



Cyberscope

Audit Report

# Tangible USDR Ecosystem

January 2023

Github <https://github.com/TangibleTNFT/usdr-contracts>

Commit [ac5c4bf0e17df891c3a43daf2215238c84459dd1](#)

Audited by © cyberscope

# Table of Contents

<b>Table of Contents</b>	<b>1</b>
<b>Contracts Review</b>	<b>5</b>
<b>Audit Updates</b>	<b>5</b>
<b>Testing Deploy</b>	<b>6</b>
<b>Source Files</b>	<b>8</b>
<b>DAO Infrastructure Architecture Review</b>	<b>12</b>
<b>Team's Reply 20 December 2022</b>	<b>12</b>
<b>Decimal Architecture</b>	<b>13</b>
<b>Recommendation</b>	<b>14</b>
<b>Team's Reply 14 December 2022</b>	<b>14</b>
<b>Roles Architecture</b>	<b>15</b>
<b>Team's Reply 14 December 2022</b>	<b>15</b>
<b>Contract Diagnostics</b>	<b>16</b>
<b>AFI - Affiliate Token Issue</b>	<b>18</b>
<b>Description</b>	<b>18</b>
<b>Recommendation</b>	<b>19</b>
<b>Team's Reply 14 December 2022</b>	<b>19</b>
<b>STI - Staking Token Issue</b>	<b>20</b>
<b>Description</b>	<b>20</b>
<b>Recommendation</b>	<b>20</b>
<b>Team's Reply 14 December 2022</b>	<b>21</b>
<b>DMI - Defractionalize Manipulation Issue</b>	<b>22</b>
<b>Description</b>	<b>22</b>
<b>Recommendation</b>	<b>22</b>
<b>Team's Reply 14 December 2022</b>	<b>22</b>
<b>TBI - Token Balance Inconsistency</b>	<b>23</b>

Description	23
Recommendation	23
Team's Reply 14 December 2022	23
PRD - Pair Reserves Diversion	24
Description	24
Recommendation	25
Team's Reply 14 December 2022	25
TAZFA - Transferred Amount Zero Fees Assumption	26
Description	26
Recommendation	26
Team's Reply 14 December 2022	27
AIC - Arguments Inconsistency	28
Description	28
Recommendation	29
Team's Reply 14 December 2022	29
ELFM - Exceeds Fees Limit	30
Description	30
Recommendation	30
Team's Reply 14 December 2022	30
DSM - Decimal Scale Missconcern	31
Description	31
Recommendation	32
Team's Reply 14 December 2022	32
PIL - Potential Infinite Loop	33
Description	33
Recommendation	33
Team's Reply 14 December 2022	33
STC - Succeed Transfer Check	35

Description	35
Recommendation	36
Team's Reply 14 December 2022	36
CO - Code Optimization	37
Description	37
Recommendation	39
Team's Reply 14 December 2022	39
L04 - Conformance to Solidity Naming Conventions	40
Description	40
Recommendation	40
Team's Reply 14 December 2022	41
L09 - Dead Code Elimination	42
Description	42
Recommendation	42
Team's Reply 14 December 2022	42
L11 - Unnecessary Boolean equality	43
Description	43
Recommendation	43
Team's Reply 14 December 2022	43
L12 - Using Variables before Declaration	44
Description	44
Recommendation	44
Team's Reply 14 December 2022	44
L13 - Divide before Multiply Operation	45
Description	45
Recommendation	45
Team's Reply 14 December 2022	45
L14 - Uninitialized Variables in Local Scope	46

<b>Description</b>	<b>46</b>
<b>Recommendation</b>	<b>46</b>
<b>Team's Reply 14 December 2022</b>	<b>46</b>
<b>L15 - Local Scope Variable Shadowing</b>	<b>48</b>
<b>Description</b>	<b>48</b>
<b>Recommendation</b>	<b>48</b>
<b>Team's Reply 14 December 2022</b>	<b>48</b>
<b>Contract Functions</b>	<b>49</b>
<b>Contract Flow</b>	<b>86</b>
<b>Summary</b>	<b>87</b>
<b>Disclaimer</b>	<b>88</b>
<b>About Cyberscope</b>	<b>89</b>

## Contracts Review

<b>Github</b>	<a href="https://github.com/TangibleTNFT/usdr-contracts">https://github.com/TangibleTNFT/usdr-contracts</a>
<b>Commit</b>	ac5c4bf0e17df891c3a43daf2215238c84459dd1

## Audit Updates

<b>Initial Audit</b>	24th November 2022 <a href="https://github.com/cyberscope-io/audits/blob/main/TNGBL/v1/audit.pdf">https://github.com/cyberscope-io/audits/blob/main/TNGBL/v1/audit.pdf</a>
<b>Corrected phase 1</b>	15 December 2022 <a href="https://github.com/cyberscope-io/audits/blob/main/TNGBL/v2/audit.pdf">https://github.com/cyberscope-io/audits/blob/main/TNGBL/v2/audit.pdf</a>
<b>Corrected phase 2</b>	17 January 2022

# Testing Deploy

Contract Name	Explorer
USDR	<a href="https://testnet.bscscan.com/address/0xE4fCB9fDc79e2172a6734903EAE85C8cdc26D49">https://testnet.bscscan.com/address/0xE4fCB9fDc79e2172a6734903EAE85C8cdc26D49</a>
pDAI	<a href="https://testnet.bscscan.com/address/0xE2B7335874c7d6C653DA1b9250A25e6fFe485fBb">https://testnet.bscscan.com/address/0xE2B7335874c7d6C653DA1b9250A25e6fFe485fBb</a>
USDRMigration	<a href="https://testnet.bscscan.com/address/0xC60bBfA612364F6560B73228138AF1634a8bA42">https://testnet.bscscan.com/address/0xC60bBfA612364F6560B73228138AF1634a8bA42</a>
WadRayMath	<a href="https://testnet.bscscan.com/address/0x14CCD639dA77017a3ec6A1FB8da3067270676426">https://testnet.bscscan.com/address/0x14CCD639dA77017a3ec6A1FB8da3067270676426</a>
AddressProvider	<a href="https://testnet.bscscan.com/address/0xE04b59EeA56F57341D4a1e3e2989A0b5853D8205">https://testnet.bscscan.com/address/0xE04b59EeA56F57341D4a1e3e2989A0b5853D8205</a>
DAIBond	<a href="https://testnet.bscscan.com/address/0x82ab99DA543460a7e376195430cac4A6F8f185b4">https://testnet.bscscan.com/address/0x82ab99DA543460a7e376195430cac4A6F8f185b4</a>
IncentiveVault	<a href="https://testnet.bscscan.com/address/0x8cd4889264485050269A070578D7025a0BC275C1">https://testnet.bscscan.com/address/0x8cd4889264485050269A070578D7025a0BC275C1</a>
RWACalculator	<a href="https://testnet.bscscan.com/address/0x9C9891F7AeCde5964ff790f1276718fA9cdCB8e1">https://testnet.bscscan.com/address/0x9C9891F7AeCde5964ff790f1276718fA9cdCB8e1</a>
AffiliateExchange	<a href="https://testnet.bscscan.com/address/0xAAb3f7a587d68C9A1E406c4ab5f4177141A2a652">https://testnet.bscscan.com/address/0xAAb3f7a587d68C9A1E406c4ab5f4177141A2a652</a>
TNGBLPriceOracle	<a href="https://testnet.bscscan.com/address/0x3f628f6f3084460B197eEfBA2E465C1527C17054">https://testnet.bscscan.com/address/0x3f628f6f3084460B197eEfBA2E465C1527C17054</a>
TokenSwap	<a href="https://testnet.bscscan.com/address/0x9a3555ce5191bC44eE8bCaa8C5F343785F1Af5a0">https://testnet.bscscan.com/address/0x9a3555ce5191bC44eE8bCaa8C5F343785F1Af5a0</a>
TreasuryTracker	<a href="https://testnet.bscscan.com/address/0x64E40EE397324ada86B0D9B66798c3F4b3455913">https://testnet.bscscan.com/address/0x64E40EE397324ada86B0D9B66798c3F4b3455913</a>
USDRTreasury	<a href="https://testnet.bscscan.com/address/0x5CF531496A9571197c165243c88fa98c3d490285">https://testnet.bscscan.com/address/0x5CF531496A9571197c165243c88fa98c3d490285</a>

<b>USDRBonding</b>	<a href="https://testnet.bscscan.com/address/0x054C017372a96b59b0D34D1ca58CF2019140d7dE">https://testnet.bscscan.com/address/0x054C017372a96b59b0D34D1ca58CF2019140d7dE</a>
<b>USDRExchange</b>	<a href="https://testnet.bscscan.com/address/0xeae9b78601aEcdDC4581299f22761607c844c9eb">https://testnet.bscscan.com/address/0xeae9b78601aEcdDC4581299f22761607c844c9eb</a>
<b>USDRExchangeProxy</b>	<a href="https://testnet.bscscan.com/address/0x1bc20907d0B2AbD05fa0915D461e28Eca270fD56">https://testnet.bscscan.com/address/0x1bc20907d0B2AbD05fa0915D461e28Eca270fD56</a>
<b>GoldPurchaseManager</b>	<a href="https://testnet.bscscan.com/address/0x395453Ce327586e2749784730e3a9C06db3d0a17">https://testnet.bscscan.com/address/0x395453Ce327586e2749784730e3a9C06db3d0a17</a>
<b>GoldSellManager</b>	<a href="https://testnet.bscscan.com/address/0x2aD1cDeE617ea36f49456af31f1A967a9113a954">https://testnet.bscscan.com/address/0x2aD1cDeE617ea36f49456af31f1A967a9113a954</a>
<b>REPurchaseManager</b>	<a href="https://testnet.bscscan.com/address/0x6C61C6745984F0fA89606E6CF87d2A14f107FD3E">https://testnet.bscscan.com/address/0x6C61C6745984F0fA89606E6CF87d2A14f107FD3E</a>
<b>ReSellManager</b>	<a href="https://testnet.bscscan.com/address/0x675358F10e8708839E604501e7385a74FC291F17">https://testnet.bscscan.com/address/0x675358F10e8708839E604501e7385a74FC291F17</a>
<b>LiquidityTokenMath</b>	<a href="https://testnet.bscscan.com/address/0x5CC78a6C7Ed42f3E26dC1593f3E7893526353955">https://testnet.bscscan.com/address/0x5CC78a6C7Ed42f3E26dC1593f3E7893526353955</a>
<b>TNGBLLockedValue</b>	<a href="https://testnet.bscscan.com/address/0xebBca671b43006994Af072ecc761229BFaE5f118">https://testnet.bscscan.com/address/0xebBca671b43006994Af072ecc761229BFaE5f118</a>
<b>WrappedUSDR</b>	<a href="https://testnet.bscscan.com/address/0x48528765e6458ddc6a48B6fE10eF4eFe08a4f50a">https://testnet.bscscan.com/address/0x48528765e6458ddc6a48B6fE10eF4eFe08a4f50a</a>



# Source Files

Filename	SHA256
<b>AddressAccessor.sol</b>	46ad8af4ff314b835974a6e026ce942ff3f56fd5c319c68d3f128c670b00a169
<b>AddressProvider.sol</b>	e463c6d775f0ad1c4c5bef219b2391fefb0d264c45b14d848e9d8a0762be2eb4
<b>AffiliateExchange.sol</b>	9003678791100a2fb76b982cd167eb59d8179227d77b888e77747f0f6e872717
<b>constants/addresses.sol</b>	c31754a2076cb7d559c2bf0f1df0a8a63da1e6eead5d879e019d4dd7d75b257e
<b>constants/constants.sol</b>	7d2a05b12ba5b135dfbc67ab2693d65a82817e5f3d7a40832ba32847ba318073
<b>constants/roles.sol</b>	493938a66b0c64554700a1b7c1f2233709889f04ff961cca33a27f034358068e
<b>DAIBond.sol</b>	9a80923bdc8ff83e41752f6e2bf7325f53ea76aa9d936f04db640c73ab23568f
<b>exchangeHelpers/CurveWrapper.sol</b>	2526ceb9cb5618a2cf467c9af0db65db761f2271d74c6d6e6ac2185f1c654a4a
<b>helpers/LiquidityTokenMath.sol</b>	50da2cb411eb82e34385a7a1b4527473b99348983c01e018a85d7c63f1704900
<b>IncentiveVault.sol</b>	5d830e1d1335998d956109274caea5591262823f1017c449c5271b7fb06d5b95
<b>interfaces/IExchange.sol</b>	b1eac05067975153ee292030fcdf7b13c0bcf811f3f1fc7adc520732789216af
<b>interfaces/ILiquidityManager.sol</b>	ce89617a1b542c489905cf5caa47f60cde10a1d7f53538a91a922bb51fc864cc
<b>interfaces/ILiquidityTokenMath.sol</b>	ba076342c2a184981c420825e099837e5ec5f534ab07856dc5132e1623958769
<b>interfaces/IPriceOracle.sol</b>	ae7217ef73c1e591bb8c2011b02725ebb9a23858483f6386f83918cdec86f0b
<b>interfaces/IRateProvider.sol</b>	c7fb782fd9b4fe9463af17758064f98e808e72a0e492403efc3ab2983fdcb972

<b>interfaces/IRWACalculator.sol</b>	d51f50b9c236b701e7f5718832ac39da016f52cea295f2ca51fd8f8a2d375f6c
<b>interfaces/ITokenSwap.sol</b>	95d277d47fbfe5cfd4a628f6070911535f243369def359b3fdf752fdb1bc11b8
<b>interfaces/ITreasury.sol</b>	18d830188842c79a31cdc10529ced43d1244024e8360d24959fbc540c94a4daf
<b>interfaces/ITreasuryTracker.sol</b>	907f6bad9a6c3b544c1d87f1d240342eee4e415caa919f59aa002be3ba7a7f61
<b>interfaces/IUSDR.sol</b>	20dbd287b49061fb82e184c66e9b2514ad711df2e61bdb904cd5cde6ec4f3e1e
<b>managers/GoldPurchaseManager.sol</b>	d0a7570c64fe2690da8f05fea4f061c782e6ae0a5483b5c378a7d02d4f37a26a
<b>managers/GoldSellManager.sol</b>	5d66782cd962cdaef72c0b7048eb98a045f9b2490b11d999254c74d0f4d17e7d
<b>managers/LiquidityManager.sol</b>	74f881f75ca317027043842ff18ee82e3be1dcb80f264f6638b7300030c6dd86
<b>managers/OnSaleTracker.sol</b>	3a8dc69619590a69e4a58da5338ee84a282cb445cc5a40e3d67a1648e0d170c9
<b>managers/PurchaseManager.sol</b>	eab50fc8a9f54d0711f6574d2eeb3a385356e8478d2b0a2f27d0bda9f7575409
<b>managers/REPurchaseManager.sol</b>	ea93548deb60fa54eb82d03735bae0e26759ccb287b9581a2db39b437aa9a882
<b>managers/RESellManager.sol</b>	62aee70bf24b91fce1e7bfa909d7ea1ab6c1edbf1a96c7824895d64a92e4c7cd
<b>managers/TNGBLLiquidityManager.sol</b>	1d0529019857c343be6db059108ed1029d88c8235d1321e7ce3bb764bea24235
<b>mocks/MockDAIUSDOracle.sol</b>	e4ac02cbf89a705bfd92c4048c7c6c61a83e3b57efa9c68b6be478b41819ec09
<b>mocks/MockERC20.sol</b>	7da7e85841d1a129c3c8a3f5a90946d5b67171aa999425bb1926540ab179a1d7
<b>mocks/MockOracle.sol</b>	bbf25e4605c4e4db4c31347926822f62460a0fd772afc662d0e93a7b7b502246
<b>mocks/MockRouter.sol</b>	5355d45c89f5ca4eadbb3f42669399a1dbc4161217f4fdc57354243725f2b605

<b>mocks/MockTreasury.sol</b>	b8e333e74b74ab4463b8093cdd350c87d2620784c6fdcb09b2ea5780e2a91531
<b>RWACalculator.sol</b>	ce4607e1b89cd49f37bf0c0c0469e6ef4477678640da9184bf550f2822085f47
<b>strategies/IStrategy.sol</b>	d31577b6d9baae01d854796d8f08261290fc6890271325dd4ae5f06933902186
<b>strategies/sushi/IMiniChefV2.sol</b>	a8c2b43069a70f4edc0927e764620eb31f874ab6898e285ed375bedd2daa1d49
<b>strategies/sushi/SushiLPStrategy.sol</b>	81231011b0ba41e89f9e984151a2c914752cafbfac16df8d2109caacef68f957
<b>tangibleInterfaces/ICurrencyFeed.sol</b>	2963764bbc69fa849f23f8fa9dd0a72934b7075e5a015bda21beaa090e130668
<b>tangibleInterfaces/IInstantLiquidity.sol</b>	c20fb58fda62afee9c2827bd1e97593f9e6e69039091501043593b181d5e59f2
<b>tangibleInterfaces/IPriceOracle.sol</b>	2c5550244dbe7a670f802ce047735b83046a05f1640c3a850e6b53470146f140
<b>tangibleInterfaces/ITangibleInterfaces.sol</b>	d9cdd973e5dfd240a66bd5e13e03474b81be5682b97e41c01b9595273f8054a0
<b>tangibleInterfaces/ITangibleMarketplace.sol</b>	ea7e05dece3b0256027e7e8b0daa89e1676e4e9bf3fb39a78728bc0873c11243
<b>tangibleInterfaces/ITangiblePiNFT.sol</b>	ea4fc5a17bb5a8a1c3237fe87195d53a1310cb83f58e9db6f0df89eb571cc0e3
<b>tangibleInterfaces/ITangibleRentShare.sol</b>	2b3c3508bc80f59984f65392983ea1737f13cd4f21ed6e8b2f941f2b550c0fcb
<b>tangibleInterfaces/ITangibleRevenueShare.sol</b>	01ec5c77269060ee39e176e835913ec23b1e0cf72c277a0ab57611a0a5913ac0
<b>TNGBLLockedValue.sol</b>	5e6d60767026c31b183ea80f9bbdaa50239527c137297417a65ab599d221ae1c
<b>TNGBLPriceOracle.sol</b>	2cee6ecdaa1478a4ffba5846e2266b245c6d49d95dffa6d2663dc1121674437
<b>tokens/interfaces/IMintableERC20.sol</b>	49695950b7dae818cdb5e2dd5d67ba5b0f63bb129089a349da58f986532f145f
<b>tokens/interfaces/ITangibleERC20.sol</b>	933e943c676ae403d1c2594985d54de001fa4125bc98a900ddfe46145460b116

<b>tokens/pDAI.sol</b>	296342f000bb85c43f0cab1e11ef3d257c c71044378022674835cf6b637b0ae8
<b>tokens/USDR.sol</b>	23e7581f6b671543e55d59241fba4d28e 3a95706ae416b41becf9f01f4783984
<b>tokens/USDRMigration.sol</b>	3124f3d094fa4a2317c04f26a0256e2989 eacd4dfe8f3c2f729d7a75b350a728
<b>tokens/WadRayMath.sol</b>	af5f9434986200e7960994953c4964c7a7 ec275b2f18d46bd0e10a5351ba9dc1
<b>tokens/WUSDR.sol</b>	22fd9f99071c498de6d988e3646e645c3a e5e4f193a4c10835a75136cd98152b
<b>TokenSwap.sol</b>	d6df5354075fef529ba95d7626e8302ddf 0ee1031dabd8d158e153d04916f048
<b>TreasuryTracker.sol</b>	f083e0eab88f134875ab71743615b2263c c7cf025afd1977698a1d93288e4f9d
<b>USDRBonding.sol</b>	25e1a6577269848d8cb29d8cfc4163eca dcd8ef5c8d4ac48850246551e33dcd5
<b>USDRExchange.sol</b>	a7276250db8ec1120aec122dd1bdf5629 bb40d58ecd736a2eb37e7a9c60e871e
<b>USDRExchangeProxy.sol</b>	296f086f87701597cae8954762c230fd8fb 6b4804c1d689331defc5c21dabcc7
<b>USDRTreasury.sol</b>	d7a34e381c371d53971a5fd61dde3cc4a ccf0da576b1eee40583f61bdae711ef
<b>wUSDRRateProvider.sol</b>	e8eea2bc5b1f6b6e8caa35e5cc0050ccc 97a0d71fdedb12114d2d97b40fdbab8

# DAO Infrastructure Architecture Review

The Tangible contracts depend on the correct administrator's configuration. One category of configuration is the allocation of funds. The amount of funds that will be moved to the Bond program, the Affiliate and Incentive features are determined by the administrators. Another category of configuration is direct state manipulation.

## Recommendation

We advise the team to create contracts that will implement the expected business logic. These contracts could communicate with Tangible contracts and replace the required human interaction. For instance, a contract could check the accumulated amounts of the treasury and decide how to distribute the funds to the vaults like (Bond, Affiliate, Incentive), liquidity, reserves, USDR binding balance, etc. This methodology will remove the error-prone human interaction and make the ecosystem operate as autonomously as possible.

## Team's Reply 20 December 2022

The team has acknowledged that this is not a security issue and states:

*"Many DAOs allow changes to the business logic by the DAO multisig. This allows holders to vote on changes to the business logic and members of the multisig to implement those changes."*

*"This is in our mind is definitely a feature not a bug. No one has ever created a stablecoin backed by real estate before and the exact percentages of the backing and many other parts of the business logic may need to be changed based on data that we gather whilst live. The ability to change this logic via a tx rather than a redeployment and tokenswap is very much so deliberate."*

# Decimal Architecture

The Tangible contracts utilize many tokens that interact with each other. Each token may have a different amount of decimals. As a result, the contracts are normalizing the tokens in order to proceed with the calculations.

The contracts do not have a single point of decimals normalization mechanism. As a result, a lot of issues may be produced:

There are many sections that are **repeating the decimals normalization logic**.

```
function _scaleAmount(  
    uint256 amount,  
    uint8 fromDecimals,  
    uint8 toDecimals  
) private pure returns (uint256)  
  
function _convertToCorrectDecimals(  
    uint256 price,  
    uint8 salePriceDecimals,  
    uint8 treasuryTokenDecimals  
)  
  
function scaleToUnderlying(uint256 amount) external view returns (uint256)
```

Creates **unnecessary dependencies** between contracts.

```
// IncentiveVault.sol  
uint256 amount = (IExchange(exchange).scaleToUnderlying(  
    IERC20(USDR).totalSupply()  
) * apr) / 3_650_000;
```

Using **fixed values from token addresses** that may change. The [Decimal Scale Misconcern](#) finding describes a related issue.

```
uint256 amount = ((minted - redeemed) * 1e26) /  
IPriceOracle(tngblOracle).quote(1e18);
```

## Recommendation

The Tangible Ecosystem could implement a single point of decimals handling functionality similar to the Address Provider contract. All decimal-related calculations should always use the token's decimals and not a fixed number.

## Team's Reply 14 December 2022

The team has acknowledged that this is not a security issue and states:

*"We had different functions for different needs when we needed to handle decimals. They were made by different devs and work the same so essentially, they could be merged into one, but that is just code optimization, and our main goal and functionality of the contracts are NOT changed nor affected. Hardcoded values are there because decimals of TNGBL token, DAI, and our USDR will not change, so we decided to do it like this to optimize gas spending, and contract size."*

# Roles Architecture

The Tangible contracts have a role-based access mechanism. Every contract contains its own access layer. The roles that are used are:

- BURNER
- MINTER
- CONTROLLER
- TRACKER
- ROUTER\_POLICY

The team is advised to use a multi-sig wallet which can provide an additional level of security. Additionally, the team should make sure that each role has a clear set of responsibilities, and that these responsibilities are not overlapping or ambiguous.

The following example depicts a potential conflict between the CONTROLLER role and the treasury address. Since this method is solely accessed by the treasury address then the role-based permissions are redundant.

```
function rebase(uint256 supplyDelta)
    external
    onlyRole(CONTROLLER_ROLE)
    whenNotPaused
    {
        (address treasury, address exchange) = abi.decode(
            addressProvider.getAddresses(
                abi.encode(TREASURY_ADDRESS, USDR_EXCHANGE_ADDRESS)
            ),
            (address, address)
        );
        require(msg.sender == treasury, "caller is not treasury");
```

## Team's Reply 14 December 2022

The team has acknowledged that this is not a security issue and states:

*"We have general architecture in place for controller role and this is in place to make sure we are ok in possible migrations for this particular function you mentioned – address in AddressProvider could return treasury that doesn't have controller role."*



# Contract Diagnostics

● Critical   ● Medium   ● Minor / Informative

Severity	Code	Description	Status
●	AFI	Affiliate Token Issue	Acknowledged
●	STI	Staking Token Issue	Acknowledged
●	DMI	Defractionalize Manipulation Issue	Acknowledged
●	TBI	Token Balance Inconsistency	Acknowledged
●	PRD	Pair Reserves Diversion	Acknowledged
●	TAZFA	Transferred Amount Zero Fees Assumption	Acknowledged
●	AIC	Arguments Inconsistency	Acknowledged
●	ELFM	Exceeds Fees Limit	Acknowledged
●	DSM	Decimal Scale Missconcern	Acknowledged
●	PIL	Potential Infinite Loop	Acknowledged
●	STC	Succeed Transfer Check	Acknowledged
●	CO	Code Optimization	Acknowledged
●	L04	Conformance to Solidity Naming Conventions	Acknowledged

●	L09	Dead Code Elimination	Acknowledged
●	L11	Unnecessary Boolean equality	Acknowledged
●	L12	Using Variables before Declaration	Acknowledged
●	L13	Divide before Multiply Operation	Acknowledged
●	L14	Uninitialized Variables in Local Scope	Acknowledged
●	L15	Local Scope Variable Shadowing	Acknowledged

## AFI - Affiliate Token Issue

<b>Criticality</b>	medium
<b>Location</b>	contracts/AffiliateExchange.sol#82
<b>Status</b>	Acknowledged

### Description

There are functions that are used to swap a specified amount of tokens. There fee or tax is an amount that is charged when tokens are swapped. According to the specification, the swapped amount could potentially be less than the expected amount. This may produce inconsistency between the expected and the actual behavior.

For instance, the contract utilizes the variable `affiliatePayout` which contains the total rewarded amount.

1. Every affiliate participation requires an amount of USDR. If the contract's USDT is not sufficient, then the proportional amount of TGBL is swapped to cover the cap.
2. The `swapFromTNGBL` is used to swap TGBL for USDR. The `swapFromTNGBL` method applies fees on the transaction. Thus, the actual total rewards will be lower than the aggregated amount.
3. During the redemption phase, the `sendRewards` will send more than the actual contract's reserves.

This will cause the method to revert since the balance of the contract will be less than the calculated `_pending` variable.

```
function _mint( uint256 amountIn, address receiver, uint256 affiliatePayout )
private {
    (
        address underlying,
        address exchange,
        address tngbl,
        address oracle,
        address usdr
    ) = abi.decode(
        //..
        uint256 excessUSDR = IERC20(usdr).balanceOf(address(this)) - _pending;
```

```
if (affiliatePayout > excessUSDR) {
    uint256 mintExtra = affiliatePayout - excessUSDR;
    uint256 tngblPrice = IPriceOracle(oracle).quote(1e18);
    uint256 mintExtraInTNGBL = (mintExtra * 1e27) / tngblPrice;
    IERC20(tngbl).approve(exchange, mintExtraInTNGBL);
    USDRExchange(exchange).swapFromTNGBL(
        mintExtraInTNGBL,
        0,
        address(this)
    );
}
_pending += affiliatePayout;
}
```

## Recommendation

The team is advised to take into consideration the actual amount that has been transferred instead of the expected.

## Team's Reply 14 December 2022

The team has acknowledged that this is not a security issue and states:

*"The contracts in question are there in marketing and incentive purposes and they are tailored to the current situation. When they become obsolete, we will replace them. For now, there is no fee on minting with TNGBL and these contracts work in alignment with that. If there will be a time to introduce a fee, we will replace those contracts with new ones."*

## STI - Staking Token Issue

<b>Criticality</b>	medium
<b>Location</b>	contracts/DAIBonding.sol#102
<b>Status</b>	Acknowledged

### Description

There are functions that are used to swap a specified amount of tokens. The fee or tax is an amount that is charged when tokens are swapped. According to the specification, the swapped amount could potentially be less than the expected amount. This may produce inconsistency between the expected and the actual behavior.

For instance, during the deposit phase, the contract mints USDR tokens in order to cover the amount that is going to be redeemed.

1. During the mint phase, if the USDR is not sufficient to cover the redeemed amount, the TNGBL is swapped.
2. The swapFromTNGBL method is used to swap TNGBL for USDR. The swapFromTNGBL applies fees but these fees are not deducted from the redeemed amount.

As a result, during the redeem phase, the user will not be able to claim the tokens since the actual USDR tokens of the contract will be less than the expected USDR amount. The expression `uint256 remainingUSDR = v.usdr - amount;` will revert.

```
function claim(uint256 amount) public {
    address who = msg.sender;
    uint256 maxAmount = claimable(msg.sender);
    //..
    uint256 totalClaimed = v.claimed + amount;
    USDRBonding(bonding).withdraw(amount, who);
    uint256 remainingUSDR = v.usdr - amount;
    //..
}
```

### Recommendation

The contract should take into consideration the corresponding fees in the total vested amount aggregation.

## Team's Reply 14 December 2022

The team has acknowledged that this is not a security issue and states:

*"The contracts in question are there in marketing and incentive purposes and they are tailored to the current situation. When they become obsolete, we will replace them. For now, there is no fee on minting with TNGBL and these contracts work in alignment with that. If there will be a time to introduce a fee, we will replace those contracts with new ones."*

## DMI - Defractionalize Manipulation Issue

<b>Criticality</b>	minor / informative
<b>Location</b>	contracts/USDTreasury.sol#325
<b>Status</b>	Acknowledged

### Description

The nft defractionalize mechanism depends on the external function onERC721Received. The external function onERC721Received can be called by any user. Hence, the defractionalize mechanism could be manipulated.

```
if (
    (lastReceivedNFT == address(ftnft.tnft())) &&
    (lastReceivedTokenId == ftnft.tnftTokenId())
) {
    tracker.ftnftTreasuryPlaced(address(ftnft), tokenIds[0], false);
    tracker.tnftTreasuryPlaced(
        lastReceivedNFT,
        lastReceivedTokenId,
        true
    );
} else {
    tracker.updateFractionData(address(ftnft), tokenIds[0]);
}
```

### Recommendation

The contract should take into consideration the public access of the onERC721Received() method.

### Team's Reply 14 December 2022

The team states:

*"Since this is not crucial to treasury functioning, we will address this while migrating to V3."*

## TBI - Token Balance Inconsistency

<b>Criticality</b>	minor / informative
<b>Location</b>	contracts/USDR.sol#61
<b>Status</b>	Acknowledged

### Description

The contract balance is deleted when the account balance is equal to the transaction amount. On the contrary, if the transaction amount is greater than the account balance, then the balance is set to zero. As a result, it creates an inconsistency with the logic of the code. Since both statements fulfill the purpose.

```
if (accountBalance == amount) {
    _totalSupply -= _balances[account];
    delete _balances[account];
} else {
    uint256 amount_ = amount.wadToRay().rayDiv(liquidityIndex);
    if (amount_ > _balances[account]) {
        amount_ = _balances[account];
    }
    _totalSupply -= amount_;
    _balances[account] -= amount_;
}
```

### Recommendation

We state that both deleting and setting to zero produce the same result. But in terms of code readability and maintainability, the contract should manage the balances in the same manner.

### Team's Reply 14 December 2022

The team has acknowledged that this is not a security issue and states:

*"This is not much of a problem, bytecode is the same, delete does the same thing as setting to 0."*



## PRD - Pair Reserves Diversion

<b>Criticality</b>	minor / informative
<b>Location</b>	contracts/USDR.sol#95
<b>Status</b>	Acknowledged

### Description

The contract balance could diverge in relation to the pair reserve balance. The pair addresses keep the reserve ratio proportionally to the balance of the token. The amount of the reserves is cached in the pair contract. So if the balance changes because of the rebase, then the pair should be synced manually.

```
function rebase(uint256 supplyDelta) external onlyRole(CONTROLLER_ROLE)
    whenNotPaused
{
    (address treasury, address exchange) = abi.decode(
        addressProvider.getAddresses(
            abi.encode(TREASURY_ADDRESS, USDR_EXCHANGE_ADDRESS)
        ),
        (address, address)
    );
    require(msg.sender == treasury, "caller is not treasury");
    uint256 ts = totalSupply();
    if (supplyDelta > 0) {
        supplyDelta =
        IExchange(exchange).scaleFromUnderlying(supplyDelta);
        uint256 maxSupplyDelta = MAX_UINT128 - ts;
        if (supplyDelta > maxSupplyDelta) {
            supplyDelta = maxSupplyDelta;
        }
        if (supplyDelta > 0) {
            liquidityIndex = (liquidityIndex * (ts + supplyDelta)) / ts;
            int128[7] memory delta;
            delta[6] = int128(uint128(totalSupply() - ts));
            IExchange(exchange).updateMintingStats(delta);
        }
    }
    emit Rebase( block.number, block.timestamp / 1 days, ts, supplyDelta,
    liquidityIndex);
}
```

## Recommendation

The contract should call the “sync” method from the pair contract every time the rebase mechanism is taking place.

## Team’s Reply 14 December 2022

The team has acknowledged that this is not a security issue and states:

*“Curve protocol doesn’t require syncing.”*

## TAZFA - Transferred Amount Zero Fees Assumption

<b>Criticality</b>	minor / informative
<b>Location</b>	contracts
<b>Status</b>	Acknowledged

### Description

The contract performs token transactions without taking into account possible transaction fees. Additionally, the token properties are mutable. The contract owner has the authority to add any tokens. As a result, the actual transferred amount might diverge.

The following table depicts the inconsistency that may be produced

Function Calls	Fees	Transferred Amount	Assumed Transfer Amount
IERC20.transfer(user, 100);	0%	100	100
IERC20.transfer(user, 100);	10%	90	100

```
IERC20(underlying).transferFrom(msg.sender, address(this), amountIn);
IERC20(underlying).transfer(msg.sender, amount);
IERC20(underlying).transferFrom(onBehalfOf, address(this), deposit);
IERC20(underlying).transfer(treasury, amountIn);
IERC20(underlying).transferFrom( msg.sender, address(this), depositAmount)
IERC20(underlying).transferFrom( treasury, address(this), underlyingAmount);
IERC20(underlying).transferFrom( treasury, address(this), underlyingAmount);
..
..
```

### Recommendation

The contract could take into consideration the underlying fees, or assert in case of the underlying token apply fees.

## Team's Reply 14 December 2022

The team states:

*"We're not planning to use underlying tokens that incur transfer fees in those places when we do transfers, and we don't have in plan to change any of that. And most of our contracts don't hold any tokens except Treasury contracts."*

## AIC - Arguments Inconsistency

<b>Criticality</b>	minor / informative
<b>Location</b>	contracts/TokenSwap.sol#L25  contracts/AffiliateExchange.sol#L129  contract/USDRTreasury.sol#L421
<b>Status</b>	Acknowledged

### Description

The contract utilizes arguments that might diverge from the expected value.

The contract receives the `routerPathReverse` and the `routerPath` as arguments. Those arguments should be the opposite of each other but this is not guaranteed by the method's checks.

```
function addSwapRoute(  
    address router,  
    address tokenInAddress,  
    address tokenOutAddress,  
    address[] calldata routerPath,  
    address[] calldata routerPathReverse  
) external onlyRole(ROUTER_POLICY_ROLE)
```

The contract receives the argument `total` as an aggregation of the `amounts`. The `total` arguments could be greater or lower than the aggregation of the amount array. Thus, the `total` argument could produce an inconsistency.

```
function sendRewards(  
    address batchSender,  
    uint256 total,  
    address[] calldata recipients,  
    uint256[] calldata amounts  
) external whenNotPaused onlyRole(CONTROLLER_ROLE)
```

The contract is utilizing a `placed` argument on the function `updateTrackerTnft`. The `placed` argument determines if the NFT is minted to an account. The argument could produce inconsistency if it's true but the NFT already exists in the `TreasureTracker`.

```
function updateTrackerTnft(  
    address tnft,  
    uint256 tokenId,  
    bool placed  
) internal
```

## Recommendation

The contract could compose the `routerPathReverse` as the reserve path from `routerPath`.

The contract could calculate the `total` amount from the sum of the amounts.

The contract could incorporate the function `ownerOf(tokenId)` of the ERC721 interface or the internal structures instead of `placed` arguments.

## Team's Reply 14 December 2022

The team has acknowledged that this is not a security issue and states:

*“addSwapRoute – only a config function and data provided are pairs on defi exchanges that we use, uniswap and curve so as long uniswap and curve don't change anything we are ok. Because of the nature of the rewards and that their calculation is done off-chain – we made sure that we are the only ones who can send rewards. UpdateTrackerTnft is made so that it can be called from contract functions and none of those functions is without proper role.”*

## ELFM - Exceeds Fees Limit

<b>Criticality</b>	minor / informative
<b>Location</b>	contracts/USDRExchange.sol#L129
<b>Status</b>	Acknowledged

### Description

The contract owner has the authority to increase the fees over the denominator value of 10000. This could be produced by calling the `setFees` function with a value greater than 10000.

```
function setFees(uint256 depositFee_, uint256 withdrawalFee_)  
    external  
    onlyRole(DEFAULT_ADMIN_ROLE)  
{  
    depositFee = depositFee_;  
    withdrawalFee = withdrawalFee_;  
}
```

### Recommendation

The contract could embody a check for the maximum acceptable value.

### Team's Reply 14 December 2022

The team states:

*"Acknowledged, but right now it is not worth redeployment and we will address it in the future V3 release."*

## DSM - Decimal Scale Missconcern

<b>Criticality</b>	minor / informative
<b>Location</b>	contracts/USDRExchange.sol#L274
<b>Status</b>	Acknowledged

### Description

The function `_computeScale` assumes that the first argument is always lower than the second argument. An unexpected value will be produced If the first argument is greater than the second argument. For instance,

Function Call	Result
<code>_computeScale(1, 2)</code>	10
<code>_computeScale(2, 1)</code>	3402823669209384634633746074317682114560

```
function _computeScale(uint8 usdrDecimals, uint8 underlyingDecimals)
    private
    pure
    returns (uint256 scale)
{
    assembly {
        switch lt(usdrDecimals, underlyingDecimals)
        case 0 {
            scale := shl(
                128,
                exp(10, sub(usdrDecimals, underlyingDecimals))
            )
        }
        default {
            scale := exp(10, sub(underlyingDecimals, usdrDecimals))
        }
    }
}
```



## Recommendation

The team is advised to carefully check if the implementation follows the expected business logic. The contract could embody a check for not allowing the first argument to be greater than the second one or could handle this case.

### Team's Reply 14 December 2022

The team has acknowledged that this is not a security issue and states:

*"If we change the underlying to something with less than 9 decimals (e.g. USDC), computeScale will in fact return a huge number. This, however, is not unexpected. scale[To|From]Underlying rely on that behavior in order to decide whether amounts need to be scaled up or down."*

## PIL - Potential Infinite Loop

<b>Criticality</b>	minor / informative
<b>Location</b>	contracts/PurchaseManager.sol#L147
<b>Status</b>	Acknowledged

### Description

If the function “ITokenSwap(tokenSwap).quoteOut” returns a value that is lower than the amount then an infinite loop may be produced.

```
do {
    reserveAmount = ITokenSwap(tokenSwap).quoteOut(
        underlying,
        address(paymentToken),
        calcAmount
    );

    if (reserveAmount < amount) {
        calcAmount =
            calcAmount +
            _convertToCorrectDecimals(
                reserveAmount - amount,
                paymentDecimals,
                underlyingDecimals
            ) +
            10**uint256(underlyingDecimals); // add 1 dollar
    }
} while (reserveAmount < amount);
```

### Recommendation

It is recommended to add a limitation in the potential number of loops due to the fact that the loops may escalate. As a result, the loop could easily exceed the maximum block size and make the contract run out of gas.

### Team's Reply 14 December 2022

The team has acknowledged that this is not a security issue and states:

*“This was implemented that way intentionally. It is only used when calculating stable to stable because unfortunately, curve contract doesn’t have a way to provide input with exact output, so this was our only gas-optimized option.”*

## STC - Succeed Transfer Check

<b>Criticality</b>	minor / informative
<b>Location</b>	contracts/AffiliateExchange.sol#L68,79  contract/DAIBond.sol#L68,79  contracts/IncentiveVault.sol#36,52  contract/USDRBonding.sol#44,51,110,119,122,145  contracts/USDRExchange.sol#230  contract/USDRTreasury.sol#126  contracts/LiquidityManager.sol#109,213,231  contract/TNGBLLiquidityManager.sol#35,79,101,115,165,166,249,265  contracts/USDRMigration.sol#67
<b>Status</b>	Acknowledged

### Description

According to the ERC20 specification, the transfer methods should be checked if the result is successful. Otherwise, the contract may wrongly assume that the transfer has been established.

```
IERC20(usdr).transfer(msg.sender, excessUSDR);  
erc20.transfer(msg.sender, balance);  
  
IERC20(token).transfer(msg.sender, balance);  
  
IERC20(token).transfer(msg.sender, balance);  
IERC20(underlying).transfer(msg.sender, amount);  
  
IERC20(token).transfer(receiver, amount);  
IERC20(token).transfer(_owner, amount);  
IERC20(underlying).transferFrom(onBehalfOf, address(this), deposit);
```

## Recommendation

The contract should check if the result of the transfer methods is successful.

### Team's Reply 14 December 2022

The team has acknowledged that this is not a security issue and states:

*"Acknowledged, but not critical at this point. We're not using tokens, that don't follow the standard (like USDT)."*

## CO - Code Optimization

Criticality	minor / informative
Location	contracts/USDRBonding.sol#L57,155
	contracts/USDR.sol#L95
	contracts/RWACalculator.sol#L451
	contract/TokenSwap.sol#L68
Status	Acknowledged

## Description

There are code segments that could be optimized. A segment may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer operations.

The variable `percentage` and the function `setPercentage` are not utilized in contract implementation. Hence, they are redundant.

```
uint16 public percentage;
function setPercentage(uint16 value) external onlyRole(DEFAULT_ADMIN_ROLE) {
    percentage = value;
}
```

The `fetchPaymentTokenAndAmountTnft` method is using an entire code segment that is not used anywhere.

```

HelperStruct memory hs;
hs.weSellAt = new uint256[](1);
hs.lockedAmount = new uint256[](1);
hs.tokenIds = new uint256[](1);
hs.tokenIds[0] = tokenId;
ITNFTPriceManager priceManager = _getPriceManager();
(hs.weSellAt, , , hs.lockedAmount) = priceManager
    .itemPriceBatchTokenIds(

```

```
ITangibleNFT(tnft),  
paymentToken,  
hs.tokenIds  
);
```

The internal function `_verifyBacking` is utilized with constant arguments. As a result, the function arguments are redundant.

```
function _verifyBacking(uint8 threshold, bool includeTNGBL) internal view {  
    address usdr = _fetchAddress(USDR_ADDRESS);  
    ITreasury.TreasuryValue memory tv = getTreasuryValue();  
    uint256 scaledMarketCap = IERC20(usdr).totalSupply() * 1e9;  
    uint256 backing = tv.total;  
    if (!includeTNGBL) {  
        backing = backing - tv.tngbl - tv.tngblLiquidity.tngbl;  
    }  
    require(  
        (scaledMarketCap * threshold) / 100 <= backing,  
        "insufficient backing"  
    );  
}
```

The `exchange` function doesn't check if the corresponding path exists on the `swappers` mapping, the caller should be aware of this instead of a generic revert error.

```
function exchange(  
    address tokenIn,  
    address tokenOut,  
    uint256 amountIn,  
    uint256 amountOut,  
    EXCHANGE_TYPE exchangeType  
) external override returns (uint256) {  
    bytes memory tokenized = abi.encodePacked(tokenIn, tokenOut);  
    address[] memory path = swappers[tokenized].path;
```

The `exchange` function performs an extra transaction. The contract initially transfers the exchanged amount to the contract and then from the contract to the `msg.sender`.

```
if (exchangeType == EXCHANGE_TYPE.EXACT_INPUT) {
    amounts = IUniswapV2Router01(swappers[tokenized].router)
        .swapExactTokensForTokens(
            amountIn,
            amountOut,
            path,
            address(this),
            block.timestamp + 30 // on sushi?
        );
} else if (exchangeType == EXCHANGE_TYPE.EXACT_OUTPUT) {
    amounts = IUniswapV2Router01(swappers[tokenized].router)
```

## Recommendation

The authors are advised to rewrite some code segments so the runtime will be more performant.

The contract could remove redundant arguments and functions.

The contract could have an informative message that the pair does not exist.

The contract could swap the exchanged amount directly to the `msg.sender`.

## Team's Reply 14 December 2022

The team has acknowledged that this is not a security issue and states:

*“Dead code just increases the size of the contract and is only concerned in deployment. The end user doesn’t have any inconvenience about this. We will address this in future version V3. The Exchange contract is a helper contract where we wrapped Uniswap v3 and Curve and we only use a subset of tokens that we need, so we save space and gas needed thereby making sure our protocol uses only what we need.”*



## L04 - Conformance to Solidity Naming Conventions

<b>Criticality</b>	minor / informative
<b>Location</b>	contracts/AddressAccessor.sol#L24,12  contracts/interfaces/ITokenSwap.sol#L5  contracts/USDRExchange.sol#L240  contracts/USDRTreasury.sol#L75,478,76,78,477,77  contracts/RWACalculator.sol#L407,148
<b>Status</b>	Acknowledged

### Description

Solidity defines a naming convention that should be followed. Rule exceptions:

- Allow constant variable name/symbol/decimals to be lowercase.
- Allow `_` at the beginning of the mixed\_case match for private variables and unused parameters.

```
_addressProvider  
EXCHANGE_TYPE  
_purchaseStableMintedRedeemedThreshold  
_tokenIds  
_purchaseStableMarketcapThreshold  
_incentiveThreshold  
_nft  
_tngblThreshold  
_years  
...
```

### Recommendation

Follow the Solidity naming convention.

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

## Team's Reply 14 December 2022

The team has acknowledged that this is not a security issue and states:

*“Naming is not too big of a problem, different team members have different coding styles, we will address it in possible future migrations.”*

## L09 - Dead Code Elimination

<b>Criticality</b>	minor / informative
<b>Location</b>	contracts/RWACalculator.sol#L163
<b>Status</b>	Acknowledged

### Description

Functions that are not used in the contract, and make the code's size bigger.

```
_isGoldTnft
```

### Recommendation

Remove unused functions.

### Team's Reply 14 December 2022

The team has acknowledged that this is not a security issue and states:

*"Code size only matters on deployment, and we use Polygon where gas prices are very cheap."*

## L11 - Unnecessary Boolean equality

<b>Criticality</b>	minor / informative
<b>Location</b>	contracts/USDRTreasury.sol#L86
<b>Status</b>	Acknowledged

### Description

The comparison to boolean constants is redundant. Boolean constants can be used directly and do not need to be compared to true or false.

```
success == false
```

### Recommendation

Remove the equality to the boolean constant.

### Team's Reply 14 December 2022

The team has acknowledged that this is not a security issue and states:

*"We chose this because of readability."*

## L12 - Using Variables before Declaration

<b>Criticality</b>	minor / informative
<b>Location</b>	contract/USDRTreasury.sol#L131  contract/TNGBLPriceOracle.sol#L47
<b>Status</b>	Acknowledged

### Description

The contract is using a variable before the declaration. This is usually happening either if it has not been declared yet or the variable has been declared in a different scope.

```
success  
amount
```

### Recommendation

The variables should be declared before any usage of them.

### Team's Reply 14 December 2022

The team has acknowledged that this is not a security issue and states:

*"We are just exploiting language abilities to save on gas."*

## L13 - Divide before Multiply Operation

<b>Criticality</b>	minor / informative
<b>Location</b>	contract/IncentiveVault.sol#L55  contract/RWACalculator.sol#L352
<b>Status</b>	Acknowledged

### Description

Performing divisions before multiplications may cause loss of prediction.

```
amount = (IExchange(exchange).scaleToUnderlying(IERC20(USDR).totalSupply()) * apr) /  
3_650_000  
topPrice = (((hs.weSellAt[0] + hs.lockedAmount[0]) * share) / 10000000) *  
priceThreshold[address(tnft)] / fullPercent
```

### Recommendation

The multiplications should be prior to the divisions.

### Team's Reply 14 December 2022

The team has acknowledged that this is not a security issue and states:

*"First example doesn't have precision loss, and the second example is used only by our scripts and contracts where we make decisions on fraction purchases, where small precision losses are not significant in our use-case."*

## L14 - Uninitialized Variables in Local Scope

<b>Criticality</b>	minor / informative
<b>Location</b>	contract/TreasuryTracker.sol#L155,92,509,248  contract/TokenSwap.sol#L82  contract/USDRTreasury.sol#L131,221,375,494  contract/TNGBLPriceOracle.sol#L47  contract/RWACalculator.sol#L331,451,285,372,276,322,308,326,421,280,262,164
<b>Status</b>	Acknowledged

### Description

There are variables that are defined in the local scope and are not initialized.

```
i  
fData  
amounts  
success  
ah  
amount  
j  
hs_scope_0  
hs  
...
```

### Recommendation

All the local scoped variables should be initialized.

### Team's Reply 14 December 2022

The team has acknowledged that this is not a security issue and states:

*“Those are mostly helper structs that we need because of issues with stack having too many local variables. We rely on Solidity here, that it initializes variables dependent on variable types <https://docs.soliditylang.org/en/v0.8.17/types.html>.”*



## L15 - Local Scope Variable Shadowing

<b>Criticality</b>	minor / informative
<b>Location</b>	contract/tangibleInterfaces/ITangibleInterfaces.sol#L34  contract/interfaces/ILiquidityManager.sol#L15
<b>Status</b>	Acknowledged

### Description

There are variables that are defined in the local scope containing the same name from an upper scope.

```
fullShare  
liquidity
```

### Recommendation

The local variables should have different names from the upper scoped variables.

### Team's Reply 14 December 2022

The team has acknowledged that this is not a security issue and states:

*"That has no effect on interface declarations"*

# Contract Functions

Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
<b>AddressAccesssor</b>	Implementation	AccessControl		
	setAddressProvider	Public	✓	onlyRole
<b>AddressAccesssorUpgradable</b>	Implementation	AccessControlUpgradable		
	setAddressProvider	Public	✓	onlyRole
<b>AddressProvider</b>	Implementation	OwnableUpgradable		
	initialize	Public	✓	initializer
	setAddress	External	✓	onlyOwner
	getAddresses	External		-
<b>IBatchSender</b>	Interface			
	send	External	✓	-
<b>IExchangeProxy</b>	Interface			

	swapFromToken	External	✓	-
<b>AffiliateExchange</b>	Implementation	Pausable, AddressAcc essor		
		Public	✓	-
	mint	External	✓	whenNotPaus ed
	pause	External	✓	whenNotPaus ed onlyRole
	unpause	External	✓	whenPaused onlyRole
	sendRewards	External	✓	whenNotPaus ed onlyRole
	withdrawExcessUSDR	External	✓	onlyRole
	withdrawToken	External	✓	whenPaused onlyRole
	_mint	Private	✓	
<b>DAIBond</b>	Implementation	AddressAcc essor, Pausable		
		Public	✓	-
	claimAll	External	✓	-
	deposit	External	✓	whenNotPaus ed
	earned	External		-

	pause	External	✓	onlyRole whenNotPaused
	recoverLostTokens	External	✓	onlyRole
	unpause	External	✓	onlyRole whenPaused
	claimable	Public		-
	claim	Public	✓	-
	_earned	Private		
<b>CurveExchanges</b>	Interface			
	underlying_coins	External	✓	-
	get_exchange_amount	External		-
	get_input_amount	External		-
	exchange	External	✓	-
<b>CurveRegistry</b>	Interface			
	find_pool_for_coins	External		-
	get_coin_indices	External		-
<b>CurveWrapper</b>	Implementation	AccessControl		
		Public	✓	-

	addPoolForTokens	External	✓	onlyRole
	getAmountsIn	External		-
	getAmountsOut	External		-
	swapExactTokensForTokens	External	✓	-
	swapTokensForExactTokens	External	✓	-
<b>LiquidityToken Math</b>	Implementation			
	getTokenAmounts	External		-
<b>IncentiveVault</b>	Implementation	AddressAcc essor, Pausable		
		Public	✓	-
	pause	External	✓	onlyRole
	recoverLostTokens	External	✓	onlyRole
	setAPR	External	✓	onlyRole
	unpause	External	✓	onlyRole
	withdraw	External	✓	onlyRole
	availableAmount	Public		-
<b>IExchange</b>	Interface			
	scaleFromUnderlying	External		-

	scaleToUnderlying	External		-
	swapFromUnderlying	External	✓	-
	updateMintingStats	External	✓	-
<b>ILiquidityManager</b>	Interface			
	getTokenAmounts	External		-
	liquidity	External		-
	increaseLiquidity	External	✓	-
<b>ILiquidityTokenMath</b>	Interface			
	getTokenAmounts	External		-
<b>IPriceOracle</b>	Interface			
	quote	External		-
<b>IRateProvider</b>	Interface			
	getRate	External		-
<b>IRWACalculator</b>	Interface			
	calculate	External		-

	fetchPaymentTokenAndAmountFtnft	External		-
	fetchPaymentTokenAndAmountTnft	External		-
	calcFractionNativeValue	External		-
	calcTnftNativeValue	External		-
<b>ITokenSwap</b>	Interface			
	quoteOut	External		-
	quoteIn	External		-
	exchange	External	✓	-
<b>ITreasury</b>	Interface			
	purchaseStableMintedRedeemedThreshold	External		-
	purchaseStableMarketcapThreshold	External		-
	multicall	External	✓	-
	withdraw	External	✓	-
	getTreasuryValue	External		-
	updateTrackerFtnftExt	External	✓	-
	updateTrackerTnftExt	External	✓	-
	purchaseReInitialSale	External	✓	-

<b>ITreasuryTracker</b>	Interface			
	tnftTreasuryPlaced	External	✓	-
	ftnftTreasuryPlaced	External	✓	-
	updateFractionData	External	✓	-
	getFractionTokensDataInTreasury	External		-
	getRwaUsdValue	External		-
	addValueAfterPurchase	External	✓	-
	subValueAfterPurchase	External	✓	-
<b>IUSDR</b>	Interface	IERC20Upgradable		
	burn	External	✓	-
	mint	External	✓	-
	rebase	External	✓	-
<b>GoldPurchaseManager</b>	Implementation	PurchaseManager, IERC721Receiver		
		Public	✓	-
	setGoldTnfts	External	✓	onlyRole
	purchaseTnft	External	✓	onlyRole
	purchaseTnftCb	External	✓	-



	purchaseFtnft	External	✓	onlyRole
	purchaseFtnftCb	External	✓	-
	_fetchGoldAddressFromFraction	Internal		
	onERC721Received	External	✓	-
	_onERC721Received	Private	✓	
<b>GoldSellManager</b>	Implementation	OnSaleTracker, IERC721Receiver		
		Public	✓	-
	setGoldTnfts	External	✓	onlyRole
	sellTnft	External	✓	onlyRole
	sellTnftCb	External	✓	-
	modifyTnftSale	External	✓	onlyRole
	sellFtnft	External	✓	onlyRole
	sellFtnftCb	External	✓	-
	modifyFtnftSale	External	✓	onlyRole
	sellFtnftInitial	External	✓	onlyRole
	sellFtnftInitialCb	External	✓	-
	_fetchGoldAddressFromFraction	Internal		
	stopSellFtnft	External	✓	onlyRole
	stopSellTnft	External	✓	onlyRole

	withdrawToken	External	✓	-
	onERC721Received	External	✓	-
	_onERC721Received	Private	✓	
<b>ICurveFactory</b>	Interface			
	deploy_metapool	External	✓	-
	get_base_pool	External		-
	get_meta_n_coins	External		-
	get_underlying_balances	External		-
	get_underlying_decimals	External		-
<b>ICurvePool</b>	Interface			
	underlying_coins	External		-
<b>ICurveZapper</b>	Interface			
	add_liquidity	External	✓	-
	exchange_underlying	External	✓	-
<b>LiquidityManager</b>	Implementation	AddressAcc essor, Pausable		
		Public	✓	-

	increaseLiquidity	External	✓	whenNotPaused
	initializePool	External	✓	whenPaused onlyRole
	liquidity	External		-
	missingLiquidity	External		-
	setMinSize	External	✓	onlyRole
	sweepToken	Public	✓	onlyRole
	sweepTokens	External	✓	onlyRole
	withdrawLPToken	External	✓	onlyRole
	_getCoinIndex	Private		
	_getPoolBalances	Private		
	_rebalancePool	Private	✓	
<b>OnSaleTracker</b>	Implementation	AddressAcc essor		
	getFractionContractsOnSale	External		-
	fractionContractsOnSaleSize	External		-
	getFractionTokensOnSale	External		-
	getFractionTokensOnSaleBatch	External		-
	fractionTokensOnSaleSize	External		-
	getTnftCategoriesOnSale	External		-
	tnftCategoriesOnSaleSize	External		-

	getTnftTokensOnSale	External		-
	getTnftTokensOnSaleBatch	External		-
	tnftTokensOnSaleSize	External		-
	tnftSalePlacedExt	External	✓	onlyRole
	tnftSalePlaced	Internal	✓	
	ftnftSalePlacedExt	External	✓	onlyRole
	ftnftSalePlaced	Internal	✓	
	_removeCurrentlySellingFraction	Internal	✓	
	_removeCurrentlySellingTnft	Internal	✓	
	_removeCategory	Internal	✓	
	_removeFraction	Internal	✓	
<b>IUSDRExchange</b>	Interface			
	mintingStats	External		-
<b>IUSDR</b>	Interface	IERC20Upgradable		
	totalSupply	External		-
<b>PurchaseManager</b>	Implementation	AddressAccessor		
	_validatePurchase	Internal		

	_convertTreasuryTokenToPayment	Internal	✓	
	_convertToCorrectDecimals	Internal		
	_checkPaymentTokenAndAmountNeeded	Internal		
<b>RePurchaseManager</b>	Implementation	PurchaseManager, IERC721Receiver		
		Public	✓	-
	setRETnft	External	✓	onlyRole
	purchaseTnft	External	✓	onlyRole
	purchaseTnftCb	External	✓	-
	purchaseFnft	External	✓	onlyRole
	purchaseFnftCb	External	✓	-
	_fetchReAddressFromFraction	Internal		
	onERC721Received	External	✓	-
	_onERC721Received	Private	✓	
<b>ReSellManager</b>	Implementation	OnSaleTracker, IERC721Receiver		
		Public	✓	-
	setRETnft	External	✓	onlyRole

	sellTnft	External	✓	onlyRole
	sellTnftCb	External	✓	-
	modifyTnftSale	External	✓	onlyRole
	sellFtnft	External	✓	onlyRole
	sellFtnftCb	External	✓	-
	modifyFtnftSale	External	✓	onlyRole
	sellFtnftInitial	External	✓	onlyRole
	sellFtnftInitialCb	External	✓	-
	stopSellFtnft	External	✓	onlyRole
	stopSellTnft	External	✓	onlyRole
	_fetchReAddressFromFraction	Internal		
	withdrawToken	External	✓	-
	onERC721Received	External	✓	-
	_onERC721Received	Private	✓	
<b>TNGBLLiquidit yManager</b>	Implementation	AddressAcc essor, Pausable		
		Public	✓	-
	depositLPToken	External	✓	whenPaused onlyRole
	getTokenAmounts	External		-

	increaseLiquidity	External	✓	whenNotPaused
	initializePool	External	✓	whenPaused onlyRole
	missingLiquidity	External		-
	setMinSize	External	✓	onlyRole
	sweepTokens	External	✓	onlyRole
	withdrawLPToken	External	✓	whenNotPaused onlyRole
	_addLiquidity	Private	✓	
	_collectFees	Private	✓	
	_createPool	Private	✓	
	_getPoolInfo	Private		
	_getTokenAmounts	Private		
	_toTNGBL	Private		
<b>Owned</b>	Implementation			
		Public	✓	-
	transferOwnership	External	✓	onlyOwner
	acceptOwnership	External	✓	-
<b>AggregatorInterface</b>	Interface			
	latestAnswer	External		-

	latestTimestamp	External		-
	latestRound	External		-
	getAnswer	External		-
	getTimestamp	External		-
<b>AggregatorV3Interface</b>	Interface			
	decimals	External		-
	description	External		-
	version	External		-
	getRoundData	External		-
	latestRoundData	External		-
<b>AggregatorV2V3Interface</b>	Interface	AggregatorInterface, AggregatorV3Interface		
<b>MockDAIUSDOracle</b>	Implementation	AggregatorV2V3Interface, Owned		
	latestAnswer	Public		-
	latestRoundData	Public		-
	decimals	External		-
	version	External		-



	description	External		-
	getAnswer	External		-
	getTimestamp	External		-
	latestRound	External		-
	latestTimestamp	External		-
	getRoundData	External		-
<b>MockERC20</b>	Implementation	ERC20, AccessControl		
		Public	✓	ERC20
	decimals	Public		-
	mint	External	✓	whitelisted
	transfer	Public	✓	whitelisted
	transferFrom	Public	✓	whitelisted
	toggleWhitelist	External	✓	onlyRole
	whitelist	External	✓	onlyRole
<b>MockOracle</b>	Implementation			
	consult	External		-
<b>MockRouter</b>	Implementation			

	getAmountsOut	External		-
<b>MockTreasury</b>	Implementation	ITreasury		
	withdraw	External	✓	-
	setStableValue	External	✓	-
	setUSDRValue	External	✓	-
	setRWAValue	External	✓	-
	setTNGBLValue	External	✓	-
	setLiquidityValue	External	✓	-
	setTNGBLLiquidityValue	External	✓	-
	setDebtValue	External	✓	-
	getTreasuryValue	External		-
	multicall	External	✓	-
	updateTrackerFtnftExt	External	✓	-
	updateTrackerTnftExt	External	✓	-
	purchaseReInitialSale	External	✓	-
	purchaseStableMintedRedeemedThreshold	External		-
	purchaseStableMarketcapThreshold	External		-
	_recomputeTotal	Private	✓	

<b>ITreasuryTrackerExt</b>	Interface	ITreasuryTracker		
	getFractionContractsInTreasury	External		-
	fractionContractsInTreasurySize	External		-
	getFractionTokensInTreasury	External		-
	fractionTokensInTreasurySize	External		-
	getFractionContractsOnSale	External		-
	fractionContractsOnSaleSize	External		-
	getFractionTokensOnSale	External		-
	fractionTokensOnSaleSize	External		-
	getTnftCategoriesInTreasury	External		-
	tnftCategoriesInTreasurySize	External		-
	getTnftTokensInTreasury	External		-
	tnftTokensInTreasurySize	External		-
	getTnftCategoriesOnSale	External		-
	tnftCategoriesOnSaleSize	External		-
	getTnftTokensOnSale	External		-
	tnftTokensOnSaleSize	External		-
<b>ITNFTPriceManager</b>	Interface			
	itemPriceBatchTokenIds	External		-

	itemPriceBatchFingerprints	External		-
	getPriceOracleForCategory	External		-
<b>RWACalculator</b>	Implementation	AddressAcc essor, IRWACalcul ator		
		Public	✓	-
	setGoldTnfts	External	✓	onlyRole
	setPriceAboveMarketThreshold	External	✓	onlyRole
	_isGoldTnft	Internal		
	_getPriceManager	Internal		
	_getTangibleMarketplace	Internal		
	_convertToCorrectDecimals	Internal		
	calculate	External		-
	_calculateTnftsOnSale	Internal		
	_calculateFtnftsOnSale	Internal		
	fetchPaymentTokenAndAmountFtnft	External		-
	fetchPaymentTokenAndAmountTnft	External		-
	calcFractionNativeValue	External		-
	calcTnftNativeValue	External		-
	_getTnftNativeValue	Internal		
	_checkStorageValue	Internal		

<b>IStrategy</b>	Interface			
	treasury	External		-
	want	External		-
	deposit	External	✓	-
	withdraw	External	✓	-
	balanceOf	External		-
	balanceOfWant	External		-
	balanceOfPool	External		-
	harvest	External	✓	-
	retire	External	✓	-
	panic	External	✓	-
	router	External		-
<b>IMiniChefV2</b>	Interface			
	poolLength	External		-
	updatePool	External	✓	-
	userInfo	External		-
	deposit	External	✓	-
	withdraw	External	✓	-
	harvest	External	✓	-

	withdrawAndHarvest	External	✓	-
	emergencyWithdraw	External	✓	-
<b>SushiLPStrategy</b>	Implementation	AccessControl, Pausable, IStrategy		
		Public	✓	-
	deposit	Public	✓	whenNotPaused
	withdraw	External	✓	onlyTreasury
	harvest	External	✓	whenNotPaused onlyRole
	addLiquidity	Internal	✓	
	balanceOf	Public		-
	balanceOfWant	Public		-
	balanceOfPool	Public		-
	retire	External	✓	onlyTreasury
	panic	Public	✓	onlyRole
	pause	Public	✓	onlyRole
	unpause	External	✓	onlyRole
	_ensureAllowance	Internal	✓	
	_removeAllowances	Internal	✓	

<b>ICurrencyFeed</b>	Interface			
	currencyPriceFeeds	External		-
	conversionPremiums	External		-
<b>IInstantLiquidity</b>	Interface			
	sellInstant	External	✓	-
	buyInstant	External	✓	-
	sellInstantFraction	External	✓	-
	buyFractionInstant	External	✓	-
	withdrawUSDC	External	✓	-
	withdrawTNGBL	External	✓	-
<b>IPriceOracle</b>	Interface			
	latestPrices	External		-
	decimals	External		-
	marketPriceNativeCurrency	External		-
<b>ITangibleNFT</b>	Interface			
	storagePricePerYear	External		-
	storagePercentagePricePerYear	External		-

	storagePriceFixed	External		-
	storageRequired	External		-
	tnftToPassiveNft	External		-
	claim	External	✓	-
	tokensFingerprint	External		-
<b>ITangibleFracti onsNFT</b>	Interface			
	defractionalize	External	✓	-
	tnft	External		-
	tnftTokenId	External		-
	tnftFingerprint	External		-
	fractionShares	External		-
	fullShare	External		-
	claim	External	✓	-
	claimableIncome	External		-
<b>IFractionStora geManager</b>	Interface			
	payShareStorage	External	✓	-
<b>IFactoryExt</b>	Interface			



	storageManagers	External		-
	defUSD	External		-
	paymentTokens	External		-
	fractionToTnftAndId	External		-
	initReSeller	External		-
<b>ITangibleMarketplace</b>	Interface			
	marketplace	External		-
	marketplaceFract	External		-
	factory	External		-
	sellBatch	External	✓	-
	stopBatchSale	External	✓	-
	buy	External	✓	-
	buyUnminted	External	✓	-
	buyFraction	External	✓	-
	sellFraction	External	✓	-
	payStorage	External	✓	-
	sellFractionInitial	External	✓	-
	stopFractSale	External	✓	-

<b>ITangiblePiNF T</b>	Interface			
	claim	External	✓	-
	claimableIncome	External		-
<b>ITangibleRent Share</b>	Interface			
	forToken	External	✓	-
<b>ITangibleReve nueShare</b>	Interface			
	claimForToken	External	✓	-
	revenueToken	External		-
<b>TNGBLLocked Value</b>	Implementation	AddressAcc essor		
		Public	✓	-
	getTngblLockedValue	External		-
<b>IBaseOracle</b>	Interface			
	consult	External		-
<b>TNGBLPriceOr acle</b>	Implementation	AddressAcc essor, IPriceOracle		

		Public	✓	-
	quote	External		-
	setOracleLookBackPeriod	External	✓	onlyRole
	setSwapRoute	Public	✓	onlyRole
<b>IMintableERC20</b>	Interface			
	mint	External	✓	-
<b>ITangibleERC20</b>	Interface			
	approve	External	✓	-
	burn	External	✓	-
<b>pDAI</b>	Implementation	AddressAcc essor, ERC20Perm it, Pausable		
		Public	✓	ERC20 ERC20Permit
	mint	External	✓	onlyRole whenNotPaus ed
	pause	External	✓	onlyRole
	redeem	External	✓	-
	redeemFor	Public	✓	whenNotPaus ed

	unpause	External	✓	onlyRole
<b>USDR</b>	Implementation	IUSDR, AddressAcc essorUpgra dable, ERC20Perm itUpgradeab le, PausableUp gradeable		
	initialize	Public	✓	initializer
	burn	External	✓	whenNotPaus ed
	mint	External	✓	onlyRole whenNotPaus ed
	pause	External	✓	onlyRole whenNotPaus ed
	rebase	External	✓	onlyRole whenNotPaus ed
	unpause	External	✓	onlyRole whenPaused
	allowance	Public		-
	approve	Public	✓	-
	balanceOf	Public		-
	decimals	Public		-
	decreaseAllowance	Public	✓	-
	increaseAllowance	Public	✓	-

	totalSupply	Public		-
	transfer	Public	✓	whenNotPaused
	transferAll	Public	✓	whenNotPaused
	transferAllFrom	Public	✓	whenNotPaused
	transferFrom	Public	✓	whenNotPaused
	_approve	Internal	✓	
<b>IStaking</b>	Interface			
	unstake	External	✓	-
	usdrMarketCap	External		-
<b>LegacySUSDR</b>	Interface			
	transferAll	External	✓	-
	transferAllFrom	External	✓	-
<b>USDRMigration</b>	Implementation	AddressAccessor		
		Public	✓	-
	initialize	External	✓	onlyRole
	migrate	External	✓	-

<b>WadRayMath</b>	Library			
	rayDiv	Internal		
	rayMul	Internal		
	rayToWad	Internal		
	wadToRay	Internal		
<b>WrappedUSD R</b>	Implementation	AccessCont rolUpgradea ble, ERC20Perm itUpgradeab le, IERC4626U pgradeable		
	initialize	Public	✓	initializer
	decimals	Public		-
	totalAssets	External		-
	convertToShares	External		-
	convertToAssets	External		-
	maxDeposit	External		-
	previewDeposit	External		-
	deposit	External	✓	-
	maxMint	External		-
	previewMint	External		-

	mint	External	✓	-
	maxWithdraw	External		-
	previewWithdraw	External		-
	withdraw	External	✓	-
	maxRedeem	External		-
	previewRedeem	External		-
	redeem	External	✓	-
	_getRate	Private		
	_convertToSharesUp	Private		
	_convertToAssetsUp	Private		
	_convertToSharesDown	Private		
	_convertToAssetsDown	Private		
	_pullAssets	Private	✓	
	_pushAssets	Private	✓	
<b>TokenSwap</b>	Implementation	ITokenSwap , AccessCont rol		
		Public	✓	-
	addSwapRoute	External	✓	onlyRole
	removeSwapRoute	External	✓	onlyRole
	exchange	External	✓	-

	quoteOut	External		-
	quoteIn	External		-
<b>TreasuryTracker</b>	Implementation	AddressAccessor, ITreasuryTracker		
	getFractionContractsInTreasury	External		-
	getTnftFractionContractsInTreasury	External		-
	fractionContractsInTreasurySize	External		-
	tnftFractionContractsInTreasurySize	External		-
	getFractionTokensInTreasury	External		-
	getFractionTokensInTreasuryBatch	External		-
	fractionTokensInTreasurySize	External		-
	getFractionTokensDataInTreasury	External		-
	getTnftCategoriesInTreasury	External		-
	tnftCategoriesInTreasurySize	External		-
	getTnftTokensInTreasury	External		-
	getTnftTokensInTreasuryBatch	External		-
	tnftTokensInTreasurySize	External		-
		Public	✓	-
	tnftTreasuryPlaced	External	✓	-
	ftnftTreasuryPlaced	External	✓	-



	_fetchTnftForFraction	Internal		
	_removeCurrentlySellingFraction	Internal	✓	
	updateFractionData	External	✓	-
	_removeCurrentlySellingTnft	Internal	✓	
	_removeCategory	Internal	✓	
	_removeFraction	Internal	✓	
	_removeFractionFromTnftMap	Internal	✓	
	setCurrencyData	External	✓	onlyRole
	updateTotalNativeValue	External	✓	onlyRole
	addValueAfterPurchase	External	✓	-
	_convertToCorrectDecimals	Internal		
	subValueAfterPurchase	External	✓	-
	removeCurrency	External	✓	onlyRole
	currencySize	External		-
	getRwaUsdValue	External		-
	convertPriceToUSDCustom	Internal		
<b>IEExchange</b>	Interface			
	scaleFromUnderlying	External		-
	swapFromUnderlying	External	✓	-
	swapFromTNGBL	External	✓	-

<b>BondingVault</b>	Implementation			
		Public	✓	-
	withdraw	External	✓	-
	sweep	External	✓	-
<b>USDRBonding</b>	Implementation	AddressAcc essor		
		Public	✓	-
	sweepVault	External	✓	onlyRole
	mint	External	✓	onlyRole
	recoverLostTokens	External	✓	onlyRole
	reset	External	✓	onlyRole
	setPercentage	External	✓	onlyRole
	withdraw	External	✓	onlyRole
<b>USDRExchang e</b>	Implementation	AddressAcc essor, IExchange, Pausable		
		Public	✓	-
	pause	External	✓	onlyRole
	scaleFromUnderlying	External		-

	updateMintingStats	External	✓	-
	maxTNGBLMintingAmount	External		-
	mintAgainstGains	External	✓	onlyRole
	scaleToUnderlying	External		-
	setFees	External	✓	onlyRole
	swapToPromissory	External	✓	whenNotPaused
	swapToTNGBL	External	✓	whenNotPaused
	swapToUnderlying	External	✓	whenNotPaused
	swapFromTNGBL	External	✓	whenNotPaused
	swapFromUnderlying	External	✓	whenNotPaused
	unpause	External	✓	onlyRole
	setAddressProvider	Public	✓	onlyRole
	updateUnderlying	Public	✓	onlyRole
	_applyFee	Private		
	_computeScale	Private		
	_maxTNGBLMintingAmount	Private		
	_prepareTNGBLWithdrawal	Private	✓	
	_preparePromissoryWithdrawal	Private	✓	
	_prepareUnderlyingWithdrawal	Private	✓	

	_scaleAmount	Private		
	_scaleFromUnderlying	Private		
	_scaleToUnderlying	Private		
<b>USDRExchangeProxy</b>	Implementation	AddressAcc essor, Pausable		
		Public	✓	-
	pause	External	✓	onlyRole
	swapFromToken	External	✓	whenNotPaus ed
	unpause	External	✓	onlyRole
<b>ITreasuryTrackerExt</b>	Interface	ITreasuryTra cker		
	getFractionContractsInTreasury	External		-
	getFractionTokensInTreasury	External		-
	getTnftCategoriesInTreasury	External		-
	getTnftTokensInTreasury	External		-
<b>USDRTreasury</b>	Implementation	AddressAcc essor, ITreasury, IERC721Re ceiver		
		Public	✓	-

	setThresholds	External	✓	onlyRole
	multicall	External	✓	onlyRole
	triggerRebase	External	✓	onlyRole
	purchaseReInitialSale	External	✓	-
	_swapToTreasuryToken	Internal	✓	
	getTreasuryValue	Public		-
	withdraw	External	✓	validToken
	defractionalize	External	✓	onlyRole
	updateTrackerFtnftExt	External	✓	onlyRole
	updateTrackerFtnft	Internal	✓	
	updateTrackerTnftExt	External	✓	onlyRole
	updateTrackerTnft	Internal	✓	
	toggleStop	External	✓	onlyRole
	withdrawToken	External	✓	onlyRole
	withdrawDepositNFT	External	✓	onlyRole
	_withdrawDepositNFT	Internal	✓	
	claimRentForToken	External	✓	onlyRole
	claimTngblRevenue	External	✓	-
	payFractionStorage	External	✓	onlyRole
	onERC721Received	External	✓	-
	_fetchAddress	Internal		

	_verifyBacking	Internal		
<b>WrappedUSD RRateProvider</b>	Implementation	IRateProvid er		
		Public	✓	-
	getRate	External		-

# Contract Flow



## Summary

The Tangible USDR Ecosystem implements a stable coin mechanism. This audit focused on investigating possible security issues and potential improvements. It investigates the diversion from a classic DAO implementation and comments about the architectural decisions.



## 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>