



Cyberscope

Audit Report

Dual Pools Trend Token

April 2023

Github <https://github.com/JavisJL/dualpools/tree/main/TrendTokenAudit>

Commit [ee03d8eeb0b412dd944ea4aeb61fbf26aa2aca5e](https://github.com/JavisJL/dualpools/commit/ee03d8eeb0b412dd944ea4aeb61fbf26aa2aca5e)

Audited by © cyberscope

Table of Contents

Table of Contents	1
Review	3
Audit Updates	3
Source Files	4
Findings Breakdown	7
Diagnostics	8
UIA - Unsafe Indexing Assumption	9
Description	9
Recommendation	9
Team Update	9
MT - Mints Tokens	10
Description	10
Recommendation	10
Team Update	10
IDI - Immutable Declaration Improvement	11
Description	11
Recommendation	11
L02 - State Variables could be Declared Constant	12
Description	12
Recommendation	12
Team Update	13
L04 - Conformance to Solidity Naming Conventions	14
Description	14
Recommendation	15
L07 - Missing Events Arithmetic	16
Description	16
Recommendation	16
L19 - Stable Compiler Version	17
Description	17
Recommendation	17
Team Update	17
Functions Analysis	19
Inheritance Graph	36
Flow Graph	37
Summary	38
Disclaimer	39
About Cyberscope	40

Review

Repository	https://github.com/JavisJL/dualpools
Commit	e0c29231be299882db58d4c354c25f20c7cca75f

Audit Updates

Initial Audit	16 Apr 2023 https://github.com/cyberscope-io/audits/blob/main/xdp/v1/trendToken.pdf
Corrected Phase 2	24 Apr 2023 https://github.com/cyberscope-io/audits/blob/main/xdp/v2/trendToken.pdf
Corrected Phase 3	28 Apr 2023

Source Files

Filename	SHA256
Address.sol	df780035071456dfc4b41b4647f1ba1e69da5e97eff3b0c3c1f2ed5eb4c24994
AggregatorV2V3Interface.sol	a5523f8072d46e9a121390beefd22cf9b78d00efec1e1f1274c010cacb019495
CompStorageTT.sol	3e0959f600bd1eb11cb54a9a98df75244cbce6ba91f38fcd2c8971c6680a5242
CompTT.sol	4a37ff340d53f8d96439514975b532cf5fdbdd824bf9066c10a33389ae096d6e
DualPool.sol	ddf24f1548fab966f15e4fb3c02ed950fa3f4a1221c86787b5716a377d2dde62
DualPoolStorage.sol	d0b5fdd0579d88997b23be8a436bddbf0dbdc69429a7a96c1295d83a5e73a306
ERC20.sol	6b52089f84c5f7e6fbf7f2219ebe31f43e104ce89b44b28a72a4371c19a847e6
ERC20Detailed.sol	edbc6746642b1fdb6066c5dacd1c3756f04c4af57521594fbc8864c61e5b648
IChainlinkOracle.sol	27cac8821c279a09cdf6b3e0e1985867db49ef35cce6763df8154daafb77a31e
ICompDP.sol	1fbb60793e8bf070e6e70f05d3e9c67557ec73aa637630146f4cdc4206073b48
ICompTT.sol	b71da5599b0e6bbe9fc1d163880a8c3d0290f9d315c97ac212f99ecde715e818
IERC20.sol	23221a896472eeee23d71500d71f40bce31112b9198389310d2e7ff7d0be093

IIncentiveModelSimple.sol	a3a4ee3d50e418aa6163a3a63c55e451941a0f931848fb259158b41cc489f520
IncentiveModelSimple.sol	6c3951c5bcbc23db96cc61954314ccaa7713f43ab0fe41ded7776fdc6532dbec
ITrendToken.sol	7eac00453f65769353394e73abfc46ef366ec79bd640069669afd08438abbe00
ITrendTokenTkn.sol	b3286f0e63b4fb4f77cc7d012fff28c06b6390686542f9c107a65e3b9549137e
IVBep20.sol	4388fcfcdf67909073a1af7353687e653b9180653281418aced5a607afcc78b8
IVBNB.sol	53487fc7336311df84bdb8faa5d7999ae61b48b9f72f462ceae53dc34f631435
IXTT.sol	36ff8e43a69fc4b0da21352c652aee0b733c841be27fcf3035d9583ea6519ac7
Lens.sol	dfd684164358d48b86653c429aa994f52f15b244656a1bed2c3da8b30082e9f6
Lib.sol	38b01b1d3513df71ab674cd6061e93c1d434f5f57efecab71c2131d07a8965e9
SafeERC20.sol	02ac28e67c7239a7a8137b864ef01063b4b649f8c4609edfb62f1829a8ee9185
SafeMath.sol	4a47d15402f20ec26b0fe15d61f4f6e946e7949b7beaa6398957b5cadee42931
SignedSafeMath.sol	533257d850b02a32792adfa2e02af99e926a1b08af501b37eae82cc174b9c06a
TrendToken.sol	78b1611d00bf1a12939938ba55a5f96616e1aba6f18b9e2ad8fff26b6f938bd6
TrendTokenStorage.sol	a04cba0979d88757882babddeceeb360f037579d98f590177740056b695b55f5

TrendTokenTkn.sol	0022a6b8e6d71e032073f4f6aad22a294 d19098e72d51c097bf82e179a7f44a8
UniTT.sol	57442216e7dc8993ab5fe99cb5699b23 865cd42fd0f8b983e87746de31e32003
XTTgov.sol	11160e5e1becd7e1f6f43af23b635a245 92ade0b1627b4d79252ff64ed6e292b

Findings Breakdown



Critical	1
Medium	0
Minor / Informative	6

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	1	0	0
Medium	0	0	0	0
Minor / Informative	2	4	0	0

Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	UIA	Unsafe Indexing Assumption	Acknowledged
●	MT	Mints Tokens	Acknowledged
●	IDI	Immutable Declaration Improvement	Unresolved
●	L02	State Variables could be Declared Constant	Acknowledged
●	L04	Conformance to Solidity Naming Conventions	Acknowledged
●	L07	Missing Events Arithmetic	Unresolved
●	L19	Stable Compiler Version	Acknowledged

UIA - Unsafe Indexing Assumption

Criticality	Critical
Status	Acknowledged

Description

The contract assumes that the `desiredAllocations` indexes are identical to the market's tokens indexes. This assumption could be broken if the `desiredAllocations` or `market's tokens` change without updating the corresponding structures. As a result, the entire contract will yield to an unexpected state.

```
desiredAllocations = _allocations;
emit SetDesiredAllocationsFresh(getMarkets(), oldAllocations,
desiredAllocations);

//...

if (IVBep20(dTokens[i]) == dTokensInOut[0]) {
    tokenEquityInOut[0] = conVals[i].add(colVals[i]);
    desiredAllocations[0] = desiredAllocations[i];
}
```

Recommendation

The team is advised to carefully investigate the circumstances where the indexes could be diverse. A recommended way could be to allow only synchronized modifications of these properties.

Team Update

Trading bot will ensure proper index when changing order of markets (tokens) or adjusting desired allocations.

We acknowledge the risk and make sure to update appropriately, but think this is more of a "medium" or even "minor" risk severity.

MT - Mints Tokens

Criticality	Minor / Informative
Location	DualPool.sol#L298
Status	Acknowledged

Description

The manager role has the authority to arbitrary mint tokens. The role may take advantage of it by calling the `_supplyCollateral` function. As a result, the contract tokens will be highly inflated.

```
function _supplyCollateral(IERC20 _depositBep20, uint supplyAmt)
onlyManager external {
    IVBep20 dToken = dTokenSupportedRequire(_depositBep20);
    collateralSupply(_depositBep20,dToken, supplyAmt);
}
```

Recommendation

The team should carefully manage the private keys and the implementation of the trading bot account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. Some suggestions are:

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-sign wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.
- Renouncing the ownership will eliminate the threats but it is non-reversible.

Team Update

This only affects supplying and redeeming supplied assets to the lending/borrowing protocol Dual Pools (fork of Venus).

IDI - Immutable Declaration Improvement

Criticality	Minor / Informative
Location	TrendTokenTkn.sol#L21
Status	Unresolved

Description

The contract is using variables that initialize them only in the constructor. The other functions are not mutating the variables. These variables are not defined as `immutable`.

```
minter
```

Recommendation

By declaring a variable as immutable, the Solidity compiler is able to make certain optimizations. This can reduce the amount of storage and computation required by the contract, and make it more gas-efficient.

L02 - State Variables could be Declared Constant

Criticality	Minor / Informative
Location	TrendTokenStorage.sol#L101,108,122,129,135,141,147,154,160,166,173,180,187,201,209,217,227DualPoolStorage.sol#L16,22,28,34,40CompStorageTT.sol#L13,18,23,28,38,87,93
Status	Acknowledged

Description

State variables can be declared as constant using the constant keyword. This means that the value of the state variable cannot be changed after it has been set. Additionally, the constant variables decrease gas consumption of the corresponding transaction.

```
bool internal _notEntered
uint public contractFactor = 0.90e18
uint public maxSupply = 1000e18
address public manager
address public tradingBot
address payable public feeRecipient
IIncentiveModelSimple public incentiveModel
bool public trendTokenPaused = false
uint public referralReward = 0.40e18
uint public performanceFee = 0.10e18
uint public trendTokenRedeemBurn = 0.50e18
uint public accruedXDPTtoFeeRecipient = 0.50e18
uint public maxDisableTokenValue = 1e18
ITrendTokenTkn public trendToken

...
```

Recommendation

Constant state variables can be useful when the contract wants to ensure that the value of a state variable cannot be changed by any function in the contract. This can be useful for storing values that are important to the contract's behavior, such as the contract's address or the maximum number of times a certain function can be called. The team is advised to add the constant keyword to state variables that never change.

Team Update

State variables cannot be declared constant. All variables in storage (i.e TrendTokenStorage.sol) are changeable in other files (i.e TrendToken.sol), therefore, they cannot be declared constant

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	TrendTokenTkn.sol#L31,35TrendTokenStorage.sol#L101TrendToken.sol#L93,149,180,211,227,240,253,265,275,286,296,305,319,331,345,375,385,401,412,431,440,454,473,485,508,517,534,574,603,666,696,742,779,804,825,864,872,893,912,934,971,1031,1090,1140Lib.sol#L27,32DualPool.sol#L51,70,82,95,105,116,128,144CompTT.sol#L201,210,219,231,255,293,332,352,361,371,381,392,403
Status	Acknowledged

Description

The Solidity style guide is a set of guidelines for writing clean and consistent Solidity code. Adhering to a style guide can help improve the readability and maintainability of the Solidity code, making it easier for others to understand and work with.

The followings are a few key points from the Solidity style guide:

1. Use camelCase for function and variable names, with the first letter in lowercase (e.g., myVariable, updateCounter).
2. Use PascalCase for contract, struct, and enum names, with the first letter in uppercase (e.g., MyContract, UserStruct, ErrorEnum).
3. Use uppercase for constant variables and enums (e.g., MAX_VALUE, ERROR_CODE).
4. Use indentation to improve readability and structure.
5. Use spaces between operators and after commas.
6. Use comments to explain the purpose and behavior of the code.
7. Keep lines short (around 120 characters) to improve readability.

```
address _recipient
uint256 _amount
bool internal _notEntered
address _owner
address _compTT
```

...

Recommendation

By following the Solidity naming convention guidelines, the codebase increased the readability, maintainability, and makes it easier to work with.

Find more information on the Solidity documentation

<https://docs.soliditylang.org/en/v0.8.17/style-guide.html#naming-convention>.

L07 - Missing Events Arithmetic

Criticality	Minor / Informative
Location	TrendToken.sol#L242
Status	Unresolved

Description

Events are a way to record and log information about changes or actions that occur within a contract. They are often used to notify external parties or clients about events that have occurred within the contract, such as the transfer of tokens or the completion of a task.

It's important to carefully design and implement the events in a contract, and to ensure that all required events are included. It's also a good idea to test the contract to ensure that all events are being properly triggered and logged.

```
referralReward = _referralReward
```

Recommendation

By including all required events in the contract and thoroughly testing the contract's functionality, the contract ensures that it performs as intended and does not have any missing events that could cause issues with its arithmetic.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	TrendTokenTkn.sol#L2TrendTokenStorage.sol#L2TrendToken.sol#L2Lib.sol#L2ITrendTokenTkn.sol#L2ITrendToken.sol#L2ICompTT.sol#L2ICompDP.sol#L2DualPoolStorage.sol#L2DualPool.sol#L2CompTT.sol#L2CompStorageTT.sol#L2
Status	Acknowledged

Description

The `^` symbol indicates that any version of Solidity that is compatible with the specified version (i.e., any version that is a higher minor or patch version) can be used to compile the contract. The version lock is a mechanism that allows the author to specify a minimum version of the Solidity compiler that must be used to compile the contract code. This is useful because it ensures that the contract will be compiled using a version of the compiler that is known to be compatible with the code.

```
pragma solidity ^0.5.16;
```

Recommendation

The team is advised to lock the pragma to ensure the stability of the codebase. The locked pragma version ensures that the contract will not be deployed with an unexpected version. An unexpected version may produce vulnerabilities and undiscovered bugs. The compiler should be configured to the lowest version that provides all the required functionality for the codebase. As a result, the project will be compiled in a well-tested LTS (Long Term Support) environment.

Team Update

We will make sure to deploy using 0.5.16 compiler

Functions Analysis

Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
Address	Library			
	isContract	Internal		
	toPayable	Internal		
	sendValue	Internal	✓	
AggregatorV2V3Interface	Interface			
	latestAnswer	External		-
	latestTimestamp	External		-
	latestRound	External		-
	getAnswer	External		-
	getTimestamp	External		-
	decimals	External		-
	description	External		-
	version	External		-
	getRoundData	External		-
	latestRoundData	External		-
UnitrollerAdminStorage	Implementation			

ComptrollerStorage	Implementation	UnitrollerAdminStorage		
CompTT	Implementation	ComptrollerStorage		
		Public	✓	-
	ensureAdmin	Private		
	ensureNonzeroAddress	Private		
	depositOrRedeemAllowed	External		onlyProtocolAllowed
	tradeAllowed	External		onlyProtocolAllowed
	getBlockNumber	External		-
	getXVSAddress	External		-
	returnDToken	External		onlyProtocolAllowed
	trendTokenIsListed	External		-
	trendTokenIsActive	External		-
	trendTokenIsTrade	External		-
	trendTokenAllowedDualPools	External		-
	trendTokenMaxTradeFee	External		-
	trendTokenMaxPerformanceFee	External		-
	trendTokenMaxDisableValue	External		-
	_become	External	✓	-
	_setProtocolPaused	External	✓	validPauseState
	_setMintPaused	External	✓	validPauseState

	_setPriceOracle	External	✓	-
	_setPauseGuardian	External	✓	-
	supportTokenFresh	Internal	✓	
	_supportToken	External	✓	-
	trendTokenSupported	Internal		
	_supportTrendTokenFresh	Internal	✓	
	_supportTrendToken	External	✓	-
	_newIsActive	External	✓	onlySupportedTrendTokens
	_newIsTrade	External	✓	onlySupportedTrendTokens
	_newAllowedDualPools	External	✓	onlySupportedTrendTokens
	_newMaxTradeFee	External	✓	onlySupportedTrendTokens
	_newMaxPerformanceFee	External	✓	onlySupportedTrendTokens
	_newMaxDisableValue	External	✓	onlySupportedTrendTokens
DualPoolIntegration	Implementation	DualPoolStorage		
		Public	✓	-
		External	Payable	-
	screenshot	Internal		
	getMarkets	Internal		
	priceBEP20	Internal		
	exchangeVBEP20	Internal		
	enableCol	Internal	✓	

	disableCol	Internal	✓	
	tokenEntered	Internal		
	collateralSupply	Internal	✓	
	collateralRedeem	Internal	✓	
DualPoolStorage	Implementation			
ERC20	Implementation	IERC20		
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	_transfer	Internal	✓	
	_mint	Internal	✓	
	_burn	Internal	✓	
	_approve	Internal	✓	
	_burnFrom	Internal	✓	
ERC20Detailed	Implementation	IERC20		

		Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
IOracle	Interface			
	getUnderlyingPrice	External		-
	getFeed	External		-
	getChainlinkPrice	External		-
ICompDP	Interface			
	enterMarkets	External	✓	-
	claimXDP	External	✓	-
	venusAccrued	External		-
	getAssetsIn	External		-
	markets	External		-
	getAccountLiquidity	External		-
	closeFactorMantissa	External		-
	exitMarket	External	✓	-
	getHypotheticalAccountLiquidity	External		-
	checkMembership	External		-
	iUSDaddress	External		-

ICompTT	Implementation			
	oracle	External		-
	protocolPaused	External		-
	depositOrRedeemAllowed	External		-
	tradeAllowed	External		-
	returnDToken	External		-
	trendTokenIsListed	External		-
	trendTokenIsActive	External		-
	trendTokenIsTrade	External		-
	trendTokenAllowedDualPools	External		-
	trendTokenMaxTradeFee	External		-
	trendTokenMaxPerformanceFee	External		-
	trendTokenMaxDisableValue	External		-
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
IIncentiveMode ISimple	Interface			

	protocolFeeTrade	External		-
	protocolFeeDeposit	External		-
	protocolFeeRedeem	External		-
	isIncentiveModel	External		-
	totalDepositFee	External		-
	totalRedeemFee	External		-
	valueOutAfterSell	External		-
	valueOutAfterBuy	External		-
IncentiveModel Simple	Implementation	IIncentiveModelSimple		
		Public	✓	-
	_updateTradeFeeDiscounts	External	✓	-
	feeDiscount	Public		-
	feePerToken	Public		-
	depositRewardOrFee	Public		-
	redeemRewardOrFee	Public		-
	totalDepositFee	Public		-
	totalRedeemFee	Public		-
	valueOutAfterSell	External		-
	valueOutAfterBuy	External		-
ITrendToken	Interface			
	incentiveModel	External		-

	storedEquity	External		-
	trendToken	External		-
	performanceFee	External		-
	desiredAllocations	External		-
	lastRebalance	External		-
	isTrendToken	External		-
	compDP	External		-
	dBNB	External		-
	priceExt	External		-
	trendTokenToUSDext	External		-
	trendTokenOutExternal	External		-
	trendTokenInExternal	External		-
	tradeInfoExt	External		-
ITrendTokenTk n	Interface			
	mint	External	✓	-
	name	External		-
	burn	External	✓	-
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-

	transfersFrom	External	✓	-
IVBep20	Interface			
	balanceOf	External		-
	balanceOfUnderlying	External	✓	-
	mint	External	✓	-
	repayBorrow	External	✓	-
	redeemUnderlying	External	✓	-
	exchangeRateStored	External		-
	borrowBalanceCurrent	External	✓	-
	borrow	External	✓	-
	getCash	External		-
	getAccountSnapshot	External		-
	borrowBalanceStored	External		-
	totalBorrowsCurrent	External	✓	-
	accrueInterest	External	✓	-
	amountsOut	External		-
	swapExactTokensForTokens	External	✓	-
	underlying	External		-
IVBNB	Interface			
	balanceOf	External		-
	balanceOfUnderlying	External	✓	-

	mint	External	Payable	-
	repayBorrow	External	Payable	-
	redeemUnderlying	External	✓	-
	exchangeRateStored	External		-
	borrowBalanceCurrent	External	✓	-
	borrow	External	✓	-
	getAccountSnapshot	External		-
	borrowBalanceStored	External		-
	totalBorrowsCurrent	External	✓	-
	swapExactETHForTokens	External	Payable	-
IXTT	Interface			
	burn	External	✓	-
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transfersFrom	External	✓	-
TrendLens	Implementation			
		Public	✓	-
	trendTokenData	Public		-

	trendTokenDataAll	Public		-
	swapPairsOld	External		-
	trendTokenPortfolio	Public		-
	removeUnderlyingFromPortfolio	Public		-
	portfolioAddTrendToken	Public		-
	swapPairs	External		-
	trendTokenOutAndFee	Public		-
	underlyingInAndFee	Public		-
	depositAmountsAndFee	Public		-
	trendTokenInAndFee	Public		-
	underlyingOutAndFee	Public		-
	redeemAmountsAndFees	Public		-
	balance	Public		-
	walletBalance	Public		-
	amountsAndFees	External		-
	amountsAndFeesTradeHelper	Internal		
	amountsAndFeesTrade	External		-
Lib	Library			
	pathGenerator2	Internal		
	getValue	Internal		
	getAssetAmt	Internal		

SafeERC20	Library			
	safeTransfer	Internal	✓	
	safeTransferFrom	Internal	✓	
	safeApprove	Internal	✓	
	safeIncreaseAllowance	Internal	✓	
	safeDecreaseAllowance	Internal	✓	
	callOptionalReturn	Private	✓	
SafeMath	Library			
	add	Internal		
	add	Internal		
	sub	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	div	Internal		
	mod	Internal		
	mod	Internal		
SignedSafeMath	Library			
	mul	Internal		
	div	Internal		
	sub	Internal		

	add	Internal		
TrendToken	Implementation	DualPoolIntegration, TrendTokenStorage		
		Public	✓	DualPoolIntegration
		External	Payable	-
	onlyModifiers	Internal		
	_updateCompAndModels	External	✓	onlyManager
	_updateManagerRecipientAndBot	External	✓	onlyManager
	_newPerformanceFee	External	✓	onlyManager
	_updateFeeDistribution	External	✓	onlyManager
	_setReferralReward	External	✓	onlyManager
	_maxDisableValue	External	✓	onlyManager
	_setMaxSupply	External	✓	onlyManager
	setContractFactor	Internal	✓	
	_setContractFactor	External	✓	onlyManager
	_supplyCollateral	External	✓	onlyManager
	_redeemCollateral	External	✓	onlyManager
	_depositsDisabled	External	✓	onlyTradingBot
	_pauseTrendToken	External	✓	onlyTradingBot
	dTokenSupportedRequire	Internal		
	_setDesiredAllocationsFresh	Internal	✓	
	_setDesiredAllocations	External	✓	onlyTradingBot

	_enableTokens	External	✓	onlyTradingBot
	checkActiveToken	Internal		
	_disableToken	External	✓	onlyTradingBot
	_redeemPerformanceFee	External	✓	onlyManager
	_reduceTrendTokenReservesToRecipient	External	✓	onlyManager
	_redeemXDP	External	✓	onlyManager
	_reduceXDPtoRecipient	External	✓	onlyManager
	priceExt	External		-
	balanceXDPExt	External		-
	trendTokenToUSDExt	External		-
	trendTokenOutExternal	External		-
	trendTokenInExternal	External		-
	storedEquityExternal	External		-
	tokenInfoExternal	External		-
	tradeInfoExt	External		-
	calculatePerformanceFee	Internal		
	sendPerformanceFee	Internal	✓	
	balanceXDP	Internal		
	contractBal	Internal		
	storedEquity	Internal		
	trendTokenToUSD	Internal		
	trendTokenToUSD	Internal		
	tokenInfo	Internal		

	tokenEquityVal	Internal		
	distributeReferralReward	Internal	✓	
	trendTokenOutCalculations	Internal		
	depositFresh	Internal	✓	pausedTrendToken
	depositBNB	External	Payable	nonReentrant
	deposit	External	✓	nonReentrant
	sendUnderlyingOut	Internal	✓	
	trendTokenInCalculations	Internal		
	redeemFresh	Internal	✓	pausedTrendToken
	redeem	External	✓	nonReentrant
	tradeInfo	Internal		
	executeTrade	Internal	✓	pausedTrendToken
	swapExactTokensForTokens	External	✓	nonReentrant
	swapExactETHForTokens	External	Payable	nonReentrant
	returnUnderlying	Internal		
	singleSupplyAndRedeemRebalance	Internal	✓	
	publicSupplyAndRedeemRebalance	External	✓	pausedTrendToken
TrendTokenStorage	Implementation			
TrendTokenToken	Implementation	ERC20, ERC20Detailed		
		Public	✓	ERC20Detailed

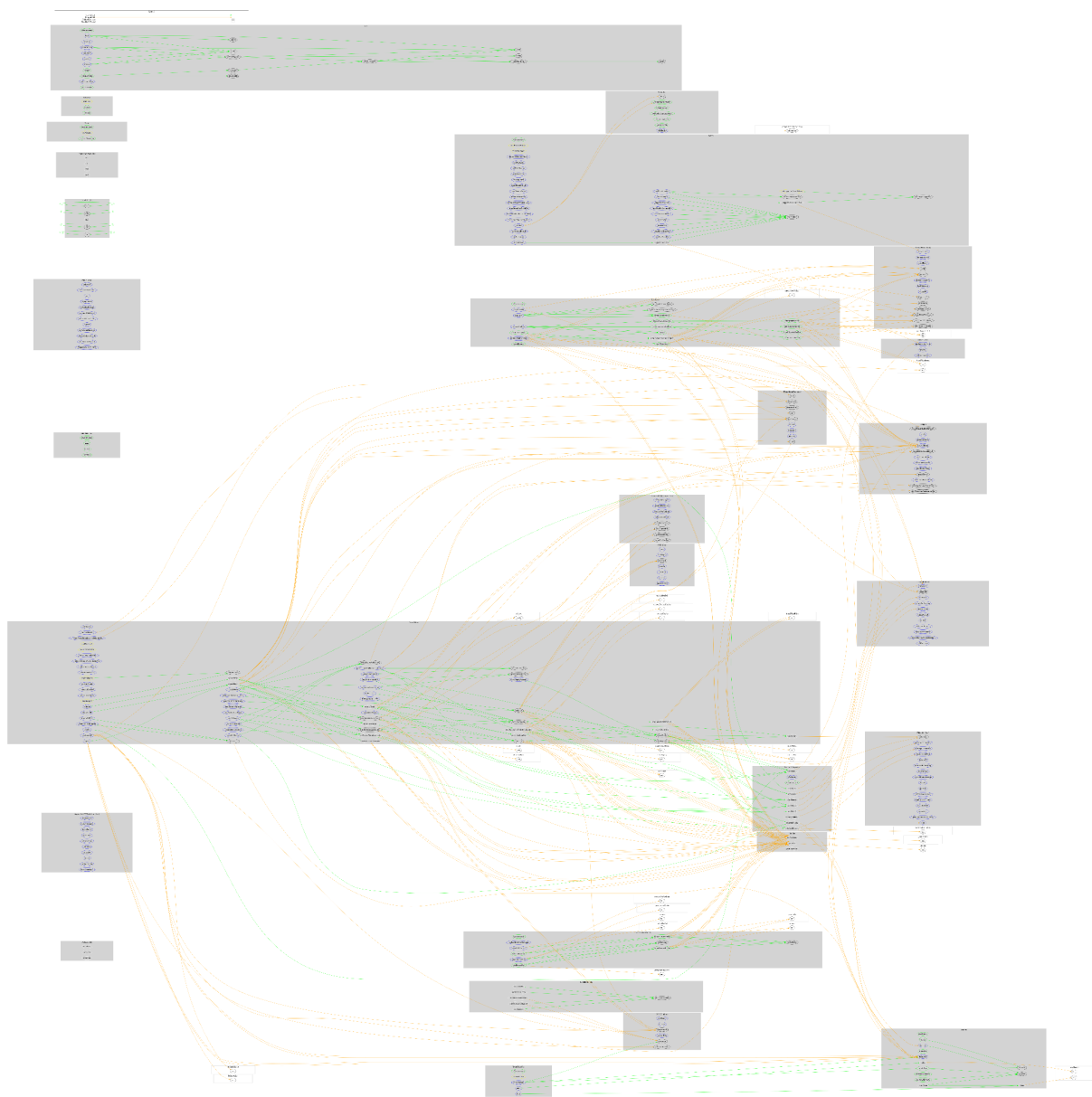
	mint	External	✓	requireMinter
	burn	External	✓	-
	transfersFrom	External	✓	-
Unitroller	Implementation	UnitrollerAdminStorage		
		Public	✓	-
	_setPendingImplementation	Public	✓	-
	_acceptImplementation	Public	✓	-
	_setPendingAdmin	Public	✓	-
	_acceptAdmin	Public	✓	-
		External	Payable	-
Owned	Implementation			
		Public	✓	-
	transferOwnership	Public	✓	onlyOwner
Tokenlock	Implementation	Owned		
	freeze	Public	✓	onlyOwner
	unfreeze	Public	✓	onlyOwner
XTT	Implementation	Tokenlock		
		Public	✓	-
	burn	External	✓	-

	allowance	External		-
	approve	External	✓	validLock
	balanceOf	External		-
	transfer	External	✓	validLock
	transferFrom	External	✓	validLock
	delegate	Public	✓	validLock
	delegateBySig	Public	✓	validLock
	getCurrentVotes	External		-
	getPriorVotes	Public		-
	_delegate	Internal	✓	
	_transferTokens	Internal	✓	
	_moveDelegates	Internal	✓	
	_writeCheckpoint	Internal	✓	
	safe32	Internal		
	safe96	Internal		
	add96	Internal		
	sub96	Internal		
	sub256	Internal		
	getChainId	Internal		

Inheritance Graph



Flow Graph



Summary

Dual Pools contract implements a token mechanism. This audit investigates security issues, business logic concerns, and potential improvements.

Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Cyberscope's prior written consent. This report is not nor should be considered an "endorsement" or "disapproval" of any particular project or team. This report is not nor should be regarded as an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Cyberscope to perform a security assessment. This document does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors' business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment 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. Cyberscope's position is that each company and individual are responsible for their own due diligence and continuous security. Cyberscope's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. You agree that your access and/or use including but not limited to any services reports and materials will be at your sole risk on an as-is where-is and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.



The Cyberscope team

<https://www.cyberscope.io>