Trend Token Audit Feedback

No change needed (5) - we do not think there is an error
Resolved (16) - recommendation was applied to potentially resolve the error
Partially Resolved (2) - issue was solved in some suggested cases, acknowledged in the other cases

Notes
Contract byte size limits were preventing the resolution of MEM, L07, L20. To fix this, _swapXDPforBNB() was replaced with a simpler function _reduceXDPtoRecipient() and the swap of XDP to BNB can occur externally by the feeRecipient. This can only occur to XDP earned from dual pools as a staking reward, so not risk to client funds. This change removed the reliance of pancakeSwap and some variables in TrendTokenStorage.sol which freed up a lot of space.

Severity	**Code**	**Description**	**Status**	**Description**
Critical	PSU	Potential Subtraction Overflow	No Change Needed	Int allows for negative values. SignedSafeMath prevents underlow if value too negative
Critical	UIA	Unsafe Indexing Assumption	No Change Needed	Trading bot will ensure proper index when changing order of markets (tokens) or adjusting desired allocations. Not critical
Medium	PTAI	Potential Transfer Amount Inconsistency	Resolved (tested)	Checked values before and after transfer to ensure it accounted for any tax upon transfer
Medium	DSD	Data Storage Duplication	Resolved (tested)	Required underlying doesn't already exist when trying to support. Also require new dToken and underlying are not zero addresses
Minor	RCE	Redundant Code Execution	Resolved (tested)	Removed netEquity from entire singleSupplyAndRedeemRebalance(), tokenEquity >0 when netEquity>0 so just relied on tokenEquity
Minor	MT	Mint Token	Resolved (tested)	More secure onlyManager keys are used. Low security risk as this only allows for supplying or redeeming collateral.
Minor	MEM	Misleading Error Message	Resolved (tested)	Separate the two conditions and have separate error messages.
Minor	REE	Redundant Equity Evaluation	Resolved (tested)	Reworked tokenEquityVal() to not rely on storedEquity() since it only needs to evaluate one token.
Minor	TPOE	Transitive Property of Equality	Resolved (tested)	Removed check for each token exceeding 1e18 in _setDesiredAllocationsFresh()
Minor	MUT	Mutating Unsupported Tokens	Resolved (tested)	Added onlySupportedTrendTokens(trendToken) modifier to functions that set values for Trend Tokens to ensure its already supported
Minor	RDC	Redundant Duplicate Check	Resolved (tested)	Removed trendToken.isTrendToken() in _supportTrendToken() CompTT ~line 350
Minor	RG	Redundant Getters	Resolved (no test needed)	Removed getAllTrendTokens() CompTT.sol line ~130
Minor	UM	Unused Modifier	Partially Resolved (no test needed)	Removed onlyListedTrendTokens(). The other external functions are used for a frontend lens which we will tolerate

Minor	IDI	Immutable Declaration Improvement	No Change Needed	Immutable was not introduced until Solidity ^0.6.5
Minor	L02	State Variable could be Declared Constant	No Change Needed	The variables are changeable, but not in the files they are in. For example, TrendToken.sol changes variables in TrendTokenStorage.sol
Minor	L04	Conformance to Solidity Naming Conventions	Partially Resolved (no test needed)	Only a couple changed. We will acknowledge and tolerate the rest.
Minor	L07	Missing Event Arithmetic	Resolved (tested)	Suggested events added, and more (such as ExecuteTrade)
Minor	L09	Dead Code Elimination	Resolved (no test needed)	Some dead code was eliminated, but many were required in other files
Minor	L14	Uninitialized Variables in Local Scope	Resolved (tested)	Set mintTrendToken to 0 in TrendToken.sol line ~680. Acknowledge and tolerate other examples
Minor	L15	Local Scope Variables Shadowing	Resolved (tested)	Renamed local variable to desiredAllos[]
Minor	L16	Validate Variable Setters	No Change Needed	pauseGuardian is not a variable that can be configured on user-supplied input. Please explain more, not sure if I understand the issue
Minor	L19	Stable Compiler Version	Resolved	Changed ^0.5.0 to ^0.5.16;
Minor	L20	Succeeded Transfer Check	Resolved	Using the SafeERC20

Critical Criticality

1) PSU - Potential Subtraction Underflow (done)

Solution

No change needed.

Code

```
uint tokenEquity = conVals[i].add(colVals[i]);
uint priorAllocation = Lib.getAssetAmt(tokenEquity,netEquity);
priorDelta = int(desiredAllocations[i]).sub(int(priorAllocation));
```

Explanation

I do not believe there is the chance of subtraction underflow here. PriorDelta is “int” so it can hold negative values. Furthermore, `sub()` from `SignedSafeMath` properly handles the potential of subtraction underflow if the value becomes too negative.

Other parts in the contract account for the potential negative value
Negative → protocol desires ***less*** of that asset
Positive → protocol desires ***more*** of that asset

PLEASE CORRECT IF I AM WRONG

2) UIA - Unsafe Indexing Assumptions (done)

Solution

No change needed. PLEASE CORRECT IF I AM WRONG

Code

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

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

Explanation

Although there is the potential for unexpected states if the assumption of `desiredAllocations` corresponding to `market's tokens` indexes is not met, this is the responsibility of `onlyManager` and `onlyTradingBot` to properly update `desiredAllocations` according to the order of `market's tokens`.

When updating `desiredAllocations` or `market's tokens`, the require check `require(_allocations.length == getMarkets().length, "!length");` is checked to ensure the lengths are of the same length. Although this does not guarantee the intended order is met. Nonetheless, I do not think this would be a "critical" issue as the desired position sizes for each token would be different than expected, and could be easily corrected without any critical issues (i.e funds lost). Whenever the tokens are updated (added or removed) it demands desiredAllocations are also updated.

PLEASE CORRECT IF I AM WRONG

Medium Criticality

1) PTAI - Potential Transfer Amount Inconsistency (done)

Solution

Calculated the actual amount by fetching the difference between transfer calls.
TrendToken.sol line ~860

Alternatively, our team could have prevented the addition of tokens that have a tax upon transfer, but the solution below was chosen to provide more flexibility and reduce potential errors in the future.

Code

```
852
853     /**
854     *   Payable function for buying Trend Tokens with BNB
855     */
856     function deposit(IERC20 _depositBep20, uint _sellAmtBEP20, address payable _referrer) external nonReentrant {
857         uint balanceBefore = _depositBep20.balanceOf(address(this));
858         _depositBep20.transferFrom(msg.sender, address(this), _sellAmtBEP20);
859         uint balanceAfter = _depositBep20.balanceOf(address(this));
860         uint actualTransferredAmount = balanceAfter.sub(balanceBefore);
861         depositFresh(_depositBep20, actualTransferredAmount,_referrer);
862     }
863
864
```

2) Data Storage Duplication (done)

Solution

Required underlying didn't exist and require the dToken is not the zero address to prevent duplicates and unexpected data storage.
CompTT.sol line ~300

Minor/Informative Criticality

1) RCE - Redundant Code Execution (done)

Solution

Remove netEquity from entire singleSupplyAndRedeemRebalance(), if tokenEquity>0 then netEquity>0 so no reason for both conditions to be met
TrendToken.sol, line ~1100

Also removed borVals from storedEquity() because it was not used in the contracts

2) MT - Mint Tokens (done)

Solution

No changes needed. CORRECT IF I AM WRONG

Explanation

Although we do agree that tradingBot private keys should be safe, this mint action will not result in contract tokens to be highly inflated. This action simply supplied collateral to Dual Pools (fork of Venus we previously got audited by you) and received dTokens (vTokens in the case of Venus) in return, which can be used to redeem the collateral at a future date. We cannot see how this could result in an exploit that an attacker could take advantage of.

Although, if there was borrowing positions allowed, then an attacker (if they obtained trading bot keys) could redeem collateral from Dual Pools until liquidation. But since there is no borrowing capabilities, this is not possible. The only downside is less interest earnings if an attacker withdrew collateral from Dual Pools

3) MEM - Misleading Error Messages (done)

Solution

Separate the two conditions and have separate error messages.

4) REE - Redundant Equity Evaluation (done)

Solution

Redid tokenEquityVal() to not rely on storedEquity() since it only needs to evaluate one token.

5) TPOE - Transitive Property of Equality (done)

Solution

Removed check for each token exceeding 1e18 in _setDesiredAllocationsFresh()

6) MUT - Mutating Unsupported Tokens (done)

Solution

Added onlySupportedTrendTokens(trendToken) modifier to functions that set values for Trend Tokens to ensure its already supported

7) RDC - Redundant Duplicate Checks (done)

Solution

Removed trendToken.isTrendToken() in _supportTrendToken()
CompTT ~line 350

8) RG - Redundant Getters (done)

Solution

Removed getAllTrendTokens() CompTT.sol line ~130

9) UM - Unused Modifiers (done)

Solution

Removed onlyListedTrendTokens(). The other external functions are used for a frontend lens, which may decrease readability but this will be tolerated.

10) IDI - Immutable Declaration Improvement (done)

Solution

No change allowed. Immutable was not introduced until Solidity ^0.6.5

11) L02 - State Variables could be Declared Constant (done)

Solution

No change needed.

Explanation

All these values can be changed, just not in the files they are found in. For example, TrendTokenStorage.sol values can be changed from TrendToken.sol since it inherits from it.

12) L04 - Conformance to Solidity Naming Conventions (done)

Solution

Only a couple changed. We will acknowledge and tolerate the rest.

Explanation

Although the names could be more detailed, we will acknowledge and tolerate

13) L07 - Missing Event Arithmetic (done)

Solution

No change allowed. Restricted by contract byte size. If any room becomes available, we will include such Event Arithmetic.

14) L09 - Dead Code elimination (done)

Removed
Lib.sol ⇒ pathGenerator3(), min(), and countValueArray(), although others were required for
DualPool.sol ⇒ _newCompDPinternal(), amtAccruedXDP(), exchangeVBEP20()
CompTT.sol ⇒ adminOrInitializing()

The other functions were required by files that inherited them, for example, TrendTokens.sol used many of the functions in DualPools

15) L14 - Uninitialized Variables in Local Scope (done)

Solution

Set mintTrendToken to 0 in TrendToken.sol line ~680. Acknowledge and tolerate other examples

16) L15 - Local Scope Variable Shadowing (done)

Solution

Great catch! Renamed local variable to desiredAllos[]

17) L16 - Validate Variable Setters (done)

Solution

No change needed (correct if I am wrong)

Explanation - I am not exactly sure about the issue here. Only the admin can change pauseGuardian with _setPauseGuardian(). There is a check ensureNonzeroAddress() to ensure newPauseGuardian() input is nonZero

18) L19 - Stable Compiler Version (done)

Solution

Changed ^0.5.0 to ^0.5.16;

19) L20 - Succeeded Transfer Check (done)

Solution

Use safeERC20