SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

WinMiner

Audit

Security Assessment

25.July,2022

For

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	21.July,2022	• Layout project • Automated- /Manual-Security Testing • Summary

Network

Binance (BSC)

Website

https://winbinary.net/

Twitter

https://twitter.com/Win_Binary01

Telegram

https://t.me/WinBinary01

Discord

https://discord.com/invite/DVkbScDM

Description

WinBinary is a Binance Smart Chain Dividend Token protocol that rewards BUSD to its holders. The BUSD rewards are reflected from 10% buy and sell tax transactions and weekly manual reward pooled from all the listed utilities development revenue 50% share.
Diverse Ecosystems Win Binary is an ecosystem with 5 different features where the center is our token $WINB. We provide different features using a huge amount of data in our database, where players and users can be rewarded $WINB with a compounding interest, from betting activities to trading options and cryptos.

Project Engagement

During the 21th of July 2022, **Win Miner** 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/address/0x2F3d450075E77c4099B1e02992FA6E3E76Ca6c35#code

https://bscscan.com/address/0xc3565A272EbdE9D7dBF5ca5a60CE1038C9890702#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.

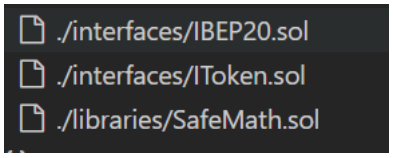
2. Testing and automated analysis that includes the following:
    i) Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
    ii) Symbolic execution, which is analyzing a program 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:



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.

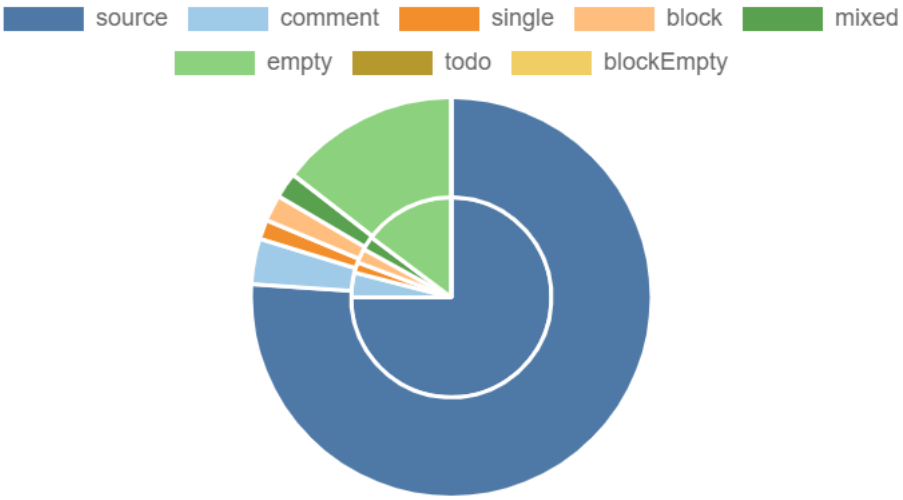
v1.0

File Name	SHA-1 Hash
contracts/interfaces/IBEP20.sol	55aeaad24307d5746e53875778bac834060d6bb4
contracts/interfaces/IToken.sol	2425cd98cb852897690be6a65b0b004a33305859
contracts/WinMinerWINB.sol	40cd25bbb5a146f79d2ecf30b806cffd74cf08c1
contracts/libraries/SafeMath.sol	58a7153fc9024246014ee35299b08170fbe51f38
contracts/WinMiner.sol	9d4af77bc89384cf7ddb011037e12290b0f87c6c

Metrics

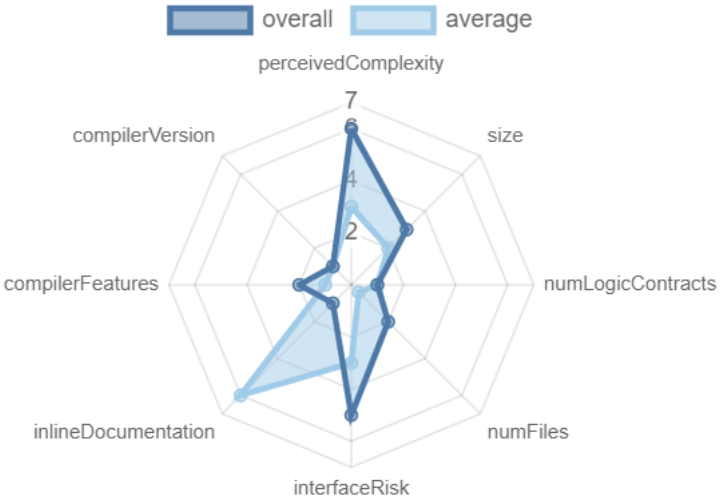
Source Lines

v1.0



Risk Level

v1.0



Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	2	1	2	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	122	0

Version	External	Internal	Private	Pure	View
1.0	72	95	2	7	54

State Variables

Version	Total	Public
1.0	108	72

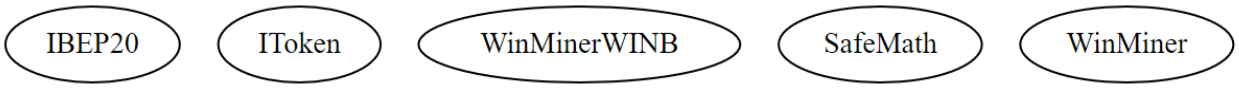
Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	`0.8.14`			Yes	

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

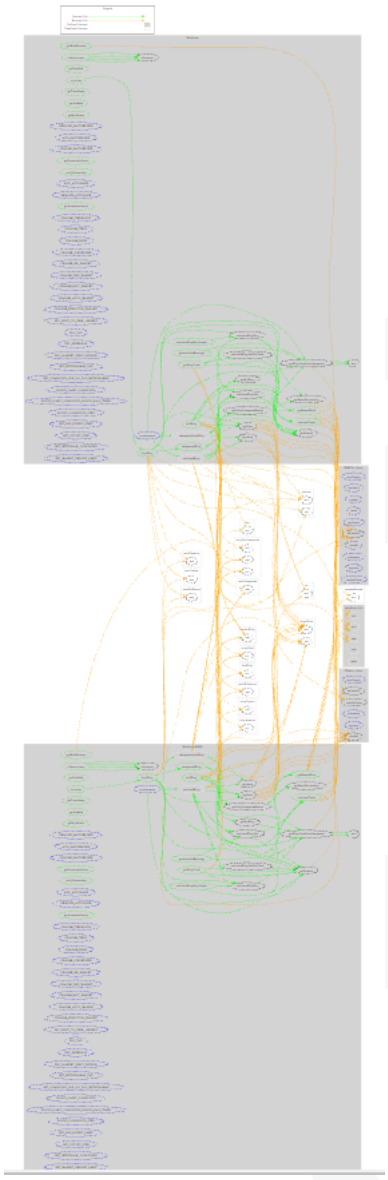
Inheritance Graph

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**
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 contracts

v1.0

1. ADD_AUTOMATE	12. CHANGE_PARTNERSHIP
2. ADD_PARTNERSHIP	13. CHANGE_TIERBONUS
3. BONUS_COMPOUND_STEP	14. CHANGE_TIERS
4. BONUS_DAILY_COMPOUND	15. CHANGE_WINB
5. BONUS_DAILY_COMPOUND_BONUS_MAX_TIMES	16. PRC_MARKET_WINY_DIVISOR
6. CHANGE_AUTO_WALLET	17. PRC_REFERRAL
7. CHANGE_BB_WALLET	18. PRC_TAX
8. CHANGE_DEV_WALLET	19. PRC_WINY_TO_HIRE_1MINERS
9. CHANGE_EXECUTOR_WALLET	20. REMOVE_AUTOMATE
10. CHANGE_MKT_WALLET	21. REMOVE_PARTNERSHIP
11. CHANGE_OWNERSHIP	22. SET_COMPOUND_FOR_NO_TAX_WITHDRAWAL

23. SET_CUTOFF_STEP

24. SET_MIN_INVEST_LIMIT

25. SET_WALLET_DEPOSIT_LIMIT

26. SET_WITHDRAWAL_TAX

27. SET_WITHDRAW_COOLDOWN

28. buyWiny

29. compoundWiny

30. initialize

31. runAutomate

32. sellWiny

Deployer cannot mint any new tokens

Name	Exist	Tested	Status
Deployer cannot mint			
Max / Total Supply	N/A		

Deployer cannot burn or lock user funds

Name	Exist	Tested	Status
Deployer cannot lock			
Deployer cannot burn			

Deployer cannot pause the contract

Name	Exist	Tested	Status
Deployer cannot pause			

Overall checkup (Smart Contract Security)

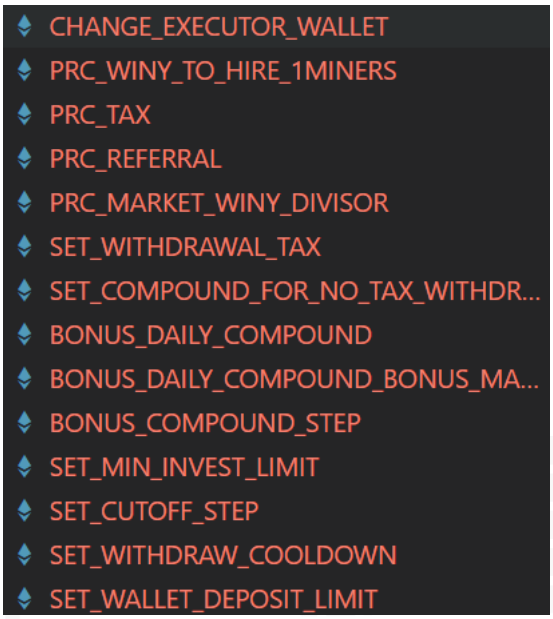
Tested	Verified

Legend

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

Modifiers and public functions

v1.0



Comments:

- Some of the functions of these two contracts use indirect use of authority. For example, the contract uses a “require” check to see if the caller of the function is the owner of the contract or not. All the functions in both the contract with error message that says “Admin use only” are the only owner functions.
- Owner can perform certain actions in the contract like, setting fee, adding/removing partnerships, etc.

Source Units in Scope

v1.0

File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score
contracts/interfaces/IBEP20.sol	———	1	40	5	3	1	21
contracts/interfaces/IToken.sol	———	1	32	5	3	1	13
contracts/WinMinerWINB.sol	1	———	858	784	654	31	425
contracts/libraries/SafeMath.sol	1	———	34	34	28	1	6
contracts/WinMiner.sol	1	———	683	681	549	32	425
Totals	**3**	**2**	**1647**	**1509**	**1237**	**66**	**890**

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
#2	WinMiner.sol	Missing Events	423, 432, 441, 486, 500, 546, 555, 564, 570, 575, 580, 585, 590, 595, 605, 611,620, 626, 633, 639, 644,	Emit an event for critical parameter changes. In this case, minting, burning of tokens, etc.

			650, 656, 661, 666, 671,677	
#2	WinMinerWINB.sol	Missing Events	In the same functions as #2	Emit an event for critical parameter changes. In this case, minting, burning of tokens, etc.
#3	WinMiner.sol	Missing zero check	110, 741, 746, 751, 756, 761,766	Check that the address is not zero
#4	WinMinerWINB.sol	Missing zero check	In the same functions as #2	Check that the address is not zero
#5	WinMiner.sol	Missing „“isContract“ check	575, 580, 585, 590	In the constructor there is a check to stop adding a contract address as fee addresses, bb address, and execute order address. But in the setter function there is no check that does the same. Thus, a contract address can be set in place of these contracts after deployment.
#6	WinMinerWINB.sol	Missing „“isContract“ check	In the same functions as #5	In the constructor there is a check to stop adding a contract address as fee addresses, bb address, and execute order address. But in the setter function there is no check that does the same. Thus, a contract address can be set in place of these contracts after deployment.
#7	WinMinerWINB.sol	Missing zero value check	802, 832,837, 842	These values could be set as zero which may lead to unexpected arithmetic errors
#8	WinMine.sol	Missing zero value check	In the same functions as #7	These values could be set as zero which may lead to unexpected arithmetic errors
#9	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

Informational issues

Issue	File	Type	Line	Description
#1	WinMiner.sol	Variables that can be constant	13, 18, 24, 26, 45, 46	These state variables are never changed and should be declared constant
#2	WinMiner WINB	Variables that can be constant	13, 19, 25, 27, 46, 47	These state variables are never changed and should be declared constant
#3	WinMiner.sol	State Variable Default Visibility	46,47	Labeling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable. Explicitly define visibility for all state variables.
#4	WinMiner WINB.sol	State Variable Default Visibility	45, 46	Labeling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable. Explicitly define visibility for all state variables.
#5	Main	NatSpec documentation missing	-	If you started to comment your code, also comment 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.

25.July,2022:

- There is still an owner (Owner still has not renounced ownership)
- The owner can set the other addresses used in the contract such as devAddress, marketing address, executor address, auto address, and bb address as a contract address which was prohibited in the contract's constructor but it can be done by calling the other setter functions. Moreover, the owner can

also set all of these address as the same one because there is no prevention against it

- Read the whole report and modifiers section for more information.

SWC Attacks

ID	Title	Relationships	Status
SWC-136	Unencrypted Private Data On-Chain	CWE-767: Access to Critical Private Variable via Public Method	PASSED
SWC-135	Code With No Effects	CWE-1164: Irrelevant Code	PASSED
SWC-134	Message call with hardcoded gas amount	CWE-655: Improper Initialization	PASSED
SWC-133	Hash Collisions With Multiple Variable Length Arguments	CWE-294: Authentication Bypass by Capture-replay	PASSED
SWC-132	Unexpected Ether balance	CWE-667: Improper Locking	PASSED
SWC-	Presence of unused variables	CWE-1164: Irrelevant Code	PASSED

131			PASSED
SWC-130	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	
SWC-129	Typographical Error	CWE-480: Use of Incorrect Operator	
SWC-128	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	
SWC-127	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	
SWC-125	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	
SWC-	Write to Arbitrary	CWE-123: Write-what-where Condition	

124	Storage Location		
SWC-123	Requirement Violation	CWE-573: Improper Following of Specification by Caller	PASSED
SWC-122	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	PASSED
SWC-121	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	PASSED
SWC-120	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	PASSED
SWC-119	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	PASSED

SWC-118	Incorrect Constructor Name	CWE-665: Improper Initialization	PASSED
SWC-117	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	PASSED
SWC-116	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SWC-115	Authorization through tx.origin	CWE-477: Use of Obsolete Function	PASSED
SWC-114	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	PASSED
SWC-113	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	PASSED

SWC-112	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SWC-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	PASSED
SWC-110	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	PASSED
SWC-109	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	PASSED
SWC-108	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	NOT PASSED
SWC-107	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	PASSED

SWC-106	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	PASSED
SWC-105	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	PASSED
SWC-104	Unchecked Call Return Value	CWE-252: Unchecked Return Value	PASSED
SWC-103	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	PASSED
SWC-102	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	PASSED
SWC-101	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	PASSED

SWC-100	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED