



SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

Meetway

Audit

Security Assessment

06. May, 2022

For



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	20
Source Units in Scope	22
Critical issues	23
High issues	23
Medium issues	23
Low issues	23
Informational issues	23
Audit Comments	24
SWC Attacks	25

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	28. April 2022	<ul style="list-style-type: none">• Layout project• Automated- /Manual-Security Testing• Summary

Network

Binance Smart Chain (BEP20)

Website

<https://meetway.io/>

Telegram

<https://t.me/MEETWAYofficial>

Twitter

<https://twitter.com/meetwayofficial>

Discord

<https://discord.gg/meetway>

TikTok

<https://www.tiktok.com/@meetwayofficial>

Description

MeetWay is a mobile M2E app that allows people to connect, exercise, and earn.

MeetWay's ecosystem is a combination of music and the real world. This encourages users to engage more with the people and community around them as well as pushes them to build a healthy lifestyle.

Accordingly, MeetWay introduces the first Metaverse e-fitness, complete with unique features that haven't seen in any other health app. Running/cycling with teammates, weather notifications, terrain, achievement systems, and other features of the app will show its utility while also giving entertainment, encouraging users to practice every day. MeetWay is special in that it allows us to adopt healthy habits while still earning money.

Project Engagement

During the 25th of April 2022, **Meetway 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/meetway2022/meetway>
 - Commit: fcef61cc26b69f084d7a76e64b40a52884802f8b
- Bsc
 - <https://bscscan.com/address/0x1bB7a6024a4c4878cA870B58cC64c92b98b945Fc>

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 there were discovered.

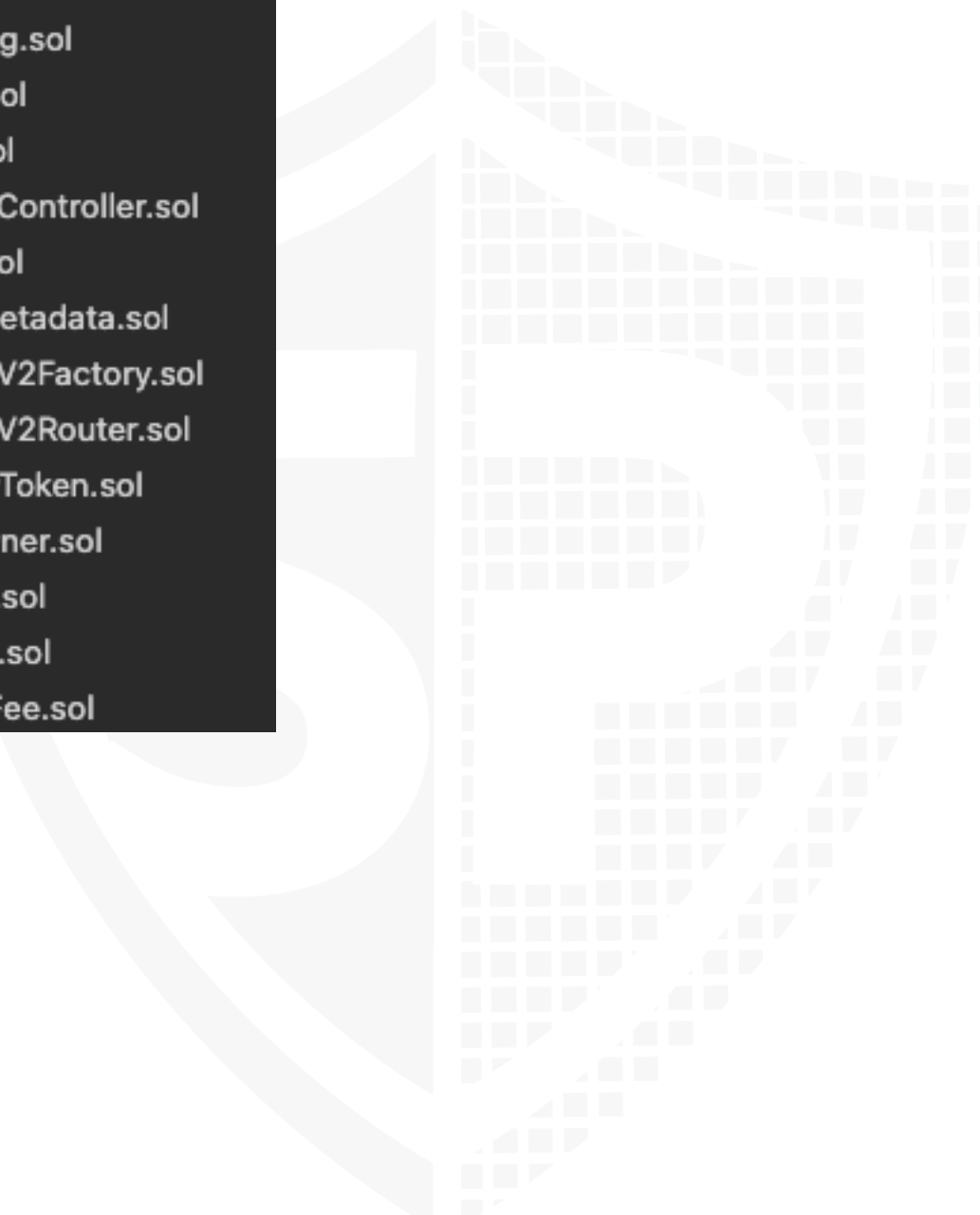
Methodology

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:



```
Context.sol
DexListing.sol
DexPair.sol
ERC20.sol
GasPriceController.sol
IERC20.sol
IERC20Metadata.sol
IUniswapV2Factory.sol
IUniswapV2Router.sol
MeetWayToken.sol
OriginOwner.sol
Ownable.sol
Pausable.sol
TransferFee.sol
```


Tested Contract Files

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

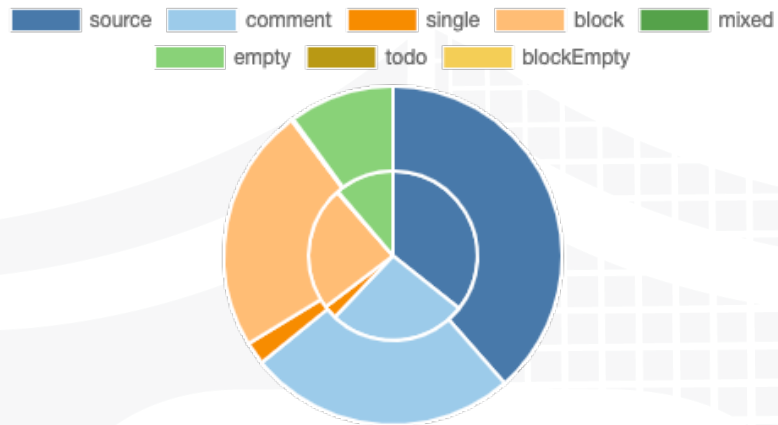
A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

v1.0

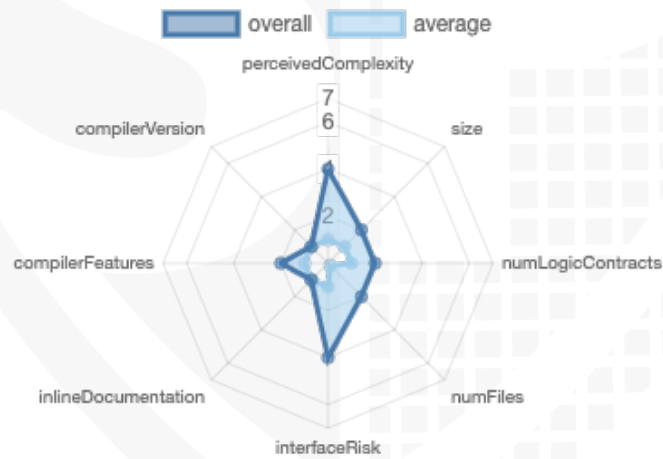
File Name	SHA-1 Hash
contracts/Context.sol	164272374e4a7a8499d9f75af6b33f37a75b61e8
contracts/IUniswapV2Factory.sol	fec5564db2888e0a6c0f495000ff6cbc7be2c732
contracts/DexListing.sol	89adc16ef4b1a9700d1ac776994131ce5ca2e7a3
contracts/IERC20Metadata.sol	d42a8bd1ccbe0bcf085972b58f6095da433370b4
contracts/TransferFee.sol	b3263360bc53b3d7d3cd4c984e83b75c151f3366
contracts/OriginOwner.sol	6a71a7567ae1d672a0e445113ca0e15c96322913
contracts/DexPair.sol	fc3e1cc9b9d44a59e4cd7291dd78766ecb922b6e
contracts/GasPriceController.sol	4bfa65c8d94be6b1849b724fd9100b41134d0709
contracts/IUniswapV2Router.sol	65717068028153627c9c07e27e84b8f4061e8172
contracts/Ownable.sol	7db188fad5e4ea3e0b1a6741c83f8380653d2801
contracts/Pausable.sol	d42ae199d2cfb083f59069b5de92fd36ae28e7c5
contracts/MeetWayToken.sol	a6bc5a8e2a896b0bb63265099cf9786cdec76e
contracts/ERC20.sol	0356a839afdce94958f278a28feb56cfe35fe173
contracts/IERC20.sol	6c31117e13085c9011db480856e8d7a99998000d

Metrics

Source Lines v1.0



Risk Level v1.0



Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	6	1	5	3

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	46	0

Version	External	Internal	Private	Pure	View
1.0	22	75	3	1	25

State Variables

Version	Total	Public
1.0	25	5

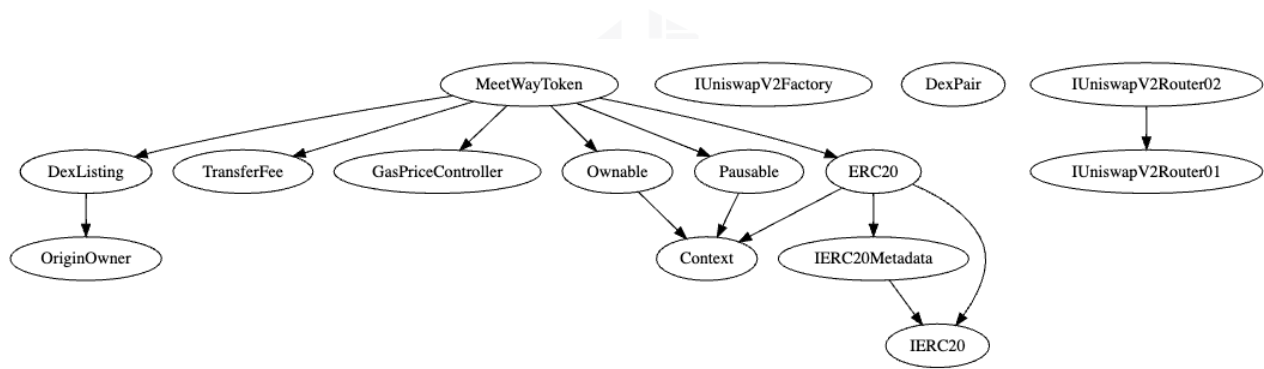
Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	<code>^0.8.4</code> <code>^0.8.0</code>			yes (1 asm blocks)	

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

1.0	yes			yes		
-----	-----	--	--	-----	--	--

Inheritance 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. Correct implementation of Token standard
2. Deployer cannot mint any new tokens
3. Deployer cannot burn or lock user funds
4. Deployer cannot pause the contract
5. Overall checkup (Smart Contract Security)

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	✓	✓	✓

Write functions of contract v1.0

addBlackList
addBlackLists
removeBlackList
removeBlackLists
setMaxAmount
setMaxGasPrice
setTransferFee
pause
unpause
withdrawBalance
withdrawTokens

renounceOwnership
transferOwnership

transfer
approve
transferFrom
increaseAllowance
decreaseAllowance

Deployer cannot mint any new tokens

Name	Exist	Tested	Status
Deployer cannot mint	✓	✓	✓
Max / Total Supply	200000		



Deployer cannot burn or lock user funds

Name	Exist	Tested	Status
Deployer cannot lock	✓	✓	✗
Deployer cannot burn	✓	✓	✓

Comments:

v1.0

- Owner can lock user funds by
 - blacklisting addresses

Deployer cannot pause the contract

Name	Exist	Tested	Status
Deployer cannot pause	✓	✓	✗

Comments:

v1.0

- Owner can pause contract



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

▼  addBlackList  onlyOwner	▼  renounceOwnership  onlyOwner
▼  addBlackLists  onlyOwner	▼  transferOwnership  onlyOwner
▼  removeBlackList  onlyOwner	
▼  removeBlackLists  onlyOwner	
▼  setMaxAmount  onlyOwner	
▼  setMaxGasPrice  onlyOwner	
▼  setTransferFee  onlyOwner	
▼  pause  onlyOwner	
▼  unpause  onlyOwner	
▼  withdrawBalance  onlyOwner	
▼  withdrawTokens  onlyOwner	

Comments

- Deployer can set following state variables without any limitations
 - `_transferFee.to`
 - `_transferFee.buy`
 - `_transferFee.sell`
 - `_transferFee.normal`
- Deployer can enable/disable following state variables
 - `blackListedList`
 - `_paused`

























- Existing Modifiers
 - onlyValidGasPrice
 - onlyOriginOwner
 - onlyOwner
 - whenNotPaused
 - whenPaused
- Max tx amount has no functionality

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/Context.sol	1	————	24	24	9	12	1	————
	contracts/IUniswapV2Factory.sol	————	1	9	6	3	1	3	————
	contracts/DexListing.sol	1	————	118	92	65	12	32	————
	contracts/IERC20Metadata.sol	————	1	28	17	4	16	9	
	contracts/TransferFee.sol	1	————	77	55	43	1	20	
	contracts/OriginOwner.sol	1	————	73	51	35	5	18	————
	contracts/DexPair.sol	1	————	42	31	17	5	21	
	contracts/GasPriceController.sol	1	————	37	29	21	1	8	————
	contracts/IUniswapV2Router.sol	————	2	10	7	4	1	6	————
	contracts/Ownable.sol	1	————	76	76	28	38	23	————
	contracts/Pausable.sol	1	————	91	91	29	51	16	————
	contracts/MeetWayToken.sol	1	————	168	144	103	17	104	
	contracts/ERC20.sol	1	————	356	336	103	194	80	
	contracts/IERC20.sol	————	1	82	27	17	58	13	
	Totals	10	5	1191	986	481	412	354	  

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	All	A floating pragma is set	At the top of source file	The current pragma Solidity directive is „^0.“.
#3	Main	Local variables shadowing	55, 142	Rename the local variables that shadow another component

Informational issues

Issue	File	Type	Line	Description
#1	DexPair	Unused state variables	18, 17	Remove unused state variables

#2	Transfer Fee	Unused state variables	9	Remove unused state variables
#3	Main	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/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.

28. April 2022:

- Read whole report 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	NOT 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

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

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

SW C-1 05	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	PASSED
SW C-1 04	Unchecked Call Return Value	CWE-252: Unchecked Return Value	PASSED
SW C-1 03	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	NOT PASSED
SW C-1 02	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	PASSED
SW C-1 01	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	PASSED
SW C-1 00	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED

The logo features the words "Solid Proofed" in a white, elegant script font. The word "Solid" is positioned above "Proofed". Behind the text is a faint, stylized shield emblem with a grid-like pattern, rendered in a darker shade of blue. The entire composition is set against a solid blue background.

Solid
Proofed

Blockchain Security | Smart Contract Audits | KYC

A small horizontal bar representing the German flag, with black, red, and gold stripes.

MADE IN GERMANY