SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY

SmarDex

Audit

Security Assessment
24. March, 2023

For

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	23. March 2023	• Layout project • Automated- /Manual-Security Testing • Summary

Network
Ethereum (ERC20)

Website
https://smardex.io

Telegram
https://t.me/realSmarDex
https://t.me/realSmarDexChat

Twitter
https://twitter.com/realSmarDex

Medium
https://github.com/SmarDex-Dev/smart-contracts

Description

SMARDEX is an Automated Market Maker (AMM) that addresses the issue of Impermanent Loss (IL) and in some cases transforms it into Impermanent Gain (IG). It is an open-source Smart Contract, which is a decentralized software that runs on compatible Ethereum Virtual Machine blockchains (such as Ethereum, Binance Smart Chain, Avalanche, Polygon, etc.). These blockchains are data exchange protocols that, similar to the Bitcoin blockchain, allow for the storage and transmission of information in a public, immutable, and decentralized manner. By using SMARDEX, users can exchange decentralized ERC20 tokens, which are digital assets.

Project Engagement

During the 20th of March 2023, **SmarDex 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/SmarDex-Dev/smart-contracts
  - Commit: 15e4dea

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
@openzeppelin/contracts/access/Ownable.sol	1
@openzeppelin/contracts/interfaces/IERC20.sol	3
@openzeppelin/contracts/security/ReentrancyGuard.sol	1
@openzeppelin/contracts/token/ERC20/ERC20.sol	1
@openzeppelin/contracts/token/ERC20/IERC20.sol	3
@openzeppelin/contracts/token/ERC20/extensions/draft-ERC20Permit.sol	2
@openzeppelin/contracts/token/ERC20/extensions/draft-IERC20Permit.sol	3
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol	3
@openzeppelin/contracts/utils/math/Math.sol	1
@openzeppelin/contracts/utils/math/SafeCast.sol	3

Tested Contract Files

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

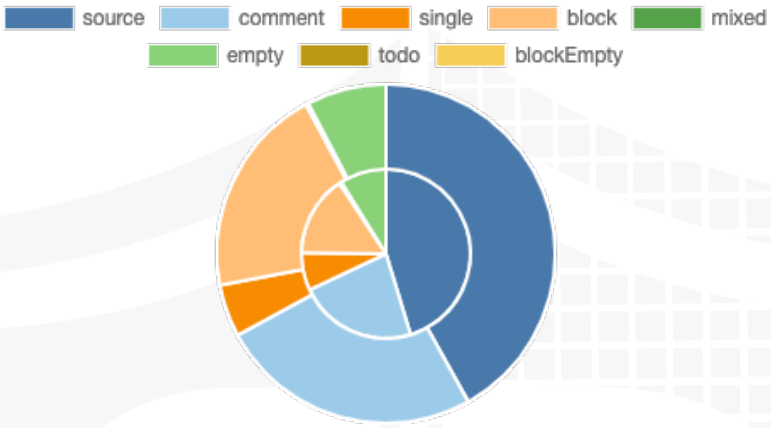
A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

v1.0

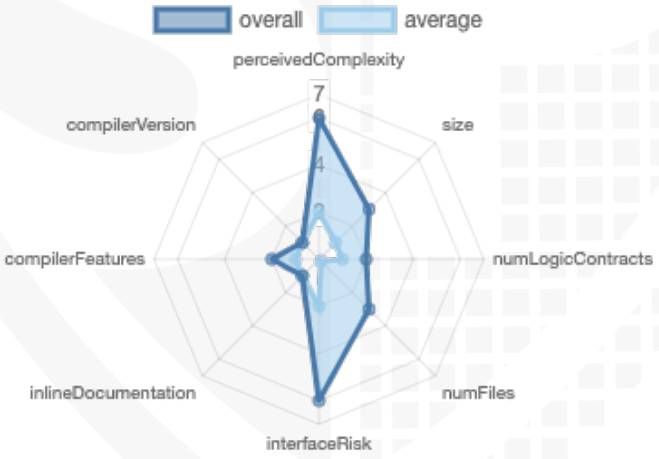
File Name	SHA-1 Hash
contracts/SmardexToken/SmardexToken.sol	e30b252b7322df1eb02f9fd07820c2bc149873b1
contracts/periphery/interfaces/IWETH.sol	24d04d452816528676d6ea027639a0b34fc30fdb
contracts/periphery/interfaces/ISmardexRouter.sol	0da3d4cd442102c208b2b4b18d8cb1bc5f27ad0d
contracts/periphery/SmardexRouter.sol	b3767646ce9f7061c4c2c0f44368a95bc7c393a6
contracts/periphery/libraries/BytesLib.sol	2392c85dd39d3a76458b51df18138f3e2ad9d5e4
contracts/periphery/libraries/PoolAddress.sol	9ec8b2e08baf0fd0ba7051b6330bdbaca5f3d268
contracts/periphery/libraries/Path.sol	6ad08235c27c5fdc878137f9acd254dae271dec6
contracts/periphery/libraries/PoolHelpers.sol	e141f5e2d71b4618dbcbd7830f5355bc0c879c7b
contracts/rewards/Staking.sol	ba54d43b7da9e5d5a2b72316ae4cf2ba4f15a011
contracts/rewards/AutoSwapper.sol	9f0f2e2772a4502fe1db98c594c648e2229825e1
contracts/rewards/interfaces/IFarmingRange.sol	8ea24e9de977d501df084de945b005ce9289aa55
contracts/rewards/interfaces/IStaking.sol	c40b99f3bdd7f033eb50cb173d3f32f403ab8140
contracts/rewards/interfaces/IAutoSwapper.sol	68bd6ff693fdf9b5771e4a170bb5e0e01c8d83ed
contracts/rewards/interfaces/IRewardManager.sol	94ae8fbf21377b402b22345211e3e20b46221f5d
contracts/rewards/FarmingRange.sol	ac7531d16ea3714c3d4ff1173d571ef7067d2233
contracts/rewards/RewardManager.sol	d0ceef7e7560acd4f8d1615632fe25305b02875a
contracts/core/interfaces/ISmardexPair.sol	7d62020e883e947ca7ea90c8cbd7b6e84c62e6a5
contracts/core/interfaces/ISmardexFactory.sol	3db7107eb57baf9143f19c274a5568cdc6839cba
contracts/core/interfaces/ISmardexSwapCallback.sol	6a19c3ee657a98fcf7537de14c0c7c63bd63858c
contracts/core/interfaces/ISmardexMintCallback.sol	a4d3bd5460016cb827fec80b94aa598917f9ac33
contracts/core/SmardexPair.sol	426f4cc94152c0955810182c90c18ff1127f51d2
contracts/core/SmardexFactory.sol	f0fbbd3045781fbb56ca0e3657b99c8374c02f43
contracts/core/libraries/SmardexLibrary.sol	f385f68196a2cbbea4227f5b8610f4ee5c9937e2
contracts/core/libraries/TransferHelper.sol	e48daf2db08016a10f6f9c2876e175a8ea7fc094

Metrics

Source Lines
v1.0



Risk Level
v1.0



Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	8	6	10	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	156	8

Version	External	Internal	Private	Pure	View
1.0	142	165	5	32	58

State Variables

Version	Total	Public
1.0	58	32

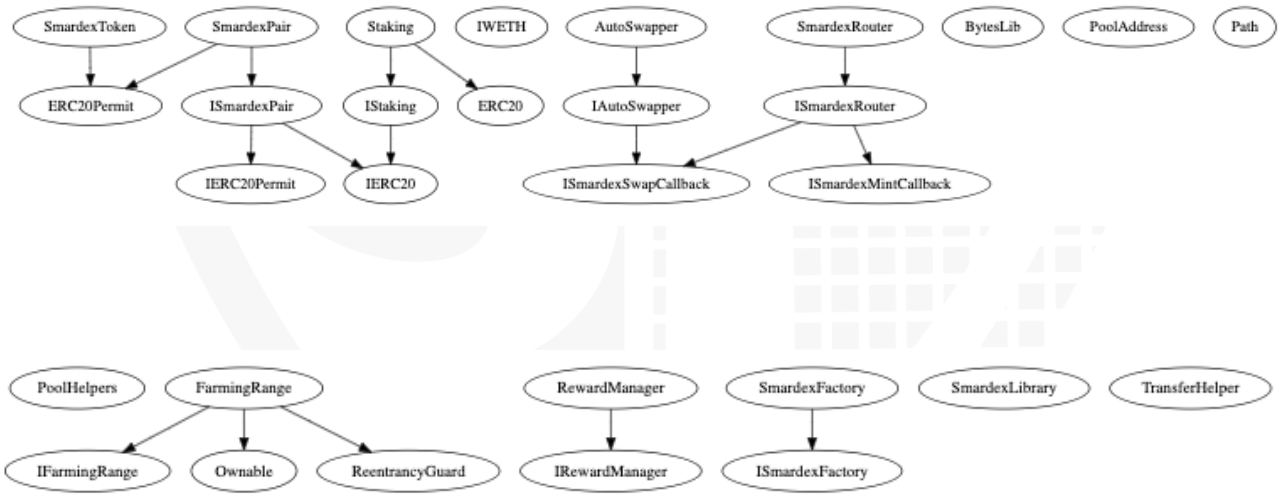
Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	`0.8.17` `>=0.8.17` `=0.8.17` `^0.8.17`		`yes`	`yes` (3 asm blocks)	

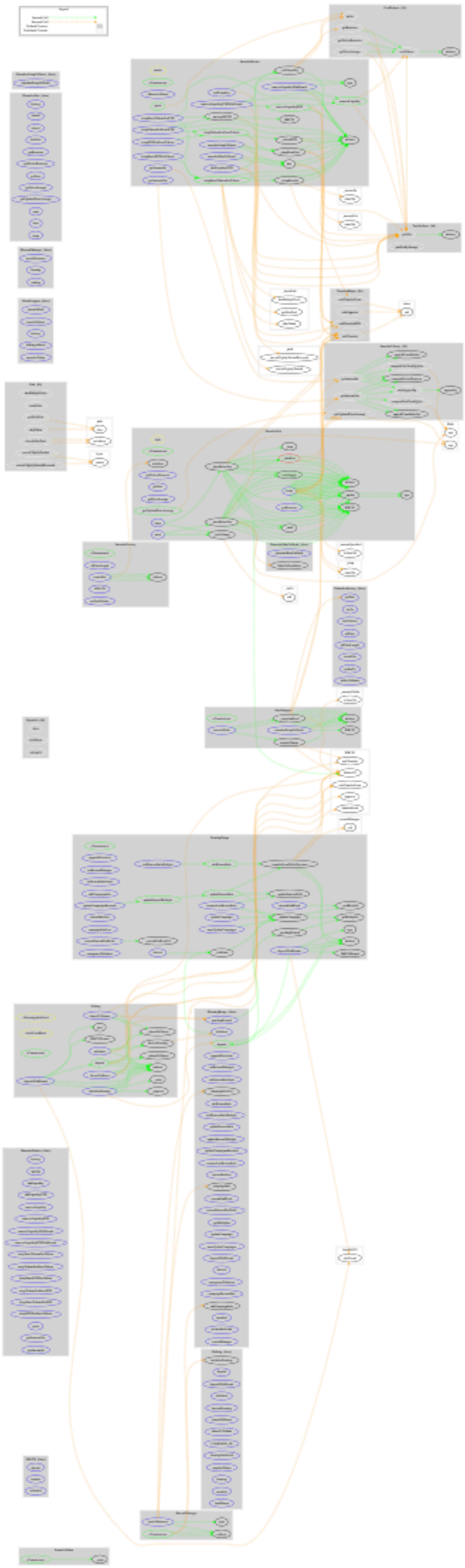
Version	Transfers ETH	Low-Level Calls	DelegateCall	Uses Hash Functions	ECRecover	New/Create/Create2
1.0	`yes`			`yes`		`yes` → `NewContract:FarmingRange` → `NewContract:Staking`

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. Overall checkup (Smart Contract Security)

Is contract an upgradeable

Name	
Is contract an upgradeable?	No

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

SmardexFactory

- createPair
- setFeeTo
- setFeeToSetter

SmarexPair

- initialize
- mint
- burn
- swap
  - lock

AutoSwapper

- executeWork
- transferTokens
- smardexSwapCallback

RewardManager

- resetAllowance

FarmingRange

- upgradePrecision
  - onlyOwner
- setRewardManager
  - onlyOwner
- setRewardInfoLimit
  - onlyOwner
- addCampaignInfo
  - onlyOwner
- addRewardInfo
  - onlyOwner
- addRewardInfoMultiple
  - onlyOwner
- updateRewardInfo
  - onlyOwner
- updateRewardMultiple
  - onlyOwner
- updateCampaignsRewards
  - onlyOwner
- removeLastRewardInfo
  - onlyOwner
- updateCampaign
  - nonReentrant
- massUpdateCampaigns
  - nonReentrant
- deposit
  - nonReentrant
- depositWithPermit
- withdraw
  - nonReentrant
- harvest
  - nonReentrant
- emergencyWithdraw
  - nonReentrant

SmarexRouter

```
smardexSwapCallback
smardexMintCallback
addLiquidity
    ensure
addLiquidityETH
    ensure
removeLiquidity
    ensure
removeLiquidityETH
    ensure
removeLiquidityWithPermit
removeLiquidityETHWithPermit
swapExactTokensForTokens
    ensure
swapTokensForExactTokens
    ensure
swapTokensForExactETH
    ensure
swapETHForExactTokens
    ensure
swapExactETHForTokens
    ensure
swapExactTokensForETH
    ensure
```

Staking

```
initializeFarming
deposit
isFarmingInitialized
checkUserBlock
depositWithPermit
withdraw
isFarmingInitialized
checkUserBlock
harvestFarming
```

Note:

- Modified UniswapV2 fork
  - SmardexFactory
  - SmardexPair
  - SmardexRouter
- Modified Alpaca Finance fork
  - FarmingRange (GrazingRange)

Comments

- *Deployer can set following addresses*
  - SmardexFactory
    - feeTo
    - feeToSetter

- *Existing Modifiers*
  - SmardexPair
    - lock
  - SmardexRouter
    - Ensure
  - Staking
    - isFarmingInitialized
    - checkUserBlock
- FarmingRange
  - Owner is able to
    - Remove last reward info
    - set
      - Reward info
      - Reward manager
    - Update
      - reward info
      - campaigns reward
    - Add
      - reward info
      - Campaign info
    - Upgrade precision

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/SmardexToken/SmardexToken.sol	1	—	14	14	7	5	6	—
🔍	contracts/periphery/interfaces/IWETH.sol	—	1	26	21	3	12	10	💰
🔍	contracts/periphery/interfaces/ISmardexRouter.sol	—	1	368	13	5	202	48	💰
📝	contracts/periphery/SmardexRouter.sol	1	—	591	440	283	111	224	💰🔷
📚	contracts/periphery/libraries/BytesLib.sol	1	—	98	98	52	30	140	💻
📚	contracts/periphery/libraries/PoolAddress.sol	1	—	50	46	25	18	11	🧮
📚	contracts/periphery/libraries/Path.sol	1	—	76	76	37	28	23	—
📚	contracts/periphery/libraries/PoolHelpers.sol	1	—	100	88	40	41	22	—
📝	contracts/rewards/Staking.sol	1	—	156	149	93	26	75	—
📝	contracts/rewards/AutoSwapper.sol	1	—	186	186	127	40	74	🧮
🔍	contracts/rewards/interfaces/IFarmingRange.sol	—	1	370	142	37	239	57	—
🔍	contracts/rewards/interfaces/IStaking.sol	—	1	130	41	11	82	29	—
🔍	contracts/rewards/interfaces/IAutoSwapper.sol	—	1	40	16	6	23	13	—
🔍	contracts/rewards/interfaces/IRewardManager.sol	—	1	26	13	5	14	7	—
📝	contracts/rewards/FarmingRange.sol	1	—	528	494	355	98	225	♻️
📝	contracts/rewards/RewardManager.sol	1	—	46	46	22	16	48	🧮🌀
🔍	contracts/core/interfaces/ISmardexPair.sol	—	1	179	62	16	114	29	—
🔍	contracts/core/interfaces/ISmardexFactory.sol	—	1	61	17	4	40	17	—
🔍	contracts/core/interfaces/ISmardexSwapCallback.sol	—	1	12	11	3	7	3	—
🔍	contracts/core/interfaces/ISmardexMintCallback.sol	—	1	26	25	10	13	3	—
📝	contracts/core/SmardexPair.sol	1	—	490	455	324	83	206	🧮
📝	contracts/core/SmardexFactory.sol	1	—	57	57	37	13	40	🧮
📚	contracts/core/libraries/SmardexLibrary.sol	1	—	520	428	236	152	42	—
📚	contracts/core/libraries/TransferHelper.sol	1	—	42	42	28	10	26	—
📝📚🔍	**Totals**	**14**	**10**	**4192**	**2980**	**1766**	**1417**	**1378**	💻💰🔷🧮🌀♻️

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

No medium issues

Low issues

Issue	File	Type	Line	Description
#1	FarmingRange.sol	Missing Zero Address Validation (missing-zero-check)	47	Check that the address is not zero
#2	Staking.sol	Comparison to Boolean constants	29, 49	It is not recommended to compare variables with boolean constants in the "*require*" statements because it is not needed
#3	SmardexFactory.sol	Missing Events Arithmetic	47, 53	Emit an event for critical parameter changes
#4	RewardManager.sol	Missing Events Arithmetic	40	Emit an event for critical parameter changes

Informational issues

Issue	File	Type	Line	Description
#1	SmardexPair.sol	State variables that could be declared immutable (immutable-states)	55	Add the `immutable` attributes to state variables that never change

#2	Reward Manager.sol	Unused return values	40	Ensure that all the return values of the function calls are used and handle both success and failure cases if needed by the business logic
#3	PoolHelpers.sol	Functions that are not used	52, 74	Remove unused functions. Before removing check the function, it could be possible, that you forget to implement it into the contract
#4	All	NatSpec documentation missing	—	If you started to comment your code, also comment all other functions, variables etc.

Audit Comments

We recommend you to use the special form of comments (NatSpec Format, Follow 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 variables, functions etc. do.

24. March 2023:

- There is still an owner (Owner still has not renounced ownership)
- Owner can deploy a new version of the contract which can change any limit and give owner new privileges
- The balance of the AutoSwapper contract will be sent to the staking address that will be set by the deployer at the time of deployment and it cannot be changed.
- We recommend SmarDex team to thoroughly unit test all the contracts to rule out any possibility of unintended behaviour by the contracts.
- Read 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	**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

SolidProofed

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY