



SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

Audit

Security Assessment
19. August, 2021

For



Disclaimer	3
Description	5
Project Engagement	5
Logo	5
Contract Link	5
Methodology	7
Used Code from other Frameworks/Smart Contracts (direct imports)	8
Tested Contract Files	9
Source Lines	10
Risk Level	10
Capabilities	11
Scope of Work	13
Inheritance Graph	13
Verify Claims	14
CallGraph	20
Source Units in Scope	21
Critical issues	22
High issues	22
Medium issues	22
Low issues	22
Informational issues	22
Audit Comments	23
SWC Attacks	24

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	19. August 2021	<ul style="list-style-type: none">• Layout project• Automated- /Manual-Security Testing• Summary

Network

Binance Smart Chain (BEP20)

Website

<https://www.thecoinbandit.com/>

Telegram

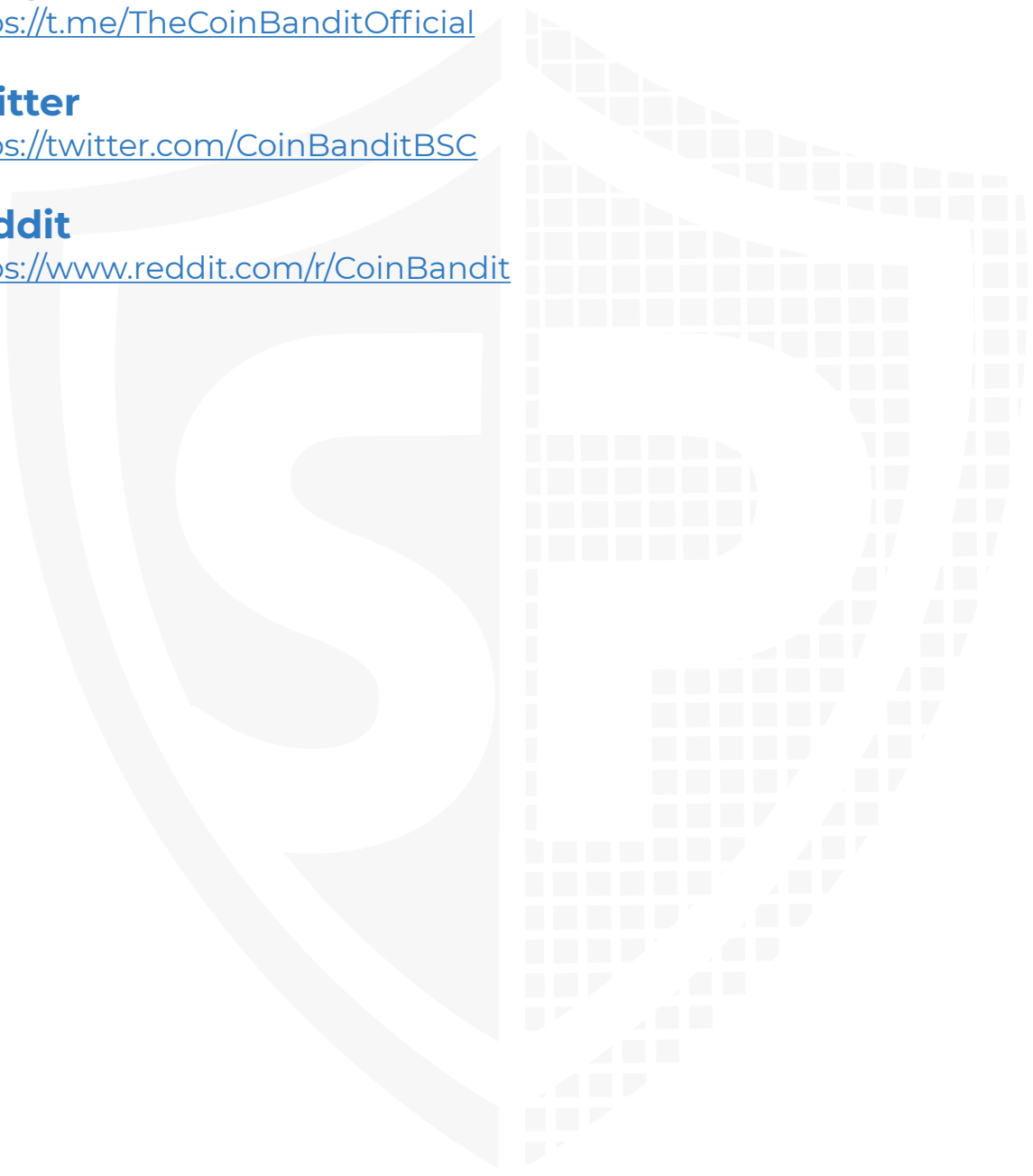
<https://t.me/TheCoinBanditOfficial>

Twitter

<https://twitter.com/CoinBanditBSC>

Reddit

<https://www.reddit.com/r/CoinBandit>



Description

TBA

Project Engagement

During the 18th of August 2021, **Coin Bandit 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. **Coin Bandit Team** provided Solidproof.io with access to their code repository and whitepaper.

Logo



Contract Link

<https://bscscan.com/address/0x3a1847b68e9fe7cd0e612654b811bc69737b7067#code>

Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system. Risk Level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
Critical	9 - 10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
High	7 – 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon as possible.
Medium	4 – 6.9	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.	Implementation of corrective actions in a certain period.
Low	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. To do so, reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as there were discovered.

Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
 - i) Review of the specifications, sources, and instructions provided to SolidProof to make sure we understand the size, scope, and functionality of the smart contract.
 - ii) Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
 - iii) Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to SolidProof describe.
2. Testing and automated analysis that includes the following:
 - i) Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
 - ii) Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

Used Code from other Frameworks/Smart Contracts (direct imports)

Imported packages:

- OpenZeppelin
 - Address
 - Ownable
 - SafeMath
 - SafeMathInt
 - ERC20
 - IERC20Metadata
 - IERC20
- Uniswap
 - UniswapV2Factory
 - UniswapV2Pair
 - UniswapV2Router01
 - UniswapV2Router02

Tested Contract Files

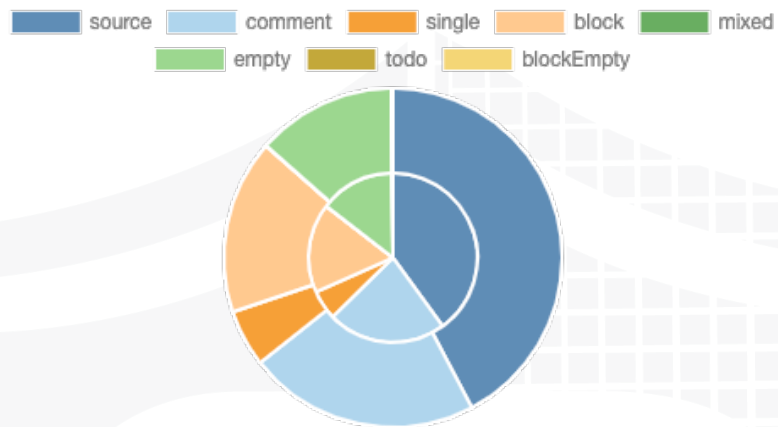
This audit covered the following files listed below with a SHA-1 Hash.

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.

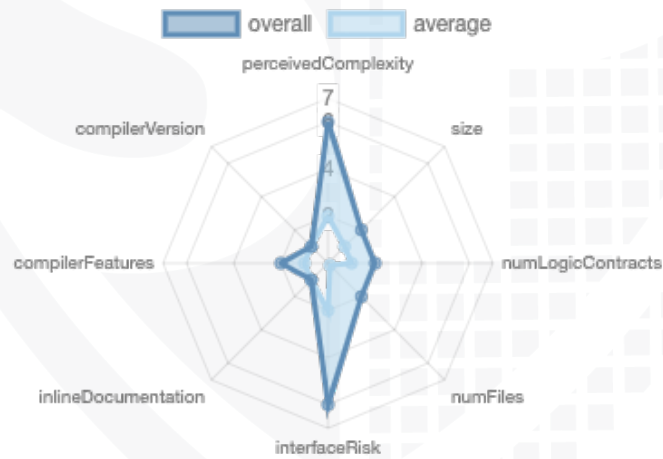
File Name	SHA-1 Hash
contracts/DividendPayingToken.sol	2272eafe86a659505b417d8fa39a817781e5f2b8
contracts/SafeMathUint.sol	6f99bcb4dd5ead52b8228c5b87f19e99fa52bc0a
contracts/IUniswapV2Pair.sol	7207d056afb16ab682366921ca9999716fdf0856
contracts/Context.sol	8cce9893723e51ed8050e79c3165f9e053fad9d
contracts/IERC20Metadata.sol	606030b128b534388035e75cb6134b39eb548dc3
contracts/SafeMathInt.sol	bb7854f7dc17282a4ac110b658449dfbebec1327
contracts/CoinBanditTransferHelper.sol	285d65411837dce1425405c7dca382f1ce7fdd19
contracts/CoinBandit.sol	28b55fbd2ad456414b797d8986e0e3aa43244b94
contracts/SafeMath.sol	6affa39a0c096511d93fe5c75a9f0a7abbd60bab
contracts/IUniswapV2Router.sol	9e3917429e23e27635ee08260deaeef1f0b628a01
contracts/Ownable.sol	bbe4bd73a5bbc3a23530eb251389825c43bf35e9
contracts/IterableMapping.sol	5cce9efb0720a00a04add294b5ba94ea2fceecc7
contracts/ERC20.sol	66c486ace86e7a4cb4be33fbd2f89f59191e6280
contracts/DividendPayingTokenOptionalInterface.sol	2699e35292de5292c15e74a69c28c34cc03eebb2
contracts/DividendPayingTokenInterface.sol	79bd44d69ac17efddb8a7e009ad37665676046a8
contracts/IERC20.sol	a9401814cae232061d41e31ccfc85d2243edcecd
contracts/IUniswapV2Factory.sol	525f9ecf7bd7b4dd8a44a976211127f7a668f9e2

Metrics

Source Lines



Risk Level



Capabilities

Components

Contracts	Libraries	Interfaces	Abstract
6	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.

Public	Payable
148	6

External	Internal	Private	Pure	View
92	119	8	26	63

State Variables

Total	Public
42	23

Capabilities

Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
<code>^0.6.2</code>		yes	**** (0 asm blocks)	

Transfers ETH	Low-Level Calls	Delegate Call	Uses Hash Functions	ECRecover	New/Create/Create2
---------------	-----------------	---------------	---------------------	-----------	--------------------

yes					yes → NewContract: COINBANDITDividendTracker → NewContract: COINBANDITTransferHelper
-----	--	--	--	--	--



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

Name	Exist	Tested	Verified	File
Deployer cannot mint	✓	✓	✓	Main
Comment	Line: -			

Max / Total Supply: 100.000.000.000

```

constructor(address routerAddress, address payable marketingWalletAddress) public ERC20("COIN BANDIT", "COINBANDIT") {
    dividendTracker = new COINBANDITDividendTracker();
    transferHelper = new COINBANDITTransferHelper(routerAddress);

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

    uniswapV2Router = _uniswapV2Router;
    uniswapV2Pair = _uniswapV2Pair;

    _setAutomatedMarketMakerPair(_uniswapV2Pair, true);
    setMarketingWallet(marketingWalletAddress);

    // exclude from receiving dividends
    dividendTracker.excludeFromDividends(address(dividendTracker));
    dividendTracker.excludeFromDividends(address(transferHelper));
    dividendTracker.excludeFromDividends(address(this));
    dividendTracker.excludeFromDividends(owner());
    dividendTracker.excludeFromDividends(deadWallet);
    dividendTracker.excludeFromDividends(address(_uniswapV2Router));

    // exclude from paying fees or having max transaction amount
    excludeFromFees(owner(), true);
    excludeFromFees(address(this), true);
    excludeFromFees(address(transferHelper), true);

    // exclude from max wallet
    excludeFromMaxWallet(owner(), true);
    excludeFromMaxWallet(address(this), true);
    excludeFromMaxWallet(deadWallet, true);
    excludeFromMaxWallet(address(0), true);
    excludeFromMaxWallet(address(transferHelper), true);

    // set default fees (dividends, marketing, liquidity)
    setBuyFees(7, 5, 3);
    setSellFees(7, 5, 3);

    /*
    _mint is an internal function in ERC20.sol that is only called here,
    and CANNOT be called ever again
    */
    _mint(owner(), 100000000000 * (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	Exist	Tested	Verified
Deployer cannot lock	✓	✓	✓
Deployer cannot burn	✓	✓	✓

1. approve	→
2. blacklistAddress	→
3. claim	→
4. decreaseAllowance	→
5. excludeFromDividends	→
6. excludeFromFees	→
7. excludeFromMaxWallet	→
8. increaseAllowance	→
9. processDividendTracker	→
10. recover	→
11. renounceOwnership	→
12. setAutomatedMarketMakerPair	→
13. setBuyFees	→
14. setTradingEnabled	→
15. setMarketingWallet	→
16. setMaxWalletAmount	→
17. setMaxWalletMode	→
18. setSellFees	→
19. setWhaleFee	→
20. transfer	→
21. transferFrom	→
22. transferOwnership	→
23. updateClaimWait	→
24. updateDividendTracker	→
25. updateGasForProcessing	→
26. updateUniswapV2Router	→

Deployer cannot pause the contract

Name	Exist	Tested	Verified
Deployer cannot pause	✓	✓	✓

1. approve	→
2. blacklistAddress	→
3. claim	→
4. decreaseAllowance	→
5. excludeFromDividends	→
6. excludeFromFees	→
7. excludeFromMaxWallet	→
8. increaseAllowance	→
9. processDividendTracker	→
10. recover	→
11. renounceOwnership	→
12. setAutomatedMarketMakerPair	→
13. setBuyFees	→
14. setIsTradingEnabled	→
15. setMarketingWallet	→
16. setMaxWalletAmount	→
17. setMaxWalletMode	→
18. setSellFees	→
19. setWhaleFee	→
20. transfer	→
21. transferFrom	→
22. transferOwnership	→
23. updateClaimWait	→
24. updateDividendTracker	→
25. updateGasForProcessing	→
26. updateUniswapV2Router	→

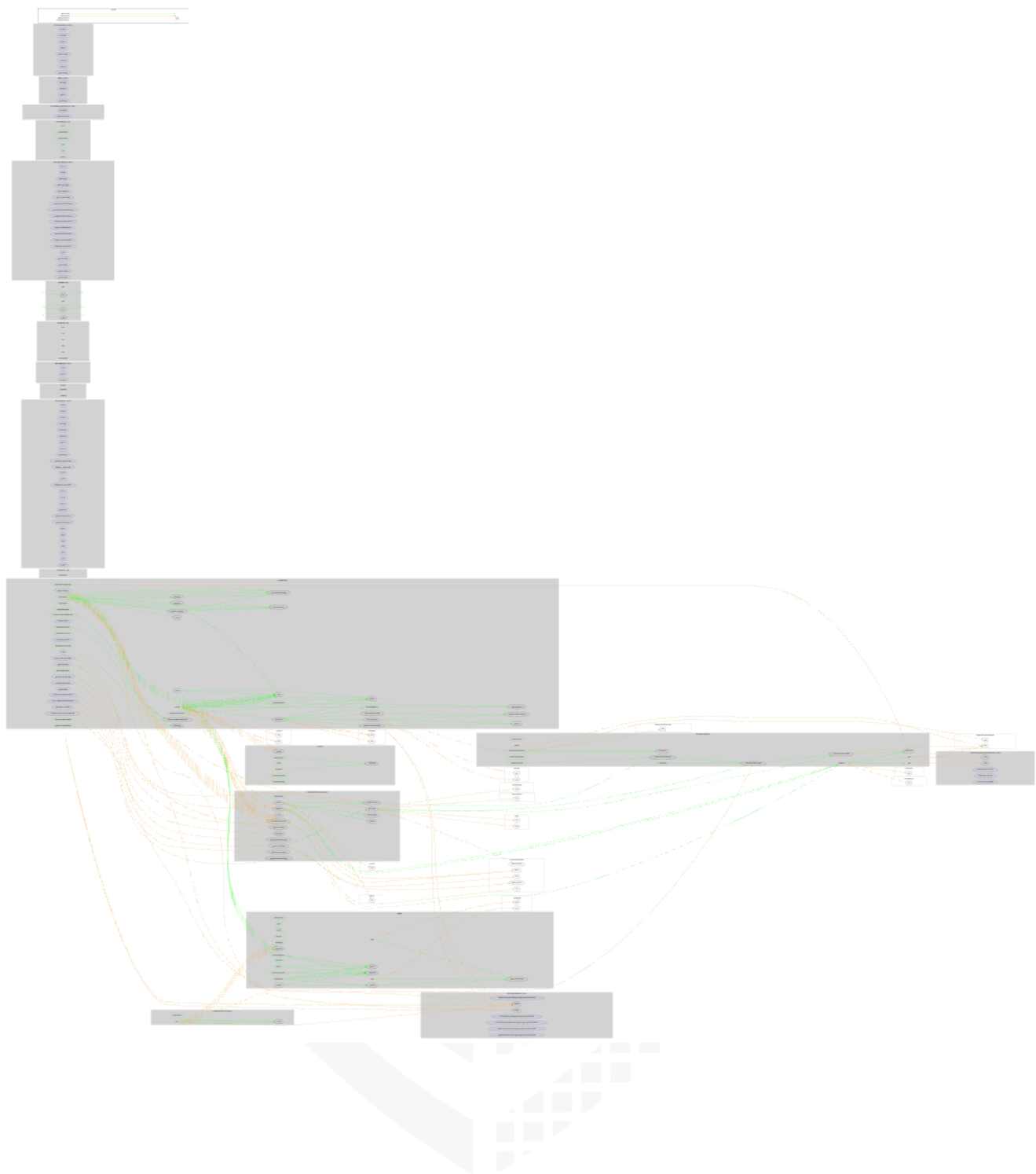
Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓



























Legend

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

CallGraph



Source Units in Scope

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
	contracts/DividendPayingToken.sol	1	————	173	173	87	51	76	
	contracts/SafeMathUint.sol	1	————	15	15	8	5	3	————
	contracts/IUniswapV2Pair.sol	————	1	54	9	5	1	55	————
	contracts/Context.sol	1	————	24	24	10	12	1	————
	contracts/IERC20Metadata.sol	————	1	27	16	4	15	9	
	contracts/SafeMathInt.sol	1	————	92	92	33	47	16	————
	contracts/CoinBanditTransferHelper.sol	1	————	35	35	23	4	37	
	contracts/CoinBandit.sol	2	————	637	628	442	33	436	
	contracts/SafeMath.sol	1	————	146	146	39	93	10	
	contracts/IUniswapV2Router.sol	————	2	142	7	4	2	64	
	contracts/Ownable.sol	1	————	57	57	27	21	25	————
	contracts/IterableMapping.sol	1	————	63	63	49	2	19	————
	contracts/ERC20.sol	1	————	310	294	85	178	82	————
	contracts/DividendPayingTokenOptionalInterface.sol	————	1	25	13	3	14	7	————
	contracts/DividendPayingTokenInterface.sol	————	1	36	13	3	16	5	————
	contracts/IERC20.sol	————	1	81	26	17	57	13	
	contracts/IUniswapV2Factory.sol	————	1	19	8	4	1	17	————
	Totals	11	8	1936	1619	843	552	875	

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

Issue	File	Type	Line	Description
#1	Coinba ndit.sol	A floating pragma is set	15	The current pragma Solidity directive is ""^0.6.2"".
#4	Coinba ndit.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

Informational issues

Issue	File	Type	Line	Description
#1	Coinba ndit.sol	State variables that could be declared constant (constable-states)	53, 51, 55,	Add the `constant` attributes to state variables that never change.
#2	Coinba nditDivi dendTra cker.sol	State variables that could be declared constant (constable-states)	467	Add the `constant` attributes to state variables that never change.

Audit Comments

19. August 2021:

- There is still an owner (Owner still has not renounced ownership)
- When maxWalletMode is Hard, receiver is not automatedMrketMakerPairs and not excluded and the owner set _maxWalletAmount to 0 you are not allowed to transfer
- Deployer can set isTradingEnabled to true/false
 - You are not allowed to use transfer function



SWC Attacks

ID	Title	Relationships	Status
SW C-13 6	Unencrypted Private Data On-Chain	CWE-767: Access to Critical Private Variable via Public Method	PASSED
SW C-13 5	Code With No Effects	CWE-1164: Irrelevant Code	PASSED
SW C-13 4	Message call with hardcoded gas amount	CWE-655: Improper Initialization	PASSED
SW C-13 3	Hash Collisions With Multiple Variable Length Arguments	CWE-294: Authentication Bypass by Capture-replay	PASSED
SW C-13 2	Unexpected Ether balance	CWE-667: Improper Locking	PASSED
SW C-13 1	Presence of unused variables	CWE-1164: Irrelevant Code	PASSED
SW C-13 0	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	PASSED
SW C-12 9	Typographical Error	CWE-480: Use of Incorrect Operator	PASSED
SW C-12 8	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	PASSED

SW C-12 7	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	PASSED
SW C-12 5	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	PASSED
SW C-12 4	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	PASSED
SW C-12 3	Requirement Violation	CWE-573: Improper Following of Specification by Caller	PASSED
SW C-12 2	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	PASSED
SW C-12 1	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	PASSED
SW C-12 0	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	PASSED
SW C-11 9	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	PASSED
SW C-11 8	Incorrect Constructor Name	CWE-665: Improper Initialization	PASSED
SW C-11 7	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	PASSED

SW C-11 6	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SW C-11 5	Authorization through tx.origin	CWE-477: Use of Obsolete Function	PASSED
SW C-11 4	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	PASSED
SW C-11 3	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	PASSED
SW C-11 2	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SW C-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	PASSED
SW C-11 0	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	PASSED
SW C-10 9	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	PASSED
SW C-10 8	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED
SW C-10 7	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	PASSED
SW C-10 6	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	PASSED

SW C-10 5	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	PASSED
SW C-10 4	Unchecked Call Return Value	CWE-252: Unchecked Return Value	PASSED
SW C-10 3	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	NOT PASSED
SW C-10 2	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	PASSED
SW C-10 1	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	PASSED
SW C-10 0	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED

The logo features the words "Solid Proofed" in a white, elegant script font. The word "Solid" is positioned above "Proofed". Behind the text is a faint, stylized shield emblem with a grid-like pattern, rendered in a darker shade of blue. The entire composition is set against a solid blue background.

Solid
Proofed

Blockchain Security | Smart Contract Audits | KYC

A small horizontal representation of the German flag, consisting of three equal-width horizontal stripes of black, red, and gold.

MADE IN GERMANY