



**SOLID**Proof  
*Bring trust into your projects*

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

MADE IN GERMANY

# Metaverse-DAO Audit

**Security Assessment**  
**30. 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
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 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	30. 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://www.metaverse-dao.io/>

## **Telegram**

[https://t.me/metaverse\\_dao\\_community](https://t.me/metaverse_dao_community)

## **Twitter**

[https://twitter.com/METADAO\\_Offical](https://twitter.com/METADAO_Offical)

## **Facebook**

<https://www.facebook.com/Metaverse-DAO-107675525207534>

## **Reddit**

[https://www.reddit.com/r/Metaverse\\_DAO](https://www.reddit.com/r/Metaverse_DAO)

## **Medium**

<https://medium.com/@Metaverse-DAO>

## **Youtube**

<https://www.youtube.com/watch?v=SEHWStUUD9g>

## Description

Metaverse-DAO (METADA0) is a decentralized autonomous organization (DAO) for the development, management of decentralized network based games, NFTs, finance (Defi) and other projects.

Our mission is to build strong virtual worlds (part of the metaverse), create really fun games and Dapps, and manage all the projects and assets with our community members, allowing token holders to share the profits.

## Project Engagement

During the 27th of March 2022, **Metaverse-DAO 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**

- <https://bscscan.com/token/0x6195D22d416E2D2c7C4A883c977Dd9Bfd4e65432>

# 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:

```
Address.sol
Context.sol
DividendPayingToken.sol
DividendTracker.sol
ERC20.sol
IDividendPayingToken.sol
IDividendPayingTokenOptional.sol
IERC20.sol
IterableMapping.sol
IUniswapV2Factory.sol
IUniswapV2Pair.sol
IUniswapV2Router01.sol
IUniswapV2Router02.sol
Math.sol
metaversedao.sol
Ownable.sol
SafeMath.sol
SafeMathInt.sol
SafeMathUint.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.*

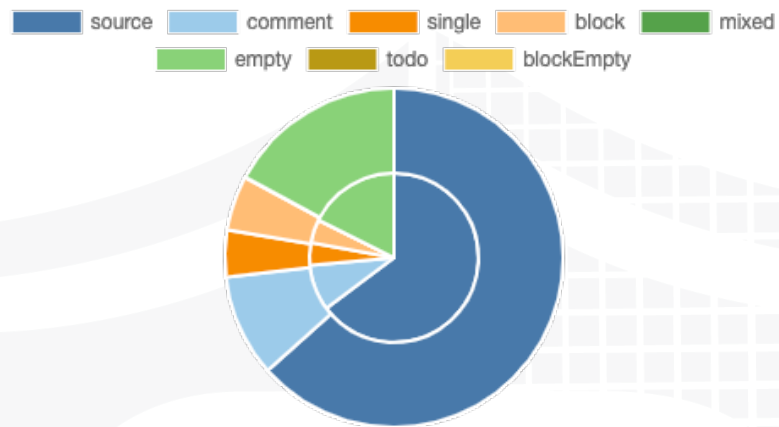
### v1.0



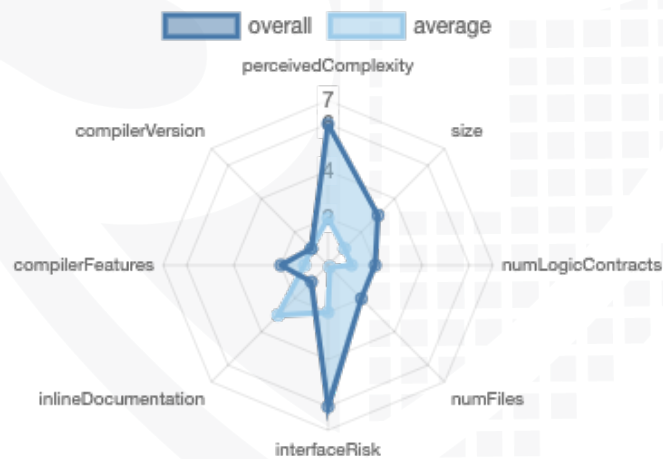
File Name	SHA-1 Hash
contracts/DividendPayingToken.sol	9fa1f87a7a39c81e7c75f1dfea6975f2f1f8848f
contracts/IUniswapV2Pair.sol	93e24ad0ad08c5480ffa45dba9cb55b7203688f4
contracts/SafeMathUint.sol	519fd1937a3d76270282d3ccb9ec7163992708aa
contracts/Math.sol	f6455bdcf310b73ce5fb9d6a6035ba0512e9cf6c
contracts/Context.sol	15561b12740d13715a13ca534aa6036c1227d8c0
contracts/IUniswapV2Factory.sol	a5a1c7cf581ea1fcdda5fb226eaa494c69e1df11
contracts/SafeMathInt.sol	300538bf12abba4c0bea23ead3079bbb43e27e49
contracts/IDividendPayingTokenOptional.sol	3031c68dfb51a079268098ce88e70b728bb55889
contracts/Address.sol	cd7032c7c90284aa2c03fd6fe1d958297c2a928f
contracts/SafeMath.sol	f04c94c0600e0edb6f63fbd3861c74722094401e
contracts/IDividendPayingToken.sol	521bae9924be430526a46ea2f7abda0ac2410a2a
contracts/Ownable.sol	b2d7d4c0ec2a1b3901a1d9e75d34b85c8b1ef67f
contracts/IterableMapping.sol	a6754b03427b59bbe7dcc7f6fc52ff22bbdee72
contracts/IUniswapV2Router02.sol	ceeb197905988a225c220f9cb959571be4a2140b
contracts/IUniswapV2Router01.sol	f5eb22b303c24a061253d191fe59e32486c1fe66
contracts/ERC20.sol	906459a5d75b875289f06bc26fbd9a1376ddf391
contracts/DividendTracker.sol	787d7f648818c47d5fe54cc3b45b263065edd289
contracts/metaversedao.sol	9efaf281603b4db22ba5cc211cbd8923c3637bef
contracts/IERC20.sol	124d442a8a0abcba5afbc2ed8b5a10ec2292c5e3

# Metrics

## Source Lines v1.0



## Risk Level v1.0



## Capabilities

### Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	5	6	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	165	8

Version	External	Internal	Private	Pure	View
1.0	79	151	15	27	78

### State Variables

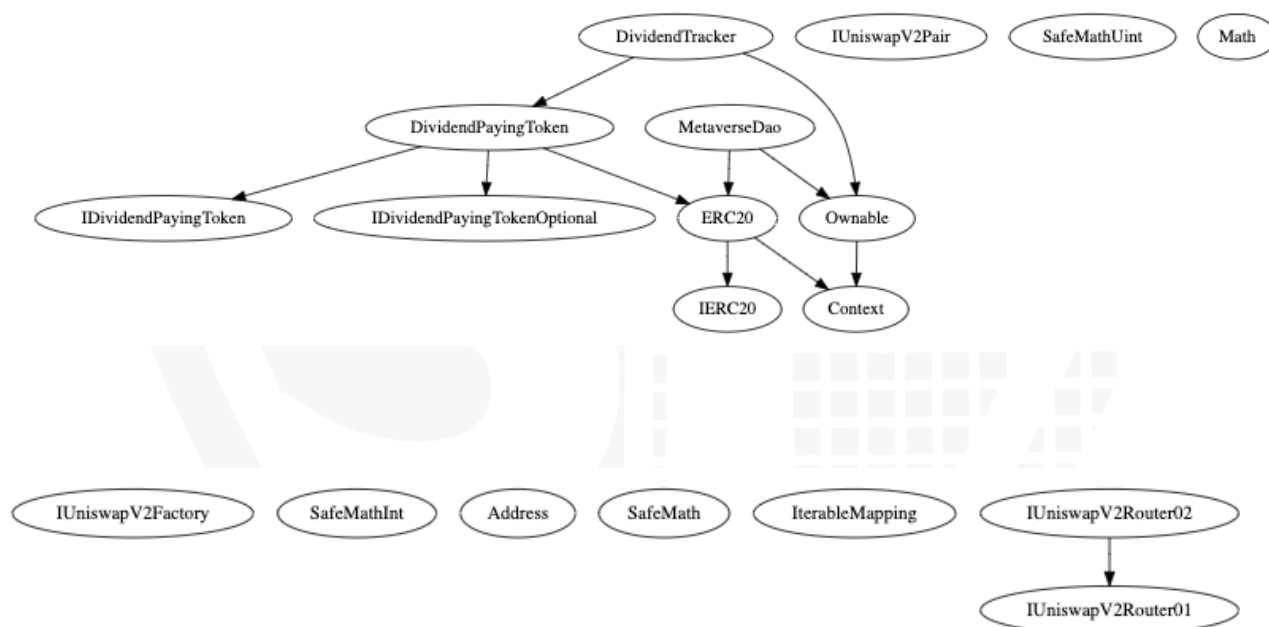
Version	Total	Public
1.0	62	39

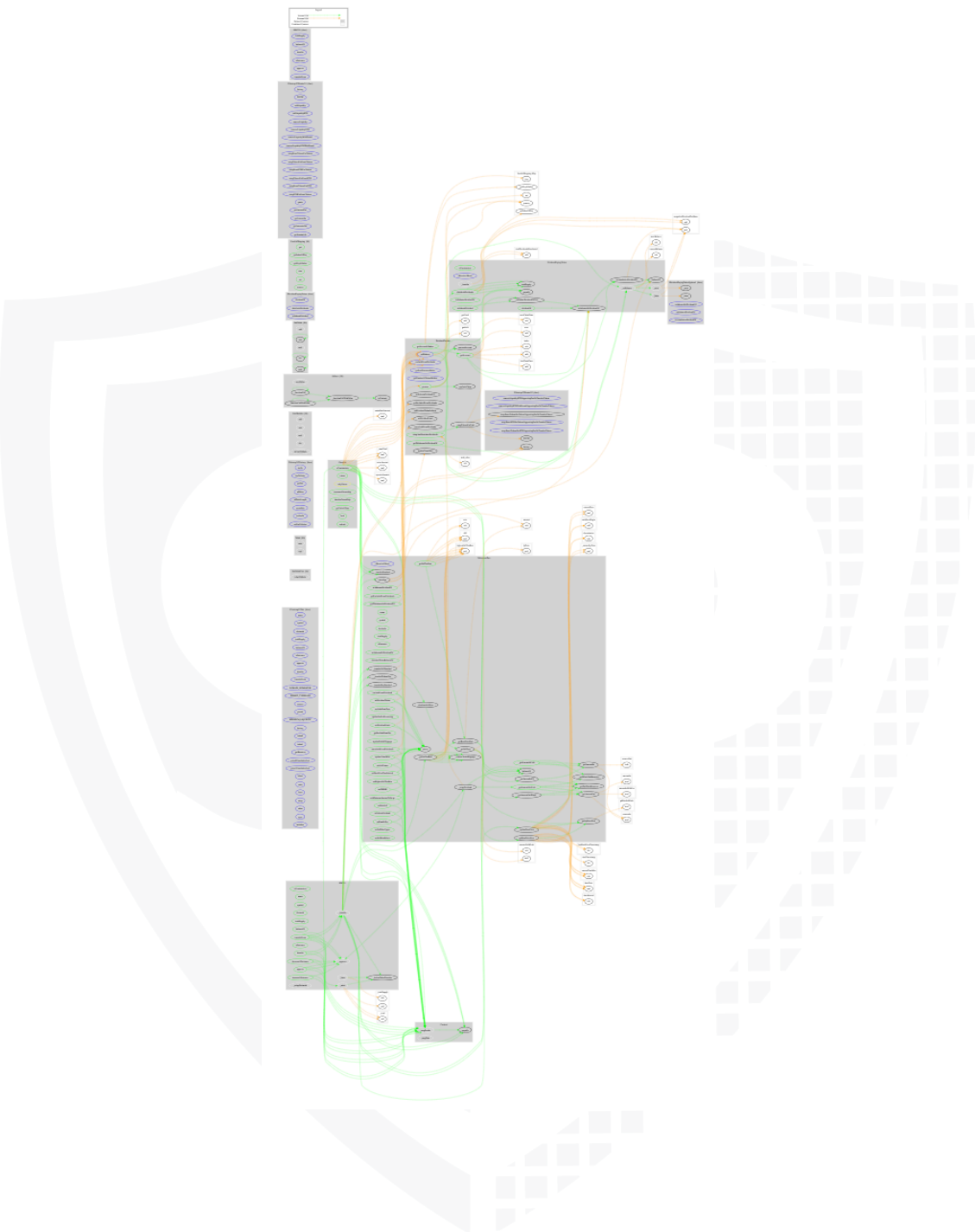
### Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	<code>^0.6.12</code>	<code>ABIEncoderV2</code>	<code>yes</code>	<code>yes</code> (2 asm blocks)	

Version	Transfers ETH	Low-Level Calls	DelegateCall	Uses Hash Functions	EC Recover	New/Create/Create2
1.0	yes					yes → NewContract: DividendTracker

## Inheritance Graph 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).

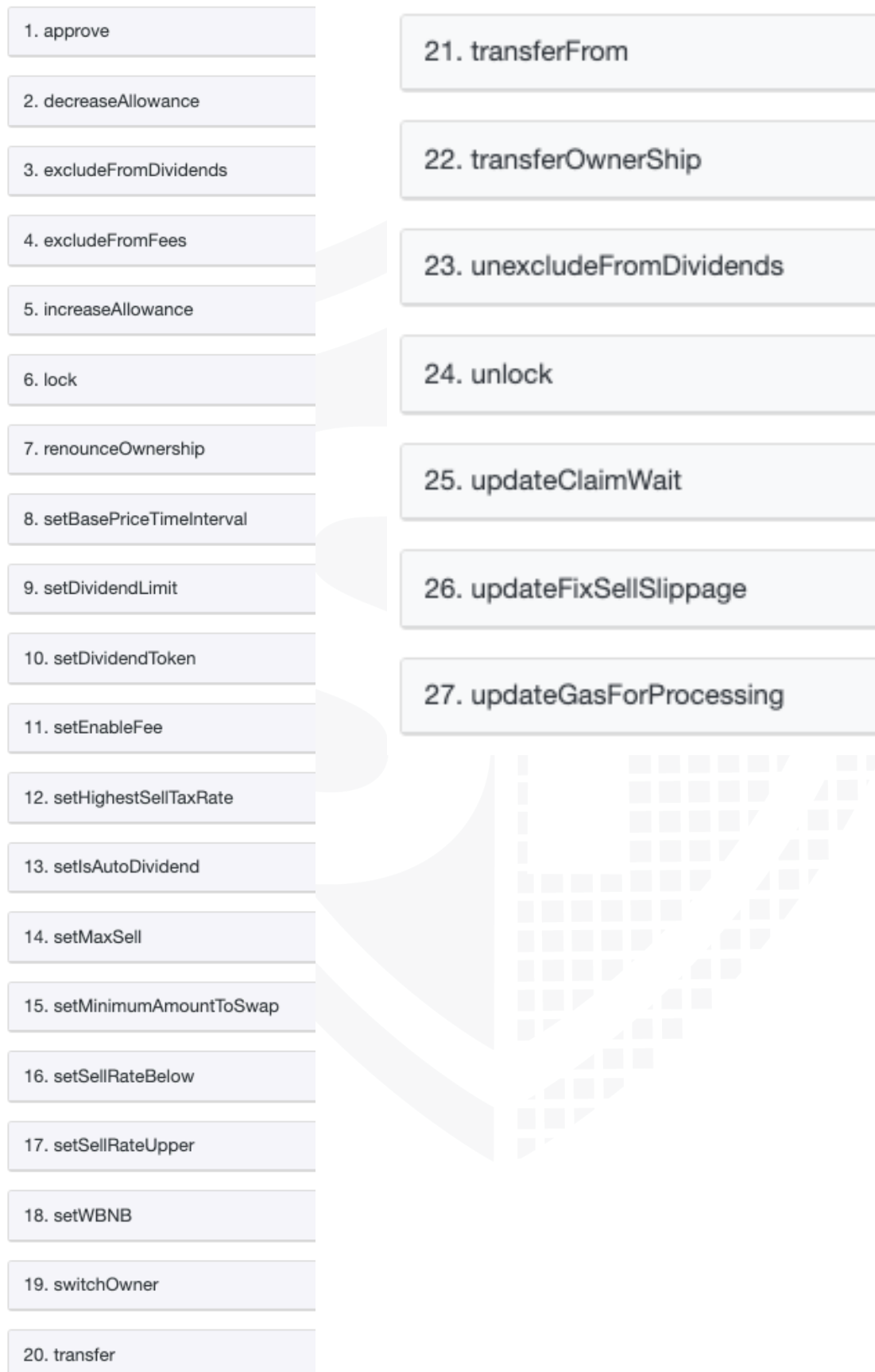
We will verify the following claims:

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

ERC20				
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



## Deployer cannot mint any new tokens

Name	Exist	Tested	Status
Deployer cannot mint	✓	✓	✓
Max / Total Supply	350.000.000		





## Deployer cannot burn or lock user funds

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

Comments:

### v1.0

- `_setBalance` in `DividendPayingToken` will mint/burn tokens but deployer cannot burn tokens for specific addresses
- Deployer can lock user funds by
  - Setting `max sell` variable to 0

## Deployer cannot pause the contract

Name	Exist	Tested	Status
Deployer cannot pause	—	—	—



## 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

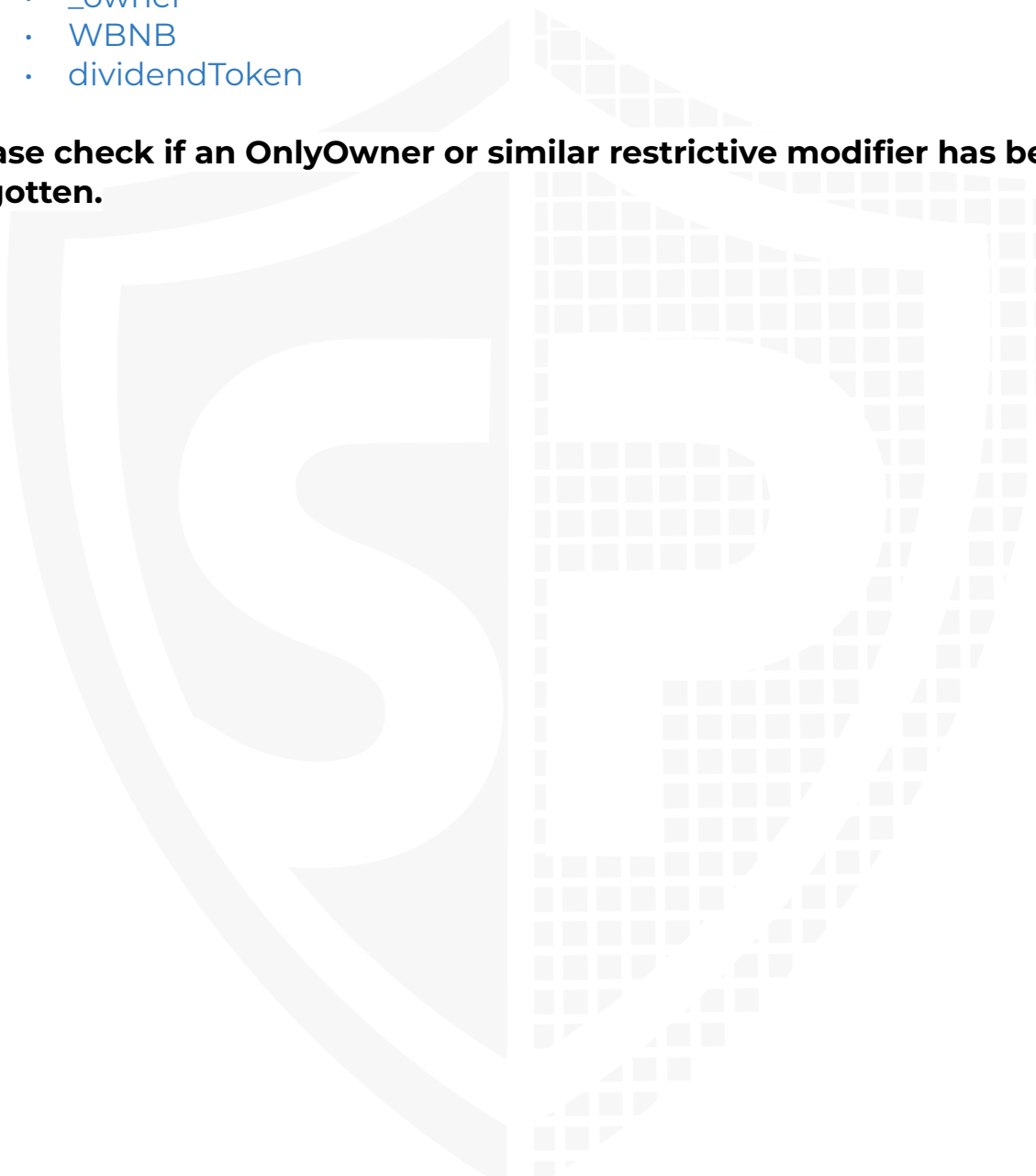
transfer	
transferFrom	
approve	
switchOwner	
setBasePriceTimeInterval	
setHighestSellTaxRate	
setWBNB	
setMinimumAmountToSwap	
setMaxSell	
setIsAutoDividend	
excludeFromFees	
onlyOwner	
updateGasForProcessing	
onlyOwner	
setEnabledFee	
updateFixSellSlippage	
onlyOwner	
setDividendLimit	
setDividendToken	
updateClaimWait	
excludeFromDividends	
unexcludeFromDividends	
setSellRateUpper	
onlyOwner	
setSellRateBelow	
onlyOwner	
renounceOwnership	
onlyOwner	
transferOwnership	
onlyOwner	
lock	
onlyOwner	
unlock	

## Comments

- Deployer can set following state variables without any limitations
  - basePriceTimeInterval
  - highestSellTaxRate
  - minimumAmountToSwap
  - \_maxSell
  - fixSellSlippage
  - \_dividendLimitUsd
  - sellRateUpper
  - sellRateBelow








































- Deployer can enable/disable following state variables
  - isAutoDividend
  - \_isExcludedFromFee
  - enableFee
  - excludedFromDividends
- Deployer can set following addresses
  - \_owner
  - WBNB
  - dividendToken

**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	————	199	199	101	53	94	
	contracts/IUniswapV2Pair.sol	————	1	74	9	6	1	55	
	contracts/SafeMathUint.sol	1	————	11	11	9	1	3	
	contracts/Math.sol	1	————	25	25	19	3	5	
	contracts/Context.sol	1	————	13	13	10	2	2	————
	contracts/IUniswapV2Factory.sol	————	1	19	8	5	1	17	
	contracts/SafeMathInt.sol	1	————	33	33	27	1	8	
	contracts/IDividendPayingTokenOptional.sol	————	1	27	15	4	16	7	
	contracts/Address.sol	1	————	76	64	38	18	37	
	contracts/SafeMath.sol	1	————	65	65	40	14	10	
	contracts/IDividendPayingToken.sol	————	1	43	15	4	22	10	
	contracts/Ownable.sol	1	————	70	70	45	12	40	
	contracts/IterableMapping.sol	1	————	64	64	50	2	19	
	contracts/IUniswapV2Router02.sol	————	1	50	9	5	1	16	
	contracts/IUniswapV2Router01.sol	————	1	97	6	4	1	48	
	contracts/ERC20.sol	1	————	138	138	89	25	79	
	contracts/DividendTracker.sol	1	————	274	256	184	1	141	
	contracts/metaversedao.sol	1	————	658	658	540	3	544	
	contracts/IERC20.sol	————	1	60	18	12	38	13	
	<b>Totals</b>	<b>12</b>	<b>7</b>	<b>1996</b>	<b>1676</b>	<b>1192</b>	<b>215</b>	<b>1148</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	All	A floating pragma is set	At the top of source files	The current pragma Solidity directive is „^0.6.12”.
#3	Dividen dTracke r	Missing Zero Address Validation (missing-zero-check)	240	Check that the address is not zero
#4	Metaver seDao	State variables shadowing	20, 22, 37, 27, 23, 28	Rename the state variables that shadow another component
#5	Metaver seDao	Local variables shadowing	176, 520	Rename the local variables that shadow another component

#6	Dividen dPaying Token	Local variables shadowing	57	Rename the local variables that shadow another component
#7	Metaver seDao	Missing Events Arithmetic	640, 636, 587	Emit an event for critical parameter changes

## Informational issues

Issue	File	Type	Line	Description
#1	Dividen dPaying Token	State variables that could be declared constant (constable- states)	33	Add the `constant` attributes to state variables that never change
#2	Metaver seDao	State variables that could be declared constant (constable- states)	55, 37, 54, 33, 27, 56, 57, 28, 45, 29, 30	Add the `constant` attributes to state variables that never change
#3	Dividen dPaying Token	Unused state variables	33	Remove unused state variables
#4	Metaver seDao	Misspelling	See description	Change following words:  - reciever to receiver L128  Make sure to change it everywhere else as well.



#5	Dividend Paying Token	Unreachable code	156-162	<p>Remove the following coloured part from function because it will never executed:</p> <pre> function _transfer(address from, address to, uint256 value) internal virtual override {     require(false);      int256 _magCorrection = magnifiedDividendPerShare. mul(value).toInt256Safe();      magnifiedDividendCorrectio ns[from] = magnifiedDividendCorrectio ns[from].add(_magCorrectio n);      magnifiedDividendCorrectio ns[to] = magnifiedDividendCorrectio ns[to].sub(_magCorrection); } </pre>
----	-----------------------	------------------	---------	--

## 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.

### 30. 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>	NOT 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>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>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 "SolidProof" in a white, handwritten-style script. The letters are fluid and connected, with a large 'P' and 'S'. The text is set against a dark blue background. Behind the text is a faint, stylized shield emblem. The shield has a grid-like pattern on its right side and a solid blue area on its left. The overall design conveys a sense of security and reliability.

SolidProof

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

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

MADE IN GERMANY