SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC
Development | Marketing

MADE IN GERMANY

RDB Token

Audit

Security Assessment
07. July, 2023

For

SolidProof_io

@solidproof_io

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	14. June 2023	• Layout project • Automated- /Manual-Security Testing • Summary
1.1	6. July 2023	• Reaudit

Note - This Audit report comprises a security analysis of the **RDB Token** smart contracts. This analysis did not include functional testing (or unit testing) of the contract's logic.

Network

Binance Smart Chain (BSC)

Website

https://rdbtoken.com/

Description

TBA

Project Engagement

During the Date of 12 June 2023, **RDB Token Team** engaged Solidproof.io to audit smart contracts that they created. The engagement was technical 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

- Provided as Files

v1.1

- https://bscscan.com/token/0x8cF19f8296556015AbA929BfA427CC4fA6999c94#code

Note for Investors: We only Audited two almost identical token contracts for **RDBTeam** with only one function. However, If the project has other contracts (for example, a Presale or staking contract etc), and they were not provided to us in the audit scope, then we cannot comment on its security, and we are not responsible for it in any way.

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 aspossible.
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.

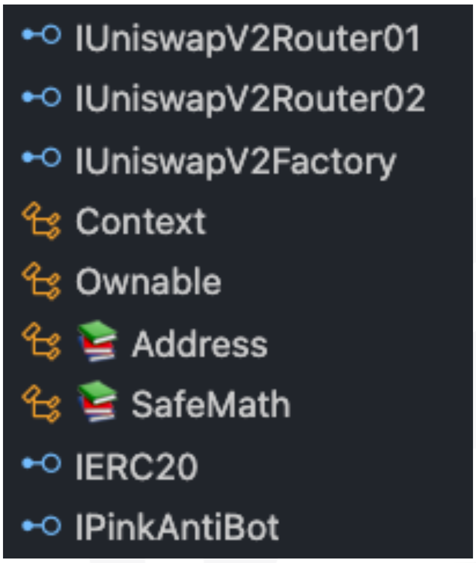
2. Testing and automated analysis that includes the following:
    i) Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
    ii) Symbolic execution, which is analysing a program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



IUniswapV2Router01
IUniswapV2Router02
IUniswapV2Factory
Context
Ownable
Address
SafeMath
IERC20
IPinkAntiBot

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.

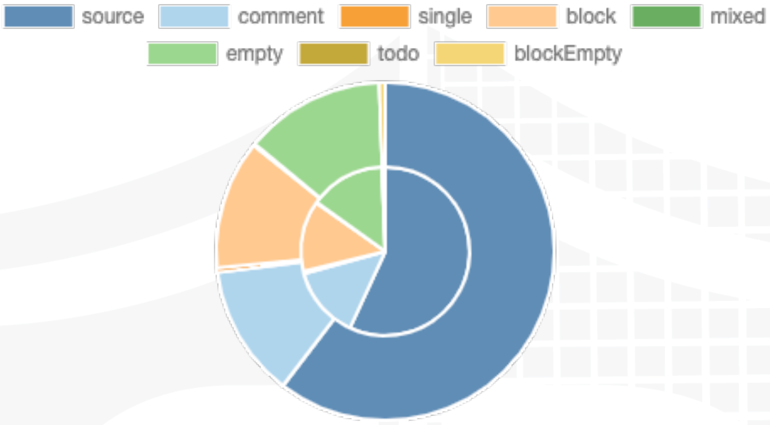
v1.1

File Name	SHA-1 Hash
contracts/RDB.sol	e725c8b4763f901e43c3ba954caee76b7d07da09
contracts/TSTtokenPinkBot.sol	b79d8b5d113bf7e908dd7613a10029513d07eb3a

Metrics

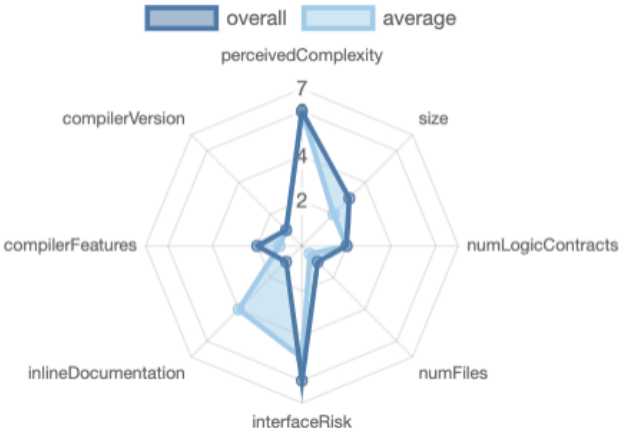
Source Lines

v1.0



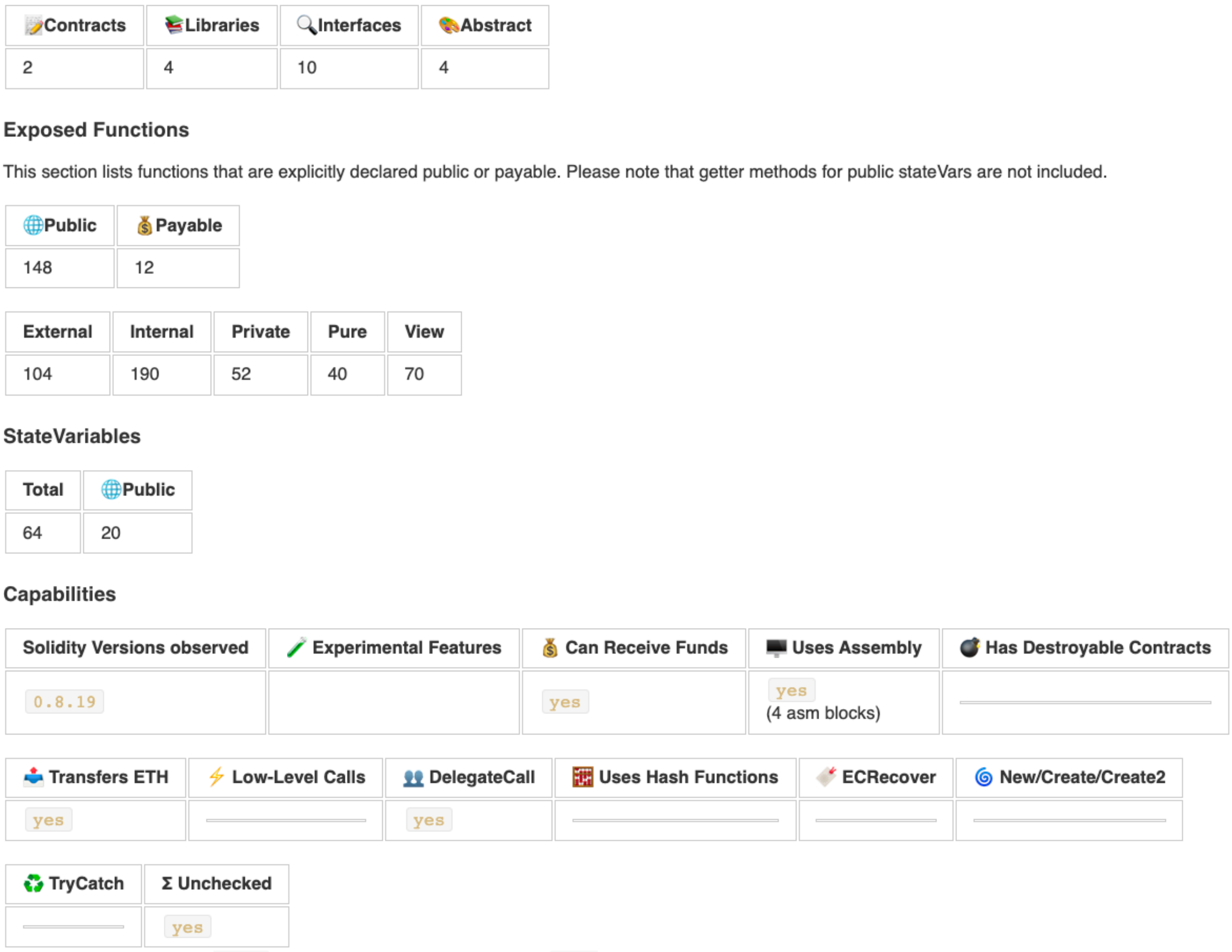
Risk Level

v1.0



Capabilities

Components



📝Contracts	📚Libraries	🔍Interfaces	🎨Abstract
2	4	10	4

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
148	12

External	Internal	Private	Pure	View
104	190	52	40	70

StateVariables

Total	🌐Public
64	20

Capabilities

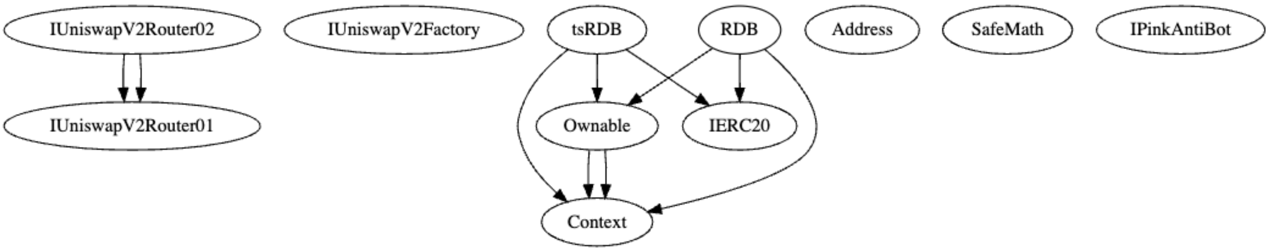
Solidity Versions observed	🧪 Experimental Features	💰 Can Receive Funds	🖥 Uses Assembly	💣 Has Destroyable Contracts
`0.8.19`		`yes`	`yes` (4 asm blocks)	

📤 Transfers ETH	⚡ Low-Level Calls	👥 DelegateCall	🧮 Uses Hash Functions	🔖 ECRecover	🌀 New/Create/Create2
`yes`		`yes`			

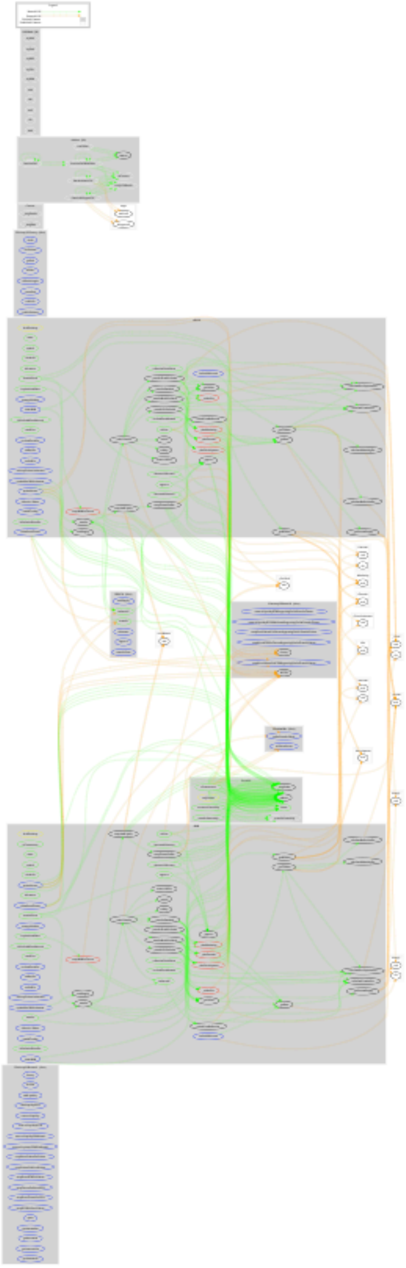
♻️ TryCatch	Σ Unchecked
	`yes`

Inheritance Graph

v1.0



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	✓	✓	✓

Deployer cannot mint any new tokens

Name	Exist	Tested	Status
Deployer cannot mint	✓	✓	✓
Max / Total Supply	100_000_000		

Comments:

v1.0

- Owner cannot mint new tokens

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	–	–	–

Deployer cannot set fees

Name	Exist	Tested	Status
Deployer cannot set fees over 10%	✓	✓	✓
Deployer cannot set fees to nearly 100% or to 100%	✓	✓	✓

Comments:

v1.0

- Fees cannot be set without any limitations

Deployer can blacklist/antisnipe addresses

Name	Exist	Tested	Status
Deployer cannot blacklist/antisnipe addresses	–	–	–

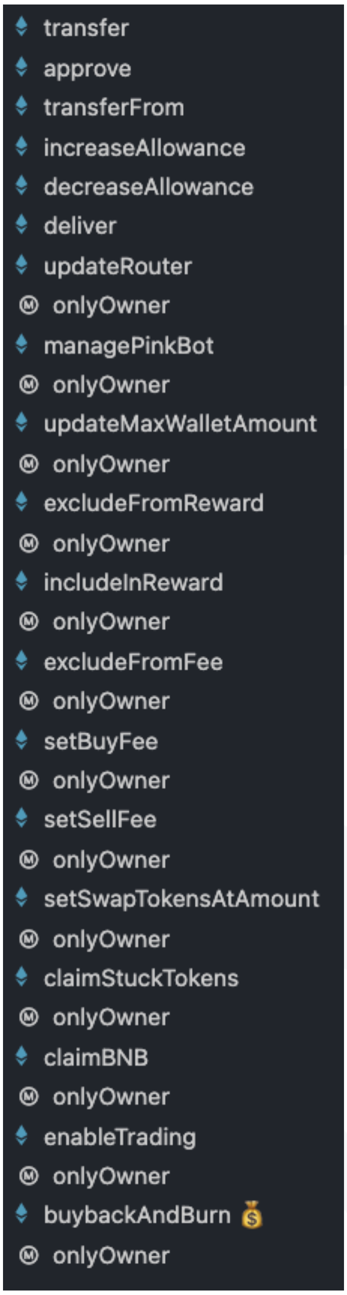
Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

Legend

Attribute	Symbol
Verified / Checked	✓
Partly Verified	🚩
Unverified / Not checked	✗
Not available	–

Modifiers and public functions
v1.1



```
♦ transfer
♦ approve
♦ transferFrom
♦ increaseAllowance
♦ decreaseAllowance
♦ deliver
♦ updateRouter
Ⓜ onlyOwner
♦ managePinkBot
Ⓜ onlyOwner
♦ updateMaxWalletAmount
Ⓜ onlyOwner
♦ excludeFromReward
Ⓜ onlyOwner
♦ includeInReward
Ⓜ onlyOwner
♦ excludeFromFee
Ⓜ onlyOwner
♦ setBuyFee
Ⓜ onlyOwner
♦ setSellFee
Ⓜ onlyOwner
♦ setSwapTokensAtAmount
Ⓜ onlyOwner
♦ claimStuckTokens
Ⓜ onlyOwner
♦ claimBNB
Ⓜ onlyOwner
♦ enableTrading
Ⓜ onlyOwner
♦ buybackAndBurn 💰
Ⓜ onlyOwner
```

Ownership Privileges

❖ RDB. sol -

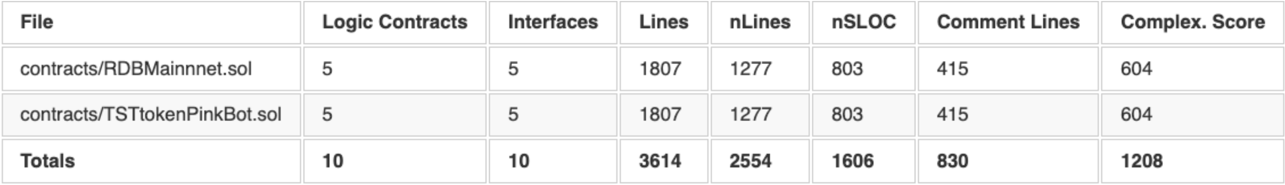
- Update router address
- Enable/Disable antibot
- Update max wallet but it must be more than 1% of the total supply
- Include/Exclude wallets from rewards and fees
- Set buy and sell fee, but not more than 10%
- Set liquidity threshold to any arbitrary value

    - Withdraw any type of tokens from the contract except for the native ones
    - Buy back and burn tokens manually
    - Enable Trading but cannot disable it
    - Update wallet addresses
    - Enable/Disable swap and liquify
    - Manually buyback and burn tokens

- *TSTtokenPinkBot.sol.so*l -
    - The privileges are as same as RDBMainnet.sol because the code is identical.

Please check if an OnlyOwner or similar restrictive modifier has been forgotten.

Source Units in Scope
v1.0



File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score
contracts/RDBMainnnet.sol	5	5	1807	1277	803	415	604
contracts/TSTtokenPinkBot.sol	5	5	1807	1277	803	415	604
Totals	**10**	**10**	**3614**	**2554**	**1606**	**830**	**1208**

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

Critical issues

No critical issues

High issues

No high issues

Medium issues

Medium Issues Fixed

Issue	File	Type	Line	Description	Status
#1	RDB Mainnet.sol	Trading can be done without Enabling it.	597, 674	The trading needs to be enabled by the owner in order for regular users to transfer tokens. On the contrary, the owner can exclude addresses from the fees manually and those addresses will be able to trade tokens. This functionality can be exploited in the following way. For example, there is a presale and the wallets used for the presale can be excluded from fees by the owner. All the tokens obtained can be consolidated into a final wallet address and facilitate trading and selling of the acquired tokens, the last wallet address can be excluded from fees.	Fixed
#2	RDB Mainnet.sol	Native tokens	437	The owner is able to pass the own contract address to “claimStuckTokens” that means if the contract holds token of “RDB” the owner can get it out. Here is is recommended to prevent passing the own contract address.	Fixed

Low issues

Issue	File	Type	Line	Description	Status
#1	RDBMainnet.sol	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	Fixed
#2	RDBMainnet.sol	Missing Zero Address Validation (missing-zero-check)	1210	Check that the address is not zero	Fixed
#3	RDBMainnet.sol	State variable visibility is not set	35	It is best practice to set the visibility of state variables explicitly	Fixed
#4	RDBMainnet.sol	State variables shadowing	652, 181	Rename the state variables that shadow another component. "owner" in this case	Fixed
#5	RDBMainnet.sol	Missing Events Arithmetic	All setter functions	Emit an event for critical parameter changes Emit in the updateRouter function also the uniswapV2Pair because it is possible that it can be changed.	Fixed

Informational issues

Issue	File	Type	Line	Description	Status
#1	RDBMainnet.sol	State variables that could be declared constant (constable-states)	49, 47, 48, 40, 84	Add the `constant` attributes to state variables that never change	Fixed

#2	RDB Mainnet.sol	SwapAndLiquify Enabled cannot be changed to false	84	The owner is unable to set the “swapAndLiquifyEnabled” variable to false. Implementing a function that the owner has the privilege to set it is recommended.	Fixed
#3	RDB Mainnet.sol	Functions that are not used	357-693	Remove unused functions. Before removing check the function, it could be possible, that you forget to implement it into the contract	Fixed
#4	RDB Mainnet.sol	Error message is missing	324, 596, 767, 402, 423,	Provide an error message for require statement	Fixed
#5	RDB Mainnet.sol	NatSpec documentation missing	—	If you started to comment your code, also comment all other functions, variables etc.	Fixed
#6	RDB Mainnet.sol	Replace placeholder	139-141	The addresses are only placeholder. Before deploying the contract to the mainnet replace the placeholder with your addresses.	Fixed

Audit Comments

We recommend you use the particular form of comments (NatSpec Format, Follow the link for more information https://docs.soliditylang.org/en/latest/natspec-format.html) for your contracts to provide rich documentation for functions, return variables and more. This helps investors to make clear what that variable, functions etc., do.

14. June 2023:

- Unit tests with 95% code coverage were not provided to SolidProof so we cannot ensure complete functional correctness of the code’s logic.
- We recommend **RDB Token** team conduct unit and fuzz tests thoroughly to rule out the possibility of unwanted logical and calculation errors.
- There is still an owner (The owner still has not renounced ownership)
- Read the whole report and modifiers section for more information

15. June 2023:

- setPresaleAddress function has been added to the contract
- updateWallets function has been added to the contract

07. July 2023:

- All the open issues are fixed
- Unit tests with 95% code coverage were still not provided to SolidProof so we cannot ensure complete functional correctness of the code's logic.
- We recommend **RDB Token** team conduct unit and fuzz tests thoroughly to rule out the possibility of unwanted logical and calculation errors.
- There is still an owner (The owner still has not renounced ownership)
- 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-131	Presence of unused variables	CWE-1164: Irrelevant Code	PASSED
SWC-130	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	PASSED
SWC-129	Typographical Error	CWE-480: Use of Incorrect Operator	PASSED
SWC-128	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	PASSED

SWC-127	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	PASSED
SWC-125	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	PASSED
SWC-124	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	PASSED
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	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

SolidProof_io @solidproof_io

Solid Proofed

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY