



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

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

MADE IN GERMANY

# AutoDCA

# Audit

**Security Assessment  
09. March, 2023**

For



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



[@solidproof\\_io](https://@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	11
Risk Level	11
Capabilities	12
Inheritance Graph	13
CallGraph	14
Scope of Work/Verify Claims	15
Modifiers and public functions	23
Source Units in Scope	25
Critical issues	26
High issues	26
Medium issues	26
Low issues	26
Informational issues	26
Audit Comments	26
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	4. March 2023	<ul style="list-style-type: none"><li>• Layout project</li><li>• Automated- /Manual-Security Testing</li><li>• Summary</li></ul>
1.1	09. March 2023	<ul style="list-style-type: none"><li>• Reaudit</li></ul>

## **Network**

Arbitrum

## **Website**

<https://autodca.io/>

## **Twitter**

[https://twitter.com/AutoDCA\\_io](https://twitter.com/AutoDCA_io)

## Description

We develop distinctive investment strategies that provide exposure to a variety of cryptocurrencies, market trends, and narratives.

## Project Engagement

During the Date of 4 March 2023, **AutoDCA 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

- <https://github.com/autodca/dca-contracts/tree/master/contracts>
- Commit: 3853f60

### v1.1

- <https://github.com/autodca/dca-contracts/tree/master/contracts>
- Commit: 1cd8de7

**Note** - The AutoDCA team has decided to remove the Fee Collector contract's code from the audit scope in the latest version of the audit(1.1) by declaring it as Not for Public use contract.

# 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 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:

```
@openzeppelin/contracts/access/Ownable.sol  
@openzeppelin/contracts/access/AccessControl.sol  
@openzeppelin/contracts/token/ERC20/IERC20.sol  
@uniswap/v3-periphery/contracts/interfaces/ISwapRouter.sol  
hardhat/console.sol  
.IAccessManager.sol  
.IFeeManager.sol  
.DCATypes.sol  
.IFeeCollector.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/ DefaultAccessManager.sol	3944c1fe75aad4d95ea6ee9500bb3081 6dc61883
contracts/IFeeManager.sol	ef206b9c1279fa5029518d8bef6bca8a3 f747fc0
contracts/ IAccessManager.sol	014fc111829324d6e6bc3858048ce845 8cca7544
contracts/IFeeCollector.sol	fb5333d9bad98cba017ca021a68bd41b 4be200fb
contracts/ DefaultFeeManager.sol	4172212cf5ca703811e29bb1af0eb83f 18004bd
contracts/FeeCollector.sol	b32f573ea4307a4790559a5976edb911 faf33a33
contracts/DCATypes.sol	ec1fdef35b855fbca63f51ee4ee88fa49a 3eebee
contracts/ DCAStrategyManager.sol	a6fb671a228b542875ef52cbb3d888fdc 84c27a9

## v1.1

File Name	SHA-1 Hash
contracts/ DCAStrategyManager.sol	3ded30051e01685841527a6e8254b532 2a43d38f
contracts/ DefaultAccessManager.sol	3944c1fe75aad4d95ea6ee9500bb3081 6dc61883
contracts/IFeeManager.sol	ef206b9c1279fa5029518d8bef6bca8a3f 747fc0
contracts/ IAccessManager.sol	014fc111829324d6e6bc3858048ce8458 cca7544
contracts/ DefaultFeeManager.sol	4172212cfe5ca703811e29bb1af0eb83f1 8004bd
contracts/DCATypes.sol	ec1fdef35b855fbca63f51ee4ee88fa49a 3eebee

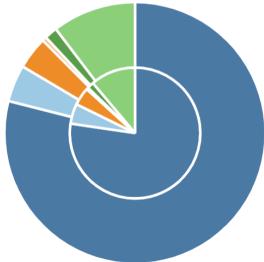
## Comments

The AutoDCA team has decided to remove the Fee Collector contract's code from the audit scope in the latest version of the audit by declaring it as Not for Public use contract.

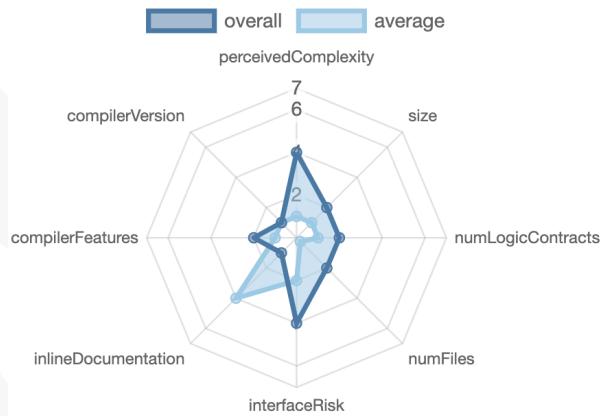
# Metrics

## Source Lines v1.0

source comment single block mixed  
empty todo blockEmpty



## Risk Level v1.0



# Capabilities

## Components v1.0

🌐 Public	💰 Payable
31	3

External	Internal	Private	Pure	View
17	29	0	0	12

### StateVariables

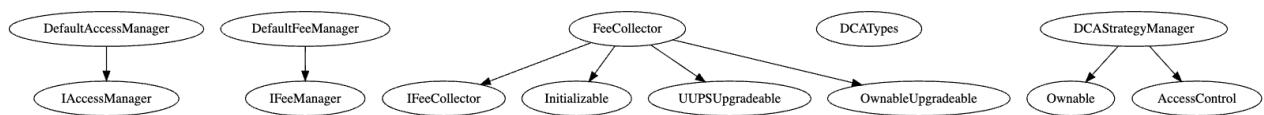
Total	🌐 Public
18	10

### Capabilities

Solidity Versions observed	🧪 Experimental Features	💰 Can Receive Funds	💻 Uses Assembly	💣 Has Destroyable Contracts
^0.8.9		yes		
.Transfer ETH	⚡ Low-Level Calls	👥 DelegateCall	⛓️ Uses Hash Functions	💥 ECRecover
yes			yes	
♻️ TryCatch	Σ Unchecked			
yes				

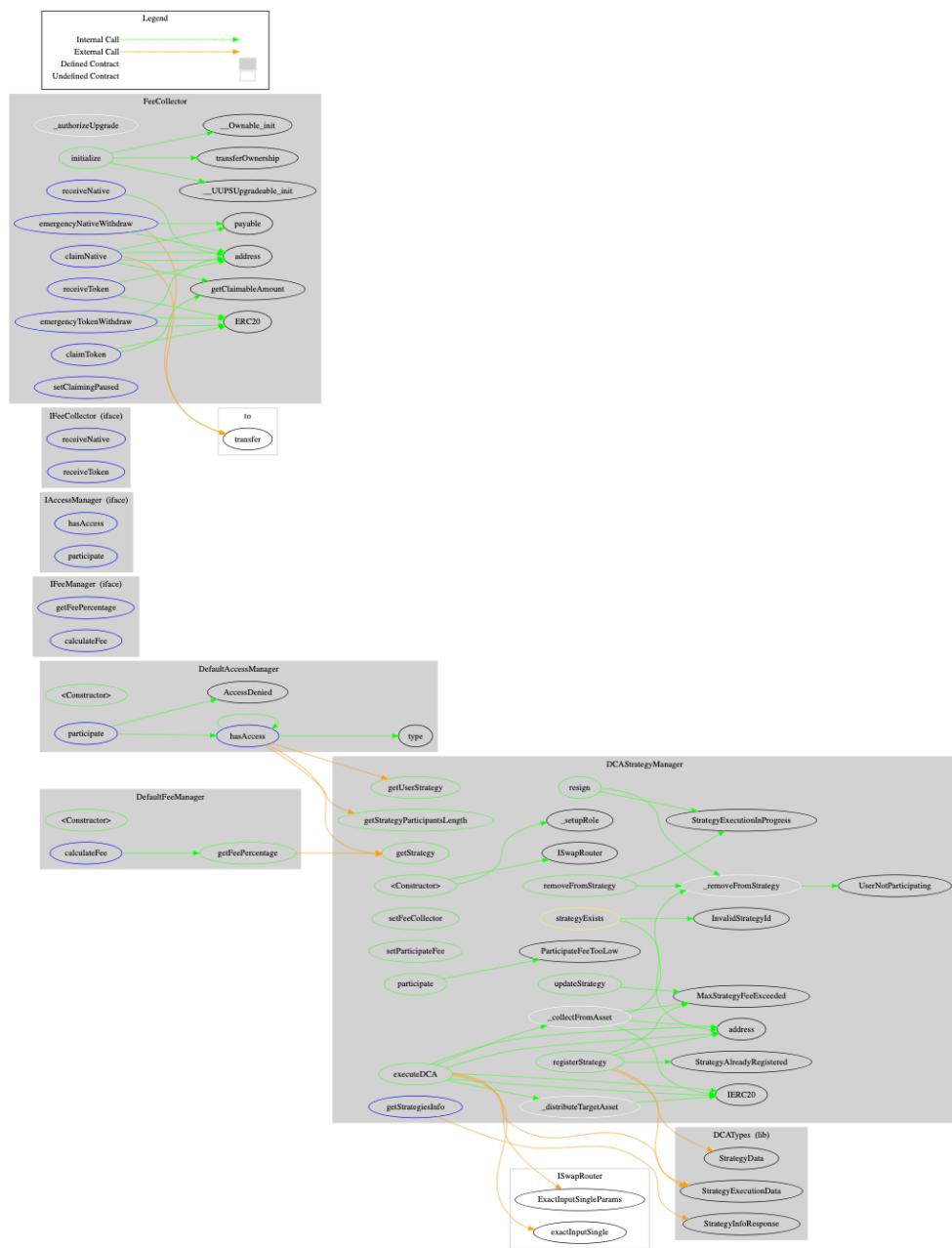
# Inheritance Graph

## 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. Is contract an upgradeable
2. Deployer cannot lock user funds
3. Deployer cannot pause the contract
4. Deployer cannot set fees
5. Deployer cannot blacklist/antisnipe addresses
6. Overall checkup (Smart Contract Security)

## Is contract an upgradeable

Name	
Is contract an upgradeable?	No

Comments:

v1.1

- The upgradeable contract was removed from the audit scope in version 1.1 by the AutoDCA team.

## Write functions of contract

### v1.1

- ◆ setFeeCollector
- ◆ setParticipateFee
- ◆ registerStrategy
- ◆ updateStrategy
- ◆ participate 💰
- ◆ resign
- ◆ removeFromStrategy
- ◆ executeDCA

## Deployer cannot burn or lock user funds

Name	Exist	Tested	Status
Deployer can lock	✓	✓	✓

## Deployer cannot pause the contract

Name	Exist	Tested	Status
Deployer can pause	-	-	-

## Deployer cannot set fees

Name	Exist	Tested	Status
Deployer cannot set fees over 25%	✓	✓	✓
Deployer cannot set fees to nearly 100% or to 100%	✓	✓	✓

Comments:

**v1.0**

- Cannot set strategy execution fee higher than 2.5%

## Deployer can blacklist/antisnipe addresses

Name	Exist	Tested	Status
Deployer cannot blacklist/antisnipe addresses	-	-	-

## Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

### Legend

Attribute	Symbol
Verified / Checked	✓
Partly Verified	🚩
Unverified / Not checked	✗
Not available	-

# Modifiers and public functions

v1.1

DCAStrategyManager

```
◆ setFeeCollector
Ⓜ️ onlyOwner
◆ setParticipateFee
Ⓜ️ onlyOwner
◆ registerStrategy
Ⓜ️ onlyOwner
◆ updateStrategy
Ⓜ️ onlyOwner
Ⓜ️ strategyExists
◆ participate 💰
Ⓜ️ strategyExists
◆ resign
◆ removeFromStrategy
Ⓜ️ onlyOwner
◆ executeDCA
Ⓜ️ onlyRole
```

## Comments

- AutoDCA:
  - Set fee collector address
  - Set participate one-time entry fee to any arbitrary value
  - Register Strategy
  - Update the strategy, and change uniswapFeeTier value, maxParticipants number, access manager, and fee manager addresses to any arbitrary value.
  - Remove accounts from strategy at any given time, and the participation fees paid by the user will not be refunded. It cannot be done while executing the DCA.
  - The account with the OPERATOR\_ROLE can execute the DCA
- 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

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



# Source Units in Scope

## v1.1

File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score
contracts/DefaultAccessManager.sol	1	—	70	59	48	2	20
contracts/IFeeManager.sol	—	1	17	7	3	3	5
contracts/IAccessManager.sol	—	1	16	6	3	2	5
contracts/DefaultFeeManager.sol	1	—	36	29	21	3	12
contracts/DCATypes.sol	1	—	65	65	57	14	1
contracts/DCAStrategyManager.sol	1	—	561	517	459	10	172
<b>Totals</b>	<b>4</b>	<b>2</b>	<b>765</b>	<b>683</b>	<b>591</b>	<b>34</b>	<b>215</b>

### 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

No low issues

## Informational issues

Issue	File	Type	Line	Description
#1	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.

## 09. March 2023:

- There is still an owner (Owner still has not renounced ownership)
- We recommend to put a hardcap on the participation fees
- Read whole report and modifiers section for more information

## SWC Attacks

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

<a href="#"><u>SW C-1 27</u></a>	Arbitrary Jump with Function Type Variable	<a href="#">CWE-695: Use of Low-Level Functionality</a>	PASSED
<a href="#"><u>SW C-1 25</u></a>	Incorrect Inheritance Order	<a href="#">CWE-696: Incorrect Behavior Order</a>	PASSED
<a href="#"><u>SW C-1 24</u></a>	Write to Arbitrary Storage Location	<a href="#">CWE-123: Write-what-where Condition</a>	PASSED
<a href="#"><u>SW C-1 23</u></a>	Requirement Violation	<a href="#">CWE-573: Improper Following of Specification by Caller</a>	PASSED
<a href="#"><u>SW C-1 22</u></a>	Lack of Proper Signature Verification	<a href="#">CWE-345: Insufficient Verification of Data Authenticity</a>	PASSED
<a href="#"><u>SW C-1 21</u></a>	Missing Protection against Signature Replay Attacks	<a href="#">CWE-347: Improper Verification of Cryptographic Signature</a>	PASSED
<a href="#"><u>SW C-1 20</u></a>	Weak Sources of Randomness from Chain Attributes	<a href="#">CWE-330: Use of Insufficiently Random Values</a>	PASSED
<a href="#"><u>SW C-11 9</u></a>	Shadowing State Variables	<a href="#">CWE-710: Improper Adherence to Coding Standards</a>	PASSED
<a href="#"><u>SW C-11 8</u></a>	Incorrect Constructor Name	<a href="#">CWE-665: Improper Initialization</a>	PASSED
<a href="#"><u>SW C-11 7</u></a>	Signature Malleability	<a href="#">CWE-347: Improper Verification of Cryptographic Signature</a>	PASSED

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

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



Solid  
Proofed

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

MADE IN GERMANY