



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

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

MADE IN GERMANY

**Pepe CEO**

**Audit**

**Security Assessment**

**25.May,2023**

**For**



**SolidProof\_io**



**@solidproof\_io**

<b>Disclaimer</b>	<b>2</b>
<b>Description</b>	<b>5</b>
<b>Project Engagement</b>	<b>5</b>
<b>Logo</b>	<b>5</b>
<b>Contract Link</b>	<b>5</b>
<b>Methodology</b>	<b>7</b>
<b>Used Code from other Frameworks/Smart Contracts (direct imports)</b>	<b>8</b>
<b>Tested Contract Files</b>	<b>10</b>
<b>Source Lines</b>	<b>11</b>
<b>Risk Level</b>	<b>11</b>
<b>Capabilities</b>	<b>12</b>
<b>Inheritance Graph</b>	<b>13</b>
<b>CallGraph</b>	<b>14</b>
<b>Scope of Work/Verify Claims</b>	<b>15</b>
<b>Modifiers and public functions</b>	<b>21</b>
<b>Source Units in Scope</b>	<b>23</b>
<b>Critical issues</b>	<b>24</b>
<b>High issues</b>	<b>24</b>
<b>Medium issues</b>	<b>24</b>
<b>Low issues</b>	<b>24</b>
<b>Informational issues</b>	<b>25</b>
<b>Audit Comments</b>	<b>25</b>
<b>SWC Attacks</b>	<b>27</b>

# 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	25.May,2023	<ul style="list-style-type: none"><li>• Layout project</li><li>• Automated- /Manual-Security Testing</li><li>• Summary</li></ul>

**Network**  
**Binance Smart Chain(BSC)**

**Website**  
**<https://www.pepesceo.org/>**

**Twitter**  
**[https://twitter.com/BSC\\_PepeCEO](https://twitter.com/BSC_PepeCEO)**

**Telegram**  
**<https://t.me/pepeceochat>**



## Description

In the BSC ecosystem, Pepe CEO is a Meme token worth community-driven and uncontrollable, PepeCEO is based on Shiba Token and Elon Musk's idea. The Pepe CEO meme will undoubtedly have the same possibility of success as Pepe, Doge, and other memes with the community's backing and Elon Musk's encouragement.

Announcing PepeCEO, a fun new game where players compete to use artificial intelligence to make the funniest, most inventive memes!

You can compete against other users in a competition to earn prizes in BNB or set the goal of creating the ideal meme for yourself in Shiba CEO. You can instantly create original memes that are certain to attract laughs and shares using our AI-powered meme generator.

Also, our website offers a forum where users can communicate and post memes for the world to see. So get ready to compete for prizes in BNB using Pepe CEO by being innovative!

## Project Engagement

During the 25<sup>th</sup> of May 2023, Pepe CEO team engaged Solidproof.io to audit the smart contracts that they created. The engagement was technical in nature and focused on identifying the security flaws in the design and implementation of the contracts. They provided Solidproof.io with access to their code repository and whitepaper.

## Logo



## Contract Links

v1.0

<https://bscscan.com/token/0xadfaed188291ae479f458ee2d3f88fe7e8cc712f#readContract>

# 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.

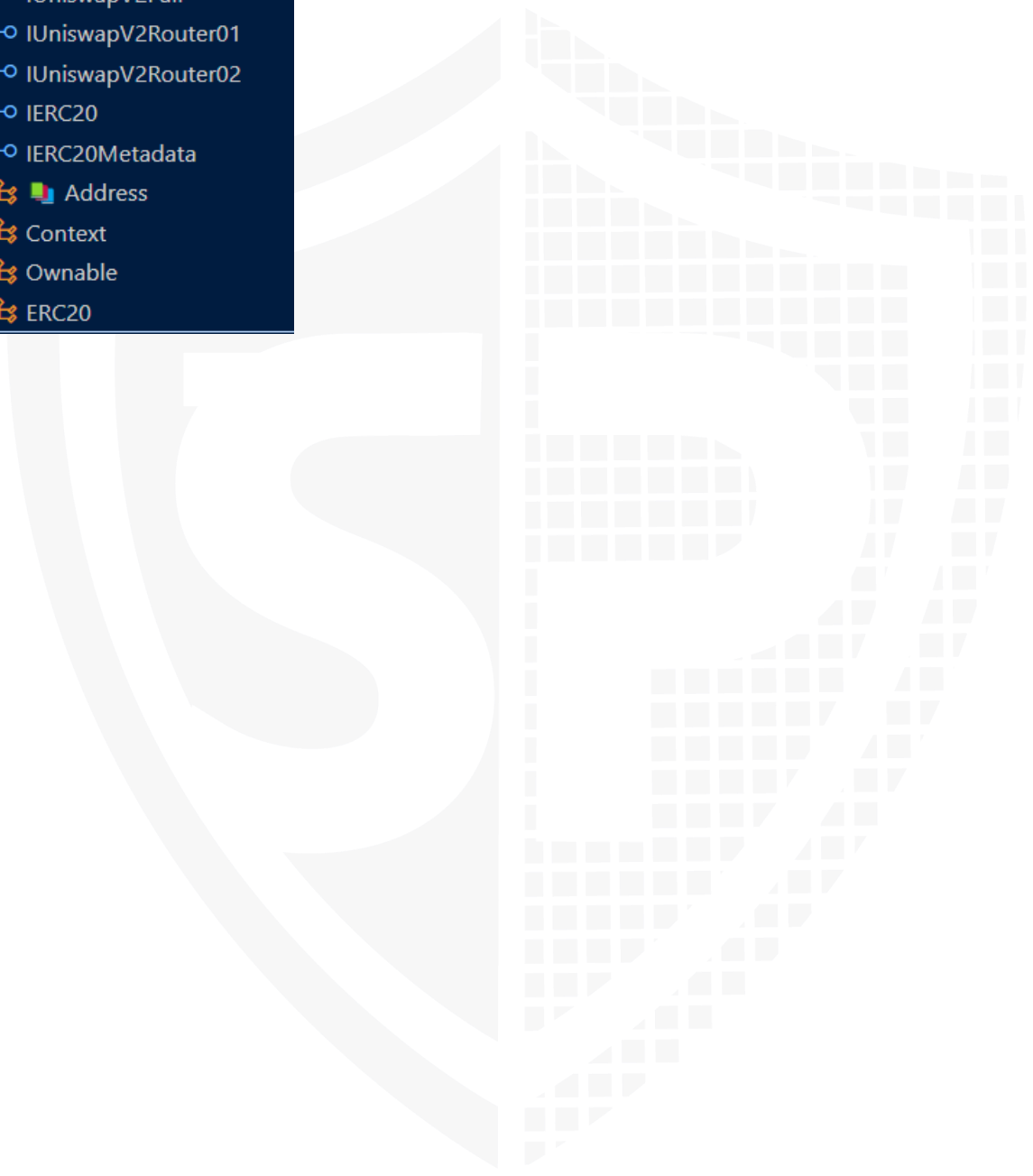
## Methodology

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 analyzing 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:



```
> IUniswapV2Factory
> IUniswapV2Pair
> IUniswapV2Router01
> IUniswapV2Router02
> IERC20
> IERC20Metadata
> Address
> Context
> Ownable
> ERC20
```



## Tested Contract Files

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

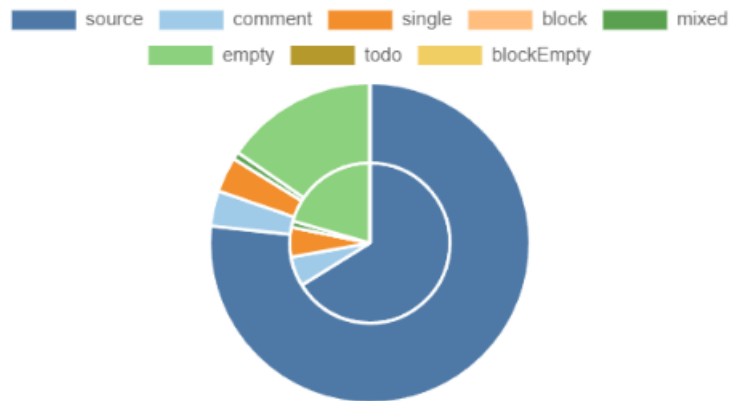
A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

### v1.0

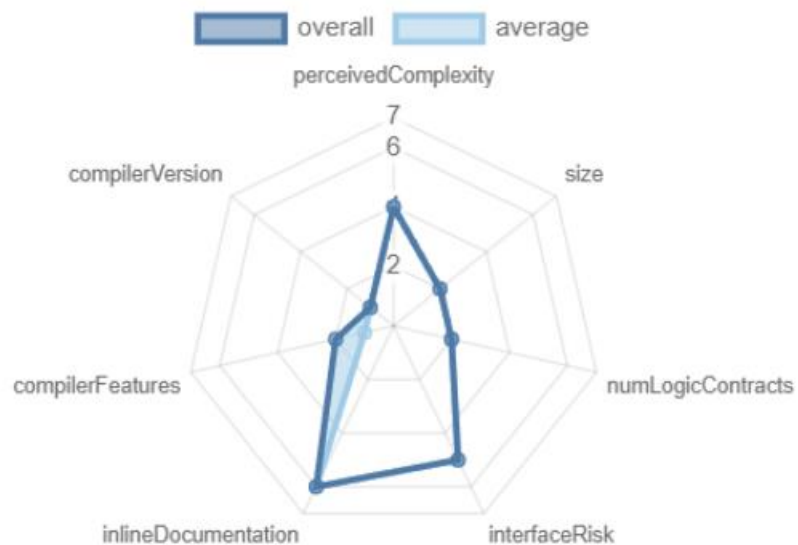
File Name	SHA-1 Hash
pepeceo.sol	f8c1e34ed5f44e99b06dfc63b692b22ebdf33256

# Metrics

## Source Lines v1.0



## Risk Level v1.0



# Capabilities

## Components

 Contracts	 Libraries	 Interfaces	 Abstract
2	1	6	2


### Exposed Functions

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.





 Public	 Payable
87	5







External	Internal	Private	Pure	View
72	77	1	12	39


### StateVariables

Total	 Public
11	4

### Capabilities

Solidity Versions observed	 Experimental Features	 Can Receive Funds	 Uses Assembly	 Has Destroyable Contracts
0.8.17		yes	yes (1 asm blocks)	

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

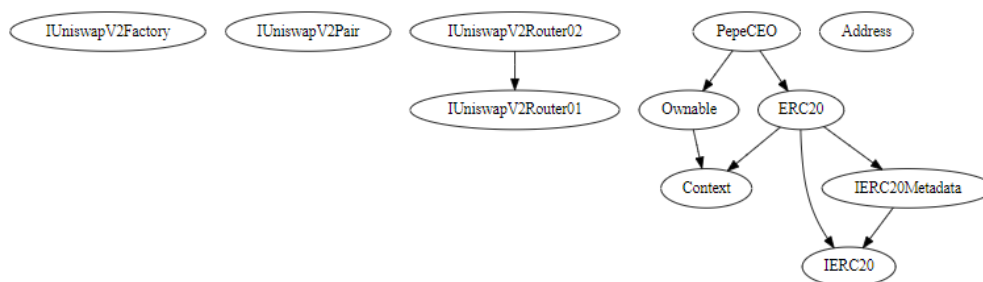
 TryCatch	$\Sigma$ Unchecked
	yes

## Capabilities

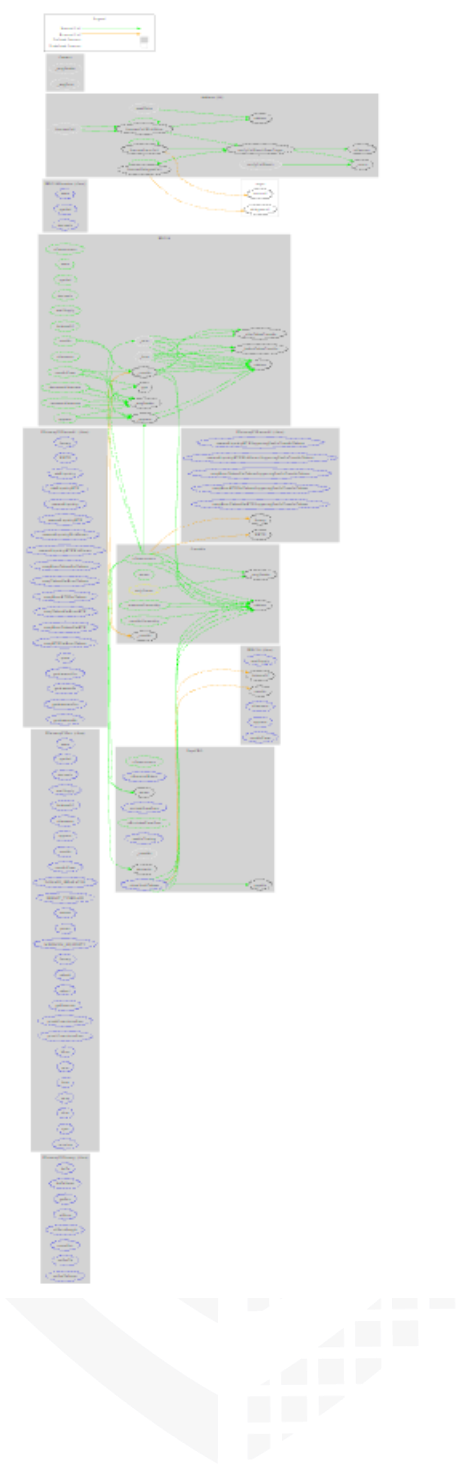
Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	0.8.17		Yes	Yes(1 asm blocks)	

Version	Transfers ETH	Low-Level Calls	Delegate Call	Uses Hash Functions	EC Recover	New/Create/Create2
1.0	Yes		Yes			

## Inheritance Graph v1.0



# Call 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. Is contract an upgradeable
2. Correct implementation of Token standard
3. Deployer cannot mint any new tokens
4. Deployer cannot burn or lock user funds
5. Deployer cannot pause the contract
6. Overall checkup (Smart Contract Security)

Is contract an upgradeable

Name	
Is contract an upgradeable?	No



## Correct implementation of Token standard

ERC20				
Function	Description	Exist	Tested	Verified
<b>totalSupply</b>	<b>Provides information about the total token supply</b>	<b>Yes</b>	<b>Yes</b>	<b>Yes</b>
<b>balanceOf</b>	<b>Provides account balance of the owner's account</b>	<b>Yes</b>	<b>Yes</b>	<b>Yes</b>
<b>transfer</b>	<b>Executes transfers of a specified number of tokens to a specified address</b>	<b>Yes</b>	<b>Yes</b>	<b>Yes</b>
<b>transferFrom</b>	<b>Executes transfers of a specified number of tokens from a specified address</b>	<b>Yes</b>	<b>Yes</b>	<b>Yes</b>
<b>approve</b>	<b>Allow a spender to withdraw a set number of tokens from a specified account</b>	<b>Yes</b>	<b>Yes</b>	<b>Yes</b>
<b>allowance</b>	<b>Returns a set number of tokens from a spender to the owner</b>	<b>Yes</b>	<b>Yes</b>	<b>Yes</b>



## Deployer cannot mint any new tokens

Name	Exist	Tested	Status
Deployer cannot mint	N/A	N/A	N/A
Max / Total Supply and last Token ID	N/A		



## Deployer cannot burn or lock user funds

Name	Exist	Tested	Status
Deployer cannot lock	N/A	N/A	N/A
Deployer cannot burn	N/A	N/A	N/A



## Deployer cannot pause the contract

Name	Exist	Tested	Status
Deployer cannot pause	N/A	N/A	N/A



# Overall checkup (Smart Contract Security)

Tested	Verified

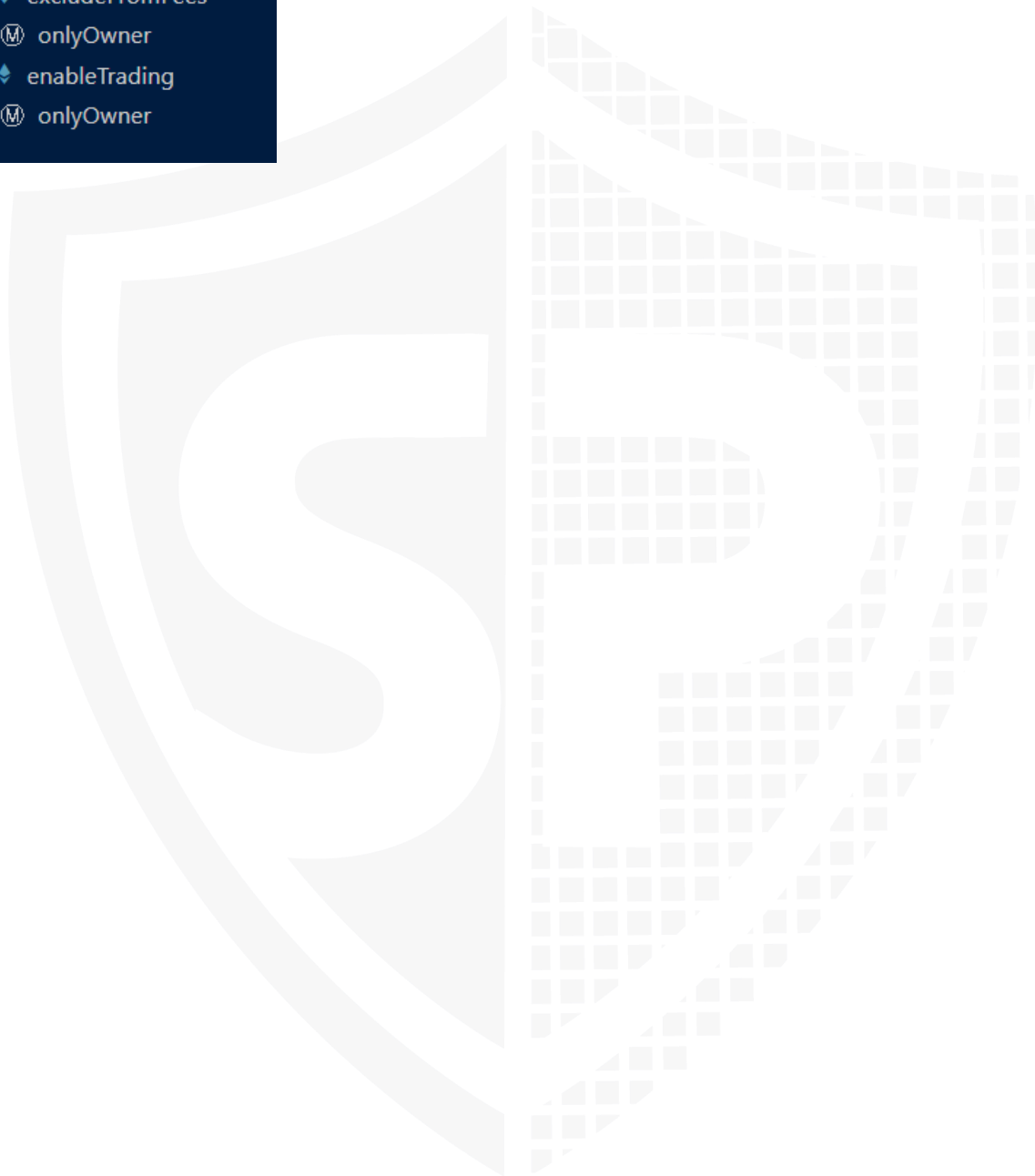
## Legend

Attribute	Symbol
Verified / Checked	
Partly Verified	
Unverified / Not checked	
Not available	

## Modifiers and public functions

### v1.0





	◆	<Constructor>	👛
✓	◆	claimStuckTokens	
	Ⓜ	onlyOwner	
✓	◆	excludeFromFees	
	Ⓜ	onlyOwner	
✓	◆	enableTrading	
	Ⓜ	onlyOwner	



## Ownership Privileges:

- Owner can claim stuck tokens.
- Owner can whitelist the addresses.
- Owner can enable trading only once.

## Source Units in Scope v1.0

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
	pepeceo.sol	5	6	640	365	264	24	370	
	Totals	5	6	640	365	264	24	370	

## 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**

**No Low issues**

## Informational issues

Issue	File	Type	Line	Description
#1	Pepeceo.sol	Natspec documentation missing	-	If you started to comment on your code, also comment on all other functions, variables, etc.

## 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.

## SWC Attacks

ID	Title	Relationships	Status
<a href="#">SWC-136</a>	Unencrypted Private Data On-Chain	<a href="#">CWE-767: Access to Critical Private Variable via Public Method</a>	PASSED
<a href="#">SWC-135</a>	Code With No Effects	<a href="#">CWE-1164: Irrelevant Code</a>	PASSED
<a href="#">SWC-134</a>	Message call with hardcode	<a href="#">CWE-655: Improper Initialization</a>	PASSED



	d gas amount		
<a href="#"><u>SWC-133</u></a>	Hash Collisions With Multiple Variable Length Arguments	<a href="#"><u>CWE-294: Authentication Bypass by Capture-replay</u></a>	PASSED
<a href="#"><u>SWC-132</u></a>	Unexpected Ether balance	<a href="#"><u>CWE-667: Improper Locking</u></a>	PASSED
<a href="#"><u>SWC-131</u></a>	Presence of unused variables	<a href="#"><u>CWE-1164: Irrelevant Code</u></a>	PASSED
<a href="#"><u>SWC-130</u></a>	Right-To-Left-Override control character (U+202E)	<a href="#"><u>CWE-451: User Interface (UI) Misrepresentation of Critical Information</u></a>	PASSED
<a href="#"><u>SWC-129</u></a>	Typographical Error	<a href="#"><u>CWE-480: Use of Incorrect Operator</u></a>	PASSED
<a href="#"><u>SWC-128</u></a>	DoS With Block Gas Limit	<a href="#"><u>CWE-400: Uncontrolled Resource Consumption</u></a>	PASSED
<a href="#"><u>SWC-127</u></a>	Arbitrary Jump with Function	<a href="#"><u>CWE-695: Use of Low-Level Functionality</u></a>	PASSED

	Type Variable		
<a href="#"><u>SWC-125</u></a>	Incorrect Inheritance Order	<a href="#"><u>CWE-696: Incorrect Behavior Order</u></a>	PASSED
<a href="#"><u>SWC-124</u></a>	Write to Arbitrary Storage Location	<a href="#"><u>CWE-123: Write-what-where Condition</u></a>	PASSED
<a href="#"><u>SWC-123</u></a>	Requirement Violation	<a href="#"><u>CWE-573: Improper Following of Specification by Caller</u></a>	PASSED
<a href="#"><u>SWC-122</u></a>	Lack of Proper Signature Verification	<a href="#"><u>CWE-345: Insufficient Verification of Data Authenticity</u></a>	PASSED
<a href="#"><u>SWC-121</u></a>	Missing Protection against Signature Replay Attacks	<a href="#"><u>CWE-347: Improper Verification of Cryptographic Signature</u></a>	PASSED
<a href="#"><u>SWC-120</u></a>	Weak Sources of Randomness from Chain Attributes	<a href="#"><u>CWE-330: Use of Insufficiently Random Values</u></a>	PASSED

<a href="#"><u>SWC-119</u></a>	Shadowing State Variables	<a href="#"><u>CWE-710: Improper Adherence to Coding Standards</u></a>	PASSED
<a href="#"><u>SWC-118</u></a>	Incorrect Constructor Name	<a href="#"><u>CWE-665: Improper Initialization</u></a>	PASSED
<a href="#"><u>SWC-117</u></a>	Signature Malleability	<a href="#"><u>CWE-347: Improper Verification of Cryptographic Signature</u></a>	PASSED
<a href="#"><u>SWC-116</u></a>	Timestamp Dependence	<a href="#"><u>CWE-829: Inclusion of Functionality from Untrusted Control Sphere</u></a>	PASSED
<a href="#"><u>SWC-115</u></a>	Authorization through tx.origin	<a href="#"><u>CWE-477: Use of Obsolete Function</u></a>	PASSED
<a href="#"><u>SWC-114</u></a>	Transaction Order Dependence	<a href="#"><u>CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')</u></a>	PASSED
<a href="#"><u>SWC-113</u></a>	DoS with Failed Call	<a href="#"><u>CWE-703: Improper Check or Handling of Exceptional Conditions</u></a>	PASSED
<a href="#"><u>SWC-112</u></a>	Delegate call to Untrusted Callee	<a href="#"><u>CWE-829: Inclusion of Functionality from Untrusted Control Sphere</u></a>	PASSED

<a href="#"><u>SWC-111</u></a>	Use of Deprecat ed Solidity Function s	<a href="#"><u>CWE-477: Use of Obsolete Function</u></a>	PASSED
<a href="#"><u>SWC-110</u></a>	Assert Violation	<a href="#"><u>CWE-670: Always-Incorrect Control Flow Implementation</u></a>	PASSED
<a href="#"><u>SWC-109</u></a>	Uninitiali zed Storage Pointer	<a href="#"><u>CWE-824: Access of Uninitialized Pointer</u></a>	PASSED
<a href="#"><u>SWC-108</u></a>	State Variable Default Visibility	<a href="#"><u>CWE-710: Improper Adherence to Coding Standards</u></a>	PASSED
<a href="#"><u>SWC-107</u></a>	Reentran cy	<a href="#"><u>CWE-841: Improper Enforcement of Behavioral Workflow</u></a>	PASSED
<a href="#"><u>SWC-106</u></a>	Unprot ec ted SELFDES TRUCT Instructio n	<a href="#"><u>CWE-284: Improper Access Control</u></a>	PASSED
<a href="#"><u>SWC-105</u></a>	Unprot ec ted Ether Withdra wal	<a href="#"><u>CWE-284: Improper Access Control</u></a>	PASSED
<a href="#"><u>SWC-104</u></a>	Uncheck ed Call Return Value	<a href="#"><u>CWE-252: Unchecked Return Value</u></a>	PASSED

<a href="#"><u>SWC-103</u></a>	<b>Floating Pragma</b>	<a href="#"><u>CWE-664: Improper Control of a Resource Through its Lifetime</u></a>	<b>PASSED</b>
<a href="#"><u>SWC-102</u></a>	<b>Outdated Compiler Version</b>	<a href="#"><u>CWE-937: Using Components with Known Vulnerabilities</u></a>	<b>PASSED</b>
<a href="#"><u>SWC-101</u></a>	<b>Integer Overflow and Underflow</b>	<a href="#"><u>CWE-682: Incorrect Calculation</u></a>	<b>PASSED</b>
<a href="#"><u>SWC-100</u></a>	<b>Function Default Visibility</b>	<a href="#"><u>CWE-710: Improper Adherence to Coding Standards</u></a>	<b>PASSED</b>



[SolidProof\\_io](https://twitter.com/SolidProof_io)



[@solidproof\\_io](https://t.me/solidproof_io)

*Solid  
Proofed*

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

  
MADE IN GERMANY