



SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

Audit

Security Assessment
15. December, 2021

For



Chartered
Investment

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
Modifiers	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
Test Protocol	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	16. December 2021	<ul style="list-style-type: none">• Layout project• Automated- /Manual-Security Testing• Summary

Network

Binance Smart Chain (BEP20)

Website

<https://chartered-investment.com/>



Description

TBA

Project Engagement

During the 15th of December 2021, **CharteredInvestment 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



Chartered Investment

Contract Link

v1.0

- Gitlab
 - <https://gitlab.com/tokenforge-gmbh/launchpad/token-forge/e-sec-contracts/-/tree/main>
 - Commit: d7f5aefc211b918d0d4865992d861b90c491fb8b

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:

Dependency / Import Path	Count
@openzeppelin/contracts/access/AccessControlEnumerable.sol	4
@openzeppelin/contracts/token/ERC20/ERC20.sol	1
@openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.sol	1
@openzeppelin/contracts/utils/Context.sol	4
@openzeppelin/contracts/utils/Counters.sol	3



Tested Contract Files

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

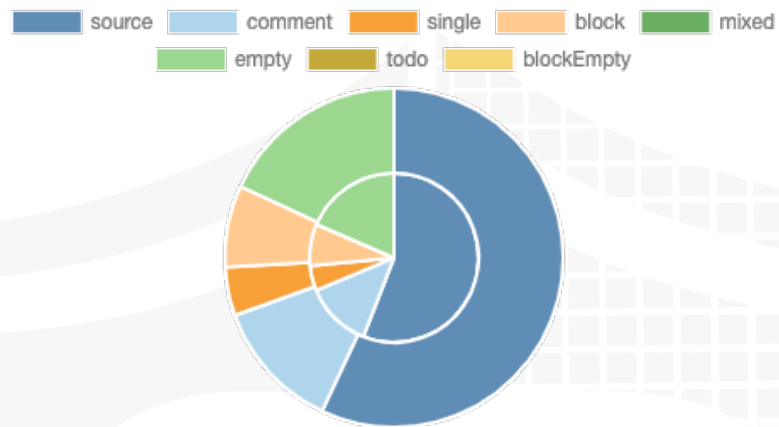
A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

v1.0

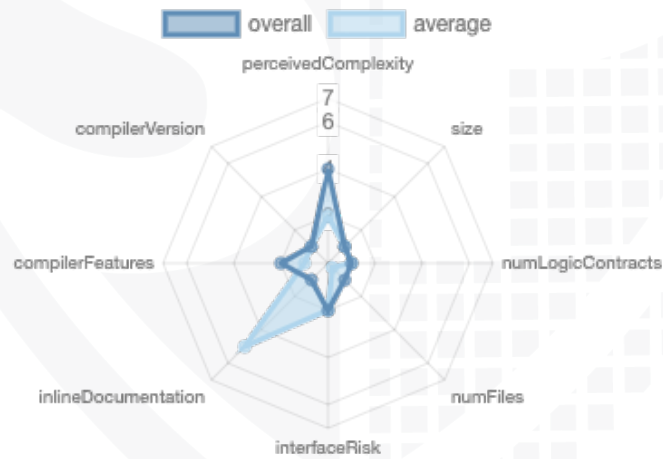
File Name	SHA-1 Hash
contracts/OPUS.sol	d5759c26e8fdc020e00f15b594e78642e565d3ce
contracts/Blacklist.sol	70a5856a2fa63d11fe5f24d2ee4522c42c52b169
contracts/Whitelist.sol	346baffac6e27b160752629b3ff23e81818d47a6
contracts/ESECF1.sol	67cdfc8d2b3957e4909ef356f485682920d676b7

Metrics

Source Lines v1.0



Risk Level v1.0



Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	4	0	0	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	19	0

Version	External	Internal	Private	Pure	View
1.0	2	18	0	0	7

State Variables

Version	Total	Public
1.0	16	14

Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	>=0.8.0 <0.9.0				

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

1.0				yes		yes → New Contract:0 PUS
-----	--	--	--	-----	--	-----------------------------------



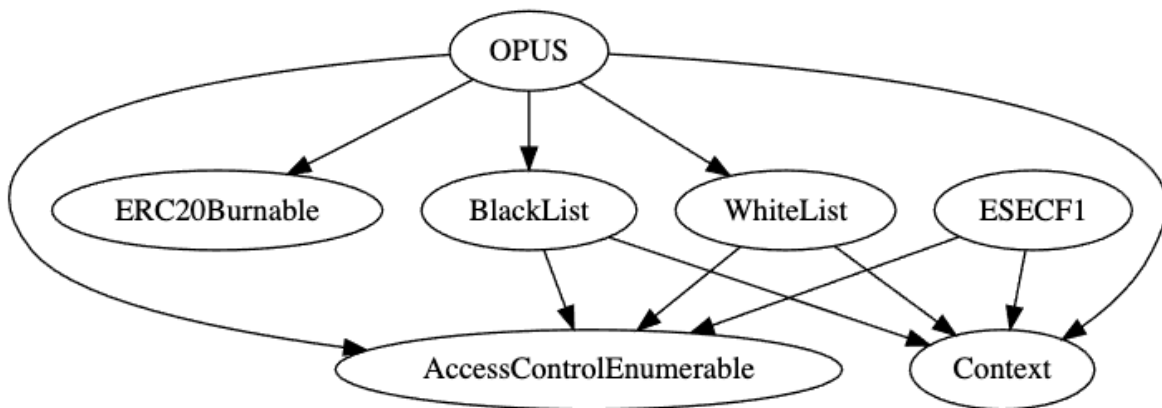
Scope of Work

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

▼ ESECF1	▼ OPUS
create	addToBlackList
grantRole	addToWhiteList
renounceRole	approve
revokeRole	burn
	burnFrom
	decreaseAllowance
	grantBlackListerRole
	grantRole
	grantWhiteListerRole
	increaseAllowance
	mint
	removeFromBlackList
	removeFromWhiteList
	renounceRole
	revokeBlackListerRole
	revokeRole
	revokeWhiteListerRole
	setParameters
	setProperty
	transfer
	transferFrom

Deployer cannot mint any new tokens

Name	Exist	Tested	Verified
Deployer cannot mint	✓	✓	✗

Comments:

v1.0

- MINTER_ROLE can mint only



Deployer cannot burn or lock user funds

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

Comments:

v1.0

- Deployer can lock with blacklisting of addresses and if the address to which is sent is not whitelisted

Deployer cannot pause the contract

Name	Exist	Tested	Verified
Deployer cannot pause	—	—	—



Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

Legend

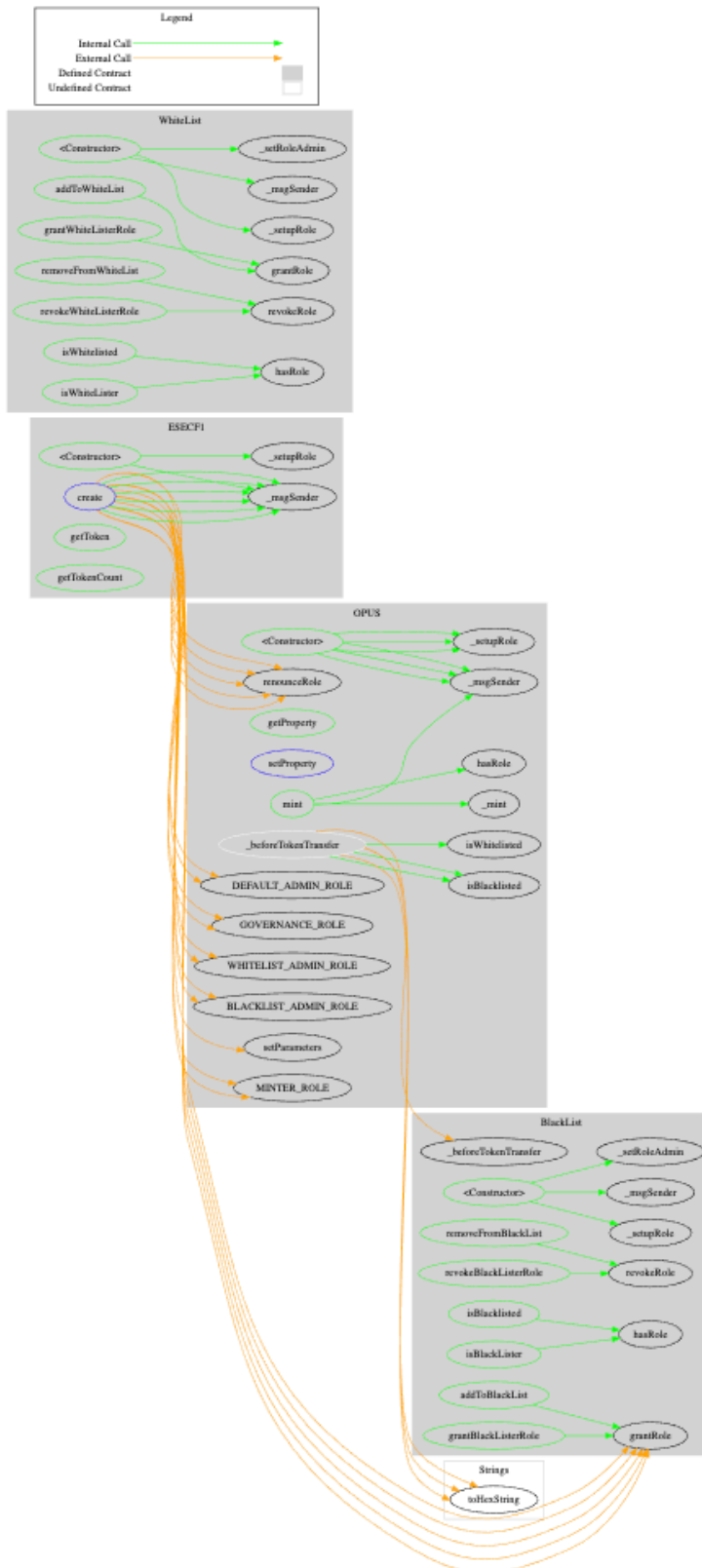
Attribute	Symbol
Verified / Checked	✓
Partly Verified	⚠
Unverified / Not checked	✗
Not available	—

Modifiers

- ESECF1
 - onlyRole -> DEFAULT_ADMIN_ROLE
 - create
- OPUS
 - onlyRole -> GOVERNANCE_ROLE
 - setParamaters
 - setProperty
 - onlyRole -> MINTER_ROLE
 - Mint












CallGraph



Source Units in Scope

v1.0

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
	contracts/OPUS.sol	1	————	126	122	82	19	95	
	contracts/Blacklist.sol	1	————	43	43	30	3	32	
	contracts/Whitelist.sol	1	————	43	43	30	3	32	
	contracts/ESECF1.sol	1	————	69	65	33	15	62	
	Totals	4	————	281	273	175	40	221	 

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	Blacklist	A floating pragma is set	5	The current pragma Solidity directive is „>=0.8.0 <0.9.0“.
#2	Whitelist	A floating pragma is set	5	The current pragma Solidity directive is „>=0.8.0 <0.9.0“.
#3	ESECF1	A floating pragma is set	5	The current pragma Solidity directive is „>=0.8.0 <0.9.0“.
#4	OPUS	A floating pragma is set	5	The current pragma Solidity directive is „>=0.8.0 <0.9.0“.

Informational issues

Issue	File	Type	Line	Description
#1	OPUS	Describe functions	56, 69, 73, 99	If you start to describe you functions describe the other functions also
#2	Blacklist	Public function that could be declared external	20, 24, 32, 36, 40	Use the `external` attribute for functions never called from the contract

#3	Whitelis t	Public function that could be declared external	20, 24, 35, 36, 40	Use the `external` attribute for functions never called from the contract
#4	Blacklist	No match with filename and contract name	11	Change filename of Blacklist.sol file to BlackList.sol or change contract name to Blacklist (and every function name and parameters where BlackList is used)
#5	Whitelis t	No match with filename and contract name	11	Change filename of Whitelist.sol file to WhiteList.sol or change contract name to WhiteList (and every function name and parameters where WhiteList is used)

Audit Comments

16. December 2021:

- Deployer can lock with blacklisting of addresses and if the address to which is sent is not whitelisted
- Read whole report for more information

Test Protocol

OPUS

we can mint tokens

- ✓ should mint tokens to Axel successfully

we can mint even more tokens

- ✓ should mint tokens to Ben successfully as well

e-Sec-Properties works as expected

...with ISIN

- ✓ can read ISIN properly
- ✓ can change ISIN properly
- ✓ it will revert if setISIN won't make any changes
- ✓ it will revert if non-governance role will try to change ISIN

...with Issuer Name

- ✓ can read ISSUER_NAME properly
- ✓ can change ISSUER_NAME properly
- ✓ it will revert if setIssuerName won't make any changes
- ✓ it will revert if non-governance role will try to change ISIN

token transfer should work

- ✓ mint and transfer should work properly

OPUS

blacklisting works as expected

- ✓ Axel is not blacklisted at the moment
- ✓ Null-Address is not blacklisted without further actions

Blacklisting Axel

- ✓ Axel is blacklisted successfully
- ✓ Since Axel is blacklisted, he will not be allowed to receive tokens from Ben
- ✓ After de-blacklisting Axel, he will be allowed to receive tokens from Ben again
- ✓ Blacklisting of Zero address is allowed

Permissions on blacklisting work properly

- ✓ BlackLister Roles are known and setup correctly
- ✓ Axel can't grant BlackLister Role to himself
- ✓ Ben can't revoke BlackLister Role from Governance
- ✓ Axel won't be able to blacklist himself without further permissions
- ✓ Axel won't be able to blacklist Ben without further permissions
- ✓ Axel won't be able to blacklist Chantal without further permissions
- ✓ Ben won't be able to blacklist Axel without Axel WILL be able to blacklist himself

further permissions

Axel will become Blacklister-Role

- ✓ Axel will have Role Blacklister
- ✓ Axel will be BlackLister
- ✓ Axel WILL be able to blacklist himself
- ✓ Axel WILL be able to blacklist Chantal
- ✓ Ben won't be able to blacklist Axel without further permissions

And we revoke Axel as blacklister

- ✓ Axel won't be able to blacklist himself again
- ✓ Axel won't be able to blacklist Ben again
- ✓ Axel won't be able to blacklist Chantal again

TokenFactory

- ✓ Admin Role via Chantal was setup correctly
- we can create Token per factory
- ✓ Roles for created token contract are correctly set up
- ✓ [factory] we can list this token contract
- ✓ [factory] roles are set correctly
- we can mint tokens on factory-created token contract
- ✓ [factory] should mint tokens to Axel successfully

TokenMinting

- ✓ minting for non-minter accounts should not be allowed
- minter-role can mint tokens
- ✓ should mint tokens to Axel successfully
- ✓ minting for Minter2 account should not be allowed
- ✓ adding another minter should work and let them mint tokens finally

OPUS

whitelisting works as expected

- ✓ Axel is not whitelisted at the moment
- ✓ Null-Address is not whitelisted without further actions
- ✓ Axel is not allowed to receive tokens due to missing whitelisting

Whitelisting Axel

- ✓ Axel was whitelisted successfully
- ✓ After whitelisting Axel, he will be allowed to receive tokens from Ben
- ✓ After de-whitelisting Axel, he will NOT be allowed ANY MORE to receive tokens

from Ben

- ✓ Whitelisting of Zero address is allowed

Permissions on whitelisting work properly

- ✓ WhiteLister Roles are known and setup correctly
- ✓ Axel can't grant WhiteLister Role to himself
- ✓ Ben can't revoke WhiteLister Role from Governance
- ✓ Axel won't be able to whitelist himself without further permissions
- ✓ Axel won't be able to whitelist Ben without further permissions
- ✓ Axel won't be able to whitelist Chantal without further permissions
- ✓ Ben won't be able to whitelist Axel without further permissions

Axel will become Whitelister-Role

- ✓ Axel will have Role WHITELISTER
- ✓ Axel will be WhiteLister
- ✓ Axel WILL be able to whitelist himself
- ✓ Axel WILL be able to whitelist Chantal
- ✓ Ben won't be able to whitelist Axel without further permissions

And we revoke Axel as whitelister

- ✓ Axel won't be able to whitelist himself again
- ✓ Axel won't be able to whitelist Ben again
- ✓ Axel won't be able to whitelist Chantal again

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	NOT 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 "SolidProof" in a white, elegant script font. The "P" is particularly large and stylized, with a long horizontal stroke that extends to the left. The background is a solid blue color with a faint, large shield emblem. The shield has a grid-like pattern on its right side and a solid blue area on its left side.

SolidProof

Blockchain Security | Smart Contract Audits | KYC

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

MADE IN GERMANY