



SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

Audit

Security Assessment
23. December, 2021

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
Scope of Work	12
Inheritance Graph	12
Verify Claims	13
Modifiers	19
CallGraph	20
Source Units in Scope	21
Critical issues	22
High issues	22
Medium issues	22
Low issues	22
Informational issues	23
Commented Code exist	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 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	23. December 2021	<ul style="list-style-type: none">• Layout project• Automated- /Manual-Security Testing• Summary

Network

Binance Smart Chain (BEP20)

Website

<https://www.wonderworldtoken.io/>

Telegram

https://t.me/Wonder_world_1

Twitter

https://twitter.com/wonderworld_bsc

Reddit

[https://www.reddit.com/u/mdidje?
utm_medium=android_app&utm_source=share](https://www.reddit.com/u/mdidje?utm_medium=android_app&utm_source=share)

Youtube

<https://youtu.be/mAES0N9eWII>

Description

The magical world of the Middle Ages re-divided the magical world into kingdoms and territories due to a wonderful magical power. The order of Wonderworld will be re-established. The kings from the Austin Kingdom and the knights of multiple kingdoms continue to fight, the wonderful world A lot of new warriors and missions have been dispatched. This will be a new kingdom war. Warriors from the elves, dark and knights from the meta-enchantment will participate in the formulation of the order of this wonderful world and the battle for kingdom territories. , The order of civilization will be defined. The mysterious power of Wonderworld will give this universe a brand new civilization. Players will start the journey of exploration as the warriors of Wonderworld. In the end, the king and the strongest knights will have the decision-making power of Wonderworld and participate in the new rules. The Warriors will hold Durandall's Sword and start a new journey.

Project Engagement

During the 23th of December 2021, **WonderWorld 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/address/0x0b5cad3cdda7d242ce16ddf81a555d5f8e3a6054#code>

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)
Critical	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.
High	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.
Medium	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.
Low	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.
Informational	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:

```
./IBEP20.sol  
./SafeMath.sol  
./Context.sol  
./Ownable.sol  
./IPancakeswapV2Factory.sol  
./IPancakeswapV2Router02.sol
```


Tested Contract Files

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

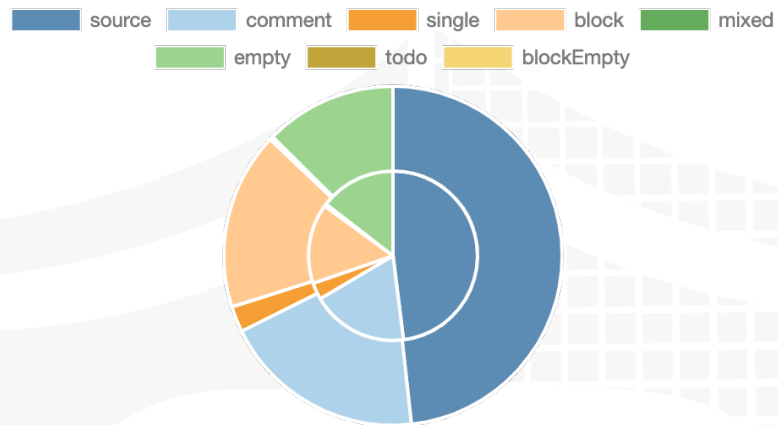
A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

v1.0

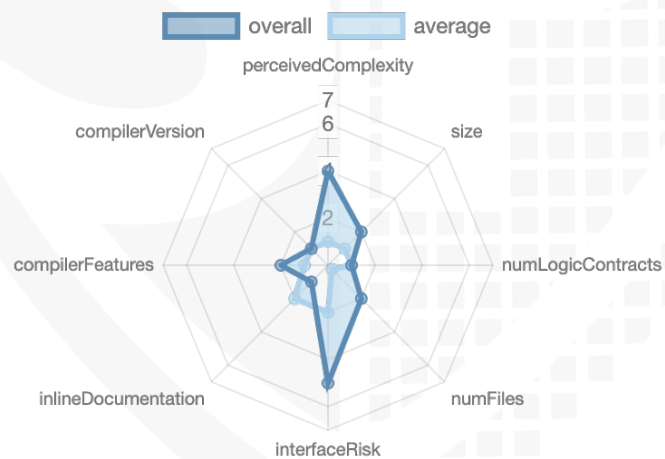
File Name	SHA-1 Hash
contracts/IBEP20.sol	7f7a828b01cf8ff021acddf7bb792da74bfc85c4
contracts/IPancakeswapV2Factory.sol	044b54346092fe7c8e60ef1ebb9025a11f6c3f53
contracts/Context.sol	7102228ad4154b4b4a06c0d974d27fe7d8c5e4d4
contracts/IPancakeswapV2Router01.sol	ed78bcd5e8c13308357b55ae8d9a242ce47dc2c5
contracts/IPancakeswapV2Router02.sol	cf9bd7cb2dbfb13fdd27f0f7954b641189b28017
contracts/SafeMath.sol	2e9d55e2f6fb89519fd7a96bdbac03da45444764
contracts/Ownable.sol	6224e805b23308915f1638419a19cc097b7a55bd
contracts/WWOD.sol	b47464505064b864948cd176bc693031f204801d

Metrics

Source Lines v1.0



Risk Level v1.0



Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	1	1	4	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.

Version	Public	Payable
1.0	72	5

Version	External	Internal	Private	Pure	View
1.0	59	63	10	18	28

State Variables

Version	Total	Public
1.0	31	14

Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	^0.8.0		yes		

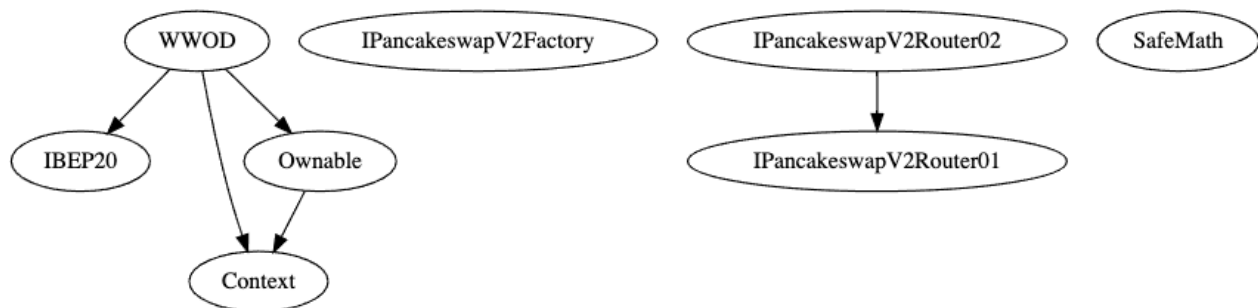
Scope of Work

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)

Inheritance Graph v1.0



Verify Claims

Correct implementation of Token standard

Tested	Verified
✓	✓

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

1. approve

2. decreaseAllowance

3. excludeFromFee

4. includeInFee

5. increaseAllowance

6. lock

7. renounceOwnership

8. setAddr

9. setCoolDownTime

10. setFees

11. setGuard

12. setMaxTxAmount

13. setNumTokensToSwap

14. setSwapAndLiquifyEnabled

15. setTreasury

16. transfer

17. transferFrom

18. transferOwnership

19. unlock

Deployer cannot mint any new tokens

Name	Exist	Tested	Verified
Deployer cannot mint	✓	✓	✓

Max / Total Supply: 1.000.000.000



Deployer cannot burn or lock user funds

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

Comments:

v1.0

- Deployer can lock user funds by setting `_maxTxAmount` to 0

Deployer cannot pause the contract

Name	Exist	Tested	Verified
Deployer cannot pause	✓	—	—



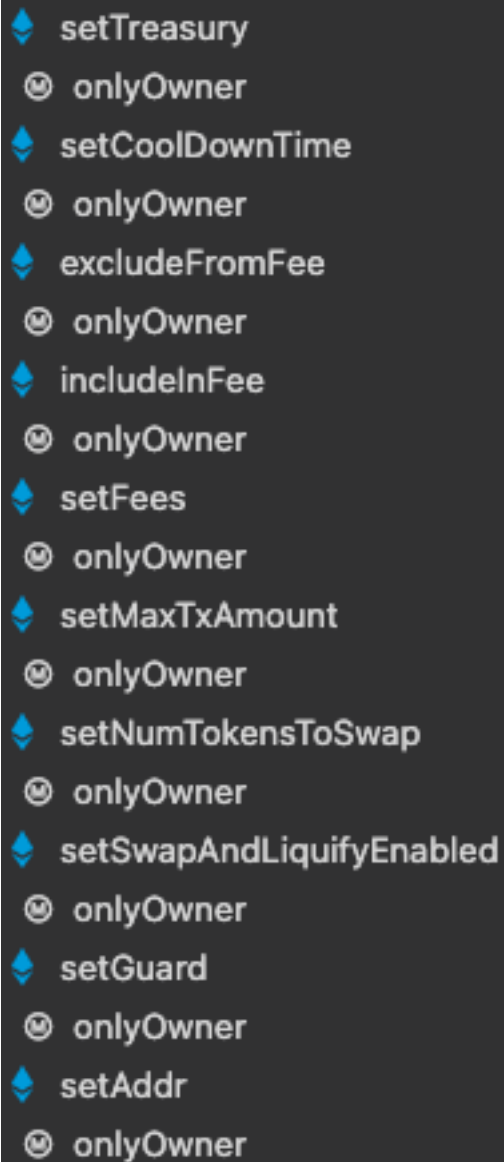
Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

Legend

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

Modifiers



- setTreasury
- onlyOwner
- setCoolDownTime
- onlyOwner
- excludeFromFee
- onlyOwner
- includeInFee
- onlyOwner
- setFees
- onlyOwner
- setMaxTxAmount
- onlyOwner
- setNumTokensToSwap
- onlyOwner
- setSwapAndLiquifyEnabled
- onlyOwner
- setGuard
- onlyOwner
- setAddr
- onlyOwner















Comments

- Everything can be set without any limitations

CallGraph

Source Units in Scope

v1.0

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
	contracts/IBEP20.sol	_____	1	96	25	17	69	21	_____
	contracts/IPancakeswapV2Factory.sol	_____	1	20	9	4	1	17	_____
	contracts/Context.sol	1	_____	22	22	10	12	1	_____
	contracts/IPancakeswapV2Router01.sol	_____	1	97	6	3	1	48	
	contracts/IPancakeswapV2Router02.sol	_____	1	46	8	4	1	16	
	contracts/SafeMath.sol	1	_____	217	217	69	134	10	
	contracts/Ownable.sol	1	_____	55	55	45	1	38	_____
	contracts/WWOD.sol	1	_____	416	412	297	16	213	
	Totals	4	4	969	754	449	235	364	

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 found -

High issues

- no high issues found -

Medium issues

- no medium issues found -

Issue	File	Type	Line	Description
#1	Main	Reentrancy vulnerabilities		Apply the [`check-effects-interactions pattern`](http://solidity.readthedocs.io/en/v0.4.21/security-considerations.html#re-entrancy).or nonReentrant modifier from OpenZeppelin

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.“.
#3	Main	Missing Zero Address Validation (missing-zero-check)	92, 257	Check that the address is not zero
#4	Main	State variable visibility is not set	17, 43	It is best practice to set the visibility of state variables explicitly

#5	Main	Local variables shadowing	294, 155	Change owner to owner_ Rename the local variables that shadow another component
	Ownable	Previous owner can get back the ownership	42	Fix getting ownership back by previous owner after renounced ownership If the owner lock once _previousOwner will be set. After renouncing/transferring ownership the _previousOwner can call unlock function to get back the ownership

Informational issues

Issue	File	Type	Line	Description
#1	Main	State variables that could be declared constant (constable-states)	53, 50, 52, 47	Add the `constant` attributes to state variables that never change
#2	Main	Unused return values	393	Ensure that all the return values of the function calls are used and handle both success and failure cases if needed by the business logic
#3	Main	Functions that are not used	388	Remove unused functions
#4	Main	Unused state variables	48, 67	Remove unused state variables
	Main	Require error message is missing	-	Provide an error message to any require statements

Commented Code exist

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

Line	Comment
94	//IPancakeswapV2Router02 _pancakeswapV2Router = IPancakeswapV2Router02(0x9Ac64Cc6e4415144C455BD8E4837Fea55603e5c3);

Recommendation

Remove the commented code, or address them properly.

Audit Comments

23. December 2021:

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



SWC Attacks

ID	Title	Relationships	Status
SW C-1 36	Unencrypted Private Data On-Chain	CWE-767: Access to Critical Private Variable via Public Method	PASSED
SW C-1 35	Code With No Effects	CWE-1164: Irrelevant Code	PASSED
SW C-1 34	Message call with hardcoded gas amount	CWE-655: Improper Initialization	PASSED
SW C-1 33	Hash Collisions With Multiple Variable Length Arguments	CWE-294: Authentication Bypass by Capture-replay	PASSED
SW C-1 32	Unexpected Ether balance	CWE-667: Improper Locking	PASSED
SW C-1 31	Presence of unused variables	CWE-1164: Irrelevant Code	NOT PASSED
SW C-1 30	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	PASSED
SW C-1 29	Typographical Error	CWE-480: Use of Incorrect Operator	PASSED
SW C-1 28	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	PASSED

SW C-1 27	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	PASSED
SW C-1 25	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	PASSED
SW C-1 24	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	PASSED
SW C-1 23	Requirement Violation	CWE-573: Improper Following of Specification by Caller	PASSED
SW C-1 22	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	PASSED
SW C-1 21	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	PASSED
SW C-1 20	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	PASSED
SW C-11 9	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	PASSED
SW C-11 8	Incorrect Constructor Name	CWE-665: Improper Initialization	PASSED
SW C-11 7	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	PASSED

SW C-11 6	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SW C-11 5	Authorization through tx.origin	CWE-477: Use of Obsolete Function	PASSED
SW C-11 4	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	PASSED
SW C-11 3	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	PASSED
SW C-11 2	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SW C-11 1	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	PASSED
SW C-11 0	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	PASSED
SW C-1 09	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	PASSED
SW C-1 08	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	NOT PASSED
SW C-1 07	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	PASSED
SW C-1 06	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	PASSED

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

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