TECHRIGHT

Security Assessment

Dotlab - Audit

TechRight Verified on 05 May, 2023

Table of contents

Disclaimer

TechRight.io Reports do not constitute an endorsement or disapproval of any specific project or team, and they should not be taken as an indication of the economic value of any product or asset created by a team. Additionally, TechRight.io does not perform testing or auditing of integration with external contracts or services like Unicrypt, Uniswap, PancakeSwap, and others.

TechRight.io Audits do not offer any assurance or pledge about the complete absence of bugs in the evaluated technology, and they do not give any hint about the owners of the technology. These audits should not be relied upon to make any investment or participation decisions in any specific project, nor should they be used as any form of investment advice.

TechRight.io Reports involve a comprehensive auditing process to support our clients in enhancing their code quality while reducing the risk associated with blockchain technology and cryptographic assets. Please note that every company and individual is responsible for conducting their own due diligence and maintaining continuous security. Please note that TechRight does not guarantee the security or functionality of the technology we confirm to evaluate.

Description

Network

Arbitrum

Website

https://www.dotlab.app

Telegram

https://t.me/Dotlabofficial

Twitter

https://twitter.com/dotlabofficial

DApp

https://ans.dotlab.app

Whitepaper

https://dotlab.gitbook.io/dotlab-whitepaper

Zealy (Crew3)

https://dotlab.zealy.io

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 has informational character but is not affecting any of the code.	An observation that does not determine a level of risk

Auditing Strategy and Techniques Applied

During the evaluation process, the repository was thoroughly examined to identify any security-related concerns, assess code quality, and ensure adherence to specifications and best practices. Our team of expert pentesters and smart contract developers reviewed the code line-by-line and documented any issues identified.

Methodology

The auditing process follows a step-by-step routine:

1. Code review that includes:
   i. Review of the specifications, sources and instructions provided to TechRight to ensure a thorough understanding of the size, scope, and functionality of the smart contract's.

   ii. Manual review of code, which involves carefully reading the source code line-by-line to identify potential vulnerabilities.

   iii. Comparison to specification, which is the process of confirming whether the code performs as described in the specifications, sources, and instructions provided.

2. Testing and automated analysis that includes the following:
   i. Test coverage analysis, which involves assessing the degree to which test cases cover the code and how much of the code is executed while running those test cases.

   ii. Symbolic execution, which refers to the analysis of a program to identify the inputs that trigger each component of the program to execute.

3. Best practices review, which involves evaluating smart contracts to enhance efficiency, effectiveness, clarity, maintainability, security, and control in accordance with industry and academic practices, recommendations, and research.

4. Specific, itemized, actionable recommendations that enable you to take necessary measures to secure your smart contracts.

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

Scope

This section lists files that are in scope for the metrics report.

- **Project:** `Dotlab`
- **Included Files:**
  - ``
- **Excluded Paths:**
  - ``
- **File Limit:** `undefined`
  - **Exclude File list Limit:** `undefined`
- **Workspace Repository:** `unknown` (`undefined` @ `undefined`)

Source Units in Scope

Source Units Analyzed: **`1`**
Source Units in Scope: **`1`** (**100%**)

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝🔍	StakeDTL.sol	1	1	255	249	176	12	183	🔼
📝🔍	**Totals**	**1**	**1**	**255**	**249**	**176**	**12**	**183**	🔼

Legend:

- **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, ...)

Out of Scope

Excluded Source Units

Source Units Excluded: **`0`**

File
None

Duplicate Source Units

Duplicate Source Units Excluded: **`0`**

File
None

Doppelganger Contracts

Doppelganger Contracts: **`0`**

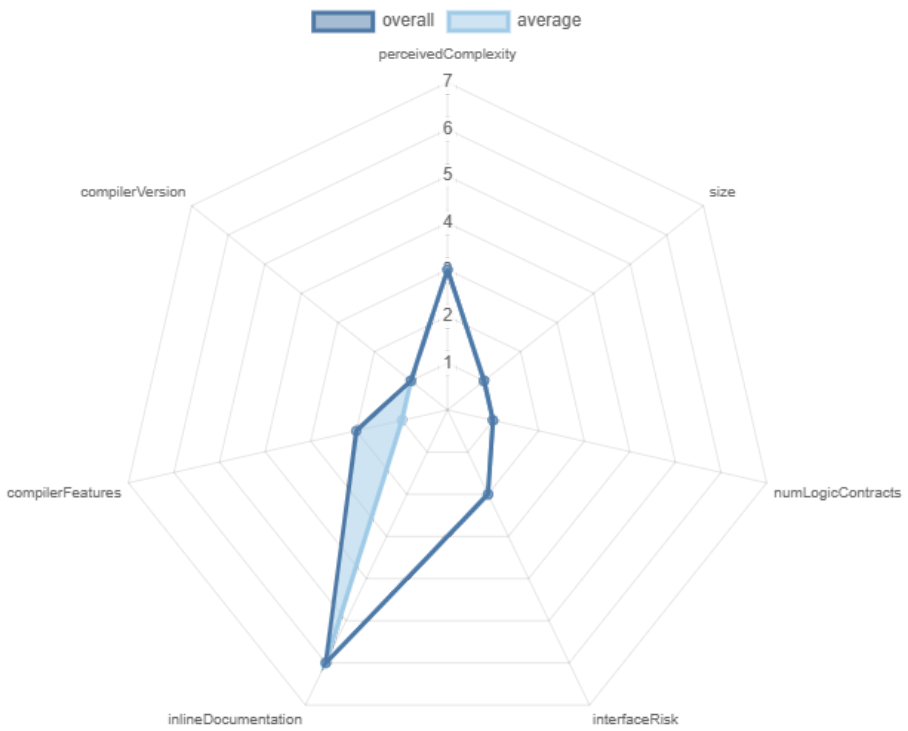
File	Contract	Doppelganger

Report

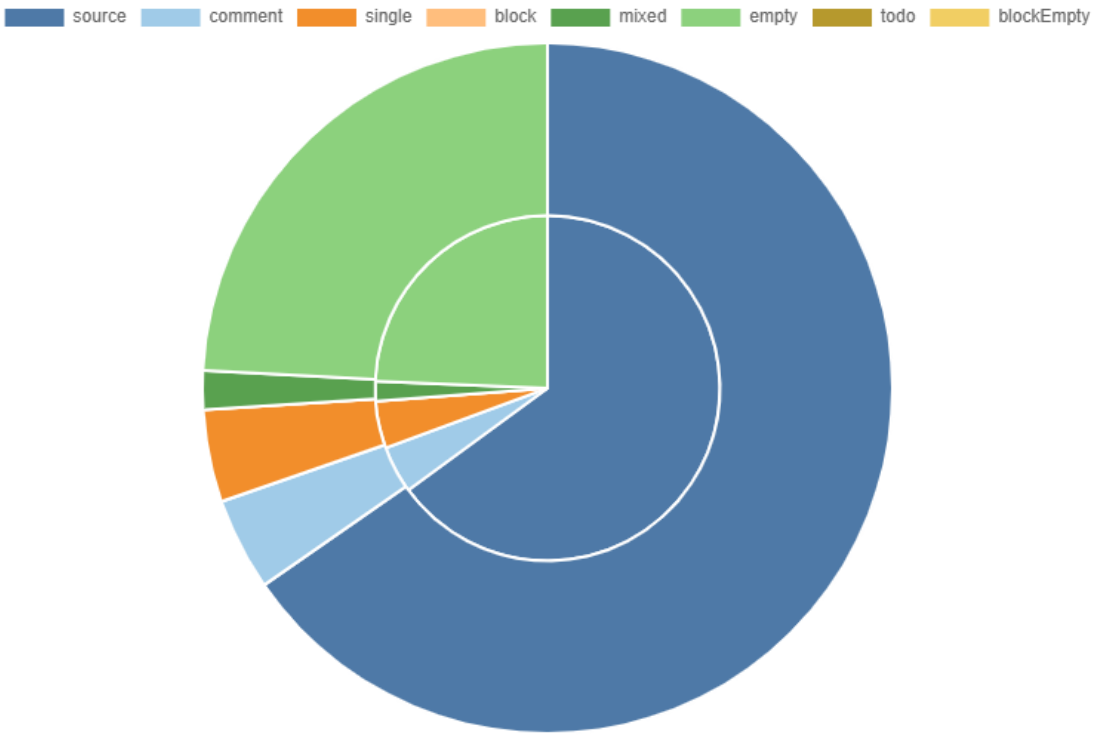
Overview

The analysis finished with `0` errors and `0` duplicate files.

Risk



Source Lines (sloc vs. nsloc)



Inline Documentation

- **Comment-to-Source Ratio:** On average there are `15.08` code lines per comment (lower=better).

- **ToDo's:** `0`

Components

📝Contracts	📚Libraries	🔍Interfaces	🎨Abstract
1	0	1	0

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
16	0

External	Internal	Private	Pure	View
16	16	0	1	4

StateVariables

Total	🌐Public
15	11

Capabilities

Solidity Versions observed	🧪 Experimental Features	💰 Can Receive Funds	📟 Uses Assembly	💣 Has Destroyable Contracts
`^0.8.18`		—	—	—

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

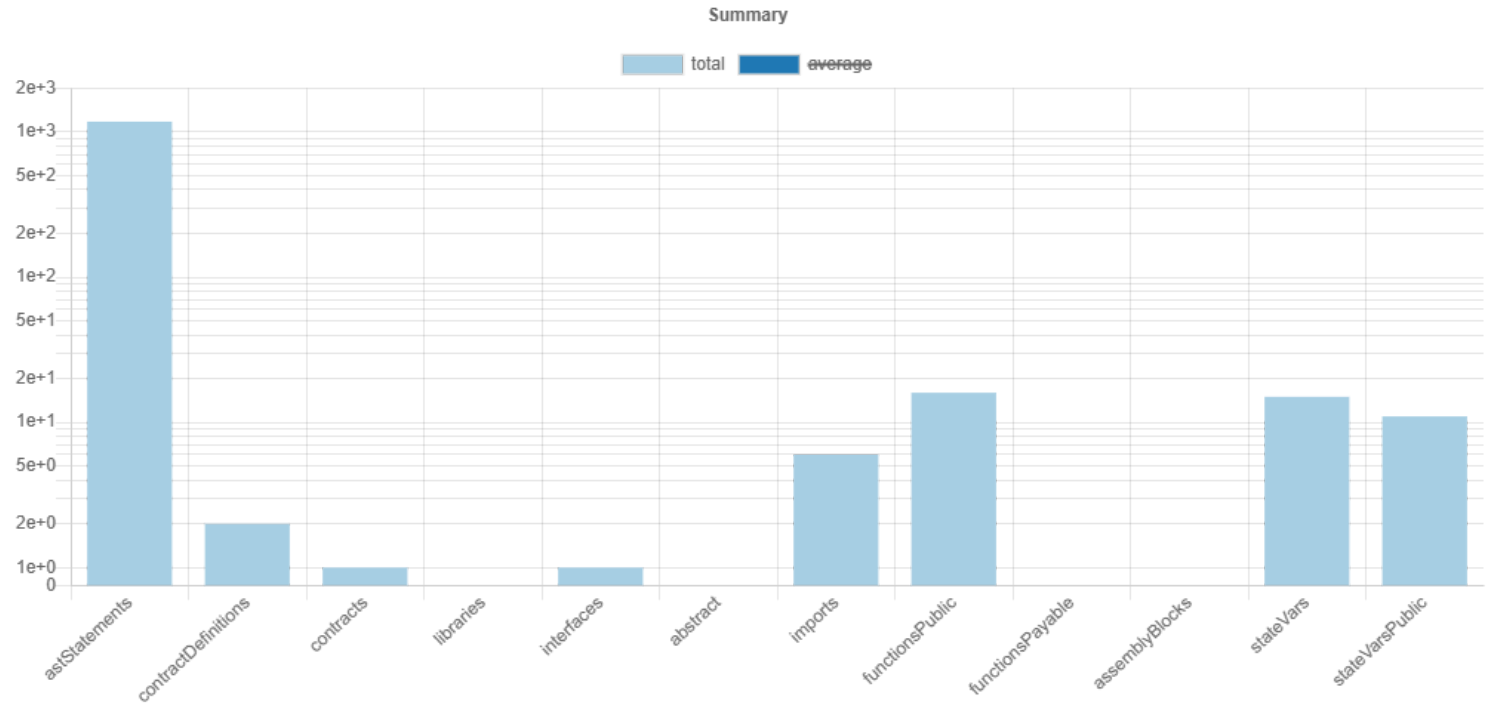
♻ TryCatch	Σ Unchecked
—	—

Dependencies / External Imports

Dependency / Import Path	Count
@openzeppelin/contracts/access/Ownable.sol	1
@openzeppelin/contracts/security/Pausable.sol	1
@openzeppelin/contracts/security/ReentrancyGuard.sol	1
@openzeppelin/contracts/utils/Context.sol	1
@openzeppelin/contracts/utils/math/SafeMath.sol	1

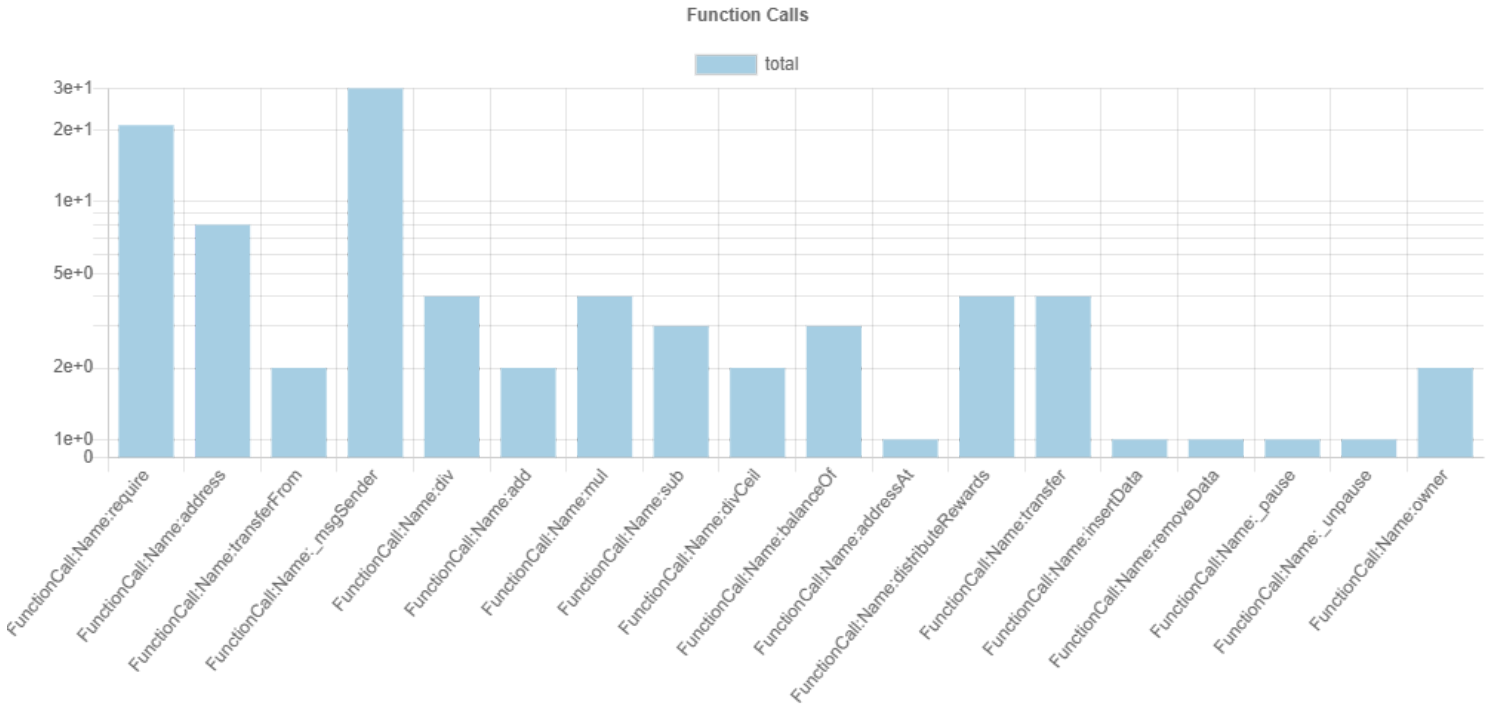
Totals

Summary

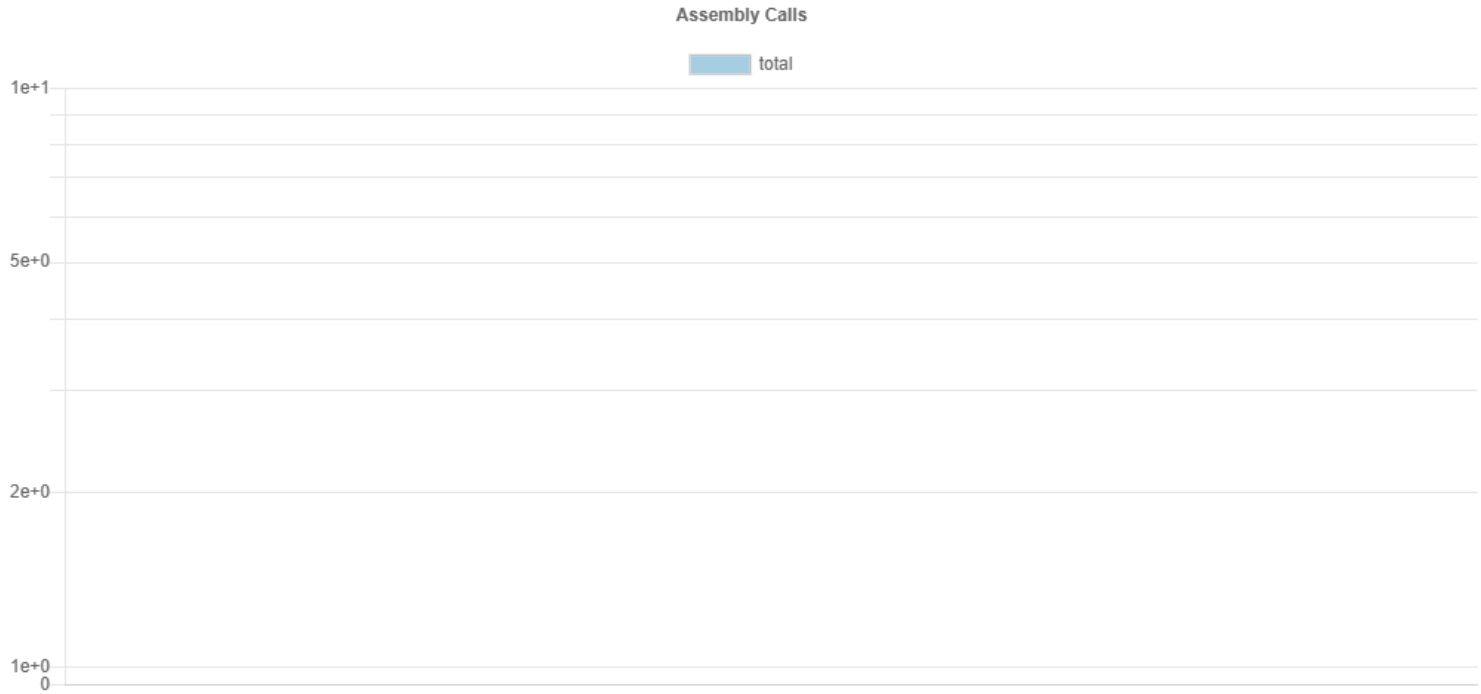


AST Node Statistics

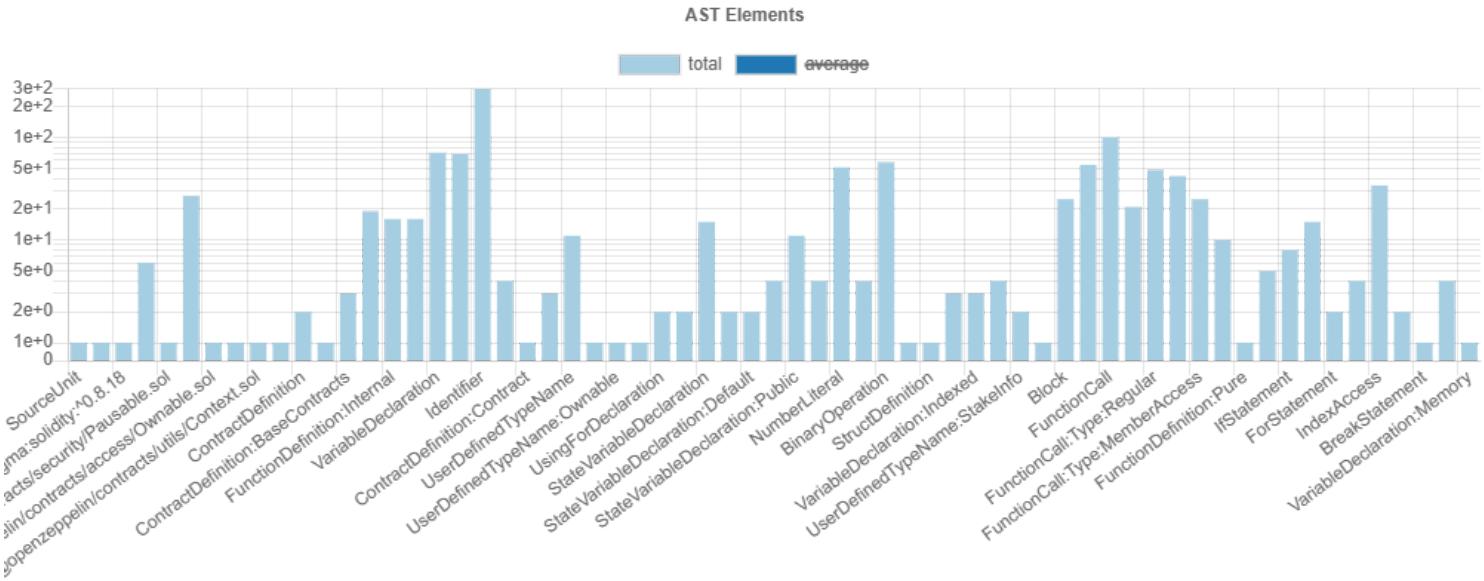
Function Calls



Assembly Calls



AST Total



Inheritance Graph

Contract Summary

Sūrya's Description Report Files Description Table

File Name	SHA-1 Hash
StakeDTL.sol	fa536a0dc5bb30b918816e6b026b4a2ff0bd87f8

Contracts Description Table

Contract	Type	Bases		
L	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
Token	Interface			
L	transfer	External ❗	🛑	NO❗
L	balanceOf	External ❗		NO❗
L	transferFrom	External ❗	🛑	NO❗
L	approve	External ❗	🛑	NO❗
StakeDTL	Implementation	Pausable, Ownable, ReentrancyGuard		
L		Public ❗	🛑	NO❗
L	addReward	External ❗	🛑	onlyOwner
L	divCeil	Internal 🔒		
L	distributeRewards	Internal 🔒	🛑	
L	claimAllRewards	External ❗	🛑	nonReentrant
L	stakeToken	External ❗	🛑	whenNotPaused nonReentrant
L	unstake	External ❗	🛑	nonReentrant
L	getStakeInstances	External ❗		NO❗
L	getTokenExpiry	External ❗		NO❗
L	pause	External ❗	🛑	onlyOwner
L	unpause	External ❗	🛑	onlyOwner
L	getClaimedRewards	External ❗		NO❗
L	distributeRewardsPublic	External ❗	🛑	onlyOwner
L	setRewardPercentage	External ❗	🛑	onlyOwner
L	emergencyWithdraw	External ❗	🛑	onlyOwner

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

Detectors Issue

Description	Check	Impact	Confidence
StakeDTL.distributeRewards() (contracts/stakedlt/StakeDTL.sol#87-128) performs a multiplication on the result of a division: - rewardsForTwentyFourHours = totalRewards.mul(rewardPercentage).div(100) (contracts/stakedlt/StakeDTL.sol#98) - rewardsToDistribute = divCeil(rewardsForTwentyFourHours.mul(elapsedTime),86400).div(PRECISION) (contracts/stakedlt/StakeDTL.sol#99)	divide-before-multiply	Medium	Medium
Reentrancy in StakeDTL.stakeToken(uint256,uint8) (contracts/stakedlt/StakeDTL.sol#147-174): External calls: - require(bool,string) (dtlToken.transferFrom(*msgSender(),address(this),stakeAmount),Token transfer failed!) (contracts/stakedlt/StakeDTL.sol#152) *State variables written after the call(s):* - *stakeInfos[*msgSender()].push(StakeInfo(block.timestamp,block.timestamp + lockPeriods[lockPeriodIndex],stakeAmount,shares,lockPeriodIndex)) (contracts/stakedlt/StakeDTL.sol#158-164) StakeDTL.stakeInfos (contracts/stakedlt/StakeDTL.sol#50) can be used in cross function reentrancies: - StakeDTL.distributeRewards() (contracts/stakedlt/StakeDTL.sol#87-128) - StakeDTL.getStakeInstances(address) (contracts/stakedlt/StakeDTL.sol#215-217) - StakeDTL.getTokenExpiry(uint256) (contracts/stakedlt/StakeDTL.sol#219-222) - StakeDTL.stakeInfos (contracts/stakedlt/StakeDTL.sol#50) - totalShares += shares (contracts/stakedlt/StakeDTL.sol#155) StakeDTL.totalShares (contracts/stakedlt/StakeDTL.sol#30) can be used in cross function reentrancies: - StakeDTL.constructor(Token,Token) (contracts/stakedlt/StakeDTL.sol#58-69) - StakeDTL.distributeRewards() (contracts/stakedlt/StakeDTL.sol#87-128) - StakeDTL.totalShares (contracts/stakedlt/StakeDTL.sol#30) - totalStakers += 1 (contracts/stakedlt/StakeDTL.sol#168) StakeDTL.totalStakers (contracts/stakedlt/StakeDTL.sol#32) can be used in cross function reentrancies: - StakeDTL.distributeRewards() (contracts/stakedlt/StakeDTL.sol#87-128) - StakeDTL.totalStakers (contracts/stakedlt/StakeDTL.sol#32) - userTotalShares[_msgSender()] += shares (contracts/stakedlt/StakeDTL.sol#156) StakeDTL.userTotalShares (contracts/stakedlt/StakeDTL.sol#51) can be used in cross function reentrancies: - StakeDTL.distributeRewards() (contracts/stakedlt/StakeDTL.sol#87-128) - StakeDTL.userTotalShares (contracts/stakedlt/StakeDTL.sol#51)	reentrancy-no-eth	Medium	Medium
Reentrancy in StakeDTL.unstake(uint256) (contracts/stakedlt/StakeDTL.sol#176-213): External calls: - require(bool,string)(dtlToken.transfer(*msgSender(),stakeAmount),Token transfer failed!) (contracts/stakedlt/StakeDTL.sol#185) *- require(bool,string) (rewardToken.transfer(*msgSender(),stakerUnclaimedRewards),Token transfer failed!) (contracts/stakedlt/StakeDTL.sol#205) State variables written after the call(s): - unclaimedRewards[_msgSender()] = 0 (contracts/stakedlt/StakeDTL.sol#208) StakeDTL.unclaimedRewards (contracts/stakedlt/StakeDTL.sol#52) can be used in cross function reentrancies: - StakeDTL.distributeRewards() (contracts/stakedlt/StakeDTL.sol#87-128) - StakeDTL.unclaimedRewards (contracts/stakedlt/StakeDTL.sol#52)	reentrancy-no-eth	Medium	Medium
Reentrancy in StakeDTL.claimAllRewards() (contracts/stakedlt/StakeDTL.sol#131-142): External calls: - require(bool,string) (rewardToken.transfer(*msgSender(),stakerUnclaimedRewards),Token transfer failed!) (contracts/stakedlt/StakeDTL.sol#137) *State variables written after the call(s):* - *unclaimedRewards[*msgSender()] = 0 (contracts/stakedlt/StakeDTL.sol#140) StakeDTL.unclaimedRewards (contracts/stakedlt/StakeDTL.sol#52) can be used in cross function reentrancies: - StakeDTL.distributeRewards() (contracts/stakedlt/StakeDTL.sol#87-128) - StakeDTL.unclaimedRewards (contracts/stakedlt/StakeDTL.sol#52)	reentrancy-no-eth	Medium	Medium

Description	Check	Impact	Confidence
Reentrancy in StakeDTL.unstake(uint256) (contracts/stakedlt/StakeDTL.sol#176-213): External calls: - require(bool,string)(dtlToken.transfer(*msgSender(),stakeAmount),Token transfer failed!) (contracts/stakedlt/StakeDTL.sol#185)* *State variables written after the call(s):* *- delete stakeInfos[*msgSender()]stakeIndex StakeDTL.stakeInfos (contracts/stakedlt/StakeDTL.sol#50) can be used in cross function reentrancies: - StakeDTL.distributeRewards() (contracts/stakedlt/StakeDTL.sol#87-128) - StakeDTL.getStakeInstances(address) (contracts/stakedlt/StakeDTL.sol#215-217) - StakeDTL.getTokenExpiry(uint256) (contracts/stakedlt/StakeDTL.sol#219-222) - StakeDTL.stakeInfos (contracts/stakedlt/StakeDTL.sol#50) - totalShares -= stakeShares (contracts/stakedlt/StakeDTL.sol#187) StakeDTL.totalShares (contracts/stakedlt/StakeDTL.sol#30) can be used in cross function reentrancies: - StakeDTL.constructor(Token,Token) (contracts/stakedlt/StakeDTL.sol#58-69) - StakeDTL.distributeRewards() (contracts/stakedlt/StakeDTL.sol#87-128) - StakeDTL.totalShares (contracts/stakedlt/StakeDTL.sol#30) - totalStakers -= 1 (contracts/stakedlt/StakeDTL.sol#198) StakeDTL.totalStakers (contracts/stakedlt/StakeDTL.sol#32) can be used in cross function reentrancies: - StakeDTL.distributeRewards() (contracts/stakedlt/StakeDTL.sol#87-128) - StakeDTL.totalStakers (contracts/stakedlt/StakeDTL.sol#32) - userTotalShares[_msgSender()] -= stakeShares (contracts/stakedlt/StakeDTL.sol#188) StakeDTL.userTotalShares (contracts/stakedlt/StakeDTL.sol#51) can be used in cross function reentrancies: - StakeDTL.distributeRewards() (contracts/stakedlt/StakeDTL.sol#87-128) - StakeDTL.userTotalShares (contracts/stakedlt/StakeDTL.sol#51)	reentrancy-no-eth	Medium	Medium
StakeDTL.setRewardPercentage(uint256) (contracts/stakedlt/StakeDTL.sol#240-243) should emit an event for: - rewardPercentage = newRewardPercentage (contracts/stakedlt/StakeDTL.sol#242)	events-maths	Low	Medium
StakeDTL.addReward(uint256) (contracts/stakedlt/StakeDTL.sol#72-75) should emit an event for: - totalRewards += amount (contracts/stakedlt/StakeDTL.sol#74)	events-maths	Low	Medium
Reentrancy in StakeDTL.addReward(uint256) (contracts/stakedlt/StakeDTL.sol#72-75): External calls: - require(bool,string) (rewardToken.transferFrom(_msgSender(),address(this),amount),Token transfer failed!) (contracts/stakedlt/StakeDTL.sol#73) State variables written after the call(s): - totalRewards += amount (contracts/stakedlt/StakeDTL.sol#74)	reentrancy-benign	Low	Medium
Reentrancy in StakeDTL.unstake(uint256) (contracts/stakedlt/StakeDTL.sol#176-213): External calls: - require(bool,string)(dtlToken.transfer(*msgSender(),stakeAmount),Token transfer failed!) (contracts/stakedlt/StakeDTL.sol#185)* *- require(bool,string) (rewardToken.transfer(*msgSender(),stakerUnclaimedRewards),Token transfer failed!) (contracts/stakedlt/StakeDTL.sol#205) State variables written after the call(s): - claimedRewards[_msgSender()] += stakerUnclaimedRewards (contracts/stakedlt/StakeDTL.sol#206)	reentrancy-benign	Low	Medium
Reentrancy in StakeDTL.claimAllRewards() (contracts/stakedlt/StakeDTL.sol#131-142): External calls: - require(bool,string) (rewardToken.transfer(*msgSender(),stakerUnclaimedRewards),Token transfer failed!) (contracts/stakedlt/StakeDTL.sol#137)* *State variables written after the call(s):* *- claimedRewards[*msgSender()] += stakerUnclaimedRewards (contracts/stakedlt/StakeDTL.sol#138)	reentrancy-benign	Low	Medium
Reentrancy in StakeDTL.emergencyWithdraw(uint256) (contracts/stakedlt/StakeDTL.sol#246-252): External calls:	reentrancy-events	Low	Medium

Description	Check	Impact	Confidence
- require(bool,string)(rewardToken.transfer(owner(),amount),Token transfer failed!) (contracts/stakedlt/StakeDTL.sol#250) Event emitted after the call(s): - EmergencyWithdraw(owner(),amount) (contracts/stakedlt/StakeDTL.sol#251)			
StakeDTL.unstake(uint256) (contracts/stakedlt/StakeDTL.sol#176-213) uses timestamp for comparisons Dangerous comparisons: - require(bool,string)(stakeInfos[_msgSender()][stakeIndex].endTS < block.timestamp,Stake Time is not over yet) (contracts/stakedlt/StakeDTL.sol#180)	timestamp	Low	Medium
StakeDTL.distributeRewards() (contracts/stakedlt/StakeDTL.sol#87-128) uses timestamp for comparisons Dangerous comparisons: - elapsedTime > 0 (contracts/stakedlt/StakeDTL.sol#96) - require(bool,string)(rewardToken.balanceOf(address(this)) >= rewardsToDistribute,Insufficient reward token balance) (contracts/stakedlt/StakeDTL.sol#101) - stakeInfos[staker][j].endTS < lastRewardDistribution (contracts/stakedlt/StakeDTL.sol#115)	timestamp	Low	Medium
Different versions of Solidity are used: - Version used: ['^0.8.0', '^0.8.18'] - ^0.8.0 (node*modules/@openzeppelin/contracts/access/Ownable.sol#4)* - *^0.8.0 (node*modules/@openzeppelin/contracts/security/Pausable.sol#4) - ^0.8.0 (node*modules/@openzeppelin/contracts/security/ReentrancyGuard.sol#4)* - *^0.8.0 (node*modules/@openzeppelin/contracts/utils/Context.sol#4) - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#4) - ^0.8.18 (contracts/stakedlt/MerkleTree.sol#2) - ^0.8.18 (contracts/stakedlt/StakeDTL.sol#1)	pragma	Informational	High
SafeMath.tryDiv(uint256,uint256) (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#64-69) is never used and should be removed	dead-code	Informational	Medium
SafeMath.tryMod(uint256,uint256) (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#76-81) is never used and should be removed	dead-code	Informational	Medium
SafeMath.sub(uint256,uint256,string) (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#168-177) is never used and should be removed	dead-code	Informational	Medium
SafeMath.tryAdd(uint256,uint256) (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#22-28) is never used and should be removed	dead-code	Informational	Medium
SafeMath.mod(uint256,uint256,string) (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#217-226) is never used and should be removed	dead-code	Informational	Medium
SafeMath.div(uint256,uint256,string) (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#191-200) is never used and should be removed	dead-code	Informational	Medium
Context.*msgData() (node*modules/@openzeppelin/contracts/utils/Context.sol#21-23) is never used and should be removed	dead-code	Informational	Medium
SafeMath.mod(uint256,uint256) (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#151-153) is never used and should be removed	dead-code	Informational	Medium
SafeMath.tryMul(uint256,uint256) (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#47-57) is never used and should be removed	dead-code	Informational	Medium
SafeMath.trySub(uint256,uint256) (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#35-40) is never used and should be removed	dead-code	Informational	Medium
MerkleTree.length(MerkleTree.Tree) (contracts/stakedlt/MerkleTree.sol#41-43) is never used and should be removed	dead-code	Informational	Medium

Description	Check	Impact	Confidence
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#4) allows old versions	solc-version	Informational	High
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/security/ReentrancyGuard.sol#4) allows old versions	solc-version	Informational	High
Pragma version^0.8.18 (contracts/stakedlt/MerkleTree.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16	solc-version	Informational	High
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/math/SafeMath.sol#4) allows old versions	solc-version	Informational	High
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/security/Pausable.sol#4) allows old versions	solc-version	Informational	High
solc-0.8.19 is not recommended for deployment	solc-version	Informational	High
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#4) allows old versions	solc-version	Informational	High
Pragma version^0.8.18 (contracts/stakedlt/StakeDTL.sol#1) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.16	solc-version	Informational	High
StakeDTL.dtlToken (contracts/stakedlt/StakeDTL.sol#24) should be immutable	immutable-states	Optimization	High

Summary

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL	OPTIMIZATION
Passed	**Passed**	5 Issues	8 Issues	20 Issues	1 issues

Owner privileges

No.	Issue	Description	Status
1	**No critical issues found**	The contract does not contain issues of high or medium criticality. This means that no known vulnerabilities were found in the source code.	**Passed**
2	**Contract owner cannot mint**	It is no possible to mint new tokens.	**Passed**
3	**Contract owner cannot blacklist addresses**	It is not possible to lock user funds by blacklisting addresses.	**Passed**
4	**Contract owner cannot set high fees**	The fees, if applicable, can be a maximum of 25% or lower. The contract can therefore not be locked. Please take a look in the comment section for more details.	**Passed**
5	**Contract owner cannot blacklist addresses**	It is not possible to lock user funds by blacklisting addresses	**Passed**
6	**Contract cannot be locked**	Owner cannot lock any user funds.	**Passed**