



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

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

MADE IN GERMANY

**v1.0: 29. January, 2022**

# Audit

**Security Assessment**  
**01. February, 2022**

**For**



**Beli Finance**

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	18
Source Units in Scope	21
Critical issues	22
High issues	22
Medium issues	22
Low issues	22
Informational issues	22
Audit Comments	22
SWC Attacks	23

## 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	29. January 2022	<ul style="list-style-type: none"><li>• Layout project</li><li>• Automated- /Manual-Security Testing</li><li>• Summary</li></ul>
1.1	01. February 2022	<ul style="list-style-type: none"><li>• Reaudit</li></ul>

## **Network**

Binance Smart Chain (BEP20)

## **Website**

<https://beli.finance/>



## Description

Beli Finance is a decentralized reserve policy-controlled currency protocol, Multi-Chain Yield Optimizer Aggregator platform that convert yield farm into \$BELI token and still earn small compound interest on their crypto holdings automatically. Each \$BELI token is backed by a basket of assets (e.g., USDT, BNB, BUSD Tokens etc etc) in the treasury, giving it an intrinsic value that it cannot fall below. Beli Finance also introduces economic and game-theoretic dynamics into the market through staking and compounding.

## Project Engagement

During the 27th of January 2022, **Beli Finance 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



# Beli Finance

## Contract Link

### v1.0

- Github
  - <https://github.com/belifinance/beli-finance>
  - Commit: fe03b705871f742e0eabe3cad79ba9e75c781ff6

### v1.1

- Github
  - <https://github.com/belifinance/beli-finance>
  - Commit: b4038bd4f1426e7d9bbf283e11bbb2caafef9139

# 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:

```
./interfaces/IERC20.sol  
./libraries/SafeERC20.sol  
./helpers/AccessControl.sol  
./helpers/ReentrancyGuard.sol  
./interfaces/IVault.sol  
./interfaces/IFarm.sol  
./interfaces/ILocker.sol  
./interfaces/IFeeReceiver.sol
```



## Tested Contract Files

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

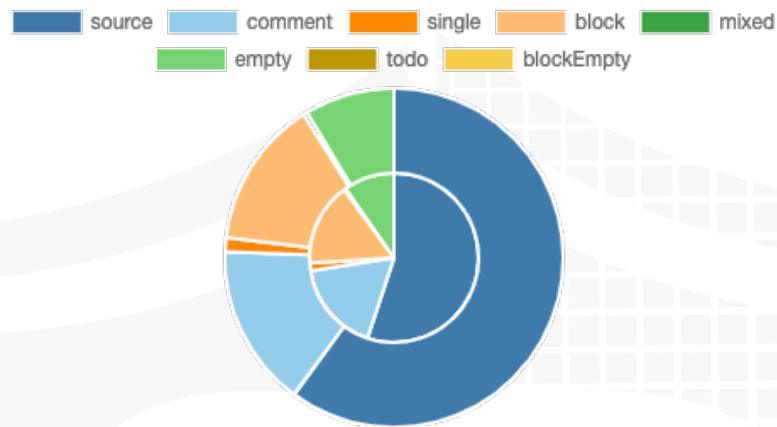
*A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.*

### v1.0

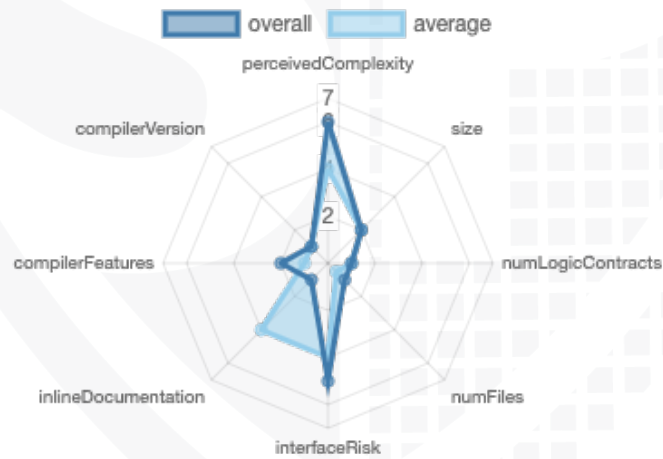
File Name	SHA-1 Hash
contracts/TimelockTreasury.sol	7790bd2f654b2ea954d47af96cba7e83b5d96cf2
contracts/TimelockControllers.sol	5ebf09327e8037af545840d02b880a39b2a5d131

# Metrics

## Source Lines v1.0



## Risk Level v1.0



## Capabilities

### Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	2	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	62	10

Version	External	Internal	Private	Pure	View
1.0	8	53	8	4	11

### State Variables

Version	Total	Public
1.0	22	18

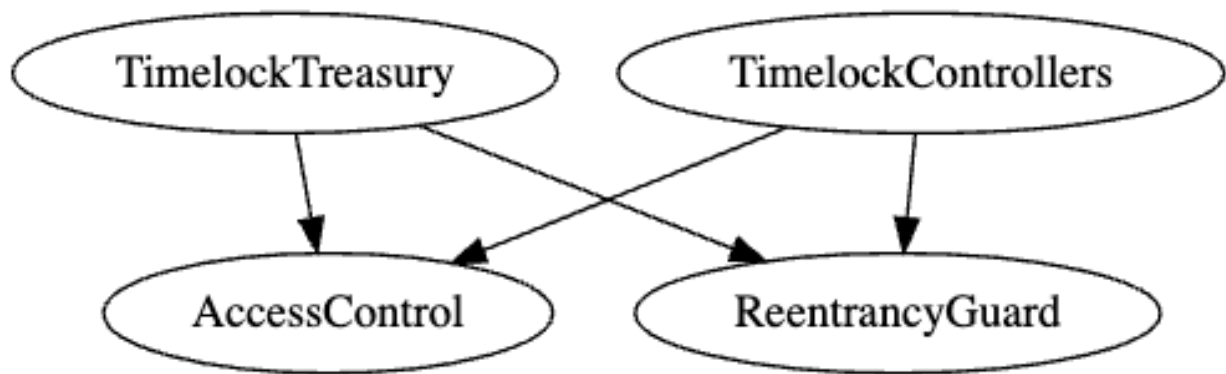
### Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	0.6.12	ABIEncoderV2	yes		

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

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

## Inheritance Graph v1.0



[illegible]

## 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. Overall checkup (Smart Contract Security)



## Write functions of contract v1.0

### TimelockController

schedule  
scheduleBatch  
cancel  
execute 💰  
executeBatch 💰  
updateMinDelay  
updateMinDelayReduced  
setDevWalletAddress  
scheduleSetFarmSettings  
executeSetFarmSettings  
scheduleSet  
executeSet 💰  
withdrawBNB 💰  
withdrawBEP20 💰  
add  
earn  
scheduleSetVaultSettings  
executeSetVaultSettings 💰  
farm  
pause  
unpause  
rebalance  
deleverageOnce  
leverageOnce  
wrapBNB  
setBeneficiaries  
noTimeLockFunc1  
noTimeLockFunc2  
noTimeLockFunc3

grantRole  
revokeRole  
renounceRole

## TimelockTreasury

schedule  
scheduleBatch  
cancel  
execute 💰  
executeBatch 💰  
updateMinDelay  
updateMinDelayReduced  
setDevWalletAddress  
convertTokensToReserve  
convertTokensToWBNB  
distributeReservesTo  
distributeRewards  
removeBeneficiary  
addReserveToken  
removeReserveToken  
setBeneficiaries

grantRole  
revokeRole  
renounceRole



## Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

### Legend

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

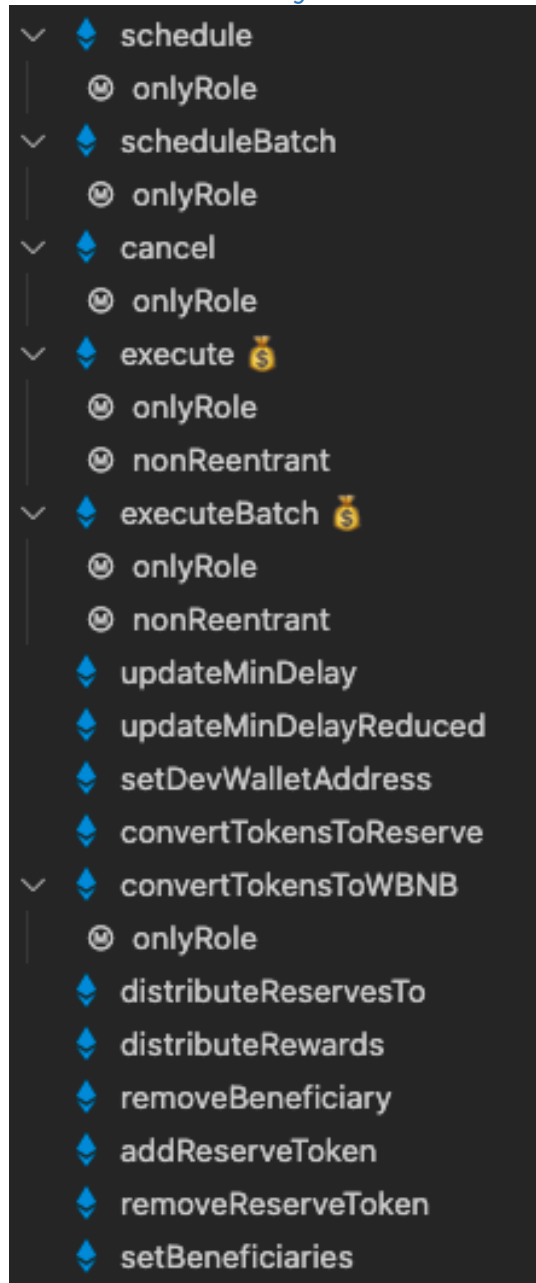
# Modifiers and public functions

## v1.0

### TimelockControllers

▼ 🔹 schedule 🔒 onlyRole	▼ 🔹 add 🔒 onlyRole
▼ 🔹 scheduleBatch 🔒 onlyRole	▼ 🔹 earn 🔒 onlyRole
▼ 🔹 cancel 🔒 onlyRole	▼ 🔹 scheduleSetVaultSettings 🔒 onlyRole
▼ 🔹 execute 💰 🔒 onlyRole 🔒 nonReentrant	▼ 🔹 executeSetVaultSettings 💰 🔒 onlyRole 🔒 nonReentrant
▼ 🔹 executeBatch 💰 🔒 onlyRole 🔒 nonReentrant 🔹 updateMinDelay 🔹 updateMinDelayReduced 🔹 setDevWalletAddress	▼ 🔹 farm 🔒 onlyRole
▼ 🔹 scheduleSetFarmSettings 🔒 onlyRole	▼ 🔹 pause 🔒 onlyRole
▼ 🔹 executeSetFarmSettings 🔒 onlyRole	▼ 🔹 unpause 🔒 onlyRole
▼ 🔹 scheduleSet 🔒 onlyRole	▼ 🔹 rebalance 🔒 onlyRole
▼ 🔹 executeSet 💰 🔒 onlyRole 🔒 nonReentrant 🔹 withdrawBNB 💰 🔹 withdrawBEP20 💰	▼ 🔹 deleverageOnce 🔒 onlyRole
	▼ 🔹 leverageOnce 🔒 onlyRole
	▼ 🔹 wrapBNB 🔒 onlyRole 🔹 setBeneficiaries
	▼ 🔹 noTimeLockFunc1 🔒 onlyRole
	▼ 🔹 noTimeLockFunc2 🔒 onlyRole
	▼ 🔹 noTimeLockFunc3 🔒 onlyRole

## TimelockTreasury



## Comments







- Deployer can set following state variables without any limitations
  - TimelockControllers
    - minDelay
    - minDelayReduced
- Deployer can enable/disable following state variables
- OnlyDevWalletAddress can
  - withdraw contract balance
  - Withdraw BEP20 token balance

**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/TimelockTreasury.sol	1	————	762	682	452	150	312	
	contracts/TimelockControllers.sol	1	————	845	709	484	147	305	
	<b>Totals</b>	<b>2</b>	————	<b>1607</b>	<b>1391</b>	<b>936</b>	<b>297</b>	<b>617</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	TimelockControllers	Missing Zero Address Validation (missing-zero-check)	137, 459, 131, 130, 467	Check that the address is not zero

## Informational issues

**No informational issues**

## Audit Comments

### 29. January 2022:

- [Read whole report for more information](#)

### 01. February 2022:

- Issues were fixed by bell finance team
- [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>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 C-1 05</a>	Unprotected Ether Withdrawal	<a href="#">CWE-284: Improper Access Control</a>	<b>PASSED</b>
<a href="#">SW C-1 04</a>	Unchecked Call Return Value	<a href="#">CWE-252: Unchecked Return Value</a>	<b>PASSED</b>
<a href="#">SW C-1 03</a>	Floating Pragma	<a href="#">CWE-664: Improper Control of a Resource Through its Lifetime</a>	<b>PASSED</b>
<a href="#">SW C-1 02</a>	Outdated Compiler Version	<a href="#">CWE-937: Using Components with Known Vulnerabilities</a>	<b>PASSED</b>
<a href="#">SW C-1 01</a>	Integer Overflow and Underflow	<a href="#">CWE-682: Incorrect Calculation</a>	<b>PASSED</b>
<a href="#">SW C-1 00</a>	Function Default Visibility	<a href="#">CWE-710: Improper Adherence to Coding Standards</a>	<b>PASSED</b>

The logo features the words "SolidProof" in a white, handwritten-style script. The "P" is large and stylized, with a long horizontal stroke that extends to the left. The background is a solid blue color with a faint, large, light-blue shield-like shape in the center. Inside the shield, there is a grid pattern of small squares.

SolidProof

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

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

MADE IN GERMANY