



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

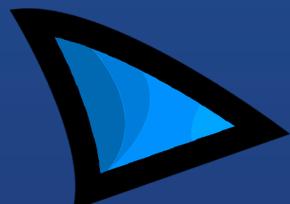
**Blockchain Security | Smart Contract Audits | KYC  
Development | Marketing**

MADE IN GERMANY

# Sharky Swap - Router Audit

**Security Assessment  
21. February, 2023**

For



# SharkySwap



[@SolidProof\\_io](https://twitter.com/SolidProof_io)



[@solidproof\\_io](https://t.me/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	18
Source Units in Scope	19
Critical issues	20
High issues	20
Medium issues	20
Low issues	20
Informational issues	20
Audit Comments	21
SWC Attacks	22

# 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	21. February 2023	<ul style="list-style-type: none"><li>• Layout project</li><li>• Automated- /Manual-Security Testing</li><li>• Summary</li></ul>

## **Network**

Arbitrum One

## **Website**

<https://sharkyswap.xyz/>

## **Twitter**

<https://twitter.com/SharkySwapFi>

## **Github**

<https://github.com/sharkyswap/>

## **Medium**

<https://sharkyswap.medium.com/>

## **Discord**

<https://discord.gg/sharkyswap>

## Description

TBA

## Project Engagement

During the 18th of February 2023, **SharkySwap 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://arbiscan.io/address/0x29631E68d81b0e02a507f54a799402E8Cfa6a4d8#code>

# 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:

Dependency / Import Path	Count
@uniswap/lib/contracts/libraries/TransferHelper.sol	1

## Tested Contract Files

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

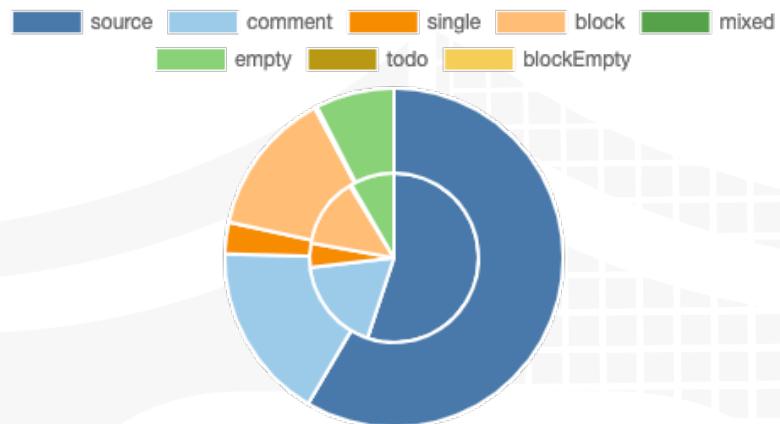
*A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.*

v1.0

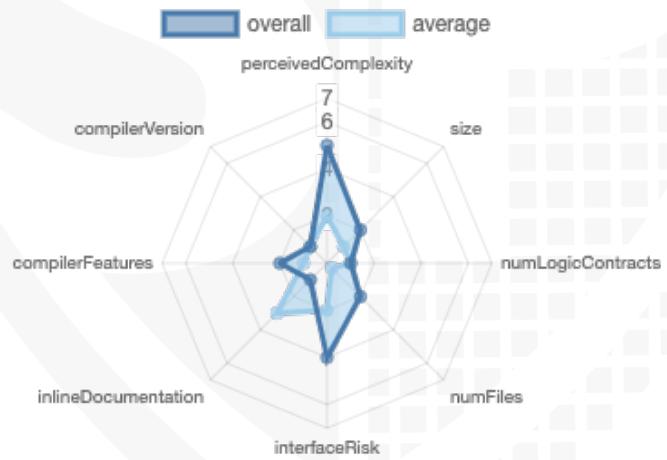
File Name	SHA-1 Hash
contracts/sources/interfaces/IWETH.sol	dab2b01305ed3451e26c1c3db23416c66c6f418d
contracts/sources/interfaces/ISharkyFactory.sol	02c54e4bbfbcc805ab80ca537414fd48628bb725
contracts/sources/interfaces/ISharkyRouter02.sol	aa22b83cfb89dc35e59b3ae047cc1f8459b04f8f
contracts/sources/interfaces/IERC20.sol	048e4e60de6bec1d7cb15896f40c17863e3a51f6
contracts/sources/SharkyRouter.sol	6e4017805fd7f7841ca386853d5926379ef6a52d
contracts/sources/libraries/SafeMath.sol	832880a37284c01f3e5ede51cbf2943ec8761d87
contracts/sources/libraries/SharkyLibrary.sol	c2cf431e0bd0e10550360ed5dde77e5080442dd6

# Metrics

## Source Lines v1.0



## Risk Level v1.0



# Capabilities

## Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	1	2	4	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	51	7

Version	External	Internal	Private	Pure	View
1.0	42	50	0	18	18

## State Variables

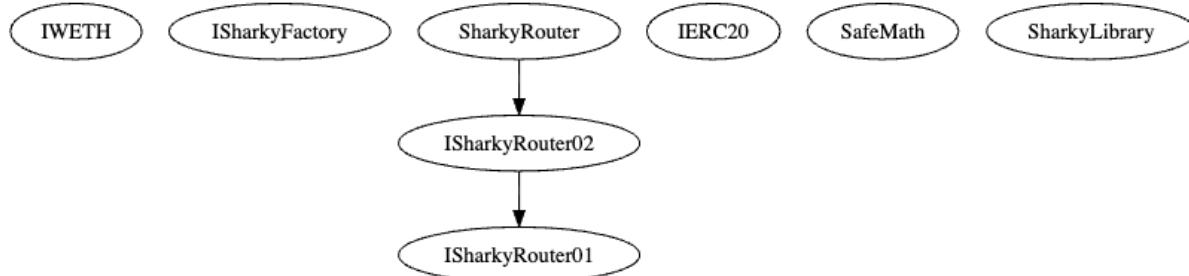
Version	Total	Public
1.0	2	2

## Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	>=0.5. 0 >=0.6. 2 >=0.4. 0 =0.6.6		yes		

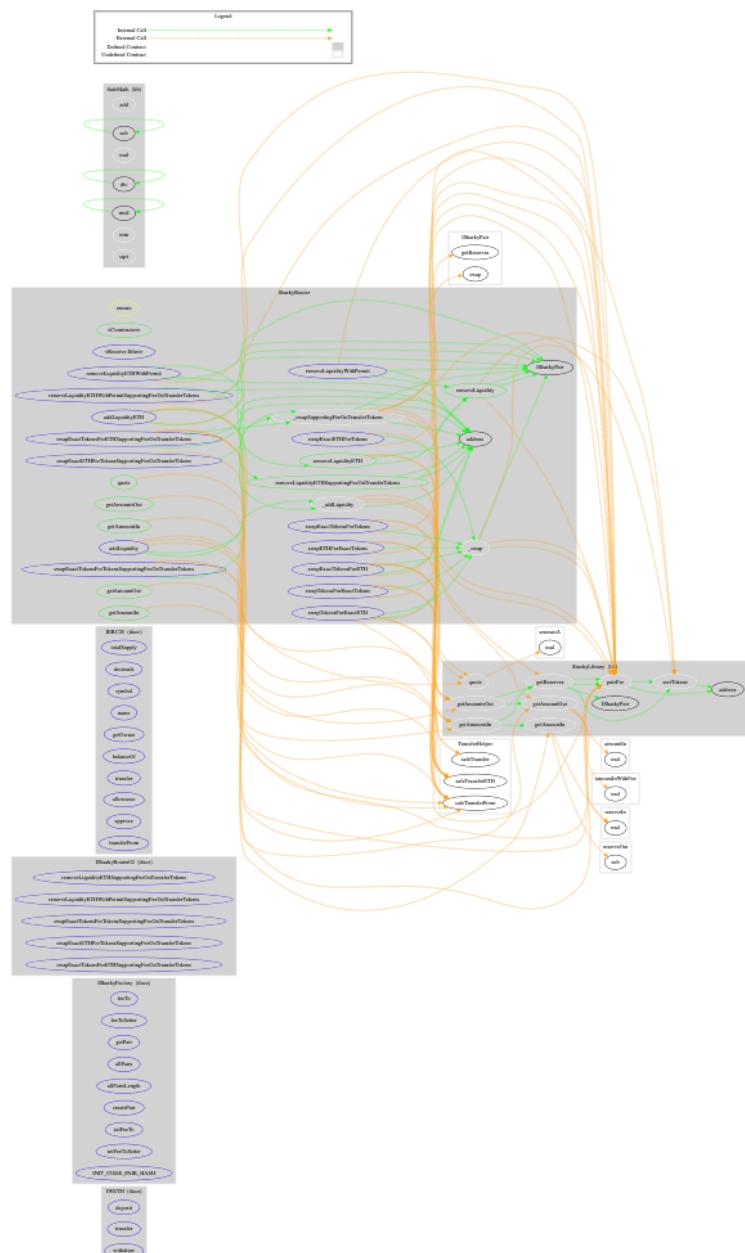
Version	Transfers ETH	Low-Level Calls	DelegateCall	Uses Hash Functions	EC Recover	New/Create/Create2
1.0	yes			yes		

## 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. Overall checkup (Smart Contract Security)

## Is contract an upgradeable

Name	
Is contract an upgradeable?	No



# Write functions of contract

## v1.0

1. addLiquidity
2. addLiquidityETH
3. removeLiquidity
4. removeLiquidityETH
5. removeLiquidityETHSupportingFeeOnTransferTokens
6. removeLiquidityETHWithPermit
7. removeLiquidityETHWithPermitSupportingFeeOnTransferTokens
8. removeLiquidityWithPermit
9. swapETHForExactTokens
10. swapExactETHForTokens
11. swapExactETHForTokensSupportingFeeOnTransferTokens
12. swapExactTokensForETH
13. swapExactTokensForETHSupportingFeeOnTransferTokens
14. swapExactTokensForTokens
15. swapExactTokensForTokensSupportingFeeOnTransferTokens
16. swapTokensForExactETH
17. swapTokensForExactTokens

## 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

```
✓ ⚡ addLiquidity
  ⚡ ensure
✓ ⚡ addLiquidityETH 💸
  ⚡ ensure
✓ ⚡ removeLiquidity
  ⚡ ensure
✓ ⚡ removeLiquidityETH
  ⚡ ensure
    ⚡ removeLiquidityWithPermit
    ⚡ removeLiquidityETHWithPermit
✓ ⚡ removeLiquidityETHSupportingFeeOnTransferTokens
  ⚡ ensure
    ⚡ removeLiquidityETHWithPermitSupportingFeeOnTransferTokens
✓ ⚡ swapExactTokensForTokens
  ⚡ ensure
✓ ⚡ swapTokensForExactTokens
  ⚡ ensure
✓ ⚡ swapExactETHForTokens 💸
  ⚡ ensure
✓ ⚡ swapTokensForExactETH
  ⚡ ensure
✓ ⚡ swapExactTokensForETH
  ⚡ ensure
✓ ⚡ swapETHForExactTokens 💸
  ⚡ ensure
✓ ⚡ swapExactTokensForTokensSupportingFeeOnTransferTokens
  ⚡ ensure
✓ ⚡ swapExactETHForTokensSupportingFeeOnTransferTokens 💸
  ⚡ ensure
✓ ⚡ swapExactTokensForETHSupportingFeeOnTransferTokens
  ⚡ ensure
```

## Comments

Note: these project files are a fork of the pancake swap router

**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/sources/interfaces/IWETH.sol	_____	1	10	5	3	1	10	.💰
🔍	contracts/sources/interfaces/ISharkyFactory.sol	_____	1	24	7	4	1	19	_____
🔍	contracts/sources/interfaces/ISharkyRouter02.sol	_____	1	51	7	4	1	16	.💰
🔍	contracts/sources/interfaces/IERC20.sol	_____	1	98	23	17	66	21	_____
📝	contracts/sources/SharkyRouter.sol	1	_____	499	320	274	16	310	.💰🌐
📚	contracts/sources/libraries/SafeMath.sol	1	_____	189	177	54	107	14	_____
📚	contracts/sources/libraries/SharkyLibrary.sol	1	_____	118	90	71	10	72	🌐
📝🔍📚	<b>Totals</b>	<b>3</b>	<b>4</b>	<b>989</b>	<b>629</b>	<b>427</b>	<b>202</b>	<b>462</b>	<b>.💰🌐</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

Issue	File	Type	Line	Description
#1	All libraries/ All interfaces	A floating pragma is set	2	The current pragma Solidity directive is „>=0.5.0”.
#2	SharkyR outer	Missing Zero Address Validation (missing-zero-check)	25	Check that the address is not zero

## Informational issues

**No informational issues**

## 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.

## 21. February 2023:

- 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>	PASSED
<a href="#"><u>SW C-1 04</u></a>	Unchecked Call Return Value	<a href="#"><u>CWE-252: Unchecked Return Value</u></a>	PASSED
<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>	NOT PASSED
<a href="#"><u>SW C-1 02</u></a>	Outdated Compiler Version	<a href="#"><u>CWE-937: Using Components with Known Vulnerabilities</u></a>	PASSED
<a href="#"><u>SW C-1 01</u></a>	Integer Overflow and Underflow	<a href="#"><u>CWE-682: Incorrect Calculation</u></a>	PASSED
<a href="#"><u>SW C-1 00</u></a>	Function Default Visibility	<a href="#"><u>CWE-710: Improper Adherence to Coding Standards</u></a>	PASSED



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

MADE IN GERMANY