SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

Audit

Security Assessment
17. August, 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	21. July 2021	• Layout project • Automated- /Manual-Security Testing • Summary
1.1	17. August 2021	• Recheck new contract

Network
Binance Smart Chain (BEP20)

Website
https://www.lildogecoin.dog/

Telegram
https://t.me/lildoge_coin

Twitter
https://twitter.com/lildoge_coin

Youtube
https://www.youtube.com/results?search_query=lil+doge

Github
https://github.com/lildogecoin

Medium
https://medium.com/@lildogecoin

Reddit
https://www.reddit.com/r/lildogecoin/

Instagram
https://www.instagram.com/lildoge_coin/

Discord
https://discord.gg/27t6Bxgz69

Description

Lil Doge is very giving and is designed to reward long term holders. Community is first and foremost. Lil Doge is proud to run with DOGE and provide transparency and fairness to the community.

Project Engagement

During the 19th of July 2021, **LilDoge 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. **LilDoge Team** provided Solidproof.io with access to their code repository and whitepaper.

During the 16th of August 2021, **LilDoge Team** engaged Solidproof.io to reaudit their smart contract.

Logo

Contract Link
v1.0
https://bscscan.com/address/
0xa638e525abb8aa42d0aeadd9b17f497c702902b0#code

v1.1
https://bscscan.com/address/
0x329861d96c862e879142a19c5c6ed8a046f25ea3#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 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
    - SafeMath
- 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

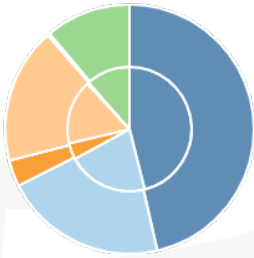| File Name	SHA-1 Hash
contracts/lildoge.sol	01a9fbd751b953a5e7e8b31a434f46a7f8246c7f

v1.1

File Name	SHA-1 Hash
contracts/lildoge.sol	f9cb27ae36939150375f67786bd79951192c417d

Metrics

Source Lines

v1.0

v1.1

Risk Level

v1.0

v1.1
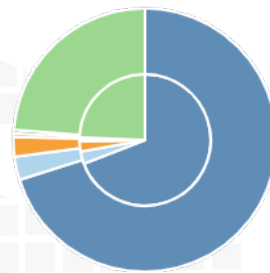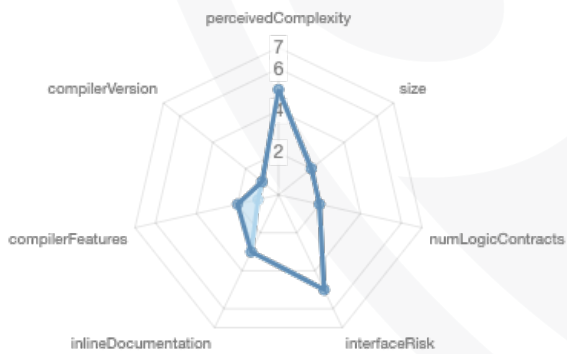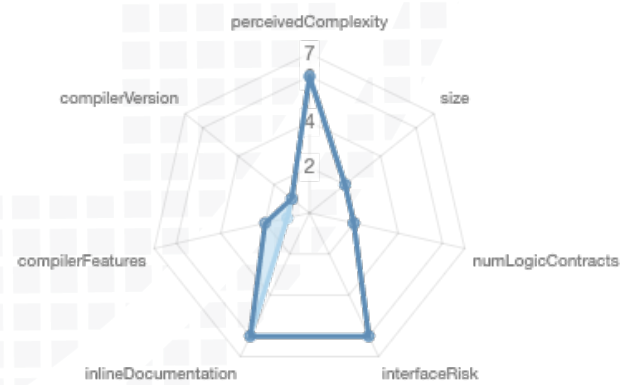
Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	2	2	5	1
1.1	4	4	7	2

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	100	5
1.1	148	5

Version	External	Internal	Private	Pure	View
1.0	69	86	22	19	44
1.1	112	137	9	31	57

State Variables

Version	Total	Public
1.0	26	8
1.1	48	32

Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	`^0.6.12`		yes	yes (2 asm blocks)	
1.1	`^0.8.4`		yes	**** (0 asm blocks)	

| Version	Transfers ETH	Low-Level Calls	DelegateCall	Uses Hash Functions	ECRecover	New/ Create/ Create 2
1.0	yes					
1.1	yes					yes → `New Contract:DogeDividendTracker`

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).
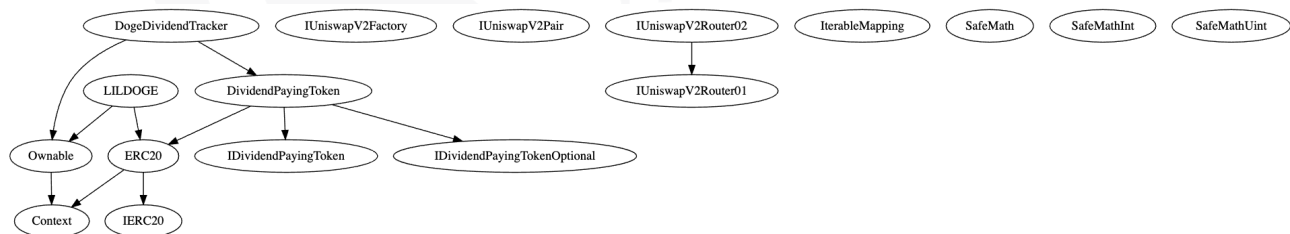
We will verify the following claims:
1. Correct implementation of Token standard
2. Deployer cannot mint any new tokens
3. Deployer cannot burn or lock user funds
4. Deployer cannot pause the contract
5. Overall checkup (Smart Contract Security)

Inheritance Graph
v1.0
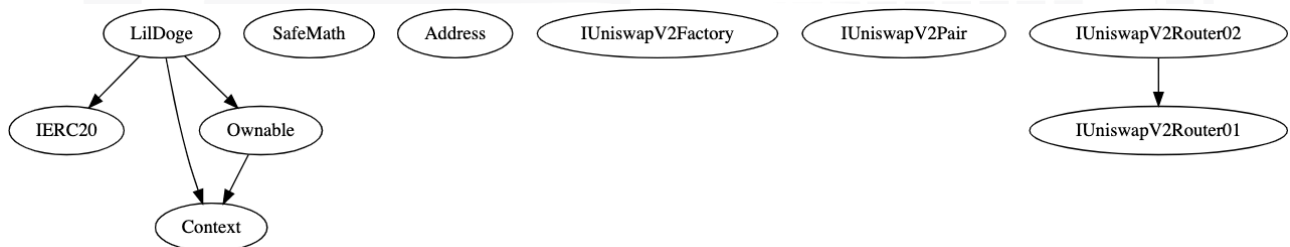
v1.1

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

```solidity
constructor() ERC20("Lil Doge Coin", "LILDOGE") {
    dogeDividendTracker = new DogeDividendTracker();

    marketingWallet = 0xe2A4c688EA049F5Ef2D4306C9C2DA1ebb2FEc84a;
    dogeDividendToken = 0xbA2aE424d960c26247Dd6c32edC70B295c744C43;

    IUniswapV2Router02 _uniswapV2Router = IUniswapV2Router02(0x10ED43C718714eb63d5aA57B78B54704E256024E);
     // 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);

    excludeFromDividend(address(dogeDividendTracker));
    excludeFromDividend(address(this));
    excludeFromDividend(address(_uniswapV2Router));
    excludeFromDividend(deadAddress);

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

    setAuthOnDividends(owner());

    /*
        _mint is an internal function in ERC20.sol that is only called here,
        and CANNOT be called ever again
    */
    _mint(owner(), 100000000000 * (10**18));
}
```

```solidity
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	✓	✓	✓

v1.1

1. approve →

2. claim →

3. decreaseAllowance →

4. excludeFromDividend →

5. excludeFromFees →

6. excludeMultipleAccountsFromFees →

7. increaseAllowance →

8. prepareForPartherOrExchangeListing →

9. processDividendTracker →

10. renounceOwnership →

11. setAuthOnDividends →

12. setAutomatedMarketMakerPair →

13. setDogeDividendEnabled →

14. setMarketingEnabled →

15. setMaxBuyTransaction →

16. setMaxSellTransaction →

17. setSellTransactionMultiplier →

18. setSwapAndLiquifyEnabled →

19. setSwapTokensAtAmount →

20. setTradingIsEnabled →

21. setWalletBalance →

22. transfer →

23. transferFrom →

24. transferOwnership →

25. updateClaimWait →

26. updateDogeDividendRewardFee →

27. updateDogeDividendToken →

28. updateDogeDividendTracker →

29. updateGasForProcessing →

30. updateLiquidityFee →

31. updateMarketingFee →

32. updateMarketingWallet →

33. updateMinimumBalanceForDividends →

34. updateUniswapV2Router →

35. whitelistDxSale →

Browse source code

Deployer cannot pause the contract

Name	Exist	Tested	Verified
Deployer cannot pause	✓	✓	✓

v1.1

1. approve →

2. claim →

3. decreaseAllowance →

4. excludeFromDividend →

5. excludeFromFees →

6. excludeMultipleAccountsFromFees →

7. increaseAllowance →

8. prepareForPartherOrExchangeListing →

9. processDividendTracker →

10. renounceOwnership →

11. setAuthOnDividends →

12. setAutomatedMarketMakerPair →

13. setDogeDividendEnabled →

14. setMarketingEnabled →

15. setMaxBuyTransaction →

16. setMaxSellTransaction →

17. setSellTransactionMultiplier →

18. setSwapAndLiquifyEnabled →

19. setSwapTokensAtAmount →

20. setTradingIsEnabled →

21. setWalletBalance →

22. transfer →

23. transferFrom →

24. transferOwnership →

25. updateClaimWait →

26. updateDogeDividendRewardFee →

27. updateDogeDividendToken →

28. updateDogeDividendTracker →

29. updateGasForProcessing →

30. updateLiquidityFee →

31. updateMarketingFee →

32. updateMarketingWallet →

33. updateMinimumBalanceForDividends →

34. updateUniswapV2Router →
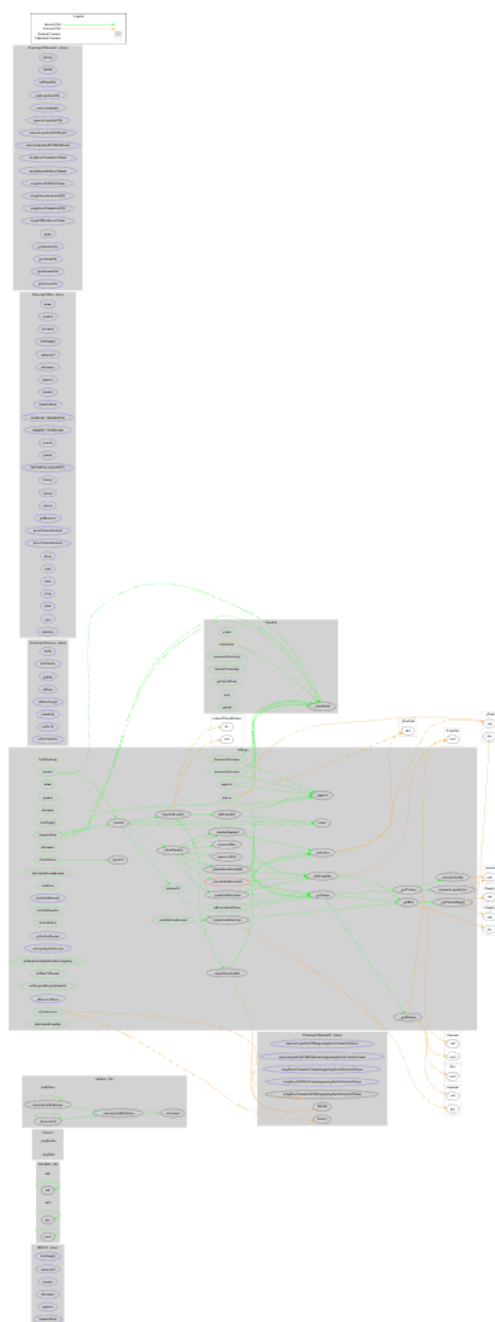
35. whitelistDxSale →

Browse source code

Overall checkup (Smart Contract Security)

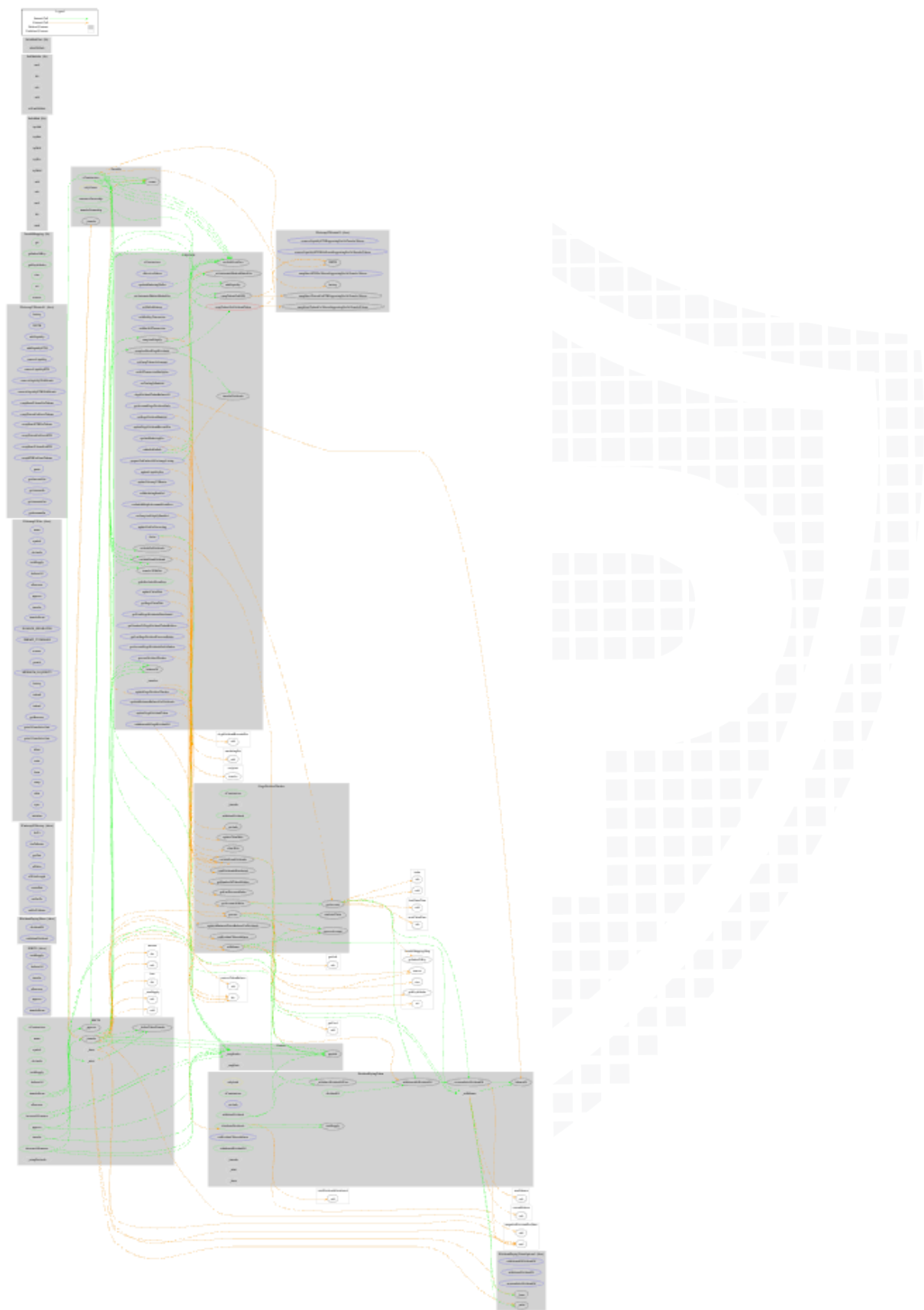| Tested	Verified
✓	✓

Legend

| Attribute	Symbol
Verfified / Checked	✓
Partly Verified	🚩
Unverified / Not checked	✘
Not available	–

CallGraph
v1.0

v1.1

Source Units in Scope
v1.0

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝📚🔍🐍	contracts/lildoge.sol	5	5	1165	885	522	314	528	🖥️💰📥☀️
📝📚🔍🐍	**Totals**	**5**	**5**	**1165**	**885**	**522**	**314**	**528**	🖥️💰📥☀️

v1.1

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝📚🔍🐍	contracts/lildoge.sol	10	7	1411	1235	889	42	886	💰📥🌀☀️
📝📚🔍🐍	**Totals**	**10**	**7**	**1411**	**1235**	**889**	**42**	**886**	💰📥🌀☀️

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

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	Main	A floating pragma is set	7	The current pragma Solidity directive is ""^0.8.4"".
#2	Main	State variable visibility is not set	225	

Informational issues
- no informational issues found -

Audit Comments
17. August 2021:
- There is still an owner (Owner still has not renounced ownership)
- Deployer can enable/disable „tradingIsEnabled" variable
  - When condition „tradingIsEnable" equals to false, you are not allowed to transfer or when from or to is excluded from Fees you can transfer

```
1020          require(tradingIsEnabled || (isExcludedFromFees[from] || isExcludedFromFees[to]), "LILDOGE: Trading has not started yet");
```

- Deployer can setBuyTransactionAmount/setMaxSellTransaction

```
if (
    tradingIsEnabled &&
    automatedMarketMakerPairs[from] &&
    !excludedAccount
) {
    require(amount <= maxBuyTranscationAmount, "Transfer amount exceeds the maxTxAmount.");

} else if (
    tradingIsEnabled &&
    automatedMarketMakerPairs[to] &&
    !excludedAccount
) {
    require(amount <= maxSellTransactionAmount, "Sell transfer amount exceeds the maxSellTransactionAmount.");
}
```

- Deployer can setSwapAndLiquifyEnabled
- Deployer can setSwapTokenAtAmount

```
if(!automatedMarketMakerPairs[to] && tradingIsEnabled && !excludedAccount){
    require(balanceOf(to).add(amount) <= maxWalletBalance, 'Wallet balance is exceeding maxWalletBalance');
}
```

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**

27

[SWC-116](#)	Timestamp Dependence	[CWE-829: Inclusion of Functionality from Untrusted Control Sphere](#)	**PASSED**
[SWC-115](#)	Authorization through tx.origin	[CWE-477: Use of Obsolete Function](#)	**PASSED**
[SWC-114](#)	Transaction Order Dependence	[CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')](#)	**PASSED**
[SWC-113](#)	DoS with Failed Call	[CWE-703: Improper Check or Handling of Exceptional Conditions](#)	**PASSED**
[SWC-112](#)	Delegatecall to Untrusted Callee	[CWE-829: Inclusion of Functionality from Untrusted Control Sphere](#)	**PASSED**
[SWC-111](#)	Use of Deprecated Solidity Functions	[CWE-477: Use of Obsolete Function](#)	**PASSED**
[SWC-110](#)	Assert Violation	[CWE-670: Always-Incorrect Control Flow Implementation](#)	**PASSED**
[SWC-109](#)	Uninitialized Storage Pointer	[CWE-824: Access of Uninitialized Pointer](#)	**PASSED**
[SWC-108](#)	State Variable Default Visibility	[CWE-710: Improper Adherence to Coding Standards](#)	**NOT PASSED**
[SWC-107](#)	Reentrancy	[CWE-841: Improper Enforcement of Behavioral Workflow](#)	**PASSED**
[SWC-106](#)	Unprotected SELFDESTRUCT Instruction	[CWE-284: Improper Access Control](#)	**PASSED**

SWC-105	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	PASSED
SWC-104	Unchecked Call Return Value	CWE-252: Unchecked Return Value	PASSED
SWC-103	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	NOT 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

Solid Proofed

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY