



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

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

MADE IN GERMANY

# Flashminer Audit

**Security Assessment**  
**11. March, 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	24
Commented Code exist	25
Audit Comments	26
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 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	10. March 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**

<https://flashminer.io/>

## **Telegram**

<https://t.me/flashminerChannel>

## **Twitter**

<https://twitter.com/FlashminerGame>

## **Discord**

<https://discord.com/invite/8ZvAwRTzr3>

## **Instagram**

<https://www.instagram.com/flashminer/>

## **Medium**

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

## Description

FlashMiner is a Play-to-Earn game on the Binance Smart Chain that involves generating the best possible computing power. The goal is to win GGW by fighting enemies with computing power as your main weapon. You will have to make strategic decisions to achieve your goal.

## Project Engagement

During the 9th of March 2022, **Flashminer 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**

- Provided 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:

```
IERC20  
IERC20Metadata  
Context  
📦 SafeMath  
Ownable  
IPancakeFactory  
IPancakeRouter  
IPancakePair  
TokenGuard
```

```
IERC20  
IERC20Metadata  
Context  
📦 SafeMath  
Ownable  
IPancakeFactory  
IPancakeRouter  
IPancakePair  
TokenGuard
```



## Tested Contract Files

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

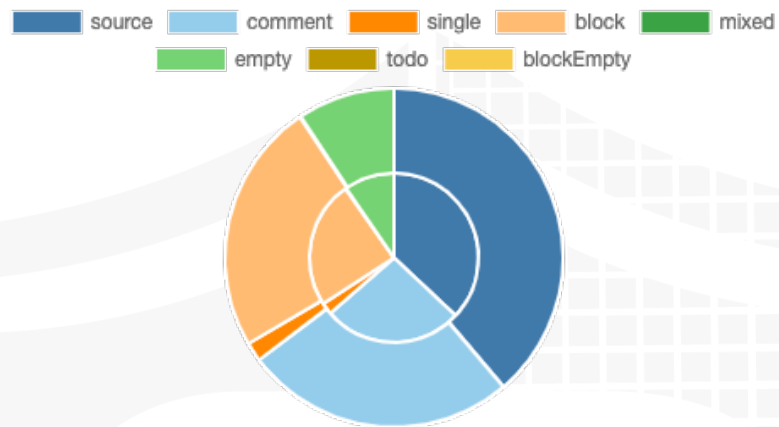
*A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.*

### v1.0

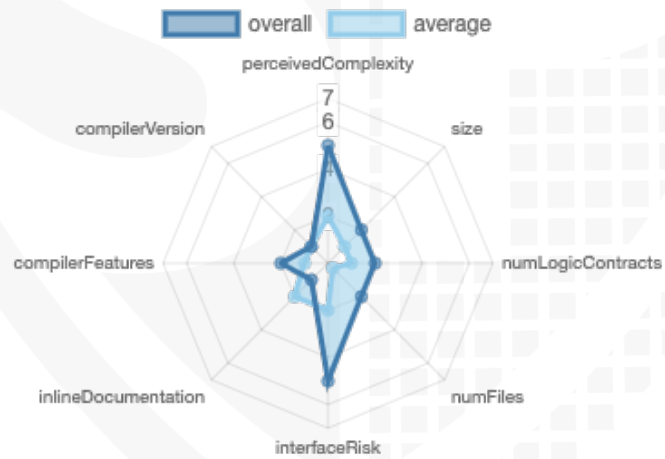
File Name	SHA-1 Hash
contracts/interfaces/IAAllowContract.sol	b6dacd634c28bcfcfa42ab014d34ca0a3a1e7d40
contracts/interfaces/IToken.sol	1e57a8cbd4f0e2bc33c6876e3e7b3f526ff867d3
contracts/base/AllowContract.sol	09b61460e4ea2491f902a96ae519c7ebfeced974
contracts/base/BaseContract.sol	4b78e85426de08cd7e6bda7a2cb214535fc6aba3
contracts/base/Ownable.sol	ec6717ef79f0d75bc16f6b92c7099c06781d08a1
contracts/CrazyMinerTokenGuard.sol	0554d84e0835c66a000d149fc9d54b734aecff54
contracts/CrazyMinerToken.sol	d26ad48113a0a30ec0ae160d5b7dd2291c2b0a0a

# Metrics

## Source Lines v1.0



## Risk Level v1.0



## Capabilities

### Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	3	1	8	5

### 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	70	0

Version	External	Internal	Private	Pure	View
1.0	25	67	3	15	34

### State Variables

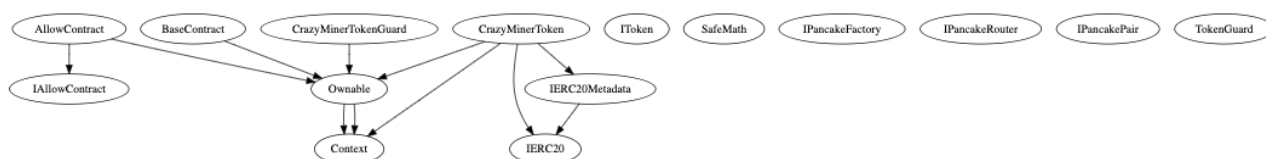
Version	Total	Public
1.0	24	14

### Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	<code>&gt;=0.8.0</code> <code>&lt;0.9.0</code>			yes (1 asm blocks)	

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

### CrazyMinerToken

```
setTGAddress  
excludeFromFee  
includeInFee  
setDexSellTax1Address  
setDexBuyTaxFee1AddressAddress  
sweep  
transfer  
approve  
transferFrom  
increaseAllowance  
decreaseAllowance  
renounceOwnership  
transferOwnership
```

### BaseContract

```
setPaused  
setAllowContractAddress  
setSignAddress  
ownerSweep  
ownerWithdraw
```

### CrazyMinerTokenGuard

```
addBlackList  
removeBlackList  
addWhiteList  
removeWhiteList  
batchAddWhiteList  
batchRemoveWhiteList  
setLimit  
setWhiteCheck
```

## Deployer cannot mint any new tokens

Name	Exist	Tested	Status
Deployer cannot mint	✓	✓	✓
Max / Total Supply	100.000.000		





## Deployer cannot burn or lock user funds

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

Comments:

### v1.0

- Deployer can lock user funds by
  - Setting blacklist addresses to true
  - Setting transactionLimit to 0

## Deployer cannot pause the contract

Name	Exist	Tested	Status
Deployer cannot pause	✓	✓	✗

Comments:

**v1.0**

- Deployer 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

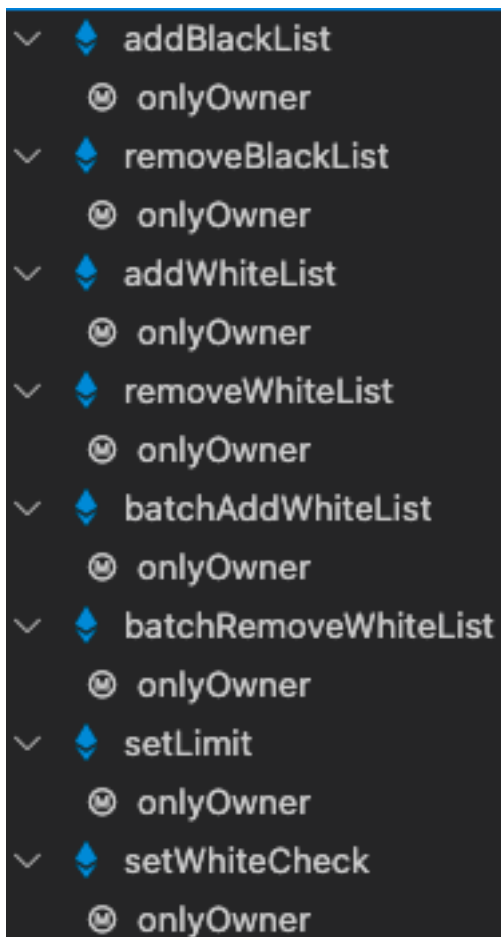
v1.0

- ✓  setTGAddress
  - Ⓜ onlyOwner
- ✓  excludeFromFee
  - Ⓜ onlyOwner
- ✓  includeInFee
  - Ⓜ onlyOwner
- ✓  setDexSellTax1Address
  - Ⓜ onlyOwner
- ✓  setDexBuyTaxFee1AddressAddress
  - Ⓜ onlyOwner
- ✓  sweep
  - Ⓜ onlyOwner
-  transfer
-  approve
-  transferFrom
-  increaseAllowance
-  decreaseAllowance

- ✓  renounceOwnership
  - Ⓜ onlyOwner
- ✓  transferOwnership
  - Ⓜ onlyOwner

- ✓  setPaused
  - Ⓜ onlyOwner
- ✓  setAllowContractAddress
  - Ⓜ onlyOwner
- ✓  setSignAddress
  - Ⓜ onlyOwner
- ✓  ownerSweep
  - Ⓜ onlyOwner
- ✓  ownerWithdraw
  - Ⓜ onlyOwner

- ✓  **set**
  - Ⓜ onlyOwner



A screenshot of a smart contract interface with a dark background. It lists several functions, each preceded by a blue diamond icon and a checkmark. Each function is followed by a modifier icon (a circle with a 'u') and the text 'onlyOwner'. The functions are: addBlackList, removeBlackList, addWhiteList, removeWhiteList, batchAddWhiteList, batchRemoveWhiteList, setLimit, and setWhiteCheck.

- ✓ addBlackList
  - Ⓢ onlyOwner
- ✓ removeBlackList
  - Ⓢ onlyOwner
- ✓ addWhiteList
  - Ⓢ onlyOwner
- ✓ removeWhiteList
  - Ⓢ onlyOwner
- ✓ batchAddWhiteList
  - Ⓢ onlyOwner
- ✓ batchRemoveWhiteList
  - Ⓢ onlyOwner
- ✓ setLimit
  - Ⓢ onlyOwner
- ✓ setWhiteCheck
  - Ⓢ onlyOwner



















## Comments

- Deployer can set following state variables without any limitations
  - transactionLimit
- Deployer can enable/disable following state variables
  - blackList
  - whiteList
  - whiteCheck
  - \_isExcludedFromFee
- Deployer can set following addresses
  - tgAddress
  - dexSellTax1Address
  - dexBuyTaxFee1Address

**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/interfaces/IAllowContract.sol	—————	1	6	5	3	1	3	
	contracts/interfaces/IToken.sol	—————	1	24	7	3	3	15	—————
	contracts/base/AllowContract.sol	1	—————	21	21	14	3	16	—————
	contracts/base/BaseContract.sol	1	—————	80	78	50	13	45	
	contracts/base/Ownable.sol	2	—————	68	68	35	23	24	—————
	contracts/CrazyMinerTokenGuard.sol	1	—————	90	82	59	8	65	
	contracts/CrazyMinerToken.sol	4	6	960	743	344	387	260	  
	<b>Totals</b>	9	8	1249	1004	508	438	428	   

## 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	All	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 the source file	The current pragma Solidity directive is „>=0.8.0 <0.9.0 "".
#3	BaseContract	Missing Zero Address Validation (missing-zero-check)	47, 52	Check that the address is not zero
#4	CrazyMinerToken	Missing Zero Address Validation (missing-zero-check)	600, 594, 579	Check that the address is not zero
#5	AllowContract	State variable visibility is not set	8	It is best practice to set the visibility of state variables explicitly

#6	CrazyMi nterTok en	Local variables shadowing	950, 716	Rename the local variables that shadow another component
#7	CrazyMi nterTok enGuar d	Missing Events Arithmetic	47	Emit an event for critical parameter changes
#8	BaseCo ntract	Missing Events Arithmetic	48	Emit an event for critical parameter changes

## Informational issues

Issue	File	Type	Line	Description
#1	CrazyMi nterTok en	State variables that could be declared constant (constable- states)	522, 520, 521, 526, 524	Add the `constant` attributes to state variables that never change
#2	Main	NatSpec documentation missing	-	If you started to comment your code, also comment all other functions, variables etc.
#3	CrazyMi nterToke nGuard	Wrong BaseContract path	6	Remove one "." In the path. It should look like the following  Import "./base/ BaseContract.sol"
#4	IToken	Wrong SPDX License	2	MI231321T is wrong. You can choose one of the following licenses from the following page:  <a href="https://spdx.org/licenses/">https://spdx.org/licenses/</a>
#5	BaseCo ntract	Wrong SPDX License	2	MI231321T is wrong. You can choose one of the following licenses from the following page:  <a href="https://spdx.org/licenses/">https://spdx.org/licenses/</a>
#6	Ownabl e	Wrong SPDX License	2	MI231321T is wrong. You can choose one of the following licenses from the following page:  <a href="https://spdx.org/licenses/">https://spdx.org/licenses/</a>



#7	All	Safemath is unnecessary	-	<p>Pragma version above 0.8.x implements the overflow/underflow check automatically.</p> <p>If you want to remove the library please make sure to change every safemath operation to raw mathematical operations</p>
#8	CrazyMinerToken	Ineffective unchecked statement	889	Unchecked statement will be used to protect the contract from the overflow/underflow effect (because it's automatically implemented in pragma version above 0.8.x) while mathematical operations but you used safemath function which prevent the overflow/underflow. So in this case it is inefficient
#9	CrazyMinerToken	Use descriptive parameters	94	Please use a descriptive parameter instead of "t"
#10	CrazyMinerTokenGuard	BaseContract is not used	6	Remove imported file
#11	AllowContract	BaseContract is not used	6	Remove imported file

## Commented Code exist

There are some instances of code being commented out in the following files that should be removed:

File	Line	Comment
CrazyMinerToken	556	//IPancakeRouter _router = IPancakeRouter(0x9Ac64Cc6e4415144C455BD8E4837Fea55603e5c3); //for test

## Recommendation

Remove the commented code, or address them properly.

## 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.

### 11. March 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>	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>NOT 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>NOT 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 word "SolidProofed" in a white, handwritten-style script. 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.

SolidProofed

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

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

MADE IN GERMANY