
#4	Main	Local variables shadowing	335, 238	Rename the local variables that shadow another component. Recommendation: ownerAddress

Informational issues

Issue	File	Type	Line	Description
#1	Main	State variables that could be declared constant (constable-states)	38	Add the `constant` attributes to state variables that never change
#2	Main	Error message is missing	306, 325	Provide an error message for require statement

Audit Comments

We recommend you to use the special form of comments (NatSpec Format, Follow link for more information <https://docs.soliditylang.org/en/v0.5.10/natspec-format.html>) for your contracts to provide rich documentation for functions, return variables and more. This helps investors to make clear what that variables, functions etc. do.

08. July 2022:

- Read whole report and modifiers section for more information

SWC Attacks

ID	Title	Relationships	Status
SW C-1 36	Unencrypted Private Data On-Chain	CWE-767: Access to Critical Private Variable via Public Method	PASSED
SW C-1 35	Code With No Effects	CWE-1164: Irrelevant Code	PASSED
SW C-1 34	Message call with hardcoded gas amount	CWE-655: Improper Initialization	PASSED
SW C-1 33	Hash Collisions With Multiple Variable Length Arguments	CWE-294: Authentication Bypass by Capture-replay	PASSED
SW C-1 32	Unexpected Ether balance	CWE-667: Improper Locking	PASSED
SW C-1 31	Presence of unused variables	CWE-1164: Irrelevant Code	PASSED
SW C-1 30	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	PASSED
SW C-1 29	Typographical Error	CWE-480: Use of Incorrect Operator	PASSED
SW C-1 28	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	PASSED

<u>SW C-1 27</u>	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	PASSED
<u>SW C-1 25</u>	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	PASSED
<u>SW C-1 24</u>	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	PASSED
<u>SW C-1 23</u>	Requirement Violation	CWE-573: Improper Following of Specification by Caller	PASSED
<u>SW C-1 22</u>	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	PASSED
<u>SW C-1 21</u>	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	PASSED
<u>SW C-1 20</u>	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	PASSED
<u>SW C-11 9</u>	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	NOT PASSED
<u>SW C-11 8</u>	Incorrect Constructor Name	CWE-665: Improper Initialization	PASSED
<u>SW C-11 7</u>	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	PASSED

<u>SW C-11 6</u>	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
<u>SW C-11 5</u>	Authorization through tx.origin	CWE-477: Use of Obsolete Function	PASSED
<u>SW C-11 4</u>	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	PASSED
<u>SW C-11 3</u>	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	PASSED
<u>SW C-11 2</u>	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
<u>SW C-11 1</u>	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	PASSED
<u>SW C-11 0</u>	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	PASSED
<u>SW C-1 09</u>	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	PASSED
<u>SW C-1 08</u>	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED
<u>SW C-1 07</u>	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	PASSED
<u>SW C-1 06</u>	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	PASSED

<u>SW C-1 05</u>	Unprotected Ether Withdrawal	<u>CWE-284: Improper Access Control</u>	PASSED
<u>SW C-1 04</u>	Unchecked Call Return Value	<u>CWE-252: Unchecked Return Value</u>	PASSED
<u>SW C-1 03</u>	Floating Pragma	<u>CWE-664: Improper Control of a Resource Through its Lifetime</u>	PASSED
<u>SW C-1 02</u>	Outdated Compiler Version	<u>CWE-937: Using Components with Known Vulnerabilities</u>	PASSED
<u>SW C-1 01</u>	Integer Overflow and Underflow	<u>CWE-682: Incorrect Calculation</u>	PASSED
<u>SW C-1 00</u>	Function Default Visibility	<u>CWE-710: Improper Adherence to Coding Standards</u>	PASSED

Solid
Proofed

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY