



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

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

MADE IN GERMANY

# Juni Audit

**Security Assessment**  
**31. August, 2022**

**For**



**SolidProof\_io**



**@solidproof\_io**

Disclaimer	3
Description	5
Project Engagement	5
Logo	5
Contract Link	5
Methodology	8
Used Code from other Frameworks/Smart Contracts (direct imports)	9
Tested Contract Files	10
Source Lines	11
Risk Level	11
Capabilities	12
Inheritance Graph	14
CallGraph	15
Scope of Work/Verify Claims	17
Modifiers and public functions	27
Source Units in Scope	31
Critical issues	33
High issues	33
Medium issues	33
Low issues	33
Informational issues	33
Audit Comments	33
SWC Attacks	35

# 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 Uniswap, 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	19. August 2022	<ul style="list-style-type: none"><li>• Layout project</li><li>• Automated- /Manual-Security Testing</li><li>• Summary</li></ul>
1.1	22. August 2022	<ul style="list-style-type: none"><li>• Reaudit</li></ul>
1.2	24. August 2022	<ul style="list-style-type: none"><li>• Mainnet address added</li></ul>
1.3	27. August 2022	<ul style="list-style-type: none"><li>• Audit changed code</li></ul>
1.4	31. August 2022	<ul style="list-style-type: none"><li>• Referral address function has been added</li></ul>

## **Network**

Binance Smart Chain (BEP20)

## **Website**

<https://www.juni.gg/about>

<https://juni.gg/>

## **Telegram**

<https://t.me/JackpotUniverse>

## **Twitter**

<https://twitter.com/JUNIBSC>

## **Facebook**

<https://www.facebook.com/JackpotUniverse-103182732442228>

## Description

JUNI is a **completely new approach** for jackpot projects on blockchain. A **never before seen** smart contract which not only **increases the price floor** for its token constantly, but also **rewards players** of the Jackpot Game with **huge prizes in BNB**.

**7 separate and unique smart contracts** are running simultaneously to give our holders and players a **seamless user experience** in the JUNI Ecosystem.

The JUNI team has proven itself as first movers inside the BSC community.

## Project Engagement

During the 18th of August 2022, **JUNI 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

### v1.1

- Provided as files

### v1.2

- <https://bscscan.com/address/0xB7755d3A6aCc671D941C83cbc9993Dc4C8F3B6E1>

### v1.3

- Provided as files

## v1.4

- <https://bscscan.com/address/0x95Aa33319698CF67C7AB33bb23A65E9d38397187#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)
<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:

Dependency / Import Path	Count
@openzeppelin/contracts/token/ERC20/IERC20.sol	3
@openzeppelin/contracts/utils/Address.sol	3
@openzeppelin/contracts/utils/structs/EnumerableSet.sol	2
@uniswap/v2-core/contracts/interfaces/IUniswapV2Factory.sol	1
@uniswap/v2-core/contracts/interfaces/IUniswapV2Pair.sol	1
@uniswap/v2-periphery/contracts/interfaces/IUniswapV2Router02.sol	2

## 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/Treasury.sol	22900d01f97428a23abbdd3aec894158e31ae7de
contracts/JackpotToken.sol	12ffa188c155c51d4dab2f194dc725f684ecdd82
contracts/IJackpotGuard.sol	6d5b444e9960514f5a4c3d364e6d42bf38deecdd
contracts/JackpotUniverse.sol	5394d93e2fc6257dc424e15c9ae2af47aab802a2
contracts/Ownable.sol	d454cd2693b50bf37253d76510e246fd9d7cffb0
contracts/IJackpotBroker.sol	bb39093e412b6a5d1d99d248c41cb65f90feef9c

### v1.3

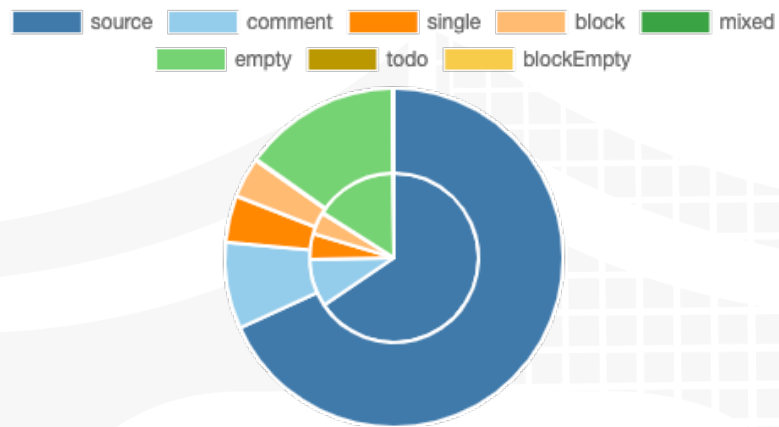
File Name	SHA-1 Hash
contracts/Treasury.sol	5c2fa224a7e3b0cf4f737d306dd4a171577a8bc6
contracts/JackpotToken.sol	eb47bc6d9f4444f589c11841f7d1279df919f290
contracts/IJackpotGuard.sol	4f1e5444f86169066ce7cfe8d89b881e91dca58a
contracts/JackpotUniverse.sol	22ec30273817b2154199de53573820be81cbf44b
contracts/IJackpotReferral.sol	e5c6f38287e8045a1afb12a6f772e34d90302ff7
contracts/Ownable.sol	ae53d6d899ce77ca582f108e5dde07cb9148b542
contracts/IJackpotBroker.sol	5f79cd457daf468646603e44e4b18c8325c6ef95

### v1.4

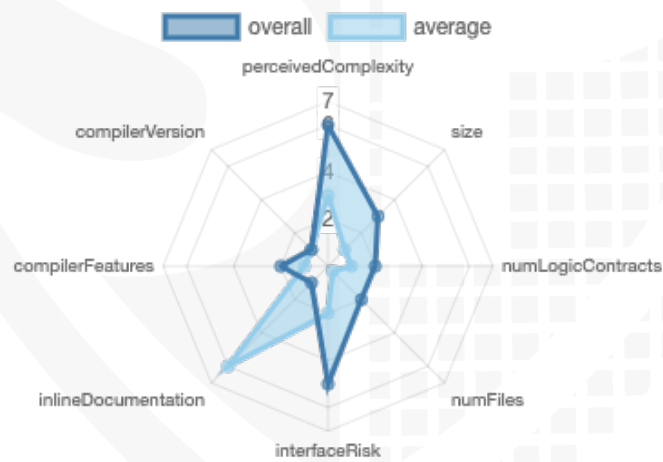
File Name	SHA-1 Hash
contracts/JackpotUniverse.sol	5ec237865fcb3cb2c0306d42cc9819723ec78c43

# Metrics

## Source Lines v1.4



## Risk Level v1.4



## Capabilities

### Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	2	0	2	3
1.3	2	0	3	3

### 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	99	5
1.3	98	4

Version	External	Internal	Private	Pure	View
1.0	46	103	12	6	37
1.3	46	103	12	6	37

## State Variables

Version	Total	Public
1.0	62	15
1.3	64	13

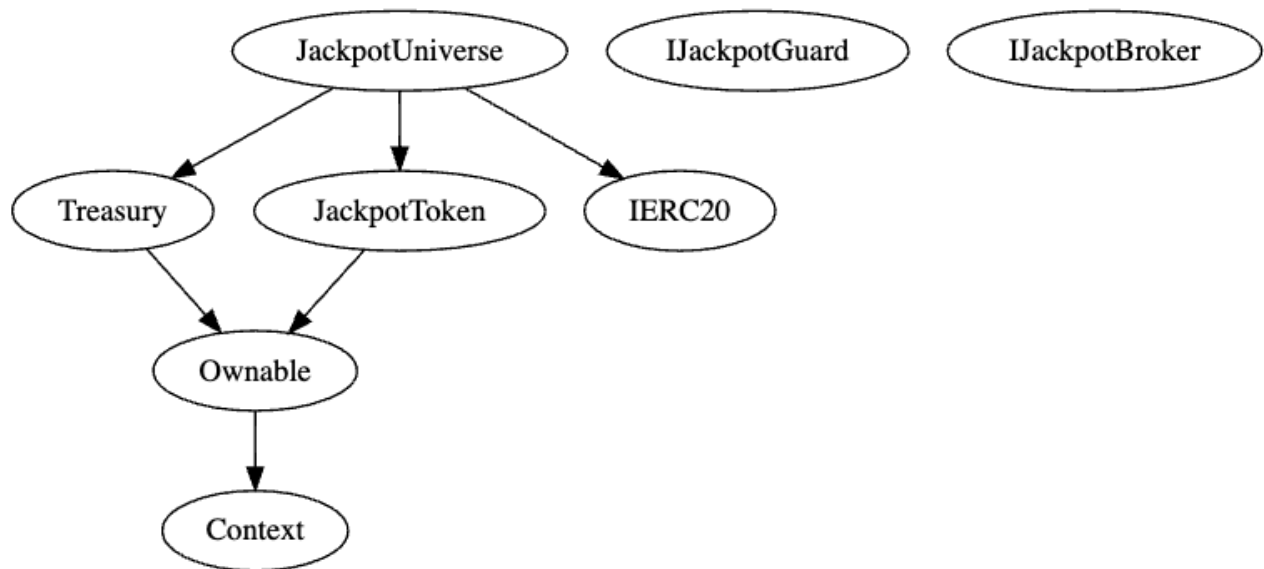
## Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	0.8.16		yes		
1.3	0.8.16		yes		

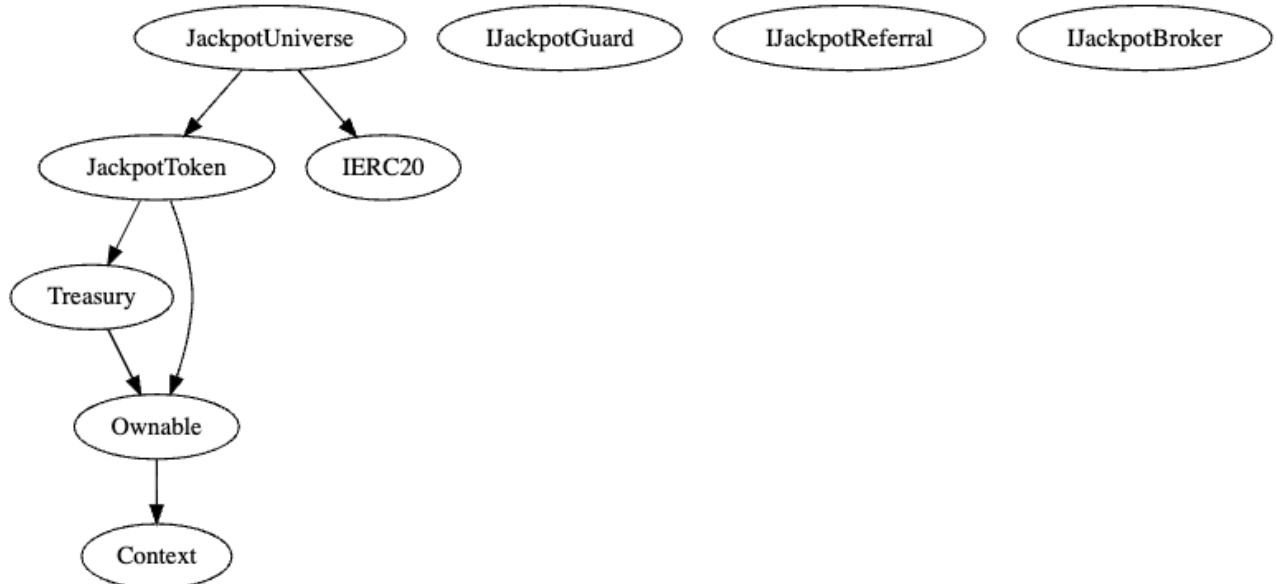
Version	Transfers ETH	Low-Level Calls	DelegateCall	Uses Hash Functions	EC Recover	New/Create/Create2
1.0	yes					
1.3	yes					

# Inheritance Graph

## v1.0

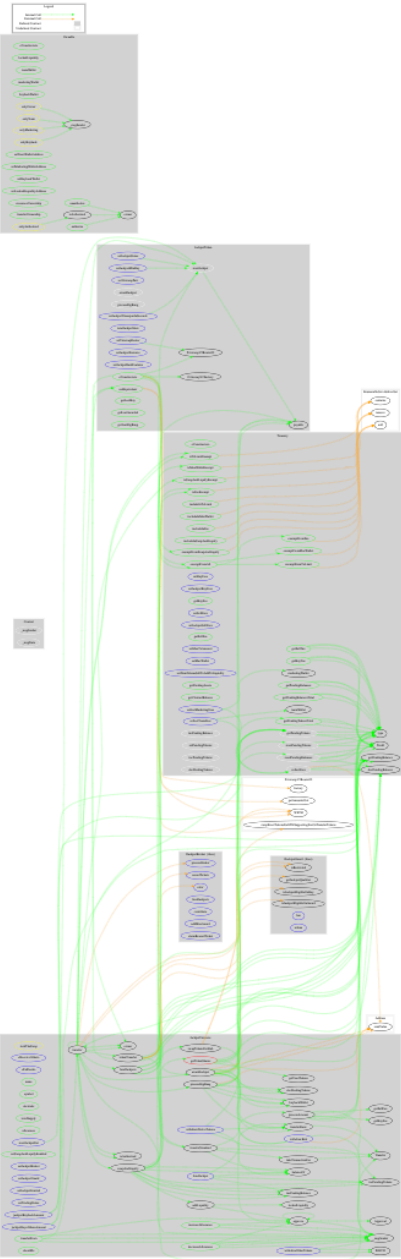


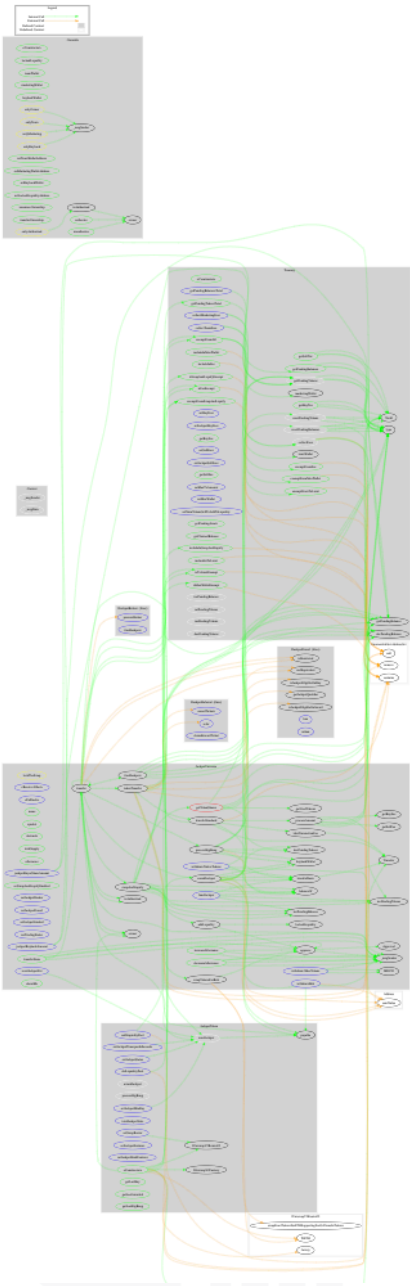
## v1.3



# CallGraph

## 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. Deployer cannot set fees
7. Deployer cannot blacklist/antisnipe addresses
8. 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 contract v1.0

approve	setJackpotStat...
authorize	setJackpotTim...
collectMarketi...	setLockedLiqui...
collectTeamFe...	setMarketing...
decreaseAllow...	setMaxTxAmo...
exemptFromAll	setMaxWallet
exemptFromFee	setNumTokens...
exemptFromM...	setSellFees
exemptFromS...	setSwapAndLi...
exemptFromTx...	setTeamWalle...
fundJackpot	setTradingStat...
includeInFee	setUniswapPair
includeInMax...	setUniswapRo...
includeInSwap...	transfer
includeInTxLimit	transferFrom
increaseAllow...	transferOwner...
renounceOwn...	unauthorize
resetJackpotExt	withdrawBnb
setBuybackW...	withdrawNati...
setBuyFees	withdrawOthe...
setJackpotBro...	
setJackpotBuy...	
setJackpotFeat...	
setJackpotGua...	
setJackpotHar...	
setJackpotLimi...	
setJackpotMin...	
setJackpotSel...	

## Deployer cannot mint any new tokens

Name	Exist	Tested	Status
Deployer cannot mint	—	—	—



## Deployer cannot burn or lock user funds

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

Comments:

**v1.0**

- Owner can lock user funds by
  - Setting tradingOpen to false

## Deployer cannot pause the contract

Name	Exist	Tested	Status
Deployer cannot pause	✓	✓	✗

Comments:

**v1.0**

- Owner can pause contract by setting tradingOpen to false



## Deployer cannot set fees

Name	Exist	Tested	Status
Deployer cannot set fees over 25%	✓	✓	✓
Deployer cannot set fees to nearly 100% or to 100%	✓	✓	✓





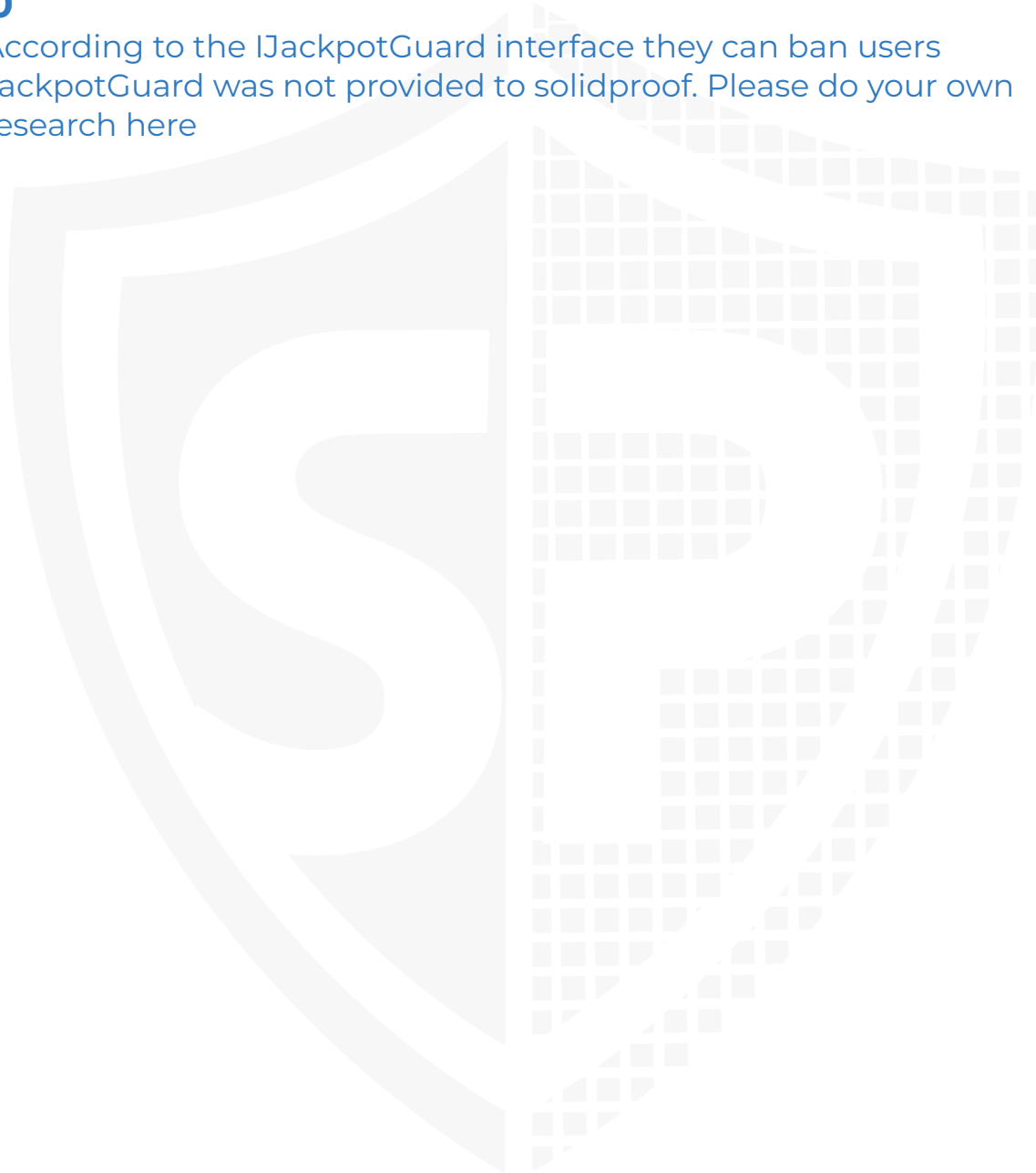
## Deployer can blacklist/antisnipe addresses

Name	Exist	Tested	Status
Deployer cannot blacklist/antisnipe addresses	✓	✓	✗

Comments:

### v1.0

- According to the IJackpotGuard interface they can ban users
- JackpotGuard was not provided to solidproof. Please do your own research here



## 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

### Treasury

✓	⚡	exemptFromTxLimit	
	🔒	onlyAuthorized	
✓	⚡	includeInTxLimit	
	🔒	onlyAuthorized	
✓	⚡	exemptFromMaxWallet	
	🔒	onlyAuthorized	
✓	⚡	includeInMaxWallet	
	🔒	onlyAuthorized	
✓	⚡	exemptFromFee	
	🔒	onlyAuthorized	
✓	⚡	includeInFee	
	🔒	onlyAuthorized	
✓	⚡	exemptFromSwapAndLiquify	
	🔒	onlyOwner	
✓	⚡	includeInSwapAndLiquify	
	🔒	onlyOwner	
✓	⚡	exemptFromAll	
	🔒	onlyOwner	
✓	⚡	setBuyFees	
	🔒	onlyAuthorized	
✓	⚡	setJackpotBuyFees	
	🔒	onlyAuthorized	
✓	⚡	setSellFees	
	🔒	onlyAuthorized	
✓	⚡	setJackpotSellFees	
	🔒	onlyAuthorized	
✓	⚡	setMaxTxAmount	
	🔒	onlyAuthorized	
✓	⚡	setMaxWallet	
	🔒	onlyAuthorized	
✓	⚡	setNumTokensSellToAddToLiquidity	
	🔒	onlyAuthorized	
✓	⚡	collectMarketingFees	
	🔒	onlyMarketing	
✓	⚡	collectTeamFees	
	🔒	onlyTeam	

### JackpotUniverse

	⚡	transfer	
	⚡	approve	
	⚡	transferFrom	
	⚡	increaseAllowance	
	⚡	decreaseAllowance	
✓	⚡	fundJackpot 💰	
	🔒	onlyAuthorized	
✓	⚡	setSwapAndLiquifyEnabled	
	🔒	onlyOwner	
✓	⚡	setJackpotBroker	
	🔒	onlyOwner	
✓	⚡	setJackpotGuard	
	🔒	onlyOwner	
✓	⚡	setJackpotLimited	
	🔒	onlyOwner	
✓	⚡	withdrawBnb	
	🔒	onlyOwner	
✓	⚡	withdrawNativeTokens	
	🔒	onlyOwner	
✓	⚡	withdrawOtherTokens	
	🔒	onlyOwner	
✓	⚡	setTradingStatus	
	🔒	onlyOwner	
✓	⚡	resetJackpotExt	
	🔒	onlyAuthorized	

## Ownable

```
✓ 🔹 setTeamWalletAddress
  | 🗑 onlyOwner
✓ 🔹 setMarketingWalletAddress
  | 🗑 onlyOwner
✓ 🔹 setBuybackWallet
  | 🗑 onlyOwner
✓ 🔹 setLockedLiquidityAddress
  | 🗑 onlyOwner
✓ 🔹 renounceOwnership
  | 🗑 onlyOwner
✓ 🔹 transferOwnership
  | 🗑 onlyOwner
✓ 🔹 authorize
  | 🗑 onlyOwner
✓ 🔹 unauthorize
  | 🗑 onlyOwner
```

## JackpotToken

```
✓ 🔹 setUniswapRouter
  | 🗑 onlyOwner
✓ 🔹 setUniswapPair
  | 🗑 onlyOwner
✓ 🔹 setJackpotStatus
  | 🗑 onlyAuthorized
✓ 🔹 setJackpotMinBuy
  | 🗑 onlyAuthorized
✓ 🔹 setJackpotFeatures
  | 🗑 onlyAuthorized
✓ 🔹 setJackpotHardFeatures
  | 🗑 onlyAuthorized
✓ 🔹 setJackpotTimespanInSeconds
  | 🗑 onlyAuthorized
```

Note: Not listed functions/modifiers were provided from implemented libraries

## Comments

- Deployer can set following state variables without any limitations
  - JackpotToken
    - jackpotTimespan
    - jackpotHardLimit
    - jackpotMinBuy
  - Treasury
    - numTokensSellToAddToLiquidity
- Deployer can enable/disable following state variables
  - JackpotToken
    - jackpotEnabled
  - Treasury
    - \_feeExempt
    - \_maxWalletExempt
    - \_txLimitExempt

- `_swapExempt`
- Ownable
  - `authorizations`
- JackpotUniverse
  - `tradingOpen`
  - `jackpotLimited`
  - `swapAndLiquifyEnabled`
- Deployer can set following addresses
  - JackpotToken
    - `uniswapV2Pair`
    - `uniswapV2Router`
  - Ownable
    - `_owner`
    - `_lockedLiquidity`
      - Can be set once only
    - `_buybackWallet`
    - `_marketingWallet`
    - `_teamWallet`
  - JackpotUniverse
    - `jGuard`
    - `jBroker`
- Existing Modifiers
  - `onlyAuthorized`
  - `onlyBuyback`
  - `onlyMarketing`
  - `onlyTeam`
  - `onlyOwner`
  - `lockTheSwap`
- There are several authorities which are authorized to call some functions, that means, if the owner is renounced, another address is still authorized to call functions
  - Be aware of this
- JackpotUniverse
  - Authorized address are able to reset Jackpot (will set `_lastBuyTimestamp` to 0 and `_lastBuyer` to contract address)
  - `fundJackpot` function
    - Authorized addresses are able to call `fundJackpot` with a `msg.value` of 0 in L247. If it is necessary to send an amount of bnb, we recommend you to implement a “require statement” that this function is only callable when the

caller pays some bnb like it was applied to the fallback/  
receive function

- awardJackpot/fundJackpots
  - We recommend you to check the balances before go deeper in the awardJackpot function
- AddLiquidity
  - Be aware of the lockedLiquidity address. The “\_lockedLiquidity” contract is not locked in this case because it is just a wallet address where the liquidity will be transferred. The owner can drain the liquidity out of the contract if it has a private key. Also he/she is able to change the address.
- JackpotToken
  - jackpotCashout
    - If value is set to 0 they bnb/tokens amount of buyBack (JackpotUniverse.sol, function: jackpotBuybackAmount L434) will always be 0 and also in the awardJackpot function L486 it will be 0 for tokens out L488 and cashedOut L487. The same applies to the “jackpotBuyerShare” value
      - We recommend you to implement a range above 0 and below MAX\_PCT to make sure, that the lastBuyer investor can get an award

### v1.3

- JackpotToken
  - Added functions
    - addLiquidityPool
    - delLiquidityPool
  - Removed functions
    - usdEquivalent













### v1.4

- JackpotUniverse
  - Added functions
    - setJackpotReferral














**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/Treasury.sol	1	————	478	449	342	14	238	————
	contracts/JackpotToken.sol	1	————	251	210	159	11	109	————
	contracts/IJackpotGuard.sol	————	1	29	6	3	1	17	————
	contracts/JackpotUniverse.sol	1	————	771	705	515	80	380	 
	contracts/Ownable.sol	2	————	250	234	142	61	83	————
	contracts/IJackpotBroker.sol	————	1	76	49	42	1	21	
	<b>Totals</b>	<b>5</b>	<b>2</b>	<b>1855</b>	<b>1653</b>	<b>1203</b>	<b>168</b>	<b>848</b>	 

### v1.3

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
	contracts/Treasury.sol	1	————	476	447	340	14	230	————
	contracts/JackpotToken.sol	1	————	271	227	169	13	110	————
	contracts/IJackpotGuard.sol	————	1	34	6	3	1	19	————
	contracts/JackpotUniverse.sol	1	————	789	723	532	79	386	 
	contracts/IJackpotReferral.sol	————	1	17	8	4	1	7	————
	contracts/Ownable.sol	2	————	252	236	142	63	83	————
	contracts/IJackpotBroker.sol	————	1	56	49	42	1	8	
	<b>Totals</b>	<b>5</b>	<b>3</b>	<b>1895</b>	<b>1696</b>	<b>1232</b>	<b>172</b>	<b>843</b>	 

### v1.4

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
	contracts/JackpotUniverse.sol	7	8	2968	2232	1467	551	1088	   
	<b>Totals</b>	<b>7</b>	<b>8</b>	<b>2968</b>	<b>2232</b>	<b>1467</b>	<b>551</b>	<b>1088</b>	   

## Legend

Attribute	Description
Lines	total lines of the source unit
nLines	normalised lines of the source unit (e.g. normalises functions spanning multiple lines)
nSLOC	normalised 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

**No informational issues**

### 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.

### 22. August 2022:

- Following files were not provided to solidproof
  - JackpotGuard
- We recommend you to do your own research here.

- Read whole report and modifiers section for more information, don't skip that section!
- No test cases were provided by the team
- Team acknowledged informational issues like
  - Unused state variables
  - Unused functions
  - Missing zero address
  - NatSpec documentation

## **27. August 2022:**

- Following files were not provided to solidproof
  - JackpotReferral
  - JackpotBroker
- Read whole report and modifiers section for more information, don't skip that section!

## **31. August 2022:**

- Read whole report and modifiers section for more information, don't skip that section!
- Audit version 1.4 is based on the JackpotUniverse contract

## 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>

  
Solid  
Proofed

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

  
MADE IN GERMANY