



**SOLID**Proof  
*Bring trust into your projects*

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

MADE IN GERMANY

# CleverAgent Finance

# Audit

**Security Assessment**

09.August,2022

**For**



[SolidProof.io](https://solidproof.io)



[@solidproof\\_io](https://t.me/solidproof_io)

Disclaimer	2
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	10
Source Lines	11
Risk Level	11
Capabilities	12
Inheritance Graph	13
CallGraph	14
Scope of Work/Verify Claims	15
Modifiers and public functions	22
Source Units in Scope	23
Critical issues	24
High issues	24
Medium issues	24
Low issues	24
Informational issues	25
Audit Comments	25
SWC Attacks	27

## 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.August,2022	<ul style="list-style-type: none"><li>• Layout project</li><li>• Automated- /Manual-Security Testing</li><li>• Summary</li></ul>
1.1	09.August,2022	<ul style="list-style-type: none"><li>• Reaudit</li></ul>

## Network

Binance Network (BSC)

## Website

<https://cleveragent.finance>

## Twitter

<https://twitter.com/Ceveragent>

## Discord

<https://discord.io/Cleveragent>

## Telegram

<https://t.me/cleveragent>

## YouTube

<https://www.youtube.com/channel/UCL86G-Mlwi8LiB8Aj3vBE1g>

## Medium

<https://medium.com/@cleveragentfinance>

## Description

Clever Agent is a saving protocol that's based on the Binance Smart Chain network which offer investors huge opportunities to earn up to 15% interest rate per year on the deposited stablecoins (USDT, USDC, BUSD or DAI) with instant paid out.

Clever Agent also offers an additional APY in the form of the lottery for lucky depositors besides the constant APY. The more and earlier your stablecoins are deposited, the higher APY and the more chance you will win the lottery.

## Project Engagement

During the 1<sup>st</sup> of August 2022, **Clever Agent** team engaged Solidproof.io to audit the smart contracts that they created. The engagement was technical in nature and focused on identifying the security flaws in the design and implementation of the contracts. They provided Solidproof.io with access to their code repository and whitepaper.

## Logo



## Contract Links

v1.0

<https://github.com/cleveragentfinance/contract/tree/main/contracts>

Commit: 78a531b8191ff1dc8efc88b6b75442c4c7db433a

V1.1

<https://github.com/cleveragentfinance/contract/tree/main/contracts>

Commit: 4c528112e7fc87da4a91d352313ef4e1bbe98467

# 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)
<b>Critical</b>	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.
<b>High</b>	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.
<b>Medium</b>	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.
<b>Low</b>	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.
<b>Informational</b>	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 analyzing 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:

### AgentManager

```
@openzeppelin/contracts/token/ERC20/IERC20.sol
@openzeppelin/contracts/access/Ownable.sol
@openzeppelin/contracts/utils/math/SafeMath.sol
@uniswap/v2-periphery/contracts/interfaces/IUniswapV2Router01.sol
@openzeppelin/contracts/proxy/transparent/TransparentUpgradeableProxy.sol
./interfaces/IAgent.sol
```

### AutoFarm

```
@openzeppelin/contracts/utils/Address.sol
@openzeppelin/contracts/token/ERC20/ERC20.sol
@openzeppelin/contracts/token/ERC20/IERC20.sol
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol
@openzeppelin/contracts/access/Ownable.sol
@openzeppelin/contracts/utils/math/SafeMath.sol
@openzeppelin/contracts/security/ReentrancyGuard.sol
@uniswap/v2-periphery/contracts/interfaces/IUniswapV2Router02.sol
```

### CALottery

```
@openzeppelin/contracts/access/Ownable.sol
@openzeppelin/contracts/security/ReentrancyGuard.sol
@openzeppelin/contracts/token/ERC20/IERC20.sol
@openzeppelin/contracts/utils/Address.sol
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol
./interfaces/IRandomNumberGenerator.sol
./interfaces/ICALotteryOld.sol
```

### Free

```
@openzeppelin/contracts/access/Ownable.sol
@openzeppelin/contracts/utils/math/SafeMath.sol
@openzeppelin/contracts/token/ERC20/ERC20.sol
```

### InsuraceAgent

```
@openzeppelin/contracts/utils/Address.sol
@openzeppelin/contracts/token/ERC20/ERC20.sol
@openzeppelin/contracts/token/ERC20/IERC20.sol
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol
@openzeppelin/contracts/access/Ownable.sol
@openzeppelin/contracts/utils/math/SafeMath.sol
@openzeppelin/contracts/security/ReentrancyGuard.sol
@uniswap/v2-periphery/contracts/interfaces/IUniswapV2Router02.sol
```



## Liquidity

- 📄 @openzeppelin/contracts/utils/Address.sol
- 📄 @openzeppelin/contracts/token/ERC20/ERC20.sol
- 📄 @openzeppelin/contracts/token/ERC20/IERC20.sol
- 📄 @openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol
- 📄 @openzeppelin/contracts/access/Ownable.sol
- 📄 @openzeppelin/contracts/utils/math/SafeMath.sol
- 📄 @openzeppelin/contracts/security/ReentrancyGuard.sol
- 📄 @uniswap/v2-periphery/contracts/interfaces/IUniswapV2Router02.sol

## RandomNumberGenerator

- 📄 @openzeppelin/contracts/access/Ownable.sol
- 📄 @openzeppelin/contracts/token/ERC20/IERC20.sol
- 📄 @openzeppelin/contracts/utils/Address.sol
- 📄 @openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol
- 📄 @chainlink/contracts/src/v0.8/VRFConsumerBase.sol
- 📄 ./interfaces/IRandomNumberGenerator.sol
- 📄 ./interfaces/ICALottery.sol

## Tested Contract Files

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

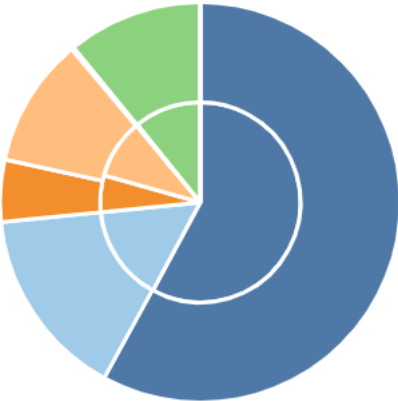
A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

v1.0

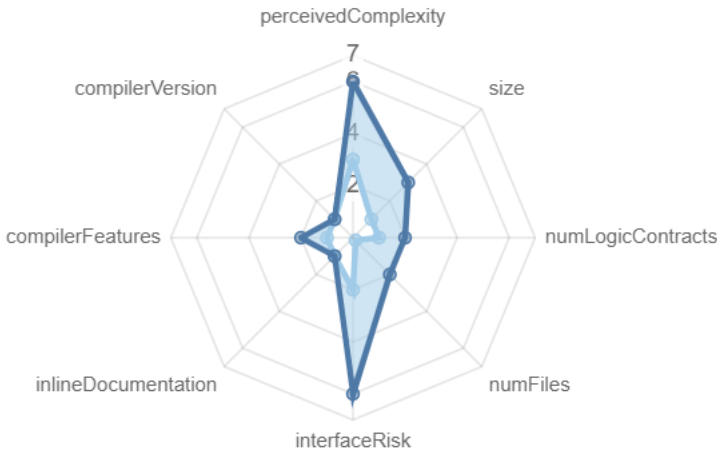
File Name	SHA-1 Hash
contracts/interfaces/IAgentManager.sol	9a9b3e78f7726db06133ae57f2d3531534c38f20
contracts/interfaces/ICALotteryOld.sol	31486420ca96565dece9ca9ab09f3ee11682e550
contracts/interfaces/IRandomNumberGenerator.sol	e7df1ce109dff9a7ef1806e226daa35480d50364
contracts/interfaces/ICALottery.sol	cc505efeb0bbb17ff7429e4925015b8eea6db4a4
contracts/interfaces/IAgent.sol	fedd99032ef5e14e30841c73e7889091908a553b
contracts/Free.sol	931127b24168f6ce60b9b1621b520f5a34130c2f
contracts/AgentManager.sol	26a08991a7a89e801a50708cdb236dce02c8b6a5
contracts/CALottery.sol	6e142ee282537db0d75b6fb28848d2528c7ecaf6
contracts/Helper.sol	59e3fa0674107facf83d313689018592a656dd12
contracts/InsurAceAgent.sol	a378c3fa38a947518837b3fec0050f92a6b271ef
contracts/Timelock.sol	701687061ce625d0cb6f7bfb51c897231d2bcc77
contracts/Liquidity.sol	eb2336b994bcd9b0477b8581920915cc6d5b3b3a
contracts/Randomnumbergenerator.sol	e0b223e13136f087e001ad224b40767ef7ebfa7a
contracts/Autofarm.sol	d706dc954ab423364905648e91c4268515887788

# Metrics

## Source Lines v1.0



## Risk Level v1.0



## Capabilities

### Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	9	1	13	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	238	10

Version	External	Internal	Private	Pure	View
1.0	157	144	0	36	93

### State Variables

Version	Total	Public
1.0	80	74

## Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	<div>^0.8.0</div> <div>^0.8.4</div>		Yes		

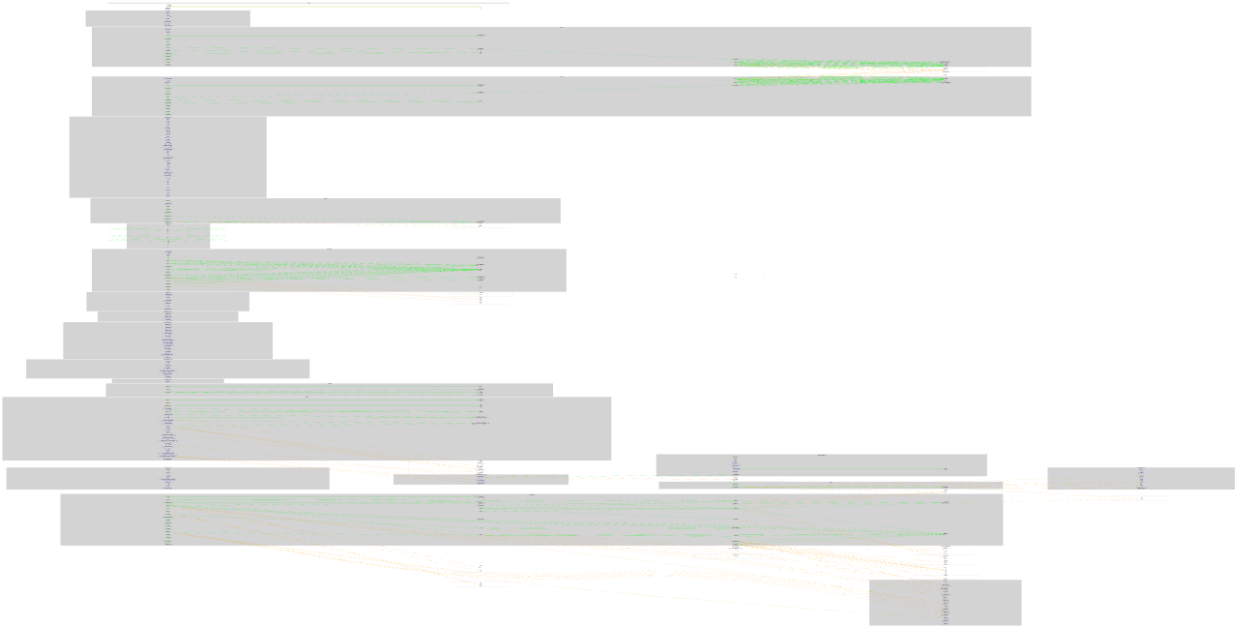
Version	Transfers ETH	Low-Level Calls	DelegateCall	Uses Hash Functions	EC Recover	New/Create/Create2
1.0	Yes			Yes		

## Inheritance Graph v1.0



# Call Graph

v1.0

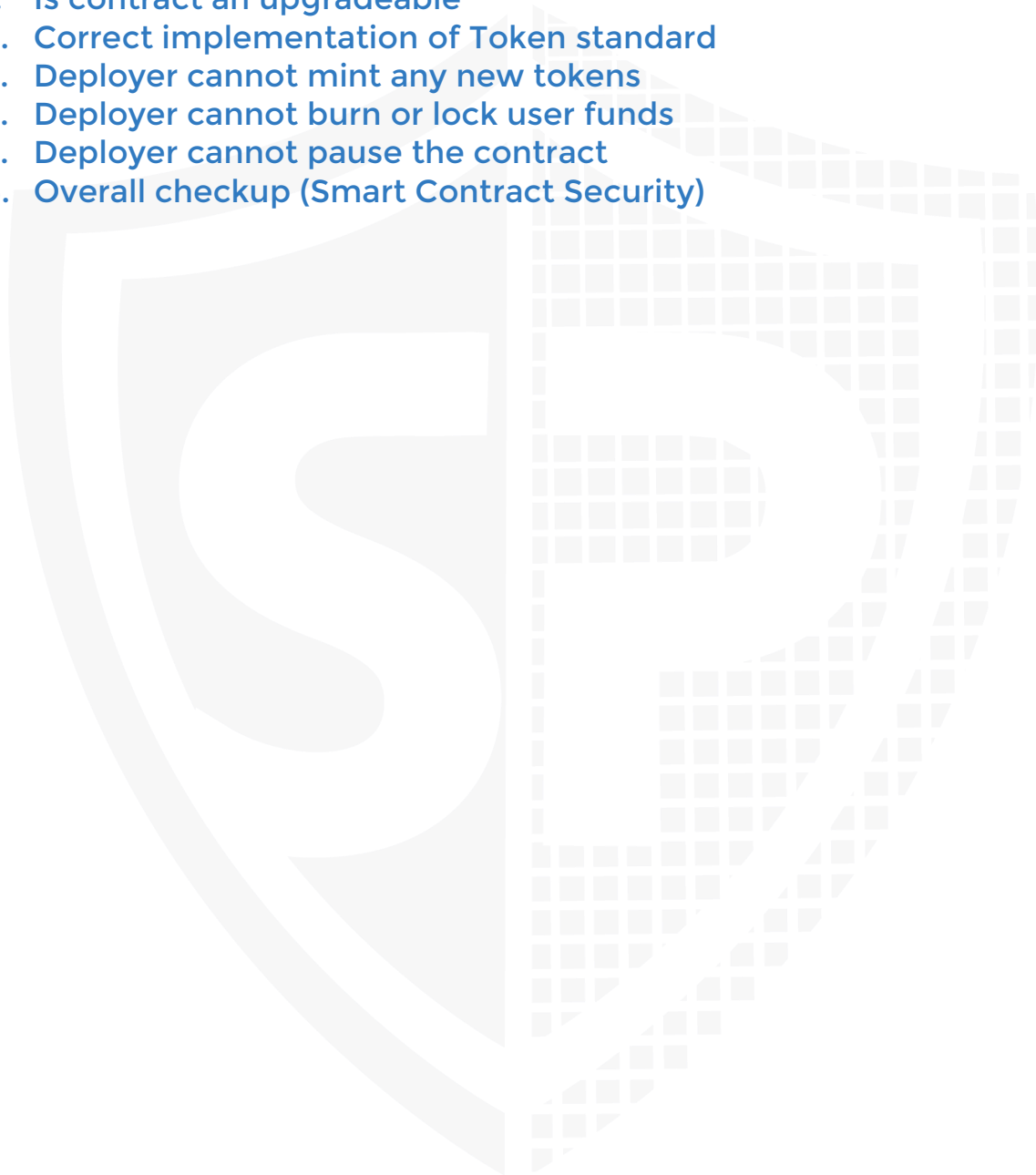


## Scope of Work/Verify Claims

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. Overall checkup (Smart Contract Security)



## Is contract an upgradeable

Name	
Is contract an upgradeable?	Yes

### Comment:

The owner can update the contract code after deployment by uploading a new one





## 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 and last Token ID	N/A		



## 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 cannot pause			



## 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

## AgentManager

```

♦ initialize
♦ updateUser
♦ updatePool
♦ massUpdatePools
♦ deposit
♦ withdraw
♦ emergencyWithdraw
♦ cancelPendingWithdraw
♦ harvest
♦ harvestAll
♦ add
  (M) onlyOwner
♦ addTarget
  (M) onlyOwner
♦ UpdateAgents
  (M) onlyOwner
♦ updateAgentToNewContract
  (M) onlyOwner
♦ setAutoTarget
  (M) onlyOwner
♦ toggleAutoTarget
  (M) onlyOwner
♦ setFeeAddress
♦ updateConfig
  (M) onlyOwner
♦ distributeProfit
  (M) onlyOwner
  
```

## AutoFarm

```

♦ <Constructor> 💰
♦ init
♦ deposit
  (M) onlyOwner
♦ withdraw
  (M) onlyOwner
  
```

## RandomNumberGenerator

```

♦ <Constructor>
♦ getRandomNumber
♦ setFee
  (M) onlyOwner
♦ setKeyHash
  (M) onlyOwner
♦ setLotteryAddress
  (M) onlyOwner
♦ withdrawTokens
  
```

## InsuraceAgent

```

♦ <Constructor> 💰
♦ init
♦ deposit
  (M) onlyOwner
♦ withdraw
  (M) onlyOwner
♦ unlockWithdraw
  (M) onlyOwner
♦ harvest
  (M) onlyOwner
♦ unlockHarvest
  (M) onlyOwner
  
```

## CALottery

```























♦ buyTickets
  (M) notContract
  (M) nonReentrant
♦ claimTickets
  (M) notContract
  (M) nonReentrant
♦ closeLottery
  (M) onlyOperator
  (M) nonReentrant
♦ drawFinalNumberAndMakeLotteryClaimable
  (M) onlyOperator
  (M) nonReentrant
♦ changeRandomGenerator
  (M) onlyOwner
♦ injectFunds
  (M) onlyOwnerOrInjector
♦ startLottery
  (M) onlyOperator
  
```

## Comments:

- The owner add targets, update agents to a new contract and distribute profits.
- Owner can manage the lottery like, start lottery, end lottery and inject funds into it.
- Owner can withdraw and harvest from farm

# Source Units in Scope

## v1.0

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
	contracts/interfaces/IAgentManager.sol	—————	1	46	39	31	3	15	—————
	contracts/interfaces/ICALotteryOld.sol	—————	1	70	11	3	43	15	—————
	contracts/interfaces/IRandomNumberGenerator.sol	—————	1	19	8	3	10	7	—————
	contracts/interfaces/ICALottery.sol	—————	1	68	11	3	42	15	—————
	contracts/interfaces/IAgent.sol	—————	1	20	5	3	1	31	—————
	contracts/Free.sol	1	—————	39	35	20	9	16	—————
	contracts/AgentManager.sol	1	1	561	551	461	43	419	
	contracts/CALottery.sol	1	—————	710	649	379	163	290	
	contracts/Helper.sol	1	—————	40	40	26	8	23	—————
	contracts/InsurAceAgent.sol	1	4	267	170	145	7	311	
	contracts/Timelock.sol	2	—————	300	288	137	106	87	
	contracts/Liquidity.sol	1	1	232	185	156	9	265	
	contracts/Randomnumbergenerator.sol	1	—————	110	110	48	50	46	—————
	contracts/Autofarm.sol	1	2	255	198	169	9	298	
	Totals	10	13	2737	2300	1584	503	1838	

## 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	All	A floating pragma is set	7	The current pragma Solidity directive is „^0.8.0/6“.
#2	AgentManager.sol	Access Control	257	The function has public visibility which means users can pass some random _pid for a particular amount. We suggest to put a check whether the user exists in the particular pool



#3	AgentManager.sol	Missing zero check	91-96,209,370,434,456,469	Check that the address is not zero
#4	AgentManager.sol	Missing Events	90,354,370,388,445,452,456,461	Emit an event for critical parameter changes.
#5	AgentManager.sol	Unnecessary Check	393	Unnecessary check because it was checked before and uint256 can't be below 0. It will always be true
#6	Liquidity.sol	Missing Events	134	Emit an event for critical parameter changes.
#7	AutoFarm.sol	Missing Events	100,115	Emit an event for critical parameter changes.
#8	Free.sol	Missing zero check	19	Check that the address is not zero
#9	RandomNumberGenerator.sol	Missing zero check	75	Check that the address is not zero
#10	Timelock.sol	Missing zero check	207	Check that the address is not zero

## Informational issues

Issue	File	Type	Line	Description
#1	AutoFarm.sol	Redundant Code	182,186,190,234,238,248,252	These functions are redundant and have no functionality in code. They, should either be removed or used.
#2	AgentManager/AutoFarm/InsuranceAgent/CALottery/Liquidity.sol	Unused Return Value	-	Always check and take care of the return value from a function call

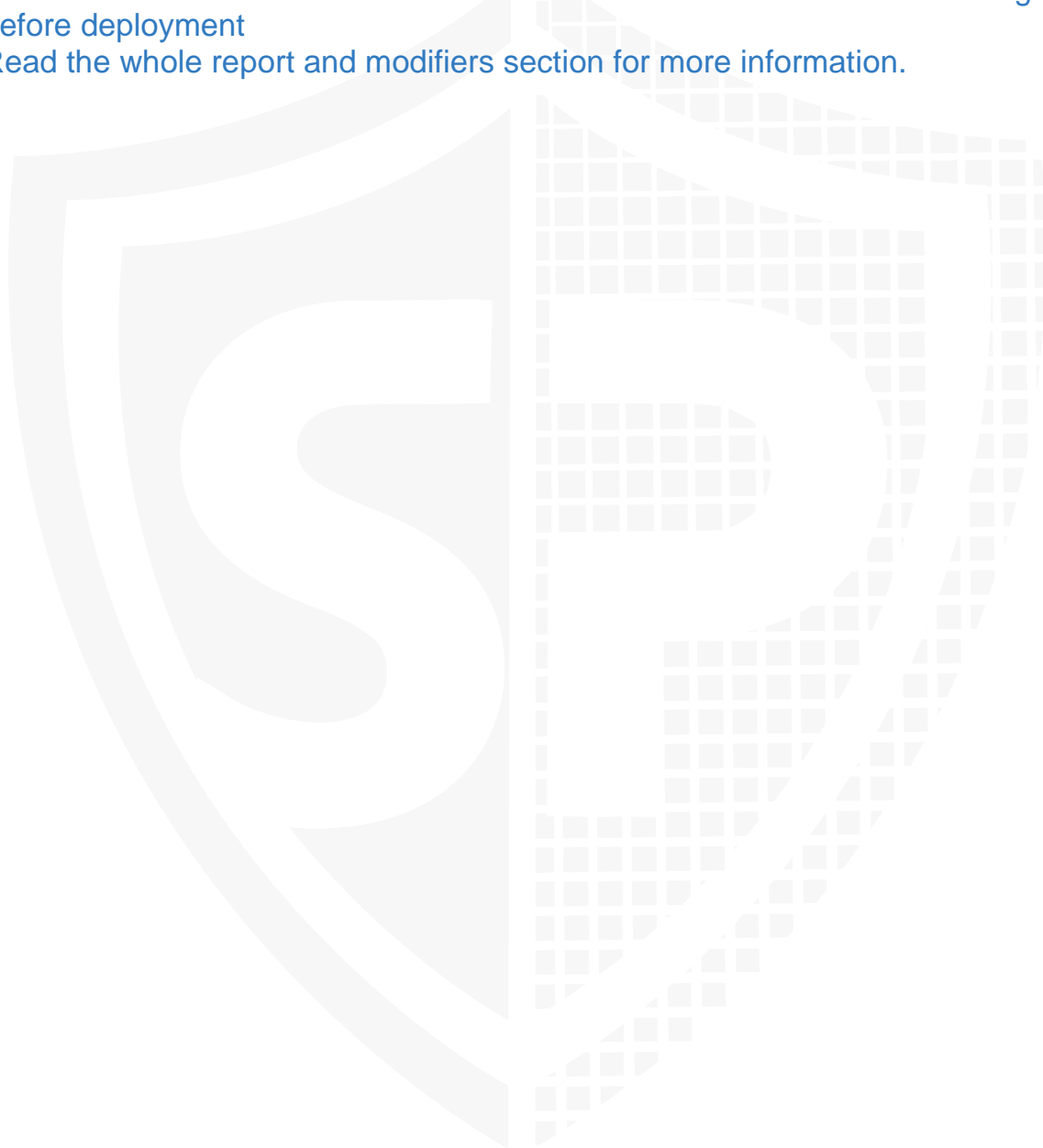
## Audit Comments

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.

## 09. August, 2022:

- There is still an owner (Owner still has not renounced ownership)
- We recommend to use randomizations by external sources like VRF because solidity has no randomization feature of its own.
- We recommend to unit test these tests with more than 95% of test coverage before deployment
- Read the whole report and modifiers section for more information.



## SWC Attacks

ID	Title	Relationships	Status
<a href="#">SWC-1136</a>	Unencrypted Private Data On-Chain	<a href="#">CWE-767: Access to Critical Private Variable via Public Method</a>	PASSED
<a href="#">SWC-1135</a>	Code With No Effects	<a href="#">CWE-1164: Irrelevant Code</a>	PASSED
<a href="#">SWC-1134</a>	Message call with hardcoded gas amount	<a href="#">CWE-655: Improper Initialization</a>	PASSED
<a href="#">SWC-1133</a>	Hash Collisions With Multiple Variable Length Arguments	<a href="#">CWE-294: Authentication Bypass by Capture-replay</a>	PASSED
<a href="#">SWC-1132</a>	Unexpected Ether balance	<a href="#">CWE-667: Improper Locking</a>	PASSED
<a href="#">SWC-1131</a>	Presence of unused variables	<a href="#">CWE-1164: Irrelevant Code</a>	PASSED

131			
SWC-130	Right-To-Left-Override control character (U+202E)	<a href="#">CWE-451: User Interface (UI) Misrepresentation of Critical Information</a>	PASSED
SWC-129	Typographical Error	<a href="#">CWE-480: Use of Incorrect Operator</a>	PASSED
SWC-128	DoS With Block Gas Limit	<a href="#">CWE-400: Uncontrolled Resource Consumption</a>	PASSED
SWC-127	Arbitrary Jump with Function Type Variable	<a href="#">CWE-695: Use of Low-Level Functionality</a>	PASSED
SWC-125	Incorrect Inheritance Order	<a href="#">CWE-696: Incorrect Behavior Order</a>	PASSED
SWC-1	Write to Arbitrary	<a href="#">CWE-123: Write-what-where Condition</a>	PASSED

<u>1</u> <u>2</u> <u>4</u>	Storage Location		
<u>S</u> <u>W</u> <u>C</u> : <u>1</u> <u>2</u> <u>3</u>	Requirement Violation	<a href="#">CWE-573: Improper Following of Specification by Caller</a>	PASSED
<u>S</u> <u>W</u> <u>C</u> : <u>1</u> <u>2</u> <u>2</u>	Lack of Proper Signature Verification	<a href="#">CWE-345: Insufficient Verification of Data Authenticity</a>	PASSED
<u>S</u> <u>W</u> <u>C</u> : <u>1</u> <u>2</u> <u>1</u>	Missing Protection against Signature Replay Attacks	<a href="#">CWE-347: Improper Verification of Cryptographic Signature</a>	PASSED
<u>S</u> <u>W</u> <u>C</u> : <u>1</u> <u>2</u> <u>0</u>	Weak Sources of Randomness from Chain Attributes	<a href="#">CWE-330: Use of Insufficiently Random Values</a>	PASSED
<u>S</u> <u>W</u> <u>C</u> : <u>1</u> <u>1</u> <u>1</u> <u>9</u>	Shadowing State Variables	<a href="#">CWE-710: Improper Adherence to Coding Standards</a>	NOT PASSED

<a href="#">S</a> <a href="#">W</a> <a href="#">C</a> : <a href="#">1</a> <a href="#">1</a> <a href="#">8</a>	Incorrect Constructor Name	<a href="#">CWE-665: Improper Initialization</a>	PASSED
<a href="#">S</a> <a href="#">W</a> <a href="#">C</a> : <a href="#">1</a> <a href="#">1</a> <a href="#">7</a>	Signature Malleability	<a href="#">CWE-347: Improper Verification of Cryptographic Signature</a>	PASSED
<a href="#">S</a> <a href="#">W</a> <a href="#">C</a> : <a href="#">1</a> <a href="#">1</a> <a href="#">6</a>	Timestamp Dependence	<a href="#">CWE-829: Inclusion of Functionality from Untrusted Control Sphere</a>	PASSED
<a href="#">S</a> <a href="#">W</a> <a href="#">C</a> : <a href="#">1</a> <a href="#">1</a> <a href="#">5</a>	Authorization through tx.origin	<a href="#">CWE-477: Use of Obsolete Function</a>	PASSED
<a href="#">S</a> <a href="#">W</a> <a href="#">C</a> : <a href="#">1</a> <a href="#">1</a> <a href="#">4</a>	Transaction Order Dependence	<a href="#">CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')</a>	PASSED
<a href="#">S</a> <a href="#">W</a> <a href="#">C</a> : <a href="#">1</a> <a href="#">1</a> <a href="#">3</a>	DoS with Failed Call	<a href="#">CWE-703: Improper Check or Handling of Exceptional Conditions</a>	PASSED

<a href="#">S</a> <a href="#">W</a> <a href="#">C</a> : 1 1 2	Delegatecall to Untrusted Callee	<a href="#">CWE-829: Inclusion of Functionality from Untrusted Control Sphere</a>	PASSED
<a href="#">S</a> <a href="#">W</a> <a href="#">C</a> : 1 1 1	Use of Deprecated Solidity Functions	<a href="#">CWE-477: Use of Obsolete Function</a>	PASSED
<a href="#">S</a> <a href="#">W</a> <a href="#">C</a> : 1 1 0	Assert Violation	<a href="#">CWE-670: Always-Incorrect Control Flow Implementation</a>	PASSED
<a href="#">S</a> <a href="#">W</a> <a href="#">C</a> : 1 0 9	Uninitialized Storage Pointer	<a href="#">CWE-824: Access of Uninitialized Pointer</a>	PASSED
<a href="#">S</a> <a href="#">W</a> <a href="#">C</a> : 1 0 8	State Variable Default Visibility	<a href="#">CWE-710: Improper Adherence to Coding Standards</a>	PASSED
<a href="#">S</a> <a href="#">W</a> <a href="#">C</a> : 1 0 7	Reentrancy	<a href="#">CWE-841: Improper Enforcement of Behavioral Workflow</a>	PASSED

<a href="#">S</a> <a href="#">W</a> <a href="#">C</a> : : 1 0 6	Unprotected SELFDESTR UCT Instruction	<a href="#">CWE-284: Improper Access Control</a>	PASSED
<a href="#">S</a> <a href="#">W</a> <a href="#">C</a> : : 1 0 5	Unprotected Ether Withdrawal	<a href="#">CWE-284: Improper Access Control</a>	PASSED
<a href="#">S</a> <a href="#">W</a> <a href="#">C</a> : : 1 0 4	Unchecked Call Return Value	<a href="#">CWE-252: Unchecked Return Value</a>	PASSED
<a href="#">S</a> <a href="#">W</a> <a href="#">C</a> : : 1 0 3	Floating Pragma	<a href="#">CWE-664: Improper Control of a Resource Through its Lifetime</a>	NOT PASSED
<a href="#">S</a> <a href="#">W</a> <a href="#">C</a> : : 1 0 2	Outdated Compiler Version	<a href="#">CWE-937: Using Components with Known Vulnerabilities</a>	PASSED
<a href="#">S</a> <a href="#">W</a> <a href="#">C</a> : : 1 0 1	Integer Overflow and Underflow	<a href="#">CWE-682: Incorrect Calculation</a>	PASSED



<div> <div> <div>S</div> <div>W</div> <div>C</div> <div>.</div> <div>1</div> <div>1</div> <div>0</div> <div>0</div> <div>0</div> <div>1</div> </div> </div>	<div>Function Default Visibility</div>	<div> <div>CWE-710: Improper Adherence to Coding Standards</div> </div>	<div>PASSED</div>
---	--	---	-------------------





[SolidProof.io](https://SolidProof.io)



[@solidproof\\_io](https://t.me/solidproof_io)

*Solid  
Proofed*

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

  
MADE IN GERMANY