



ERD – Ethereum Reserve Dollar

Smart Contract Security
Assessment

Prepared by: Halborn

Date of Engagement: June 7th, 2023 – July 12th, 2023

Visit: Halborn.com

DOCUMENT REVISION HISTORY	4
CONTACTS	5
1 EXECUTIVE OVERVIEW	6
1.1 INTRODUCTION	7
1.2 ASSESSMENT SUMMARY	7
1.3 SCOPE	8
1.4 TEST APPROACH & METHODOLOGY	10
2 RISK METHODOLOGY	11
2.1 EXPLOITABILITY	12
2.2 IMPACT	13
2.3 SEVERITY COEFFICIENT	15
3 ASSESSMENT SUMMARY & FINDINGS OVERVIEW	17
4 FINDINGS & TECH DETAILS	18
4.1 (HAL-01) PRICE MANIPULATION RISK IN STETHORACLE CONTRACT - MEDIUM(5.0)	20
Description	20
Code Location	20
Proof of Concept	21
BVSS	21
Recommendation	21
Remediation Plan	21
4.2 (HAL-02) NON-TRANSFERABLE OWNER IN MIGRATIONS CONTRACT - LOW(2.5)	22
Description	22
Code Location	22

	BVSS	22
	Recommendation	22
	Remediation Plan	23
4.3	(HAL-03) POSSIBLE DOS DUE TO COLLATERALMAN- AGER.COLLATERALSUPPORT SIZE - INFORMATIONAL(1.0)	24
	Description	24
	Code Location	24
	BVSS	25
	Recommendation	25
	Remediation Plan	25
4.4	(HAL-04) MISSING A CAP FOR EUSD GAS COMPENSATION - INFORMATI- ONAL(0.0)	26
	Description	26
	Code Location	26
	BVSS	26
	Recommendation	26
	Remediation Plan	26
4.5	(HAL-05) LONG LITERAL UINT256 USED IN COLLATERALMANAGER - IN- FORMATIONAL(0.0)	27
	Description	27
	Code Location	27
	BVSS	27
	Recommendation	28
	Remediation Plan	28
4.6	(HAL-06) MISSING REQUIREISCONTRACT CHECK IN COLLATERALMAN- AGER.SETADDRESSES() - INFORMATIONAL(0.0)	29
	Description	29

	Code Location	29
	BVSS	30
	Recommendation	30
	Remediation Plan	30
5	MANUAL TESTING	31
6	AUTOMATED TESTING	41
6.1	STATIC ANALYSIS REPORT	42
	Description	42
	Slither results	42
6.2	AUTOMATED SECURITY SCAN	45
	Description	45
	MythX results	45

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	07/07/2023	Omar Alshaeb
0.2	Document Updates	07/07/2023	Omar Alshaeb
0.3	Document Updates	07/12/2023	Omar Alshaeb
0.4	Draft Review	07/13/2023	Gokberk Gulgun
0.5	Draft Review	07/13/2023	Gabi Urrutia
0.6	Document Updates	07/17/2023	Omar Alshaeb
1.0	Remediation Plan	07/17/2023	Omar Alshaeb
1.1	Remediation Plan Review	07/17/2023	Gabi Urrutia
1.2	Document Updates	07/18/2023	Omar Alshaeb
1.3	Document Updates Review	07/18/2023	Gabi Urrutia

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

ERD is a decentralized protocol that allows Ether/LSD (Liquid Staking Derivatives) holders to obtain maximum liquidity against their collateral while paying low interest. After locking up ETH/wrapper ETH as collateral in a smart contract and creating an individual position called a **trove**, the user can get instant liquidity by minting EUSD, a USD-pegged stablecoin. Each trove is required to be collateralized at a minimum of 110%. Any owner of EUSD can redeem their stablecoins for the underlying collateral at any time. The redemption mechanism along with algorithmically adjusted fees guarantee a minimum stablecoin value of USD 1.

ERD engaged Halborn to conduct a security assessment on their **smart contracts** beginning on June 7th, 2023 and ending on July 12th, 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 successfully addressed by the ERD team.

1.3 SCOPE

1. IN-SCOPE:

The security assessment was scoped to the following [smart contracts](#):

- [contracts/Dependencies/*](#)
- [contracts/Interfaces/*](#)
- [contracts/Oracles/*](#)
- [contracts/ActivePool.sol](#)
- [contracts/BorrowerOperations.sol](#)
- [contracts/CollSurplusPool.sol](#)
- [contracts/CollateralManager.sol](#)
- [contracts/CommunityIssuance.sol](#)
- [contracts/DataTypes.sol](#)
- [contracts/DefaultPool.sol](#)
- [contracts/EToken.sol](#)
- [contracts/EUSDToken.sol](#)
- [contracts/Errors.sol](#)
- [contracts/GasPool.sol](#)
- [contracts/HintHelpers.sol](#)
- [contracts/LiquidityIncentive.sol](#)
- [contracts/Migrations.sol](#)
- [contracts/MultiTroveGetter.sol](#)
- [contracts/PriceFeed.sol](#)
- [contracts/SortedTrove.sol](#)
- [contracts/StabilityPool.sol](#)
- [contracts/Treasury.sol](#)
- [contracts/TroveDebt.sol](#)
- [contracts/TroveInterestRateStrategy.sol](#)
- [contracts/TroveLogic.sol](#)
- [contracts/TroveManager.sol](#)
- [contracts/TroveManagerDataTypes.sol](#)
- [contracts/TroveManagerLiquidations.sol](#)
- [contracts/TroveManagerRedemptions.sol](#)

Commit ID: [c46e664f30a3ed28a0420afd11788b045527a39a](#)

2. REMEDIATION PR/COMMITTS:

- Fix Commit ID (HAL-01): 6657817edcc30b48e41836756a3f41fa34ef779d
- Fix Commit ID (HAL-02): 95ad8f438291ec082f34dab97dc57ecf2494209c
- Fix Commit ID (HAL-03): 93c803ae22a7676e05a1fa6ec884589de28fd619
- Fix Commit ID (HAL-04): 1961c7fc04181f770468d575b5b402a07b8ab239
- Fix Commit ID (HAL-05): 0aaf1539e5897aca96034f20f82a0ec1a8d45182
- Fix Commit ID (HAL-06): f77899108075aef9f90e1e31ab4e1ab22c20d89c

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
0	0	1	1	4

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
(HAL-01) PRICE MANIPULATION RISK IN STETHORACLE CONTRACT	Medium (5.0)	SOLVED - 07/17/2023
(HAL-02) NON-TRANSFERABLE OWNER IN MIGRATIONS CONTRACT	Low (2.5)	SOLVED - 07/17/2023
(HAL-03) POSSIBLE DOS DUE TO COLLATERALMANAGER.COLLATERALSUPPORT SIZE	Informational (1.0)	SOLVED - 07/17/2023
(HAL-04) MISSING A CAP FOR EUSD GAS COMPENSATION	Informational (0.0)	SOLVED - 07/17/2023
(HAL-05) LONG LITERAL UINT256 USED IN COLLATERALMANAGER	Informational (0.0)	SOLVED - 07/17/2023
(HAL-06) MISSING REQUIREISCONTRACT CHECK IN COLLATERALMANAGER.SETADDRESSES()	Informational (0.0)	SOLVED - 07/18/2023



FINDINGS & TECH DETAILS



4.1 (HAL-01) PRICE MANIPULATION RISK IN STETHORACLE CONTRACT – MEDIUM (5.0)

Description:

If the owner's private key of the contract `StETHOracle.sol` gets stolen, or the owner himself acts maliciously, it is possible to directly manipulate the price oracle by calling the `setPrice()` function and updating the `lastGoodPrice` storage variable without using Chainlink. Hence, all parts of the protocol using `fetchPrice_view()` would get as a result an incorrect price for the token.

Code Location:

Listing 1: StETHOracle.sol (Line 339)

```
338 function setPrice(uint _price) external onlyOwner {  
339     lastGoodPrice = _price;  
340     emit LastGoodPriceUpdated(_price);  
341 }
```

Listing 2: StETHOracle.sol (Line 139)

```
138 function fetchPrice_view() external view override returns (uint256  
    ↳ ) {  
139     return lastGoodPrice;  
140 }
```

Proof of Concept:

1. The owner calls `setPrice()` and significantly decreases the token price.
2. All active troves now have $ICR < MCR$, hence can be liquidated.
3. Liquidate all troves and distribute all the rewards.
4. Set the correct token price again.

BVSS:

A0:A/AC:L/AX:L/C:N/I:H/A:N/D:M/Y:M/R:P/S:U (5.0)

Recommendation:

Updating the `lastGoodPrice` storage variable is only recommended by requesting it to Chainlink and checking if the response is acceptable.

Remediation Plan:

SOLVED: The `ERD team` solved the issue with the following commit ID.

Commit ID : `6657817edcc30b48e41836756a3f41fa34ef779d`

4.2 (HAL-02) NON-TRANSFERABLE OWNER IN MIGRATIONS CONTRACT - LOW (2.5)

Description:

The owner of the `Migrations.sol` contract is set in the `constructor()` and cannot be changed anymore. If there is any issue with the owner account, the contract can be left useless without being able to change the ownership to a new address.

Code Location:

Listing 3: `Migrations.sol` (Line 14)

```
13 constructor() {  
14     owner = msg.sender;  
15 }
```

Listing 4: `Migrations.sol` (Line 10)

```
9 modifier restricted() {  
10     if (msg.sender == owner) _;  
11 }
```

BVSS:

A0:A/AC:L/AX:L/C:N/I:N/A:M/D:N/Y:N/R:P/S:U (2.5)

Recommendation:

The use of `Ownable` contract from `OpenZeppelin` is recommended to handle the ownership of the contract.

Remediation Plan:

SOLVED: The **ERD team** solved the issue with the following commit ID.

Commit ID : [95ad8f438291ec082f34dab97dc57ecf2494209c](#)

4.3 (HAL-03) POSSIBLE DOS DUE TO COLLATERALMANAGER.COLLATERALSUPPORT SIZE - INFORMATIONAL (1.0)

Description:

The owner of the `CollateralManager.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 pushed to an array (`collateralSupport`), the size of this array can grow considerably over time.

Hence, when the protocol calls `priceUpdate()` to update the price of all collaterals supported by the protocol, it iterates over all the collaterals fetching their price from their oracles. In the case the size of the array has grown significantly, it could be possible the price update will revert due to reaching the transaction gas limit.

Code Location:

Listing 5: `CollateralManager.sol` (Line 127)

```
111 function addCollateral(
112     address _collateral,
113     address _oracle,
114     address _eTokenAddress,
115     uint256 _ratio
116 ) external override onlyOwner {
117     require(!getIsSupport(_collateral), Errors.CM_COLL_EXISTS);
118     _requireRatioLegal(_ratio);
119
120     collateralParams[_collateral] = DataTypes.CollateralParams(
121         _ratio,
122         _eTokenAddress,
123         _oracle,
124         DataTypes.CollStatus(1),
125         collateralsCount
```

```

126     );
127     collateralSupport.push(_collateral);
128     collateralsCount = collateralsCount.add(1);
129 }

```

Listing 6: CollateralManager.sol (Line 246)

```

242 function priceUpdate() public override {
243     if (collateralsCount < 2) {
244         return;
245     }
246     for (uint256 i = 1; i < collateralsCount; ) {
247         IOracle(collateralParams[collateralSupport[i]].oracle).
        ↳ fetchPrice();
248         unchecked {
249             i++;
250         }
251     }
252 }

```

BVSS:

A0:S/AC:L/AX:L/C:N/I:N/A:C/D:N/Y:N/R:P/S:U (1.0)

Recommendation:

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

Remediation Plan:

SOLVED: The **ERD team** solved the issue with the following commit ID.

Commit ID : [93c803ae22a7676e05a1fa6ec884589de28fd619](#)

4.4 (HAL-04) MISSING A CAP FOR EUSD GAS COMPENSATION – INFORMATIONAL (0.0)

Description:

The owner of the `CollateralManager.sol` contract, when setting the `EUSD_GAS_COMPENSATION` protocol parameter within `setGasCompensation()` function, there are no checks regarding the quantity being set.

Code Location:

Listing 7: `CollateralManager.sol` (Line 756)

```
755 function setGasCompensation(uint256 _gas) external override
    ↳ onlyOwner {
756     EUSD_GAS_COMPENSATION = _gas;
757 }
```

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 `EUSD_GAS_COMPENSATION`, which plays a vital role, it is strongly recommended to set a cap for it.

Remediation Plan:

SOLVED: The `ERD team` solved the issue with the following commit ID.

Commit ID : `1961c7fc04181f770468d575b5b402a07b8ab239`

4.5 (HAL-05) LONG LITERAL UINT256 USED IN COLLATERALMANAGER – INFORMATIONAL (0.0)

Description:

Critical protocol parameters are set within the `initialize()` function of `CollateralManager.sol` contract. Specifically, `MCR` (minimum collateral ratio) and `CCR` (critical collateral ratio) are 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 8: `CollateralManager.sol` (Lines 63,64)

```

60 function initialize() public initializer {
61     __Ownable_init();
62     BOOTSTRAP_PERIOD = 14 days;
63     MCR = 1100000000000000000; // 110%
64     CCR = 1300000000000000000; // 130%
65     EUSD_GAS_COMPENSATION = 200e18;
66     MIN_NET_DEBT = 1800e18;
67     BORROWING_FEE_FLOOR = (DECIMAL_PRECISION / 10000) * 75; //
    ↳ 0.75%
68
69     REDEMPTION_FEE_FLOOR = (DECIMAL_PRECISION / 10000) * 75; //
    ↳ 0.75%
70     RECOVERY_FEE = (DECIMAL_PRECISION / 10000) * 25; // 0.25%
71     MAX_BORROWING_FEE = (DECIMAL_PRECISION / 100) * 5; // 5%
72 }

```

BVSS:

A0: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 (11e17, 13e17, etc.) is recommended instead.

Remediation Plan:

SOLVED: The **ERD team** solved the issue with the following commit ID.

Commit ID : [0aaf1539e5897aca96034f20f82a0ec1a8d45182](#)

4.6 (HAL-06) MISSING REQUIREISCONTRACT CHECK IN COLLATERALMANAGER.SETADDRESSES() – INFORMATIONAL (0.0)

Description:

In the `CollateralManager.sol` contract, when setting the contract addresses using `setAddresses()`, the `_requireIsContract()` check is missing for `_troveManagerRedemptionsAddress`.

Code Location:

Listing 9: `CollateralManager.sol` (Line 98)

```

74 function setAddresses(
75     address _activePoolAddress,
76     address _borrowerOperationsAddress,
77     address _defaultPoolAddress,
78     address _priceFeedAddress,
79     address _troveManagerAddress,
80     address _troveManagerRedemptionsAddress,
81     address _wethAddress
82 ) external override onlyOwner {
83     _requireIsContract(_activePoolAddress);
84     _requireIsContract(_borrowerOperationsAddress);
85     _requireIsContract(_defaultPoolAddress);
86     _requireIsContract(_priceFeedAddress);
87     _requireIsContract(_wethAddress);
88     _requireIsContract(_troveManagerAddress);
89
90     borrowerOperationsAddress = _borrowerOperationsAddress;
91     activePool = IActivePool(_activePoolAddress);
92     defaultPool = IDefaultPool(_defaultPoolAddress);
93     priceFeed = IPriceFeed(_priceFeedAddress);
94     wethAddress = _wethAddress;
95
96     troveManager = ITroveManager(_troveManagerAddress);
97

```

```

98     troveManagerRedemptionsAddress =
99     ↪ _troveManagerRedemptionsAddress;
100
101     emit ActivePoolAddressChanged(_activePoolAddress);
102     emit BorrowerOperationsAddressChanged(
103     ↪ _borrowerOperationsAddress);
104     emit DefaultPoolAddressChanged(_defaultPoolAddress);
105     emit PriceFeedAddressChanged(_priceFeedAddress);
106     emit TroveManagerAddressChanged(_troveManagerAddress);
107     emit TroveManagerRedemptionsAddressChanged(
108     ↪ _troveManagerRedemptionsAddress
109 );
110     emit WETHAddressChanged(_wethAddress);
111 }

```

BVSS:

A0: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 is the case with `_troveManagerRedemptionsAddress`, it is recommended to check it as done for the rest of the parameters.

Remediation Plan:

SOLVED: The `ERD team` solved the issue with the following commit ID.

Commit ID : `f77899108075aef9f90e1e31ab4e1ab22c20d89c`



MANUAL TESTING



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

[illegible]

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

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

```
[488] collateralManager::MIN_DEBT() [staticcall]
- 1600000000000000000000
[489] eUSDToken::balanceOf(borrower: [0x05650444aCe15A01762Bd97EE8fDeb495b3C2436]) [staticcall]
- 2000000000000000000000
[227423] borrower::repayEUSD(2000000000000000000000, borrower: [0x05650444aCe15A01762Bd97EE8fDeb495b3C2436], borrower: [0x05650444aCe15A01762Bd97EE8fDeb495b3C2436])
[247] priceFeedTestnet::fetchPrice()
- 2000000000000000000000
[446] collateralManager::priceOfEther()
- 0
[14287] troveManager::checkRecoveryMode(2000000000000000000000) [staticcall]
[214] troveDebt::totalSupply() [staticcall]
- 1600000000000000000000
[474] troveManager::getTroveNormalizedDebt() [staticcall]
- 1600000000000000000000
[387] defaultPool::getEUSDDebt() [staticcall]
- 0
[685] eUSDToken::balanceOf(gasPool: [0x9A743489af33E59C9CD78203b311594F80ea36A9]) [staticcall]
- 2000000000000000000000
[5765] collateralManager::getEntireCollValue(2000000000000000000000) [staticcall]
[498] eToken::totalSupply() [staticcall]
- 1600000000000000000000
[534] WeiH::balanceOf(defaultPool: [0xC6F8f4FA75A4F1f4F289c000A871B2ea26b988]) [staticcall]
- 0
[0xC02aa39b223FE800A0e5C4F27aAD983C756Cc2], [1600000000000000000000], 3200000000000000000000
[426] collateralManager::getCCR() [staticcall]
- 1300000000000000000000
- false
[664] troveManager::getTroveStatus(borrower: [0x05650444aCe15A01762Bd97EE8fDeb495b3C2436]) [staticcall]
- 1
[1178] troveManager::applyPendingRewards(borrower: [0x05650444aCe15A01762Bd97EE8fDeb495b3C2436])
[1120] collateralManager::getCollateralSupport() [staticcall]
- [0xC02aa39b223FE800A0e5C4F27aAD983C756Cc2]
- 0
[9874] troveManager::getTroveColla(borrower: [0x05650444aCe15A01762Bd97EE8fDeb495b3C2436]) [staticcall]
[5843] collateralManager::getTroveColla(borrower: [0x05650444aCe15A01762Bd97EE8fDeb495b3C2436]) [staticcall]
[469] eToken::balanceOf(borrower: [0x05650444aCe15A01762Bd97EE8fDeb495b3C2436]) [staticcall]
- 1600000000000000000000
[469] eToken::balanceOf(borrower: [0x05650444aCe15A01762Bd97EE8fDeb495b3C2436]) [staticcall]
- 1600000000000000000000
[1600000000000000000000], [1600000000000000000000], [0xC02aa39b223FE800A0e5C4F27aAD983C756Cc2]
[1600000000000000000000], [1600000000000000000000], [0xC02aa39b223FE800A0e5C4F27aAD983C756Cc2]
[3936] collateralManager::getValue([0xC02aa39b223FE800A0e5C4F27aAD983C756Cc2], [1600000000000000000000], 2000000000000000000000) [staticcall]
- 3200000000000000000000
[2407] collateralManager::mintToken([], [], borrower: [0x05650444aCe15A01762Bd97EE8fDeb495b3C2436], 2000000000000000000000)
- 0
[2434] collateralManager::burnToken([], [], borrower: [0x05650444aCe15A01762Bd97EE8fDeb495b3C2436], 2000000000000000000000)
- 0
[4465] troveManager::getTroveDebt(borrower: [0x05650444aCe15A01762Bd97EE8fDeb495b3C2436]) [staticcall]
[483] collateralManager::getEUSDGasCompensation() [staticcall]
- 2000000000000000000000
[2435] troveDebt::balanceOf(borrower: [0x05650444aCe15A01762Bd97EE8fDeb495b3C2436]) [staticcall]
[474] troveManager::getTroveNormalizedDebt() [staticcall]
- 1600000000000000000000
- 2015000000000000000000
- 2215000000000000000000
[382] collateralManager::getCCR() [staticcall]
- 1300000000000000000000
[5765] collateralManager::getEntireCollValue(2000000000000000000000) [staticcall]
[498] eToken::totalSupply() [staticcall]
- 1600000000000000000000
[534] WeiH::balanceOf(defaultPool: [0xC6F8f4FA75A4F1f4F289c000A871B2ea26b988]) [staticcall]
- 0
[0xC02aa39b223FE800A0e5C4F27aAD983C756Cc2], [1600000000000000000000], 3200000000000000000000
[214] troveDebt::totalSupply() [staticcall]
[474] troveManager::getTroveNormalizedDebt() [staticcall]
- 1600000000000000000000
- 2015000000000000000000
[327] defaultPool::getEUSDDebt() [staticcall]
- 0
[685] eUSDToken::balanceOf(gasPool: [0x9A743489af33E59C9CD78203b311594F80ea36A9]) [staticcall]
- 2000000000000000000000
```

A vertical bar chart with a black background. A single red bar is positioned on the left side, extending from the bottom to the top of the chart area.

- 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 EUSD and collaterals tokens when active troves are updated.
- Check the flow of the EUSD and collaterals tokens when a trove is liquidated.

[illegible]

1. **Introduction**
 2. **Background**
 3. **Methodology**
 4. **Results**
 5. **Discussion**
 6. **Conclusion**
 7. **References**
 8. **Appendix**
 9. **Figure 1**
 10. **Figure 2**
 11. **Figure 3**
 12. **Figure 4**
 13. **Figure 5**
 14. **Figure 6**
 15. **Figure 7**
 16. **Figure 8**
 17. **Figure 9**
 18. **Figure 10**
 19. **Figure 11**
 20. **Figure 12**
 21. **Figure 13**
 22. **Figure 14**
 23. **Figure 15**
 24. **Figure 16**
 25. **Figure 17**
 26. **Figure 18**
 27. **Figure 19**
 28. **Figure 20**
 29. **Figure 21**
 30. **Figure 22**
 31. **Figure 23**
 32. **Figure 24**
 33. **Figure 25**
 34. **Figure 26**
 35. **Figure 27**
 36. **Figure 28**
 37. **Figure 29**
 38. **Figure 30**
 39. **Figure 31**
 40. **Figure 32**
 41. **Figure 33**
 42. **Figure 34**
 43. **Figure 35**
 44. **Figure 36**
 45. **Figure 37**
 46. **Figure 38**
 47. **Figure 39**
 48. **Figure 40**
 49. **Figure 41**
 50. **Figure 42**
 51. **Figure 43**
 52. **Figure 44**
 53. **Figure 45**
 54. **Figure 46**
 55. **Figure 47**
 56. **Figure 48**
 57. **Figure 49**
 58. **Figure 50**
 59. **Figure 51**
 60. **Figure 52**
 61. **Figure 53**
 62. **Figure 54**
 63. **Figure 55**
 64. **Figure 56**
 65. **Figure 57**
 66. **Figure 58**
 67. **Figure 59**
 68. **Figure 60**
 69. **Figure 61**
 70. **Figure 62**
 71. **Figure 63**
 72. **Figure 64**
 73. **Figure 65**
 74. **Figure 66**
 75. **Figure 67**
 76. **Figure 68**
 77. **Figure 69**
 78. **Figure 70**
 79. **Figure 71**
 80. **Figure 72**
 81. **Figure 73**
 82. **Figure 74**
 83. **Figure 75**
 84. **Figure 76**
 85. **Figure 77**
 86. **Figure 78**
 87. **Figure 79**
 88. **Figure 80**
 89. **Figure 81**
 90. **Figure 82**
 91. **Figure 83**
 92. **Figure 84**
 93. **Figure 85**
 94. **Figure 86**
 95. **Figure 87**
 96. **Figure 88**
 97. **Figure 89**
 98. **Figure 90**
 99. **Figure 91**
 100. **Figure 92**
 101. **Figure 93**
 102. **Figure 94**
 103. **Figure 95**
 104. **Figure 96**
 105. **Figure 97**
 106. **Figure 98**
 107. **Figure 99**
 108. **Figure 100**
 109. **Figure 101**
 110. **Figure 102**
 111. **Figure 103**
 112. **Figure 104**
 113. **Figure 105**
 114. **Figure 106**
 115. **Figure 107**
 116. **Figure 108**
 117. **Figure 109**
 118. **Figure 110**
 119. **Figure 111**
 120. **Figure 112**
 121. **Figure 113**
 122. **Figure 114**
 123. **Figure 115**
 124. **Figure 116**
 125. **Figure 117**
 126. **Figure 118**
 127. **Figure 119**
 128. **Figure 120**
 129. **Figure 121**
 130. **Figure 122**
 131. **Figure 123**
 132. **Figure 124**
 133. **Figure 125**
 134. **Figure 126**
 135. **Figure 127**
 136. **Figure 128**
 137. **Figure 129**
 138. **Figure 130**
 139. **Figure 131**
 140. **Figure 132**
 141. **Figure 133**
 142. **Figure 134**
 143. **Figure 135**
 144. **Figure 136**
 145. **Figure 137**
 146. **Figure 138**
 147. **Figure 139**
 148. **Figure 140**
 149. **Figure 141**
 150. **Figure 142**
 151. **Figure 143**
 152. **Figure 144**
 153. **Figure 145**
 154. **Figure 146**
 155. **Figure 147**
 156. **Figure 148**
 157. **Figure 149**
 158. **Figure 150**
 159. **Figure 151**
 160. **Figure 152**
 161. **Figure 153**
 162. **Figure 154**
 163. **Figure 155**
 164. **Figure 156**
 165. **Figure 157**
 166. **Figure 158**
 167. **Figure 159**
 168. **Figure 160**
 169. **Figure 161**
 170. **Figure 162**
 171. **Figure 163**
 172. **Figure 164**
 173. **Figure 165**
 174. **Figure 166**
 175. **Figure 167**
 176. **Figure 168**
 177. **Figure 169**
 178. **Figure 170**
 179. **Figure 171**
 180. **Figure 172**
 181. **Figure 173**
 182. **Figure 174**
 183. **Figure 175**
 184. **Figure 176**
 185. **Figure 177**
 186. **Figure 178**
 187. **Figure 179**
 188. **Figure 180**
 189. **Figure 181**
 190. **Figure 182**
 191. **Figure 183**
 192. **Figure 184**
 193. **Figure 185**
 194. **Figure 186**
 195. **Figure 187**
 196. **Figure 188**
 197. **Figure 189**
 198. **Figure 190**
 199. **Figure 191**
 200. **Figure 192**
 201. **Figure 193**
 202. **Figure 194**
 203. **Figure 195**
 204. **Figure 196**
 205. **Figure 197**
 206. **Figure 198**
 207. **Figure 199**
 208. **Figure 200**
 209. **Figure 201**
 210. **Figure 202**
 211. **Figure 203**
 212. **Figure 204**
 213. **Figure 205**
 214. **Figure 206**
 215. **Figure 207**
 216. **Figure 208**
 217. **Figure 209**

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

[illegible]

5. Tests focused on liquidations (checking the mode of the protocol and the different scenarios for closing the troves and distributing the rewards)

- Basic checks when $ICR < MCR$, the trove can be liquidated.
- Basic checks when $ICR > MCR$ and $TCR < CCR$, the trove can be liquidated as well.

```
[3181] priceFeedTestnet::setPrice(1500000000000000000)
    ↳ true
[0] VM::startPrank(borrower: {0x85650444aCe15A01762Bd97E88F0eb495b3C2436})
    ↳ ()
[432185] troveManager::liquidate(borrower2: {0x083d18A2EaE244be272a53f86e617cd9f33c8D68}, borrower: {0x85650444aCe15A01762Bd97E88F0eb495b3C2436})
    ↳ [398615] troveManager::liquidation::batchLiquidateTrove({0x083d18A2EaE244be272a53f86e617cd9f33c8D68}, borrower: {0x85650444aCe15A01762Bd97E88F0eb495b3C2436})
    ↳ [247] priceFeedTestnet::setPrice()
    ↳ ↳ 1500000000000000000
    ↳ [446] collateralManager::updateCollateral()
    ↳ ↳ ()
    ↳ [2404] stabilityPool::getTotalETHDeposits() [staticcall]
    ↳ ↳ 0
    ↳ [14287] troveManager::checkRecoveryMode(1500000000000000000) [staticcall]
    ↳ [2414] troveManager::totalSupply() [staticcall]
    ↳ ↳ [474] troveManager::getTroveNormalizedDebt() [staticcall]
    ↳ ↳ ↳ 1000000000000000000
    ↳ ↳ ↳ 3000000000000000000
    ↳ [387] defaultPool::getETHDebt() [staticcall]
    ↳ ↳ 0
    ↳ [606] eETHToken::balanceOf(gasPool: {0x9A743489af3E59C9CD78203b311594F80ea36A9}) [staticcall]
    ↳ ↳ 4000000000000000000
    ↳ [5785] collateralManager::getEntireCollValue(1500000000000000000) [staticcall]
    ↳ ↳ [298] eToken::totalSupply() [staticcall]
    ↳ ↳ ↳ 6000000000000000000
    ↳ ↳ [534] WEH::balanceOf(defaultPool: {0xc6f8f4FA75A4F1f4F289c000A871Bb2ea26D9B8}) [staticcall]
    ↳ ↳ ↳ 0
    ↳ ↳ [0xC02aa339b223FE8D0A0e5C4F27eAD9883C756Cc2], [4000000000000000000], 9000000000000000000
    ↳ [426] collateralManager::getTCR() [staticcall]
    ↳ ↳ 1200000000000000000
    ↳ ↳ true
    ↳ [2114] troveDebt::totalSupply() [staticcall]
    ↳ ↳ [674] troveManager::getTroveNormalizedDebt() [staticcall]
    ↳ ↳ ↳ 10000000000000000000
    ↳ ↳ ↳ 8000000000000000000
    ↳ [327] defaultPool::getETHDebt() [staticcall]
    ↳ ↳ 0
    ↳ [606] eETHToken::balanceOf(gasPool: {0x9A743489af3E59C9CD78203b311594F80ea36A9}) [staticcall]
    ↳ ↳ 4000000000000000000
    ↳ [6015] activePool::getTotalCollateral() [staticcall]
    ↳ [2788] troveManager::getCollateralSupport() [staticcall]
    ↳ ↳ [1120] collateralManager::getCollateralSupport() [staticcall]
    ↳ ↳ ↳ [0xC02aa339b223FE8D0A0e5C4F27eAD9883C756Cc2]
    ↳ ↳ ↳ [0xC02aa339b223FE8D0A0e5C4F27eAD9883C756Cc2], [4000000000000000000], 9000000000000000000
    ↳ [534] WEH::balanceOf(activePool: {0xd44ad51c26E1d66a13b7F5F11bcf771E5688A89a2}) [staticcall]
    ↳ ↳ ↳ 6000000000000000000
    ↳ ↳ ↳ 6000000000000000000, [0xC02aa339b223FE8D0A0e5C4F27eAD9883C756Cc2], [4000000000000000000]
    ↳ [6077] defaultPool::getTotalCollateral() [staticcall]
    ↳ [2788] troveManager::getCollateralSupport() [staticcall]
    ↳ ↳ [1120] collateralManager::getCollateralSupport() [staticcall]
    ↳ ↳ ↳ [0xC02aa339b223FE8D0A0e5C4F27eAD9883C756Cc2]
    ↳ ↳ ↳ [0xC02aa339b223FE8D0A0e5C4F27eAD9883C756Cc2]
    ↳ [534] WEH::balanceOf(defaultPool: {0xc6f8f4FA75A4F1f4F289c000A871Bb2ea26D9B8}) [staticcall]
    ↳ ↳ ↳ 0
    ↳ ↳ [0xC02aa339b223FE8D0A0e5C4F27eAD9883C756Cc2], [0]
    ↳ [3036] collateralManager::getValue([0xC02aa339b223FE8D0A0e5C4F27eAD9883C756Cc2], [4000000000000000000], 1500000000000000000) [staticcall]
    ↳ ↳ 4000000000000000000, [9000000000000000000]
    ↳ [283] collateralManager::getMCR() [staticcall]
    ↳ ↳ 1100000000000000000
    ↳ [664] troveManager::getTroveStatus(borrower2: {0x083d18A2EaE244be272a53f86e617cd9f33c8D68}) [staticcall]
    ↳ ↳ 1
    ↳ [25706] troveManager::getCurrentTCR(borrower2: {0x083d18A2EaE244be272a53f86e617cd9f33c8D68}, 1500000000000000000) [staticcall]
    ↳ ↳ [1120] collateralManager::getCollateralSupport() [staticcall]
    ↳ ↳ ↳ [0xC02aa339b223FE8D0A0e5C4F27eAD9883C756Cc2]
    ↳ [5646] collateralManager::getTroveColl(borrower2: {0x083d18A2EaE244be272a53f86e617cd9f33c8D68}) [staticcall]
    ↳ ↳ [460] eToken::balanceOf(borrower2: {0x083d18A2EaE244be272a53f86e617cd9f33c8D68}) [staticcall]
    ↳ ↳ ↳ 3000000000000000000
    ↳ ↳ [469] eToken::shareOf(borrower2: {0x083d18A2EaE244be272a53f86e617cd9f33c8D68}) [staticcall]
    ↳ ↳ ↳ 2000000000000000000
    ↳ ↳ ↳ [3000000000000000000], [3000000000000000000], [0xC02aa339b223FE8D0A0e5C4F27eAD9883C756Cc2]
    ↳ [1120] collateralManager::getCollateralSupport() [staticcall]
    ↳ ↳ ↳ [0xC02aa339b223FE8D0A0e5C4F27eAD9883C756Cc2]
```

6. Tests depending on the current mode of the protocol (normal or recovery mode, to check which protocol actions are permitted and not permitted during recovery mode)

- Check if actions that mint EUSD are not permitted during recovery mode.
- Check if actions that burn EUSD are incentivized for users during recovery mode.

```
[3181] priceFeedTestnet::getPrice(15000000000000000000)
└─ true
[9] V0: startPlan(borrower: [0x05650444aCe15A01762Bd97EE8Fdeb495b3C2436])
└─ ()
[7028] borrowerOperations::borrowColl([0x02aa39b223FE80A0e5C4F27eAD983C756C2], [1000000000000000000], borrower: [0x05650444aCe15A01762Bd97EE8Fdeb495b3C2436], borrower: [0x05650444aCe15A01762Bd97EE8Fdeb495b3C2436])
└─ [272] collateralManager::getSupportCollateral([0x02aa39b223FE80A0e5C4F27eAD983C756C2]) [staticcall]
└─ true
[5269] collateralManager::getShare([0x02aa39b223FE80A0e5C4F27eAD983C756C2], [1000000000000000000]) [staticcall]
└─ [238] token::getShare(1000000000000000000) [staticcall]
└─ * 1000000000000000000
[247] priceFeedTestnet::getPrice()
└─ * 1500000000000000000
[446] collateralManager::priceUpdate()
└─ ()
[14287] troveManager::checkRecoveryMode(15000000000000000000) [staticcall]
└─ [214] troveDebt::totalSupply() [staticcall]
└─ [674] troveManager::getTroveNormalizedDebt() [staticcall]
└─ * 1000000000000000000
└─ * 860000000000000000
[337] defaultPool::getDebt() [staticcall]
└─ 0
[685] troveManager::balanceOf(gasPool: [0x9A74349a73E59C9C072836311596F80a36A9]) [staticcall]
└─ * 4000000000000000000
[5755] collateralManager::getEntireCollValue(15000000000000000000) [staticcall]
└─ [398] token::totalSupply() [staticcall]
└─ * 4000000000000000000
└─ [634] WITH::balanceOf(defaultPool: [0xC6f8f4FA75A4F1f4f289c000aB71b2ea26b9B8]) [staticcall]
└─ 0
└─ * [0x02aa39b223FE80A0e5C4F27eAD983C756C2], [00000000000000000000], 90000000000000000000
[426] collateralManager::getCDR() [staticcall]
└─ * 1000000000000000000
└─ true
[464] troveManager::getTroveStatus(borrower: [0x05650444aCe15A01762Bd97EE8Fdeb495b3C2436]) [staticcall]
└─ 1
[4178] troveManager::applyPendingRewards(borrower: [0x05650444aCe15A01762Bd97EE8Fdeb495b3C2436])
└─ [1128] collateralManager::getCollateralSupport() [staticcall]
└─ * [0x02aa39b223FE80A0e5C4F27eAD983C756C2]
└─ ()
[9074] troveManager::getTroveColl(borrower: [0x05650444aCe15A01762Bd97EE8Fdeb495b3C2436]) [staticcall]
└─ [8546] collateralManager::getTroveColl(borrower: [0x05650444aCe15A01762Bd97EE8Fdeb495b3C2436]) [staticcall]
└─ [608] token::balanceOf(borrower: [0x05650444aCe15A01762Bd97EE8Fdeb495b3C2436]) [staticcall]
└─ * 2000000000000000000
└─ [609] token::balanceOf(borrower: [0x05650444aCe15A01762Bd97EE8Fdeb495b3C2436]) [staticcall]
└─ * 2000000000000000000
└─ * [0x02aa39b223FE80A0e5C4F27eAD983C756C2], [0x02aa39b223FE80A0e5C4F27eAD983C756C2]
└─ [5000000000000000000], [2000000000000000000], [0x02aa39b223FE80A0e5C4F27eAD983C756C2]
[2032] collateralManager::getCollateralValue([0x02aa39b223FE80A0e5C4F27eAD983C756C2], [2000000000000000000], 15000000000000000000) [staticcall]
└─ * 4500000000000000000, [45000000000000000000]
[2457] collateralManager::mintToken([], [], borrower: [0x05650444aCe15A01762Bd97EE8Fdeb495b3C2436], 15000000000000000000)
└─ [1], 0
[2749] collateralManager::burnToken([0x02aa39b223FE80A0e5C4F27eAD983C756C2], [1000000000000000000], borrower: [0x05650444aCe15A01762Bd97EE8Fdeb495b3C2436], 15000000000000000000)
└─ [4142] token::burn(borrower: [0x05650444aCe15A01762Bd97EE8Fdeb495b3C2436], 1000000000000000000)
└─ emit Transfer(Trove, borrower: [0x05650444aCe15A01762Bd97EE8Fdeb495b3C2436], to: 0000000000000000000000000000000000000000, value: 1000000000000000000)
└─ * 1000000000000000000
└─ [1000000000000000000], [15000000000000000000]
[1645] troveManager::getTroveDebt(borrower: [0x05650444aCe15A01762Bd97EE8Fdeb495b3C2436]) [staticcall]
└─ [403] collateralManager::getEUSDasCompensation() [staticcall]
└─ * 2000000000000000000
[2435] troveManager::balanceOf(borrower: [0x05650444aCe15A01762Bd97EE8Fdeb495b3C2436]) [staticcall]
└─ [674] troveManager::getTroveNormalizedDebt() [staticcall]
└─ * 10000000000000000000
└─ * 402000000000000000000
└─ * 422000000000000000000
└─ * "16"
```

- 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 EUSD.
- Set 2 redeemers of EUSD.
- Change the price of the collateral in the oracle.
- Redeemers redeem collateral.
- Troves being liquidated.
- Check the flow of EUSD and collateral tokens and check the priority collateral logic is properly working.

[illegible]

8. Deeply test all the possible cases of the system state and a specific trove being liquidated to ensure distributions of collaterals and rewards are correctly done as explained within the documentation (between depositors, active troves and the different pools).

- $ICR < MCR \ \& \ SP.EUSD \geq trove.debt \ \& \ TCR \geq 130\%$
- $ICR < MCR \ \& \ SP.EUSD < trove.debt \ \& \ TCR \geq 130\%$
- $ICR < MCR \ \& \ SP.EUSD = 0 \ \& \ TCR \geq 130\%$
- $ICR \leq 100\% \ \& \ TCR < 130\%$
- $100\% < ICR < MCR \ \& \ SP.EUSD > trove.debt \ \& \ TCR < 130\%$
- $100\% < ICR < MCR \ \& \ SP.EUSD < trove.debt \ \& \ TCR < 130\%$
- $MCR \leq ICR < TCR \ \& \ SP.EUSD \geq trove.debt \ \& \ TCR < 130\%$

```

(423907) troveManager::liquidateTrove(1)
[400895] troveManagerLiquidations::liquidateTrove(1, borrower: [0x05650444aC015A01762Bd97EE8F0eb495b3C2436])
[1120] collateralManager::getCollateralSupport() [staticcall]
    [0xC02aa39b223FE80A0e5C4F27eAD9883C756cC2]
[247] priceFeed::fetchPrice()
    [0x0000000000000000]
[446] collateralManager::updatePrice()
    [0]
[2404] stabilityPool::getTotalEUSDDeposits() [staticcall]
    [0]
[14287] troveManager::checkRecoveryMode(15000000000000000000) [staticcall]
[2144] troveManager::totalSupply() [staticcall]
    [0]
[674] troveManager::getTroveNormalisedDebt() [staticcall]
    [1000000000000000000]
[327] defaultPool::getEUSDDebt() [staticcall]
    [0]
[406] eUSDToken::balanceOf(gasPool: [0x9A743489aF3E59C9D78203b31159f80ea36A9]) [staticcall]
    [4000000000000000000]
[5785] collateralManager::getEntireCollValue(15000000000000000000) [staticcall]
    [0]
[328] eUSDToken::balanceOf() [staticcall]
    [6000000000000000000]
[524] WEH::balanceOf(defaultPool: [0xC68F4FA75A4F14F289c080A871Bb2ea26b988]) [staticcall]
    [0]
    [0xC02aa39b223FE80A0e5C4F27eAD9883C756cC2], [6000000000000000000], 9000000000000000000
[426] collateralManager::getCDR() [staticcall]
    [1300000000000000000]
    [true]
[2114] troveDebt::totalSupply() [staticcall]
[674] troveManager::getTroveNormalisedDebt() [staticcall]
    [1000000000000000000]
[327] defaultPool::getEUSDDebt() [staticcall]
    [0]
[406] eUSDToken::balanceOf(gasPool: [0x9A743489aF3E59C9D78203b31159f80ea36A9]) [staticcall]
    [4000000000000000000]
[6015] activePool::getTotalCollateral() [staticcall]
[2788] troveManager::getCollateralSupport() [staticcall]
    [1120] collateralManager::getCollateralSupport() [staticcall]
        [0xC02aa39b223FE80A0e5C4F27eAD9883C756cC2]
        [0xC02aa39b223FE80A0e5C4F27eAD9883C756cC2]
[524] WEH::balanceOf(activePool: [0x644a0516261A6a13b7FBf10cF721E688A89a2]) [staticcall]
    [6000000000000000000]
    [0xC02aa39b223FE80A0e5C4F27eAD9883C756cC2], [6000000000000000000]
[4077] defaultPool::getTotalCollateral() [staticcall]
[2788] troveManager::getCollateralSupport() [staticcall]
    [1120] collateralManager::getCollateralSupport() [staticcall]
        [0xC02aa39b223FE80A0e5C4F27eAD9883C756cC2]
        [0xC02aa39b223FE80A0e5C4F27eAD9883C756cC2]
[524] WEH::balanceOf(defaultPool: [0xC68F4FA75A4F14F289c080A871Bb2ea26b988]) [staticcall]
    [0]
    [0xC02aa39b223FE80A0e5C4F27eAD9883C756cC2], [0]
[3036] collateralManager::getValue([0xC02aa39b223FE80A0e5C4F27eAD9883C756cC2], [6000000000000000000], 15000000000000000000) [staticcall]
    [0]
[420] sortedTrove::getLast() [staticcall]
    [0]
    [0x05650444aC015A01762Bd97EE8F0eb495b3C2436]
[266] sortedTrove::getFirst() [staticcall]
    [0]
    [0x083d18A2E440e272a53f86e617cd9F33c8b68]
[388] collateralManager::getMCR() [staticcall]
    [1000000000000000000]
[621] sortedTrove::getPrev(borrower: [0x05650444aC015A01762Bd97EE8F0eb495b3C2436]) [staticcall]
    [0]
    [0x083d18A2E440e272a53f86e617cd9F33c8b68]
[25704] troveManager::getCurrentColl(borrower: [0x05650444aC015A01762Bd97EE8F0eb495b3C2436], 15000000000000000000) [staticcall]
    [1120] collateralManager::getCollateralSupport() [staticcall]
        [0xC02aa39b223FE80A0e5C4F27eAD9883C756cC2]
    [5040] eToken::balanceOf(borrower: [0x05650444aC015A01762Bd97EE8F0eb495b3C2436]) [staticcall]
        [660] eToken::balanceOf(borrower: [0x05650444aC015A01762Bd97EE8F0eb495b3C2436]) [staticcall]
            [3000000000000000000]
        [669] eToken::balanceOf(borrower: [0x05650444aC015A01762Bd97EE8F0eb495b3C2436]) [staticcall]
            [3000000000000000000]
            [3000000000000000000], [3000000000000000000], [0xC02aa39b223FE80A0e5C4F27eAD9883C756cC2]

```


9. 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.
- Check how the system handles when priority collateral logic is executed.
- Check how collaterals are calculated when opening a trove.
- Check how collaterals are recalculated when adding a specific collateral to an already active trove.
- Check how the system merges the ETH with collaterals into the protocol collaterals array.

[illegible]

10. 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



AUTOMATED TESTING

42

contracts/CollateralManager.sol

```
ERDBase.collateralManager (src/Dependencies/ERDBase.sol#29) is never initialized. It is used in:  
  - ERDBase.getEntireSystemColl() (uint256) (src/Dependencies/ERDBase.sol#90-104)  
ERDBase.audToken (src/Dependencies/ERDBase.sol#32) is never initialized. It is used in:  
  - ERDBase.getEntireSystemDebt() (src/Dependencies/ERDBase.sol#106-115)  
ERDBase.troveBorrow (src/Dependencies/ERDBase.sol#33) is never initialized. It is used in:  
  - ERDBase.getEntireSystemDebt() (src/Dependencies/ERDBase.sol#106-115)  
ERDBase.gasPoolAddress (src/Dependencies/ERDBase.sol#37) is never initialized. It is used in:  
  - ERDBase.getEntireSystemDebt() (src/Dependencies/ERDBase.sol#106-115)  
Reference: https://github.com/crytic/slither/wiki/detector-documentation#uninitialized-state-variables  
  
CollateralManager.initialize() (src/CollateralManager.sol#68-72) performs a multiplication on the result of a division:  
  - BORROWING_FEE_FLOOR = (DECIMAL_PRECISION / 10000) * 75 (src/CollateralManager.sol#67)  
CollateralManager.initialize() (src/CollateralManager.sol#68-72) performs a multiplication on the result of a division:  
  - REDEMPTION_FEE_FLOOR = (DECIMAL_PRECISION / 10000) * 75 (src/CollateralManager.sol#69)  
CollateralManager.initialize() (src/CollateralManager.sol#68-72) performs a multiplication on the result of a division:  
  - RECOVERY_FEE = (DECIMAL_PRECISION / 10000) * 25 (src/CollateralManager.sol#70)  
CollateralManager.initialize() (src/CollateralManager.sol#68-72) performs a multiplication on the result of a division:  
  - MAX_BORROWING_FEE = (DECIMAL_PRECISION / 100) * 5 (src/CollateralManager.sol#71)  
CollateralManager_collateral(address,uint256) (src/CollateralManager.sol#229-241) performs a multiplication on the result of a division:  
  value = coll_adjust_fetchPrice_view().mul(valueRatio).div(DECIMAL_PRECISION).mul(amount).div(DECIMAL_PRECISION).mul(price).div(DECIMAL_PRECISION) (src/CollateralManager.sol#300-307)  
ERDMath.calculateCompoundedInterest(uint256,uint40,uint256) (src/Dependencies/ERDMath.sol#250-280) performs a multiplication on the result of a division:  
  - ratePerSec = rate / SECONDS_PER_YEAR (src/Dependencies/ERDMath.sol#246)  
  - wadRayMath.mul().add(ratePerSecond.mul(exp)).add(secondTerm).add(thirdTerm) (src/Dependencies/ERDMath.sol#276-279)  
Reference: https://github.com/crytic/slither/wiki/detector-documentation#divide-before-multiply  
  
ERDMath.calculateCompoundedInterest(uint256,uint40,uint256) (src/Dependencies/ERDMath.sol#250-280) uses a dangerous strict equality:  
  - exp == 0 (src/Dependencies/ERDMath.sol#258)  
Reference: https://github.com/crytic/slither/wiki/detector-documentation#dangerous-strict-equalities  
  
ERDMath_getArrayCopy(uint256[]).i (src/Dependencies/ERDMath.sol#210) is a local variable never initialized  
ERDMath_getArrayLen(uint256[]).i (src/Dependencies/ERDMath.sol#196) is a local variable never initialized  
Reference: https://github.com/crytic/slither/wiki/detector-documentation#uninitialized-local-variables  
  
CollateralManager_priceUpdate() (src/CollateralManager.sol#242-252) ignores return value by Oracle(collateralParams[collateralSupport[0]].oracle).fetchPrice() (src/CollateralManager.sol#247)  
CollateralManager_validAdjustment(address,address,uint256) (src/CollateralManager.sol#606-633) ignores return value by Oracle(collateralParams[collateral].oracle).fetchPrice() (src/CollateralManager.sol#616)  
Reference: https://github.com/crytic/slither/wiki/detector-documentation#unused-return
```

contracts/PriceFeed.sol

```
ERDMath.calculateCompoundedInterest(uint256,uint40,uint256) (src/Dependencies/ERDMath.sol#250-280) performs a multiplication on the result of a division:  
  - ratePerSec = rate / SECONDS_PER_YEAR (src/Dependencies/ERDMath.sol#246)  
  - wadRayMath.mul().add(ratePerSecond.mul(exp)).add(secondTerm).add(thirdTerm) (src/Dependencies/ERDMath.sol#276-279)  
Reference: https://github.com/crytic/slither/wiki/detector-documentation#divide-before-multiply  
  
ERDMath.calculateCompoundedInterest(uint256,uint40,uint256) (src/Dependencies/ERDMath.sol#250-280) uses a dangerous strict equality:  
  - exp == 0 (src/Dependencies/ERDMath.sol#258)  
Reference: https://github.com/crytic/slither/wiki/detector-documentation#dangerous-strict-equalities  
  
PriceFeed_getPrevChainLinkResponse(uint8,uint8).timestamp (src/PriceFeed.sol#678) is a local variable never initialized  
ERDMath_getArrayCopy(uint256[]).i (src/Dependencies/ERDMath.sol#210) is a local variable never initialized  
PriceFeed_getCurrentTellerResponse().timestampRetrieved (src/PriceFeed.sol#616) is a local variable never initialized  
ERDMath_arrayLen(uint256[]).i (src/Dependencies/ERDMath.sol#196) is a local variable never initialized  
PriceFeed_getCurrentChainLinkResponse().timestamp (src/PriceFeed.sol#650) is a local variable never initialized  
PriceFeed_getCurrentChainLinkResponse().decimals (src/PriceFeed.sol#637) is a local variable never initialized  
PriceFeed_getPrevChainLinkResponse(uint8,uint8).roundId (src/PriceFeed.sol#678) is a local variable never initialized  
PriceFeed_getPrevChainLinkResponse(uint8,uint8).answer (src/PriceFeed.sol#676) is a local variable never initialized  
PriceFeed_getCurrentChainLinkResponse().ifRetrieve (src/PriceFeed.sol#614) is a local variable never initialized  
PriceFeed_getCurrentChainLinkResponse().roundId (src/PriceFeed.sol#647) is a local variable never initialized  
PriceFeed_getChainLinkPrice(uint256,uint256).index (src/PriceFeed.sol#655) is a local variable never initialized  
PriceFeed_getCurrentTellerResponse().value (src/PriceFeed.sol#615) is a local variable never initialized  
PriceFeed_getCurrentChainLinkResponse().answer (src/PriceFeed.sol#648) is a local variable never initialized  
Reference: https://github.com/crytic/slither/wiki/detector-documentation#uninitialized-local-variables  
  
PriceFeed_getCurrentChainLinkResponse() (src/PriceFeed.sol#608-629) ignores return value by tellorBols.getTellerCurrentValue(ETHUSD_TELOR_REG_ID) (src/PriceFeed.sol#613-628)  
PriceFeed_getCurrentChainLinkResponse() (src/PriceFeed.sol#631-663) ignores return value by priceAggregator.decimals() (src/PriceFeed.sol#637-643)  
PriceFeed_getCurrentChainLinkResponse() (src/PriceFeed.sol#631-663) ignores return value by priceAggregator.latestRoundData() (src/PriceFeed.sol#646-662)  
PriceFeed_getPrevChainLinkResponse(uint8,uint8) (src/PriceFeed.sol#665-692) ignores return value by priceAggregator.getRoundData(currentRoundId - 1) (src/PriceFeed.sol#674-691)  
Reference: https://github.com/crytic/slither/wiki/detector-documentation#unused-return  
  
Variable 'PriceFeed_getCurrentTellerResponse().ifRetrieve (src/PriceFeed.sol#614)' in PriceFeed_getCurrentTellerResponse() (src/PriceFeed.sol#608-629) potentially used before declaration: tellorResponse.ifRetrieve  
Variable 'PriceFeed_getCurrentTellerResponse().value (src/PriceFeed.sol#615)' in PriceFeed_getCurrentTellerResponse() (src/PriceFeed.sol#608-629) potentially used before declaration: tellorResponse.value = value  
Variable 'PriceFeed_getCurrentTellerResponse().timestampRetrieved (src/PriceFeed.sol#616)' in PriceFeed_getCurrentTellerResponse() (src/PriceFeed.sol#608-629) potentially used before declaration: tellorResponse.  
Variable 'PriceFeed_getCurrentChainLinkResponse().decimals (src/PriceFeed.sol#637)' in PriceFeed_getCurrentChainLinkResponse() (src/PriceFeed.sol#631-663) potentially used before declaration: chainLinkResponse.de  
Variable 'PriceFeed_getCurrentChainLinkResponse().timestamp (src/PriceFeed.sol#650)' in PriceFeed_getCurrentChainLinkResponse() (src/PriceFeed.sol#631-663) potentially used before declaration: chainLinkResponse.res  
Variable 'PriceFeed_getCurrentChainLinkResponse().answer (src/PriceFeed.sol#648)' in PriceFeed_getCurrentChainLinkResponse() (src/PriceFeed.sol#631-663) potentially used before declaration: chainLinkResponse.answ  
Variable 'PriceFeed_getCurrentChainLinkResponse().roundId (src/PriceFeed.sol#647)' in PriceFeed_getCurrentChainLinkResponse() (src/PriceFeed.sol#631-663) potentially used before declaration: chainLinkResponse.c  
Variable 'PriceFeed_getPrevChainLinkResponse(uint8,uint8).roundId (src/PriceFeed.sol#678)' in PriceFeed_getPrevChainLinkResponse(uint8,uint8) (src/PriceFeed.sol#665-692) potentially used before declaration: pre  
682)  
Variable 'PriceFeed_getPrevChainLinkResponse(uint8,uint8).answer (src/PriceFeed.sol#676)' in PriceFeed_getPrevChainLinkResponse(uint8,uint8) (src/PriceFeed.sol#665-692) potentially used before declaration: prev  
Variable 'PriceFeed_getPrevChainLinkResponse(uint8,uint8).timestamp (src/PriceFeed.sol#678)' in PriceFeed_getPrevChainLinkResponse(uint8,uint8) (src/PriceFeed.sol#665-692) potentially used before declaration: p  
684)  
Reference: https://github.com/crytic/slither/wiki/detector-documentation#pre-declaration-usage-of-local-variables
```

contracts/StabilityPool.sol

```
StabilityPool_sendCollateralGainToDepositor(address[],uint256[]) (src/StabilityPool.sol#1048-1092) sends eth to arbitrary user  
Dangerous call:  
  - (success) = msg.sender.call(value: ETHAmount)() (src/StabilityPool.sol#1084)  
Reference: https://github.com/crytic/slither/wiki/detector-documentation#functions-that-send-ether-to-arbitrary-destinations  
  
ERDBase.defaultPool (src/Dependencies/ERDBase.sol#24) is never initialized. It is used in:  
  - ERDBase.getEntireSystemColl() (src/Dependencies/ERDBase.sol#76-90)  
  - ERDBase.getEntireSystemDebt() (src/Dependencies/ERDBase.sol#106-115)  
ERDBase.troveDebt (src/Dependencies/ERDBase.sol#23) is never initialized. It is used in:  
  - ERDBase.getEntireSystemDebt() (src/Dependencies/ERDBase.sol#106-115)  
ERDBase.gasPoolAddress (src/Dependencies/ERDBase.sol#37) is never initialized. It is used in:  
  - ERDBase.getEntireSystemDebt() (src/Dependencies/ERDBase.sol#106-115)  
Reference: https://github.com/crytic/slither/wiki/detector-documentation#uninitialized-state-variables  
  
ERDMath.calculateCompoundedInterest(uint256,uint40,uint256) (src/Dependencies/ERDMath.sol#250-280) performs a multiplication on the result of a division:  
  - ratePerSec = rate / SECONDS_PER_YEAR (src/Dependencies/ERDMath.sol#246)  
  - wadRayMath.mul().add(ratePerSecond.mul(exp)).add(secondTerm).add(thirdTerm) (src/Dependencies/ERDMath.sol#276-279)  
StabilityPool_computeGainPerUnitStaked(uint256,uint256) (src/StabilityPool.sol#520-545) performs a multiplication on the result of a division:  
  - gainPerUnitStaked = gainMultiplier.div(totalUSDDeposits) (src/StabilityPool.sol#539)  
  - lastGainError = gainMultiplier.sub(gainPerUnitStaked.mul(totalUSDDeposits)) (src/StabilityPool.sol#540-542)  
Reference: https://github.com/crytic/slither/wiki/detector-documentation#divide-before-multiply  
  
ERDMath.calculateCompoundedInterest(uint256,uint40,uint256) (src/Dependencies/ERDMath.sol#250-280) uses a dangerous strict equality:  
  - exp == 0 (src/Dependencies/ERDMath.sol#258)  
Reference: https://github.com/crytic/slither/wiki/detector-documentation#dangerous-strict-equalities  
  
StabilityPool_updateFrontEndTakeAndSnapshot(address,uint256) (src/StabilityPool.sol#1186-1214) deletes IStabilityPool.Snapshots (src/Interfaces/IStabilityPool.sol#36-42) which contains a mapping:  
  - delete fromEndSnapshots[frontEnd] (src/StabilityPool.sol#1192)  
Reference: https://github.com/crytic/slither/wiki/detector-documentation#deletion-on-mapping-containing-a-structure
```

contracts/TroveManagerLiquidations.sol

```

EROMath.calculateCompoundInterest(uint256,uint48,uint256) (src/Dependencies/EROMath.sol#250-288) performs a multiplication on the result of a division:
    -ratePerSecound = rate / SECUNDS_PER_YEAR (src/Dependencies/EROMath.sol#246)
    -wadRayMach.ray().add(ratePerSecound.mul(exp)).add(secondTerm).add(thirdTerm) (src/Dependencies/EROMath.sol#276-279)
TroveManagerLiquidations._getOffsetAndRedistributionVals(uint256,uint256,address[],uint256[]) (src/TroveManagerLiquidations.sol#388-465) performs a multiplication on the result of a division:
    -offset = coll[i].mul(totalValueToSendToSP.mul(_lumpct).div(values[i])).div(_lumpct) (src/TroveManagerLiquidations.sol#448-460)
TroveManagerLiquidations._getCappedCollPortion(address[],uint256[],uint256) (src/TroveManagerLiquidations.sol#516-555) performs a multiplication on the result of a division:
    -offset = coll[i].mul(portion.mul(_lumpct).div(values[i])).div(_lumpct) (src/TroveManagerLiquidations.sol#544-546)
Reference: https://github.com/crytic/slither/wiki/detector-documentation#divide-before-multiply

EROMath.calculateCompoundInterest(uint256,uint48,uint256) (src/Dependencies/EROMath.sol#250-288) uses a dangerous strict equality:
    -exp == 0 (src/Dependencies/EROMath.sol#258)
TroveManagerLiquidations._getCappedCollPortion(address[],uint256[],uint256) (src/TroveManagerLiquidations.sol#516-555) uses a dangerous strict equality:
    -values[i] == 0 || portion == 0 (src/TroveManagerLiquidations.sol#552)
TroveManagerLiquidations._getCappedCollPortion(address[],uint256[],uint256) (src/TroveManagerLiquidations.sol#516-555) uses a dangerous strict equality:
    -i == 0 (src/TroveManagerLiquidations.sol#534)
TroveManagerLiquidations._getCappedCollPortion(address[],uint256[],uint256) (src/TroveManagerLiquidations.sol#516-555) uses a dangerous strict equality:
    -i == 0 (src/TroveManagerLiquidations.sol#550)
TroveManagerLiquidations._getOffsetAndRedistributionVals(uint256,uint256,address[],uint256[]) (src/TroveManagerLiquidations.sol#388-465) uses a dangerous strict equality:
    -i == 0 (src/TroveManagerLiquidations.sol#420)
TroveManagerLiquidations._getOffsetAndRedistributionVals(uint256,uint256,address[],uint256[]) (src/TroveManagerLiquidations.sol#388-465) uses a dangerous strict equality:
    -totalValueToSendToSP == 0 (src/TroveManagerLiquidations.sol#435)
TroveManagerLiquidations._getOffsetAndRedistributionVals(uint256,uint256,address[],uint256[]) (src/TroveManagerLiquidations.sol#388-465) uses a dangerous strict equality:
    -i == 0 (src/TroveManagerLiquidations.sol#438)
TroveManagerLiquidations._getOffsetAndRedistributionVals(uint256,uint256,address[],uint256[]) (src/TroveManagerLiquidations.sol#388-465) uses a dangerous strict equality:
    -i == 0 (src/TroveManagerLiquidations.sol#458)
TroveManagerLiquidations._getTotalFromBatchLiquidate_RecoveryMode(ICollateralManager,IPool,IdeaultPool,uint256,uint256,address[]) (src/TroveManagerLiquidations.sol#968-1074) uses a dangerous strict equality:
    -vars.iOR > wcr && vars.remainingEUSDInStabPool == 0 (src/TroveManagerLiquidations.sol#1081)
TroveManagerLiquidations._getTotalFromBatchLiquidate_RecoveryMode(DataTypes.ContractsCache,uint256,uint256,uint256) (src/TroveManagerLiquidations.sol#690-795) uses a dangerous strict equality:
    -vars.iOR > wcr && vars.remainingEUSDInStabPool == 0 (src/TroveManagerLiquidations.sol#722)
Reference: https://github.com/crytic/slither/wiki/detector-documentation#dangerous-strict-equalities

TroveManagerLiquidations._getOffsetAndRedistributionVals(uint256,uint256,address[],uint256[]) (src/TroveManagerLiquidations.sol#388-465) contains a tautology or contradiction:
    -i >= 0 (src/TroveManagerLiquidations.sol#420)
TroveManagerLiquidations._getCappedCollPortion(address[],uint256[],uint256) (src/TroveManagerLiquidations.sol#516-555) contains a tautology or contradiction:
    -i >= 0 (src/TroveManagerLiquidations.sol#531)
Reference: https://github.com/crytic/slither/wiki/detector-documentation#tautology-or-contradiction

TroveManagerLiquidations.batchLiquidateTrove(address[],address).vars (src/TroveManagerLiquidations.sol#855) is a local variable never initialized
TroveManagerLiquidations._liquidateNormalMode(IPool,IdeaultPool,uint256,uint256).vars (src/TroveManagerLiquidations.sol#157) is a local variable never initialized
TroveManagerLiquidations._getTotalFromBatchLiquidate_NormalMode(IPool,IdeaultPool,uint256,uint256,address[]).vars (src/TroveManagerLiquidations.sol#1408) is a local variable never initialized
TroveManagerLiquidations._getTotalFromBatchLiquidate_RecoveryMode(DataTypes.ContractsCache,uint256,uint256,uint256).vars (src/TroveManagerLiquidations.sol#697) is a local variable never initialized
EROMath._getArrayCopy(uint256[]).i (src/Dependencies/EROMath.sol#219) is a local variable never initialized
TroveManagerLiquidations.liquidateTrove(uint256,address).vars (src/TroveManagerLiquidations.sol#579) is a local variable never initialized
TroveManagerLiquidations.liquidateRecoveryMode(IPool,IdeaultPool,uint256,uint256,uint256).vars (src/TroveManagerLiquidations.sol#227) is a local variable never initialized
EROMath._arrayNonZero(uint256[]).i (src/Dependencies/EROMath.sol#196) is a local variable never initialized
TroveManagerLiquidations.liquidateRecoveryMode(IPool,IdeaultPool,uint256,uint256,uint256).zeroVals (src/TroveManagerLiquidations.sol#378) is a local variable never initialized
TroveManagerLiquidations._getTotalFromBatchLiquidate_RecoveryMode(ICollateralManager,IPool,IdeaultPool,uint256,uint256,address[]).vars (src/TroveManagerLiquidations.sol#976) is a local variable never initialized
TroveManagerLiquidations._getTotalFromBatchLiquidate_RecoveryMode(IPool,IdeaultPool,uint256,uint256,uint256).vars (src/TroveManagerLiquidations.sol#885) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/detector-documentation#uninitialized-local-variables

```

contracts/TroveManagerRedemptions.sol

```

EROMath.calculateCompoundInterest(uint256,uint48,uint256) (src/Dependencies/EROMath.sol#250-288) performs a multiplication on the result of a division:
    -ratePerSecound = rate / SECUNDS_PER_YEAR (src/Dependencies/EROMath.sol#246)
    -wadRayMach.ray().add(ratePerSecound.mul(exp)).add(secondTerm).add(thirdTerm) (src/Dependencies/EROMath.sol#276-279)
TroveManagerRedemptions._calculateCollat(uint256,address[],uint256[]) (src/TroveManagerRedemptions.sol#232-284) performs a multiplication on the result of a division:
    -portion = EUSDOf_wad(_lumpct).div(value) (src/TroveManagerRedemptions.sol#272)
    -offset = coll.mul(portion).div(_lumpct) (src/TroveManagerRedemptions.sol#272)
Reference: https://github.com/crytic/slither/wiki/detector-documentation#divide-before-multiply

EROMath.calculateCompoundInterest(uint256,uint48,uint256) (src/Dependencies/EROMath.sol#250-288) uses a dangerous strict equality:
    -exp == 0 (src/Dependencies/EROMath.sol#258)
TroveManagerRedemptions._calculateCollat(uint256,address[],uint256[]) (src/TroveManagerRedemptions.sol#232-284) uses a dangerous strict equality:
    -EUSDOf == 0 (src/TroveManagerRedemptions.sol#258)
TroveManagerRedemptions._redemCollateralFromTrove(DataTypes.ContractsCache,address,uint256,uint256,address,uint256) (src/TroveManagerRedemptions.sol#121-238) uses a dangerous strict equality:
    -newDust == gas (src/TroveManagerRedemptions.sol#176)
Reference: https://github.com/crytic/slither/wiki/detector-documentation#dangerous-strict-equalities

TroveManagerRedemptions._calculateCollat(uint256,address[],uint256[]) (src/TroveManagerRedemptions.sol#232-284) contains a tautology or contradiction:
    -i_scope_0 >= 0 (src/TroveManagerRedemptions.sol#254)
TroveManagerRedemptions._calculateCollat(uint256,address[],uint256[]) (src/TroveManagerRedemptions.sol#232-284) contains a tautology or contradiction:
    -i >= 0 (src/TroveManagerRedemptions.sol#240)
Reference: https://github.com/crytic/slither/wiki/detector-documentation#tautology-or-contradiction

EROMath._getArrayCopy(uint256[]).i (src/Dependencies/EROMath.sol#219) is a local variable never initialized
EROMath._arrayNonZero(uint256[]).i (src/Dependencies/EROMath.sol#196) is a local variable never initialized
TroveManagerRedemptions.redemCollateral(uint256,address,address,uint256,uint256,uint256).totals (src/TroveManagerRedemptions.sol#373) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/detector-documentation#uninitialized-local-variables

TroveManagerRedemptions._redemCollateralFromTrove(DataTypes.ContractsCache,address,uint256,uint256,address,uint256) (src/TroveManagerRedemptions.sol#121-238) ignores return value by contractsCache.troveMan
Redemptions.sol#210)
Reference: https://github.com/crytic/slither/wiki/detector-documentation#unused-return

```

- As a result of the tests carried out with the Slither tool, some results were obtained and reviewed by Halborn. Based on the re-sults 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/45ddfe8e-4b94-441b-b69e-2b7d87c34cfb

Line	SWC Title	Severity	Short Description
3	(SWC-108) FloatingPragma	Low	A floating pragma is set.
22	(SWC-123) RequirementViolation	Low	Requirement violation.
124	(SWC-123) RequirementViolation	Low	Requirement violation.

Report for src/BorrowerOperations.sol
https://dashboard.mythx.io/#/console/analyses/c4b0223a-11fc-461f-9c5d-ef85e1a25d8a

Line	SWC Title	Severity	Short Description
3	(SWC-108) FloatingPragma	Low	A floating pragma is set.
31	(SWC-108) StateVariableDefaultVisibility	Low	State variable visibility is not set.
33	(SWC-108) StateVariableDefaultVisibility	Low	State variable visibility is not set.

Report for src/CollateralManager.sol
https://dashboard.mythx.io/#/console/analyses/e264e5a0-6d27-498f-8b74-63b78aa74b4a

Line	SWC Title	Severity	Short Description
2	(SWC-108) FloatingPragma	Low	A floating pragma is set.

Report for src/PriceFeed.sol
https://dashboard.mythx.io/#/console/analyses/52c8583c-cd7d-4d8b-bf4b-c12d68b72e28

Line	SWC Title	Severity	Short Description
3	(SWC-108) FloatingPragma	Low	A floating pragma is set.
31	(SWC-108) StateVariableDefaultVisibility	Low	State variable visibility is not set.
32	(SWC-108) StateVariableDefaultVisibility	Low	State variable visibility is not set.

Report for src/StabilityPool.sol
https://dashboard.mythx.io/#/console/analyses/2ec42f1b-9f4b-48f1-9bcl-172448b84dd0

Line	SWC Title	Severity	Short Description
3	(SWC-108) FloatingPragma	Low	A floating pragma is set.

Report for src/TroveManagerLiquidations.sol
https://dashboard.mythx.io/#/console/analyses/826a7e21-222d-4424-983a-15f24244c7b7

Line	SWC Title	Severity	Short Description
3	(SWC-108) FloatingPragma	Low	A floating pragma is set.
27	(SWC-108) StateVariableDefaultVisibility	Low	State variable visibility is not set.

Report for src/TroveManagerRedemptions.sol
https://dashboard.mythx.io/#/console/analyses/8fecc8dc-3434-4e26-8f5e-3edc344fb26b

Line	SWC Title	Severity	Short Description
3	(SWC-108) FloatingPragma	Low	A floating pragma is set.
27	(SWC-108) StateVariableDefaultVisibility	Low	State variable visibility is not set.

- No major issues found by Mythx.



THANK YOU FOR CHOOSING

// HALBORN

