



Lybra Finance - V2

Smart Contract Security Assessment

Prepared by: **Halborn**

Date of Engagement: **July 26th, 2023 - August 21st, 2023**

Visit: **Halborn.com**

| | |
|--|----|
| DOCUMENT REVISION HISTORY | 6 |
| CONTACTS | 6 |
| 1 EXECUTIVE OVERVIEW | 7 |
| 1.1 INTRODUCTION | 8 |
| 1.2 ASSESSMENT SUMMARY | 9 |
| 1.3 TEST APPROACH & METHODOLOGY | 10 |
| 2 RISK METHODOLOGY | 11 |
| 2.1 EXPLOITABILITY | 12 |
| 2.2 IMPACT | 13 |
| 2.3 SEVERITY COEFFICIENT | 15 |
| 2.4 SCOPE | 17 |
| 3 ASSESSMENT SUMMARY & FINDINGS OVERVIEW | 18 |
| 4 FINDINGS & TECH DETAILS | 20 |
| 4.1 (HAL-01) CURVEPOOL.EXCHANGEUNDERLYING() CALL COULD CONSTANTLY REVERT IN THE DISTRIBUTEREWARDS FUNCTION - HIGH(7.5) | 22 |
| Description | 22 |
| BVSS | 23 |
| Recommendation | 23 |
| Remediation Plan | 23 |
| 4.2 (HAL-02) PEUSD.CONVERTTOPEUSD() CALL COULD CAUSE A DENIAL OF SERVICE IN THE DISTRIBUTEREWARDS FUNCTION - HIGH(7.5) | 24 |
| Description | 24 |
| BVSS | 26 |
| Recommendation | 26 |
| Remediation Plan | 26 |

| | |
|--|----|
| 4.3 (HAL-03) LYBRACONFIGURATOR CONTRACT ASSUMES THAT THE USDC PRICE WILL BE ALWAYS PEGGED TO 1 USD - MEDIUM(6.2) | 27 |
| Description | 27 |
| BVSS | 28 |
| Recommendation | 28 |
| Remediation Plan | 28 |
| 4.4 (HAL-04) LIQUIDATION CALLS CAN BE FRONTRUN STEALING THE REWARD2KEEPER FEE - MEDIUM(6.2) | 29 |
| Description | 29 |
| BVSS | 32 |
| Recommendation | 32 |
| Remediation Plan | 32 |
| 4.5 (HAL-05) LYBRAGOVERNANCE PROPOSAL CREATION CAN BE DOS'ED - MEDIUM(6.2) | 33 |
| Description | 33 |
| References | 34 |
| BVSS | 34 |
| Recommendation | 34 |
| Remediation Plan | 34 |
| 4.6 (HAL-06) CENTRALIZATION ISSUE: LYBRAGOVERNANCE DEPLOYER HAS DAO AND GOV ROLES - MEDIUM(6.2) | 35 |
| Description | 35 |
| BVSS | 36 |
| Recommendation | 36 |
| Remediation Plan | 36 |
| 4.7 (HAL-07) MISSING STALENESS CHECKS IN THE CHAINLINK.LATESTROUND() CALLS - MEDIUM(5.0) | 37 |
| Description | 37 |

| | |
|--|-----------|
| Code location | 37 |
| BVSS | 38 |
| Recommendation | 38 |
| Remediation Plan | 38 |
| 4.8 (HAL-08) MISSING BURNENABLED MODIFIER IN THE PEUSDMAIN-NET.CONVERTTOEUSD() FUNCTION - LOW(3.1) | 39 |
| Description | 39 |
| BVSS | 40 |
| Recommendation | 40 |
| Remediation Plan | 40 |
| 4.9 (HAL-09) MISSING MINTENABLED MODIFIER IN THE PEUSDMAIN-NET.CONVERTTOPEUSD() FUNCTION - LOW(3.1) | 41 |
| Description | 41 |
| BVSS | 42 |
| Recommendation | 42 |
| Remediation Plan | 42 |
| 4.10 (HAL-10) LYBRARETHVAULT.DEPOSITETHERTOMINT() FUNCTION CALLS MAY REVERT - LOW(2.5) | 43 |
| Description | 43 |
| BVSS | 44 |
| Recommendation | 44 |
| Remediation Plan | 44 |
| 4.11 (HAL-11) HARDCODED DSTCHAINID IN STAKINGREWARDSONARBI CONTRACT - LOW(2.5) | 45 |
| Description | 45 |
| BVSS | 45 |
| Recommendation | 46 |

| | |
|---|----|
| Remediation Plan | 46 |
| 4.12 (HAL-12) COLLATERAL MINIMUM DEPOSIT AMOUNT CAN BE BYPASSED - LOW(2.5) | 47 |
| Description | 47 |
| BVSS | 48 |
| Recommendation | 48 |
| Remediation Plan | 48 |
| 4.13 (HAL-13) LYBRAGOVERNANCE HARDCODED 4 PERCENT QUORUM - LOW(2.5) | 49 |
| Description | 49 |
| BVSS | 51 |
| Recommendation | 51 |
| Remediation Plan | 51 |
| 4.14 (HAL-14) EUSDPRICEFEED HARDCODED DECIMALS - INFORMATIONAL(0.0) | 52 |
| Description | 52 |
| BVSS | 53 |
| Recommendation | 53 |
| Remediation Plan | 53 |
| 4.15 (HAL-15) STATE VARIABLES MISSING IMMUTABLE MODIFIER - INFORMATIONAL(0.0) | 54 |
| Description | 54 |
| BVSS | 54 |
| Recommendation | 54 |
| Remediation Plan | 55 |

| | |
|--|----|
| 4.16 (HAL-16) LACK OF A DOUBLE-STEP TRANSFER OWNERSHIP PATTERN IN MULTIPLE CONTRACTS - INFORMATIONAL(0.0) | 56 |
| Description | 56 |
| BVSS | 56 |
| Recommendation | 56 |
| Remediation Plan | 58 |
| 4.17 (HAL-17) FLOATING PRAGMA - INFORMATIONAL(0.0) | 59 |
| Description | 59 |
| Code Location | 59 |
| BVSS | 59 |
| Recommendation | 59 |
| Remediation Plan | 59 |
| 5 RECOMMENDATIONS OVERVIEW | 60 |
| 6 AUTOMATED TESTING | 62 |
| 6.1 STATIC ANALYSIS REPORT | 63 |
| Description | 63 |
| Slither results | 63 |
| 6.2 AUTOMATED SECURITY SCAN | 81 |
| Description | 81 |
| MythX results | 81 |

DOCUMENT REVISION HISTORY

| VERSION | MODIFICATION | DATE | AUTHOR |
|---------|-------------------------|------------|-----------------|
| 0.1 | Document Creation | 07/26/2023 | Roberto Reigada |
| 0.2 | Document Updates | 08/21/2023 | Roberto Reigada |
| 0.3 | Draft Review | 08/21/2023 | Gokberk Gulgund |
| 0.4 | Draft Review | 08/21/2023 | Gabi Urrutia |
| 1.0 | Remediation Plan | 08/25/2023 | Roberto Reigada |
| 1.1 | Remediation Plan Review | 08/28/2023 | 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 Gulgund | Halborn | Gokberk.Gulgund@halborn.com |
| Roberto Reigada | Halborn | Roberto.Reigada@halborn.com |

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Lybra Finance is a decentralized protocol that allows users to mint eUSD, an interest-bearing stablecoin using ETH as collateral.

Lybra Finance is introducing peUSD, an Omnicoin DeFi utility version of eUSD. This token provides additional utility and versatility.

In this V2 update, the types of Liquid Staking Tokens (LSTs) that can be used as collateral are diversified. Alongside existing options, LybraV2 will be accepting rETH and WBETH for eUSD and peUSD minting from the onset. This expanded offering provides users with more flexibility in their interactions with the protocol.

One of our most distinctive features in V2 is the ability to convert eUSD to peUSD without sacrificing the accruing interest on the initial eUSD. Further, enhancing the protocol's resilience, the amount of converted eUSD can be availed for flash loans, aiding in efficient liquidation and contributing to the stability of our protocol's fund.

Lybra Finance engaged Halborn to conduct a security assessment on their smart contracts beginning on July 26th, 2023 and ending on August 21st, 2023. The security assessment was scoped to the smart contracts provided in the following GitHub repository:

- [LybraFinance/LybraV2](https://github.com/LybraFinance/LybraV2).

1.2 ASSESSMENT SUMMARY

The team at Halborn was provided four weeks for the engagement and assigned a full-time security engineer to verify the security of the smart contracts. 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 mostly addressed by the Lybra Finance team.

1.3 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 code and can quickly identify items that do not follow the security best practices. The following phases and associated tools were used during 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 hot-spots 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 |

2.4 SCOPE

1. IN-SCOPE TREE & COMMIT :

The security assessment was scoped to the following smart contracts:

GitHub repository: [LybraFinance/LybraV2](#)

Commit ID: [90285107de8a6754954c303cd69d97b5fdb4e248](#)

Fixed Commit ID: [77e8bc3664fb1b195fd718c2ce1d49af8530f981](#)

Smart contracts in scope:

- LybraConfigurator.sol
- LybraProxy.sol
- LybraProxyAdmin.sol
- AdminTimelock.sol
- GovernanceTimelock.sol
- LybraGovernance.sol
- EUSDMiningIncentives.sol
- ProtocolRewardsPool.sol
- esLBRBoost.sol
- stakerewardPoolOnArbi.sol
- stakerewardV2pool.sol
- LybraRETHVault.sol
- LybraStETHVault.sol
- LybraWbETHVault.sol
- LybraWstETHVault.sol
- LybraEUSDVaultBase.sol
- LybraPeUSDVaultBase.sol
- EUSD.sol
- LBR.sol
- LBRL2.sol
- PeUSD.sol
- PeUSDMainnet.sol
- esLBR.sol
- LBRMinerFromL2.sol

3. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

| CRITICAL | HIGH | MEDIUM | LOW | INFORMATIONAL |
|----------|------|--------|-----|---------------|
| 0 | 2 | 5 | 6 | 4 |

EXECUTIVE OVERVIEW

| SECURITY ANALYSIS | RISK LEVEL | REMEDIATION DATE |
|--|--------------|---------------------|
| (HAL-01) CURVEPOOL.EXCHANGEUNDERLYING() CALL COULD CONSTANTLY REVERT IN THE DISTRIBUTEREWARDS FUNCTION | High (7.5) | SOLVED - 08/24/2023 |
| (HAL-02) PEUSD.CONVERTTOPEUSD() CALL COULD CAUSE A DENIAL OF SERVICE IN THE DISTRIBUTEREWARDS FUNCTION | High (7.5) | SOLVED - 08/24/2023 |
| (HAL-03) LYBRACONFIGURATOR CONTRACT ASSUMES THAT THE USDC PRICE WILL BE ALWAYS PEGGED TO 1 USD | Medium (6.2) | RISK ACCEPTED |
| (HAL-04) LIQUIDATION CALLS CAN BE FRONTRUN STEALING THE REWARD2KEEPER FEE | Medium (6.2) | SOLVED - 08/24/2023 |
| (HAL-05) LYBRAGOVERNANCE PROPOSAL CREATION CAN BE DOS'ED | Medium (6.2) | SOLVED - 08/24/2023 |
| (HAL-06) CENTRALIZATION ISSUE: LYBRAGOVERNANCE DEPLOYER HAS DAO AND GOV ROLES | Medium (6.2) | SOLVED - 08/28/2023 |
| (HAL-07) MISSING STALENESS CHECKS IN THE CHAINLINK.LATESTROUNDATA() CALLS | Medium (5.0) | RISK ACCEPTED |
| (HAL-08) MISSING BURNENABLED MODIFIER IN THE PEUSDMAINNET.CONVERTTOEUSD() FUNCTION | Low (3.1) | RISK ACCEPTED |
| (HAL-09) MISSING MINTENABLED MODIFIER IN THE PEUSDMAINNET.CONVERTTOPEUSD() FUNCTION | Low (3.1) | RISK ACCEPTED |
| (HAL-10) LYBRAETHVAULT.DEPOSITETHERTOMINT() FUNCTION CALLS MAY REVERT | Low (2.5) | RISK ACCEPTED |
| (HAL-11) HARDCODED DSTCHAINID IN STAKINGREWARDSONARBI CONTRACT | Low (2.5) | RISK ACCEPTED |
| (HAL-12) COLLATERAL MINIMUM DEPOSIT AMOUNT CAN BE BYPASSED | Low (2.5) | RISK ACCEPTED |

EXECUTIVE OVERVIEW

| | | |
|--|---------------------|---------------------|
| (HAL-13) LYBRAGOVERNANCE HARDCODED 4 PERCENT QUORUM | Low (2.5) | SOLVED - 08/24/2023 |
| (HAL-14) EUSDPRISEFEED HARDCODED DECIMALS | Informational (0.0) | ACKNOWLEDGED |
| (HAL-15) STATE VARIABLES MISSING IMMUTABLE MODIFIER | Informational (0.0) | ACKNOWLEDGED |
| (HAL-16) LACK OF A DOUBLE-STEP TRANSFEROWNERSHIP PATTERN IN MULTIPLE CONTRACTS | Informational (0.0) | ACKNOWLEDGED |
| (HAL-17) FLOATING PRAGMA | Informational (0.0) | ACKNOWLEDGED |



FINDINGS & TECH DETAILS



4.1 (HAL-01) CURVEPOOL.EXCHANGEUNDERLYING() CALL COULD CONSTANTLY REVERT IN THE DISTRIBUTEREWARDS FUNCTION - HIGH (7.5)

Commit IDs affected:

- 90285107de8a6754954c303cd69d97b5fdb4e248

Description:

In the `LybraConfigurator` contract, the function `distributeRewards()` internally calls the `curvePool.exchange_underlying()` function to convert eUSD into peUSD:

Listing 1: LybraConfigurator.sol (Line 323)

```
316 function distributeRewards() external {
317     uint256 balance = EUSD.balanceOf(address(this));
318     if (balance >= 1e21) {
319         if(premiumTradingEnabled){
320             (, int price, , ) = eUSDPriceFeed.latestRoundData();
321             if(price >= 100_500_000){
322                 EUSD.approve(address(curvePool), balance);
323                 uint256 amount = curvePool.exchange_underlying(0,
324 ↳ 2, balance, balance * uint(price) * 998 / 1e23);
325                 IERC20(stableToken).safeTransfer(address(
326 ↳ lybraProtocolRewardsPool), amount);
327                 lybraProtocolRewardsPool.notifyRewardAmount(amount
328 ↳ , 1);
329                 emit SendProtocolRewards(stableToken, amount,
330 ↳ block.timestamp);
331             }
332             uint256 peUSDBalance = peUSD.balanceOf(address(this));
```

```
333     if(peUSDBalance >= 1e21) {  
334         peUSD.transfer(address(lybraProtocolRewardsPool),  
↳ peUSDBalance);  
335         lybraProtocolRewardsPool.notifyRewardAmount(peUSDBalance,  
↳ 0);  
336         emit SendProtocolRewards(address(peUSD), peUSDBalance,  
↳ block.timestamp);  
337     }  
338 }
```

This call makes 2 assumptions wrongly:

1. Assumes that the USDC price will always be pegged to 1 USD.
2. Sets a maximum slippage of 0.2%.

As most of the Curve Pools have by default a 0.015% total fee (DAO fee + protocol fee) and USDC price may not be exactly pegged to 1 USD it is entirely possible that the `curvePool.exchange_underlying()` call reverts due to high slippage. In that case, any `distributeRewards()` call would revert leaving the `ProtocolRewards` contract without rewards.

BVSS:

A0:A/AC:L/AX:L/C:N/I:H/A:N/D:N/Y:N/R:N/S:U (7.5)

Recommendation:

It is recommended to calculate the price of USDC with high precision using an external oracle. It is also recommended to implement a setter to update the maximum slippage tolerance in the `curvePool.exchange_underlying()` call.

Remediation Plan:

SOLVED: The Lybra Finance team solved the issue by increasing the maximum slippage to 0.5%.

Commit ID : [8c4ec31df7978acd25bd8eb95a22dc056a9be553](#).

4.2 (HAL-02) PEUSD.CONVERTTOPEUSD() CALL COULD CAUSE A DENIAL OF SERVICE IN THE DISTRIBUTEREWARDS FUNCTION - **HIGH (7.5)**

Commit IDs affected:

- 90285107de8a6754954c303cd69d97b5fdb4e248

Description:

In the `LybraConfigurator` contract, the function `distributeRewards()` internally calls `PeUSDMainnet.convertToPeUSD()` function to exchange eUSD for USDC:

Listing 2: LybraConfigurator.sol (Line 329)

```

316 function distributeRewards() external {
317     uint256 balance = EUSD.balanceOf(address(this));
318     if (balance >= 1e21) {
319         if(premiumTradingEnabled){
320             (, int price, , ) = eUSDPriceFeed.latestRoundData();
321             if(price >= 100_500_000){
322                 EUSD.approve(address(curvePool), balance);
323                 uint256 amount = curvePool.exchange_underlying(0,
324                     2, balance, balance * uint(price) * 998 / 1e23);
325                 IERC20(stableToken).safeTransfer(address(
326                     lybraProtocolRewardsPool), amount);
327                 lybraProtocolRewardsPool.notifyRewardAmount(amount
328                     , 1);
329                 emit SendProtocolRewards(stableToken, amount,
330                     block.timestamp);
331             }
332             peUSD.convertToPeUSD(address(this), balance);
333         }
334         uint256 peUSDBalance = peUSD.balanceOf(address(this));
335         if(peUSDBalance >= 1e21) {

```

```

334         peUSD.transfer(address(lybraProtocolRewardsPool),
335             peUSDBalance);
335         lybraProtocolRewardsPool.notifyRewardAmount(peUSDBalance,
336             0);
336         emit SendProtocolRewards(address(peUSD), peUSDBalance,
337             block.timestamp);
337     }
338 }
```

Although, this call could always revert as there is a limit of peUSD that can be minted by using eUSD which is given by the `LybraConfigurator.getEUSDMaxLocked()` function:

Listing 3: PeUSDMainnet.sol (Line 78)

```

70 /**
71  * @notice Allows the user to deposit eUSD and mint peUSD tokens.
72  * @param user The address of the user who wants to deposit eUSD
73  *             and mint peUSD. It can only be the contract itself or the msg.
74  *             sender.
75  * @param eusdAmount The amount of eUSD to deposit and mint peUSD
76  *                   tokens.
77 */
78 function convertToPeUSD(address user, uint256 eusdAmount) public {
79     require(_msgSender() == user || _msgSender() == address(this),
80             "MDM");
81     require(eusdAmount != 0, "ZA");
82     require(EUSD.balanceOf(address(this)) + eusdAmount <=
83             configurator.getEUSDMaxLocked(), "ESL");
84     bool success = EUSD.transferFrom(user, address(this),
85             eusdAmount);
86     require(success, "TF");
87     userConvertInfo[user].depositedEUSDShares += EUSD.
88     getSharesByMintedEUSD(eusdAmount);
89     userConvertInfo[user].mintedPeUSD += eusdAmount;
90     _mint(user, eusdAmount);
91     emit ConvertToPeUSD(msg.sender, eusdAmount, eusdAmount, block.
92             timestamp);
93 }
```

If this limit is reached and the price of eUSD is lower than 1,005 USD,

any `distributeRewards()` call would revert, leaving the `ProtocolRewards` contract without rewards.

BVSS:

A0:A/AC:L/AX:L/C:N/I:H/A:N/D:N/Y:N/R:N/S:U (7.5)

Recommendation:

It is recommended to add a check to the `convertToPeUSD()` function that if it is used for dividend conversion, it will not be subject to the `getEUSDMaxLocked()` limitation.

Remediation Plan:

SOLVED: The `Lybra Finance` team solved the issue by implementing the recommended solution.

Commit ID : `d050e08072633dccc69155a36901c9434dfdb58`.

4.3 (HAL-03) LYBRACONFIGULATOR CONTRACT ASSUMES THAT THE USDC PRICE WILL BE ALWAYS PEGGED TO 1 USD - MEDIUM (6.2)

Commit IDs affected:

- [90285107de8a6754954c303cd69d97b5fdb4e248](#)

Description:

The `LybraConfigurator.distributeRewards()` function, under certain conditions, exchanges eUSD for USDC using Curve.

This function assumes that the USDC price will always remain pegged to 1 USD. This assumption introduces several concerns:

1. **Peg Deviation**: Even though USDC is a stablecoin designed to maintain a peg to the US dollar, there's no guarantee it will always be exactly 1 dollar. Market dynamics, liquidity concerns, or unforeseen events can cause temporary or even long-term deviations.
2. **Contract Logic Vulnerability**: If the contract's logic heavily relies on the USDC being always pegged at 1 dollar, any deviation can result in incorrect calculations, leading to potential losses, denial of service or other undesired outcomes.
3. **Reduced Resilience to Black Swan Events**: In the case of unprecedented events in the crypto space, USDC, like any other asset, can face challenges. Banking issues, regulatory crackdowns, or problems with Circle (the issuer of USDC) might affect the peg.
4. **Arbitrage Opportunities**: If USDC diverges from its \$1 peg and the contract assumes it's always \$1, it can create arbitrage opportunities. Malicious actors might exploit these to drain funds or gain unfair advantages.

For all these reasons, it is always recommended to use an external oracle

to retrieve and validate the price of USDC with high precision, preventing all the risks mentioned above.

BVSS:

A0:A/AC:L/AX:L/C:N/I:M/A:N/D:N/Y:N/R:N/S:C (6.2)

Recommendation:

It is recommended to use an external oracle to retrieve and validate the price of USDC with high precision.

Remediation Plan:

RISK ACCEPTED: The Lybra Finance team accepted the risk of this finding.

4.4 (HAL-04) LIQUIDATION CALLS CAN BE FRONTRUN STEALING THE REWARD2KEEPER FEE - MEDIUM (6.2)

Commit IDs affected:

- 90285107de8a6754954c303cd69d97b5fdb4e248

Description:

In the `LybraPeUSDVaultBase` contract, the function `liquidation()` liquidates borrowers whose collateral ratio is below `badCollateralRatio`, using peUSD provided by the Liquidation Provider:

Listing 4: LybraPeUSDVaultBase.sol (Line 132)

```

114 /**
115 * @notice Keeper liquidates borrowers whose collateral ratio is
116 * below badCollateralRatio, using peUSD provided by Liquidation
117 * Provider.
118 *
119 * Requirements:
120 * - onBehalfOf Collateral Ratio should be below
121 * badCollateralRatio
122 * - assetAmount should be less than 50% of collateral
123 * - provider should authorize Lybra to utilize peUSD
124 * @dev After liquidation, borrower's debt is reduced by
125 * assetAmount * assetPrice, providers and keepers can receive up to
126 * an additional 10% liquidation reward.
127 */
128 function liquidation(address provider, address onBehalfOf, uint256
129 assetAmount) external virtual {
130     uint256 assetPrice = getAssetPrice();
131     uint256 onBehalfOfCollateralRatio = (depositedAsset[onBehalfOf
132     ] * assetPrice * 100) / getBorrowedOf(onBehalfOf);
133     require(onBehalfOfCollateralRatio < configurator.
134     getBadCollateralRatio(address(this)), "Borrowers collateral ratio
135     should below badCollateralRatio");
136
137     require(assetAmount * 2 <= depositedAsset[onBehalfOf], "a max
138     of 50% collateral can be liquidated");

```

```

129     require(PeUSD.allowance(provider, address(this)) != 0 || msg.
130     sender == provider, "provider should authorize to provide
131     liquidation peUSD");
132     uint256 peusdAmount = (assetAmount * assetPrice) / 1e18;
133
132     _repay(provider, onBehalfOf, peusdAmount);
133     uint256 reducedAsset = assetAmount;
134     if(onBehalfOfCollateralRatio > 1e20 &&
135     onBehalfOfCollateralRatio < 11e19) {
135         reducedAsset = assetAmount * onBehalfOfCollateralRatio / 1
136         e20;
136     }
137     if(onBehalfOfCollateralRatio >= 11e19) {
138         reducedAsset = assetAmount * 11 / 10;
139     }
140     depositedAsset[onBehalfOf] -= reducedAsset;
141     uint256 reward2keeper;
142     uint256 keeperRatio = configurator.vaultKeeperRatio(address(
143     this));
143     if (msg.sender != provider && onBehalfOfCollateralRatio >= 1
144     e20 + keeperRatio * 1e18) {
144         reward2keeper = assetAmount * keeperRatio / 100;
145         collateralAsset.safeTransfer(msg.sender, reward2keeper);
146     }
147     collateralAsset.safeTransfer(provider, reducedAsset -
148     reward2keeper);
148     emit LiquidationRecord(provider, msg.sender, onBehalfOf,
149     peusdAmount, reducedAsset, reward2keeper, false, block.timestamp);
149 }
```

The `liquidation()` function calls internally the `_repay()` function:

Listing 5: LyraPeUSDVaultBase.sol (Lines 204,210)

```

192 /**
193 * @notice Burn _provideramount peUSD to payback minted peUSD for
194 * _onBehalfOf.
194 *
195 * @dev Refresh LBR reward before reducing providers debt. Refresh
195 * Lybra generated service fee before reducing totalPeUSDCirculation
195 * .
196 */
197 function _repay(address _provider, address _onBehalfOf, uint256
```

```

↳ _amount) internal virtual {
198     configurator.refreshMintReward(_onBehalfOf);
199     _updateFee(_onBehalfOf);
200     uint256 totalFee = feeStored[_onBehalfOf];
201     uint256 amount = borrowed[_onBehalfOf] + totalFee >= _amount ?
↳ _amount : borrowed[_onBehalfOf] + totalFee;
202     if(amount > totalFee) {
203         feeStored[_onBehalfOf] = 0;
204         PeUSD.transferFrom(_provider, address(configurator),
↳ totalFee);
205         PeUSD.burn(_provider, amount - totalFee);
206         borrowed[_onBehalfOf] -= amount - totalFee;
207         poolTotalCirculation -= amount - totalFee;
208     } else {
209         feeStored[_onBehalfOf] = totalFee - amount;
210         PeUSD.transferFrom(_provider, address(configurator),
↳ amount);
211     }
212     try configurator.distributeRewards() {} catch {}
213     emit Burn(_provider, _onBehalfOf, amount, block.timestamp);
214 }
```

In order to call the `liquidation` function, the provider must first approve the Vault contract. Based on this, the normal flow would be:

1. Provider PeUSD approves the Vault.
2. Provider calls `Vault.liquidation(<Provider address>, <user liquidated address>, <liquidated amount>)`
3. Provider receives `collateralAsset` up to a 10% discount.
4. Provider receives `reward2keeper` fee.

Although, with the current implementation that allows setting a `provider` address different from `msg.sender`, the following attack vector would be possible:

1. Provider PeUSD approves the Vault.
2. Provider calls `Vault.liquidation(<Provider address>, <user liquidated address>, <liquidated amount>)`
3. The liquidation call is frontrun by a MEV bot stealing the `reward2keeper` fee.
4. Provider receives `collateralAsset` up to a 10% discount.
5. Provider transaction reverts as it was frontrun.

Note that reward2keeper can be up to 5% as:

```
Listing 6: LybraConfigurator.sol (Line 257)

251 /**
252 * @notice Set the reward ratio for the liquidator after
253 * @param pool The address of the pool to set the reward ratio for
254 * @param newRatio The new reward ratio to set, limited to a
255 * maximum of 5%.
256 */
257 function setKeeperRatio(address pool,uint256 newRatio) external
258     checkRole(TIMELOCK) {
259     require(newRatio <= 5, "Max Keeper reward is 5%");
260     vaultKeeperRatio[pool] = newRatio;
261     emit KeeperRatioChanged(pool, newRatio);
262 }
```

BVSS:

A0:A/AC:L/AX:L/C:N/I:M/A:N/D:N/Y:M/R:N/S:U (6.2)

Recommendation:

It is recommended to remove the `address provider` parameter from the `liquidation()` function and use always `msg.sender` as the `provider`.

Remediation Plan:

SOLVED: The `Lybra Finance` team solved the issue. If users decide that others should not use their funds for liquidation they have the option to not authorize an allowance to the storage contract.

Commit ID : d050e08072633dccc69155a36901c9434dfdb58.

4.5 (HAL-05) LYBRAGOVERNANCE PROPOSAL CREATION CAN BE DOS'ED - MEDIUM (6.2)

Commit IDs affected:

- 90285107de8a6754954c303cd69d97b5fdb4e248

Description:

The contract `LybraGovernance` inherits from `GovernorTimelockControl` contract which inherits itself from the `Governor` contract.

The version 4.9.0 of the `Governor` contract is vulnerable to an issue whereby front running the creation of a proposal, an attacker can become the proposer and gain the ability to cancel it. The attacker can do this repeatedly to try to prevent a proposal from being proposed at all.

Moreover, LybraV2 project dependencies are defined in the `package.json` file:

Listing 7: package.json (Line 14)

As the `openzeppelin/contracts` version used is `^4.8.3`, upon installation of the dependencies it is entirely possible to install a vulnerable version, making the `LybraGovernance` vulnerable to this exploit.

References:

<https://github.com/OpenZeppelin/openzeppelin-contracts/security/advisories/GHSA-5h3x-9wvq-w4m2>
<https://www.coinspect.com/openzeppelin-governor-dos/>

BVSS:

A0:A/AC:L/AX:L/C:N/I:N/A:M/D:N/Y:N/R:N/S:C (6.2)

Recommendation:

It is recommended to enforce in the `package.json` file to use the `openzeppelin/contracts` version `4.9.1` where this bug is solved.

Remediation Plan:

SOLVED: The `Lybra Finance` team solved the issue by implementing the recommended solution.

Commit ID : `d050e08072633dccc69155a36901c9434dfdb58`.

4.6 (HAL-06) CENTRALIZATION ISSUE: LYBRA GOVERNANCE DEPLOYER HAS DAO AND GOV ROLES - MEDIUM (6.2)

Commit IDs affected:

- 90285107de8a6754954c303cd69d97b5fdb4e248

Description:

In the `GovernanceTimelock` contract, the deployer is directly assigned the DAO and GOV roles:

Listing 8: GovernanceTimelock.sol (Lines 19,20)

```

14 constructor(uint256 minDelay, address[] memory proposers, address
15   [] memory executors, address admin) TimelockController(minDelay,
16   proposers, executors, admin) {
17     _setRoleAdmin(DAO, GOV);
18     _setRoleAdmin(TIMELOCK, GOV);
19     _setRoleAdmin(ADMIN, GOV);
20     _grantRole(DAO, address(this));
21     _grantRole(DAO, msg.sender);
22     _grantRole(GOV, msg.sender);
23 }
```

The DAO role and hence the `GovernanceTimelock` contract's deployer has permissions to do any type of update in the `LybraConfigurator` contract. This totally defeats the purpose of a DAO, as a single address has permissions to perform any change in the protocol.

This presents several concerns:

1. **Centralization of Power:** The essence of a DAO is to decentralize decision-making and reduce single points of failure or control. By assigning the DAO role to the contract's deployer, a centralized entity gets the power to make significant protocol changes, which contradicts the principles of decentralization.

2. **Attack Surface Increase:** Granting both DAO and GOV roles to a single entity, especially at deployment, amplifies the attack surface. If the deployer's private key is compromised, an attacker could make arbitrary changes to the protocol.
3. **Loss of Credibility:** One of the draws for many users towards DAOs is trust in decentralized governance and decision-making. If users or investors discover that a single entity has the capability to alter the protocol unilaterally, it could lead to a loss of faith in the project, affecting adoption and token value.
4. **Potential for Malicious or Accidental Actions:** With such permissions, even a well-intentioned deployer might inadvertently introduce bugs or make changes that are not in the best interest of the community. This poses both security and trust risks.

BVSS:

A0:A/AC:L/AX:L/C:N/I:M/A:N/D:N/Y:N/R:N/S:C (6.2)

Recommendation:

It is recommended to not assign the DAO and GOV roles to the GovernanceTimelock contract's deployer.

Remediation Plan:

SOLVED: The Lybra Finance team solved the issue by implementing the recommended solution.

Commit ID : 77e8bc3664fb1b195fd718c2ce1d49af8530f981.

4.7 (HAL-07) MISSING STALENESS CHECKS IN THE CHAINLINK.LATESTROUNDATA() CALLS - MEDIUM (5.0)

Commit IDs affected:

- 90285107de8a6754954c303cd69d97b5fdb4e248

Description:

In multiple contracts, the `latestRoundData()` function of Chainlink price feeds is called to retrieve the price of different assets.

Although, these calls are performed without any kind of staleness checks nor price validation. There are no checks on `roundID` or `timeStamp`. If there is a problem with Chainlink starting a new round and finding consensus on the new value for the oracle (e.g. Chainlink nodes abandoning the oracle, chain congestion, vulnerability/attacks on the Chainlink system) consumers of these contracts may continue using obsolete data.

On the other hand, according to Chainlink's documentation, `latestRoundData()` does not raise an error if no response has been reached, but returns 0, in this case feeding an incorrect price to the contracts.

Code location:

`LybraConfigurator.sol`

- Line 320: `(, int price, , ,)= eUSDPPriceFeed.latestRoundData();`

`EUSDMiningIncentives.sol`

- Line 189: `(, int etherPrice, , ,)= etherPriceFeed.latestRoundData();`
- Line 190: `(, int lbrPrice, , ,)= lbrPriceFeed.latestRoundData();`
- Line 267: `(, int lbrPrice, , ,)= lbrPriceFeed.latestRoundData();`

ProtocolRewardsPool.sol

- Line 151: (, int lbrPrice, , ,)= lbrPriceFeed.latestRoundData();

BVSS:

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

Recommendation:

It is recommended to use Chainlink's `latestRoundData()` function with checks on the return data with proper revert messages if the price is stale or the round is incomplete, for example:

Listing 9: latestRoundData example call (Line 2)

```
1 uint256 private constant GRACE_PERIOD_TIME = 7200; // 2 hours
2 (uint80 roundId, int256 price, uint256 startedAt, uint256
↳ updatedAt, uint80 answeredInRound) = eUSDPPriceFeed.latestRoundData
↳ ();
3 require(answeredInRound >= roundID, "Stale price");
4 require(price > 0, "invalid price");
5 require(block.timestamp <= updatedAt + GRACE_PERIOD_TIME, "Stale
↳ price");
```

If in the case that we are dealing with Price Feeds in Arbitrum, the sequencer state should be validated, ensuring that is up before accepting any price as valid.

Remediation Plan:

RISK ACCEPTED: The Lybra Finance team accepted the risk of this finding.

4.8 (HAL-08) MISSING BURNENABLED MODIFIER IN THE PEUSDMAINNET.CONVERTTOEUSD() FUNCTION - LOW (3.1)

Commit IDs affected:

- [90285107de8a6754954c303cd69d97b5fdb4e248](#)

Description:

In the `PeUSDMainnet` contract, the modifier `burnEnabled` ensures that the burning is not paused in the `LybraConfigurator` contract:

Listing 10: PeUSDMainnet.sol (Line 51)

```
50 modifier burnEnabled() {
51     require(!configurator.vaultBurnPaused(msg.sender), "BPP");
52     _;
53 }
```

Although, the function `convertToEUSD()`, which burns peUSD by calling the `_burn()` internal function is missing this modifier:

Listing 11: PeUSDMainnet.sol (Line 111)

```
104 /**
105 * @dev Allows users to repay peUSD tokens and retrieve eUSD.
106 * @param peusdAmount The amount of peUSD tokens to burn and
107 *   retrieve eUSD. The user's balance of peUSD tokens must be greater
108 *   than or equal to this amount.
109 * Requirements:
110 *   `peusdAmount` must be greater than 0.
111 *   The user's `mintedPeUSD` must be greater than or equal to `peusdAmount`.
112 */
113 function convertToEUSD(uint256 peusdAmount) external {
114     require(peusdAmount <= userConvertInfo[msg.sender].mintedPeUSD
```

```
↳ &&peusdAmount != 0, "PCE");
113     _burn(msg.sender, peusdAmount);
114     uint256 share = (userConvertInfo[msg.sender].
115     depositedEUSDShares * peusdAmount) / userConvertInfo[msg.sender].
116     mintedPeUSD;
117     userConvertInfo[msg.sender].mintedPeUSD -= peusdAmount;
118     userConvertInfo[msg.sender].depositedEUSDShares -= share;
119     EUSD.transferShares(msg.sender, share);
120     emit ConvertToEUSD(msg.sender, peusdAmount, EUSD.
121     getMintedEUSDByShares(share), block.timestamp);
122 }
```

Hence, users would still be able to burn `peUSD` through the usage of this function even if the burning is paused for this vault in the `LybraConfigurator` contract.

BVSS:

A0:A/AC:L/AX:L/C:N/I:L/A:N/D:N/Y:N/R:N/S:C (3.1)

Recommendation:

It is recommended to add the `burnEnabled` modifier to the `PeUSDMainnet.convertToEUSD()` function.

Remediation Plan:

RISK ACCEPTED: The Lybra Finance team accepted the risk of this finding.

4.9 (HAL-09) MISSING MINTENABLED MODIFIER IN THE PEUSDMAINNET.CONVERTTOPEUSD() FUNCTION - LOW (3.1)

Commit IDs affected:

- 90285107de8a6754954c303cd69d97b5fdb4e248

Description:

In the `PeUSDMainnet` contract, the modifier `mintEnabled` ensures that the minting is not paused in the `LybraConfigurator` contract:

Listing 12: PeUSDMainnet.sol (Line 47)

```
46 modifier mintEnabled() {
47     require(!configurator.vaultMintPaused(msg.sender), "MPP");
48     _;
49 }
```

Although, the function `convertToPeUSD()`, which mints peUSD by calling the `_mint()` internal function is missing this modifier:

Listing 13: PeUSDMainnet.sol (Line 75)

```
70 /**
71 * @notice Allows the user to deposit eUSD and mint peUSD tokens.
72 * @param user The address of the user who wants to deposit eUSD
73 *             and mint peUSD. It can only be the contract itself or the msg.
74 *             sender.
75 * @param eusdAmount The amount of eUSD to deposit and mint peUSD
76 *                   tokens.
77 */
78 function convertToPeUSD(address user, uint256 eusdAmount) public {
79     require(_msgSender() == user || _msgSender() == address(this),
80             "MDM");
81     require(eusdAmount != 0, "ZA");
```

```
78     require(EUSD.balanceOf(address(this)) + eusdAmount <=
    ↳ configurator.getEUSDMaxLocked(),"ESL");
79     bool success = EUSD.transferFrom(user, address(this),
    ↳ eusdAmount);
80     require(success, "TF");
81     userConvertInfo[user].depositedEUSDShares += EUSD.
    ↳ getSharesByMintedEUSD(eusdAmount);
82     userConvertInfo[user].mintedPeUSD += eusdAmount;
83     _mint(user, eusdAmount);
84     emit ConvertToPeUSD(msg.sender, eusdAmount, eusdAmount, block.
    ↳ timestamp);
85 }
```

Hence, users would still be able to mint `peUSD` through the usage of this function even if the minting is paused for this vault in the `LybraConfigurator` contract.

BVSS:

A0:A/AC:L/AX:L/C:N/I:L/A:N/D:N/Y:N/R:N/S:C (3.1)

Recommendation:

It is recommended to add the `mintEnabled` modifier to the `PeUSDMainnet.convertToPeUSD()` function.

Remediation Plan:

RISK ACCEPTED: The `Lybra Finance` team accepted the risk of this finding.

4.10 (HAL-10)

LYBRARETHVAULT.DEPOSITETHERTOMINT()

FUNCTION CALLS MAY REVERT - LOW (2.5)

Commit IDs affected:

- 90285107de8a6754954c303cd69d97b5fdb4e248

Description:

In the `LybraRETHVault` contract, the function `depositEtherToMint()` is used to add Ether into the Rocket Pool and then provide the rETH received directly to the `LybraRETHVault` as collateral:

Listing 14: LybraRETHVault.sol (Line 34)

```

31 function depositEtherToMint(uint256 mintAmount) nonReentrant
↳ external payable override {
32     require(msg.value >= 1 ether, "DNL");
33     uint256 preBalance = collateralAsset.balanceOf(address(this));
34     IRocketDepositPool(rocketStorage.getAddress(keccak256(abi.
↳ encodePacked("contract.address", "rocketDepositPool")))).deposit{
↳ value: msg.value}();
35     uint256 balance = collateralAsset.balanceOf(address(this));
36     depositedAsset[msg.sender] += balance - preBalance;
37
38     if (mintAmount > 0) {
39         _mintPeUSD(msg.sender, msg.sender, mintAmount,
↳ getAssetPrice());
40     }
41
42     emit DepositEther(msg.sender, address(collateralAsset), msg.
↳ value, balance - preBalance, block.timestamp);
43 }
```

Although, there is a limit to the amount of Ether that can be in the Rocket pool at any given time, so it is possible that the pool could be

full until Node Operators pull Ether from it and stake that Ether on the Beacon Chain. See [Rocket Pool documentation](#).

Based on this, if the Rocket pool is full, any call to the `LybraRETHVault.depositEtherToMint()` function will revert.

BVSS:

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

Recommendation:

It is recommended to add a require statement in the `LybraRETHVault.depositEtherToMint()` that checks if the Rocket pool is full and reverts if so.

Remediation Plan:

RISK ACCEPTED: The Lybra Finance team accepted the risk of this finding.

4.11 (HAL-11) HARDCODED DSTCHAINID IN STAKINGREWARDSONARBI CONTRACT - LOW (2.5)

Commit IDs affected:

- [90285107de8a6754954c303cd69d97b5fdb4e248](#)

Description:

In the [Layer Zero integration checklist](#) is explicitly mentioned that LayerZero chain identifiers should not be hardcoded and that admin restricted setters should be used instead.

Although, in the `StakingRewardsOnArbi` contract, the `dstChainId` is set in the constructor and cannot be updated later on:

Listing 15: StakingRewardsOnArbi.sol

```
14 uint16 public immutable dstChainId;
```

Listing 16: StakingRewardsOnArbi.sol (Line 46)

```
44 constructor(address _stakingToken, uint16 _dstChainId, address
↳ _lzEndpoint) NonblockingLzApp(_lzEndpoint) {
45     stakingToken = IERC20(_stakingToken);
46     dstChainId = _dstChainId;
47 }
```

BVSS:

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

FINDINGS & TECH DETAILS

Recommendation:

It is recommended to add an `onlyOwner` setter to update the `dstChainId` state variable.

Remediation Plan:

RISK ACCEPTED: The [Lybra Finance](#) team accepted the risk of this finding.

4.12 (HAL-12) COLLATERAL MINIMUM DEPOSIT AMOUNT CAN BE BYPASSED - LOW (2.5)

Commit IDs affected:

- 90285107de8a6754954c303cd69d97b5fdb4e248

Description:

In all the vaults, there is a minimum amount of `1e18` tokens that can be deposited into the protocol:

Listing 17: LyraPeUSDVaultBase.sol (Line 60)

```

51 /**
52 * @notice Deposit staked ETH, update the interest distribution,
53 * can mint peUSD directly
54 * Emits a `DepositAsset` event.
55 *
56 * Requirements:
57 * - `assetAmount` Must be higher than 0.
58 */
59 function depositAssetToMint(uint256 assetAmount, uint256
60     mintAmount) external virtual {
61     require(assetAmount >= 1 ether, "Deposit should not be less
62     than 1 collateral asset.");
63     collateralAsset.safeTransferFrom(msg.sender, address(this),
64     assetAmount);
65
66     depositedAsset[msg.sender] += assetAmount;
67     if (mintAmount > 0) {
68         uint256 assetPrice = getAssetPrice();
69         _mintPeUSD(msg.sender, msg.sender, mintAmount, assetPrice)
70     }
71     emit DepositAsset(msg.sender, address(collateralAsset),
72     assetAmount, block.timestamp);
73 }
```

Although, this requirement can be easily bypassed by:

1. Depositing `1e18` tokens.
2. Withdrawing `1e18 - 1` tokens.

For this reason, it is recommended to add a require statement in the `withdraw()` function that checks that this minimum value is also kept after a withdrawal call.

BVSS:

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

Recommendation:

It is recommended to add a require statement in the `withdraw()` function that checks that the minimum collateral deposited is higher than `1e18`:

Listing 18: LyraPeUSDVaultBase.sol (Line 217)

```
216 function _withdraw(address _provider, address _onBehalfOf, uint256
  ↳ _amount) internal virtual {
217     require(depositedAsset[_provider] - _amount >= 1 ether, "
  ↳ Withdraw amount exceeds deposited amount.");
218     depositedAsset[_provider] -= _amount;
219     collateralAsset.safeTransfer(_onBehalfOf, _amount);
220     if (getBorrowedOf(_provider) > 0) {
221         _checkHealth(_provider, getAssetPrice());
222     }
223     emit WithdrawAsset(_provider, address(collateralAsset),
  ↳ _onBehalfOf, _amount, block.timestamp);
224 }
```

Remediation Plan:

RISK ACCEPTED: The Lybra Finance team accepted the risk of this finding.

4.13 (HAL-13) LYBRAGOVERNANCE HARDCODED 4 PERCENT QUORUM - LOW (2.5)

Commit IDs affected:

- 90285107de8a6754954c303cd69d97b5fdb4e248

Description:

In the `LybraGovernance` contract, the quorum is hardcoded to a 4 percent of the total `esLBR` supply:

Listing 19: LybraGovernance.sol (Line 54)

```

46 /**
47 * @notice module:user-config
48 * @dev Minimum number of cast voted required for a proposal to be
↳ successful.
49 *
50 * NOTE: The `timepoint` parameter corresponds to the snapshot
↳ used for counting vote. This allows to scale the
51 * quorum depending on values such as the totalSupply of a token
↳ at this timepoint (see {ERC20Votes}).
52 */
53 function quorum(uint256 timepoint) public view override returns (
↳ uint256){
54     return esLBR.getPastTotalSupply(timepoint) * 4 / 100;
55 }
```

The appropriate quorum threshold for a DAO varies depending on several factors and the specific nature of the DAO. A 4 percent quorum threshold might be sufficient for some DAOs but too low for others. Some factors to consider when determining an appropriate quorum threshold are:

1. **Size and Distribution of the Membership:** If the DAO has a large and dispersed membership, achieving a higher quorum might be challenging. In such cases, a lower threshold might be appropriate.

Conversely, in a smaller, more engaged DAO, a higher quorum might be achievable and preferred.

2. **Nature of Decisions:** For critical decisions that could have long-term impacts on the DAO or involve significant amounts of assets, a higher quorum might be preferred to ensure adequate representation and legitimacy. For routine or less impactful decisions, a lower quorum might suffice.
3. **Participation Rates:** If the DAO historically has high participation rates in proposals and voting, a 4% quorum might be considered low. If participation rates are historically low, however, setting a high quorum could result in paralysis, where the DAO cannot execute any decisions because the quorum is never met.
4. **Voting Incentives:** DAOs that reward members for voting or penalize non-voting might see higher participation, potentially justifying a higher quorum.
5. **Potential for Manipulation:** A lower quorum can be easier for a small group to manipulate. If just 4% of the members are needed to make decisions, a group that holds slightly more than this can dictate outcomes. This undermines the decentralized nature of a DAO.
6. **Adaptability:** It might be wise to allow the DAO to adjust its quorum requirements over time. If a 4% quorum proves too low (or too high) in practice, the DAO should have a mechanism to adjust it based on real-world experience and feedback from its members.
7. **Comparison with Traditional Entities:** In traditional corporations, quorum requirements for shareholder meetings can vary widely. Some might require most outstanding shares to be present or represented, while others might have lower requirements. It's useful to consider how these traditional entities operate and the reasons for their quorum levels.

While a 4% quorum might be sufficient for some DAOs given their specific context and goals, it's essential to carefully consider the factors mentioned above. Continuous monitoring and feedback from DAO members will also be critical in determining if adjustments to the quorum threshold are necessary over time.

In the current implementation the quorum is simply hardcoded while it should be a parameter that could be adjusted over time by, for example,

the DAO itself.

BVSS:

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

Recommendation:

It is recommended to add a setter to the `LybraGovernance` contract to update the quorum.

Remediation Plan:

SOLVED: The `Lybra Finance` team solved the issue by implementing the recommended solution.

Commit ID : 5f69fca8c13d6ec6974813a90e80a8eab46a31a2.

4.14 (HAL-14) EUSDPRICEFEED HARDCODED DECIMALS - INFORMATIONAL (0.0)

Commit IDs affected:

- [90285107de8a6754954c303cd69d97b5fdb4e248](#)

Description:

In the `LybraConfigurator` contract, the function `distributeRewards()` assumes that the `eUSDPriceFeed` is a Chainlink Price Feed that will always return 8 decimals:

Listing 20: LybraConfigurator.sol (Line 321)

```

307 /**
308  * @notice Distributes rewards to the LybraProtocolRewardsPool
309  * based on the available balance of peUSD and eUSD.
310  * If the balance is greater than 1e21, the distribution process
311  * is triggered.
312  * First, if the eUSD balance is greater than 1,000 and the
313  * premiumTradingEnabled flag is set to true,
314  * and the eUSD/USDC premium exceeds 0.5%, eUSD will be exchanged
315  * for USDC and added to the LybraProtocolRewardsPool.
316  * Otherwise, eUSD will be directly converted to peUSD, and the
317  * entire peUSD balance will be transferred to the
318  * LybraProtocolRewardsPool.
319  * @dev The protocol rewards amount is notified to the
320  * LybraProtocolRewardsPool for proper reward allocation.
321  */
322 function distributeRewards() external {
323     uint256 balance = EUSD.balanceOf(address(this));
324     if (balance >= 1e21) {
325         if(premiumTradingEnabled){
326             (, int price, , , ) = eUSDPriceFeed.latestRoundData();
327             if(price >= 100_500_000){
328                 EUSD.approve(address(curvePool), balance);
329                 uint256 amount = curvePool.exchange_underlying(0,
330                     2, balance, balance * uint(price) * 998 / 1e23);
331             }
332         }
333     }
334 }
```

```
324                 IERC20(stableToken).safeTransfer(address(
325                   lybraProtocolRewardsPool), amount);
326                 lybraProtocolRewardsPool.notifyRewardAmount(amount
327                   , 1);
328             emit SendProtocolRewards(stableToken, amount,
329               block.timestamp);
330         }
331     }
332     uint256 peUSDBalance = peUSD.balanceOf(address(this));
333     if(peUSDBalance >= 1e21) {
334         peUSD.transfer(address(lybraProtocolRewardsPool),
335           peUSDBalance);
336         lybraProtocolRewardsPool.notifyRewardAmount(peUSDBalance,
337           0);
338         emit SendProtocolRewards(address(peUSD), peUSDBalance,
339           block.timestamp);
340     }
341 }
```

Although many Chainlink Price Feeds return prices with 8 decimals, it's not guaranteed that this is always true. Many Chainlink Price Feeds return prices with 18 decimals.

BVSS:

A0:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:N/S:C (0.0)

Recommendation:

It is recommended to always use the `decimals()` function from the Chainlink Price Feed contracts to determine the number of decimals that the feed uses. This ensures that you always work with the correct precision.

Remediation Plan:

ACKNOWLEDGED: The Lybra Finance team acknowledged this finding.

4.15 (HAL-15) STATE VARIABLES MISSING IMMUTABLE MODIFIER - INFORMATIONAL (0.0)

Commit IDs affected:

- 90285107de8a6754954c303cd69d97b5fdb4e248

Description:

The `immutable` keyword was added to Solidity in 0.6.5. State variables can be marked `immutable` which causes them to be read-only, but only assignable in the constructor. The following state variables are missing the `immutable` modifier:

`esLBRBoost.sol`

- Line 13: `IMiningIncentives public miningIncentives;`

`EUSDMiningIncentives.sol`

- Line 32: `address public wETH;`
- Line 55: `AggregatorV3Interface internal etherPriceFeed;`

`LybraGovernance.sol`

- Line 11: `IesLBR public esLBR;`
- Line 12: `IGovernanceTimelock public GovernanceTimelock;`

BVSS:

`A0:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:N/S:U (0.0)`

Recommendation:

It is recommended to add the `immutable` modifier to the state variables mentioned.

FINDINGS & TECH DETAILS

Remediation Plan:

ACKNOWLEDGED: The Lybra Finance team acknowledged this finding.

4.16 (HAL-16) LACK OF A DOUBLE-STEP TRANSFER OWNERSHIP PATTERN IN MULTIPLE CONTRACTS - INFORMATIONAL (0.0)

Commit IDs affected:

- 90285107de8a6754954c303cd69d97b5fdb4e248

Description:

Some contracts are using the [OpenZeppelin's Ownable library](#) in which the transfer of the ownership is done in a single step by simply calling the `transferOwnership()` function.

Although, if the nominated EOA account is not a valid account, it is entirely possible that the owner may accidentally transfer ownership to an uncontrolled account, losing the access to all functions with the `onlyOwner` modifier.

Contracts affected:

- esLBRBoost
- EUSDMiningIncentives
- ProtocolRewardsPool
- LzApp (Used by StakingRewardsOnArbi and `LBRMinerFromL2` contracts).

BVSS:

A0:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:F/S:C (0.0)

Recommendation:

It is recommended to implement a two-step process where the owner nominates an account and the nominated account needs to call an `acceptOwnership()` function for the transfer of the ownership to fully succeed. This ensures

the nominated EOA account is a valid and active account. A good code example could be OpenZeppelin's `Ownable2Step` contract:

Listing 21: Ownable2Step.sol (Lines 52-56)

```
1 // SPDX-License-Identifier: MIT
2 // OpenZeppelin Contracts (last updated v4.8.0) (access/
↳ Ownable2Step.sol)
3
4 pragma solidity ^0.8.0;
5
6 import "./Ownable.sol";
7
8 /**
9  * @dev Contract module which provides access control mechanism,
10 * where
11 * there is an account (an owner) that can be granted exclusive
12 * access to
13 * specific functions.
14 *
15 * By default, the owner account will be the one that deploys the
16 * contract. This
17 * can later be changed with {transferOwnership} and {
18 * acceptOwnership}.
19 *
20 * This module is used through inheritance. It will make available
21 * all functions
22 * from parent (Ownable).
23 */
24
25 /**
26  * @dev Returns the address of the pending owner.
27  */
28 function pendingOwner() public view virtual returns (address)
29 {
30     return _pendingOwner;
31 }
```

```
32     * @dev Starts the ownership transfer of the contract to a new
33     account. Replaces the pending transfer if there is one.
34     */
35     function transferOwnership(address newOwner) public virtual
36     override onlyOwner {
37         _pendingOwner = newOwner;
38         emit OwnershipTransferStarted(owner(), newOwner);
39     }
40     /**
41     * @dev Transfers ownership of the contract to a new account
42     (`newOwner`) and deletes any pending owner.
43     * Internal function without access restriction.
44     */
45     function _transferOwnership(address newOwner) internal virtual
46     override {
47         delete _pendingOwner;
48         super._transferOwnership(newOwner);
49     }
50     /**
51     * @dev The new owner accepts the ownership transfer.
52     */
53     function acceptOwnership() external {
54         address sender = _msgSender();
55         require(pendingOwner() == sender, "Ownable2Step: caller is
56         not the new owner");
57         _transferOwnership(sender);
58 }
```

Remediation Plan:

ACKNOWLEDGED: The Lybra Finance team acknowledged this finding.

4.17 (HAL-17) FLOATING PRAGMA - INFORMATIONAL (0.0)

Commit IDs affected:

- 90285107de8a6754954c303cd69d97b5fdb4e248

Description:

Contracts should be deployed with the same compiler version and flags used during development and testing. Locking the pragma helps to ensure that contracts do not accidentally get deployed using another pragma. For example, an outdated pragma version might introduce bugs that affect the contract system negatively.

Code Location:

All the contracts in the [repository](#) are using the `pragma solidity ^0.8.17;` floating pragma.

BVSS:

A0:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:F/S:C (0.0)

Recommendation:

Consider locking the pragma version in the smart contracts. It is not recommended to use a floating pragma in production.

For example: `pragma solidity 0.8.17;`

Remediation Plan:

ACKNOWLEDGED: The Lybra Finance team acknowledged this finding.

RECOMMENDATIONS OVERVIEW

1. Calculate the price of USDC with high precision using an external oracle in the `LybraConfigurator` contract. It is also recommended to implement a setter to update the maximum slippage tolerance in the `curvePool.exchange_underlying()` call.
2. Add a check to the `convertToPeUSD()` function that if it is used for dividend conversion, it will not be subject to the `getEUSDMaxLocked()` limitation.
3. Remove the `address provider` parameter from the `LybraPeUSDVaultBase.liquidation()` function and use always `msg.sender` as the `provider`.
4. Enforce in the `package.json` file to use the `openzeppelin/contracts` version `4.9.1`.
5. Do not assign the `DAO` and `GOV` roles to the `GovernanceTimelock` contract's deployer.
6. Use Chainlink's `latestRoundData()` function with checks on the return data with proper revert messages if the price is stale or the round is incomplete.
7. Add the `burnEnabled` modifier to the `PeUSDMainnet.convertToEUSD()` function.
8. Add the `mintEnabled` modifier to the `PeUSDMainnet.convertToPeUSD()` function.
9. Add a require statement in the `LybraRETHVault.depositEtherToMint()` that checks if the Rocket pool is full and reverts if so.
10. Add an `onlyOwner` setter to update the `dstChainId` state variable in the `StakingRewardsOnArbi` contract.
11. Add a require statement in the Vaults `withdraw()` function that checks that the minimum collateral deposited is higher than `1e18`.
12. Add a setter to the `LybraGovernance` contract to update the quorum.
13. Always use the `decimals()` function from the Chainlink Price Feed contracts to determine the number of decimals that the feed uses. This ensures that you always work with the correct precision.
14. Add the `immutable` modifier to the state variables mentioned.
15. Implement a two-step process where the owner nominates an account and the nominated account needs to call an `acceptOwnership()` function for the transfer of the ownership to fully succeed in all the `Ownable` contracts.
16. Lock the pragma version in all the smart contracts.

AUTOMATED TESTING

6.1 STATIC ANALYSIS REPORT

Description:

Halborn used automated testing techniques to enhance the coverage of certain areas of the smart contracts in scope. Among the tools used was Slither, a Solidity static analysis framework. After Halborn verified the smart contracts in the repository and was able to compile them correctly into their ABIS and binary format, Slither was run against the 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:

LybraConfigurator.sol

```

INFO[Detectors]:
LybraConfigurator.distributeRewards() (contracts/lybra/configuration/LybraConfigurator.sol#316-344) ignores return value by peUSD.transfer(address(LybraProtocolRewardsPool),peUSDbalance) (contracts/lybra/configuration/LybraConfigurator.sol#316-344)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO[Detectors]:
LybraConfigurator.initToken(address,address) (contracts/lybra/configuration/LybraConfigurator.sol#118-122) ignores return value by EUSD.approve(_spender,type()>uint256).max (contracts/lybra/configuration/LybraConfigurator.sol#121)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#balance
LybraConfigurator.distributeRewards() (contracts/lybra/configuration/LybraConfigurator.sol#316-344) ignores return value by EUSD.approve(address(curvePool),balance) (contracts/lybra/configuration/LybraConfigurator.sol#322)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-return
INFO[Detectors]:
LybraConfigurator.initialize(address,address,address,address) _stableToken (contracts/lybra/configuration/LybraConfigurator.sol#95) lacks a zero-check on :
- _stableToken = _stableToken (contracts/lybra/configuration/LybraConfigurator.sol#102)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO[Detectors]:
Reentrancy in LybraConfigurator.becomeRedemptionProvider(bool) (contracts/lybra/configuration/LybraConfigurator.sol#294-298):
External calls:
- EUSD.transfer(address,refreshReward(msg.sender)) (contracts/lybra/configuration/LybraConfigurator.sol#295)
  State variables modified after the call():
  - _redeemptionProvider[msg.sender] = _bool (contracts/lybra/configuration/LybraConfigurator.sol#296)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO[Detectors]:
Reentrancy in LybraConfigurator.becomeRedemptionProvider(bool) (contracts/lybra/configuration/LybraConfigurator.sol#294-298):
External calls:
- _USDMiningIncentives.refreshReward(msg.sender) (contracts/lybra/configuration/LybraConfigurator.sol#295)
  Event emitted after the call():
  - _USDMiningIncentives.onRewardRefreshed(bool) (contracts/lybra/configuration/LybraConfigurator.sol#297)
Reentrancy in LybraConfigurator.distributeRewards() (contracts/lybra/configuration/LybraConfigurator.sol#316-344):
External calls:
- EUSD.approve(address(curvePool),balance) (contracts/lybra/configuration/LybraConfigurator.sol#322)
  amount >= curvePool.exchangeUnderlying(0,2,balance,balance * 1000000000000000000 * 200 / 122) (contracts/lybra/configuration/LybraConfigurator.sol#329)
- IERC20(curvePool).safeTransfer(address(LybraProtocolRewardsPool),amount) (contracts/lybra/configuration/LybraConfigurator.sol#330)
  - LybraProtocolRewardsPool.notifyRewardAmount(amount); (contracts/lybra/configuration/LybraConfigurator.sol#331)
  Event emitted after the call():
  - LybraProtocolRewardsPool.onRewardAmountChanged(amount,block.timestamp) (contracts/lybra/configuration/LybraConfigurator.sol#332)
Reentrancy in LybraConfigurator.distributeRewards() (contracts/lybra/configuration/LybraConfigurator.sol#316-344):
External calls:
- EUSD.approve(address(pool),balance) (contracts/lybra/configuration/LybraConfigurator.sol#322)
  amount >= curvePool.exchangeUnderlying(0,2,balance,balance * 1000000000000000000 * 200 / 122) (contracts/lybra/configuration/LybraConfigurator.sol#329)
- IERC20(pool).safeTransfer(address(LybraProtocolRewardsPool),amount) (contracts/lybra/configuration/LybraConfigurator.sol#330)
  - LybraProtocolRewardsPool.notifyRewardAmount(amount); (contracts/lybra/configuration/LybraConfigurator.sol#331)
  Event emitted after the call():
  - LybraProtocolRewardsPool.onRewardAmountChanged(amount,block.timestamp) (contracts/lybra/configuration/LybraConfigurator.sol#332)
  - SendProtocolRewards(address(peUSD),peUSDbalance,block.timestamp) (contracts/lybra/configuration/LybraConfigurator.sol#342)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO[Detectors]:
Function LybraConfigurator.intToken(address,address) (contracts/lybra/configuration/LybraConfigurator.sol#118-122) has a dubious typecast: address>address
Function LybraConfigurator.distributeRewards() (contracts/lybra/configuration/LybraConfigurator.sol#316-344) has a dubious typecast: address<=IERC20
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/dubious_typecast.md
INFO[Detectors]:
Setter Function LybraConfigurator.setMinVaultAddress(address) (contracts/lybra/configuration/LybraConfigurator.sol#130-132) does not emit an event
Setter Function LybraConfigurator.setMinVaultMaxSupply(address,uint256) (contracts/lybra/configuration/LybraConfigurator.sol#140-142) does not emit an event
Setter Function LybraConfigurator.setMinVaultUnderlying(address,uint256) (contracts/lybra/configuration/LybraConfigurator.sol#150-152) does not emit an event
Setter Function LybraConfigurator.setPresetTradingEnabled(bool) (contracts/lybra/configuration/LybraConfigurator.sol#189-191) does not emit an event
Setter Function LybraConfigurator.setProtocolFeeRatio(uint256) (contracts/lybra/configuration/LybraConfigurator.sol#209-211) does not emit an event
Setter Function LybraConfigurator.setVaultMinPinnedAddress(bool) (contracts/lybra/configuration/LybraConfigurator.sol#209-211) does not emit an event
Setter Function LybraConfigurator.setVaultStableRatio(uint256) (contracts/lybra/configuration/LybraConfigurator.sol#278-281) does not emit an event
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/event.setter_and
INFO[Detectors]:
Pragma version=0.8.17 (contracts/lybra/configuration/LybraConfigurator.sol#5) allows old versions
Pragma version=0.8.17 (contracts/lybra/interfaces/IEUDPool.sol#2) allows old versions
Pragma version=0.8.17 (contracts/lybra/interfaces/IGovernanceTimelock.sol#2) allows old versions
Pragma version=0.8.17 (contracts/lybra/interfaces/IPeUSD.sol#2) allows old versions
Solidity 0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO[Detectors]:
Function [curvePool,exchangeUnderlying(int128,int128,uint256,uint256)] (contracts/lybra/configuration/LybraConfigurator.sol#38) is not in mixedCase
Parameter [curvePool,exchangeUnderlying(int128,int128,uint256,uint256),_amount] (contracts/lybra/configuration/LybraConfigurator.sol#38) is not in mixedCase
Parameter LybraConfigurator.initialize(address,address,address,_curvePool) (contracts/lybra/configuration/LybraConfigurator.sol#95) is not in mixedCase
Parameter LybraConfigurator.initialize(address,address,address,_stableToken) (contracts/lybra/configuration/LybraConfigurator.sol#95) is not in mixedCase
Parameter LybraConfigurator.initialize(address,address,address,_USDPriceFeed) (contracts/lybra/configuration/LybraConfigurator.sol#95) is not in mixedCase
Parameter LybraConfigurator.initialize(address,address,address,_eUSDOracle) (contracts/lybra/configuration/LybraConfigurator.sol#108) is not in mixedCase
Parameter LybraConfigurator.setUSDOracleAddress(_eUSDOracle) (contracts/lybra/configuration/LybraConfigurator.sol#108) is not in mixedCase
Parameter LybraConfigurator.setTokenMinting(address[],bool[]),_contract (contracts/lybra/configuration/LybraConfigurator.sol#267) is not in mixedCase
Parameter LybraConfigurator.setUSDOracleAddress(_eUSDOracle) (contracts/lybra/configuration/LybraConfigurator.sol#267) is not in mixedCase
Parameter LybraConfigurator.setUSDOracleAddress(_eUSDOracle) (contracts/lybra/configuration/LybraConfigurator.sol#267) is not in mixedCase
Parameter LybraConfigurator.setVaultStableRatio(uint256),ratio (contracts/lybra/configuration/LybraConfigurator.sol#78) is not in mixedCase
Parameter LybraConfigurator.becomeRedemptionProvider(bool),_bool (contracts/lybra/configuration/LybraConfigurator.sol#294) is not in mixedCase
Variable LybraConfigurator.GovernanceTimelock (contracts/lybra/configuration/LybraConfigurator.sol#50) is not in mixedCase
Variable LybraConfigurator.eUSD (contracts/lybra/configuration/LybraConfigurator.sol#50) is not in mixedCase
Parameter LybraConfigurator.setUSDOracleAddress(_eUSDOracle) (contracts/lybra/configuration/LybraConfigurator.sol#108) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

```

EUSDMiningIncentives.sol

```
INFO:Detectors:
Function EUSDMiningIncentives.updateReward(address) (contracts/lybra/miner/EUSDMiningIncentives.sol#79-80) is an unprotected initializer.
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/unprotected_initializer.md
INFO:Detectors:
EUSDMiningIncentives.bwOtherEarnings(address,bal) (contracts/lybra/miner/EUSDMiningIncentives.sol#95-236) performs a multiplication on the result of a division:
- biddingFee = (reward * biddingFeeRatio) / 1e8 (contracts/lybra/miner/EUSDMiningIncentives.sol#265)
EUSDMiningIncentives.notifyRewardAmount(uint256) (contracts/lybra/miner/EUSDMiningIncentives.sol#175-200) performs a multiplication on the result of a division:
- rewardRatio = amount / duration (contracts/lybra/miner/EUSDMiningIncentives.sol#200)
- rewardRatio = amount / duration (contracts/lybra/miner/EUSDMiningIncentives.sol#200)
- rewardRatio = amount / duration (contracts/lybra/miner/EUSDMiningIncentives.sol#200)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
```

LybraGovernance.sol

```
INFO:Detectors:
In a contract LybraGovernance (contracts/lybra/governance/LybraGovernance.sol#9-210) timelock-controller is used. Revoke deployer roles.
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/timelock_controller.md
INFO:Detectors:
Pragma version<=0.8.17 (contracts/lybra/governance/AdminTimelock.sol#3) allows old versions
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

INFO:Detectors:
In a contract AdminTimelock (contracts/lybra/governance/AdminTimelock.sol#7-9) timelock-controller is used. Revoke deployer roles.
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/timelock_controller.md
INFO:Detectors:
Pragma version<=0.8.17 (contracts/lybra/governance/AdminTimelock.sol#3) allows old versions
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

INFO:Detectors:
In a contract GovernanceTimelock (contracts/lybra/governance/GovernanceTimelock.sol#8-33) timelock-controller is used. Revoke deployer roles.
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/timelock_controller.md
INFO:Detectors:
Pragma version<=0.8.17 (contracts/lybra/governance/GovernanceTimelock.sol#8) allows old versions
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

INFO:Detectors:
Parameter GovernanceTimelock.checkRole(bytes32,address), sender (contracts/lybra/governance/GovernanceTimelock.sol#30) is not in mixedCase
Parameter GovernanceTimelock.checkOnlyRole(bytes32,address), sender (contracts/lybra/governance/GovernanceTimelock.sol#82) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
In a function GovernanceTimelock.constructor(uint256,address[],address) (contracts/lybra/governance/GovernanceTimelock.sol#14-22) variable GovernanceTimelock.DAO (contracts/lybra/governance/GovernanceTimelock.sol#9) is read multiple times
In a function GovernanceTimelock.constructor(uint256,address[],address) (contracts/lybra/governance/GovernanceTimelock.sol#14-22) variable GovernanceTimelock.GOV (contracts/lybra/governance/GovernanceTimelock.sol#12) is read multiple times
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/multiple_storage_read.md
```

LybraProxy.sol

```
INFO:Detectors:
Function LybraProxy.constructor(address,address,address) (contracts/lybra/configuration/LybraConfigurator.sol#95-101) contains magic numbers: 50, 500, 5
Function LybraConfigurator.setSafeCollateralRatio(address,uint256) (contracts/lybra/configuration/LybraConfigurator.sol#147-151) contains magic number: 130
Function LybraConfigurator.setVaultWeight(address,uint256) (contracts/lybra/configuration/LybraConfigurator.sol#173-178) contains magic number: 160
Function LybraConfigurator.setSafeCollateralRatio(address,uint256) (contracts/lybra/configuration/LybraConfigurator.sol#182-186) contains magic numbers: 160, 1e19
Function LybraConfigurator.setSafeCollateralRatio(address,uint256) (contracts/lybra/configuration/LybraConfigurator.sol#187-200) contains magic number: 200
Function LybraConfigurator.setKeeperRatio(address,uint256) (contracts/lybra/configuration/LybraConfigurator.sol#205-206) contains magic number: 5
Function LybraConfigurator.setMaxStableRatio(uint256) (contracts/lybra/configuration/LybraConfigurator.sol#278-281) contains magic number: 10
Function LybraConfigurator.setSafeCollateralRatio(address,uint256) (contracts/lybra/configuration/LybraConfigurator.sol#282-285) contains magic number: 1e21
Function LybraConfigurator.getSafeCollateralRatio(address) (contracts/lybra/configuration/LybraConfigurator.sol#367-372) contains magic number: 1e19
Function LybraConfigurator.getSafeCollateralRatio(address) (contracts/lybra/configuration/LybraConfigurator.sol#374-377) contains magic number: 1e19
Function LybraConfigurator.getSafeCollateralRatio(address) (contracts/lybra/configuration/LybraConfigurator.sol#378-381) contains magic number: 1e19
Function LybraConfigurator.getSafeCollateralRatio(address) (contracts/lybra/configuration/LybraConfigurator.sol#382-385) contains magic number: 1e19
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/magic_number_md
INFO:Detectors:
In a function LybraConfigurator.distributeRewards() (contracts/lybra/configuration/LybraConfigurator.sol#316-344) variable LybraConfigurator.lybraProtocolRewardsPool (contracts/lybra/configuration/LybraConfigurator.sol#61) is read multiple times
In a function LybraConfigurator.distributeRewards() (contracts/lybra/configuration/LybraConfigurator.sol#316-344) variable LybraConfigurator.peUSD (contracts/lybra/configuration/LybraConfigurator.sol#63) is read multiple times
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/multiple_storage_read_md
```

LybraProxyAdmin.sol

```
INFO:Detectors:
Pragma version<=0.8.17 (contracts/lybra/Proxy/LybraProxyAdmin.sol#3) allows old versions
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

AdminTimelock.sol

```
INFO:Detectors:
In a contract AdminTimelock (contracts/lybra/governance/AdminTimelock.sol#7-9) timelock-controller is used. Revoke deployer roles.
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/timelock_controller.md
INFO:Detectors:
Pragma version<=0.8.17 (contracts/lybra/governance/AdminTimelock.sol#3) allows old versions
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

GovernanceTimelock.sol

```
INFO:Detectors:
In a contract GovernanceTimelock (contracts/lybra/governance/GovernanceTimelock.sol#8-33) timelock-controller is used. Revoke deployer roles.
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/timelock_controller.md
INFO:Detectors:
Pragma version<=0.8.17 (contracts/lybra/governance/GovernanceTimelock.sol#8) allows old versions
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Parameter GovernanceTimelock.checkRole(bytes32,address), sender (contracts/lybra/governance/GovernanceTimelock.sol#30) is not in mixedCase
Parameter GovernanceTimelock.checkOnlyRole(bytes32,address), sender (contracts/lybra/governance/GovernanceTimelock.sol#82) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
In a function GovernanceTimelock.constructor(uint256,address[],address) (contracts/lybra/governance/GovernanceTimelock.sol#14-22) variable GovernanceTimelock.DAO (contracts/lybra/governance/GovernanceTimelock.sol#9) is read multiple times
In a function GovernanceTimelock.constructor(uint256,address[],address) (contracts/lybra/governance/GovernanceTimelock.sol#14-22) variable GovernanceTimelock.GOV (contracts/lybra/governance/GovernanceTimelock.sol#12) is read multiple times
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/multiple_storage_read_md
```

LybraGovernance.sol

```
INFO:Detectors:
In a contract LybraGovernance (contracts/lybra/governance/LybraGovernance.sol#9-210) timelock-controller is used. Revoke deployer roles.
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/timelock_controller.md
INFO:Detectors:
LybraGovernance.CLOCK_MODE (contracts/lybra/governance/LybraGovernance.sol#154-157) uses a dangerous strict equality:
- require(bool,string)(clock) == block.number, Votes: broken clock mode (contracts/lybra/governance/LybraGovernance.sol#155)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Function LybraGovernance.constructor(string,timelockController,address) (contracts/lybra/governance/LybraGovernance.sol#41-44) has a dubious typecast: IGovernanceTimelock<address>
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/dubious_typecast_md
INFO:Detectors:
LybraGovernance._countVotes(uint256,address,uint256,bytes) (contracts/lybra/governance/LybraGovernance.sol#85-95) compares to a boolean constant:
- require(bool,string)(receipt.hasVoted == false, GovernorOrProxy::castVotesInternal: voter already voted) (contracts/lybra/governance/LybraGovernance.sol#89)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
Pragma version<=0.8.17 (contracts/lybra/governance/LybraGovernance.sol#3) allows old versions
Pragma version<=0.8.17 (contracts/lyra/interfaces/GovernanceTimelock.sol#1) allows old versions
Pragma version<=0.8.17 (contracts/lyra/interfaces/TestDR.sol#2) allows old versions
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Function LybraGovernance._CLOCK_MODE (contracts/lybra/governance/LybraGovernance.sol#154-157) is not in mixedCase
Function LybraGovernance._COUNTING_MODE (contracts/lybra/governance/LybraGovernance.sol#159-161) is not in mixedCase
Variable LybraGovernance.governanceTimelock (contracts/lybra/governance/LybraGovernance.sol#12) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable LybraGovernance.supportsInterface(bytes4) (contracts/lybra/governance/LybraGovernance.sol#190-193) is too similar to LybraGovernance.supportsInterface(bytes4), governor461d (contracts/lybra/governance/LybraGovernance.sol#190-193)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
Function LybraGovernance._quarantine(uint256) (contracts/lybra/governance/LybraGovernance.sol#53-55) contains magic number: 4
Function LybraGovernance.votingPeriod() (contracts/lybra/governance/LybraGovernance.sol#146-149) contains magic number: 50400
Function LybraGovernance._quarantine(uint256) (contracts/lybra/governance/LybraGovernance.sol#158-162) contains magic number: 14400
Function LybraGovernance._pruneAllTheWorld() (contracts/lybra/governance/LybraGovernance.sol#175-177) contains magic number: 1e23
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/magic_number_md
INFO:Detectors:
LybraGovernance.GovernanceTimelock (contracts/lybra/governance/LybraGovernance.sol#12) should be immutable
LybraGovernance.state.wslB (contracts/lybra/governance/LybraGovernance.sol#1) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
```

AUTOMATED TESTING

```

INFO[Detectors]:
EUSDMiningIncentives.getBoost(address).reDEMPTIONBoost (contracts/lybra/miner/EUSDMiningIncentives.sol#215) is a local variable never initialized
EUSDMiningIncentives.buyOtherFarmings(address[],bool).i (contracts/lybra/miner/EUSDMiningIncentives.sol#279) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO[Detectors]:
EUSDMiningIncentives.getReward() (contracts/lybra/miner/EUSDMiningIncentives.sol#27-245) ignores return value by IsesLR(esLR).mint(msg.sender,reward) (contracts/lybra/miner/EUSDMiningIncentives.sol#242)
EUSDMiningIncentives.buyOtherFarmings(address, bool) (contracts/lybra/miner/EUSDMiningIncentives.sol#257-276) ignores return value by IsesLR(esLR).burn(msg.sender,biddingFee) (contracts/lybra/miner/EUSDMiningIncentives.sol#272)
EUSDMiningIncentives.buyOtherFarmings(address, bool) (contracts/lybra/miner/EUSDMiningIncentives.sol#257-276) ignores return value by IsesLR(esLR).mint(msg.sender,reward) (contracts/lybra/miner/EUSDMiningIncentives.sol#274)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO[Detectors]:
EUSDMiningIncentives.setBIDDINGCost(uint256) (contracts/lybra/miner/EUSDMiningIncentives.sol#111-114) should emit an event for:
- biddingFeeRatio = biddingRateRatio (contracts/lybra/miner/EUSDMiningIncentives.sol#113)
EUSDMiningIncentives.setTXRateRatio(uint256) (contracts/lybra/miner/EUSDMiningIncentives.sol#116-119) should emit an event for:
EUSDMiningIncentives.setMinDollarRatio(uint256) (contracts/lybra/miner/EUSDMiningIncentives.sol#121-124) should emit an event for:
- minDollarRatio = ratio (contracts/lybra/miner/EUSDMiningIncentives.sol#123)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO[Detectors]:
EUSDMiningIncentives.constructor(address,address,address,address), _weth (contracts/lybra/miner/EUSDMiningIncentives.sol#70) lacks a zero-check on:
- wETH = weth (contracts/lybra/miner/EUSDMiningIncentives.sol#147)
EUSDMiningIncentives.constructor(address,address,address,address), oldUSD (contracts/lybra/miner/EUSDMiningIncentives.sol#70) lacks a zero-check on:
- oldLyra = oldUSD (contracts/lybra/miner/EUSDMiningIncentives.sol#76)
EUSDMiningIncentives.setETHBITLEStakePool(address) (contracts/lybra/miner/EUSDMiningIncentives.sol#111) lacks a zero-check on:
- LBR = lbr (contracts/lybra/miner/EUSDMiningIncentives.sol#91)
EUSDMiningIncentives.setETHBITLEStakePool(address) (contracts/lybra/miner/EUSDMiningIncentives.sol#139) lacks a zero-check on:
- ethBITLEStakePool = pool (contracts/lybra/miner/EUSDMiningIncentives.sol#140)
EUSDMiningIncentives.setETHBITLEStakePool(address,address), _lp (contracts/lybra/miner/EUSDMiningIncentives.sol#139) lacks a zero-check on:
- ethBITLElpoken = lp (contracts/lybra/miner/EUSDMiningIncentives.sol#141)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO[Detectors]:
EUSDMiningIncentives.setPoolTotal() (contracts/lybra/miner/EUSDMiningIncentives.sol#102-103) has external calls inside a loop: require(bool,string)(configurator.mintVault,_vaults[i]),NOT_VAULT (contracts/lybra/miner/EUSDMiningIncentives.sol#103)
EUSDMiningIncentives.setTotalStaked() (contracts/lybra/miner/EUSDMiningIncentives.sol#154-164) has external calls inside a loop: amount += vault.getPoolTotalCirculation() * configurator.getVaultWeight(vaults[i]) / 1e20 (contracts/lybra/miner/EUSDMiningIncentives.sol#154)
EUSDMiningIncentives.staked0(Address) (contracts/lybra/miner/EUSDMiningIncentives.sol#160-179) has external calls inside a loop: amount += vault.getBorrowedOfUser() * configurator.getVaultWeight(vaults[i]) / 1e20 (contracts/lybra/miner/EUSDMiningIncentives.sol#171)
EUSDMiningIncentives.staked1(Address) (contracts/lybra/miner/EUSDMiningIncentives.sol#154-164) has external calls inside a loop: amount += vault.getLoyaltyRewards() * configurator.getVaultWeight(vaults[i]) / 1e20 (contracts/lybra/miner/EUSDMiningIncentives.sol#176)
EUSDMiningIncentives.staked2(Address) (contracts/lybra/miner/EUSDMiningIncentives.sol#160-179) has external calls inside a loop: amount += Illybra(oldLyra).getBorrowedOfUser() * configurator.getVaultWeight(oldLyra) / 1e20 (contracts/lybra/miner/EUSDMiningIncentives.sol#187)
EUSDMiningIncentives.staked3(Address) (contracts/lybra/miner/EUSDMiningIncentives.sol#160-179) has external calls inside a loop: amount += Illybra(oldLyra).totalSupply() * configurator.getVaultWeight(oldLyra) / 1e20 (contracts/lybra/miner/EUSDMiningIncentives.sol#187)
EUSDMiningIncentives.getBorrowed() (contracts/lybra/miner/EUSDMiningIncentives.sol#214-220) has external calls inside a loop: configurator.isRedemptionProvider(_account) (contracts/lybra/miner/EUSDMiningIncentives.sol#216)
EUSDMiningIncentives.setETHBITLEStakePool() (contracts/lybra/miner/EUSDMiningIncentives.sol#216-220) has external calls inside a loop: 100 * 1e20 * redemptionBoost * esLRBoost * getStakePool(_account).userUpdateddt[_account].FinishDt (contract/lybra/miner/EUSDMiningIncentives.sol#219)
EUSDMiningIncentives.setStakedBRPValue(address) (contracts/lybra/miner/EUSDMiningIncentives.sol#186-195) has external calls inside a loop: totalSupply = IUSO(stakePool).totalSupply() (contracts/lybra/miner/EUSDMiningIncentives.sol#187)
EUSDMiningIncentives.setStakedBRPValue(address) (contracts/lybra/miner/EUSDMiningIncentives.sol#186-195) has external calls inside a loop: (etherPrice = IUSO(stakePool).totalSupplyData()) (contracts/lybra/miner/EUSDMiningIncentives.sol#189)
EUSDMiningIncentives.setStakedBRPValue(address) (contracts/lybra/miner/EUSDMiningIncentives.sol#186-195) has external calls inside a loop: (etherPrice = Illybra(lbr).latestRoundData()) (contracts/lybra/miner/EUSDMiningIncentives.sol#190)
EUSDMiningIncentives.setStakedBRPValue(address) (contracts/lybra/miner/EUSDMiningIncentives.sol#186-195) has external calls inside a loop: etherInlp = (IUSO(stakePool).balanceOf(etherBITLEpoken) * uint256(etherPrice)) / 1e8 (contracts/lybra/miner/EUSDMiningIncentives.sol#191)
EUSDMiningIncentives.setStakedBRPValue(address) (contracts/lybra/miner/EUSDMiningIncentives.sol#186-195) has external calls inside a loop: Illybra(lbr).balanceOf(etherBITLEpoken) * uint256(lbrPrice)) / 1e8 (contracts/lybra/miner/EUSDMiningIncentives.sol#192)
EUSDMiningIncentives.setStakedBRPValue(address) (contracts/lybra/miner/EUSDMiningIncentives.sol#186-195) has external calls inside a loop: userStaked = IUSO(stakePool).balanceOf(user) (contracts/lybra/miner/EUSDMiningIncentives.sol#193)
EUSDMiningIncentives._buyOtherFarmings(address,bool) (contracts/lybra/miner/EUSDMiningIncentives.sol#257-276) has external calls inside a loop: (lbrPrice = IllybraPriceFeed.latestRoundData()) (contracts/lybra/miner/EUSDMiningIncentives.sol#194)
EUSDMiningIncentives._buyOtherFarmings(address,bool) (contracts/lybra/miner/EUSDMiningIncentives.sol#257-276) has external calls inside a loop: success = EUSD.transferFrom(msg.sender,address(owner()),biddingFee) (contracts/lybra/miner/EUSDMiningIncentives.sol#209)
EUSDMiningIncentives._buyOtherFarmings(address,bool) (contracts/lybra/miner/EUSDMiningIncentives.sol#257-276) has external calls inside a loop: lesLR(esLR).mint(msg.sender,reward) (contracts/lybra/miner/EUSDMiningIncentives.sol#274)
EUSDMiningIncentives._buyOtherFarmings(address,bool) (contracts/lybra/miner/EUSDMiningIncentives.sol#257-276) has external calls inside a loop: IsesLR(esLR).burn(msg.sender,biddingFee) (contracts/lybra/miner/EUSDMiningIncentives.sol#272)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#calls-inside-a-loop
INFO[Detectors]:
Reentrancy in EUSDMiningIncentives._buyOtherFarmings(address,bool) (contracts/lybra/miner/EUSDMiningIncentives.sol#257-276):
  External calls:
    - Illybra(lbr).transferFrom(msg.sender,address(owner())),biddingFee (contracts/lybra/miner/EUSDMiningIncentives.sol#269)
    - IsesLR(esLR).burn(msg.sender,paddingFee) (contracts/lybra/miner/EUSDMiningIncentives.sol#277)
    - IsesLR(esLR).mint(msg.sender,reward) (contracts/lybra/miner/EUSDMiningIncentives.sol#274)
  Event emitted after the call(s):
    - Illybra(lbr).onTransfer(address, user, reward, biddingFee, user0, block.timestamp) (contracts/lybra/miner/EUSDMiningIncentives.sol#275)
  Reentrancy in EUSDMiningIncentives._getReward():
  External calls:
    - IsesLR(esLR).emit(msg.sender,reward) (contracts/lybra/miner/EUSDMiningIncentives.sol#242)
  Event emitted after the call(s):
    - Illybra(lbr).onTransfer(address, user, reward, biddingFee, user0, block.timestamp) (contracts/lybra/miner/EUSDMiningIncentives.sol#275)
  Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities
INFO[Detectors]:
EUSDMiningIncentives.setRewardsDuration(uint256) (contracts/lybra/miner/EUSDMiningIncentives.sol#134-137) uses timestamp for comparisons
  Dangerous comparisons:
    - require(bool,string)[(reward[user] < block.timestamp, reward duration not finished)] (contracts/lybra/miner/EUSDMiningIncentives.sol#135)
EUSDMiningIncentives._buyOtherFarmings(address,bool) (contracts/lybra/miner/EUSDMiningIncentives.sol#257-276) uses timestamp for comparisons
  Dangerous comparisons:
    - require(bool,string)[(rewards[user] >= 0,24)] (contracts/lybra/miner/EUSDMiningIncentives.sol#259)
EUSDMiningIncentives.setRewardsMount(uint256) (contracts/lybra/miner/EUSDMiningIncentives.sol#284-300) uses timestamp for comparisons
  Dangerous comparisons:
    - block.timestamp >> finishDt (contracts/lybra/miner/EUSDMiningIncentives.sol#288)
    - require(bool,string)[(rewardRatio >= 0, reward ratio = 0)] (contracts/lybra/miner/EUSDMiningIncentives.sol#295)
EUSDMiningIncentives.setSupported(bool) (contracts/lybra/miner/EUSDMiningIncentives.sol#302-304) uses timestamp for comparisons
  Dangerous comparisons:
    - x < y (contracts/lybra/miner/EUSDMiningIncentives.sol#303)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO[Detectors]:
Setter Function EUSDMiningIncentives.setBIDDINGCost(uint256) (contracts/lybra/miner/EUSDMiningIncentives.sol#111-114) does not emit an event
Setter Function EUSDMiningIncentives.setTXRateRatio(uint256) (contracts/lybra/miner/EUSDMiningIncentives.sol#116-119) does not emit an event
Setter Function EUSDMiningIncentives.setMinDollarRatio(uint256) (contracts/lybra/miner/EUSDMiningIncentives.sol#121-124) does not emit an event
Setter Function EUSDMiningIncentives.setETHBITLEStakePool(address) (contracts/lybra/miner/EUSDMiningIncentives.sol#111-114) does not emit an event
Setter Function EUSDMiningIncentives.setRewardsDuration(uint256) (contracts/lybra/miner/EUSDMiningIncentives.sol#134-137) does not emit an event
Setter Function EUSDMiningIncentives.setETHBITLEStakeInfo(address,address) (contracts/lybra/miner/EUSDMiningIncentives.sol#143-145) does not emit an event
Setter Function EUSDMiningIncentives.setETHBITLEStakeInfo(address,address), _pool (contracts/lybra/miner/EUSDMiningIncentives.sol#143-145) does not emit an event
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop
INFO[Detectors]:
Pragma version=0.8.17 (contracts/lybra/interfaces/IEUSD.sol#2) allows old versions
Pragma version=0.8.17 (contracts/lybra/interfaces/lybra.sol#2) allows old versions
Pragma version=0.8.17 (contracts/lybra/interfaces/IConfiguror.sol#2) allows old versions
Pragma version=0.8.17 (contracts/lybra/interfaces/IsesLR.sol#2) allows old versions
Pragma version=0.8.17 (contracts/lybra/miner/EUSDMiningIncentives.sol#3) allows old versions
Solidity 0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO[Detectors]:
Parameter IUSO.getSharesByMined(uint256), EUSD.mount (contracts/lybra/interfaces/IEUSD.sol#2) is not in mixedCase
Parameter EUSDMiningIncentives.setToken0(address,address), _lbr (contracts/lybra/miner/EUSDMiningIncentives.sol#91) is not in mixedCase
Parameter EUSDMiningIncentives.setToken1(address,address), _weth (contracts/lybra/miner/EUSDMiningIncentives.sol#91) is not in mixedCase
Parameter EUSDMiningIncentives.setLBROracle(address), lbrOracle (contracts/lybra/miner/EUSDMiningIncentives.sol#107) is not in mixedCase
Parameter EUSDMiningIncentives.setPoolAddress(), vault (contracts/lybra/miner/EUSDMiningIncentives.sol#102) is not in mixedCase
Parameter EUSDMiningIncentives.setBIDDINGCost(uint256), biddingFee (contracts/lybra/miner/EUSDMiningIncentives.sol#111) is not in mixedCase
Parameter EUSDMiningIncentives.setTXRateRatio(uint256), txRateRatio (contracts/lybra/miner/EUSDMiningIncentives.sol#116) is not in mixedCase
Parameter EUSDMiningIncentives.setMinDollarRatio(uint256), minDollarRatio (contracts/lybra/miner/EUSDMiningIncentives.sol#121) is not in mixedCase
Parameter EUSDMiningIncentives.setETHBITLEStakePool(bool), pool (contracts/lybra/miner/EUSDMiningIncentives.sol#130) is not in mixedCase
Parameter EUSDMiningIncentives.setRewardsDuration(uint256), duration (contracts/lybra/miner/EUSDMiningIncentives.sol#134) is not in mixedCase
Parameter EUSDMiningIncentives.setETHBITLEStakeInfo(address,address), _pool (contracts/lybra/miner/EUSDMiningIncentives.sol#139) is not in mixedCase
Parameter EUSDMiningIncentives.setSupported(bool), supported (contracts/lybra/miner/EUSDMiningIncentives.sol#141) is not in mixedCase
Parameter EUSDMiningIncentives.setRewardsMount(uint256), durationIP (contracts/lybra/miner/EUSDMiningIncentives.sol#143) is not in mixedCase
Parameter EUSDMiningIncentives.setETHBITLEStakeInfo(address,address), pool (contracts/lybra/miner/EUSDMiningIncentives.sol#144) is not in mixedCase
Parameter EUSDMiningIncentives.refreshAddress(), _account (contracts/lybra/miner/EUSDMiningIncentives.sol#212) is not in mixedCase
Parameter EUSDMiningIncentives.getPool(address), _account (contracts/lybra/miner/EUSDMiningIncentives.sol#222) is not in mixedCase
Variable EUSDMiningIncentives.IUSD (contracts/lybra/miner/EUSDMiningIncentives.sol#1) is not in mixedCase
Variable EUSDMiningIncentives.IUSD (contracts/lybra/miner/EUSDMiningIncentives.sol#1) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO[Detectors]:
EUSDMiningIncentives.setPools(address[]) (contracts/lybra/miner/EUSDMiningIncentives.sol#92-109) contains magic number: 10
Function EUSDMiningIncentives.setBIDDINGCost(uint256) (contracts/lybra/miner/EUSDMiningIncentives.sol#111-114) contains magic number: 0000
Function EUSDMiningIncentives.setTXRateRatio(uint256) (contracts/lybra/miner/EUSDMiningIncentives.sol#116-119) contains magic number: 1e20
Function EUSDMiningIncentives.totalStaked() (contracts/lybra/miner/EUSDMiningIncentives.sol#154-164) contains magic numbers: 1e20, 1e20
Function EUSDMiningIncentives.setMinDollarRatio(uint256) (contracts/lybra/miner/EUSDMiningIncentives.sol#121-124) contains magic numbers: 1e20, 1e20
Function EUSDMiningIncentives.setETHBITLEValue(address) (contracts/lybra/miner/EUSDMiningIncentives.sol#186-195) contains magic numbers: 1e8, 1e8
Function EUSDMiningIncentives.getBorrowed() (contracts/lybra/miner/EUSDMiningIncentives.sol#214-220) contains magic number: 1e0
Function EUSDMiningIncentives.earn(address) (contracts/lybra/miner/EUSDMiningIncentives.sol#222-224) contains magic number: 1e38
Function EUSDMiningIncentives.setSupported(bool) (contracts/lybra/miner/EUSDMiningIncentives.sol#302-304) contains magic number: 1e38
Function EUSDMiningIncentives.setRewardsMount(uint256) (contracts/lybra/miner/EUSDMiningIncentives.sol#302-304) contains magic numbers: 1e10, 1e8
Function EUSDMiningIncentives.setETHBITLEStakeVariables() (contracts/lybra/miner/EUSDMiningIncentives.sol#26-305) contains magic numbers: 1e10, 3000, 500
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#magic-number
INFO[Detectors]:
EUSDMiningIncentives._otherPicardes() (contracts/lybra/miner/EUSDMiningIncentives.sol#55) should be immutable
EUSDMiningIncentives.WETH (contracts/lybra/miner/EUSDMiningIncentives.sol#12) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variables-that-could-be-declared-immutable
INFO[Detectors]:
In a function EUSDMiningIncentives.setTotalStaked() (contracts/lybra/miner/EUSDMiningIncentives.sol#154-164) variable EUSDMiningIncentives.vaults (contracts/lybra/miner/EUSDMiningIncentives.sol#133) is read multiple times
In a function EUSDMiningIncentives.staked0(Address) (contracts/lybra/miner/EUSDMiningIncentives.sol#160-179) variable EUSDMiningIncentives.vaults (contracts/lybra/miner/EUSDMiningIncentives.sol#133) is read multiple times
In a function EUSDMiningIncentives.staked1(Address) (contracts/lybra/miner/EUSDMiningIncentives.sol#154-164) variable EUSDMiningIncentives.vaults (contracts/lybra/miner/EUSDMiningIncentives.sol#133) is read multiple times
In a function EUSDMiningIncentives.staked2(Address) (contracts/lybra/miner/EUSDMiningIncentives.sol#160-179) variable EUSDMiningIncentives.vaults (contracts/lybra/miner/EUSDMiningIncentives.sol#133) is read multiple times
In a function EUSDMiningIncentives.notifyRewardsMount(uint256) (contracts/lybra/miner/EUSDMiningIncentives.sol#284-300) variable EUSDMiningIncentives.duration (contracts/lybra/miner/EUSDMiningIncentives.sol#36) is read multiple times
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#multiple-storage-read

```


AUTOMATED TESTING

esLBRBoost.sol

```

Reentrancy in esLBRBoost.getterRoot(address,int256,uint256) (contracts/lybra/miner/esLBRBoost.sol#90-108):
    External calls:
        - esLBR.burn(msg.sender,_amount) (contracts/lybra/miner/ProtocolRewardsPool.sol#103)
        - withdraw(msg.sender) (contracts/lybra/miner/ProtocolRewardsPool.sol#104)
            > _LBR.mintUser(_amount) (contracts/lybra/miner/ProtocolRewardsPool.sol#117)
    Event emitted after the call(s):
        - UnstakeBRL(msg.sender,_amount,_block,_timestamp) (contracts/lybra/miner/ProtocolRewardsPool.sol#111)
        - WithdrawBRL(_user,_amount,_block,_timestamp) (contracts/lybra/miner/ProtocolRewardsPool.sol#120)
            > withdraw(msg.sender) (contracts/lybra/miner/ProtocolRewardsPool.sol#104)
    Reentrancy in esLBR.mintUser(address,int256) (contracts/lybra/miner/ProtocolRewardsPool.sol#114-121):
        External calls:
            - _LBR.mintUser(_amount) (contracts/lybra/miner/ProtocolRewardsPool.sol#117)
        Event emitted after the call(s):
            - UnstakeBRL(_user,_amount,_block,_timestamp) (contracts/lybra/miner/ProtocolRewardsPool.sol#120)
    Reference: https://github.com/crytic/slither/wikil/Detector-Documentation#reentrancy-vulnerabilities-1
INFODetectors:
ProtocolRewardsPool.unstake(uint256) (contracts/lybra/miner/ProtocolRewardsPool.sol#98-112):
    Dangerous comparisons:
        - timeToFullRedemption <= lockTime (contracts/lybra/miner/ProtocolRewardsPool.sol#100)
        - timeToFullRedemption[msg.sender] > block.timestamp (contracts/lybra/miner/ProtocolRewardsPool.sol#106)
ProtocolRewardsPool.unlockParemteres() (contracts/lybra/miner/ProtocolRewardsPool.sol#177-178) uses timestamp for comparisons
    Dangerous comparisons:
        - block.timestamp + executeCycle - 259200 <= timeToFullRedemption[msg.sender].END (contracts/lybra/miner/ProtocolRewardsPool.sol#128)
ProtocolRewardsPool.getClaimableBRL(address) (contracts/lybra/miner/ProtocolRewardsPool.sol#179-183) uses timestamp for comparisons
    Dangerous comparisons:
        - timeToFullRedemption[_user] > lastWithdrawTime[_user] (contracts/lybra/miner/ProtocolRewardsPool.sol#180)
        - lastWithdrawTime[_user] > timeToFullRedemption[_user] (contracts/lybra/miner/ProtocolRewardsPool.sol#181)
ProtocolRewardsPool.getReservedBalance() (contracts/lybra/miner/ProtocolRewardsPool.sol#185-189) uses timestamp for comparisons
    Dangerous comparisons:
        - timeToFullRedemption[_user] > block.timestamp (contracts/lybra/miner/ProtocolRewardsPool.sol#186)
Reference: https://github.com/crytic/slither/wikil/Detector-Documentation#block-timestamp
INFODetectors:
Setter function ProtocolRewardsPool.setTokenAddress(address,address,address) (contracts/lybra/miner/ProtocolRewardsPool.sol#57-62) does not emit an event
Setter function ProtocolRewardsPool.setGasCost(uint256) (contracts/lybra/miner/ProtocolRewardsPool.sol#64-67) does not emit an event
Setter function ProtocolRewardsPool.setBRLDecile(address) (contracts/lybra/miner/ProtocolRewardsPool.sol#69-71) does not emit an event
Setter function ProtocolRewardsPool.setGasCostBRL(address) (contracts/lybra/miner/ProtocolRewardsPool.sol#72-73)
INFODetectors:
Pragma version 0.8.17 (contracts/lybra/interfaces/I188.sol#1) allows old versions
Pragma version 0.8.17 (contracts/lybra/interfaces/IConfigurator.sol#1) allows old versions
Pragma version 0.8.17 (contracts/lybra/interfaces/IBlock.sol#1) allows old versions
Pragma version 0.8.17 (contracts/lybra/miner/ProtocolRewardsPool.sol#1) allows old versions
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wikil/Detector-Documentation#incorrect-versions-of-solidity
INFODetectors:
Parameter ProtocolRewardsPool.setTokenAddress(address,address,address,_lbr) (contracts/lybra/miner/ProtocolRewardsPool.sol#57) is not in mixedCase
Parameter ProtocolRewardsPool.setTokenAddress(address,address,address,_lbr).boos (contracts/lybra/miner/ProtocolRewardsPool.sol#57) is not in mixedCase
Parameter ProtocolRewardsPool.setTokenAddress(address,address,address,_lbr).boos.Boos (contracts/lybra/miner/ProtocolRewardsPool.sol#57) is not in mixedCase
Parameter ProtocolRewardsPool.setTokenAddress(address,address,address,_lbr).Boos (contracts/lybra/miner/ProtocolRewardsPool.sol#57) is not in mixedCase
Parameter ProtocolRewardsPool.setTokenAddress(address,address,address,_lbr).Boos.Boos (contracts/lybra/miner/ProtocolRewardsPool.sol#57) is not in mixedCase
Parameter ProtocolRewardsPool.setBRLDecile(address,_lbrDecile) (contracts/lybra/miner/ProtocolRewardsPool.sol#60) is not in mixedCase
Parameter ProtocolRewardsPool.learnedAddress(_account) (contracts/lybra/miner/ProtocolRewardsPool.sol#191) is not in mixedCase
Parameter ProtocolRewardsPool.refreshReward(Address,_account) (contracts/lybra/miner/ProtocolRewardsPool.sol#204) is not in mixedCase
Variable ProtocolRewardsPool.lastWithdrawTime (contracts/lybra/miner/ProtocolRewardsPool.sol#180) is not in mixedCase
Reference: https://github.com/crytic/slither/wikil/Detector-Documentation#solidity-naming-conventions
INFODetectors:
Function ProtocolRewardsPool.setGasCost(uint256) (contracts/lybra/miner/ProtocolRewardsPool.sol#64-67) contains magic number: 8000
Function ProtocolRewardsPool.setTokenAddress(address,address,address,_lbr) (contracts/lybra/miner/ProtocolRewardsPool.sol#127-130) contains magic number: 259200
Function ProtocolRewardsPool.setGasCostBRL(uint256) (contracts/lybra/miner/ProtocolRewardsPool.sol#72-73) contains magic number: 10, 10, 10
Function ProtocolRewardsPool.setBRLDecile(address) (contracts/lybra/miner/ProtocolRewardsPool.sol#173-177) contains magic number: 75e3
Function ProtocolRewardsPool.notifyReward(uint256,uint256) (contracts/lybra/miner/ProtocolRewardsPool.sol#221-229) contains magic number: 10
Function ProtocolRewardsPool.notifyRewarded(uint256,uint256) (contracts/lybra/miner/ProtocolRewardsPool.sol#241-251) contains magic number: 1e36
Function ProtocolRewardsPool.setLastConstructiveVariables() (contracts/lybra/miner/ProtocolRewardsPool.sol#246-252) contains magic numbers: 7770000, 3000
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/magic_numbers.ad
INFODetectors:
In a function ProtocolRewardsPool.getUserRoot(address) (contracts/lybra/miner/ProtocolRewardsPool.sol#179-183) variable ProtocolRewardsPool.lastWithdrawTime (contracts/lybra/miner/ProtocolRewardsPool.sol#41) is read multiple times
In a function ProtocolRewardsPool.getClaimableBRL(address) (contracts/lybra/miner/ProtocolRewardsPool.sol#179-183) variable ProtocolRewardsPool.lastWithdrawTime (contracts/lybra/miner/ProtocolRewardsPool.sol#180) is read multiple times
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/magic_numbers.ad
INFODetectors:
esLBRStatus.setLockStatus(uint256,uint256,bool) (contracts/lybra/miner/esLBRStatus.sol#5-71):
    External calls:
        - esLBR.burn(msg.sender,_lbrAmount) (contracts/lybra/miner/esLBRBoost.sol#53)
        - esLBR(miningIncentives.esLBR()).mint(msg.sender,_lbrAmount) (contracts/lybra/miner/esLBRBoost.sol#64)
        - miningIncentives.refreshReward(msg.sender) (contracts/lybra/miner/esLBRBoost.sol#68)
    State variables written after the call(s):
        - lockStatus (contracts/lybra/miner/esLBRStatus.sol#51) can be used in cross function reentrances:
            - esLBRStatus.getLockStatus(_address,_lockAmount,_block,_timestamp,_setting.duration,_setting.duration,_setting.miningBoost) (contracts/lybra/miner/esLBRBoost.sol#69)
            - esLBRStatus.setUserLockStatus(_address,_lockAmount,_block,_timestamp) (contracts/lybra/miner/esLBRStatus.sol#121)
            - esLBRStatus.getterRoot(_address,uint256,uint256) (contracts/lybra/miner/esLBRStatus.sol#108)
            - esLBRStatus.unlock(_address) (contracts/lybra/miner/esLBRStatus.sol#71)
        - esLBRStatus.setUserLockStatus (contracts/lybra/miner/esLBRStatus.sol#12)
    Reference: https://github.com/crytic/slither/wikil/Detector-Documentation#reentrancy-vulnerabilities-1
INFODetectors:
esLBRStatus.setLockStatus(uint256,uint256,uint256,uint256,uint256,uint256) (contracts/lybra/miner/esLBRStatus.sol#55-71) ignores return value by esLBR(miningIncentives.esLBR().burn(msg.sender,_lbrAmount)) (contracts/lybra/miner/esLBRBoost.sol#63)
esLBRStatus.setLockStatus(uint256,uint256,uint256,uint256,uint256,uint256) (contracts/lybra/miner/esLBRStatus.sol#55-71) ignores return value by esLBR(miningIncentives.esLBR().mint(msg.sender,_lbrAmount)) (contracts/lybra/miner/esLBRBoost.sol#64)
Reference: https://github.com/crytic/slither/wikil/Detector-Documentation#unused-return
INFODetectors:
Reentrancy in esLBRStatus.setLockStatus(uint256,uint256,uint256,uint256,uint256,uint256) (contracts/lybra/miner/esLBRStatus.sol#5-71):
    External calls:
        - esLBR(miningIncentives.esLBR().burn(msg.sender,_lbrAmount)) (contracts/lybra/miner/esLBRBoost.sol#63)
        - esLBR(miningIncentives.esLBR().mint(msg.sender,_lbrAmount)) (contracts/lybra/miner/esLBRBoost.sol#64)
    Event emitted after the call(s):
        - UserLockStatus(msg.sender,userLockStatus[msg.sender].lockAmount,userLockStatus[msg.sender].duration,_setting.duration,_setting.duration,_setting.miningBoost) (contracts/lybra/miner/esLBRBoost.sol#70)
Reference: https://github.com/crytic/slither/wikil/Detector-Documentation#reentrancy-vulnerabilities-1
INFODetectors:
esLBRStatus.setLockStatus(uint256,uint256,uint256,uint256,uint256,uint256) (contracts/lybra/miner/esLBRStatus.sol#55-71) uses timestamp for comparisons
Reentrancy in esLBRStatus.setLockStatus(uint256,uint256,uint256,uint256,uint256,uint256) (contracts/lybra/miner/esLBRStatus.sol#5-71):
    External calls:
        - esLBR(miningIncentives.esLBR().burn(msg.sender,_lbrAmount)) (contracts/lybra/miner/esLBRBoost.sol#63)
        - esLBR(miningIncentives.esLBR().mint(msg.sender,_lbrAmount)) (contracts/lybra/miner/esLBRBoost.sol#64)
        - miningIncentives.refreshReward(msg.sender) (contracts/lybra/miner/esLBRBoost.sol#68)
    Event emitted after the call(s):
        - UserLockStatus(msg.sender,userLockStatus[msg.sender].lockAmount,userLockStatus[msg.sender].duration,_setting.duration,_setting.duration,_setting.miningBoost) (contracts/lybra/miner/esLBRBoost.sol#70)
Reference: https://github.com/crytic/slither/wikil/Detector-Documentation#reentrancy-vulnerabilities-1
INFODetectors:
esLBRStatus.setLockStatus(uint256,uint256,uint256,uint256,uint256,uint256) (contracts/lybra/miner/esLBRStatus.sol#55-71) uses timestamp for comparisons
Dangerous comparisons:
        - userStatus.lockAmount == 0 & userStatus.lockAmount == 0 (contracts/lybra/miner/esLBRBoost.sol#60)
        - require(bool,string)(esLBR(miningIncentives.esLBR()),balanceOf(_msg.sender)) > userStatus.lockAmount + _lbrAmount,18) (contracts/lybra/miner/esLBRBoost.sol#67)
esLBRStatus.unlock() (contracts/lybra/miner/esLBRStatus.sol#75-78) uses timestamp for comparisons
Dangerous comparisons:
        - userStatus.lockAmount == 0 & userStatus.lockAmount == 0 (contracts/lybra/miner/esLBRStatus.sol#75)
        - userStatus.lockAmount == 0 & userStatus.lockAmount == 0 (contracts/lybra/miner/esLBRStatus.sol#76)
        - userStatus.lockAmount == 0 & userStatus.lockAmount == 0 (contracts/lybra/miner/esLBRStatus.sol#77)
        - block.timestamp > finish (contracts/lybra/miner/esLBRStatus.sol#10)
    Reference: https://github.com/crytic/slither/wikil/Detector-Documentation#block-timestamp
INFODetectors:
Pragma version 0.8.17 (contracts/lybra/interfaces/IMiningIncentives.sol#10) allows old versions
Pragma version 0.8.17 (contracts/lybra/interfaces/ITestIBR.sol#2) allows old versions
Pragma version 0.8.17 (contracts/lybra/miner/estLBR.sol#3) allows old versions
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slitherin/blob/master/docs/magic_number.ad
INFODetectors:
Function IMiningIncentives.esLBR() (contracts/lybra/interfaces/IMiningIncentives.sol#10) is not in mixedCase
Contract esLBRBoost (contracts/lybra/miner/esLBRBoost.sol#10-116) is not in CapWords
State variable esLBRStatus (contracts/lybra/miner/esLBRStatus.sol#10-119) is not in CapWords
Reference: https://github.com/crytic/slither/wikil/Detector-Documentation#solidity-naming-conventions
INFODetectors:
Function esLBRBoost.constructor(address) (contracts/lybra/miner/esLBRBoost.sol#15-41) contains magic numbers: 2592000, 7770000, 15552000, 31536000
INFODetectors:
esLBRStatus.miningIncentives (contracts/lybra/miner/esLBRStatus.sol#13) should be immutable
Reference: https://github.com/crytic/slither/wikil/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFODetectors:
In a function esLBRStatus.setLockStatus(uint256,uint256,uint256,uint256,uint256,uint256) (contracts/lybra/miner/esLBRStatus.sol#5-41) variable esLBRStatus.esLBRBlockSettings (contracts/lybra/miner/esLBRStatus.sol#11) is read multiple times
In a function esLBRStatus.setLockStatus(uint256,uint256,uint256,uint256,uint256,uint256) (contracts/lybra/miner/esLBRStatus.sol#5-71) variable esLBRStatus.miningIncentives (contracts/lybra/miner/esLBRStatus.sol#13) is read multiple times
In a function esLBRStatus.getRewardsLocked(address) (contracts/lybra/miner/esLBRStatus.sol#110-115) variable esLBRStatus.miningIncentives (contracts/lybra/miner/esLBRStatus.sol#13) is read multiple times
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/multiple_storage_read.md

```

```

stakerewardPoolOnArbi.sol
INFO:Detectors:
Function StakingRewardsOnArbi.updateReward(address) (contracts/lybra/miner/stakerewardPoolOnArbi.sol#50-60) is an unprotected initializer.
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/unprotected_initializer.md
INFO:Detectors:
StakingRewardsOnArbi.notifyRewardAmount(uint256) (contracts/lybra/miner/stakerewardPoolOnArbi.sol#117-130) performs a multiplication on the result of a division:
    - rewardRatio = _amount / duration (contracts/lybra/miner/stakerewardPoolOnArbi.sol#119)
    - reward = (amount * (block.timestamp - block.timestamp)) * rewardRatio (contracts/lybra/miner/stakerewardPoolOnArbi.sol#121)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
Reentrancy in StakingRewardsOnArbi.stake(uint256) (contracts/lybra/miner/stakerewardPoolOnArbi.sol#77-83):
    External calls:
        - stakingToken.safeTransferFrom(msg.sender,address(this),_amount) (contracts/lybra/miner/stakerewardPoolOnArbi.sol#79)
    State variables written after the call(s):
        - balanceOf[msg.sender] += _amount (contracts/lybra/miner/stakerewardPoolOnArbi.sol#80)
    StakingRewardsOnArbi.balanceOf() (contracts/lybra/miner/stakerewardPoolOnArbi.sol#81) can be used in cross function reentrances:
        - StakingRewardsOnArbi.withdraw(uint256) (contracts/lybra/miner/stakerewardPoolOnArbi.sol#83)
        - StakingRewardsOnArbi.earned(address) (contracts/lybra/miner/stakerewardPoolOnArbi.sol#95-97)
        - StakingRewardsOnArbi.stake(uint256) (contracts/lybra/miner/stakerewardPoolOnArbi.sol#177-183)
        - StakingRewardsOnArbi.withdraw(uint256) (contracts/lybra/miner/stakerewardPoolOnArbi.sol#186-92)
    totalsupply (contracts/lybra/miner/stakerewardPoolOnArbi.sol#183) can be used in cross function reentrances:
        - StakingRewardsOnArbi.rewardPerToken() (contracts/lybra/miner/stakerewardPoolOnArbi.sol#186-194)
        - StakingRewardsOnArbi.stake(uint256) (contracts/lybra/miner/stakerewardPoolOnArbi.sol#177-183)
        - StakingRewardsOnArbi.totalsupply() (contracts/lybra/miner/stakerewardPoolOnArbi.sol#186-192)
        - StakingRewardsOnArbi.withdraw(uint256) (contracts/lybra/miner/stakerewardPoolOnArbi.sol#186-92)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
Reentrancy in StakingRewardsOnArbi.stake(uint256) (contracts/lybra/miner/stakerewardPoolOnArbi.sol#77-83):
    External calls:
        - stakingToken.safeTransferFrom(msg.sender,address(this),_amount) (contracts/lybra/miner/stakerewardPoolOnArbi.sol#79)
    Event emitted after the call(s):
        - emit StakingRewardsOnArbi.onRewardPaid(address,uint256) (contracts/lybra/miner/stakerewardPoolOnArbi.sol#82)
    Reentrancy in StakingRewardsOnArbi.setRewardDuration(uint256) (contracts/lybra/miner/stakerewardPoolOnArbi.sol#83-92):
        - External calls:
            - stakingToken.safeTransfer(msg.sender,_amount) (contracts/lybra/miner/stakerewardPoolOnArbi.sol#90)
    Event emitted after the call(s):
        - emit StakingRewardsOnArbi.onRewardPaid(address,uint256) (contracts/lybra/miner/stakerewardPoolOnArbi.sol#91)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
StakingRewardsOnArbi.getReward(address,bytes) (contracts/lybra/miner/stakerewardPoolOnArbi.sol#100-108) uses timestamp for comparisons
    - require(bool,string)(FinishTime < block.timestamp,reward.duration not finished) (contracts/lybra/miner/stakerewardPoolOnArbi.sol#112)
StakingRewardsOnArbi.setRewardDuration(uint256) (contracts/lybra/miner/stakerewardPoolOnArbi.sol#111-114) uses timestamp for comparisons
    - require(bool,string)(reward.duration != 0,reward ratio = 0) (contracts/lybra/miner/stakerewardPoolOnArbi.sol#125)
StakingRewardsOnArbi.setRewardDuration(uint256) (contracts/lybra/miner/stakerewardPoolOnArbi.sol#132-136) uses timestamp for comparisons
    - require(bool,string)(x < y) (contracts/lybra/miner/stakerewardPoolOnArbi.sol#133)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
SetEvent Function StakingRewardsOnArbi.setRewardDuration(uint256) (contracts/lybra/miner/stakerewardPoolOnArbi.sol#111-114) does not emit an event
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/event_setter.md
INFO:Detectors:
Pragma version=0.8.17 (contracts/lybra/interfaces/les8R.sol#2) allows old versions
Pragma version=0.8.17 (contracts/lybra/miner/stakerewardPoolOnArbi.sol#2) allows old versions
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Parameter StakingRewardsOnArbi.stake(uint256), _amount (contracts/lybra/miner/stakerewardPoolOnArbi.sol#177) is not in mixedCase
Parameter StakingRewardsOnArbi.earned(address), _address (contracts/lybra/miner/stakerewardPoolOnArbi.sol#195) is not in mixedCase
Parameter StakingRewardsOnArbi.learnedAddress(), address (contracts/lybra/miner/stakerewardPoolOnArbi.sol#95) is not in mixedCase
Parameter StakingRewardsOnArbi.setRewardDuration(uint256), _duration (contracts/lybra/miner/stakerewardPoolOnArbi.sol#111) is not in mixedCase
Parameter StakingRewardsOnArbi.setRewardDuration(uint256), _amount (contracts/lybra/miner/stakerewardPoolOnArbi.sol#117) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
In a function StakingRewardsOnArbi.notifyRewardAmount(uint256) (contracts/lybra/miner/stakerewardPoolOnArbi.sol#117-130) variable StakingRewardsOnArbi.duration (contracts/lybra/miner/stakerewardPoolOnArbi.sol#17) is read multiple times
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/multiple_storage_read.md

```

stakerewardV2pool.sol

```

stakerewardV2pool.sol
INFO:Detectors:
Function StakingRewardsV2.updateReward(address) (contracts/lybra/miner/stakerewardV2pool.sol#60-70) is an unprotected initializer.
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/unprotected_initializer.md
INFO:Detectors:
Reentrancy in StakingRewardsV2.stake(uint256) (contracts/lybra/miner/stakerewardV2pool.sol#80-93):
    External calls:
        - stakingToken.safeTransferFrom(msg.sender,address(this),_amount) (contracts/lybra/miner/stakerewardV2pool.sol#89)
    State variables written after the call(s):
        - balanceOf[msg.sender] += _amount (contracts/lybra/miner/stakerewardV2pool.sol#90)
    StakingRewardsV2.balanceOf() (contracts/lybra/miner/stakerewardV2pool.sol#143) can be used in cross function reentrances:
        - StakingRewardsV2.stake(uint256) (contracts/lybra/miner/stakerewardV2pool.sol#83)
        - StakingRewardsV2.earned(address) (contracts/lybra/miner/stakerewardV2pool.sol#109-111)
        - StakingRewardsV2.withdraw(uint256) (contracts/lybra/miner/stakerewardV2pool.sol#87-93)
        - StakingRewardsV2.withdraw(uint256) (contracts/lybra/miner/stakerewardV2pool.sol#95-102)
        - totalsupply += _amount (contracts/lybra/miner/stakerewardV2pool.sol#93)
    StakingRewardsV2.totalSupply() (contracts/lybra/miner/stakerewardV2pool.sol#141) can be used in cross function reentrances:
        - StakingRewardsV2.stake(uint256) (contracts/lybra/miner/stakerewardV2pool.sol#84)
        - StakingRewardsV2.earned(address) (contracts/lybra/miner/stakerewardV2pool.sol#109-111)
        - StakingRewardsV2.withdraw(uint256) (contracts/lybra/miner/stakerewardV2pool.sol#87-93)
        - StakingRewardsV2.totalSupply() (contracts/lybra/miner/stakerewardV2pool.sol#41)
        - StakingRewardsV2.withdraw(uint256) (contracts/lybra/miner/stakerewardV2pool.sol#96-102)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
StakingRewardsV2.getReward() (contracts/lybra/miner/stakerewardV2pool.sol#114-121) ignores return value by rewardsToken.mint(msg.sender,reward) (contracts/lybra/miner/stakerewardV2pool.sol#118)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
Reentrancy in StakingRewardsV2.getReward() (contracts/lybra/miner/stakerewardV2pool.sol#114-121):
    External calls:
        - rewardsToken.mint(msg.sender,reward) (contracts/lybra/miner/stakerewardV2pool.sol#118)
    Event emitted after the call(s):
        - emit StakingRewardsV2.onRewardPaid(address,uint256) (contracts/lybra/miner/stakerewardV2pool.sol#119)
    Reentrancy in StakingRewardsV2.stake(uint256) (contracts/lybra/miner/stakerewardV2pool.sol#80-93):
        - External calls:
            - stakingToken.safeTransferFrom(msg.sender,address(this),_amount) (contracts/lybra/miner/stakerewardV2pool.sol#89)
    Event emitted after the call(s):
        - emit StakingRewardsV2.onRewardPaid(address,uint256) (contracts/lybra/miner/stakerewardV2pool.sol#96-102)
    External calls:
        - stakingToken.safeTransferFrom(msg.sender,_amount) (contracts/lybra/miner/stakerewardV2pool.sol#100)
    Event emitted after the call(s):
        - emit StakingRewardsV2.onRewardPaid(address,uint256) (contracts/lybra/miner/stakerewardV2pool.sol#101)
    WithdrawalToken msg.sender.transfer(block.timestamp) (contracts/lybra/miner/stakerewardV2pool.sol#101)
    Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
StakingRewardsV2.getReward() (contracts/lybra/miner/stakerewardV2pool.sol#114-121) uses timestamp for comparisons
    - reward > 0 (contracts/lybra/miner/stakerewardV2pool.sol#116)
StakingRewardsV2.setRewardDuration(uint256) (contracts/lybra/miner/stakerewardV2pool.sol#124-128) uses timestamp for comparisons
    - require(bool,string)(FinishTime < block.timestamp,reward.duration not finished) (contracts/lybra/miner/stakerewardV2pool.sol#125)
StakingRewardsV2.notifyRewardAmount(uint256) (contracts/lybra/miner/stakerewardV2pool.sol#137-150) uses timestamp for comparisons
    - require(bool,string)(reward.duration != 0,reward ratio = 0) (contracts/lybra/miner/stakerewardV2pool.sol#145)
StakingRewardsV2._minInitialTime(uint256) (contracts/lybra/miner/stakerewardV2pool.sol#152-154) uses timestamp for comparisons
    - require(bool,string)(x < y) (contracts/lybra/miner/stakerewardV2pool.sol#153)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Pragma version=0.8.17 (contracts/lybra/interfaces/les8R.sol#2) allows old versions
Pragma version=0.8.17 (contracts/lybra/miner/stakerewardV2pool.sol#2) allows old versions
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Parameter StakingRewardsV2.stake(uint256), _amount (contracts/lybra/miner/stakerewardV2pool.sol#87) is not in mixedCase
Parameter StakingRewardsV2.withdraw(uint256), _amount (contracts/lybra/miner/stakerewardV2pool.sol#106) is not in mixedCase

```

```

Parameter StakingRewardsV2.getBoost(address)_account (contracts/lybra/miner/stakerewardv2pool.sol#180) is not in mixedCase
Parameter StakingRewardsV2earned(address)_account (contracts/lybra/miner/stakerewardv2pool.sol#180) is not in mixedCase
Parameter StakingRewardsV2.setRewardsDuration(uint256)_duration (contracts/lybra/miner/stakerewardv2pool.sol#124) is not in mixedCase
Parameter StakingRewardsV2.setStash(address)_host (contracts/lybra/miner/stakerewardv2pool.sol#111) is not in mixedCase
Parameter StakingRewardsV2.setStashAmount(uint256)_amount (contracts/lybra/miner/stakerewardv2pool.sol#137) is not in mixedCase
Reference: https://github.com/crytic/slither/wikil/Detector--Documentation-to-solidity-naming-conventions
INFODetectors:
Variable StakingRewardsV2.constructor(address,address,address) _rewardsToken (contracts/lybra/miner/stakerewardv2pool.sol#53) is too similar to StakingRewardsV2.rewardsToken (contracts/lybra/miner/stakerewardv2pool.sol#21)
Reference: https://github.com/crytic/slither/wikil/Detector--DocumentationVariableNames-too-similar
INFODetectors:
Function StakingRewardsV2.getBoost(address) (contracts/lybra/miner/stakerewardv2pool.sol#104-106) contains magic number: 100
Function StakingRewardsV2.earned(address) (contracts/lybra/miner/stakerewardv2pool.sol#109-111) contains magic number: 1e38
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/magic_number.md
INFODetectors:
Function StakingRewardsV2.notifyRewardAmount(uint256) (contracts/lybra/miner/stakerewardv2pool.sol#117-150) variable StakingRewardsV2.duration (contracts/lybra/miner/stakerewardv2pool.sol#25) is read multiple times
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/multiple_storage_read.md

LybraRETHVault.sol
INFODetectors:
LybraUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#197-214) ignores return value by PeSDO.transferFrom(_provider,address(configurator),totalFee) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#204)
LybraUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#197-214) ignores return value by PeSDO.transferFrom(_provider,address(configurator),amount) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#204)
LybraUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#197-214)
Reference: https://github.com/crytic/slither/wikil/Detector--Documentationto-solidity-naming-conventions
INFODetectors:
Function StakingRewardsV2.getBoost(address) (contracts/lybra/miner/stakerewardv2pool.sol#104-106) contains magic number: 100
Function StakingRewardsV2.earned(address) (contracts/lybra/miner/stakerewardv2pool.sol#109-111) contains magic number: 1e38
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/magic_number.md
INFODetectors:
Function StakingRewardsV2.notifyRewardAmount(uint256) (contracts/lybra/miner/stakerewardv2pool.sol#117-150) variable StakingRewardsV2.duration (contracts/lybra/miner/stakerewardv2pool.sol#25) is read multiple times
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/multiple_storage_read.md

Reentrancy in LybraUSDVaultBase._mintPeUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#178-190):
External calls:
- configurator.refreshMinReward(_provider) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#182)
State variables written after the call(s):
- borrow[_onBehalfOf] = amount (contracts/lybra/pools/base/LybraUSDVaultBase.sol#184)
LybraUSDVaultBase._depositAsset(_onBehalfOf) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#182) can be used in cross function reentrancies:
- LybraUSDVaultBase._mintPeUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#178-190)
- LybraUSDVaultBase._newFee(address) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#241-243)
LybraUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#197-214)
LybraUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#197-259)
LybraUSDVaultBase._rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#160-173)
LybraUSDVaultBase._rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#178-190)
Reentrancy in LybraUSDVaultBase._mintPeUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#178-190):
External calls:
- PeSDO.mint(_onBehalfOf,_mintAmount) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#182)
State variables written after the call(s):
- poolTotalCirculation _mintAmount (contracts/lybra/pools/base/LybraUSDVaultBase.sol#187)
LybraUSDVaultBase._depositAsset(_onBehalfOf) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#182) can be used in cross function reentrancies:
- LybraUSDVaultBase._mintPeUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#178-190)
- LybraUSDVaultBase._newFee(address) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#241-243)
LybraUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#197-214)
LybraUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#261-263)
Reentrancy in LybraUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#197-214):
External calls:
- configurator.refreshMinReward(_onBehalfOf) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#182)
- PeSDO.transferFrom(_provider,address(configurator),totalFee) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#204)
State variables written after the call(s):
- PeSDO.burn(_provider,_amount - totalFee) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#205)
- _onBehalfOf = amount - totalFee (contracts/lybra/pools/base/LybraUSDVaultBase.sol#206)
LybraUSDVaultBase._borrow (contracts/lybra/pools/base/LybraUSDVaultBase.sol#178-190)
LybraUSDVaultBase._mintPeUSD(address,uint256,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#178-190)
- LybraUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#197-214)
LybraUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#257-259)
LybraUSDVaultBase._rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#160-173)
Reentrancy in LybraUSDVaultBase._liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#123-149):
External calls:
- repay(provider,_onBehalfOf,_peSDOAmount) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#132)
- configurator.refreshMinReward(_onBehalfOf) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#188)
- PeSDO.transferFrom(_provider,address(configurator),totalFee) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#204)
- PeSDO.burn(_provider,_amount - totalFee) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#205)
- PeSDO.transferFrom(_provider,address(configurator),amount) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#210)
- configurator.distributeRewards() (contracts/lybra/pools/base/LybraUSDVaultBase.sol#212)
State variables written after the call(s):
- depositedAsset[_provider] = collateralAmount (contracts/lybra/pools/base/LybraUSDVaultBase.sol#170)
LybraUSDVaultBase._depositAsset (contracts/lybra/pools/base/LybraUSDVaultBase.sol#172) can be used in cross function reentrancies:
- LybraUSDVaultBase._checkHealth(address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#229-232)
- LybraUSDVaultBase._withdraw(address,address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#216-224)
LybraUSDVaultBase._depositAssetToMint(uint256,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#159-69)
LybraUSDVaultBase._depositAssetToMint(uint256,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#123-149)
LybraUSDVaultBase._liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#123-149)
LybraUSDVaultBase._rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#160-173)
Reentrancy in LybraUSDVaultBase._rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#160-173):
External calls:
- repay(msg.sender,_peSDOAmount) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#167)
- configurator.refreshMinReward(_onBehalfOf) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#188)
- PeSDO.transferFrom(_provider,address(configurator),totalFee) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#204)
- PeSDO.burn(_provider,_amount - totalFee) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#205)
- PeSDO.transferFrom(_provider,address(configurator),amount) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#210)
- configurator.distributeRewards() (contracts/lybra/pools/base/LybraUSDVaultBase.sol#212)
State variables written after the call(s):
- depositedAsset[_provider] = collateralAmount (contracts/lybra/pools/base/LybraUSDVaultBase.sol#170)
LybraUSDVaultBase._checkHealth(address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#229-232)
- LybraUSDVaultBase._withdraw(address,address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#216-224)
- LybraUSDVaultBase._depositAssetToMint(uint256,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#159-69)
LybraUSDVaultBase._depositAssetToMint(uint256,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#123-149)
LybraUSDVaultBase._liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#123-149)
LybraUSDVaultBase._rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#160-173)
Reference: https://github.com/crytic/slither/wikil/Detector--DocumentationReentrancy-vulnerabilities-1
INFODetectors:
LybraUSDVaultBase._liquidation(address,address,uint256,reward2keep) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#141) is a local variable never initialized
Reference: https://github.com/crytic/slither/wikil/Detector--DocumentationLiquidationReward2keep-local-variables
INFODetectors:
LybraUSDVaultBase._mintPeUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#178-190) ignores return value by PeSDO.mint(_onBehalfOf,_mintAmount) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#180)
LybraUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#197-214) ignores return value by PeSDO.burn(_provider,_amount - totalFee) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#205)
INFODetectors:
Reentrancy in LybraUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#197-214):
External calls:
- configurator.refreshMinReward(_onBehalfOf) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#198)
State variables written after the call(s):
- _updateDebt(_onBehalfOf) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#199)
- _updateDebt(_onBehalfOf) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#200)
- _feeStored[_onBehalfOf] = 0 (contracts/lybra/pools/base/LybraUSDVaultBase.sol#201)
- _feeStored[_onBehalfOf] = totalFee - amount (contracts/lybra/pools/base/LybraUSDVaultBase.sol#200)
- _updateDebt(_onBehalfOf) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#199)
- _debtTimestamp[_onBehalfOf] = block.timestamp (contracts/lybra/pools/base/LybraUSDVaultBase.sol#202)
Reentrancy in LybraUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#197-214):
External calls:
- configurator.refreshMinReward(_onBehalfOf) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#198)
- PeSDO.transferFrom(_provider,address(configurator),totalFee) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#204)
- PeSDO.burn(_provider,_amount - totalFee) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#205)
State variables written after the call(s):
- _collateralAsset._safeTransferFrom(msg.sender,address(this),assetAmount) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#61)
- depositedAssets[_sender] += assetAmount (contracts/lybra/pools/base/LybraUSDVaultBase.sol#63)
Reentrancy in LybraUSDVaultBase._depositEther(uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#143):
External calls:
- RocketDepositPool.rocketStorage.getAddress(keccak256(0xbytes)) (abi.encodePacked(contract.address,rocketDepositPool))).deposit(value: msg.value) (contracts/lybra/pools/lybraRETHVault.sol#34)
State variables written after the call(s):
- depositedAssets[_sender] += balance - preBalance (contracts/lybra/pools/lybraRETHVault.sol#36)
Reference: https://github.com/crytic/slither/wikil/Detector--DocumentationReentrancy-vulnerabilities-2
INFODetectors:
Reentrancy in LybraUSDVaultBase._mintPeUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#178-190):
External calls:
- configurator.refreshMinReward(_onBehalfOf) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#182)
- PeSDO.mint(_onBehalfOf,_mintAmount) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#186)
Event emitted after the call(s):
- Mint(_provider,_onBehalfOf,_mintAmount,block.timestamp) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#189)
Reentrancy in LybraUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#197-214):
External calls:
- configurator.refreshMinReward(_onBehalfOf) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#198)
- PeSDO.transferFrom(_provider,address(configurator),totalFee) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#204)
- PeSDO.burn(_provider,_amount - totalFee) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#205)

```

LybraStETHVault.sol

```

- PadUSD.transferFrom(_provider,address(configurator),amount) (contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#210)
- configurator.distributeRewards() (contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#212)
Event emitted after the call(s):
- Burn(_provider,_onBehalfOf,amount,block.timestamp) (contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#220-224)
Reentrancy in LybraPadUSDVaultBase.withdraw(address,address,uint256) (contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#226-228):
    - withdraw(_call)
        - collateralAsset.safeTransfer(_onBehalfOf, amount) (contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#219)
Event emitted after the call(s):
    - withdrawAsset(_provider,address(collateralAsset),_onBehalfOf,_amount,block.timestamp) (contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#223)
Reentrancy in LybraPadUSDVaultBase.depositAssets(uint256,uint256) (contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#69-69):
    - External calls:
        - collateralAsset.safeTransferFrom(msg.sender,address(this),assetAmount) (contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#61)
        - mintPadUSD(msg.sender,msg.sender,minAmount,assetPrice) (contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#66)
        - Mint(_provider,_onBehalfOf,minAmount,assetPrice) (contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#189)
            - minAmount(msg.sender,minAmount,assetPrice) (contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#182)
            - PadUSD.mint(_onBehalfOf,minAmount) (contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#188)
Event emitted after the call(s):
    - Mint(_provider,_onBehalfOf,minAmount) (contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#186)
Reentrancy in LybraPadUSDVaultBase.depositEther(uint256) (contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#43-43):
    - External calls:
        - iRocketDepositPool(wedgeStorage,getBytes256(pool).abi.encodePacked(contract.address,rocketDepositPool))) (pool.deposit(value: msg.value()) (contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#34)
        - mintPadUSD(msg.sender,msg.sender,minAmount,assetPrice) (contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#69)
            - etherPadUSD.fetchPrice() (contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#249)
            - configurator.refreshMinReward(_provider) (contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#182)
            - PadUSD.mint(_onBehalfOf,minAmount) (contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#186)
    - External calls:
        - iRocketDepositPool(wedgeStorage,getBytes256(pool).abi.encodePacked(contract.address,rocketDepositPool))) (pool.deposit(value: msg.value()) (contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#34)
Event emitted after the call(s):
    - depositEther(msg.sender,address(collateralAsset),msg.value,balance - preBalance,block.timestamp) (Contracts/lybra/pools/LybraRETHVault.sol#42)
    - Mint(_provider,_onBehalfOf,minAmount,assetPrice) (Contracts/lybra/pools/LybraRETHVault.sol#189)
        - minAmount(msg.sender,minAmount,assetPrice) (Contracts/lybra/pools/LybraRETHVault.sol#182)
        - PadUSD.mint(_onBehalfOf,minAmount) (Contracts/lybra/pools/LybraRETHVault.sol#188)
Reentrancy in LybraPadUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#123-149):
    - External calls:
        - repayProvider(_onBehalfOf,peudoAmount) (contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#122)
            - repayProvider.refetchMinReward(_onBehalfOf) (Contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#108)
                - PadUSD.transferFrom(_provider,address(configurator),totalFee) (contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#204)
            - PadUSD.burn(_provider,amount - totalFee) (contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#205)
            - PadUSD.burn(_provider,minAmount,assetPrice) (contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#210)
            - assetPrice(_provider,minAmount) (contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#181)
        - collateralAsset.safeTransfer(msg.sender,rewardKeeper) (contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#145)
        - collateralAsset.safeTransfer(_provider,reducedAsset - rewardKeeper) (contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#147)
Event emitted after the call(s):
    - Burn(_provider,_onBehalfOf,peudoAmount,reducedAsset,rewardKeeper,fals,block.timestamp) (Contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#148)
Reentrancy in LybraPadUSDVaultBase.rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#160-173):
    - External calls:
        - repay(msg.sender,_provider,peudoAmount) (Contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#167)
            - repayProvider.refetchMinReward(_onBehalfOf) (Contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#108)
                - PadUSD.transferFrom(_provider,address(configurator),totalFee) (Contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#204)
            - PadUSD.burn(_provider,amount - totalFee) (Contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#205)
            - PadUSD.transferFrom(_provider,address(configurator),amount) (Contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#210)
            - configurator.distributeRewards() (Contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#182)
        - collateralAsset.safeTransfer(_provider,reducedAsset - rewardKeeper) (Contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#171)
    - External calls:
        - Burn(_provider,_onBehalfOf,sender,collateralAmount) (Contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#172)
Event emitted after the call(s):
    - RigidRedemption(msg.sender,_provider,peudoAmount,block.timestamp) (Contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#172)
Reference: https://github.com/crytic/slither/wiki/Detector#reentrancy-vulnerabilities
INFODetectors:
LybraPadUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#123-149) uses timestamp for comparisons
Dangerous comparisions:
    - require(bool,string)(_onBehalfOfCollateralRatio < configurator.getBadCollateralRatio(address(this)),borrows collateral ratio should be below badCollateralRatio) (Contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#126)
    - _onBehalfOfCollateralRatio >= 1e20 && _onBehalfOfCollateralRatio < 1e16 (Contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#134)
    - msg.sender <= 1e10 && _onBehalfOf <= 1e10 && _onBehalfOfCollateralRatio >= 1e10 (Contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#143)
LybraPadUSDVaultBase.rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#160-173) uses timestamp for comparisons
Dangerous comparisions:
    - require(bool,string)(borrowed[_provider] >= peudoAmount) (Contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#162)
        - require(bool,string)(_providerCollateralRatio >= 1e20 || 1e20 >= _provider) (Contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#166)
    - require(bool,string)(poolTotalCirculation + mintAmount <= configurator.mintVaultMaxSupply(address(this)),ESL) (Contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#179)
LybraPadUSDVaultBase.liquidation(address,address,uint256) (Contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#123-124) uses timestamp for comparisons
Dangerous comparisions:
    - amount >= totalFee (Contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#122)
        - borrowed[_provider] + totalFee >= _amount (Contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#201)
LybraPadUSDVaultBase.rigidRedemption(address,address,uint256) (Contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#160-224) uses timestamp for comparisons
Dangerous Comparisons:
    - getBorrowedOf(_provider) > 0 (Contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#220)
LybraPadUSDVaultBase._checkHealth(address,uint256) (Contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#229-232) uses timestamp for comparisons
LybraPadUSDVaultBase._updateAddress() (Contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#234-239) uses timestamp for comparisons
Dangerous comparions:
    - block.timestamp > updatedAt [user] (Contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#235)
Reference: https://github.com/crytic/slither/wiki/Detector#Documentation-for-block-timestamp
INFODetectors:
Function LybraRETHVault.getAssetPrice() (Contracts/lybra/pools/LybraRETHVault.sol#45-47) has a dubious typecast: IRETH<--address
Function LybraRETHVault.getSafeEtherExchangeRate() (Contracts/lybra/pools/LybraRETHVault.sol#49-51) has a dubious typecast: IRETH<--address
External calls:
    - require(bool,string)(msg.sender == 0x0) (Contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#160)
Reference: https://github.com/crytic/slither/wiki/Detector#Documentation-for-typecast-and-typeconversion
INFODetectors:
Pragma version@0.8.17 (Contracts/lyra/interfaces/IEUDSol.sol#2) allows old versions
pragma 0.8.17 not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector#Documentation-for-deployment
INFODetectors:
Parameter IEUDSol.getSharesByMintedUSD(uint256) (Contracts/lyra/pools/base/LyraPadUSDVaultBase.sol#32) is not in mixedCase
Variable LybraPadUSDVaultBase.PadUSD (Contracts/lyra/pools/base/LyraPadUSDVaultBase.sol#10) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector#Documentation-for-mixed-case-naming-conventions
INFODetectors:
Function LybraPadUSDVaultBase.depositAsset(uint256,uint256) (Contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#59-69) contains magic number: 100000000000000000000
Function LybraPadUSDVaultBase.liquidation(address,address,uint256) (Contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#123-149) contains magic numbers: 100, 1e20, 1e20, 1e20, 100, 1e19, 11, 1e20, 100
Function LybraPadUSDVaultBase.rigidRedemption(address,uint256,uint256) (Contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#160-173) contains magic numbers: 100, 100, 100
Function LybraPadUSDVaultBase.liquidation(address,address,uint256) (Contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#123-124) contains magic number: 100
Function LybraPadUSDVaultBase.depositEther(uint256) (Contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#241-243) contains magic number: 100000000000000000000
Function LybraRETHVault.depositEther(uint256) (Contracts/lybra/pools/LybraRETHVault.sol#181-183) contains magic number: 100000000000000000000
Reference: https://github.com/pessimistic-io/slither/blob/master/docs/magic_number.md
INFODetectors:
In a function LybraRETHVault.depositEther(uint256) (Contracts/lybra/pools/LybraRETHVault.sol#181-183) variable LybraPadUSDVaultBase.collateralAsset (Contracts/lybra/pools/base/LybraPadUSDVaultBase.sol#17) is read multiple times
Reference: https://github.com/pessimistic-io/slither/wiki/Detector#Documentation-for-multiple-storage-read_md

```

AUTOMATED TESTING

AUTOMATED TESTING

```

Reentrancy in LybraStETHVault.depositEtherToMint(uint256) (contracts/lybra/pools/lybraStETHVault.sol#37-52):
    External calls:
        - sharesAmount = IIdo(address(collateralAsset)).submit(value: msg.value)(address(configurator)) (contracts/lybra/pools/lybraStETHVault.sol#40)
        State variables written after the call(s):
            - depositEtherToMint(msg.sender, msg.value) (contracts/lybra/pools/lybraStETHVault.sol#44)
            - depositEtherToMint(msg.sender, msg.value) (contracts/lybra/pools/lybraStETHVault.sol#45)
            - totalDepositedAssets += msg.value (contracts/lybra/pools/lybraStETHVault.sol#43)
    Reentrancy in LybraUSDVaultBase.liquidation(address, address, uint256) (contracts/lybra/pools/base/lybraUSDVaultBase.sol#148-178):
        External calls:
            - repay(msg.sender, onBehalfOf, eUSDAmount) (contracts/lybra/pools/base/lybraUSDVaultBase.sol#157)
            - eUSD.burn_(msg.sender, amount) (contracts/lybra/pools/base/lybraUSDVaultBase.sol#183)
            - eUSD.burn_(msg.sender, amount) (contracts/lybra/pools/base/lybraUSDVaultBase.sol#183)
            - configurator.refreshMinReward_(onBehalfOf) (contracts/lybra/pools/base/lybraUSDVaultBase.sol#284)
        State variables written after the call(s):
            - totalDepositedAssets -= collateralAmount (contracts/lybra/pools/base/lybraUSDVaultBase.sol#245)
            - totalDepositedAssets -= collateralAmount (contracts/lybra/pools/base/lybraUSDVaultBase.sol#245)
    Reentrancy in LybraUSDVaultBase.rigidRedemption(address, uint256, uint256) (contracts/lybra/pools/base/lybraUSDVaultBase.sol#234-248):
        External calls:
            - _repay(msg.sender, onBehalfOf, eUSDAmount) (contracts/lybra/pools/base/lybraUSDVaultBase.sol#241)
            - eUSD.burn_(msg.sender, amount) (contracts/lybra/pools/base/lybraUSDVaultBase.sol#183)
            - eUSD.burn_(msg.sender, amount) (contracts/lybra/pools/base/lybraUSDVaultBase.sol#183)
            - configurator.refreshMinReward_(onBehalfOf) (contracts/lybra/pools/base/lybraUSDVaultBase.sol#284)
        State variables written after the call(s):
            - totalDepositedAssets -= collateralAmount (contracts/lybra/pools/base/lybraUSDVaultBase.sol#245)
            - totalDepositedAssets -= collateralAmount (contracts/lybra/pools/base/lybraUSDVaultBase.sol#245)
    INFO-Detectors:
    Reentrancy in LybraUSDVaultBase.mintUSD(address, address, uint256, uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#263-273):
        External calls:
            configurator.refreshMinReward_(provider) (contracts/lybra/pools/base/lybraUSDVaultBase.sol#265)
            eUSD.burn_(msg.sender, amount) (contracts/lybra/pools/base/lybraUSDVaultBase.sol#183)
            eUSD.burn_(msg.sender, amount) (contracts/lybra/pools/base/lybraUSDVaultBase.sol#183)
        Event emitted after the call(s):
            Mint(msg.sender, onBehalfOf, mintAmount, block.timestamp) (contracts/lybra/pools/base/lybraUSDVaultBase.sol#272)
    Reentrancy in LybraUSDVaultBase._repay(address, address, uint256) (contracts/lybra/pools/base/lybraUSDVaultBase.sol#280-290):
        External calls:
            EUSD.burn_(msg.sender, amount) (contracts/lybra/pools/base/lybraUSDVaultBase.sol#283)
            - configurator.refreshMinReward_(onBehalfOf) (contracts/lybra/pools/base/lybraUSDVaultBase.sol#284)
        Event emitted after the call(s):
            _repay(msg.sender, onBehalfOf, mintAmount, block.timestamp) (contracts/lybra/pools/base/lybraUSDVaultBase.sol#280)
    Reentrancy in LybraUSDVaultBase._mintUSD(address, uint256, uint256) (contracts/lybra/pools/base/lybraUSDVaultBase.sol#70-81):
        External calls:
            - collateralAsset.safeTransferFrom(msg.sender, address(this), assetAmount) (contracts/lybra/pools/base/lybraUSDVaultBase.sol#72)
            - _mintUSD(msg.sender, msg.sender, mintAmount, getAssetPrice()) (contracts/lybra/pools/base/lybraUSDVaultBase.sol#78)
            - eUSD.burn_(msg.sender, amount) (contracts/lybra/pools/base/lybraUSDVaultBase.sol#183)
            - eUSD.mint_(onBehalfOf, mintAmount) (contracts/lybra/pools/base/lybraUSDVaultBase.sol#268)
        Event emitted after the call(s):
            DepositAsset(asset, address(collateralAsset), assetAmount, block.timestamp) (contracts/lybra/pools/base/lybraUSDVaultBase.sol#80)
            - depositAsset(msg.sender, address(collateralAsset), assetAmount, block.timestamp) (contracts/lybra/pools/base/lybraUSDVaultBase.sol#177)
            - Mint(msg.sender, onBehalfOf, mintAmount, getAssetPrice()) (contracts/lybra/pools/base/lybraUSDVaultBase.sol#78)
    Reentrancy in LybraStETHVault._depositEtherToMint(uint256) (contracts/lybra/pools/lybraStETHVault.sol#37-52):
        External calls:
            - sharesAmount = IIdo(address(collateralAsset)).submit(value: msg.value)(address(configurator)) (contracts/lybra/pools/lybraStETHVault.sol#40)
            - _mintUSD(msg.sender, msg.sender, mintAmount, getAssetPrice()) (contracts/lybra/pools/lybraStETHVault.sol#44)
            - etherOracle.fetchPrice() (contracts/lybra/pools/base/lybraStETHVault.sol#181)
            - - configurator.refreshMinReward_(provider) (contracts/lybra/pools/base/lybraStETHVault.sol#265)
            - eUSD.mint_(onBehalfOf, mintAmount) (contracts/lybra/pools/base/lybraUSDVaultBase.sol#268)
        External calls:
            - sharesAmount = IIdo(address(collateralAsset)).submit(value: msg.value)(address(configurator)) (contracts/lybra/pools/lybraStETHVault.sol#40)
        Event emitted after the call(s):
            - DepositEther(msg.sender, address(collateralAsset), msg.value, msg.value, block.timestamp) (contracts/lybra/pools/lybraStETHVault.sol#51)
            - Mint(msg.sender, onBehalfOf, mintAmount, getAssetPrice()) (contracts/lybra/pools/base/lybraStETHVault.sol#77)
            - _mintUSD(msg.sender, msg.sender, mintAmount, getAssetPrice()) (contracts/lybra/pools/base/lybraUSDVaultBase.sol#48)
    Reentrancy in LybraStETHVault.excessIncomeDistribution(uint256) (contracts/lybra/pools/lybraStETHVault.sol#62-96):
        External calls:
            - paymentAmount = realAmount * getAssetPrice() * dutchAuctionDiscountPrice / 10_000 / 1e18 (contracts/lybra/pools/lybraStETHVault.sol#67)
            - etherOracle.fetchPrice() (contracts/lybra/pools/base/lybraStETHVault.sol#181)
            - success >= EUSD.transferFrom(msg.sender, address(configurator), income) (contracts/lybra/pools/lybraStETHVault.sol#71)
            - EUSD.burnShares(msg.sender, sharesAmount) (contracts/lybra/pools/base/lybraStETHVault.sol#182)
            - configurator.distributeRewards() (contracts/lybra/pools/lybraStETHVault.sol#188)
            - EUSD.burnShares(msg.sender, sharesAmount) (contracts/lybra/pools/base/lybraStETHVault.sol#182)
            - FeedDistributionAddress(configurator).income(block.timestamp) (contracts/lybra/pools/lybraStETHVault.sol#90)
        Event emitted after the call(s):
            - FeedDistributionAddress(configurator).paymentAmount(block.timestamp) (contracts/lybra/pools/lybraStETHVault.sol#90)
    Reentrancy in LybraStETHVault.excessIncomeDistribution(uint256) (contracts/lybra/pools/lybraStETHVault.sol#62-96):
        External calls:
            paymentAmount = realAmount * getAssetPrice() * dutchAuctionDiscountPrice / 10_000 / 1e18 (contracts/lybra/pools/lybraStETHVault.sol#67)
            etherOracle.fetchPrice() (contracts/lybra/pools/base/lybraStETHVault.sol#181)
            success >= EUSD.transferFrom(msg.sender, address(configurator), income) (contracts/lybra/pools/lybraStETHVault.sol#71)
            configurator.distributeRewards() (contracts/lybra/pools/lybraStETHVault.sol#188)
            EUSD.burnShares(msg.sender, sharesAmount) (contracts/lybra/pools/base/lybraStETHVault.sol#182)
            configurator.distributeRewards() (contracts/lybra/pools/lybraStETHVault.sol#188)
            - collateralAsset.transfer(msg.sender, realAmount) (contracts/lybra/pools/lybraStETHVault.sol#94)
        Event emitted after the call(s):
            LSDValueCaptured(realAmount, paymentAmount, dutchAuctionDiscountPrice, block.timestamp) (contracts/lybra/pools/lybraStETHVault.sol#95)
    Reentrancy in LybraStETHVault.withdraw(address, address, uint256) (contracts/lybra/pools/base/lybraUSDVaultBase.sol#148-178):
        External calls:
            - _repay(provider, onBehalfOf, eUSDAmount) (contracts/lybra/pools/base/lybraUSDVaultBase.sol#157)
            - EUSD.burn_(msg.sender, amount) (contracts/lybra/pools/base/lybraUSDVaultBase.sol#183)
            - collateralAsset.safeTransferFrom(msg.sender, rewardKeeper) (contracts/lybra/pools/base/lybraUSDVaultBase.sol#173)
            - collateralAsset.safeTransfer(provider, rewardDasset - rewardKeeper) (contracts/lybra/pools/base/lybraUSDVaultBase.sol#175)
        Event emitted after the call(s):
            RigidRedemption(msg.sender, provider, eUSDAmount, collateralAmount, block.timestamp) (contracts/lybra/pools/base/lybraUSDVaultBase.sol#247)
            Withdrawal(msg.sender, amount, withdrawalType) (contracts/lybra/pools/base/lybraUSDVaultBase.sol#213)
    Reentrancy in LybraUSDVaultBase.rigidRedemption(address, address, uint256) (contracts/lybra/pools/base/lybraUSDVaultBase.sol#234-248):
        External calls:
            - _repay(provider, onBehalfOf, eUSDAmount) (contracts/lybra/pools/base/lybraUSDVaultBase.sol#241)
            - EUSD.burn_(msg.sender, amount) (contracts/lybra/pools/base/lybraUSDVaultBase.sol#283)
            - collateralAsset.safeTransferFrom(msg.sender, rewardKeeper) (contracts/lybra/pools/base/lybraUSDVaultBase.sol#284)
            - collateralAsset.safeTransfer(provider, assetAmount - rewardKeeper) (contracts/lybra/pools/base/lybraUSDVaultBase.sol#210)
        Event emitted after the call(s):
            LiquidationRecord(provider, msg.sender, onBehalfOf, eUSDAmount, assetAmount, rewardKeeper, true, block.timestamp) (contracts/lybra/pools/base/lybraUSDVaultBase.sol#212)
    Reentrancy in LybraUSDVaultBase.withdraw(address, withdrawType) (contracts/lybra/pools/base/lybraUSDVaultBase.sol#106):
        External calls:
            - collateralAsset.safeTransferOnBehalfOf_(withdrawal) (contracts/lybra/pools/base/lybraUSDVaultBase.sol#102)
        Event emitted after the call(s):
            Withdrawal(msg.sender, amount, withdrawalType) (contracts/lybra/pools/base/lybraUSDVaultBase.sol#106)
    References: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities
    INFO-Detectors:
    LybraStETHVault.excessIncomeDistribution(uint256) (contracts/lybra/pools/lybraStETHVault.sol#62-96) uses timestamp for comparisons
    Dangerous comparisons:
        - time <= 1000 (contracts/lybra/pools/lybraStETHVault.sol#104)
    LybraUSDVaultBase.depositWithdrawable(address, depositedTime) (contracts/lybra/pools/base/lybraUSDVaultBase.sol#109-111) uses timestamp for comparisons
        - time <= 1000 (contracts/lybra/pools/base/lybraUSDVaultBase.sol#109-111)
    References: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
    INFO-Detectors:
    Function LybraStETHVault.depositEtherToMint(uint256) (contracts/lybra/pools/lybraStETHVault.sol#37-52) has a dubious typecast: IIdo<address>
    References: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
    INFO-Detectors:
    Setter Function LybraStETHVault.setLidoReleaseTime(uint256) (contracts/lybra/pools/lybraStETHVault.sol#25-28) does not emit an event
    References: https://github.com/crytic/slither/blob/master/docs/event.setter.ad
    INFO-Detectors:
    Parameter version<=0.8.17 (contracts/lybra/interfaces/IUSD.sol#2) allows old versions
    Pragma version<=0.8.17 (contracts/lybra/interfaces/IConfigurable.sol#1) allows old versions
    Pragma version<=0.8.17 (contracts/lybra/pools/lybraStETHVault.sol#8) allows old versions
    Pragma version<=0.8.17 (contracts/lybra/pools/lybraStETHVault.sol#8) allows old versions
    Pragma version<=0.8.17 (contracts/lybra/pools/lybraUSDVaultBase.sol#8) allows old versions
    solc-0.8.17 is not requested for compilation
    References: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
    INFO-Detectors:
    Parameter provider<=0x0000000000000000000000000000000000000000 (contracts/lybra/pools/lybraStETHVault.sol#2) is not in mixedCase
    Parameter type<=0x0000000000000000000000000000000000000000 (contracts/lybra/pools/lybraStETHVault.sol#2) is not in mixedCase
    Variable LybraUSDVaultBase.EUSD (contracts/lybra/pools/base/lybraUSDVaultBase.sol#10) is not in mixedCase
    References: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
    INFO-Detectors:
    Function LybraUSDVaultBase.depositAssetSetToMint(uint256, uint256) (contracts/lybra/pools/base/lybraUSDVaultBase.sol#70-81) contains magic number: 100000000000000000000
    Function LybraUSDVaultBase.checkWithdrawal(address, uint256) (contracts/lybra/pools/base/lybraUSDVaultBase.sol#109-111) contains magic numbers: 259200, 999
    Function LybraUSDVaultBase.liquidation(address, address, uint256) (contracts/lybra/pools/base/lybraUSDVaultBase.sol#148-178) contains magic numbers: 100, 1e20, 1e20, 1e19, 11, 1e20, 1e20, 100

```

Function LybraUSDVaultBase.superLiquidation(address,address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#189-213) contains magic numbers: 100, 100, 125, 1e09, 1e09, 1e20
Function LybraUSDVaultBase._rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#197-214) ignores return value by PeUSD.transferFrom(_provider,address(configurator),totalFee) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#189-213)
Function LybraUSDVaultBase._checkHealth(address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#197-224) contains magic number: 100
Function LybraUSDVaultBase._newFee() (contracts/lybra/pools/base/LybraUSDVaultBase.sol#194-306) contains magic number: 86
Function LybraUSDVaultBase._mintPeUSD(address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#197-250) contains magic number: 100000000000000000000000000000000
Function LybraUSDVaultBase._withdrawalAuthorization(uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#197-262) contains magic number: 10
Function LybraETHVault.getOrchAuctionDiscountPrice() (contracts/lybra/pool/lybraETHVault.sol#102-106) contains magic number: 86400, 1000, 10, 10
Function LybraETHVault.siltherConstructorVariables() (contracts/lybra/pools/lybraETHVault.sol#12-114) contains magic number: 43200
References: https://github.com/pessimistic-io/siltherin/blob/master/docs/magic_number.ad

INFO:Detectors:
LybraUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#197-214) ignores return value by PeUSD.transferFrom(_provider,address(configurator),amount) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#189-213)
LybraUSDVaultBase._liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#197-250) ignores return value by PeUSD.transferFrom(_provider,address(configurator),amount) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#189-213)
LybraETHVault.getOrchAuctionDiscountPrice() (contracts/lybra/pool/lybraETHVault.sol#102-106) contains magic number: 86400, 1000, 10, 10
Function LybraETHVault.siltherConstructorVariables() (contracts/lybra/pools/lybraETHVault.sol#12-114) contains magic number: 43200
References: https://github.com/pessimistic-io/siltherin/blob/master/docs/multiple_storage_readed

LybraWbETHVault.sol

INFO:Detectors:
LybraUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#197-214) ignores return value by PeUSD.transferFrom(_provider,address(configurator),totalFee) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#189-213)
LybraUSDVaultBase._liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#197-250) ignores return value by PeUSD.transferFrom(_provider,address(configurator),amount) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#189-213)
LybraETHVault.getOrchAuctionDiscountPrice() (contracts/lybra/pool/lybraETHVault.sol#102-106) contains magic number: 86400, 1000, 10, 10
Function LybraETHVault.siltherConstructorVariables() (contracts/lybra/pools/lybraETHVault.sol#12-114) contains magic number: 43200
References: https://github.com/pessimistic-io/siltherin/blob/master/docs/magic_number.ad

INFO:Detectors:
LybraUSDVaultBase._liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#197-214) performs a multiplication on the result of a division:
onBehalfOfCollateralRatio * (depositedAsset.onBehalfOfF * assetPrice * 100) / getOrchardFee() (contracts/lybra/pools/base/LybraUSDVaultBase.sol#113)
- reducedAsset * onBehalfOfCollateralRatio * 1e20 (contracts/lybra/pools/base/LybraUSDVaultBase.sol#113)
Reference: [https://github.com/cryptic/siltherin/wiki/Detector-Documentationdivide-before-multiply](https://github.com/cryptic/siltherin/wiki/Detector-Dокументationdivide-before-multiply)

INFO:Detectors:
Reentrancy in LybraUSDVaultBase._mintPeUSD(address,address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#178-190):
External calls:
- configurator.refreshMinReward_provider() (contracts/lybra/pools/base/LybraUSDVaultBase.sol#182)
State variables written after the call(s):
LybraUSDVaultBase.borrowed (contracts/lybra/pools/base/LybraUSDVaultBase.sol#223) can be used in cross function reentrancies:
- LybraUSDVaultBase._mintPeUSD(address,address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#178-190)
- LybraUSDVaultBase._newFee() (contracts/lybra/pools/base/LybraUSDVaultBase.sol#194-306) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#197-214)
- LybraUSDVaultBase._getOrchardFee() (contracts/lybra/pools/base/LybraUSDVaultBase.sol#257-259)
- LybraUSDVaultBase._rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#160-173)
Reentrancy in LybraUSDVaultBase._mintPeUSD(address,address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#178-190):
External calls:
- configurator.refreshMinReward_provider() (contracts/lybra/pools/base/LybraUSDVaultBase.sol#182)
- PeUSD.mint_onBehalfOf_mintAmount() (contracts/lybra/pools/base/LybraUSDVaultBase.sol#188)
State variables written after the call(s):
poolTotalLiquidity (contracts/lybra/pools/base/LybraUSDVaultBase.sol#191) can be used in cross function reentrancies:
- LybraUSDVaultBase._mintPeUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#178-190)
- LybraUSDVaultBase._getOrchardFee() (contracts/lybra/pools/base/LybraUSDVaultBase.sol#217-214)
Reentrancy in LybraUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#197-214):
External calls:
- configurator.refreshMinReward_onBehalfOf() (contracts/lybra/pools/base/LybraUSDVaultBase.sol#198)
- PeUSD.transferFrom_provider,address(configurator),totalFee) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#204)
- PeUSD.mint_onBehalfOf_mintAmount() (contracts/lybra/pools/base/LybraUSDVaultBase.sol#205)
State variables written after the call(s):
- borrowed = amount - totalFee (contracts/lybra/pools/base/LybraUSDVaultBase.sol#206)
LybraUSDVaultBase.borrowed (contracts/lybra/pools/base/LybraUSDVaultBase.sol#223) can be used in cross function reentrancies:
- LybraUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#197-214)
- LybraUSDVaultBase._withdraw(address) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#241)
- LybraUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#197-214)
- LybraUSDVaultBase._getOrchardFee() (contracts/lybra/pools/base/LybraUSDVaultBase.sol#257-259)
- LybraUSDVaultBase._rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#160-173)
Reentrancy in LybraUSDVaultBase._liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#123-149):
External calls:
- repay(provider,onBehalfOf,peusdAmount) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#147)
- configurator.refreshMinReward_onBehalfOf() (contracts/lybra/pools/base/LybraUSDVaultBase.sol#198)
- PeUSD.transferFrom_provider,address(configurator),totalFee) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#204)
- PeUSD.burn(provider,amount - totalFee) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#205)
- PeUSD.transferFrom_provider,address(configurator),amount) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#210)
- configurator.distributeRewards() (contracts/lybra/pools/base/LybraUSDVaultBase.sol#121)
State variables written after the call(s):
- depositedAsset.onBehalfOfF (reducedAsset) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#148)
LybraUSDVaultBase.depositAsset (contracts/lybra/pools/base/LybraUSDVaultBase.sol#222) can be used in cross function reentrancies:
- LybraUSDVaultBase._checkHealth(address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#229-232)
- LybraUSDVaultBase.withdraw(address,address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#216-224)
- LybraUSDVaultBase._depositAssetTxFee() (contracts/lybra/pools/base/LybraUSDVaultBase.sol#199-201)
- LybraUSDVaultBase._checkHealth(address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#197-209)
- LybraUSDVaultBase._liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#123-149)
- LybraUSDVaultBase._rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#160-173)
Reentrancy in LybraUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#197-214):
External calls:
- repay(msg.sender.provider,peusdAmount) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#147)
- configurator.refreshMinReward_onBehalfOf() (contracts/lybra/pools/base/LybraUSDVaultBase.sol#198)
- PeUSD.transferFrom_provider,address(configurator),totalFee) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#204)
- PeUSD.burn(provider,amount - totalFee) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#205)
- PeUSD.transferFrom_provider,address(configurator),amount) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#210)
- configurator.distributeRewards() (contracts/lybra/pools/base/LybraUSDVaultBase.sol#121)
State variables written after the call(s):
- depositedAsset.onBehalfOfF (reducedAsset) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#148)
LybraUSDVaultBase.depositAsset (contracts/lybra/pools/base/LybraUSDVaultBase.sol#222) can be used in cross function reentrancies:
- LybraUSDVaultBase._checkHealth(address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#229-232)
- LybraUSDVaultBase.withdraw(address,address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#216-224)
- LybraUSDVaultBase._depositAssetTxFee() (contracts/lybra/pools/base/LybraUSDVaultBase.sol#199-201)
- LybraUSDVaultBase._checkHealth(address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#197-209)
- LybraUSDVaultBase._liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#123-149)
- LybraUSDVaultBase._rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#160-173)
Reference: <https://github.com/cryptic/siltherin/wiki/Detector-DocumentationReentrancy-vulnerabilities-1>

INFO:Detectors:
LybraUSDVaultBase._liquidation(address,address,uint256).reward2keep (contracts/lybra/pools/base/LybraUSDVaultBase.sol#141) is a local variable never initialized
Reference: <https://github.com/cryptic/siltherin/wiki/Detector-DocumentationReentrancy-vulnerabilities-1>

INFO:Detectors:
LybraUSDVaultBase._mintPeUSD(address,address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#178-190) ignores return value by PeUSD.mint_onBehalfOf_mintAmount (contracts/lybra/pools/base/LybraUSDVaultBase.sol#188)
Reference: <https://github.com/cryptic/siltherin/wiki/Detector-DocumentationNused-return>

INFO:Detectors:
Reentrancy in LybraUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#197-214):
External calls:
- configurator.refreshMinReward_onBehalfOf() (contracts/lybra/pools/base/LybraUSDVaultBase.sol#198)
State variables written after the call(s):
- updateBalance(msg.sender,address(onBehalfOf)) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#199)
- feeStoredUser += msgFee(user) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#236)
- feeStored_onBehalfOfF = 0 (contracts/lybra/pools/base/LybraUSDVaultBase.sol#120)
- feeStored_onBehalfOfF = amount - contracts/lybra/pools/base/LybraUSDVaultBase.sol#209
- updateBalance(provider,address(onBehalfOf)) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#199)
- feePaidTimestamp = block.timestamp (contracts/lybra/pools/base/LybraUSDVaultBase.sol#237)
- feePaidTimestamp -= block.timestamp (contracts/lybra/pools/base/LybraUSDVaultBase.sol#237)
Reentrancy in LybraUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#197-214):
External calls:
- configurator.refreshMinReward_onBehalfOf() (contracts/lybra/pools/base/LybraUSDVaultBase.sol#198)
- PeUSD.burn(provider,amount - totalFee) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#205)
- PeUSD.transferFrom_provider,address(configurator),amount) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#206)
State variables written after the call(s):
- collateralAsset.safeTransferFrom(msg.sender,address(onBehalfOf),assetAmount) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#61)
- State variables written after the call(s):
- msgValue == assetAmount (contracts/lybra/pools/base/LybraUSDVaultBase.sol#63)
- PeUSD.burn(provider,amount - totalFee) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#205)
State variables written after the call(s):
- poolTotalLiquidity = amount - totalFee (contracts/lybra/pools/base/LybraUSDVaultBase.sol#207)
- collateralAsset.safeTransferFrom(msg.sender,address(onBehalfOf),assetAmount) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#61)
External calls:
- INETH(address(collateralAsset)).depositWithValue(msg.value,address(configurator)) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#24)

Reference: <https://github.com/cryptic/siltherin/wiki/Detector-DocumentationReentrancy-vulnerabilities-2>

INFO:Detectors:
Reentrancy in LybraUSDVaultBase._mintPeUSD(address,address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#178-190):
External calls:
- PeUSD.mint_onBehalfOf_provider (contracts/lybra/pools/base/LybraUSDVaultBase.sol#182)
- PeUSD.mint_onBehalfOf_mintAmount (contracts/lybra/pools/base/LybraUSDVaultBase.sol#188)
Event emitted after the call(s):
- Mint_provider_onBehalfOf_mintAmount_block.timestamp (contracts/lybra/pools/base/LybraUSDVaultBase.sol#189)
Reentrancy in LybraUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#197-214):
External calls:
- configurator.refreshMinReward_onBehalfOf() (contracts/lybra/pools/base/LybraUSDVaultBase.sol#198)
- PeUSD.transferFrom_provider,address(configurator),totalFee) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#204)
- PeUSD.burn(_provider,amount - totalFee) (contracts/lybra/pools/base/LybraUSDVaultBase.sol#205)

AUTOMATED TESTING

LybraWstETHVault.sol

INFODetectors:

- LybraPeUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#197-210) ignores return value by `PoS0.transferFrom(_provider,address(configurator),amount)` (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#170-190)
- LybraPeUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#197-214) ignores return value by `PoS0.transferFrom(_provider,address(configurator),amount)` (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#170-190)
- Reentrancy in `LybraPeUSDVaultBase._mintPesoUSD(address,address,uint256)` (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#170-190):
 - Event emitted after the call(s): `WithdrawAsset(_provider,address(collateralAsset),_onBehalfOf,_amount,block.timestamp)` (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#220-224)
 - Event emitted after the call(s): `Burn(_provider,_onBehalfOf,_amount,block.timestamp)` (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#213-217)
 - Event emitted after the call(s): `collateralAsset.safeTransferOnBehalfOf(_amount)` (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#219)
 - Event emitted after the call(s): `WithdrawAsset(_provider,address(collateralAsset),_onBehalfOf,_amount,block.timestamp)` (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#223)
- Reentrancy in `LybraPeUSDVaultBase._depositAssetsToMin(uint256,uint256)` (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#169-190):
 - External calls:
 - `collateralAsset.safeTransferFrom(msg.sender,address(this),assetAmount)` (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#161)
 - `_mintPesoUSD(msg.sender,msg.sender,minAmount,assetPrice)` (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#166)
 - `PoS0.mint(_onBehalfOf,minAmount)` (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#162)
 - `PoS0.mint(_onBehalfOf,minAmount)` (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#168)
 - Event emitted after the call(s): `DepositAssets(_asset,address(sender),minAmount,block.timestamp)` (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#68)
 - Event emitted after the call(s): `Mint(_provider,address(_onBehalfOf),msg.sender,minAmount,assetPrice)` (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#189)
 - `minPesoUSD(msg.sender,msg.sender,minAmount,assetPrice)` (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#166)
- Reentrancy in `LybraWstETHVault.depositEtherToMin(uint256)` (contracts/lybra/pools/base/LybraWstETHVault.sol#21-33):
 - External calls:
 - `INETH(address(collateralAsset)).depositValue(msg.value)(address(configurator))` (contracts/lybra/pools/base/LybraWstETHVault.sol#24)
 - `_mintPesoUSD(msg.sender,msg.sender,minAmount,getBytes16(1))` (contracts/lybra/pools/base/LybraWstETHVault.sol#20)
 - `etherToken.fetchPrice()` (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#189)
 - `configurator.refreshMinReward(_provider)` (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#182)
 - External calls:
 - `callValue(_mintPesoUSD(_onBehalfOf,minAmount))` (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#186)
 - Event emitted after the call(s): `INETH(address(collateralAsset)).depositValue(msg.value)(address(configurator))` (contracts/lybra/pools/base/LybraWstETHVault.sol#24)
 - Event emitted after the call(s): `DepositEther(_provider,address(_onBehalfOf),msg.value,block.timestamp)` (contracts/lybra/pools/base/LybraWstETHVault.sol#32)
 - Event emitted after the call(s): `Mint(_provider,address(_onBehalfOf),msg.sender,minAmount,getBytes16(1))` (contracts/lybra/pools/base/LybraWstETHVault.sol#20)
 - `minPesoUSD(msg.sender,msg.sender,minAmount,getBytes16(1))` (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#189)
- Reentrancy in `LybraPeUSDVaultBase._liquidation(address,address,uint256)` (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#123-149):
 - External calls:
 - `_repay(_provider,_onBehalfOf,pesoAmount)` (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#122)
 - `configurator.refreshMinReward(_onBehalfOf)` (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#108)
 - `PoS0.burn(_provider,amount - totalFee)` (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#204)
 - `PoS0.burn(_provider,amount - totalFee)` (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#210)
 - `configurator.refreshMinReward(_provider)` (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#182)
 - `collateralAsset.safeTransfer(msg.sender,rewardKeeper)` (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#145)
 - `collateralAsset.safeTransfer(_provider,reducedAsset - rewardKeeper)` (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#147)
 - Event emitted after the call(s): `_repay(_provider,_onBehalfOf,pesoAmount,reducedAsset,rewardKeeper,false,block.timestamp)` (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#148)
 - Reentrancy in `LybraPeUSDVaultBase._rigidRedemption(address,uint256,uint256)` (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#160-173):
 - External calls:
 - `_repay(msg.sender,_provider,pesoAmount)` (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#160)
 - `configurator.refreshMinReward(_onBehalfOf)` (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#108)
 - `PoS0.transferFrom(_provider,address(configurator),totalFee)` (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#204)
 - `PoS0.burn(_provider,amount - totalFee)` (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#205)
 - `PoS0.burn(_provider,amount - totalFee)` (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#210)
 - `configurator.refreshMinReward(_provider)` (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#182)
 - `collateralAsset.safeTransfer(msg.sender,rewardKeeper)` (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#145)
 - `collateralAsset.safeTransfer(_provider,reducedAsset - rewardKeeper)` (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#147)
 - Event emitted after the call(s): `_repay(msg.sender,_provider,pesoAmount,reducedAsset,rewardKeeper,false,block.timestamp)` (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#172)
 - Reference: <https://github.com/crytic/slither/wikidetector/Documentation#reentrancy-vulnerabilities>

```

State variables written after the call(s):
- borrowed_onBehalfOf - amount - contract(lyra/pools/base/LyraPsdVaultBase.sol#206)
LyraPsdVaultBase.borrowed(borrowed_(contracts)/lyra/pools/base/LyraPsdVaultBase.sol#202) can be used in cross function reentrancies:
- LyraPsdVaultBase.borrowed(borrowed_(contracts)/lyra/pools/base/LyraPsdVaultBase.sol#178-190)
- LyraPsdVaultBase.borrowed(notice(address)) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#241-243)
- LyraPsdVaultBase.repay(address,uint256) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#197-214)
- LyraPsdVaultBase.getborrowedOf(address) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#257-259)
- LyraPsdVaultBase.rightToRedemption(address,uint256,uint256) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#160-173)
Reentrancy: LyraPsdVaultBase.liquidation(address,address,uint256) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#123-149):
External calls:
- repay(provider,onBehalfOf,psuedoAmount) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#113)
- PeSD.transferFrom(provider,address,amount) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#198)
- PeSD.burn(provider,amount - totalFee) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#205)
- PeSD.transferFrom(provider,address,configurator,amount) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#210)
- PeSD.transferFrom(distributionReward,(contracts/lyra/pools/base/LyraPsdVaultBase.sol#212))
State variables written after the call(s):
- depositedAsset(onBehalfOf) - reducedBalanceSet (contracts/lyra/pools/base/LyraPsdVaultBase.sol#140)
LyraPsdVaultBase.depositedAsset(depositedAsset_(contracts)/lyra/pools/base/LyraPsdVaultBase.sol#142) can be used in cross function reentrancies:
- LyraPsdVaultBase.checkHealth(address,uint256) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#229-232)
- LyraPsdVaultBase.depositAsset(uint256,uint256,uint256) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#226-228)
- LyraPsdVaultBase.depositAssets(depositedAssets_(contracts)/lyra/pools/base/LyraPsdVaultBase.sol#59-69)
- LyraPsdVaultBase.depositAssets(depositedAssets_(contracts)/lyra/pools/base/LyraPsdVaultBase.sol#22)
- LyraPsdVaultBase.liquidation(address,address,uint256) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#123-149)
- LyraPsdVaultBase.rightToRedemption(address,uint256,uint256) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#160-173)
Reentrancy: LyraPsdVaultBase.rightToRedemption(address,uint256):
External calls:
- repay(msig,sender,provider,psuedoAmount) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#167)
- PeSD.transferFrom(provider,address,configurator,amount) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#198)
- PeSD.burn(provider,amount - totalFee) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#205)
- PeSD.transferFrom(provider,address,configurator,amount) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#210)
- configurator.distributionReward((contracts/lyra/pools/base/LyraPsdVaultBase.sol#212))
State variables written after the call(s):
- depositedAsset(provider) - collateralAmount (contracts/lyra/pools/base/LyraPsdVaultBase.sol#170)
LyraPsdVaultBase.depositedAsset(depositedAsset_(contracts)/lyra/pools/base/LyraPsdVaultBase.sol#172) can be used in cross function reentrancies:
- LyraPsdVaultBase.checkHealth(address,uint256) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#229-232)
- LyraPsdVaultBase.depositAsset(uint256,uint256,uint256) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#226-228)
- LyraPsdVaultBase.depositAssets(depositedAssets_(contracts)/lyra/pools/base/LyraPsdVaultBase.sol#59-69)
- LyraPsdVaultBase.depositAssets(depositedAssets_(contracts)/lyra/pools/base/LyraPsdVaultBase.sol#22)
- LyraPsdVaultBase.liquidation(address,address,uint256) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#123-149)
- LyraPsdVaultBase.rightToRedemption(address,uint256,uint256) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#160-173)
Reference: https://github.com/crytic/siltherin/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFODetectors:
LyraPsdVaultBase.liquidation(deaddress,address,uint256),rewardkeeper (contracts/lyra/pools/base/LyraPsdVaultBase.sol#141) is a local variable never initialized
Reference: https://github.com/crytic/siltherin/wiki/Detector-Documentation#uninitializing-local-variables
LyraPsdVaultBase.depositWithTokenToint(uint256) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#29-40) ignores return value by lido.approve(address(collateralAddress),msg.value) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#204)
LyraPsdVaultBase._minPsdLiquidity(address,uint256,uint256) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#178-190) ignores return value by PeSD.mint(_onBehalfOf,minIBalance)
LyraPsdVaultBase._repay(address,address,uint256) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#197-214) ignores return value by PeSD.burn(_provider,amount - totalFee)
Reference: https://github.com/crytic/siltherin/wiki/Detector-Documentation#unhandled-return
INFODetectors:
Reentrancy in LyraPsdVaultBase._repay(address,address,uint256) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#197-214):
External calls:
- configurator.refreshMinReward(_onBehalfOf) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#198)
State variables written after the call(s):
- _updateFee(_onBehalfOf) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#199)
- _feeStored[user] += _newFee(user) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#236)
- feeStored[_onBehalfOf] += _newFee(_onBehalfOf) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#199)
- _feeStored[_onBehalfOf] += _newFee(_onBehalfOf) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#209)
- _updateFee(_onBehalfOf) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#199)
- _feeStored[_user] = block.timestamp (contracts/lyra/pools/base/LyraPsdVaultBase.sol#237)
- feededStat[_user] = block.timestamp (contracts/lyra/pools/base/LyraPsdVaultBase.sol#237)
Reentrancy in LyraPsdVaultBase._depositWithTokenToint(uint256,uint256) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#59-60):
External calls:
- configurator.refreshMinReward(_onBehalfOf) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#198)
- PeSD.transferFrom(provider,address,configurator,amount) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#204)
- PeSD.burn(provider,amount - totalFee) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#205)
State variables written after the call(s):
- poolTotalCalculation -= amount - totalFee (contracts/lyra/pools/base/LyraPsdVaultBase.sol#207)
Reentrancy in LyraPsdVaultBase._depositWithTokenToint(uint256,uint256) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#59-60):
External calls:
- configurator.refreshMinReward(_onBehalfOf) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#198)
- PeSD.transferFrom(provider,address,configurator,amount) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#204)
- PeSD.burn(provider,amount - totalFee) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#205)
State variables written after the call(s):
- depositedAsset(sender) += wstETHAmount (contract/lyra/pools/base/LyraWstETHVault.sol#35)
Reference: https://github.com/crytic/siltherin/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFODetectors:
Reentrancy in LyraPsdVaultBase._mintPsdLiquidity(address,address,uint256,uint256) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#178-190):
External calls:
- configurator.refreshMinReward(_provider) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#198)
- PeSD.transferFrom(provider,address,configurator,amount) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#204)
- PeSD.burn(provider,amount - totalFee) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#205)
- PeSD.transferFrom(provider,address,configurator,amount) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#210)
- configurator.refreshMinReward(_provider) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#202)
Event emitted after the call(s):
- Burn(provider,_onBehalfOf,amount,block.timestamp) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#211)
Reentrancy in LyraPsdVaultBase._mintPsdLiquidity(address,address,uint256):
External calls:
- configurator.refreshMinReward(_onBehalfOf) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#198)
- PeSD.transferFrom(provider,address,configurator,amount) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#204)
- PeSD.burn(provider,amount - totalFee) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#205)
- PeSD.transferFrom(provider,address,configurator,amount) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#210)
- configurator.refreshMinReward(_provider) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#202)
Event emitted after the call(s):
- Mint(provider,_onBehalfOf,minAmount,block.timestamp) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#199)
Reentrancy in LyraPsdVaultBase._mintPsdLiquidity(address,address,uint256) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#197-214):
External calls:
- configurator.refreshMinReward(_onBehalfOf) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#198)
- PeSD.transferFrom(provider,address,configurator,amount) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#204)
- PeSD.burn(provider,amount - totalFee) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#205)
- PeSD.transferFrom(provider,address,configurator,amount) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#210)
- configurator.refreshMinReward(_provider) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#202)
Event emitted after the call(s):
- Burn(provider,_onBehalfOf,amount,block.timestamp) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#211)
Reentrancy in LyraPsdVaultBase._withdraw(address,address,uint256) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#216-224):
External calls:
- configurator.safeTransferFrom(_onBehalfOf,amount) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#199)
Event emitted after the call(s):
- Withdrawal(provider,address,(collateralAssetSet,_onBehalfOf,amount,block.timestamp)) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#69-89):
Reentrancy in LyraPsdVaultBase._depositWithTokenToint(uint256,uint256) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#59-60):
External calls:
- configurator.safeTransferFrom(msig,sender,address,(this),assetAmount) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#61)
- mintPsd(msig, sender, msg.sender, minAmount, assetPrice) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#166)
- configurator.refreshMinReward(provider) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#192)
- PeSD.mint(minAmount, sender, msg.sender, minAmount, assetPrice) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#166)
Event emitted after the call(s):
- Burn(provider,_onBehalfOf,minAmount,block.timestamp) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#199)
- mintPsd(msig, sender, msg.sender, minAmount, assetPrice) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#166)
Reentrancy in LyraPsdVaultBase._depositWithTokenToint(uint256,uint256) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#59-60):
External calls:
- sharesAmount = libd.submit(value,msg.value)(address(configurator)) (contracts/lyra/pools/base/LyraWstETHVault.sol#31)
- lido.approve(address(collateralAssetSet),msg.value) (contracts/lyra/pools/base/LyraWstETHVault.sol#33)
- wstETHAmount = WstETHAddress(collateralAssetSet).wrap(msg.value) (contracts/lyra/pools/base/LyraWstETHVault.sol#34)
- mintPsd(msig, sender, msg.sender, minAmount, assetPrice) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#167)
- etherOracle.fetchPrice() (contracts/lyra/pools/base/LyraPsdVaultBase.sol#249)
- configurate.refreshMinReward(_provider) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#192)
- PeSD.mint(_onBehalfOf,minAmount) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#166)
Event emitted after the call(s):
- Burn(provider,_onBehalfOf,amount,block.timestamp) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#211)
- mintPsd(msig, sender, msg.sender, minAmount, assetPrice) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#166)
Reentrancy in LyraPsdVaultBase._depositWithTokenToint(uint256,uint256) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#59-60):
External calls:
- repay(msig,sender,provider,psuedoAmount) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#167)
- msig.onBehalfOf(psuedoAmount) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#198)
- configurator.refreshMinReward(_onBehalfOf) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#199)
- PeSD.transferFrom(provider,address,configurator,amount) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#204)
- PeSD.burn(provider,amount - totalFee) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#205)
- PeSD.transferFrom(provider,address,configurator,amount) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#210)
- configurator.refreshMinReward(_provider) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#202)
- collateralAssetSet.safeTransfer(msig, sender, recipient, rewardkeeper) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#145)
- collateralAssetSet.safeTransfer(msig, sender, provider, reducedAsset - rewardkeeper) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#147)
Event emitted after the call(s):
- Burn(provider,_onBehalfOf,provider,psuedoAmount,reducedAsset,rewardkeeper,value,block.timestamp) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#148)
Reentrancy in LyraPsdVaultBase._rightToRedemption(address,address,uint256) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#160-173):
External calls:
- _repay(provider,sender,provider,psuedoAmount) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#167)
- msig.onBehalfOf(psuedoAmount) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#198)
- configurator.refreshMinReward(_onBehalfOf) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#199)
- PeSD.transferFrom(provider,address,configurator,amount) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#204)
- PeSD.burn(provider,amount - totalFee) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#205)
- PeSD.transferFrom(provider,address,configurator,amount) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#210)
- configurator.refreshMinReward(_provider) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#202)
- collateralAssetSet.safeTransfer(msig, sender, provider, collateralAmount) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#171)
Event emitted after the call(s):
- RightToRedemption(msig, sender, provider, psuedoAmount, collateralAmount, block.timestamp) (contracts/lyra/pools/base/LyraPsdVaultBase.sol#172)
Reference: https://github.com/crytic/siltherin/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

```

LybraEUSDVaultBase.sol

INFO:Dectors:

```

LybraUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraPoolsVaultBase.sol#123-149) uses timestamp for comparisons
  Dangerous comparisons:
    - require(bool,string)(onBehalfOfCollateralRatio < configurator.getBadCollateralRatio(address(this)),Borrowers collateral ratio should be below badCollateralRatio) (contracts/lybra/pools/base/LybraPoolsVaultBase.sol#126)
    - onBehalfOfCollateralRatio == 1e19 (contracts/lybra/pools/base/LybraPoolsVaultBase.sol#134)
    - msg.sender != provider && onBehalfOfCollateralRatio == 1e20 + keeperRatio * 1e16 (contracts/lybra/pools/base/LybraPoolsVaultBase.sol#143)
LybraUSDVaultBase.rigidRedemption(address,uint256,int256) (contracts/lybra/pools/base/LybraPoolsVaultBase.sol#160-171) uses timestamp for comparisons
  Dangerous comparisons:
    - amount >= borrowed[provider] && borrowed[provider] cannot surpass providers debt) (contracts/lybra/pools/base/LybraPoolsVaultBase.sol#163)
    - required(bool,string)(onBehalfOfCollateralRatio >= 100 * iei8,The provider's collateral ratio should be not less than 100%, ) (contracts/lybra/pools/base/LybraPoolsVaultBase.sol#166)
LybraUSDVaultBase.mintPoolsAddress(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraPoolsVaultBase.sol#176-190) uses timestamp for comparisons
  Dangerous comparisons:
    - poolTotalCirculation + _mintAmount <= configurator.mintVaultMaxSupply(address(this)),ESL.) (contracts/lybra/pools/base/LybraPoolsVaultBase.sol#179)
LybraUSDVaultBase.repayPoolsAddress(address,uint256) (contracts/lybra/pools/base/LybraPoolsVaultBase.sol#197-214) uses timestamp for comparisons
  Dangerous comparisons:
    - amount >= totalFee (contracts/lybra/pools/base/LybraPoolsVaultBase.sol#202)
    - required(bool,string)(onBehalfOfCollateralRatio >= amount (contracts/lybra/pools/base/LybraPoolsVaultBase.sol#203)
LybraUSDVaultBase.withdraw(address,address,uint256) (contracts/lybra/pools/base/LybraPoolsVaultBase.sol#216-224) uses timestamp for comparisons
  Dangerous comparisons:
    - getBorrowedOf_provider > 0 (contracts/lybra/pools/base/LybraPoolsVaultBase.sol#220)
LybraUSDVaultBase.checkHealth(address,uint256) (contracts/lybra/pools/base/LybraPoolsVaultBase.sol#229-232) uses timestamp for comparisons
  Dangerous comparisons:
    - (depositedAssets[asset] * price + 100 * getBorrowedOf(user)) < configurator.getSafeCollateralRatio(address(this)) (contracts/lybra/pools/base/LybraPoolsVaultBase.sol#230)
LybraUSDVaultBase.updateFee(address) (contracts/lybra/pools/base/LybraPoolsVaultBase.sol#234-239) uses timestamp for comparisons
  Dangerous comparisons:
    - onBehalfOf == updatedPools[User] (contracts/lybra/pools/base/LybraPoolsVaultBase.sol#255)

```

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp>

INFO:Dectors:

```

Function LybraETHVault.depositEtherToVault(uint256) (contracts/lybra/pools/base/LybraPoolsVaultBase.sol#29-40) has a dubious typecast: InstETH<=>address
Function LybraETHVault.withdrawEtherFromVault(uint256) (contracts/lybra/pools/base/LybraPoolsVaultBase.sol#41-52) has a dubious typecast: InstETH<=>address
Function LybraETHVault.setMinWithdrawRate(uint256) (contracts/lybra/pools/base/LybraPoolsVaultBase.sol#43-47) has a dubious typecast: InstETH<=>address

```

Reference: https://github.com/nessimistic-io/slither/blob/master/docs/dubious_typecast.md

INFO:Dectors:

```

Pragma version="0.8.17" (contracts/lybra/interfaces/IEUD.sol#2) allows old versions
Pragma version="0.8.17" (contracts/lybra/interfaces/IEUD.sol#2) allows old versions
Pragma version="0.8.17" (contracts/lybra/interfaces/IConfigurator.sol#1) allows old versions
Pragma version="0.8.17" (contracts/lybra/pools/base/LybraETHVault.sol#3) allows old versions
Pragma version="0.8.17" (contracts/lybra/pools/base/LybraETHVault.sol#3) allows old versions
solc -O1 --lto 1 (contracts/lybra/pools/base/LybraETHVault.sol#3) allows old versions

```

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity>

INFO:Dectors:

```

Parameter IUSD.getSharesByMinedUSD(uint256), eUSDamount (contracts/lybra/interfaces/IDUSD.sol#12) is not in mixedCase
Value LybraUSDVaultBase._eUSD (contracts/lybra/pools/base/LybraUSDVaultBase.sol#10) is not in mixedCase

```

Reference: https://github.com/nessimistic-io/slither/blob/master/docs/magic_name.md

INFO:Dectors:

```

In a function LybraUSDVault.depositEtherToVault(uint256) (contracts/lybra/pools/base/LybraPoolsVaultBase.sol#29-40) variable LybraPoolsVaultBase.collateralAsset (contracts/lybra/pools/base/LybraPoolsVaultBase.sol#17) is read multiple times

```

Reference: https://github.com/nessimistic-io/slither/blob/master/docs/multiple_storage_read.md

AUTOMATED TESTING

```

INFO[Detectors]:
LybraUSVaultBase._mintEUUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraUSVaultBase.sol#261-273) ignores return value by EUUSD.mint(_onBehalfOf_,_mintAmount) (contracts/lybra/pools/base/LybraUSVaultBase.sol#268)
LybraUSVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraUSVaultBase.sol#280-298) ignores return value by EUUSD.burn(_provider,_amount) (contracts/lybra/pools/base/LybraUSVaultBase.sol#283)
Reference: https://github.com/crylic/slither/wiki/Detector--Documentation#unused-return

INFO[Detectors]:
Reentrancy in LybraUSVaultBase._mintEUUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraUSVaultBase.sol#263-273):
  External calls:
    - configurator.refreshMintReward_provider() (contracts/lybra/pools/base/LybraUSVaultBase.sol#205)
      State variables written after the call(s):
        - _mintAmount (contracts/lybra/pools/base/LybraUSVaultBase.sol#206)
  State variables written before the call(s):
    - _saveReport() (contracts/lybra/pools/base/LybraUSVaultBase.sol#202)
      - _feeStored += _newFee() (contracts/lybra/pools/base/LybraUSVaultBase.sol#300)
    - _saveReport() (contracts/lybra/pools/base/LybraUSVaultBase.sol#203)
      - _lastReportTime = block.timestamp (contracts/lybra/pools/base/LybraUSVaultBase.sol#301)
  State variables written after the call(s):
    - _saveReport() (contracts/lybra/pools/base/LybraUSVaultBase.sol#204)
      - _poolTotalCirculation -= amount (contracts/lybra/pools/base/LybraUSVaultBase.sol#288)
  External calls:
    - EUUSD.burn(_provider,_amount) (contracts/lybra/pools/base/LybraUSVaultBase.sol#283)
    - EUUSD.burn(_provider,_amount) (contracts/lybra/pools/base/LybraUSVaultBase.sol#284)
  State variables written after the call(s):
    - _depositAsset(msg.sender,amount) (contracts/lybra/pools/base/LybraUSVaultBase.sol#174)
    - _depositAsset(msg.sender) == block.timestamp (contracts/lybra/pools/base/LybraUSVaultBase.sol#175)
    - _saveReport() (contracts/lybra/pools/base/LybraUSVaultBase.sol#204)
      - _lastReportTime = block.timestamp (contracts/lybra/pools/base/LybraUSVaultBase.sol#301)
  poolTotalCirculation -= amount (contracts/lybra/pools/base/LybraUSVaultBase.sol#288)

Reentrancy in LybraUSVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraUSVaultBase.sol#280-290):
  External calls:
    - EUUSD.burn(_provider,_amount) (contracts/lybra/pools/base/LybraUSVaultBase.sol#283)
    - EUUSD.burn(_provider,_amount) (contracts/lybra/pools/base/LybraUSVaultBase.sol#284)
  State variables written after the call(s):
    - _depositedAssets[_msg.sender] += assetAmount (contracts/lybra/pools/base/LybraUSVaultBase.sol#174)
    - _depositedAssets[msg.sender] == block.timestamp (contracts/lybra/pools/base/LybraUSVaultBase.sol#175)
    - _saveReport() (contracts/lybra/pools/base/LybraUSVaultBase.sol#204)
      - _lastReportTime = block.timestamp (contracts/lybra/pools/base/LybraUSVaultBase.sol#301)
  poolTotalCirculation -= amount (contracts/lybra/pools/base/LybraUSVaultBase.sol#288)

Reentrancy in LybraUSVaultBase.depositAssetToH(uint256,uint256) (contracts/lybra/pools/base/LybraUSVaultBase.sol#70-81):
  External calls:
    - collateralAsset.safeTransferFrom(msg.sender,address(this),assetAmount) (contracts/lybra/pools/base/LybraUSVaultBase.sol#172)
  State variables written after the call(s):
    - _depositedAssets[msg.sender] += assetAmount (contracts/lybra/pools/base/LybraUSVaultBase.sol#174)
    - _depositedAssets[_msg.sender] == assetAmount (contracts/lybra/pools/base/LybraUSVaultBase.sol#175)
    - _depositedAssets[_msg.sender] == block.timestamp (contracts/lybra/pools/base/LybraUSVaultBase.sol#176)
    - _depositedAssets[_msg.sender] == assetAmount (contracts/lybra/pools/base/LybraUSVaultBase.sol#177)
  State variables written before the call(s):
    - _totalDepositedAssets -= collateralAmount (contracts/lybra/pools/base/LybraUSVaultBase.sol#245)
  State variables written after the call(s):
    - _totalDepositedAssets -= collateralAmount (contracts/lybra/pools/base/LybraUSVaultBase.sol#246)
  External calls:
    - repay(msg.sender.provider,eusdMount) (contracts/lybra/pools/base/LybraUSVaultBase.sol#164)
      - EUUSD.burn(_provider,_amount) (contracts/lybra/pools/base/LybraUSVaultBase.sol#283)
      - EUUSD.burn(_provider,_amount) (contracts/lybra/pools/base/LybraUSVaultBase.sol#284)
  State variables written after the call(s):
    - _depositedAssets[_msg.sender] -= eusdMount (contracts/lybra/pools/base/LybraUSVaultBase.sol#166)
  Event emitted after the call(s):
    - Burn(_provider,_onBehalfOf_,amount,block.timestamp) (contracts/lybra/pools/base/LybraUSVaultBase.sol#280-290)

Reentrancy in LybraUSVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraUSVaultBase.sol#148-178):
  External calls:
    - repay(provider,onBehalfOf,eusdMount) (contracts/lybra/pools/base/LybraUSVaultBase.sol#157)
      - EUUSD.burn(_provider,_amount) (contracts/lybra/pools/base/LybraUSVaultBase.sol#283)
      - EUUSD.burn(_provider,_amount) (contracts/lybra/pools/base/LybraUSVaultBase.sol#284)
  State variables written after the call(s):
    - _depositedAssets[_msg.sender] -= eusdMount (contracts/lybra/pools/base/LybraUSVaultBase.sol#166)
  Event emitted after the call(s):
    - Repay(_provider,_onBehalfOf,_amount,block.timestamp) (contracts/lybra/pools/base/LybraUSVaultBase.sol#280-290)

Reentrancy in LybraUSVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraUSVaultBase.sol#280-290):
  External calls:
    - EUUSD.burn(_provider,_amount) (contracts/lybra/pools/base/LybraUSVaultBase.sol#283)
    - EUUSD.burn(_provider,_amount) (contracts/lybra/pools/base/LybraUSVaultBase.sol#284)
  Event emitted after the call(s):
    - Burn(_provider,_onBehalfOf_,amount,block.timestamp) (contracts/lybra/pools/base/LybraUSVaultBase.sol#289)

Reentrancy in LybraUSVaultBase.depositAssetToH(uint256,uint256) (contracts/lybra/pools/base/LybraUSVaultBase.sol#70-81):
  External calls:
    - collateralAsset.safeTransferFrom(msg.sender,address(this),assetAmount) (contracts/lybra/pools/base/LybraUSVaultBase.sol#172)
    - mintUSD(msg.sender,_msg.sender,_msg.sender.getAssetPrice()) (contracts/lybra/pools/base/LybraUSVaultBase.sol#78)
      - configurator.refreshMintReward_provider() (contracts/lybra/pools/base/LybraUSVaultBase.sol#205)
        - EUUSD.mint(_onBehalfOf_,_mintAmount) (contracts/lybra/pools/base/LybraUSVaultBase.sol#206)
    - depositAsset(_msg.sender,address(collateralAsset),assetAmount,block.timestamp) (contracts/lybra/pools/base/LybraUSVaultBase.sol#80)
    - Mint(_msg.sender,_onBehalfOf_,_mintAmount,block.timestamp,assetAmount.getAssetPrice()) (contracts/lybra/pools/base/LybraUSVaultBase.sol#78)
    - Mint(_msg.sender,_onBehalfOf_,_mintAmount,block.timestamp) (contracts/lybra/pools/base/LybraUSVaultBase.sol#178)
  State variables written after the call(s):
    - _depositedAssets[_msg.sender] += assetAmount (contracts/lybra/pools/base/LybraUSVaultBase.sol#166)
    - _depositedAssets[_msg.sender] == assetAmount (contracts/lybra/pools/base/LybraUSVaultBase.sol#167)
    - _depositedAssets[_msg.sender] == block.timestamp (contracts/lybra/pools/base/LybraUSVaultBase.sol#173)
    - collateralAsset.safeTransfer(provider,reducedAsset) (contracts/lybra/pools/base/LybraUSVaultBase.sol#175)
  Event emitted after the call(s):
    - repay(provider,_onBehalfOf_,eusdMount) (contracts/lybra/pools/base/LybraUSVaultBase.sol#157)
      - EUUSD.burn(_provider,_amount) (contracts/lybra/pools/base/LybraUSVaultBase.sol#283)
      - EUUSD.burn(_provider,_amount) (contracts/lybra/pools/base/LybraUSVaultBase.sol#284)
    - collateralAsset.safeTransferFrom(provider,reducedAsset) (contracts/lybra/pools/base/LybraUSVaultBase.sol#175)
  Event emitted after the call(s):
    - repay(msg.sender.provider,eusdMount) (contracts/lybra/pools/base/LybraUSVaultBase.sol#164)
      - EUUSD.burn(_provider,_amount) (contracts/lybra/pools/base/LybraUSVaultBase.sol#283)
      - EUUSD.burn(_provider,_amount) (contracts/lybra/pools/base/LybraUSVaultBase.sol#284)
    - collateralAsset.safeTransferFrom(provider,reducedAsset) (contracts/lybra/pools/base/LybraUSVaultBase.sol#175)
  Event emitted after the call(s):
    - KigRedemption(_msg.sender,_provider,_eusdMount,collateralAmount,block.timestamp) (contracts/lybra/pools/base/LybraUSVaultBase.sol#247)
  Event emitted after the call(s):
    - KigRedemption(_msg.sender,_provider,_eusdMount,collateralAmount,block.timestamp) (contracts/lybra/pools/base/LybraUSVaultBase.sol#247)

Reentrancy in LybraUSVaultBase.superLiquidation(address,address,uint256) (contracts/lybra/pools/base/LybraUSVaultBase.sol#189-213):
  External calls:
    - repay(provider,_onBehalfOf_,eusdMount) (contracts/lybra/pools/base/LybraUSVaultBase.sol#201)
      - EUUSD.burn(_provider,_amount) (contracts/lybra/pools/base/LybraUSVaultBase.sol#283)
      - EUUSD.burn(_provider,_amount) (contracts/lybra/pools/base/LybraUSVaultBase.sol#284)
    - collateralAsset.safeTransferFrom(provider,rewardKeeper) (contracts/lybra/pools/base/LybraUSVaultBase.sol#210)
  State variables written after the call(s):
    - liquidationRecord(_provider,_msg.sender,_onBehalfOf_,eusdMount,assetAmount,reward2keeper,true,block.timestamp) (contracts/lybra/pools/base/LybraUSVaultBase.sol#211)
  Event emitted after the call(s):
    - collateralAsset.safeTransferFrom(_onBehalfOf_,withDrawalAddress,uint256) (contracts/lybra/pools/base/LybraUSVaultBase.sol#203)
  External calls:
    - collateralAsset.safeTransfer(_onBehalfOf_,withDrawalAddress,uint256) (contracts/lybra/pools/base/LybraUSVaultBase.sol#192)
  Event emitted after the call(s):
    - collateralAsset.safeTransfer(provider,assetAmount - rewardKeeper) (contracts/lybra/pools/base/LybraUSVaultBase.sol#210)
  Event emitted after the call(s):
    - superLiquidationRecord(_provider,_msg.sender,_onBehalfOf_,eusdMount,assetAmount,assetAmount,reward2keeper,true,block.timestamp) (contracts/lybra/pools/base/LybraUSVaultBase.sol#211)

Reentrancy in LybraUSVaultBase.withdrawAsset(address,uint256) (contracts/lybra/pools/base/LybraUSVaultBase.sol#189-213):
  External calls:
    - collateralAsset.safeTransferFrom(_onBehalfOf_,withDrawalAddress,uint256) (contracts/lybra/pools/base/LybraUSVaultBase.sol#192)
  Event emitted after the call(s):
    - collateralAsset.safeTransfer(provider,assetAmount - rewardKeeper) (contracts/lybra/pools/base/LybraUSVaultBase.sol#210)

Reentrancy in LybraUSVaultBase._superLiquidation(address,address,uint256) (contracts/lybra/pools/base/LybraUSVaultBase.sol#189-213):
  External calls:
    - repay(provider,_onBehalfOf_,eusdMount) (contracts/lybra/pools/base/LybraUSVaultBase.sol#201)
      - EUUSD.burn(_provider,_amount) (contracts/lybra/pools/base/LybraUSVaultBase.sol#283)
      - EUUSD.burn(_provider,_amount) (contracts/lybra/pools/base/LybraUSVaultBase.sol#284)
    - configurator.refreshMintReward_provider() (contracts/lybra/pools/base/LybraUSVaultBase.sol#203)
      - EUUSD.mint(_onBehalfOf_,_mintAmount) (contracts/lybra/pools/base/LybraUSVaultBase.sol#206)
    - collateralAsset.safeTransfer(provider,rewardKeeper) (contracts/lybra/pools/base/LybraUSVaultBase.sol#210)
  State variables written after the call(s):
    - liquidationRecord(_provider,_msg.sender,_onBehalfOf_,eusdMount,assetAmount,reward2keeper,true,block.timestamp) (contracts/lybra/pools/base/LybraUSVaultBase.sol#211)
  Event emitted after the call(s):
    - collateralAsset.safeTransferFrom(_onBehalfOf_,withDrawalAddress,uint256) (contracts/lybra/pools/base/LybraUSVaultBase.sol#203)
  External calls:
    - collateralAsset.safeTransfer(_onBehalfOf_,withDrawalAddress,uint256) (contracts/lybra/pools/base/LybraUSVaultBase.sol#192)
  Event emitted after the call(s):
    - collateralAsset.safeTransfer(provider,assetAmount - rewardKeeper) (contracts/lybra/pools/base/LybraUSVaultBase.sol#210)
  Event emitted after the call(s):
    - superLiquidationRecord(_provider,_msg.sender,_onBehalfOf_,eusdMount,assetAmount,assetAmount,reward2keeper,true,block.timestamp) (contracts/lybra/pools/base/LybraUSVaultBase.sol#211)

Reentrancy in LybraUSVaultBase.superLiquidation(address,address,uint256) (contracts/lybra/pools/base/LybraUSVaultBase.sol#189-213):
  External calls:
    - repay(provider,_onBehalfOf_,eusdMount) (contracts/lybra/pools/base/LybraUSVaultBase.sol#201)
      - EUUSD.burn(_provider,_amount) (contracts/lybra/pools/base/LybraUSVaultBase.sol#283)
      - EUUSD.burn(_provider,_amount) (contracts/lybra/pools/base/LybraUSVaultBase.sol#284)
    - configurator.refreshMintReward_provider() (contracts/lybra/pools/base/LybraUSVaultBase.sol#203)
      - EUUSD.mint(_onBehalfOf_,_mintAmount) (contracts/lybra/pools/base/LybraUSVaultBase.sol#206)
    - collateralAsset.safeTransfer(provider,assetAmount - rewardKeeper) (contracts/lybra/pools/base/LybraUSVaultBase.sol#210)
  State variables written after the call(s):
    - liquidationRecord(_provider,_msg.sender,_onBehalfOf_,eusdMount,assetAmount,assetAmount,reward2keeper,true,block.timestamp) (contracts/lybra/pools/base/LybraUSVaultBase.sol#211)
  Event emitted after the call(s):
    - collateralAsset.safeTransferFrom(_onBehalfOf_,withDrawalAddress,uint256) (contracts/lybra/pools/base/LybraUSVaultBase.sol#203)
  External calls:
    - collateralAsset.safeTransfer(_onBehalfOf_,withDrawalAddress,uint256) (contracts/lybra/pools/base/LybraUSVaultBase.sol#192)
  Event emitted after the call(s):
    - collateralAsset.safeTransfer(provider,assetAmount - rewardKeeper) (contracts/lybra/pools/base/LybraUSVaultBase.sol#210)
  Event emitted after the call(s):
    - superLiquidationRecord(_provider,_msg.sender,_onBehalfOf_,eusdMount,assetAmount,assetAmount,reward2keeper,true,block.timestamp) (contracts/lybra/pools/base/LybraUSVaultBase.sol#211)

INFO[Detectors]:
LybraUSVaultBase._mintEUUSD(address,uint256,uint256) (contracts/lybra/pools/base/LybraUSVaultBase.sol#311-313) is never used and should be removed
Reference: https://github.com/crylic/slither/wiki/Detector--Documentation#read-multiple-times

INFO[Detectors]:
LybraUSVaultBase._ethenPrice() (contracts/lybra/pools/base/LybraUSVaultBase.sol#311-313) is never used and should be removed
Reference: https://github.com/crylic/slither/wiki/Detector--Documentation#read-multiple-times

INFO[Detectors]:
Pragme version@0.8.17 (contracts/lybra/interfaces/IUSD.sol#2) allows old versions
Pragme version@0.8.17 (contracts/lybra/interfaces/Configurator.sol#3) allows old versions
Pragme version@0.8.17 (contracts/lybra/interfaces/OnBehalfOf.sol#2) allows old versions
Pragme version@0.8.17 is not recommended for online contracts
Reference: https://github.com/crylic/slither/wiki/Detector--Documentation#incorrect-version-of-solidity

INFO[Detectors]:
pragma getShareByMined(uint256,EUUSDmount) (contracts/lybra/interfaces/IUSD.sol#2) is not in mixedCase
Variable LybraUSVaultBase.EUSDmount (contracts/lybra/pools/base/LybraUSVaultBase.sol#40) is in mixedCase
Reference: https://github.com/crylic/slither/wiki/Detector--Documentation#conformance-to-solidity-naming-conventions

INFO[Detectors]:
LybraUSVaultBase.deposit() (contracts/lybra/pools/base/LybraUSVaultBase.sol#43-53) does not implement functions:
  - LybraUSVaultBase.depositAssetToH(uint256) (contracts/lybra/pools/base/LybraUSVaultBase.sol#40)
  - LybraUSVaultBase.depositFromH(address,uint256) (contracts/lybra/pools/base/LybraUSVaultBase.sol#41)
  LybraUSVaultBase.getAsset2etherExchangeRate() (contracts/lybra/pools/base/LybraUSVaultBase.sol#52)
  LybraUSVaultBase.getAssetPrice() (contracts/lybra/pools/base/LybraUSVaultBase.sol#53)
Reference: https://github.com/crylic/slither/wiki/Detector--Documentation#multiple-implemented-functions

INFO[Detectors]:
In a function LybraUSVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraUSVaultBase.sol#148-178) variable LybraUSVaultBase.depositedAsset (contracts/lybra/pools/base/LybraUSVaultBase.sol#26) is read # multiple times
In a function LybraUSVaultBase.superLiquidation(address,address,uint256) (contracts/lybra/pools/base/LybraUSVaultBase.sol#189-213) variable LybraUSVaultBase.depositedAsset (contracts/lybra/pools/base/LybraUSVaultBase.sol#26) is read # multiple times
In a function LybraUSVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraUSVaultBase.sol#280-290) variable LybraUSVaultBase.borrowed (contracts/lybra/pools/base/LybraUSVaultBase.sol#27) is read multiple times
es
Reference: https://github.com/pepsi/mistic-jo/slither/blob/master/docs/multiple_storage.read.md


```

LybraPeUSDVaultBase.sol

LBR.sol

LBR2.sol

PeUSD.sol

INFO Detectors:

- Pragma version^{0.8.17} ([contracts/lybra/token/LBR.sol#3](#)) allows old versions
- Pragma version^{0.8.17} ([contracts/lybra/token/LBR.sol#3](#)) allows old versions
- solc-0.8.17 is not recommended for deployment
- Reference:

INFO Detectors:

- Constant LBR_maxSupply ([contracts/lybra/token/LBR.sol#14](#)) is not in **UPPER_CASE_WITH_UNDERSCORES**
- Reference:

INFO Detectors:

- Pragma version^{0.8.17} ([contracts/lybra/token/PeUSD.sol#3](#)) allows old versions
- Pragma version^{0.8.17} ([contracts/lybra/token/PeUSD.sol#3](#)) allows old versions
- solc-0.8.17 is not recommended for deployment
- Reference:

Reentrancy in LybraPeUSDVaultBase.rigIDRedemption(address,uint256,uint256)

```

- PeUSD.burn(_provider,amount - totalFee) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#205)
- PeUSD.transferFrom(_provider,address(configurator),amount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#210)
- Configurator.refersTransferRewards() (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#212)
- collateralAsset.safeTransfer(msg.sender,rewardKeeper) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#145)
Event emitted after the call(s)
- LiquidationRecord(provider,msg.sender,onBehalfOf,reducedAsset,rewardKeeper,false,block.timestamp) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#148)
Reentrancy in LybraPeUSDVaultBase.rigIDRedemption(address,uint256,uint256):
External calls:
- repay(msg.sender,_provider,pauseAmount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#167)
  configurator.refersMintReward_onBehalfOf() (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#168)
- PeUSD.transferFrom(_provider,address(configurator),totalFee) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#204)
- PeUSD.burn(_provider,amount - totalFee) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#205)
- PeUSD.transferFrom(_provider,address(configurator),amount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#210)
- Configurator.distributeRewards() (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#121)
- collateralAsset.safeTransfer(msg.sender,collateralAmount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#171)
Event emitted after the call(s)
- emittedLiquidation(provider,msg.sender,_provider,pauseAmount,collateralAmount,block.timestamp) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#172)

```

INFO Detectors:

LybraPeUSDVaultBase.liquidateOnBehalfOf(address,uint256) ([contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#123](#)) uses timestamp for comparisons

Dangerous comparisons:

- onBehalfOfCollateralRatio <= configurator.getSafeCollateralRatio(address(this)), borrows collateral ratio should be below badCollateralRatio ([contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#126](#))
- onBehalfOfCollateralRatio >= 1e18 && onBehalfOfCollateralRatio <= 1e18 ([contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#134](#))
- msg.sender != provider && onBehalfOfCollateralRatio >= 1e18 ([contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#137](#))

LybraPeUSDVaultBase.onBehalfOfCollateralRatio(address,uint256) ([contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#125](#)) uses timestamp for comparisons

Dangerous comparisons:

- require(bool,string)(borrowed[provider] >= pauseAmount,pauseAmount cannot surpass providers debt) ([contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#163](#))

LybraPeUSDVaultBase.onBehalfOfCollateralRatio(address,uint256) ([contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#166](#)) uses timestamp for comparisons

Dangerous comparisons:

- require(bool,string)(totalICirculation + _mintAmount <= configurator.mintVaultMaxSupply(address(this)),_SL) ([contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#179](#))

LybraPeUSDVaultBase._repay(address,address,uint256) ([contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#197-214](#)) uses timestamp for comparisons

Dangerous comparisons:

- totalFee >= amount ([contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#192](#))
- borrowed_onBehalfOf > amount ([contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#201](#))

LybraPeUSDVaultBase.withdraw(address,address,uint256) ([contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#216-224](#)) uses timestamp for comparisons

Dangerous comparisons:

- require(bool)(0 <= address(<msg.sender>),_SL) ([contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#1720](#))

LybraPeUSDVaultBase.checkHealth(address,uint256) ([contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#220-222](#)) uses timestamp for comparisons

Dangerous comparisons:

- (depositedAssets[user] * price + 100 / getBorrowedDF(user)) < configurator.getSafeCollateralRatio(address(this)) ([contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#230](#))

LybraPeUSDVaultBase.onBehalfOfCollateralRatio(address,uint256) ([contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#234-239](#)) uses timestamp for comparisons

Dangerous comparisons:

- block.timestamp >= feedUpdatedAt[user] ([contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#225](#))

INFO Detectors:

https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

LybraPeUSDVaultBase._otherPrice() ([contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#248-250](#)) is never used and should be removed

Reference:

INFO Detectors:

Pragma version^{0.8.17} ([contracts/lybra/interfaces/IPeUSD.sol#2](#)) allows old versions

Pragma version^{0.8.17} ([contracts/lybra/interfaces/IConfigurator.sol#1](#)) allows old versions

Pragma version^{0.8.17} ([contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#3](#)) allows old versions

solc-0.8.17 is not recommended for deployment

Reference:

INFO Detectors:

Variable LybraPeUSDVaultBase.PeiSD ([contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#16](#)) is not in mixedCase

Reference:

INFO Detectors:

LybraPeUSDVaultBase.depositOnBehalfOf(address,uint256) ([contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#109](#)) does not implement functions:

- LybraPeUSDVaultBase.depositOnBehalfOf(uint256) ([contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#109](#))
- LybraPeUSDVaultBase.getAsset2EtherExchangeRate() ([contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#274](#))
- LybraPeUSDVaultBase.getAssetPrice() ([contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#177](#))

Reference:

INFO Detectors:

Function LybraPeUSDVaultBase.depositAssetMint(uint256,uint256) ([contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#59-69](#)) contains magic number: 10000000000000000000

Function LybraPeUSDVaultBase.depositAssetMint(address,address,uint256) ([contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#123-140](#)) contains magic numbers: 100, 1e20, 1e20, 1e19, 11, 1e20, 100

Function LybraPeUSDVaultBase.rigIDRedemption(address,uint256,uint256) ([contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#160-173](#)) contains magic numbers: 100, 100

Function LybraPeUSDVaultBase._checkHealth(address,uint256) ([contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#209-214](#)) contains magic number: 100

Function LybraPeUSDVaultBase._depositAssets(address,uint256) ([contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#241-245](#)) contains magic number: 90

Reference:

INFO Detectors:

In a function LybraPeUSDVaultBase.liquidation(address,address,uint256) ([contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#123-149](#)) variable LybraPeUSDVaultBase.depositedAsset ([contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#22](#)) is read multiple times

In a function LybraPeUSDVaultBase._repay(address,address,uint256) ([contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#197-214](#)) variable LybraPeUSDVaultBase.borrowed ([contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#23](#)) is read multiple times

In a function LybraPeUSDVaultBase._repay(address,address,uint256) ([contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#197-214](#)) variable LybraPeUSDVaultBase.configurator ([contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#18](#)) is read multiple times

Reference:

PeUSDMainnet.sol

```

INFO:Detectors:
PeUSDMainnet.executeFlashloan(IFlashBorrower,uint256,bytes) (contracts/lybra/token/PeUSDMainnet.sol#127-139) uses arbitrary from in transferFrom: success = EUSD.transferFrom(address(receiver),address(this)),EUSD.getMintedEUSDByShares(shar
eAmount)) (contracts/lybra/token/PeUSDMainnet.sol#132)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#arbitrary-from-in-transferfrom
INFO:Detectors:
PeUSDMainnet.convertToEUSD(uint256) (contracts/lybra/token/PeUSDMainnet.sol#111-119) ignores return value by EUSD.transferShares(msg.sender,share) (contracts/lybra/token/PeUSDMainnet.sol#117)
PeUSDMainnet.executeFlashloan(IFlashBorrower,uint256,bytes) (contracts/lybra/token/PeUSDMainnet.sol#127-139) ignores return value by EUSD.transferShares(address(receiver),shareAmount) (contracts/lybra/token/PeUSDMainnet.sol#130)
PeUSDMainnet.executeFlashloan(IFlashBorrower,uint256,bytes) (contracts/lybra/token/PeUSDMainnet.sol#127-139) ignores return value by EUSD.burnShares(msg.sender,burnShare) (contracts/lybra/token/PeUSDMainnet.sol#137)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#muted-return
INFO:Detectors:
PeUSDMainnet.executeFlashloan parameter free is not related to msg.sender success = EUSD.transferFrom(address(receiver),address(this)),EUSD.getMintedEUSDByShares(shareAmount)) (contracts/lybra/token/PeUSDMainnet.sol#132)
INFO:Detectors:
Reentrancy in PeUSDMainnet.convertToEUSD(uint256) (contracts/lybra/token/PeUSDMainnet.sol#111-119):
  External calls
    - EUSD.transferShares(msg.sender,share) (contracts/lybra/token/PeUSDMainnet.sol#117)
      + EUSD.transferShares(address(receiver),share) (call())
      + ConvertToEUSD(msg.sender,shareAmount,block.timestamp) (contracts/lybra/token/PeUSDMainnet.sol#118)
Reentrancy in PeUSDMainnet.executeFlashloan(IFlashBorrower,uint256,bytes) (contracts/lybra/token/PeUSDMainnet.sol#127-139):
  External calls
    - EUSD.transferShares(address(receiver),shareAmount) (contracts/lybra/token/PeUSDMainnet.sol#130)
      + receiveOnFlashloan(shareAmount,data) (contracts/lybra/token/PeUSDMainnet.sol#131)
      + success = EUSD.transferFrom(address(receiver),address(this)),EUSD.getMintedEUSDByShares(shareAmount)) (contracts/lybra/token/PeUSDMainnet.sol#132)
      + EUSD.burnShares(msg.sender,burnShare) (contracts/lybra/token/PeUSDMainnet.sol#137)
    Event emitted after the call(s):
      + FlashloanEvent(address(receiver),shareAmount,burnShare,block.timestamp) (contracts/lybra/token/PeUSDMainnet.sol#138)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
Function PeUSDMainnet.executeFlashloan(IFlashBorrower,uint256,bytes) (contracts/lybra/token/PeUSDMainnet.sol#127-139) has a dubious typecast: address<address>
External calls
  - PeUSDMainnet.EUSD (contracts/lybra/token/PeUSDMainnet.sol#29)
Reference: https://github.com/crytic/slither/wiki/Master/docs/dubious-typecasts-and
INFO:Detectors:
Pragma version"0.8.17" (contracts/lybra/interfaces/IUSD.sol#2) allows old versions
Pragma version"0.8.17" (contracts/lybra/interfaces/IConfigurator.sol#3) allows old versions
Pragma version"0.8.17" (contracts/lybra/interfaces/ISolidity.sol#4) allows old versions
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
pragma solidity^0.8.17; (contracts/lybra/interfaces/IUSD.sol#2) is not in mixedCase
Variable PeUSDMainnet.EUSD (contracts/lybra/token/PeUSDMainnet.sol#29) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Function PeUSDMainnet.getFee(uint256) (contracts/lybra/token/PeUSDMainnet.sol#156-158) contains magic number: 10
External calls
  - PeUSDMainnet.getFee(uint256) (contracts/lybra/token/PeUSDMainnet.sol#156-158) in /slitherin/slither/master/docs/magic_numbers.md
INFO:Detectors:
In a function PeUSDMainnet.convertToEUSD(uint256) (contracts/lybra/token/PeUSDMainnet.sol#75-85) variable PeUSDMainnet.EUSD (contracts/lybra/token/PeUSDMainnet.sol#29) is read multiple times
In a function PeUSDMainnet.convertToUSD(uint256) (contracts/lybra/token/PeUSDMainnet.sol#111-119) variable PeUSDMainnet.userConvertInfo (contracts/lybra/token/PeUSDMainnet.sol#31) is read multiple times
In a function PeUSDMainnet.executeFlashloan(IFlashBorrower,uint256,bytes) (contracts/lybra/token/PeUSDMainnet.sol#127-139) variable PeUSDMainnet.EUSD (contracts/lybra/token/PeUSDMainnet.sol#29) is read multiple times
Reference: https://github.com/crytic/slither/wiki/Master/docs/multiple_storage_reads

```

esLBR.sol

```

INFO:Detectors:
Pragma version"0.8.17" (contracts/lybra/interfaces/IConfigurator.sol#3) allows old versions
Pragma version"0.8.17" (contracts/lybra/token/esLBR.sol#3) allows old versions
Solidity 0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Contract esLBR (contracts/lybra/token/esLBR.sol#16-45) is not in CapWords
Constant esLBR.maxSupply (contracts/lybra/token/esLBR.sol#16) is NOT IN UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

```

LBRMinerFromL2.sol

```

INFO:Detectors:
LBRMinerFromL2._nonblockingReceive(uint16,bytes,uint64,bytes) (contracts/lybra/miner/LBRMinerFromL2.sol#26-30) ignores return value by esLBR.mint(to,amount) (contracts/lybra/miner/LBRMinerFromL2.sol#28)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#muted-return
INFO:Detectors:
Reentrancy in LBRMinerFromL2._nonblockingReceive(uint16,bytes,uint64,bytes) (contracts/lybra/miner/LBRMinerFromL2.sol#26-30):
  External calls
    - esLBR.mint(to,amount) (contracts/lybra/miner/LBRMinerFromL2.sol#28)
      + Event emitted after the call(s).
      + esLBR.mint(to,amount,block.timestamp) (contracts/lybra/miner/LBRMinerFromL2.sol#29)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Pragma version"0.8.17" (contracts/lybra/miner/LBRMinerFromL2.sol#3) allows old versions
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

```

- The unprotected initialize issues flagged by Slither are false positives.
- All the reentrancies flagged by Slither were checked individually and can be considered false positives.
- No major issues found by Slither.

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 the smart contracts and sent the compiled results to the analyzers to locate any vulnerabilities.

MythX results:

LybraConfigurator.sol

| Line | SWC Title | Severity | Short Description |
|------|---|----------|--|
| 15 | (SWC-103) Floating Pragma | Low | A floating pragma is set. |
| 47 | (SWC-108) State Variable Default Visibility | Low | State variable visibility is not set. |
| 48 | (SWC-108) State Variable Default Visibility | Low | State variable visibility is not set. |
| 51 | (SWC-108) State Variable Default Visibility | Low | State variable visibility is not set. |
| 53 | (SWC-108) State Variable Default Visibility | Low | State variable visibility is not set. |
| 66 | (SWC-108) State Variable Default Visibility | Low | State variable visibility is not set. |
| 296 | (SWC-107) Reentrancy | Low | Read of persistent state following external call |

LybraProxy.sol

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

LybraProxyAdmin.sol

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

AdminTimelock.sol

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

GovernanceTimelock.sol

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

LybraGovernance.sol

| Line | SWC Title | Severity | Short Description |
|------|--|----------|--|
| 3 | (SWC-103) Floating Pragma | Low | A floating pragma is set. |
| 155 | (SWC-120) Weak Sources of Randomness from Chain Attributes | Low | Potential use of "block.number" as source of randomness. |
| 165 | (SWC-120) Weak Sources of Randomness from Chain Attributes | Low | Potential use of "block.number" as source of randomness. |

EUSDMiningIncentives.sol

| Line | SWC Title | Severity | Short Description |
|------|---|----------|--|
| 3 | (SWC-103) Floating Pragma | Low | A floating pragma is set. |
| 33 | (SWC-110) Assert Violation | Unknown | Public state variable with array type causing reacheable exception by default. |
| 50 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "*" discovered |
| 59 | (SWC-108) State Variable Default Visibility | Low | State variable visibility is not set. |
| 104 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "++" discovered |
| 105 | (SWC-110) Assert Violation | Unknown | Out of bounds array access |
| 156 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "++" discovered |
| 157 | (SWC-110) Assert Violation | Unknown | Out of bounds array access |
| 158 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "+=" discovered |
| 158 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "/" discovered |
| 158 | (SWC-110) Assert Violation | Unknown | Out of bounds array access |
| 158 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "*" discovered |
| 161 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "+=" discovered |
| 161 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "/" discovered |
| 161 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "*" discovered |
| 171 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "++" discovered |
| 172 | (SWC-110) Assert Violation | Unknown | Out of bounds array access |
| 173 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "/" discovered |
| 173 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "*" discovered |
| 173 | (SWC-110) Assert Violation | Unknown | Out of bounds array access |
| 173 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "+=" discovered |
| 176 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "+=" discovered |
| 176 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "/" discovered |
| 176 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "*" discovered |
| 191 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "/" discovered |
| 191 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "*" discovered |
| 192 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "/" discovered |
| 192 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "*" discovered |
| 194 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "/" discovered |
| 194 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "+" discovered |
| 194 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "*" discovered |
| 206 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "+" discovered |
| 206 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "*" discovered |
| 206 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "--" discovered |
| 206 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "/" discovered |
| 219 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "+" discovered |
| 219 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "*" discovered |
| 223 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "/" discovered |
| 223 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "+" discovered |
| 223 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "*" discovered |
| 223 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "--" discovered |

ProtocolRewardsPool.sol

| Line | SWC Title | Severity | Short Description |
|------|---|----------|---------------------------------------|
| 3 | (SWC-103) Floating Pragma | Low | A floating pragma is set. |
| 42 | (SWC-108) State Variable Default Visibility | Low | State variable visibility is not set. |

esLBRBoost.sol

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

stakerewardPoolOnArbi.sol

| Line | SWC Title | Severity | Short Description |
|------|---------------------------|----------|---------------------------|
| 2 | (SWC-103) Floating Pragma | Low | A floating pragma is set. |

stakerewardV2pool.sol

| Line | SWC Title | Severity | Short Description |
|------|---------------------------------|----------|---------------------------|
| 2 | (SWC-103) Floating Pragma | Low | A floating pragma is set. |
| 17 | (SWC-123) Requirement Violation | Low | Requirement violation. |
| 105 | (SWC-123) Requirement Violation | Low | Requirement violation. |

LybraRETHVault.sol

| Line | SWC Title | Severity | Short Description |
|------|---|----------|---------------------------------------|
| 3 | (SWC-103) Floating Pragma | Low | A floating pragma is set. |
| 22 | (SWC-108) State Variable Default Visibility | Low | State variable visibility is not set. |
| 36 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "+" discovered |
| 36 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "-" discovered |
| 42 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "--" discovered |
| 46 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "*" discovered |
| 46 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "/" discovered |

LybraStETHVault.sol

| Line | SWC Title | Severity | Short Description |
|------|--|----------|---|
| 3 | (SWC-103) Floating Pragma | Low | A floating pragma is set. |
| 43 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "+=" discovered |
| 44 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "+=" discovered |
| 63 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "-" discovered |
| 67 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "*" discovered |
| 67 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "/" discovered |
| 69 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "+" discovered |
| 76 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "-" discovered |
| 79 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "-" discovered |
| 89 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "-" discovered |
| 103 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "%" discovered |
| 103 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation ".." discovered |
| 105 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "*" discovered |
| 105 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "/" discovered |
| 105 | (SWC-101) Integer Overflow and Underflow | Unknown | Compiler-rewritable "<uint> - 1" discovered |
| 105 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "-" discovered |

LybraWbETHVault.sol

| Line | SWC Title | Severity | Short Description |
|------|--|----------|--------------------------------------|
| 3 | (SWC-103) Floating Pragma | Low | A floating pragma is set. |
| 26 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "+=" discovered |
| 26 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation ".." discovered |
| 32 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "-" discovered |
| 36 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "*" discovered |
| 36 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "/" discovered |

LybraWstETHVault.sol

| Line | SWC Title | Severity | Short Description |
|------|---|----------|---------------------------------------|
| 3 | (SWC-103) Floating Pragma | Low | A floating pragma is set. |
| 22 | (SWC-108) State Variable Default Visibility | Low | State variable visibility is not set. |
| 35 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "+=" discovered |
| 43 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "/" discovered |
| 43 | (SWC-101) Integer Overflow and Underflow | Unknown | Arithmetic operation "*" discovered |

LybraEUSDVaultBase.sol

| Line | SWC Title | Severity | Short Description |
|------|---|----------|---------------------------------------|
| 3 | (SWC-103) Floating Pragma | Low | A floating pragma is set. |
| 20 | (SWC-108) State Variable Default Visibility | Low | State variable visibility is not set. |
| 24 | (SWC-108) State Variable Default Visibility | Low | State variable visibility is not set. |
| 27 | (SWC-108) State Variable Default Visibility | Low | State variable visibility is not set. |
| 29 | (SWC-108) State Variable Default Visibility | Low | State variable visibility is not set. |

LybraPeUSDVaultBase.sol

| Line | SWC Title | Severity | Short Description |
|------|---|----------|---------------------------------------|
| 3 | (SWC-103) FloatingPragma | Low | A floating pragma is set. |
| 19 | (SWC-108) State Variable Default Visibility | Low | State variable visibility is not set. |
| 20 | (SWC-108) State Variable Default Visibility | Low | State variable visibility is not set. |
| 23 | (SWC-108) State Variable Default Visibility | Low | State variable visibility is not set. |
| 24 | (SWC-108) State Variable Default Visibility | Low | State variable visibility is not set. |
| 25 | (SWC-108) State Variable Default Visibility | Low | State variable visibility is not set. |

EUSD.sol

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

LBR.sol

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

LBRL2.sol

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

PeUSD.sol

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

PeUSDMainnet.sol

| Line | SWC Title | Severity | Short Description |
|------|--------------------------|----------|---------------------------|
| 14 | (SWC-103) FloatingPragma | Low | A floating pragma is set. |

esLBR.sol

| Line | SWC Title | Severity | Short Description |
|------|--|----------|--|
| 44 | (SWC-120) Weak Sources of Randomness from Chain Attributes | Low | Potential use of "block.number" as source of randomness. |
| 53 | (SWC-120) Weak Sources of Randomness from Chain Attributes | Low | Potential use of "block.number" as source of randomness. |

LBRMinerFromL2.sol

| Line | SWC Title | Severity | Short Description |
|------|---|----------|---------------------------------------|
| 3 | (SWC-103) FloatingPragma | Low | A floating pragma is set. |
| 18 | (SWC-108) State Variable Default Visibility | Low | State variable visibility is not set. |

- MythX flagged some integer overflows and underflows which all were false positives, as the contracts are using Solidity ^0.8.17 version.

After the Solidity version 0.8.0 Arithmetic operations revert to underflow and overflow by default.

- `block.number` is not used as a source of randomness anywhere in the code.
- MythX also flagged some assert violations, which were all considered to be false positives.
- No major issues were found by MythX.

THANK YOU FOR CHOOSING
HALBORN