



**SOLIDProof**  
*Bring trust into your projects*

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

MADE IN GERMANY

# Floki Vader

# Audit

**Security Assessment**  
**05. February, 2022**

**For**



**FLOKI VADER**  
**ELON....I AM YOUR DOG!**

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 Uniswap, 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	05. February 2022	<ul style="list-style-type: none"><li>• Layout project</li><li>• Automated- /Manual-Security Testing</li><li>• Summary</li></ul>

## **Network**

Binance Smart Chain (BEP20)

## **Website**

<http://flokivader.com/>

## **Telegram**

<https://t.me/FlokiVaderOfficialChatGroup>

## **Twitter**

<https://twitter.com/FlokiVader>

## **Instagram**

[https://www.instagram.com/flokivader\\_official/](https://www.instagram.com/flokivader_official/)

## **Youtube**

[https://www.youtube.com/channel/UCdJvD\\_fxgU9FJ7NN67xpsnQ](https://www.youtube.com/channel/UCdJvD_fxgU9FJ7NN67xpsnQ)

## Description

A token from a gala far far away. BNB auto-claim rewards and auto buyback hyper-deflationary token

## Project Engagement

During the 2nd of February 2022, **Floki Vader 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



**FLOKI VADER**  
**ELON....I AM YOUR DOG!**

## Contract Link

**v1.0**

- Team floki vader provided contracts as files

# 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 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
DividendPayingToken.sol
DividendPayingTokenInterface.sol
DividendPayingTokenOptionalInterface.sol
ERC20.sol
IDex.sol
IERC20.sol
IERC20Metadata.sol
IterableMapping.sol
Ownable.sol
SafeMath.sol
```



## Tested Contract Files

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

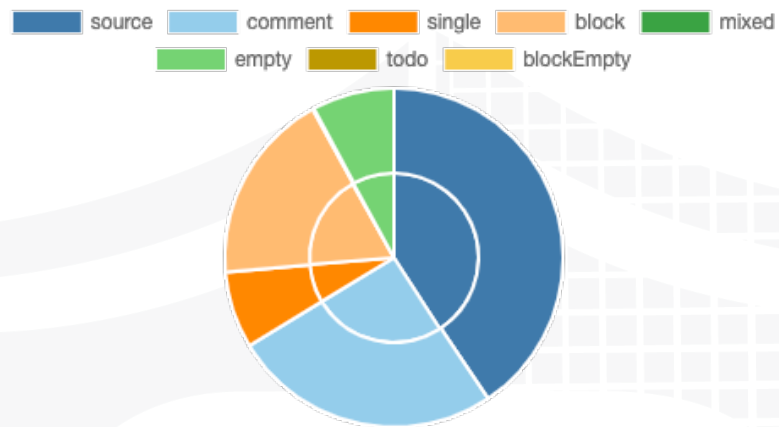
*A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.*

### v1.0

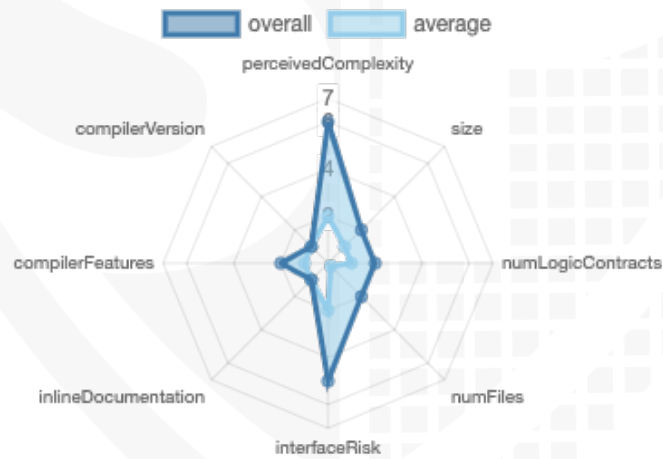
File Name	SHA-1 Hash
contracts/DividendPayingToken.sol	f9591874b30ae023267bc6c0c2aca081fea2846a
contracts/Context.sol	6a0b5b8e1b849d1ea73eabcfb1c9cd7e0cdb91b
contracts/DividendPayingTokenInterface.sol	554957118fdd968940ee2b6d42eba2a925f4d2b4
contracts/IERC20Metadata.sol	527f21f2724a24e1397572f80aeb688a61269b29
contracts/SafeMath.sol	175f5856180ba5e11815eb657cea72ee7feb0bf6
contracts/Ownable.sol	0b7fc2f6d8630101409f36f74b6ef3f218172e81
contracts/IterableMapping.sol	5af8133787de8bab0e79e83dc3e7b5ee1d815903
contracts/flokivader.sol	4ae08d8600326550b1aa8a70b36e78ab412ca91f
contracts/ERC20.sol	4e06e95e74958d71e0b44d2605537946a567a954
contracts/DividendPayingTokenOptionalInterface.sol	92dc321b7dbc744087a398cbf3c2082bb5ecdf97
contracts/IERC20.sol	9d3f2fce348740f0d4f7e5b4528b65aa29f60f94
contracts/IDex.sol	807a9a0714588b477b143cbc3fc78ca794ec1cb7

# Metrics

## Source Lines v1.0



## Risk Level v1.0



## Capabilities

### Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	5	5	7	1

### 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	93	6

Version	External	Internal	Private	Pure	View
1.0	56	99	5	18	42

### State Variables

Version	Total	Public
1.0	37	22

### Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	<code>^0.8.6</code> <code>^0.8.0</code>		<code>yes</code>		

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

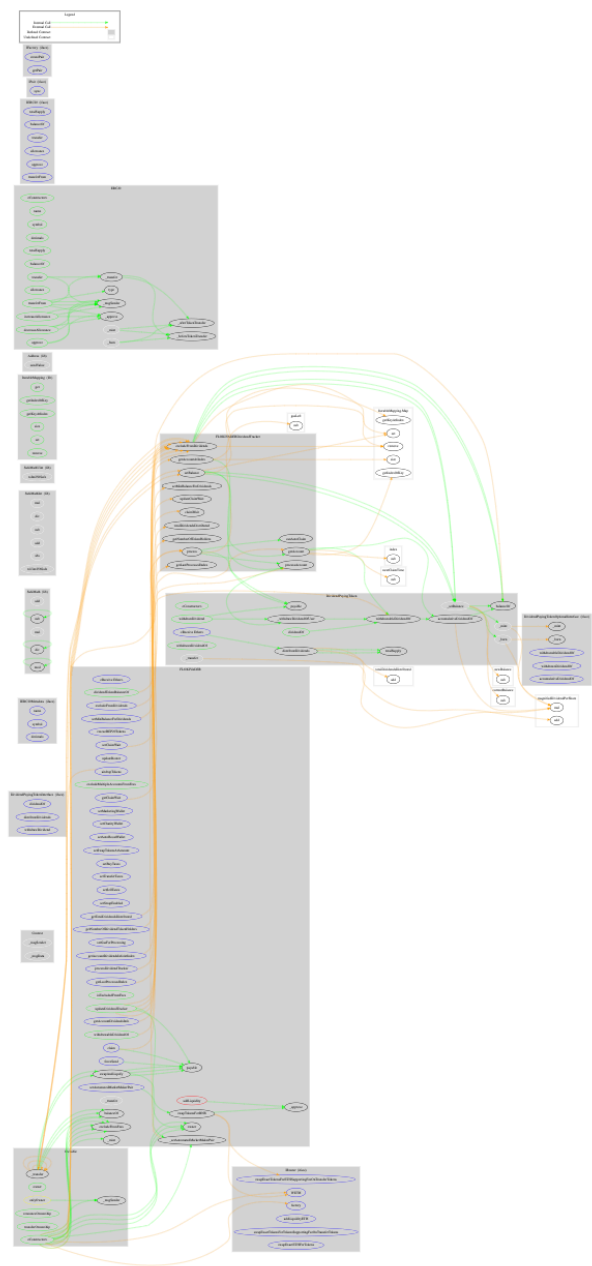
1.0	yes					yes → NewContract:FLOKIVADERDividendTracker
-----	-----	--	--	--	--	--

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. 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

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

updateDividendTracker  
processDividendTracker  
claim  
rescueBEP20Tokens  
forceSend  
updateRouter  
excludeFromFees  
excludeMultipleAccountsFromFees  
excludeFromDividends  
setMarketingWallet  
setCharityWallet  
setAutoBoostWallet  
setSwapTokensAtAmount  
setBuyTaxes  
setTransferTaxes  
setSellTaxes  
setSwapEnabled  
setAutomatedMarketMakerPair  
setMinBalanceForDividends  
setGasForProcessing  
setClaimWait  
airdropTokens

renounceOwnership  
transferOwnership

setMinBalanceForDividends  
excludeFromDividends  
updateClaimWait  
setBalance  
process  
processAccount

distributeDividends 💰  
withdrawDividend

transfer  
approve  
transferFrom  
increaseAllowance  
decreaseAllowance

## Deployer cannot mint any new tokens

Name	Exist	Tested	Status
Deployer cannot mint	✓	✓	✓
Max / Total Supply	1.000.000.000.000.000		

Comments:

**v1.0**

- `_setBalance` will mint tokens





## Deployer cannot burn or lock user funds

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

Comments:

**v1.0**

- `_setBalance` will burn tokens

## Deployer cannot pause the contract

Name	Exist	Tested	Status
Deployer cannot pause	—	—	—

Comments:

**v1.0**

.



## 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

- updateDividendTracker
  - onlyOwner
- processDividendTracker
- claim
- rescueBEP20Tokens
  - onlyOwner
- forceSend
- updateRouter
  - onlyOwner
- excludeFromFees
  - onlyOwner
- excludeMultipleAccountsFromFees
  - onlyOwner
- excludeFromDividends
  - onlyOwner
- setMarketingWallet
  - onlyOwner
- setCharityWallet
  - onlyOwner
- setAutoBoostWallet
  - onlyOwner
- setSwapTokensAtAmount
  - onlyOwner
- setBuyTaxes
  - onlyOwner
- setTransferTaxes
  - onlyOwner
- setSellTaxes
  - onlyOwner
- setSwapEnabled
  - onlyOwner
- setAutomatedMarketMakerPair
  - onlyOwner
- setMinBalanceForDividends
  - onlyOwner
- setGasForProcessing
  - onlyOwner
- setClaimWait
  - onlyOwner
- airdropTokens
  - onlyOwner

- transfer
- approve
- transferFrom
- increaseAllowance
- decreaseAllowance

- setMinBalanceForDividends
  - onlyOwner
- excludeFromDividends
  - onlyOwner
- updateClaimWait
  - onlyOwner
- setBalance
  - onlyOwner
- process
- processAccount
  - onlyOwner

- distributeDividends 💰
- withdrawDividend

## Comments

- Deployer can set following state variables without any limitations
  - swapTokensAtAmount
  - buyTaxes
  - totalBuyTax
  - transferTaxes
  - totalTransferTax
  - sellTaxes
  - totalSellTax
  - minimumTokenBalanceForDividends




























- Deployer can enable/disable following state variables
  - `_isExcludedFromFees`
  - `excludedFromDividends`
  - `swapEnabled`

**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/DividendPayingToken.sol	1	————	184	184	86	65	79	
	contracts/Context.sol	1	————	24	24	9	12	1	————
	contracts/DividendPayingTokenInterface.sol	————	1	41	13	3	20	10	
	contracts/IERC20Metadata.sol	————	1	28	17	4	16	9	
	contracts/SafeMath.sol	3	————	222	222	78	119	29	
	contracts/Ownable.sol	1	————	57	57	27	21	23	
	contracts/IterableMapping.sol	1	————	63	63	49	2	19	————
	contracts/flokivader.sol	3	————	546	504	430	53	380	  
	contracts/ERC20.sol	1	————	364	344	105	200	82	
	contracts/DividendPayingTokenOptionalInterface.sol	————	1	25	13	3	14	7	————
	contracts/IERC20.sol	————	1	87	32	17	62	13	
	contracts/IDex.sol	————	3	45	8	5	1	27	
	<b>Totals</b>	<b>11</b>	<b>7</b>	<b>1686</b>	<b>1481</b>	<b>816</b>	<b>585</b>	<b>679</b>	  

## 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	Main	A floating pragma is set	2	The current pragma Solidity directive is „^0.8.6“.
#3	Main	Missing Zero Address Validation (missing-zero-check)	156, 153, 150	Check that the address is not zero
#4	Main	Missing Events Arithmetic	167, 177, 162, 172	Emit an event for critical parameter changes

### Informational issues

Issue	File	Type	Line	Description
-------	------	------	------	-------------

#1	Main	Functions that are not used	365	Remove unused functions - addLiquidity L365
#2	Main	Unused state variables	154	Remove unused state variables

## Audit Comments

We recommend you to use the special form of comments (NatSpec Format for all of your functions, 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.

### 05. February 2022:

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

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

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

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