SOLIDProof
Bring trust into your projects

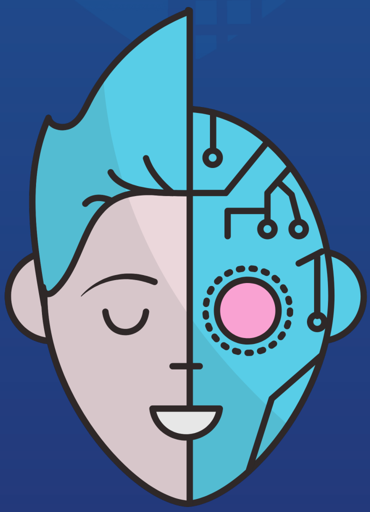
Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

Audit

Security Assessment
28. July, 2021

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	26. July 2021	Layout project
	27. July 2021	Automated- /Manual-Security Testing
	27. July 2021	Summary
1.1	29. July 2021	Blacklist function added

Network

Binance Smart Chain (BEP20)

Website

https://robotoken.app/

Telegram

https://t.me/robotokenofficial

Twitter

https://twitter.com/tokenrobo?lang=en

Github

https://github.com/roboapp

Medium

https://medium.com/@RoboToken

Instagram

https://www.instagram.com/tokenrobo/

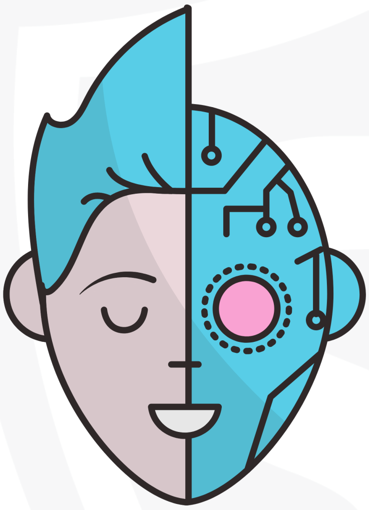
Description

RoboToken is a modern eco-system with a BNB reward system, auto-claim, buy back, dex swap, and launchpad all under one token $ROBO.

Project Engagement

During the 26th of July 2021, **RoboToken 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. **RoboToken Team** provided Solidproof.io with access to their code repository and whitepaper.

Logo



Contract Link

TBA

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:

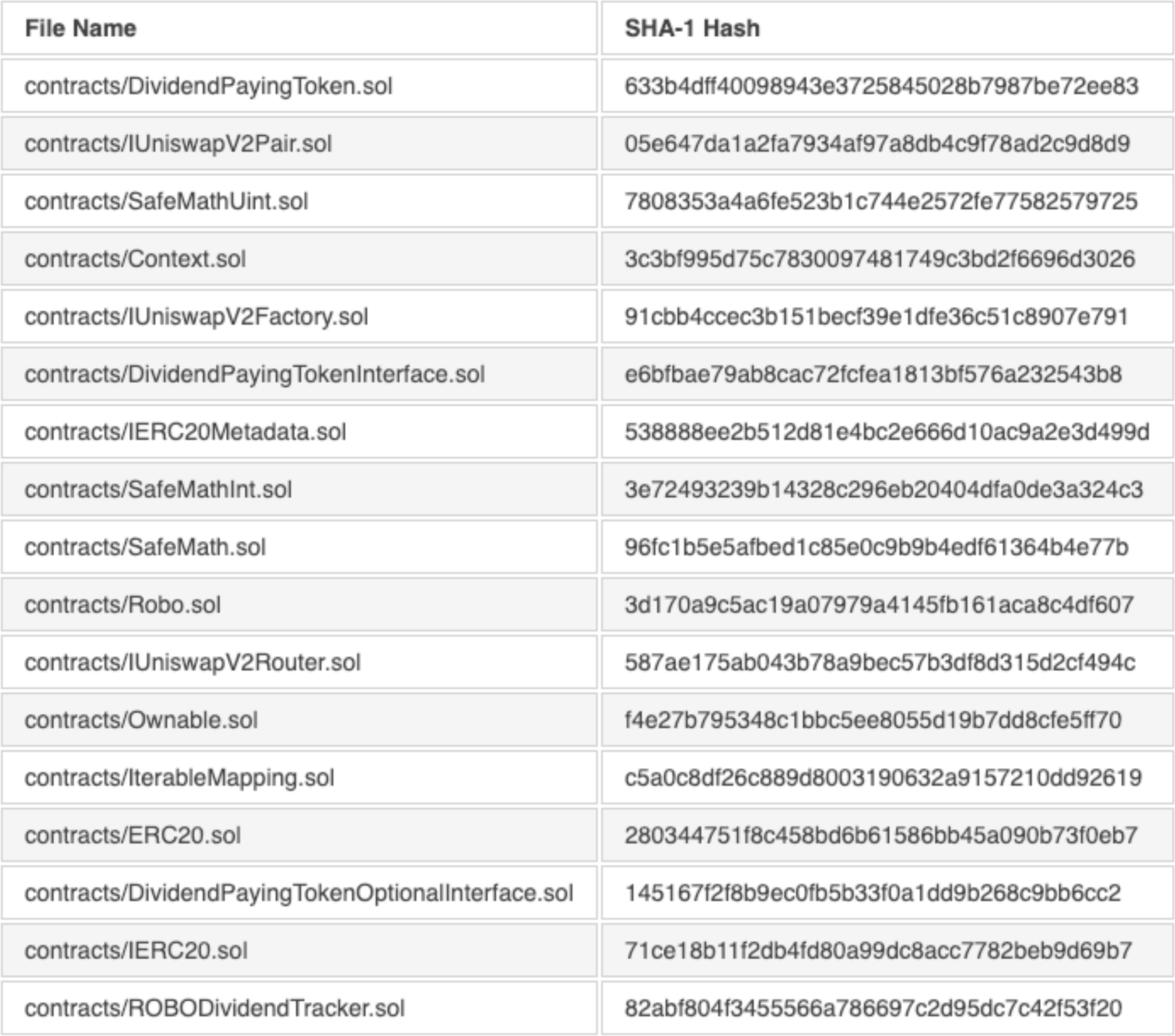
- OpenZeppelin
    - Address
    - Ownable
    - SafeMatch
- Uniswap
    - UniswapV2Factory
    - UniswapV2Pair
    - UniswapV2Router01
    - UniswapV2Router02

Tested Contract Files

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

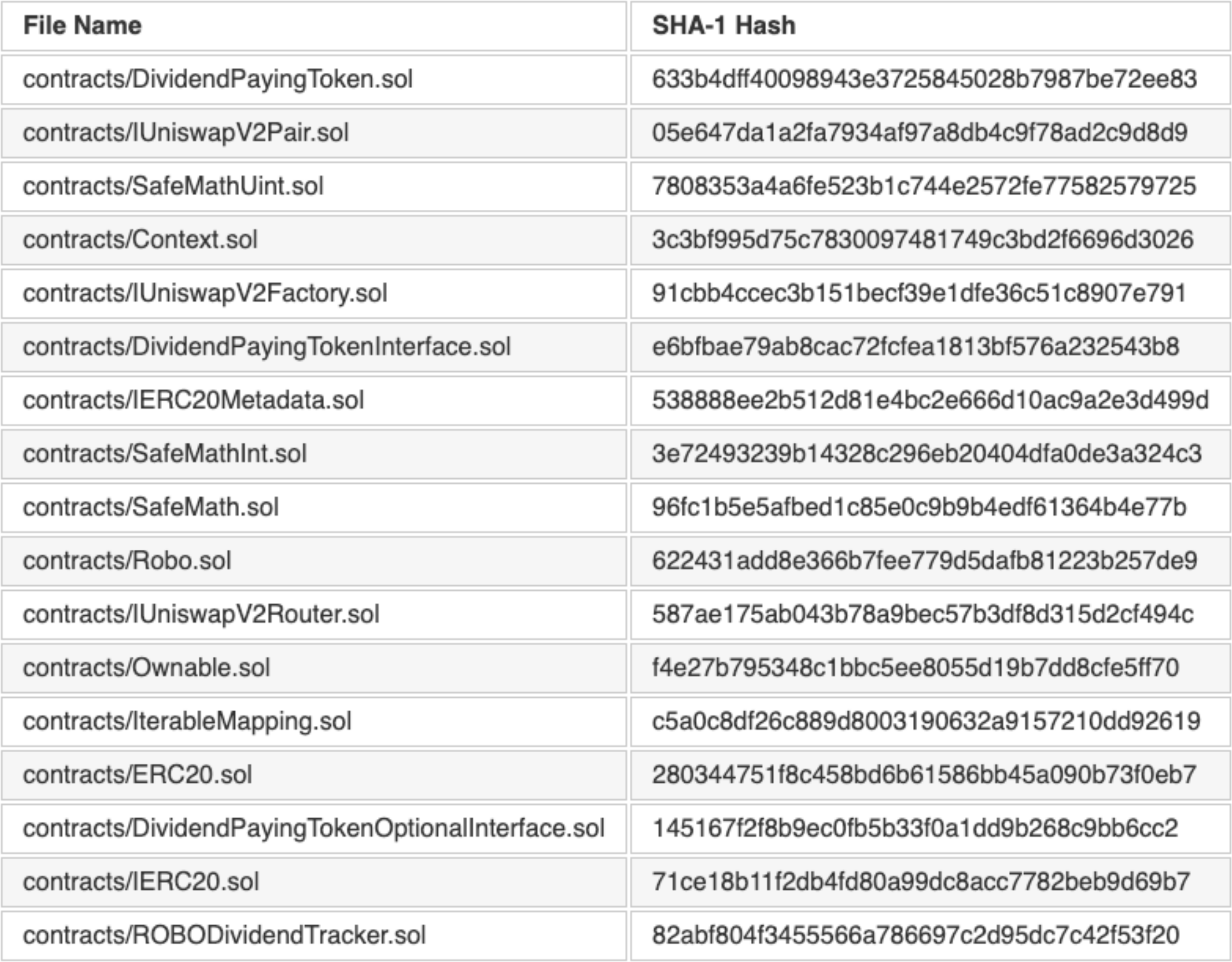
A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

v1.0



File Name	SHA-1 Hash
contracts/DividendPayingToken.sol	633b4dff40098943e3725845028b7987be72ee83
contracts/IUniswapV2Pair.sol	05e647da1a2fa7934af97a8db4c9f78ad2c9d8d9
contracts/SafeMathUint.sol	7808353a4a6fe523b1c744e2572fe77582579725
contracts/Context.sol	3c3bf995d75c7830097481749c3bd2f6696d3026
contracts/IUniswapV2Factory.sol	91cbb4ccec3b151becf39e1dfe36c51c8907e791
contracts/DividendPayingTokenInterface.sol	e6bfbae79ab8cac72fcfea1813bf576a232543b8
contracts/IERC20Metadata.sol	538888ee2b512d81e4bc2e666d10ac9a2e3d499d
contracts/SafeMathInt.sol	3e72493239b14328c296eb20404dfa0de3a324c3
contracts/SafeMath.sol	96fc1b5e5afbed1c85e0c9b9b4edf61364b4e77b
contracts/Robo.sol	3d170a9c5ac19a07979a4145fb161aca8c4df607
contracts/IUniswapV2Router.sol	587ae175ab043b78a9bec57b3df8d315d2cf494c
contracts/Ownable.sol	f4e27b795348c1bbc5ee8055d19b7dd8cfe5ff70
contracts/IterableMapping.sol	c5a0c8df26c889d8003190632a9157210dd92619
contracts/ERC20.sol	280344751f8c458bd6b61586bb45a090b73f0eb7
contracts/DividendPayingTokenOptionalInterface.sol	145167f2f8b9ec0fb5b33f0a1dd9b268c9bb6cc2
contracts/IERC20.sol	71ce18b11f2db4fd80a99dc8acc7782beb9d69b7
contracts/ROBODividendTracker.sol	82abf804f3455566a786697c2d95dc7c42f53f20

v1.1

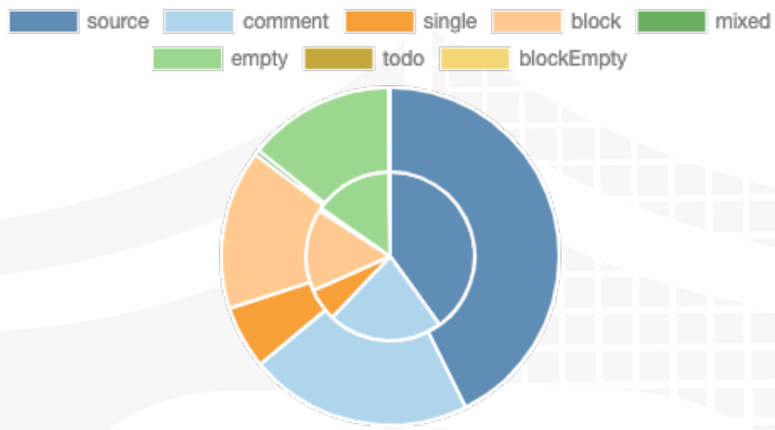


File Name	SHA-1 Hash
contracts/DividendPayingToken.sol	633b4dff40098943e3725845028b7987be72ee83
contracts/IUniswapV2Pair.sol	05e647da1a2fa7934af97a8db4c9f78ad2c9d8d9
contracts/SafeMathUint.sol	7808353a4a6fe523b1c744e2572fe77582579725
contracts/Context.sol	3c3bf995d75c7830097481749c3bd2f6696d3026
contracts/IUniswapV2Factory.sol	91cbb4ccec3b151becf39e1dfe36c51c8907e791
contracts/DividendPayingTokenInterface.sol	e6bfbae79ab8cac72fcfea1813bf576a232543b8
contracts/IERC20Metadata.sol	538888ee2b512d81e4bc2e666d10ac9a2e3d499d
contracts/SafeMathInt.sol	3e72493239b14328c296eb20404dfa0de3a324c3
contracts/SafeMath.sol	96fc1b5e5afbed1c85e0c9b9b4edf61364b4e77b
contracts/Robo.sol	622431add8e366b7fee779d5dafb81223b257de9
contracts/IUniswapV2Router.sol	587ae175ab043b78a9bec57b3df8d315d2cf494c
contracts/Ownable.sol	f4e27b795348c1bbc5ee8055d19b7dd8cfe5ff70
contracts/IterableMapping.sol	c5a0c8df26c889d8003190632a9157210dd92619
contracts/ERC20.sol	280344751f8c458bd6b61586bb45a090b73f0eb7
contracts/DividendPayingTokenOptionalInterface.sol	145167f2f8b9ec0fb5b33f0a1dd9b268c9bb6cc2
contracts/IERC20.sol	71ce18b11f2db4fd80a99dc8acc7782beb9d69b7
contracts/ROBODividendTracker.sol	82abf804f3455566a786697c2d95dc7c42f53f20

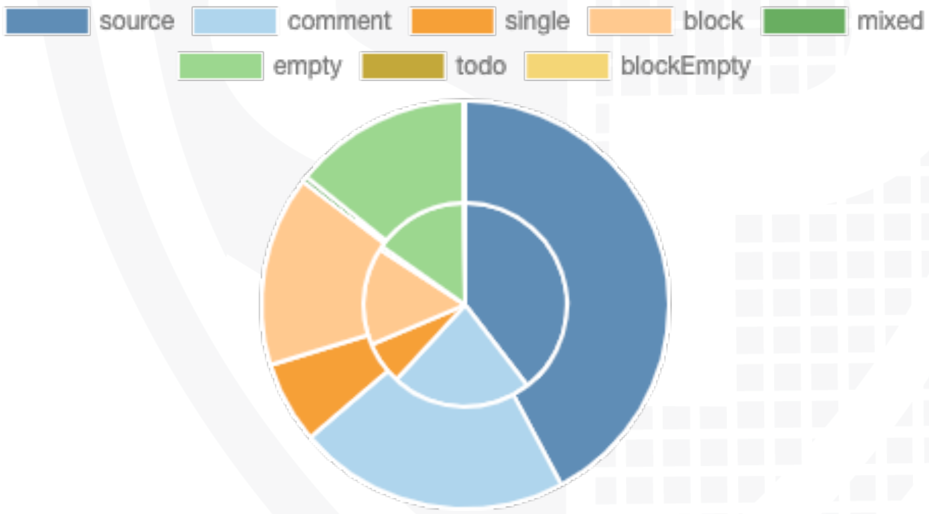
Metrics

Source Lines

v1.0

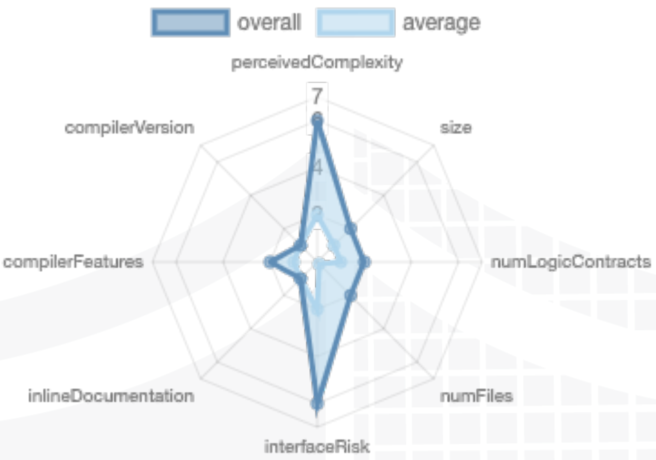


V1.1

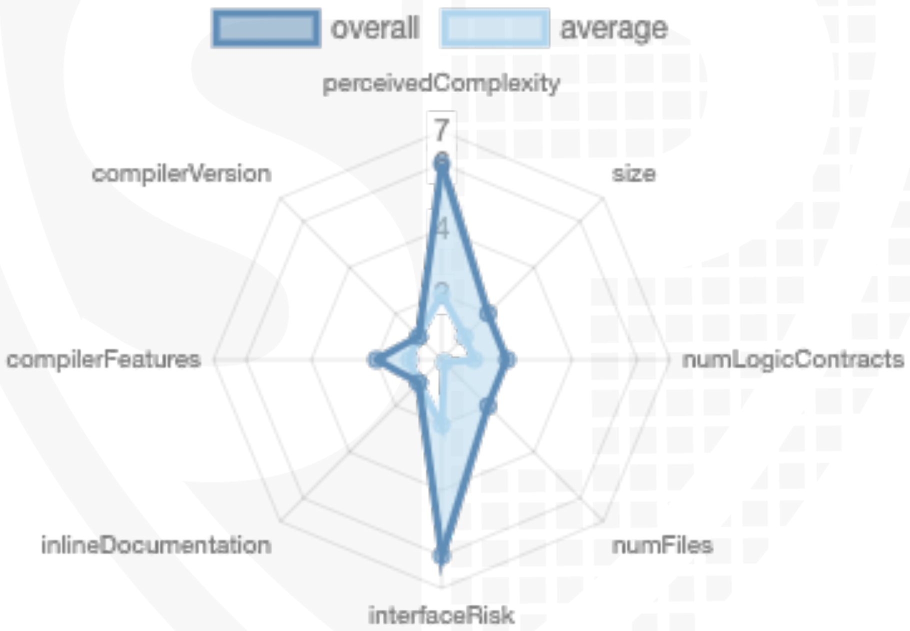


Risk Level

v1.0



v1.1



Capabilities

Components

Contracts	Libraries	Interfaces	Abstract
5	4	8	1

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	154	8
1.1	155	8

Version	External	Internal	Private	Pure	View
1.0	104	125	9	25	63
1.1	105	126	9	25	63

State Variables

Version	Total	Public
1.0	45	29
1.1	46	30

Capabilities

Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts

`^0.6.2`
`0.6.12`
		Yes	****
			(0 asm blocks)	

Transfers ETH	Low-Level Calls	Delegate Call	Uses Hash Functions	ECRecover	New/ Create/ Create2
`yes`					`yes` → `NewContract: ROBODividendTracker`

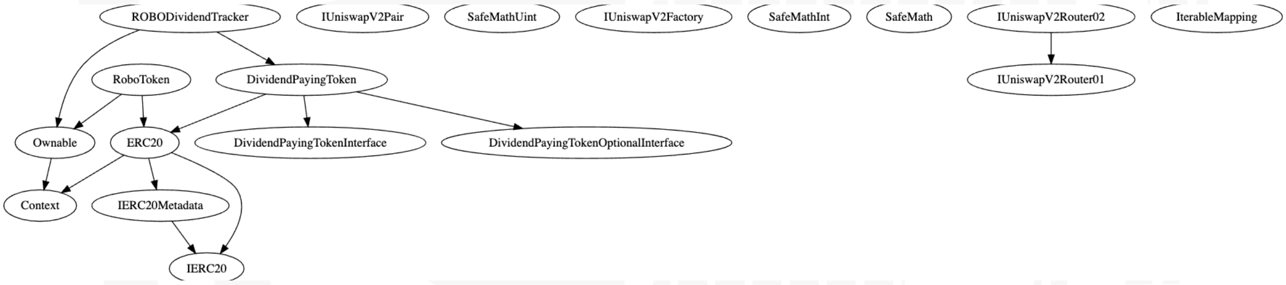
Scope of Work

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



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	✓	✓	✓

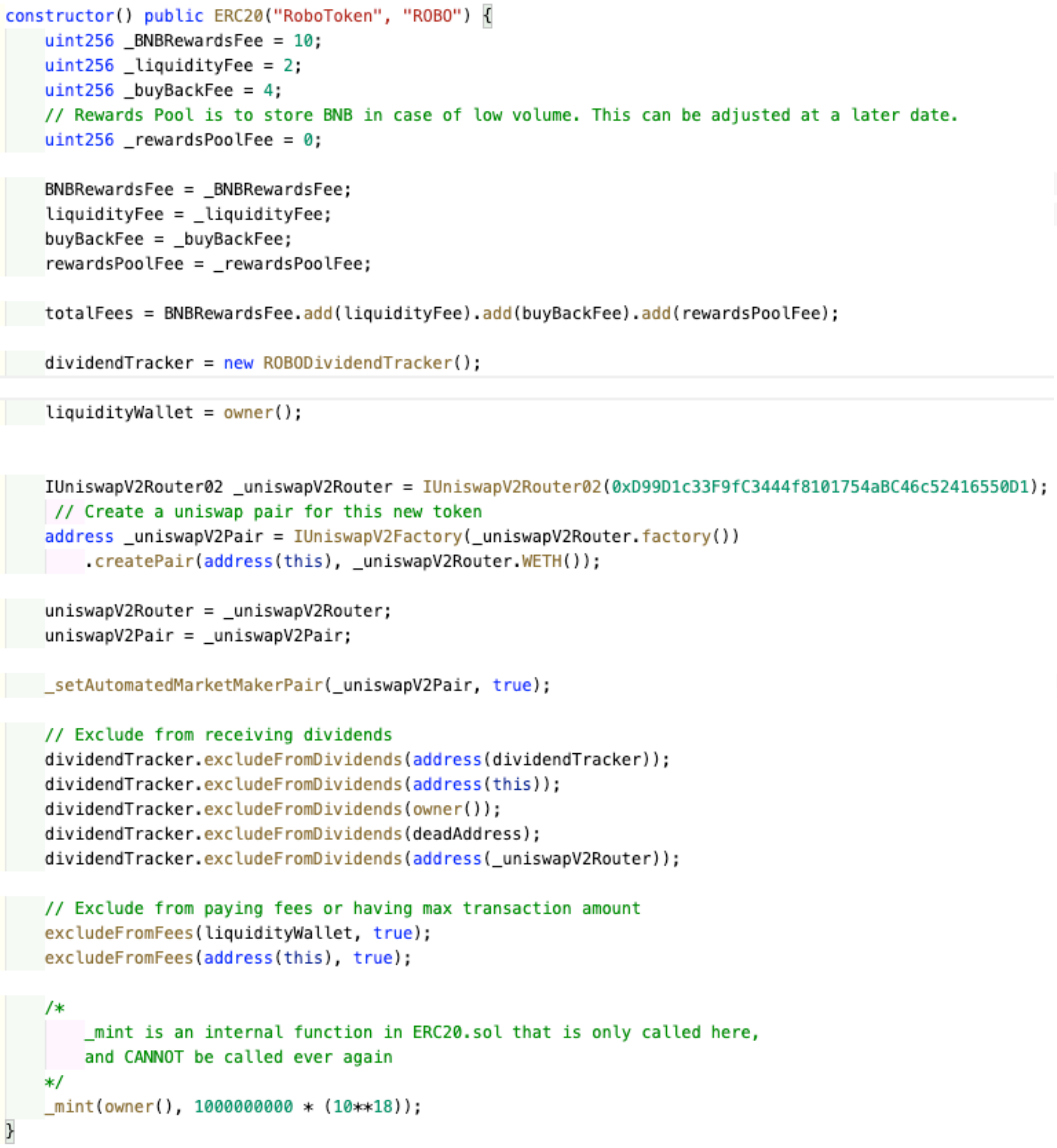
Optional implementations

Function	Description	Exist	Tested	Verified
renounceOwnership	Owner renounce ownership for more trust	✓	✓	✓

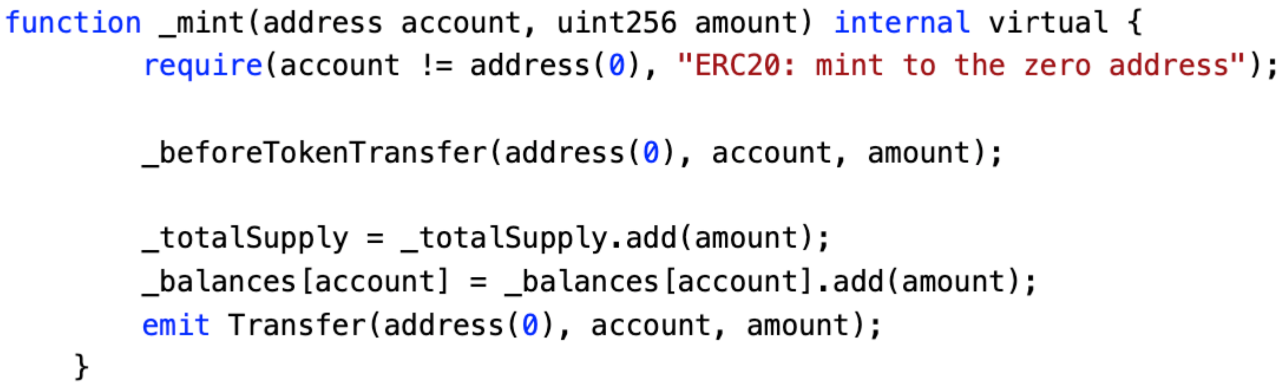
Deployer cannot mint any new tokens

Tested	Verified	File	Comment
✓	✓	Main	Line: -

Max / Total Supply: 1.000.000.000



```
constructor() public ERC20("RoboToken", "ROBO") {
    uint256 _BNBRewardsFee = 10;
    uint256 _liquidityFee = 2;
    uint256 _buyBackFee = 4;
    // Rewards Pool is to store BNB in case of low volume. This can be adjusted at a later date.
    uint256 _rewardsPoolFee = 0;

    BNBRewardsFee = _BNBRewardsFee;
    liquidityFee = _liquidityFee;
    buyBackFee = _buyBackFee;
    rewardsPoolFee = _rewardsPoolFee;

    totalFees = BNBRewardsFee.add(liquidityFee).add(buyBackFee).add(rewardsPoolFee);

    dividendTracker = new ROBODividendTracker();

    liquidityWallet = owner();


    IUniswapV2Router02 _uniswapV2Router = IUniswapV2Router02(0xD99D1c33F9fC3444f8101754aBC46c52416550D1);
     // Create a uniswap pair for this new token
    address _uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory())
        .createPair(address(this), _uniswapV2Router.WETH());

    uniswapV2Router = _uniswapV2Router;
    uniswapV2Pair = _uniswapV2Pair;

    _setAutomatedMarketMakerPair(_uniswapV2Pair, true);

    // Exclude from receiving dividends
    dividendTracker.excludeFromDividends(address(dividendTracker));
    dividendTracker.excludeFromDividends(address(this));
    dividendTracker.excludeFromDividends(owner());
    dividendTracker.excludeFromDividends(deadAddress);
    dividendTracker.excludeFromDividends(address(_uniswapV2Router));

    // Exclude from paying fees or having max transaction amount
    excludeFromFees(liquidityWallet, true);
    excludeFromFees(address(this), true);

    /*
        _mint is an internal function in ERC20.sol that is only called here,
        and CANNOT be called ever again
    */
    _mint(owner(), 1000000000 * (10**18));
}
```



```
function _mint(address account, uint256 amount) internal virtual {
        require(account != address(0), "ERC20: mint to the zero address");

        _beforeTokenTransfer(address(0), account, amount);

        _totalSupply = _totalSupply.add(amount);
        _balances[account] = _balances[account].add(amount);
        emit Transfer(address(0), account, amount);
    }
```

Deployer cannot burn or lock user funds

Name	Tested	Exist	Verified
No Lock function	✓	✓	✓
No Burn function	✓	✓	✓

Deployer cannot pause the contract

Tested	Verified	No pause function
✓	✓	✓

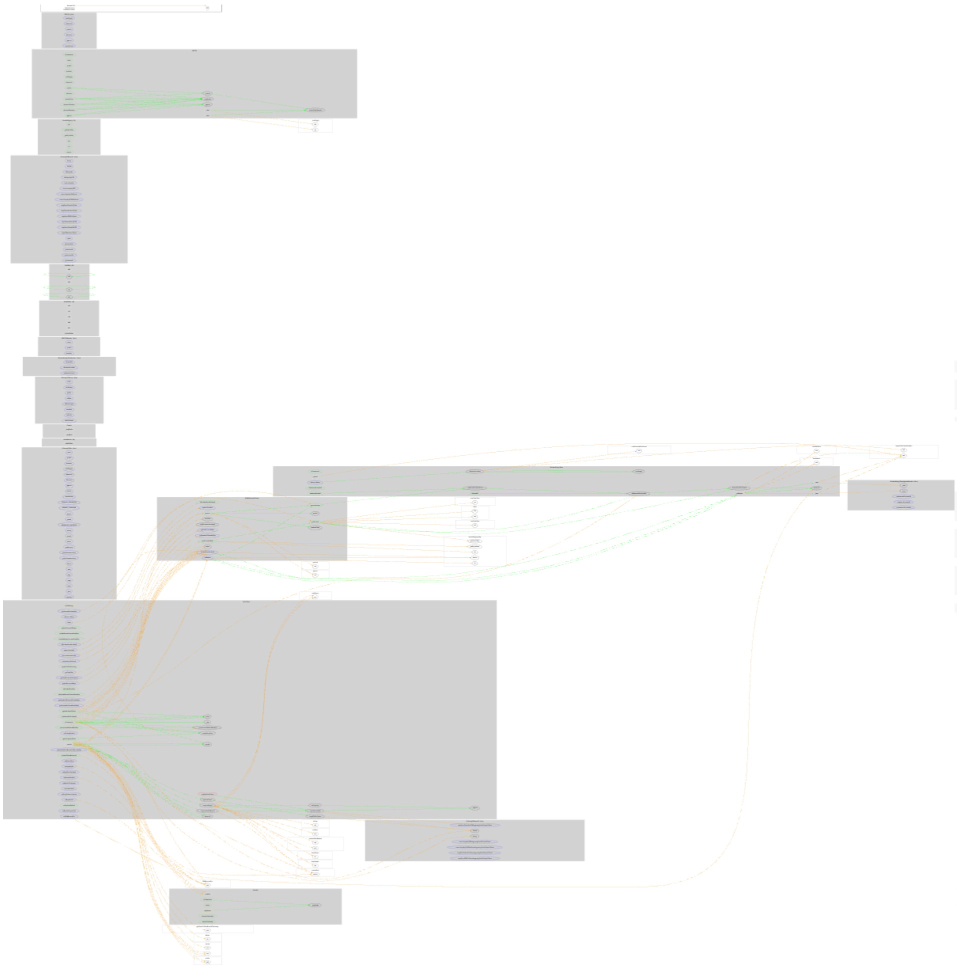
Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

Legend

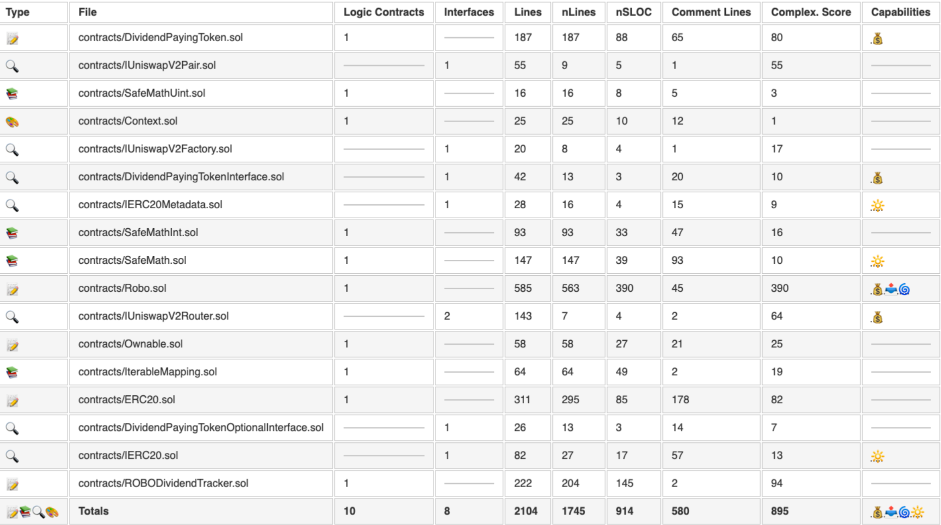
Attribute	Symbol
Verfified / Checked	✓
Partly Verified	⚑
Unverified / Not checked	✗

CallGraph



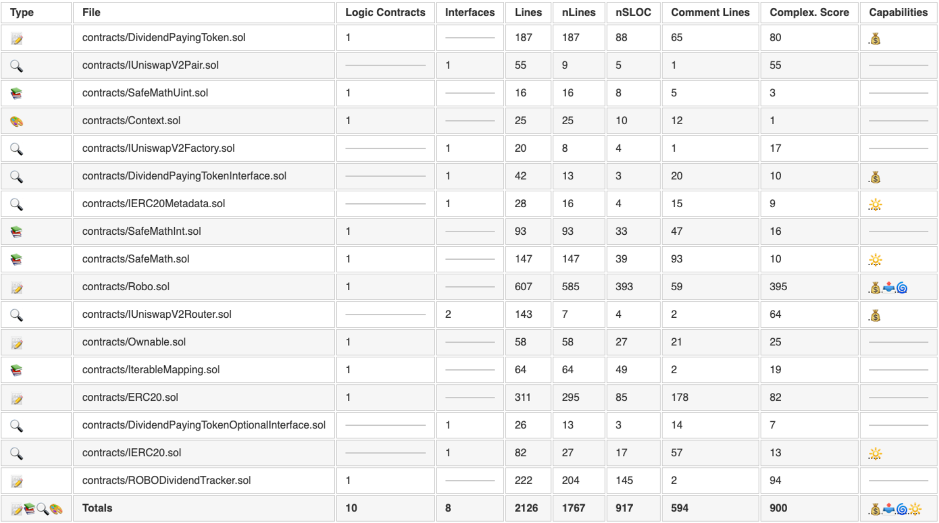
Source Units in Scope

v1.0



Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝	contracts/DividendPayingToken.sol	1	—	187	187	88	65	80	💰
🔍	contracts/IUniswapV2Pair.sol	—	1	55	9	5	1	55	—
📚	contracts/SafeMathUint.sol	1	—	16	16	8	5	3	—
🎨	contracts/Context.sol	1	—	25	25	10	12	1	—
🔍	contracts/IUniswapV2Factory.sol	—	1	20	8	4	1	17	—
🔍	contracts/DividendPayingTokenInterface.sol	—	1	42	13	3	20	10	💰
🔍	contracts/IERC20Metadata.sol	—	1	28	16	4	15	9	🔆
📚	contracts/SafeMathInt.sol	1	—	93	93	33	47	16	—
📚	contracts/SafeMath.sol	1	—	147	147	39	93	10	🔆
📝	contracts/Robo.sol	1	—	585	563	390	45	390	💰📥🌀
🔍	contracts/IUniswapV2Router.sol	—	2	143	7	4	2	64	💰
📝	contracts/Ownable.sol	1	—	58	58	27	21	25	—
📚	contracts/IterableMapping.sol	1	—	64	64	49	2	19	—
📝	contracts/ERC20.sol	1	—	311	295	85	178	82	—
🔍	contracts/DividendPayingTokenOptionalInterface.sol	—	1	26	13	3	14	7	—
🔍	contracts/IERC20.sol	—	1	82	27	17	57	13	🔆
📝	contracts/ROBODividendTracker.sol	1	—	222	204	145	2	94	—
📝📚🔍🎨	**Totals**	**10**	**8**	**2104**	**1745**	**914**	**580**	**895**	💰📥🌀🔆

v1.1



Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝	contracts/DividendPayingToken.sol	1	—	187	187	88	65	80	💰
🔍	contracts/IUniswapV2Pair.sol	—	1	55	9	5	1	55	—
📚	contracts/SafeMathUint.sol	1	—	16	16	8	5	3	—
🎨	contracts/Context.sol	1	—	25	25	10	12	1	—
🔍	contracts/IUniswapV2Factory.sol	—	1	20	8	4	1	17	—
🔍	contracts/DividendPayingTokenInterface.sol	—	1	42	13	3	20	10	💰
🔍	contracts/IERC20Metadata.sol	—	1	28	16	4	15	9	🔆
📚	contracts/SafeMathInt.sol	1	—	93	93	33	47	16	—
📚	contracts/SafeMath.sol	1	—	147	147	39	93	10	🔆
📝	contracts/Robo.sol	1	—	607	585	393	59	395	💰📥🌀
🔍	contracts/IUniswapV2Router.sol	—	2	143	7	4	2	64	💰
📝	contracts/Ownable.sol	1	—	58	58	27	21	25	—
📚	contracts/IterableMapping.sol	1	—	64	64	49	2	19	—
📝	contracts/ERC20.sol	1	—	311	295	85	178	82	—
🔍	contracts/DividendPayingTokenOptionalInterface.sol	—	1	26	13	3	14	7	—
🔍	contracts/IERC20.sol	—	1	82	27	17	57	13	🔆
📝	contracts/ROBODividendTracker.sol	1	—	222	204	145	2	94	—
📝📚🔍🎨	**Totals**	**10**	**8**	**2126**	**1767**	**917**	**594**	**900**	💰📥🌀🔆

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 -

Low issues

- no low issues found -

Issue	File	Type	Line	Description
#1	Main	Call with hardcoded gas amount.	396	rewardsPool.transfer(address (this).balance.div(2)) forwards a fixed amount of gas. This is discouraged as the gas cost of EVM instructions may change in the future, which could break this contract's assumptions.
#2	Main	Call with hardcoded gas amount.	513	rewardsPool.transfer(reward BNB) forwards a fixed amount of gas. This is discouraged as the gas cost of EVM instructions may change in the future, which could break this contract's assumptions.
#3	Main	Call with hardcoded gas amount.	575	rewardsPool.transfer(reward BNB) forwards a fixed amount of gas. This is discouraged as the gas cost of EVM instructions may change in the future, which could break this contract's assumptions.

Informational issues

- no informational issues found -

Audit Comments

27. July 2021

- Wrong Routeraddress were used (Testnet router address: 0xD99D1c33F9fC3444f8101754aBC46c52416550D1)

28. July 2021:

- Routeraddress fixed

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	NOT 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

Solid Proofed

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

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