



SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

Audit

Security Assessment
02. November, 2021

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
Scope of Work	13
Inheritance Graph	13
Verify Claims	14
OnlyRole functions	20
CallGraph	21
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	02. November 2021	<ul style="list-style-type: none">• Layout project• Automated- /Manual-Security Testing• Summary

Network

Binance Smart Chain (BEP20)

Website

<https://mahhealthcare.com/>

Twitter

<https://twitter.com/mahhealthcare>

Facebook

<https://www.facebook.com/MAH-Healthcare-Estonia-102940742147177>

Github

<https://github.com/HTRAX/htrax-token>

Instagram

<https://www.instagram.com/mah.healthcare/>

LinkedIn

<https://www.linkedin.com/company/mah-healthcare-estonia/>

Description

MAH Healthcare OU is Estonia Based rapidly growing Healthcare IT company with the Vision of Creating Brand Value in Short span of 2 years. MAH Want to Expand its service offerings across Healthcare IT industry sectors, number of customers, number of operating locations and capability for value creation. A strong leadership, investment in infrastructure and people, confidence of our People and our unwavering passion for excellence has significantly contributed to this momentum. MAH is constantly investing in state-of-the-art infrastructure, building people competence, adopting leading edge technology, and adding business value for its customer as well as in Community and everything else that is essential to achieve customer delight and employee jubilation. MAH Healthcare uses blockchain technology to securely manage health records for a collaborative, smart approach to healthcare

Project Engagement

During the 30th of October 2021, **HTRAX 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

TBA

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:

- HTRAXTokenSale:
 - AccessControl
- HTRAXToken:
 - Ownable
 - Pausable
 - AccessControl
 - ERC20
 - ERC20Snapshot



Tested Contract Files

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

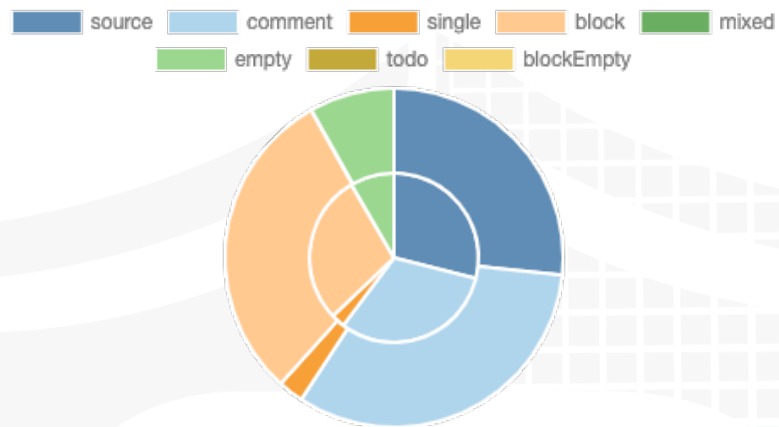
A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

v1.0

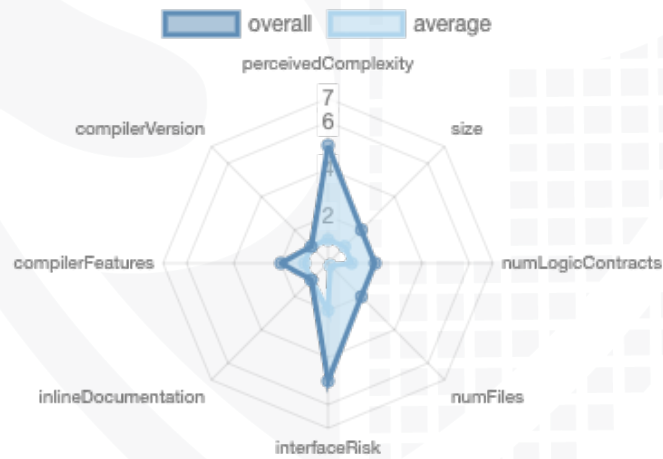
File Name	SHA-1 Hash
contracts/utills/Context.sol	5ece05a1a975617e5663847ad3a522966e3c5904
contracts/utills/Strings.sol	ccb4d394fcf0694b7dc1fed9ec2f0b292998b4be
contracts/utills/introspection/IERC165.sol	c274749941851ddff3c478cb475b65da616697c7
contracts/utills/introspection/ERC165.sol	160047f3e0c5221369ba3558f0a56ea2a7d7ca3f
contracts/utills/math/Math.sol	f04e52286764b68060e0619462f172c441b5bec4
contracts/utills/Arrays.sol	fcc6f29ca11d48ca9e2642ce8235c66604d53a3c
contracts/utills/Counters.sol	1152ad436016e04e2299d8cae70f68f3880f9de3
contracts/HTRAXToken.sol	791d710db982271e971ca1669b0ef176dfa403e1
contracts/HTRAXTokenSale.sol	8d56dbb841c00059bea0c79f3b17914893b4dafb
contracts/security/Pausable.sol	91d67f0ca43d851c1a43ec98ecf160ee14d0f504
contracts/access/Ownable.sol	b1b5d13cd2134a90082b2bad46542a323955d2eb
contracts/access/AccessControl.sol	a94eb77ca64f47afe96401336cf79174aba324ea
contracts/token/ERC20/ERC20.sol	3b01806757407473fe8110a75856f0c214ec1762
contracts/token/ERC20/IERC20.sol	d0dd54d5a489c52b30f941ac1d4a88aecbd0f2e4
contracts/token/ERC20/extensions/ERC20Snapshot.sol	993fbeb49e198ac31f45a7421a3c9f47b9f1e04
contracts/token/ERC20/extensions/IERC20Metadata.sol	6e1b1b40d095a0a5611d6500052b87ac8f04b83f

Metrics

Source Lines v1.0



Risk Level v1.0



Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	2	4	4	7

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	67	0

Version	External	Internal	Private	Pure	View
1.0	16	85	7	6	41

State Variables

Version	Total	Public
1.0	30	6

Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	0.8.9 ^0.8.9			**** (0 asm blocks)	

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

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



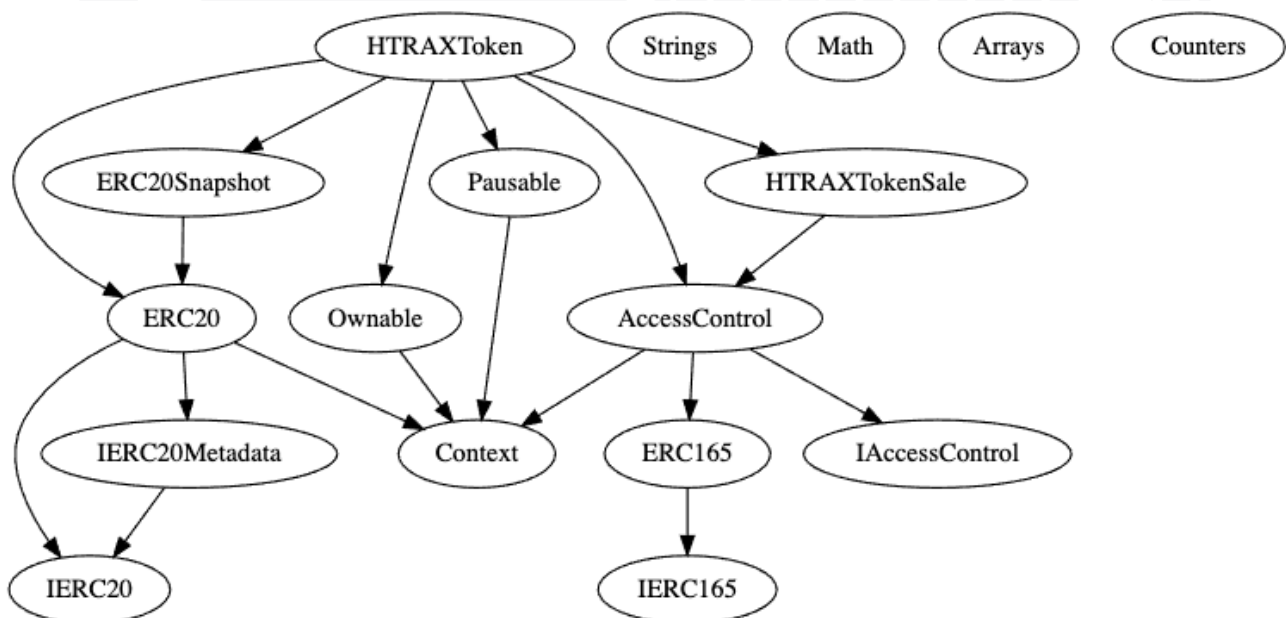
Scope of Work

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)

Inheritance Graph v1.0



Verify Claims


Correct implementation of Token standard

Tested	Verified
✓	✓

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

HTRAXToken:



addBlackList
approve
burn
burnFrom
decreaseAllow...
grantRole
increaseAllow...
mint
pause
releaseLocked...
removeBlackList
renounceRole
revokeRole
setLevel1Value
setLevel2Value
setLevel3Value
setSaleEndDate
setSaleStartD...
snapshot
transfer
transferDiscou...
transferFrom
transferOwner...
unpause
withdrawBala...

Deployer cannot mint any new tokens

Name	Exist	Tested	Verified	File
Deployer cannot mint	✓	✓	✗	Main
Comment	Line: -			

Max / Total Supply (Release): 93.750.000

Comments:

v1.0

- Addresses with MINTER_ROLE can mint tokens
 - Maximum _release tokens
 - totalSupply + totalBurned tokens + minting amount cannot be higher than _cap

```
function mint(address account↑, uint256 amount↑) public onlyRole(MINTER_ROLE) {  
    require(amount↑ <= _release, "Token for mint can't be more then allocated for each release");  
    require((totalSupply() + totalBurned() + amount↑) <= _cap, "cap exceeded");  
    super._mint(account↑, amount↑);  
}
```

- Addresses are not allowed to use mint function when pause is enabled

Deployer cannot burn or lock user funds

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

Comments:

v1.0

- Addresses with BURNER_ROLE can burn tokens
- Deployer with RISK_MANAGER_ROLE can
 - Set address to blackListed and can lock user funds
 - Lock user funds when pause is enabled
 - Can set pause/unpause
 - Add/remove blacklist addresses
- Addresses are not allowed to use burn function when pause is enabled
- Deployer with EXECUTOR_ROLE can
 - Transfer discounted/locked tokens
 - Withdraw token from smart contract to own address

Deployer cannot pause the contract

Name	Exist	Tested	Verified
Deployer cannot pause	✓	✓	✗

Comments:

v1.0

- See comments above in “Deployer cannot burn or lock user funds” section



Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

Legend

Attribute	Symbol
Verified / Checked	✓
Partly Verified	⚠
Unverified / Not checked	✗
Not available	—

OnlyRole functions

HTRAXToken

- MINTER_ROLE
 - mint
- BURNER_ROLE
 - burn
 - burnFrom
- RISK_MANAGER_ROLE
 - snapshot
 - pause
 - unpause
 - addBlackList
 - removeBlackList
- EXECUTOR_ROLE
 - transferDiscountedTokens
 - withdrawBalance

HTRAXTokenSale




























- PRESALES_MANAGER_ROLE
 - setSaleStartDate
 - setSaleEndDate
 - setLevel1Value
 - setLevel2Value
 - setLevel3Value

AdminRole functions

- grantRole
- revokeRole

Source Units in Scope

v1.0

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
	contracts/utlis/Context.sol	1	_____	27	27	10	13	1	_____
	contracts/utlis/Strings.sol	1	_____	67	67	45	15	41	
	contracts/utlis/introspection/IERC165.sol	_____	1	24	23	3	18	3	
	contracts/utlis/introspection/ERC165.sol	1	_____	28	28	7	18	6	_____
	contracts/utlis/math/Math.sol	1	_____	31	31	12	15	3	
	contracts/utlis/Arrays.sol	1	_____	47	47	24	16	6	
	contracts/utlis/Counters.sol	1	_____	38	38	21	13	2	
	contracts/HTRAXToken.sol	1	_____	185	181	96	59	139	
	contracts/HTRAXTokenSale.sol	1	_____	192	192	90	77	75	
	contracts/security/Pausable.sol	1	_____	90	90	29	50	16	_____
	contracts/access/Ownable.sol	1	_____	56	56	23	26	18	_____
	contracts/access/AccessControl.sol	1	1	243	231	102	146	65	
	contracts/token/ERC20/ERC20.sol	1	_____	364	344	108	196	83	_____
	contracts/token/ERC20/IERC20.sol	_____	1	86	26	17	60	15	_____
	contracts/token/ERC20/extensions/ERC20Snapshot.sol	1	_____	179	177	73	79	45	_____
	contracts/token/ERC20/extensions/IERC20Metadata.sol	_____	1	27	16	4	15	9	
	Totals	13	4	1684	1574	664	816	527	

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 found -

High issues

- no high issues found -

Medium issues

- no medium issues found -

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

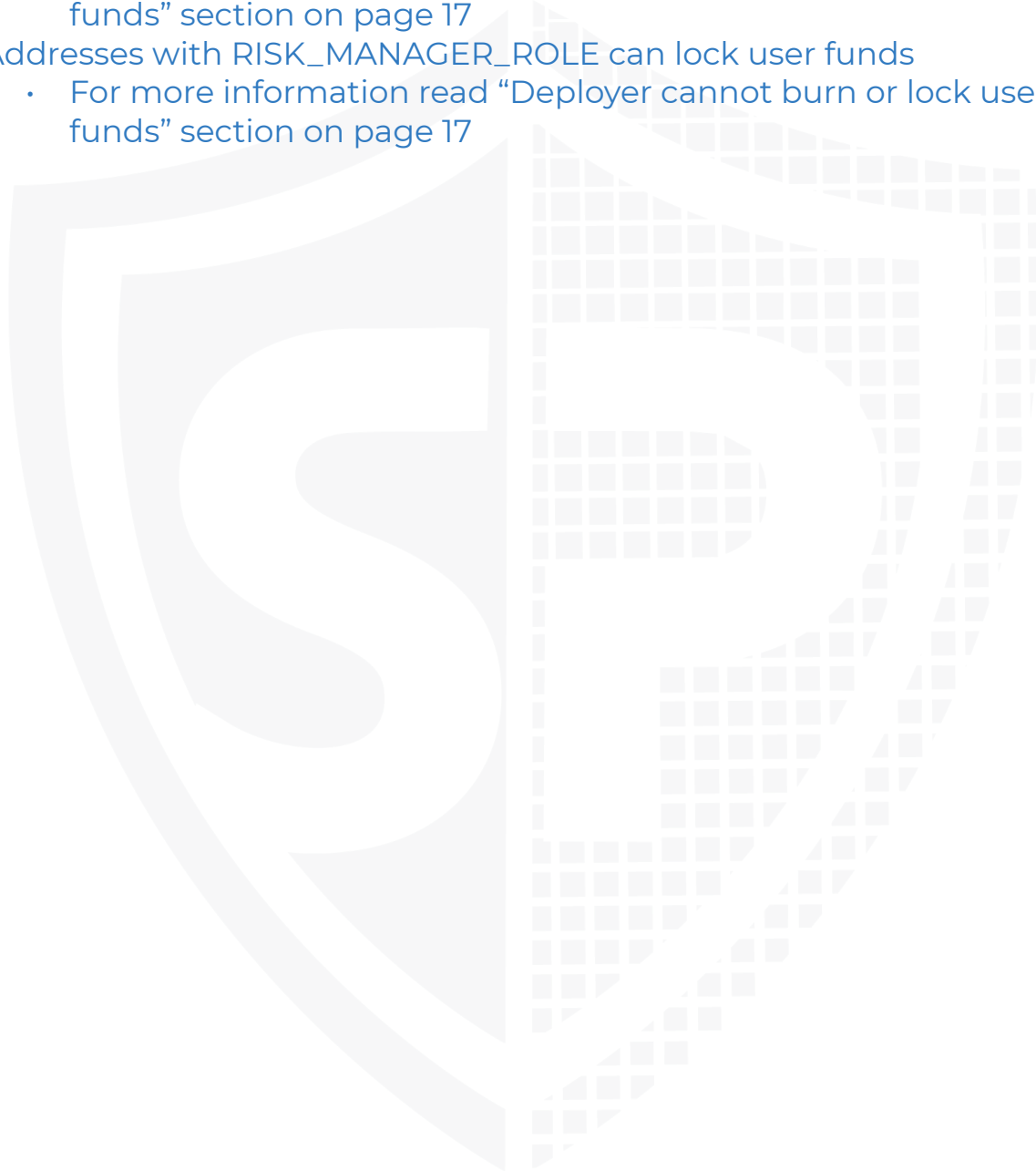
Informational issues

Issue	File	Type	Line	Description
#1	HTRAXT oken	State variables that could be declared constant (constable-states)	21, 20	Add the `constant` attributes to state variables that never change

Audit Comments

02. November 2021:

- Addresses with MINTER_ROLE can mint tokens
 - For more information read “Deployer cannot mint any new tokens” section on page 16
- Addresses with BURNER_ROLE can burn tokens
 - For more information read “Deployer cannot burn or lock user funds” section on page 17
- Addresses with RISK_MANAGER_ROLE can lock user funds
 - For more information read “Deployer cannot burn or lock user funds” section on page 17



SWC Attacks

ID	Title	Relationships	Status
SW C-13 6	Unencrypted Private Data On-Chain	CWE-767: Access to Critical Private Variable via Public Method	PASSED
SW C-13 5	Code With No Effects	CWE-1164: Irrelevant Code	PASSED
SW C-13 4	Message call with hardcoded gas amount	CWE-655: Improper Initialization	PASSED
SW C-13 3	Hash Collisions With Multiple Variable Length Arguments	CWE-294: Authentication Bypass by Capture-replay	PASSED
SW C-13 2	Unexpected Ether balance	CWE-667: Improper Locking	PASSED
SW C-13 1	Presence of unused variables	CWE-1164: Irrelevant Code	PASSED
SW C-13 0	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	PASSED
SW C-12 9	Typographical Error	CWE-480: Use of Incorrect Operator	PASSED
SW C-12 8	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	PASSED

SW C-12 7	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	PASSED
SW C-12 5	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	PASSED
SW C-12 4	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	PASSED
SW C-12 3	Requirement Violation	CWE-573: Improper Following of Specification by Caller	PASSED
SW C-12 2	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	PASSED
SW C-12 1	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	PASSED
SW C-12 0	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	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-111	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-10 9	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	PASSED
SW C-10 8	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED
SW C-10 7	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	PASSED
SW C-10 6	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	PASSED

SW C-10 5	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	PASSED
SW C-10 4	Unchecked Call Return Value	CWE-252: Unchecked Return Value	PASSED
SW C-10 3	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	PASSED
SW C-10 2	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	PASSED
SW C-10 1	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	PASSED
SW C-10 0	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED

The logo features the word "SolidProofed" in a white, elegant script font. The text is superimposed on a dark blue background that contains a faint, stylized shield emblem. The shield has a grid-like pattern and a subtle 3D effect.

SolidProofed

Blockchain Security | Smart Contract Audits | KYC

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

MADE IN GERMANY