// HALBORN

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

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 Gulgun
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 Gulgun	Halborn	Gokberk.Gulgun@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 Omnichain 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.

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) Specific (AO:S)	1 0.2
Attack Cost (AC)	Low (AC:L) Medium (AC:M) High (AC:H)	1 0.67 0.33
Attack Complexity (AX)	Low (AX:L) Medium (AX:M) High (AX:H)	1 0.67 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) Partial (R:P) Full (R:F)	1 0.5 0.25
Scope (s)	Changed (S:C) Unchanged (S:U)	1.25 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	5

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)	RISK ACCEPTED
(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.LATESTROUNDDATA() 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) LYBRARETHVAULT.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

(HAL-13) LYBRAGOVERNANCE HARDCODED 4 PERCENT QUORUM	Low (2.5)	SOLVED - 08/24/2023
(HAL-14) EUSDPRICEFEED HARDCODED DECIMALS	Informational (0.0)	ACKNOWLEDGED
(HAL-15) REDUNDANT ALLOWANCE CHECK IN LIQUIDATION() FUNCTION	Informational (0.0)	ACKNOWLEDGED
(HAL-16) STATE VARIABLES MISSING IMMUTABLE MODIFIER	Informational (0.0)	ACKNOWLEDGED
(HAL-17) LACK OF A DOUBLE-STEP TRANSFEROWNERSHIP PATTERN IN MULTIPLE CONTRACTS	Informational (0.0)	ACKNOWLEDGED
(HAL-18) 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,
 ↳ 2, balance, balance * uint(price) * 998 / 1e23);
324                 IERC20(stableToken).safeTransfer(address(
 ↳ lybraProtocolRewardsPool), amount);
325                 lybraProtocolRewardsPool.notifyRewardAmount(amount
 ↳ , 1);
326                 emit SendProtocolRewards(stableToken, amount,
 ↳ block.timestamp);
327             }
328         } else {
329             peUSD.convertToPeUSD(address(this), balance);
330         }
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:

AO: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,
 ↳ 2, balance, balance * uint(price) * 998 / 1e23);
324                 IERC20(stableToken).safeTransfer(address(
 ↳ lybraProtocolRewardsPool), amount);
325                 lybraProtocolRewardsPool.notifyRewardAmount(amount
 ↳ , 1);
326                 emit SendProtocolRewards(stableToken, amount,
 ↳ block.timestamp);
327             }
328         } else {
329             peUSD.convertToPeUSD(address(this), balance);
330         }
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 }
```

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
 ↳ and mint peUSD. It can only be the contract itself or the msg.
 ↳ sender.
73  * @param eusdAmount The amount of eUSD to deposit and mint peUSD
 ↳ tokens.
74  */
75 function convertToPeUSD(address user, uint256 eusdAmount) public {
76     require(_msgSender() == user || _msgSender() == address(this),
 ↳  "MDM");
77     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 }
```

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:

AO: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 : d050e08072633dcccd69155a36901c9434dfdb58.

4.3 (HAL-03) LYBRACONFIGURATOR 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:

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

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

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

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

```
192 /**
193  * @notice Burn _provideramount peUSD to payback minted peUSD for
 ↳ _onBehalfOf.
194  *
195  * @dev Refresh LBR reward before reducing providers debt. Refresh
 ↳  Lybra generated service fee before reducing totalPeUSDCirculation
 ↳ .
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
 ↳ liquidation.
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
 ↳ maximum of 5%.
255  */
256 function setKeeperRatio(address pool,uint256 newRatio) external
 ↳ checkRole(TIMELOCK) {
257     require(newRatio <= 5, "Max Keeper reward is 5%");
258     vaultKeeperRatio[pool] = newRatio;
259     emit KeeperRatioChanged(pool, newRatio);
260 }
```

BVSS:

AO: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:

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

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)

```
1  {
2    "name": "lybra-protocol",
3    "scripts": {
4      "build": "hardhat compile",
5      "test": "hardhat test"
6    },
7    "devDependencies": {
8      "hardhat": "^2.14.0"
9    },
10   "dependencies": {
11     "@chainlink/contracts": "^0.6.1",
12     "@layerzerolabs/solidity-examples": "^0.0.12",
13     "@nomicfoundation/hardhat-toolbox": "^2.0.2",
14     "@openzeppelin/contracts": "^4.8.3",
15     "hardhat-gas-reporter": "^1.0.9"
16   }
17 }
```

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:

AO: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 : d050e08072633dcccd69155a36901c9434dfdb58.

4.6 (HAL-06) CENTRALIZATION ISSUE: LYBRAGOVERNANCE 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
↳ [] memory executors, address admin) TimelockController(minDelay,
↳ proposers, executors, admin) {
15     _setRoleAdmin(DAO, GOV);
16     _setRoleAdmin(TIMELOCK, GOV);
17     _setRoleAdmin(ADMIN, GOV);
18     _grantRole(DAO, address(this));
19     _grantRole(DAO, msg.sender);
20     _grantRole(GOV, msg.sender);
21 }
```

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:

AO: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.LATESTROUNDDATA() 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, , , )= eUSDPriceFeed.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:

AO: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) = eUSDPriceFeed.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
 ↳ retrieve eUSD. The user's balance of peUSD tokens must be greater
 ↳ than or equal to this amount.
107  * Requirements:
108  * `peusdAmount` must be greater than 0.
109  * The user's `mintedPeUSD` must be greater than or equal to `
 ↳ peusdAmount`.
110  */
111 function convertToEUSD(uint256 peusdAmount) external {
112     require(peusdAmount <= userConvertInfo[msg.sender].mintedPeUSD
```

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

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:

AO: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
↳ and mint peUSD. It can only be the contract itself or the msg.
↳ sender.
73  * @param eusdAmount The amount of eUSD to deposit and mint peUSD
↳ tokens.
74  */
75 function convertToPeUSD(address user, uint256 eusdAmount) public {
76     require(_msgSender() == user || _msgSender() == address(this),
↳  "MDM");
77     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:

AO: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:

AO: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:

AO: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 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: LybraPeUSDVaultBase.sol (Line 60)

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

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:

AO: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: LybraPeUSDVaultBase.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:

AO: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
 ↳ based on the available balance of peUSD and eUSD.
309  * If the balance is greater than 1e21, the distribution process
 ↳ is triggered.
310  *
311  * First, if the eUSD balance is greater than 1,000 and the
 ↳ premiumTradingEnabled flag is set to true,
312  * and the eUSD/USDC premium exceeds 0.5%, eUSD will be exchanged
 ↳ for USDC and added to the LybraProtocolRewardsPool.
313  * Otherwise, eUSD will be directly converted to peUSD, and the
 ↳ entire peUSD balance will be transferred to the
 ↳ LybraProtocolRewardsPool.
314  * @dev The protocol rewards amount is notified to the
 ↳ LybraProtocolRewardsPool for proper reward allocation.
315  */
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,
 ↳ 2, balance, balance * uint(price) * 998 / 1e23);
```

```
324                 IERC20(stableToken).safeTransfer(address(
 ↳ lybraProtocolRewardsPool), amount);
325                 lybraProtocolRewardsPool.notifyRewardAmount(amount
 ↳ , 1);
326                 emit SendProtocolRewards(stableToken, amount,
 ↳ block.timestamp);
327             }
328         } else {
329             peUSD.convertToPeUSD(address(this), balance);
330         }
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 }
```

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:

AO: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) REDUNDANT ALLOWANCE CHECK IN LIQUIDATION() FUNCTION - INFORMATIONAL (0.0)

Commit IDs affected:
- 90285107de8a6754954c303cd69d97b5fdb4e248

Description:

In the LybraEUSDVaultBase and LybraPeUSDVaultBase contracts, the function liquidation() performs the following allowance check:

Listing 21: LybraPeUSDVaultBase.sol (Line 129)

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

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

This check is performed to ensure that the `_repay()` internal call will not revert. Although, this check is redundant, as the ERC20 contracts do already perform this allowance check when `transferFrom` is called and will automatically revert if the caller does not have enough allowance.

BVSS:

AO: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 remove the allowance check in the `liquidation()` function in order to reduce the gas costs.

Remediation Plan:

ACKNOWLEDGED: The Lybra Finance team acknowledged this finding.

4.16 (HAL-16) 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:

AO: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.

Remediation Plan:

ACKNOWLEDGED: The Lybra Finance team acknowledged this finding.

4.17 (HAL-17) LACK OF A DOUBLE-STEP TRANSFEROWNERSHIP 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:

AO: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 22: Ownable2Step.sol (Lines 52-56)

```
// SPDX-License-Identifier: MIT
// OpenZeppelin Contracts (last updated v4.8.0) (access/Ownable2Step.sol)

pragma solidity ^0.8.0;

import "./Ownable.sol";

/**
 * @dev Contract module which provides access control mechanism, where
 * there is an account (an owner) that can be granted exclusive access to
 * specific functions.
 *
 * By default, the owner account will be the one that deploys the contract. This
 * can later be changed with {transferOwnership} and {acceptOwnership}.
 *
 * This module is used through inheritance. It will make available all functions
 * from parent (Ownable).
 */
abstract contract Ownable2Step is Ownable {
    address private _pendingOwner;

    event OwnershipTransferStarted(address indexed previousOwner, address indexed newOwner);

    /**
     * @dev Returns the address of the pending owner.
     */
    function pendingOwner() public view virtual returns (address) {
        return _pendingOwner;
    }

    /**
```

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

Remediation Plan:

ACKNOWLEDGED: The Lybra Finance team acknowledged this finding.

4.18 (HAL-18) 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:

AO: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 slipagge 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. Remove the allowance check in the liquidation() function in order to reduce the gas costs.
15. Add the immutable modifier to the state variables mentioned.
16. 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.

17. 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#340)
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(_peusd,type()(uint256).max) (contracts/lybra/configuration/LybraConfigurator.sol#121)
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#unused-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:
        - eUSDMiningIncentives.refreshReward(msg.sender) (contracts/lybra/configuration/LybraConfigurator.sol#295)
        State variables written after the call(s):
        - redemptionProvider[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:
        - eUSDMiningIncentives.refreshReward(msg.sender) (contracts/lybra/configuration/LybraConfigurator.sol#295)
        Event emitted after the call(s):
        - RedemptionProvider(msg.sender,_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.exchange_underlying(0,2,balance,balance * uint256(price) * 99B / 1e23) (contracts/lybra/configuration/LybraConfigurator.sol#329)
        - IERC20(stableToken).safeTransfer(address(lybraProtocolRewardsPool),amount) (contracts/lybra/configuration/LybraConfigurator.sol#330)
        - lybraProtocolRewardsPool.notifyRewardAmount(amount,1) (contracts/lybra/configuration/LybraConfigurator.sol#331)
        Event emitted after the call(s):
        - SendProtocolRewards(stableToken,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(curvePool),balance) (contracts/lybra/configuration/LybraConfigurator.sol#322)
        - amount = curvePool.exchange_underlying(0,2,balance,balance * uint256(price) * 99B / 1e23) (contracts/lybra/configuration/LybraConfigurator.sol#329)
        - IERC20(stableToken).safeTransfer(address(lybraProtocolRewardsPool),amount) (contracts/lybra/configuration/LybraConfigurator.sol#330)
        - lybraProtocolRewardsPool.notifyRewardAmount(amount,1) (contracts/lybra/configuration/LybraConfigurator.sol#331)
        - peUSD.convertToPeUSD(address(this),balance) (contracts/lybra/configuration/LybraConfigurator.sol#335)
        - peUSD.transfer(address(lybraProtocolRewardsPool),peUSDBalance) (contracts/lybra/configuration/LybraConfigurator.sol#340)
        - lybraProtocolRewardsPool.notifyRewardAmount(peUSDBalance,0) (contracts/lybra/configuration/LybraConfigurator.sol#341)
        Event emitted after the call(s):
        - 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.initToken(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.setMintVault(address,bool) (contracts/lybra/configuration/LybraConfigurator.sol#130-132) does not emit an event
Setter function LybraConfigurator.setMintVaultMaxSupply(address,uint256) (contracts/lybra/configuration/LybraConfigurator.sol#140-142) does not emit an event
Setter function LybraConfigurator.setEUSDOracle(address) (contracts/lybra/configuration/LybraConfigurator.sol#180-182) does not emit an event
Setter function LybraConfigurator.setPremiumTradingEnabled(bool) (contracts/lybra/configuration/LybraConfigurator.sol#189-191) does not emit an event
Setter function LybraConfigurator.setVaultBurnPaused(address,bool) (contracts/lybra/configuration/LybraConfigurator.sol#199-201) does not emit an event
Setter function LybraConfigurator.setVaultMintPaused(address,bool) (contracts/lybra/configuration/LybraConfigurator.sol#209-211) does not emit an event
Setter function LybraConfigurator.setMaxStableRatio(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.md
INFO:Detectors:
Pragma version^0.8.17 (contracts/lybra/configuration/LybraConfigurator.sol#15) allows old versions
Pragma version^0.8.17 (contracts/lybra/interfaces/IEUSD.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
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 ICurvePool.exchange_underlying(int128,int128,uint256,uint256) (contracts/lybra/configuration/LybraConfigurator.sol#38) is not in mixedCase
Parameter ICurvePool.exchange_underlying(int128,int128,uint256,uint256).min_dy (contracts/lybra/configuration/LybraConfigurator.sol#38) is not in mixedCase
Parameter LybraConfigurator.initialize(address,address,address,address)._dao (contracts/lybra/configuration/LybraConfigurator.sol#95) is not in mixedCase
Parameter LybraConfigurator.initialize(address,address,address,address)._curvePool (contracts/lybra/configuration/LybraConfigurator.sol#95) is not in mixedCase
Parameter LybraConfigurator.initialize(address,address,address,address)._eUSDPriceFeed (contracts/lybra/configuration/LybraConfigurator.sol#95) is not in mixedCase
Parameter LybraConfigurator.initialize(address,address,address,address)._stableToken (contracts/lybra/configuration/LybraConfigurator.sol#95) is not in mixedCase
Parameter LybraConfigurator.initToken(address,address)._eusd (contracts/lybra/configuration/LybraConfigurator.sol#118) is not in mixedCase
Parameter LybraConfigurator.initToken(address,address)._peusd (contracts/lybra/configuration/LybraConfigurator.sol#118) is not in mixedCase
Parameter LybraConfigurator.setEUSDOracle(address)._eUSDOracle (contracts/lybra/configuration/LybraConfigurator.sol#180) is not in mixedCase
Parameter LybraConfigurator.setTokenMiner(address[],bool[])._contracts (contracts/lybra/configuration/LybraConfigurator.sol#267) is not in mixedCase
Parameter LybraConfigurator.setTokenMiner(address[],bool[])._bools (contracts/lybra/configuration/LybraConfigurator.sol#267) is not in mixedCase
Parameter LybraConfigurator.setMaxStableRatio(uint256)._ratio (contracts/lybra/configuration/LybraConfigurator.sol#278) 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#58) is not in mixedCase
Variable LybraConfigurator.EUSD (contracts/lybra/configuration/LybraConfigurator.sol#62) is not in mixedCase
Parameter IEUSD.getSharesByMintedEUSD(uint256)._EUSDAmount (contracts/lybra/interfaces/IEUSD.sol#32) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

```
INFO:Detectors:
Function LybraConfigurator.initialize(address,address,address,address) (contracts/lybra/configuration/LybraConfigurator.sol#95-103) contains magic numbers: 50, 500, 5
Function LybraConfigurator.setBadCollateralRatio(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: 2e20
Function LybraConfigurator.setRedemptionFee(uint256) (contracts/lybra/configuration/LybraConfigurator.sol#218-222) contains magic number: 500
Function LybraConfigurator.setSafeCollateralRatio(address,uint256) (contracts/lybra/configuration/LybraConfigurator.sol#230-238) contains magic numbers: 160, 1e19
Function LybraConfigurator.setBorrowApy(address,uint256) (contracts/lybra/configuration/LybraConfigurator.sol#245-249) contains magic number: 200
Function LybraConfigurator.setKeeperRatio(address,uint256) (contracts/lybra/configuration/LybraConfigurator.sol#256-260) contains magic number: 5
Function LybraConfigurator.setMaxStableRatio(uint256) (contracts/lybra/configuration/LybraConfigurator.sol#278-281) contains magic number: 10
Function LybraConfigurator.setFlashloanFee(uint256) (contracts/lybra/configuration/LybraConfigurator.sol#285-289) contains magic number: 10
Function LybraConfigurator.distributeRewards() (contracts/lybra/configuration/LybraConfigurator.sol#316-344) contains magic numbers: 1e21, 100, 1e21
Function LybraConfigurator.getSafeCollateralRatio(address) (contracts/lybra/configuration/LybraConfigurator.sol#367-372) contains magic number: 160
Function LybraConfigurator.getBadCollateralRatio(address) (contracts/lybra/configuration/LybraConfigurator.sol#374-377) contains magic number: 1e19
Function LybraConfigurator.getVaultWeight(address) (contracts/lybra/configuration/LybraConfigurator.sol#379-384) contains magic number: 100
Function LybraConfigurator.getEUSDMaxLocked() (contracts/lybra/configuration/LybraConfigurator.sol#399-401) contains magic number: 10
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 multip
le 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
```

LybraProxy.sol

```
INFO:Detectors:
Function LybraProxy.constructor(address,address,bytes) (contracts/lybra/Proxy/LybraProxy.sol#9) is a strange setter. Nothing is set in constructor or set in a function without using function parameters
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/strange_setter.md
INFO:Detectors:
Pragma version^0.8.17 (contracts/lybra/Proxy/LybraProxy.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
```

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#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:
Parameter GovernanceTimelock.checkRole(bytes32,address)._sender (contracts/lybra/governance/GovernanceTimelock.sol#24) is not in mixedCase
Parameter GovernanceTimelock.checkOnlyRole(bytes32,address)._sender (contracts/lybra/governance/GovernanceTimelock.sol#28) 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[],address) (contracts/lybra/governance/GovernanceTimelock.sol#14-22) variable GovernanceTimelock.DAO (contracts/lybra/governance/GovernanceTimelock.sol#9) is read mul
tiple times
In a function GovernanceTimelock.constructor(uint256,address[],address[],address) (contracts/lybra/governance/GovernanceTimelock.sol#14-22) variable GovernanceTimelock.GOV (contracts/lybra/governance/GovernanceTimelock.sol#12) is read mu
ltiple 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._countVote(uint256,address,uint8,uint256,bytes) (contracts/lybra/governance/LybraGovernance.sol#84-95) compares to a boolean constant:
        -require(bool,string)(receipt.hasVoted == false,GovernorBravo::castVoteInternal: 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/lybra/interfaces/IGovernanceTimelock.sol#2) allows old versions
Pragma version^0.8.17 (contracts/lybra/interfaces/IesLBR.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).governor43Id (contracts/lybra/governance/LybraGovernance.sol#190-193) is too similar to LybraGovernance.supportsInterface(bytes4).governor46Id (contracts/lybra/governance/LybraGovernance
.sol#196)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
Function LybraGovernance.quorum(uint256) (contracts/lybra/governance/LybraGovernance.sol#53-55) contains magic number: 4
Function LybraGovernance.votingPeriod() (contracts/lybra/governance/LybraGovernance.sol#146-148) contains magic number: 50400
Function LybraGovernance.votingDelay() (contracts/lybra/governance/LybraGovernance.sol#150-152) contains magic number: 14400
Function LybraGovernance.proposalThreshold() (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.esLBR (contracts/lybra/governance/LybraGovernance.sol#11) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
```

EUSDMiningIncentives.sol

```
INFO:Detectors:
Function EUSDMiningIncentives.updateReward(address) (contracts/lybra/miner/EUSDMiningIncentives.sol#79-89) is an unprotected initializer.
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/unprotected_initialize.md
INFO:Detectors:
EUSDMiningIncentives._buyOtherEarnings(address,bool) (contracts/lybra/miner/EUSDMiningIncentives.sol#257-276) performs a multiplication on the result of a division:
        - biddingFee = (reward * biddingFeeRatio) / 10_000 (contracts/lybra/miner/EUSDMiningIncentives.sol#265)
        - biddingFee = biddingFee * uint256(lbrPrice) / 1e8 (contracts/lybra/miner/EUSDMiningIncentives.sol#268)
EUSDMiningIncentives.notifyRewardAmount(uint256) (contracts/lybra/miner/EUSDMiningIncentives.sol#284-300) performs a multiplication on the result of a division:
        - rewardRatio = amount / duration (contracts/lybra/miner/EUSDMiningIncentives.sol#289)
        - remainingRewards = (finishAt - block.timestamp) * rewardRatio (contracts/lybra/miner/EUSDMiningIncentives.sol#291)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
```

```
INFO:Detectors:
EUSDMiningIncentives.getBoost(address).redemptionBoost (contracts/lybra/miner/EUSDMiningIncentives.sol#215) is a local variable never initialized
EUSDMiningIncentives.buyOthersEarnings(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#237-245) ignores return value by IesLBR(esLBR).mint(msg.sender,reward) (contracts/lybra/miner/EUSDMiningIncentives.sol#242)
EUSDMiningIncentives._buyOtherEarnings(address,bool) (contracts/lybra/miner/EUSDMiningIncentives.sol#257-276) ignores return value by IesLBR(LBR).burn(msg.sender,biddingFee) (contracts/lybra/miner/EUSDMiningIncentives.sol#272)
EUSDMiningIncentives._buyOtherEarnings(address,bool) (contracts/lybra/miner/EUSDMiningIncentives.sol#257-276) ignores return value by IesLBR(esLBR).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 = _biddingRatio (contracts/lybra/miner/EUSDMiningIncentives.sol#113)
EUSDMiningIncentives.setExtraRatio(uint256) (contracts/lybra/miner/EUSDMiningIncentives.sol#116-119) should emit an event for:
        - extraRatio = ratio (contracts/lybra/miner/EUSDMiningIncentives.sol#118)
EUSDMiningIncentives.setMinDlpRatio(uint256) (contracts/lybra/miner/EUSDMiningIncentives.sol#121-124) should emit an event for:
        - minDlpRatio = 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,address)._weth (contracts/lybra/miner/EUSDMiningIncentives.sol#70) lacks a zero-check on :
        - wETH = _weth (contracts/lybra/miner/EUSDMiningIncentives.sol#75)
EUSDMiningIncentives.constructor(address,address,address,address,address)._oldEUSD (contracts/lybra/miner/EUSDMiningIncentives.sol#70) lacks a zero-check on :
        - oldLybra = _oldEUSD (contracts/lybra/miner/EUSDMiningIncentives.sol#76)
EUSDMiningIncentives.setToken(address,address)._lbr (contracts/lybra/miner/EUSDMiningIncentives.sol#91) lacks a zero-check on :
        - LBR = _lbr (contracts/lybra/miner/EUSDMiningIncentives.sol#92)
EUSDMiningIncentives.setToken(address,address)._esLbr (contracts/lybra/miner/EUSDMiningIncentives.sol#91) lacks a zero-check on :
        - esLBR = _esLbr (contracts/lybra/miner/EUSDMiningIncentives.sol#93)
EUSDMiningIncentives.setEthLbrStakeInfo(address,address)._pool (contracts/lybra/miner/EUSDMiningIncentives.sol#139) lacks a zero-check on :
        - ethLbrStakePool = _pool (contracts/lybra/miner/EUSDMiningIncentives.sol#140)
EUSDMiningIncentives.setEthLbrStakeInfo(address,address)._lp (contracts/lybra/miner/EUSDMiningIncentives.sol#139) lacks a zero-check on :
        - ethLbrLpToken = _lp (contracts/lybra/miner/EUSDMiningIncentives.sol#141)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
EUSDMiningIncentives.setPools(address[]) (contracts/lybra/miner/EUSDMiningIncentives.sol#102-109) has external calls inside a loop: require(bool,string)(configurator.mintVault(_vaults[i]),NOT_VAULT) (contracts/lybra/miner/EUSDMiningIncentives.sol#105)
EUSDMiningIncentives.totalStaked() (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#158)
EUSDMiningIncentives.stakedOf(address) (contracts/lybra/miner/EUSDMiningIncentives.sol#169-179) has external calls inside a loop: amount += vault.getBorrowedOf(user) * configurator.getVaultWeight(vaults[i]) / 1e20 (contracts/lybra/miner/EUSDMiningIncentives.sol#173)
EUSDMiningIncentives.totalStaked() (contracts/lybra/miner/EUSDMiningIncentives.sol#154-164) has external calls inside a loop: amount += IEUSD(oldLybra).totalSupply() * configurator.getVaultWeight(oldLybra) / 1e20 (contracts/lybra/miner/EUSDMiningIncentives.sol#161)
EUSDMiningIncentives.stakedOf(address) (contracts/lybra/miner/EUSDMiningIncentives.sol#169-179) has external calls inside a loop: amount += ILybra(oldLybra).getBorrowedOf(user) * configurator.getVaultWeight(oldLybra) / 1e20 (contracts/lybra/miner/EUSDMiningIncentives.sol#176)
EUSDMiningIncentives.getBoost(address) (contracts/lybra/miner/EUSDMiningIncentives.sol#214-220) has external calls inside a loop: configurator.isRedemptionProvider(_account) (contracts/lybra/miner/EUSDMiningIncentives.sol#216)
EUSDMiningIncentives.getBoost(address) (contracts/lybra/miner/EUSDMiningIncentives.sol#214-220) has external calls inside a loop: 100 * 1e18 + redemptionBoost + esLBRBoost.getUserBoost(_account,userUpdatedAt[_account],finishAt) (contracts/lybra/miner/EUSDMiningIncentives.sol#219)
EUSDMiningIncentives.stakedLBRLpValue(address) (contracts/lybra/miner/EUSDMiningIncentives.sol#186-195) has external calls inside a loop: totallp = IEUSD(ethlbrLpToken).totalSupply() (contracts/lybra/miner/EUSDMiningIncentives.sol#187)
EUSDMiningIncentives.stakedLBRLpValue(address) (contracts/lybra/miner/EUSDMiningIncentives.sol#186-195) has external calls inside a loop: (etherPrice) = etherPriceFeed.latestRoundData() (contracts/lybra/miner/EUSDMiningIncentives.sol#189)
EUSDMiningIncentives.stakedLBRLpValue(address) (contracts/lybra/miner/EUSDMiningIncentives.sol#186-195) has external calls inside a loop: (lbrPrice) = lbrPriceFeed.latestRoundData() (contracts/lybra/miner/EUSDMiningIncentives.sol#190)
EUSDMiningIncentives.stakedLBRLpValue(address) (contracts/lybra/miner/EUSDMiningIncentives.sol#186-195) has external calls inside a loop: etherInLp = (IEUSD(wETH).balanceOf(ethlbrLpToken) * uint256(etherPrice)) / 1e8 (contracts/lybra/miner/EUSDMiningIncentives.sol#191)
EUSDMiningIncentives.stakedLBRLpValue(address) (contracts/lybra/miner/EUSDMiningIncentives.sol#186-195) has external calls inside a loop: lbrInLp = (IEUSD(LBR).balanceOf(ethlbrLpToken) * uint256(lbrPrice)) / 1e8 (contracts/lybra/miner/EUSDMiningIncentives.sol#192)
EUSDMiningIncentives.stakedLBRLpValue(address) (contracts/lybra/miner/EUSDMiningIncentives.sol#186-195) has external calls inside a loop: userStaked = IEUSD(ethlbrStakePool).balanceOf(user) (contracts/lybra/miner/EUSDMiningIncentives.sol#193)
EUSDMiningIncentives._buyOtherEarnings(address,bool) (contracts/lybra/miner/EUSDMiningIncentives.sol#257-276) has external calls inside a loop: (lbrPrice) = lbrPriceFeed.latestRoundData() (contracts/lybra/miner/EUSDMiningIncentives.sol#267)
EUSDMiningIncentives._buyOtherEarnings(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#269)
EUSDMiningIncentives._buyOtherEarnings(address,bool) (contracts/lybra/miner/EUSDMiningIncentives.sol#257-276) has external calls inside a loop: IesLBR(esLBR).mint(msg.sender,reward) (contracts/lybra/miner/EUSDMiningIncentives.sol#274)
EUSDMiningIncentives._buyOtherEarnings(address,bool) (contracts/lybra/miner/EUSDMiningIncentives.sol#257-276) has external calls inside a loop: IesLBR(LBR).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._buyOtherEarnings(address,bool) (contracts/lybra/miner/EUSDMiningIncentives.sol#257-276):
        External calls:
        - success = EUSD.transferFrom(msg.sender,address(owner()),biddingFee) (contracts/lybra/miner/EUSDMiningIncentives.sol#269)
        - IesLBR(LBR).burn(msg.sender,biddingFee) (contracts/lybra/miner/EUSDMiningIncentives.sol#272)
        - IesLBR(esLBR).mint(msg.sender,reward) (contracts/lybra/miner/EUSDMiningIncentives.sol#274)
        Event emitted after the call(s):
        - ClaimedOtherEarnings(msg.sender,user,reward,biddingFee,useEUSD,block.timestamp) (contracts/lybra/miner/EUSDMiningIncentives.sol#275)
Reentrancy in EUSDMiningIncentives.getReward() (contracts/lybra/miner/EUSDMiningIncentives.sol#237-245):
        External calls:
        - IesLBR(esLBR).mint(msg.sender,reward) (contracts/lybra/miner/EUSDMiningIncentives.sol#242)
        Event emitted after the call(s):
        - ClaimReward(msg.sender,reward,block.timestamp) (contracts/lybra/miner/EUSDMiningIncentives.sol#243)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
EUSDMiningIncentives.setRewardsDuration(uint256) (contracts/lybra/miner/EUSDMiningIncentives.sol#134-137) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(finishAt < block.timestamp,reward duration not finished) (contracts/lybra/miner/EUSDMiningIncentives.sol#135)
EUSDMiningIncentives._buyOtherEarnings(address,bool) (contracts/lybra/miner/EUSDMiningIncentives.sol#257-276) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(rewards[user] != 0,ZA) (contracts/lybra/miner/EUSDMiningIncentives.sol#259)
EUSDMiningIncentives.notifyRewardAmount(uint256) (contracts/lybra/miner/EUSDMiningIncentives.sol#284-300) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp >= finishAt (contracts/lybra/miner/EUSDMiningIncentives.sol#288)
        - require(bool,string)(rewardRatio != 0,reward ratio = 0) (contracts/lybra/miner/EUSDMiningIncentives.sol#295)
EUSDMiningIncentives._min(uint256,uint256) (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.setExtraRatio(uint256) (contracts/lybra/miner/EUSDMiningIncentives.sol#116-119) does not emit an event
Setter function EUSDMiningIncentives.setMinDlpRatio(uint256) (contracts/lybra/miner/EUSDMiningIncentives.sol#121-124) does not emit an event
Setter function EUSDMiningIncentives.setBoost(address) (contracts/lybra/miner/EUSDMiningIncentives.sol#126-128) does not emit an event
Setter function EUSDMiningIncentives.setV1Supported(bool) (contracts/lybra/miner/EUSDMiningIncentives.sol#130-132) 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.setEthLbrStakeInfo(address,address) (contracts/lybra/miner/EUSDMiningIncentives.sol#139-142) does not emit an event
Setter function EUSDMiningIncentives.setEUSDBuyoutAllowed(bool) (contracts/lybra/miner/EUSDMiningIncentives.sol#143-145) does not emit an event
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/event_setter.md
INFO:Detectors:
EUSDMiningIncentives.updateReward(address) (contracts/lybra/miner/EUSDMiningIncentives.sol#79-89) has costly operations inside a loop:
        - rewardPerTokenStored = rewardPerToken() (contracts/lybra/miner/EUSDMiningIncentives.sol#80)
EUSDMiningIncentives.updateReward(address) (contracts/lybra/miner/EUSDMiningIncentives.sol#79-89) has costly operations inside a loop:
        - updatedAt = lastTimeRewardApplicable() (contracts/lybra/miner/EUSDMiningIncentives.sol#81)
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/ILybra.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/IesLBR.sol#2) allows old versions
Pragma version^0.8.17 (contracts/lybra/miner/EUSDMiningIncentives.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:
Parameter IEUSD.getSharesByMintedEUSD(uint256)._EUSDAmount (contracts/lybra/interfaces/IEUSD.sol#32) is not in mixedCase
Parameter EUSDMiningIncentives.setToken(address,address)._lbr (contracts/lybra/miner/EUSDMiningIncentives.sol#91) is not in mixedCase
Parameter EUSDMiningIncentives.setToken(address,address)._esLbr (contracts/lybra/miner/EUSDMiningIncentives.sol#91) is not in mixedCase
Parameter EUSDMiningIncentives.setLBROracle(address)._lbrOracle (contracts/lybra/miner/EUSDMiningIncentives.sol#97) is not in mixedCase
Parameter EUSDMiningIncentives.setPools(address[])._vaults (contracts/lybra/miner/EUSDMiningIncentives.sol#102) is not in mixedCase
Parameter EUSDMiningIncentives.setBiddingCost(uint256)._biddingRatio (contracts/lybra/miner/EUSDMiningIncentives.sol#111) is not in mixedCase
Parameter EUSDMiningIncentives.setBoost(address)._boost (contracts/lybra/miner/EUSDMiningIncentives.sol#126) is not in mixedCase
Parameter EUSDMiningIncentives.setV1Supported(bool)._bool (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.setEthLbrStakeInfo(address,address)._pool (contracts/lybra/miner/EUSDMiningIncentives.sol#139) is not in mixedCase
Parameter EUSDMiningIncentives.setEthLbrStakeInfo(address,address)._lp (contracts/lybra/miner/EUSDMiningIncentives.sol#139) is not in mixedCase
Parameter EUSDMiningIncentives.setEUSDBuyoutAllowed(bool)._bool (contracts/lybra/miner/EUSDMiningIncentives.sol#143) is not in mixedCase
Parameter EUSDMiningIncentives.refreshReward(address)._account (contracts/lybra/miner/EUSDMiningIncentives.sol#212) is not in mixedCase
Parameter EUSDMiningIncentives.getBoost(address)._account (contracts/lybra/miner/EUSDMiningIncentives.sol#214) is not in mixedCase
Parameter EUSDMiningIncentives.earned(address)._account (contracts/lybra/miner/EUSDMiningIncentives.sol#222) is not in mixedCase
Variable EUSDMiningIncentives.EUSD (contracts/lybra/miner/EUSDMiningIncentives.sol#29) is not in mixedCase
Variable EUSDMiningIncentives.LBR (contracts/lybra/miner/EUSDMiningIncentives.sol#31) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Function EUSDMiningIncentives.setPools(address[]) (contracts/lybra/miner/EUSDMiningIncentives.sol#102-109) contains magic number: 10
Function EUSDMiningIncentives.setBiddingCost(uint256) (contracts/lybra/miner/EUSDMiningIncentives.sol#111-114) contains magic number: 8000
Function EUSDMiningIncentives.setExtraRatio(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.stakedOf(address) (contracts/lybra/miner/EUSDMiningIncentives.sol#169-179) contains magic numbers: 1e20, 1e20
Function EUSDMiningIncentives.stakedLBRLpValue(address) (contracts/lybra/miner/EUSDMiningIncentives.sol#186-195) contains magic numbers: 1e8, 1e8
Function EUSDMiningIncentives.getBoost(address) (contracts/lybra/miner/EUSDMiningIncentives.sol#214-220) contains magic number: 100
Function EUSDMiningIncentives.earned(address) (contracts/lybra/miner/EUSDMiningIncentives.sol#222-224) contains magic number: 1e30
Function EUSDMiningIncentives.isOtherEarningsClaimable(address) (contracts/lybra/miner/EUSDMiningIncentives.sol#231-235) contains magic number: 10
Function EUSDMiningIncentives._buyOtherEarnings(address,bool) (contracts/lybra/miner/EUSDMiningIncentives.sol#257-276) contains magic numbers: 10, 1e8
Function EUSDMiningIncentives.slitherConstructorVariables() (contracts/lybra/miner/EUSDMiningIncentives.sol#26-305) contains magic numbers: 10, 3000, 500
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/magic_number.md
INFO:Detectors:
EUSDMiningIncentives.etherPriceFeed (contracts/lybra/miner/EUSDMiningIncentives.sol#55) should be immutable
EUSDMiningIncentives.wETH (contracts/lybra/miner/EUSDMiningIncentives.sol#32) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Detectors:
In a function EUSDMiningIncentives.totalStaked() (contracts/lybra/miner/EUSDMiningIncentives.sol#154-164) variable EUSDMiningIncentives.vaults (contracts/lybra/miner/EUSDMiningIncentives.sol#33) is read multiple times
In a function EUSDMiningIncentives.stakedOf(address) (contracts/lybra/miner/EUSDMiningIncentives.sol#169-179) variable EUSDMiningIncentives.vaults (contracts/lybra/miner/EUSDMiningIncentives.sol#33) is read multiple times
In a function EUSDMiningIncentives.stakedLBRLpValue(address) (contracts/lybra/miner/EUSDMiningIncentives.sol#186-195) variable EUSDMiningIncentives.ethlbrLpToken (contracts/lybra/miner/EUSDMiningIncentives.sol#53) is read multiple times
In a function EUSDMiningIncentives.notifyRewardAmount(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/pessimistic-io/slitherin/blob/master/docs/multiple_storage_read.md
```

ProtocolRewardsPool.sol

```
INFO:Detectors:
ProtocolRewardsPool.getReward() (contracts/lybra/miner/ProtocolRewardsPool.sol#211-229) ignores return value by peUSD.transfer(msg.sender,peUSDAmount) (contracts/lybra/miner/ProtocolRewardsPool.sol#218)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
function ProtocolRewardsPool.updateReward(address) (contracts/lybra/miner/ProtocolRewardsPool.sol#198-202) is an unprotected initializer.
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/unprotected_initialize.md
INFO:Detectors:
ProtocolRewardsPool.grabEsLBR(uint256,bool) (contracts/lybra/miner/ProtocolRewardsPool.sol#146-160) performs a multiplication on the result of a division:
        - payAmount = amount * grabFeeRatio / 10_000 (contracts/lybra/miner/ProtocolRewardsPool.sol#149)
        - payAmount = payAmount * uint256(lbrPrice) / 1e8 (contracts/lybra/miner/ProtocolRewardsPool.sol#152)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
Reentrancy in ProtocolRewardsPool.reStake() (contracts/lybra/miner/ProtocolRewardsPool.sol#165-171):
        External calls:
        - esLBR.mint(msg.sender,amount) (contracts/lybra/miner/ProtocolRewardsPool.sol#167)
        State variables written after the call(s):
        - time2fullRedemption[msg.sender] = 0 (contracts/lybra/miner/ProtocolRewardsPool.sol#169)
        ProtocolRewardsPool.time2fullRedemption (contracts/lybra/miner/ProtocolRewardsPool.sol#39) can be used in cross function reentrancies:
        - ProtocolRewardsPool.getClaimAbleLBR(address) (contracts/lybra/miner/ProtocolRewardsPool.sol#179-183)
        - ProtocolRewardsPool.getPreUnlockableAmount(address) (contracts/lybra/miner/ProtocolRewardsPool.sol#173-177)
        - ProtocolRewardsPool.getReservedLBRForVesting(address) (contracts/lybra/miner/ProtocolRewardsPool.sol#185-189)
        - ProtocolRewardsPool.reStake() (contracts/lybra/miner/ProtocolRewardsPool.sol#165-171)
        - ProtocolRewardsPool.time2fullRedemption (contracts/lybra/miner/ProtocolRewardsPool.sol#39)
        - ProtocolRewardsPool.unlockPrematurely() (contracts/lybra/miner/ProtocolRewardsPool.sol#127-138)
        - ProtocolRewardsPool.unstake(uint256) (contracts/lybra/miner/ProtocolRewardsPool.sol#98-112)
        - unstakeRatio[msg.sender] = 0 (contracts/lybra/miner/ProtocolRewardsPool.sol#168)
        ProtocolRewardsPool.unstakeRatio (contracts/lybra/miner/ProtocolRewardsPool.sol#40) can be used in cross function reentrancies:
        - ProtocolRewardsPool.getClaimAbleLBR(address) (contracts/lybra/miner/ProtocolRewardsPool.sol#179-183)
        - ProtocolRewardsPool.getReservedLBRForVesting(address) (contracts/lybra/miner/ProtocolRewardsPool.sol#185-189)
        - ProtocolRewardsPool.reStake() (contracts/lybra/miner/ProtocolRewardsPool.sol#165-171)
        - ProtocolRewardsPool.unlockPrematurely() (contracts/lybra/miner/ProtocolRewardsPool.sol#127-138)
        - ProtocolRewardsPool.unstake(uint256) (contracts/lybra/miner/ProtocolRewardsPool.sol#98-112)
        - ProtocolRewardsPool.unstakeRatio (contracts/lybra/miner/ProtocolRewardsPool.sol#40)
Reentrancy in ProtocolRewardsPool.unlockPrematurely() (contracts/lybra/miner/ProtocolRewardsPool.sol#127-138):
        External calls:
        - LBR.mint(msg.sender,amount) (contracts/lybra/miner/ProtocolRewardsPool.sol#132)
        State variables written after the call(s):
        - time2fullRedemption[msg.sender] = 0 (contracts/lybra/miner/ProtocolRewardsPool.sol#135)
        ProtocolRewardsPool.time2fullRedemption (contracts/lybra/miner/ProtocolRewardsPool.sol#39) can be used in cross function reentrancies:
        - ProtocolRewardsPool.getClaimAbleLBR(address) (contracts/lybra/miner/ProtocolRewardsPool.sol#179-183)
        - ProtocolRewardsPool.getPreUnlockableAmount(address) (contracts/lybra/miner/ProtocolRewardsPool.sol#173-177)
        - ProtocolRewardsPool.getReservedLBRForVesting(address) (contracts/lybra/miner/ProtocolRewardsPool.sol#185-189)
        - ProtocolRewardsPool.reStake() (contracts/lybra/miner/ProtocolRewardsPool.sol#165-171)
        - ProtocolRewardsPool.time2fullRedemption (contracts/lybra/miner/ProtocolRewardsPool.sol#39)
        - ProtocolRewardsPool.unlockPrematurely() (contracts/lybra/miner/ProtocolRewardsPool.sol#127-138)
        - ProtocolRewardsPool.unstake(uint256) (contracts/lybra/miner/ProtocolRewardsPool.sol#98-112)
        - unstakeRatio[msg.sender] = 0 (contracts/lybra/miner/ProtocolRewardsPool.sol#134)
        ProtocolRewardsPool.unstakeRatio (contracts/lybra/miner/ProtocolRewardsPool.sol#40) can be used in cross function reentrancies:
        - ProtocolRewardsPool.getClaimAbleLBR(address) (contracts/lybra/miner/ProtocolRewardsPool.sol#179-183)
        - ProtocolRewardsPool.getReservedLBRForVesting(address) (contracts/lybra/miner/ProtocolRewardsPool.sol#185-189)
        - ProtocolRewardsPool.reStake() (contracts/lybra/miner/ProtocolRewardsPool.sol#165-171)
        - ProtocolRewardsPool.unlockPrematurely() (contracts/lybra/miner/ProtocolRewardsPool.sol#127-138)
        - ProtocolRewardsPool.unstake(uint256) (contracts/lybra/miner/ProtocolRewardsPool.sol#98-112)
        - ProtocolRewardsPool.unstakeRatio (contracts/lybra/miner/ProtocolRewardsPool.sol#40)
Reentrancy in ProtocolRewardsPool.unstake(uint256) (contracts/lybra/miner/ProtocolRewardsPool.sol#98-112):
        External calls:
        - esLBR.burn(msg.sender,amount) (contracts/lybra/miner/ProtocolRewardsPool.sol#103)
        - withdraw(msg.sender) (contracts/lybra/miner/ProtocolRewardsPool.sol#104)
                - LBR.mint(user,amount) (contracts/lybra/miner/ProtocolRewardsPool.sol#117)
        State variables written after the call(s):
        - time2fullRedemption[msg.sender] = block.timestamp + exitCycle (contracts/lybra/miner/ProtocolRewardsPool.sol#110)
        ProtocolRewardsPool.time2fullRedemption (contracts/lybra/miner/ProtocolRewardsPool.sol#39) can be used in cross function reentrancies:
        - ProtocolRewardsPool.getClaimAbleLBR(address) (contracts/lybra/miner/ProtocolRewardsPool.sol#179-183)
        - ProtocolRewardsPool.getPreUnlockableAmount(address) (contracts/lybra/miner/ProtocolRewardsPool.sol#173-177)
        - ProtocolRewardsPool.getReservedLBRForVesting(address) (contracts/lybra/miner/ProtocolRewardsPool.sol#185-189)
        - ProtocolRewardsPool.reStake() (contracts/lybra/miner/ProtocolRewardsPool.sol#165-171)
        - ProtocolRewardsPool.time2fullRedemption (contracts/lybra/miner/ProtocolRewardsPool.sol#39)
        - ProtocolRewardsPool.unlockPrematurely() (contracts/lybra/miner/ProtocolRewardsPool.sol#127-138)
        - ProtocolRewardsPool.unstake(uint256) (contracts/lybra/miner/ProtocolRewardsPool.sol#98-112)
        - unstakeRatio[msg.sender] = total / exitCycle (contracts/lybra/miner/ProtocolRewardsPool.sol#109)
        ProtocolRewardsPool.unstakeRatio (contracts/lybra/miner/ProtocolRewardsPool.sol#40) can be used in cross function reentrancies:
        - ProtocolRewardsPool.getClaimAbleLBR(address) (contracts/lybra/miner/ProtocolRewardsPool.sol#179-183)
        - ProtocolRewardsPool.getReservedLBRForVesting(address) (contracts/lybra/miner/ProtocolRewardsPool.sol#185-189)
        - ProtocolRewardsPool.reStake() (contracts/lybra/miner/ProtocolRewardsPool.sol#165-171)
        - ProtocolRewardsPool.unlockPrematurely() (contracts/lybra/miner/ProtocolRewardsPool.sol#127-138)
        - ProtocolRewardsPool.unstake(uint256) (contracts/lybra/miner/ProtocolRewardsPool.sol#98-112)
        - ProtocolRewardsPool.unstakeRatio (contracts/lybra/miner/ProtocolRewardsPool.sol#40)
Reentrancy in ProtocolRewardsPool.withdraw(address) (contracts/lybra/miner/ProtocolRewardsPool.sol#114-121):
        External calls:
        - LBR.mint(user,amount) (contracts/lybra/miner/ProtocolRewardsPool.sol#117)
        State variables written after the call(s):
        - lastWithdrawTime[user] = block.timestamp (contracts/lybra/miner/ProtocolRewardsPool.sol#119)
        ProtocolRewardsPool.lastWithdrawTime (contracts/lybra/miner/ProtocolRewardsPool.sol#41) can be used in cross function reentrancies:
        - ProtocolRewardsPool.getClaimAbleLBR(address) (contracts/lybra/miner/ProtocolRewardsPool.sol#179-183)
        - ProtocolRewardsPool.lastWithdrawTime (contracts/lybra/miner/ProtocolRewardsPool.sol#41)
        - ProtocolRewardsPool.withdraw(address) (contracts/lybra/miner/ProtocolRewardsPool.sol#114-121)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
ProtocolRewardsPool.stake(uint256) (contracts/lybra/miner/ProtocolRewardsPool.sol#83-87) ignores return value by LBR.burn(msg.sender,amount) (contracts/lybra/miner/ProtocolRewardsPool.sol#84)
ProtocolRewardsPool.stake(uint256) (contracts/lybra/miner/ProtocolRewardsPool.sol#83-87) ignores return value by esLBR.mint(msg.sender,amount) (contracts/lybra/miner/ProtocolRewardsPool.sol#85)
ProtocolRewardsPool.unstake(uint256) (contracts/lybra/miner/ProtocolRewardsPool.sol#98-112) ignores return value by esLBR.burn(msg.sender,amount) (contracts/lybra/miner/ProtocolRewardsPool.sol#103)
ProtocolRewardsPool.withdraw(address) (contracts/lybra/miner/ProtocolRewardsPool.sol#114-121) ignores return value by LBR.mint(user,amount) (contracts/lybra/miner/ProtocolRewardsPool.sol#117)
ProtocolRewardsPool.unlockPrematurely() (contracts/lybra/miner/ProtocolRewardsPool.sol#127-138) ignores return value by LBR.mint(msg.sender,amount) (contracts/lybra/miner/ProtocolRewardsPool.sol#132)
ProtocolRewardsPool.grabEsLBR(uint256,bool) (contracts/lybra/miner/ProtocolRewardsPool.sol#146-160) ignores return value by LBR.burn(msg.sender,payAmount) (contracts/lybra/miner/ProtocolRewardsPool.sol#156)
ProtocolRewardsPool.grabEsLBR(uint256,bool) (contracts/lybra/miner/ProtocolRewardsPool.sol#146-160) ignores return value by esLBR.mint(msg.sender,amount) (contracts/lybra/miner/ProtocolRewardsPool.sol#158)
ProtocolRewardsPool.reStake() (contracts/lybra/miner/ProtocolRewardsPool.sol#165-171) ignores return value by esLBR.mint(msg.sender,amount) (contracts/lybra/miner/ProtocolRewardsPool.sol#167)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
ProtocolRewardsPool.setGrabCost(uint256) (contracts/lybra/miner/ProtocolRewardsPool.sol#64-67) should emit an event for:
        - grabFeeRatio = _ratio (contracts/lybra/miner/ProtocolRewardsPool.sol#66)
ProtocolRewardsPool.notifyRewardAmount(uint256,uint256) (contracts/lybra/miner/ProtocolRewardsPool.sol#241-251) should emit an event for:
        - rewardPerTokenStored = rewardPerTokenStored + (amount * 1e36 / (10 ** token_decimals())) / totalStaked() (contracts/lybra/miner/ProtocolRewardsPool.sol#247)
        - rewardPerTokenStored = rewardPerTokenStored + (amount * 1e18) / totalStaked() (contracts/lybra/miner/ProtocolRewardsPool.sol#249)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
Reentrancy in ProtocolRewardsPool.unlockPrematurely() (contracts/lybra/miner/ProtocolRewardsPool.sol#127-138):
        External calls:
        - LBR.mint(msg.sender,amount) (contracts/lybra/miner/ProtocolRewardsPool.sol#132)
        State variables written after the call(s):
        - grabableAmount += burnAmount (contracts/lybra/miner/ProtocolRewardsPool.sol#136)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in ProtocolRewardsPool.getReward() (contracts/lybra/miner/ProtocolRewardsPool.sol#211-229):
        External calls:
        - peUSD.transfer(msg.sender,peUSDAmount) (contracts/lybra/miner/ProtocolRewardsPool.sol#218)
        - token.safeTransfer(msg.sender,tokenAmount) (contracts/lybra/miner/ProtocolRewardsPool.sol#222)
        Event emitted after the call(s):
        - ClaimReward(msg.sender,peUSDAmount,address(token),reward - peUSDAmount,block.timestamp) (contracts/lybra/miner/ProtocolRewardsPool.sol#223)
Reentrancy in ProtocolRewardsPool.getReward() (contracts/lybra/miner/ProtocolRewardsPool.sol#211-229):
        External calls:
        - peUSD.transfer(msg.sender,peUSDAmount) (contracts/lybra/miner/ProtocolRewardsPool.sol#218)
        Event emitted after the call(s):
        - ClaimReward(msg.sender,peUSDAmount,address(0),0,block.timestamp) (contracts/lybra/miner/ProtocolRewardsPool.sol#225)
Reentrancy in ProtocolRewardsPool.grabEsLBR(uint256,bool) (contracts/lybra/miner/ProtocolRewardsPool.sol#146-160):
        External calls:
        - success = ERC20(configurator.getEUSDAddress()).transferFrom(msg.sender,address(owner()),payAmount) (contracts/lybra/miner/ProtocolRewardsPool.sol#153)
        - LBR.burn(msg.sender,payAmount) (contracts/lybra/miner/ProtocolRewardsPool.sol#156)
        - esLBR.mint(msg.sender,amount) (contracts/lybra/miner/ProtocolRewardsPool.sol#158)
        Event emitted after the call(s):
        - GrabEsLBR(msg.sender,amount,payAmount,useEUSD,block.timestamp) (contracts/lybra/miner/ProtocolRewardsPool.sol#159)
Reentrancy in ProtocolRewardsPool.reStake() (contracts/lybra/miner/ProtocolRewardsPool.sol#165-171):
        External calls:
        - esLBR.mint(msg.sender,amount) (contracts/lybra/miner/ProtocolRewardsPool.sol#167)
        Event emitted after the call(s):
        - Restake(msg.sender,amount,block.timestamp) (contracts/lybra/miner/ProtocolRewardsPool.sol#170)
Reentrancy in ProtocolRewardsPool.stake(uint256) (contracts/lybra/miner/ProtocolRewardsPool.sol#83-87):
        External calls:
        - LBR.burn(msg.sender,amount) (contracts/lybra/miner/ProtocolRewardsPool.sol#84)
        - esLBR.mint(msg.sender,amount) (contracts/lybra/miner/ProtocolRewardsPool.sol#85)
        Event emitted after the call(s):
        - StakeLBR(msg.sender,amount,block.timestamp) (contracts/lybra/miner/ProtocolRewardsPool.sol#86)
Reentrancy in ProtocolRewardsPool.unlockPrematurely() (contracts/lybra/miner/ProtocolRewardsPool.sol#127-138):
        External calls:
        - LBR.mint(msg.sender,amount) (contracts/lybra/miner/ProtocolRewardsPool.sol#132)
        Event emitted after the call(s):
        - UnlockPrematurely(msg.sender,amount,block.timestamp) (contracts/lybra/miner/ProtocolRewardsPool.sol#137)
```

```
Reentrancy in ProtocolRewardsPool.unstake(uint256) (contracts/lybra/miner/ProtocolRewardsPool.sol#98-112):
        External calls:
        - esLBR.burn(msg.sender,amount) (contracts/lybra/miner/ProtocolRewardsPool.sol#103)
        - withdraw(msg.sender) (contracts/lybra/miner/ProtocolRewardsPool.sol#104)
            - LBR.mint(user,amount) (contracts/lybra/miner/ProtocolRewardsPool.sol#117)
        Event emitted after the call(s):
        - UnstakeLBR(msg.sender,amount,block.timestamp) (contracts/lybra/miner/ProtocolRewardsPool.sol#111)
        - WithdrawLBR(user,amount,block.timestamp) (contracts/lybra/miner/ProtocolRewardsPool.sol#120)
            - withdraw(msg.sender) (contracts/lybra/miner/ProtocolRewardsPool.sol#104)
Reentrancy in ProtocolRewardsPool.withdraw(address) (contracts/lybra/miner/ProtocolRewardsPool.sol#114-121):
        External calls:
        - LBR.mint(user,amount) (contracts/lybra/miner/ProtocolRewardsPool.sol#117)
        Event emitted after the call(s):
        - WithdrawLBR(user,amount,block.timestamp) (contracts/lybra/miner/ProtocolRewardsPool.sol#120)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
ProtocolRewardsPool.unstake(uint256) (contracts/lybra/miner/ProtocolRewardsPool.sol#98-112) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp < unLockTime (contracts/lybra/miner/ProtocolRewardsPool.sol#100)
        - time2fullRedemption[msg.sender] > block.timestamp (contracts/lybra/miner/ProtocolRewardsPool.sol#106)
ProtocolRewardsPool.unlockPrematurely() (contracts/lybra/miner/ProtocolRewardsPool.sol#127-138) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp + exitCycle - 259200 > time2fullRedemption[msg.sender],ENW) (contracts/lybra/miner/ProtocolRewardsPool.sol#128)
ProtocolRewardsPool.getClaimAbleLBR(address) (contracts/lybra/miner/ProtocolRewardsPool.sol#179-183) uses timestamp for comparisons
        Dangerous comparisons:
        - time2fullRedemption[user] > lastWithdrawTime[user] (contracts/lybra/miner/ProtocolRewardsPool.sol#180)
        - block.timestamp > time2fullRedemption[user] (contracts/lybra/miner/ProtocolRewardsPool.sol#181)
ProtocolRewardsPool.getReservedLBRForVesting(address) (contracts/lybra/miner/ProtocolRewardsPool.sol#185-189) uses timestamp for comparisons
        Dangerous comparisons:
        - time2fullRedemption[user] > block.timestamp (contracts/lybra/miner/ProtocolRewardsPool.sol#186)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Setter function ProtocolRewardsPool.setTokenAddress(address,address,address,address) (contracts/lybra/miner/ProtocolRewardsPool.sol#57-62) does not emit an event
Setter function ProtocolRewardsPool.setGrabCost(uint256) (contracts/lybra/miner/ProtocolRewardsPool.sol#64-67) does not emit an event
Setter function ProtocolRewardsPool.setLBROracle(address) (contracts/lybra/miner/ProtocolRewardsPool.sol#69-71) 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/ILBR.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/IesLBR.sol#2) allows old versions
Pragma version^0.8.17 (contracts/lybra/miner/ProtocolRewardsPool.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:
Parameter ProtocolRewardsPool.setTokenAddress(address,address,address,address)._eslbr (contracts/lybra/miner/ProtocolRewardsPool.sol#57) is not in mixedCase
Parameter ProtocolRewardsPool.setTokenAddress(address,address,address,address)._lbr (contracts/lybra/miner/ProtocolRewardsPool.sol#57) is not in mixedCase
Parameter ProtocolRewardsPool.setTokenAddress(address,address,address,address)._boost (contracts/lybra/miner/ProtocolRewardsPool.sol#57) is not in mixedCase
Parameter ProtocolRewardsPool.setTokenAddress(address,address,address,address)._lbrOracle (contracts/lybra/miner/ProtocolRewardsPool.sol#57) is not in mixedCase
Parameter ProtocolRewardsPool.setGrabCost(uint256)._ratio (contracts/lybra/miner/ProtocolRewardsPool.sol#64) is not in mixedCase
Parameter ProtocolRewardsPool.setLBROracle(address)._lbrOracle (contracts/lybra/miner/ProtocolRewardsPool.sol#69) is not in mixedCase
Parameter ProtocolRewardsPool.earned(address)._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.LBR (contracts/lybra/miner/ProtocolRewardsPool.sol#30) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Function ProtocolRewardsPool.setGrabCost(uint256) (contracts/lybra/miner/ProtocolRewardsPool.sol#64-67) contains magic number: 8000
Function ProtocolRewardsPool.unlockPrematurely() (contracts/lybra/miner/ProtocolRewardsPool.sol#127-138) contains magic number: 259200
Function ProtocolRewardsPool.grabEsLBR(uint256,bool) (contracts/lybra/miner/ProtocolRewardsPool.sol#146-160) contains magic numbers: 1e17, 10, 1e8
Function ProtocolRewardsPool.getPreUnlockableAmount(address) (contracts/lybra/miner/ProtocolRewardsPool.sol#173-177) contains magic number: 75e18
Function ProtocolRewardsPool.getReward() (contracts/lybra/miner/ProtocolRewardsPool.sol#211-229) contains magic number: 10
Function ProtocolRewardsPool.notifyRewardAmount(uint256,uint256) (contracts/lybra/miner/ProtocolRewardsPool.sol#241-251) contains magic number: 1e36
Function ProtocolRewardsPool.slitherConstructorVariables() (contracts/lybra/miner/ProtocolRewardsPool.sol#26-252) contains magic numbers: 7776000, 3000
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/magic_number.md
INFO:Detectors:
In a function ProtocolRewardsPool.getClaimAbleLBR(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.getClaimAbleLBR(address) (contracts/lybra/miner/ProtocolRewardsPool.sol#179-183) variable ProtocolRewardsPool.time2fullRedemption (contracts/lybra/miner/ProtocolRewardsPool.sol#39) is read multiple times
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/multiple_storage_read.md
```

esLBRBoost.sol

```
INFO:Detectors:
esLBRBoost.getUserBoost(address,uint256,uint256) (contracts/lybra/miner/esLBRBoost.sol#90-108) performs a multiplication on the result of a division:
        - maxBoost = ((boostEndTime - userUpdatedAt) * maxBoost) / (time - userUpdatedAt) (contracts/lybra/miner/esLBRBoost.sol#102)
        - maxBoost * userStatus.lockAmount / needLockedAmount (contracts/lybra/miner/esLBRBoost.sol#107)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
esLBRBoost.getUserBoost(address,uint256,uint256) (contracts/lybra/miner/esLBRBoost.sol#90-108) uses a dangerous strict equality:
        - userUpdatedAt >= boostEndTime || userUpdatedAt >= finishAt || userStatus.lockAmount == 0 (contracts/lybra/miner/esLBRBoost.sol#93)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Reentrancy in esLBRBoost.setLockStatus(uint256,uint256,bool) (contracts/lybra/miner/esLBRBoost.sol#55-71):
        External calls:
        - IesLBR(miningIncentives.LBR()).burn(msg.sender,lbrAmount) (contracts/lybra/miner/esLBRBoost.sol#63)
        - IesLBR(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):
        - userLockStatus[msg.sender] = LockStatus(userStatus.lockAmount + lbrAmount,block.timestamp + _setting.duration,_setting.duration,_setting.miningBoost) (contracts/lybra/miner/esLBRBoost.sol#69)
        esLBRBoost.userLockStatus (contracts/lybra/miner/esLBRBoost.sol#12) can be used in cross function reentrancies:
        - esLBRBoost.getUnlockTime(address) (contracts/lybra/miner/esLBRBoost.sol#81-83)
        - esLBRBoost.getUserBoost(address,uint256,uint256) (contracts/lybra/miner/esLBRBoost.sol#90-108)
        - esLBRBoost.setLockStatus(uint256,uint256,bool) (contracts/lybra/miner/esLBRBoost.sol#55-71)
        - esLBRBoost.unlock() (contracts/lybra/miner/esLBRBoost.sol#73-78)
        - esLBRBoost.userLockStatus (contracts/lybra/miner/esLBRBoost.sol#12)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
esLBRBoost.setLockStatus(uint256,uint256,bool) (contracts/lybra/miner/esLBRBoost.sol#55-71) ignores return value by IesLBR(miningIncentives.LBR()).burn(msg.sender,lbrAmount) (contracts/lybra/miner/esLBRBoost.sol#63)
esLBRBoost.setLockStatus(uint256,uint256,bool) (contracts/lybra/miner/esLBRBoost.sol#55-71) ignores return value by IesLBR(miningIncentives.esLBR()).mint(msg.sender,lbrAmount) (contracts/lybra/miner/esLBRBoost.sol#64)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
Reentrancy in esLBRBoost.setLockStatus(uint256,uint256,bool) (contracts/lybra/miner/esLBRBoost.sol#55-71):
        External calls:
        - IesLBR(miningIncentives.LBR()).burn(msg.sender,lbrAmount) (contracts/lybra/miner/esLBRBoost.sol#63)
        - IesLBR(miningIncentives.esLBR()).mint(msg.sender,lbrAmount) (contracts/lybra/miner/esLBRBoost.sol#64)
        Event emitted after the call(s):
        - StakeLBR(msg.sender,lbrAmount,block.timestamp) (contracts/lybra/miner/esLBRBoost.sol#65)
Reentrancy in esLBRBoost.setLockStatus(uint256,uint256,bool) (contracts/lybra/miner/esLBRBoost.sol#55-71):
        External calls:
        - IesLBR(miningIncentives.LBR()).burn(msg.sender,lbrAmount) (contracts/lybra/miner/esLBRBoost.sol#63)
        - IesLBR(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.miningBoost) (contracts/lybra/miner/esLBRBoost.sol#70)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
esLBRBoost.setLockStatus(uint256,uint256,bool) (contracts/lybra/miner/esLBRBoost.sol#55-71) uses timestamp for comparisons
        Dangerous comparisons:
        - userStatus.unlockTime > block.timestamp (contracts/lybra/miner/esLBRBoost.sol#59)
        - require(bool,string)(userStatus.duration <= _setting.duration,Your lock-in period has not ended, and the term can only be extended, not reduced.) (contracts/lybra/miner/esLBRBoost.sol#60)
        - require(bool,string)(IesLBR(miningIncentives.esLBR()).balanceOf(msg.sender) >= userStatus.lockAmount + lbrAmount,IB) (contracts/lybra/miner/esLBRBoost.sol#67)
esLBRBoost.unlock() (contracts/lybra/miner/esLBRBoost.sol#73-78) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(userStatus.unlockTime < block.timestamp,TNM) (contracts/lybra/miner/esLBRBoost.sol#75)
esLBRBoost.getUserBoost(address,uint256,uint256) (contracts/lybra/miner/esLBRBoost.sol#90-108) uses timestamp for comparisons
        Dangerous comparisons:
        - userUpdatedAt >= boostEndTime || userUpdatedAt >= finishAt || userStatus.lockAmount == 0 (contracts/lybra/miner/esLBRBoost.sol#93)
        - finishAt > boostEndTime && block.timestamp > boostEndTime (contracts/lybra/miner/esLBRBoost.sol#100)
        - userStatus.lockAmount >= needLockedAmount (contracts/lybra/miner/esLBRBoost.sol#104)
        - block.timestamp > finishAt (contracts/lybra/miner/esLBRBoost.sol#101)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Pragma version^0.8.17 (contracts/lybra/interfaces/IMiningIncentives.sol#2) allows old versions
Pragma version^0.8.17 (contracts/lybra/interfaces/IesLBR.sol#2) allows old versions
Pragma version^0.8.17 (contracts/lybra/miner/esLBRBoost.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:
Function IMiningIncentives.LBR() (contracts/lybra/interfaces/IMiningIncentives.sol#10) is not in mixedCase
Contract esLBRBoost (contracts/lybra/miner/esLBRBoost.sol#10-116) is not in CapWords
Struct esLBRBoost.esLBRLockSetting (contracts/lybra/miner/esLBRBoost.sol#16-19) is not in CapWords
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Function esLBRBoost.constructor(address) (contracts/lybra/miner/esLBRBoost.sol#35-41) contains magic numbers: 2592000, 7776000, 15552000, 31536000
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/magic_number.md
INFO:Detectors:
esLBRBoost.miningIncentives (contracts/lybra/miner/esLBRBoost.sol#13) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Detectors:
In a function esLBRBoost.constructor(address) (contracts/lybra/miner/esLBRBoost.sol#35-41) variable esLBRBoost.esLBRLockSettings (contracts/lybra/miner/esLBRBoost.sol#11) is read multiple times
In a function esLBRBoost.setLockStatus(uint256,uint256,bool) (contracts/lybra/miner/esLBRBoost.sol#55-71) variable esLBRBoost.miningIncentives (contracts/lybra/miner/esLBRBoost.sol#13) is read multiple times
In a function esLBRBoost.getAmountNeedLocked(address) (contracts/lybra/miner/esLBRBoost.sol#110-115) variable esLBRBoost.miningIncentives (contracts/lybra/miner/esLBRBoost.sol#13) is read multiple times
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/multiple_storage_read.md
```

AUTOMATED TESTING

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_initialize.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)
        - remainingRewards = (finishAt - 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#35) can be used in cross function reentrancies:
        - StakingRewardsOnArbi.balanceOf (contracts/lybra/miner/stakerewardPoolOnArbi.sol#35)
        - StakingRewardsOnArbi.earned(address) (contracts/lybra/miner/stakerewardPoolOnArbi.sol#95-97)
        - StakingRewardsOnArbi.stake(uint256) (contracts/lybra/miner/stakerewardPoolOnArbi.sol#77-83)
        - StakingRewardsOnArbi.withdraw(uint256) (contracts/lybra/miner/stakerewardPoolOnArbi.sol#86-92)
        - totalSupply += _amount (contracts/lybra/miner/stakerewardPoolOnArbi.sol#81)
        StakingRewardsOnArbi.totalSupply (contracts/lybra/miner/stakerewardPoolOnArbi.sol#33) can be used in cross function reentrancies:
        - StakingRewardsOnArbi.rewardPerToken() (contracts/lybra/miner/stakerewardPoolOnArbi.sol#68-74)
        - StakingRewardsOnArbi.stake(uint256) (contracts/lybra/miner/stakerewardPoolOnArbi.sol#77-83)
        - StakingRewardsOnArbi.totalSupply (contracts/lybra/miner/stakerewardPoolOnArbi.sol#33)
        - StakingRewardsOnArbi.withdraw(uint256) (contracts/lybra/miner/stakerewardPoolOnArbi.sol#86-92)
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):
        - StakeToken(msg.sender,_amount,block.timestamp) (contracts/lybra/miner/stakerewardPoolOnArbi.sol#82)
Reentrancy in StakingRewardsOnArbi.withdraw(uint256) (contracts/lybra/miner/stakerewardPoolOnArbi.sol#86-92):
        External calls:
        - stakingToken.safeTransfer(msg.sender,_amount) (contracts/lybra/miner/stakerewardPoolOnArbi.sol#90)
        Event emitted after the call(s):
        - WithdrawToken(msg.sender,_amount,block.timestamp) (contracts/lybra/miner/stakerewardPoolOnArbi.sol#91)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
StakingRewardsOnArbi.getReward(address,bytes) (contracts/lybra/miner/stakerewardPoolOnArbi.sol#100-108) uses timestamp for comparisons
        Dangerous comparisons:
        - reward > 0 (contracts/lybra/miner/stakerewardPoolOnArbi.sol#102)
StakingRewardsOnArbi.setRewardsDuration(uint256) (contracts/lybra/miner/stakerewardPoolOnArbi.sol#111-114) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(finishAt < block.timestamp,reward duration not finished) (contracts/lybra/miner/stakerewardPoolOnArbi.sol#112)
StakingRewardsOnArbi.notifyRewardAmount(uint256) (contracts/lybra/miner/stakerewardPoolOnArbi.sol#117-130) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp >= finishAt (contracts/lybra/miner/stakerewardPoolOnArbi.sol#118)
        - require(bool,string)(rewardRatio != 0,reward ratio = 0) (contracts/lybra/miner/stakerewardPoolOnArbi.sol#125)
StakingRewardsOnArbi._min(uint256,uint256) (contracts/lybra/miner/stakerewardPoolOnArbi.sol#132-134) uses timestamp for comparisons
        Dangerous comparisons:
        - x <= y (contracts/lybra/miner/stakerewardPoolOnArbi.sol#133)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Setter function StakingRewardsOnArbi.setRewardsDuration(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/IesLBR.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#77) is not in mixedCase
Parameter StakingRewardsOnArbi.withdraw(uint256)._amount (contracts/lybra/miner/stakerewardPoolOnArbi.sol#86) is not in mixedCase
Parameter StakingRewardsOnArbi.earned(address)._account (contracts/lybra/miner/stakerewardPoolOnArbi.sol#95) is not in mixedCase
Parameter StakingRewardsOnArbi.setRewardsDuration(uint256)._duration (contracts/lybra/miner/stakerewardPoolOnArbi.sol#111) is not in mixedCase
Parameter StakingRewardsOnArbi.notifyRewardAmount(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

```
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_initialize.md
INFO:Detectors:
Reentrancy in StakingRewardsV2.stake(uint256) (contracts/lybra/miner/stakerewardV2pool.sol#87-93):
        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#43) can be used in cross function reentrancies:
        - StakingRewardsV2.balanceOf (contracts/lybra/miner/stakerewardV2pool.sol#43)
        - StakingRewardsV2.earned(address) (contracts/lybra/miner/stakerewardV2pool.sol#109-111)
        - StakingRewardsV2.stake(uint256) (contracts/lybra/miner/stakerewardV2pool.sol#87-93)
        - StakingRewardsV2.withdraw(uint256) (contracts/lybra/miner/stakerewardV2pool.sol#96-102)
        - totalSupply += _amount (contracts/lybra/miner/stakerewardV2pool.sol#91)
        StakingRewardsV2.totalSupply (contracts/lybra/miner/stakerewardV2pool.sol#41) can be used in cross function reentrancies:
        - StakingRewardsV2.rewardPerToken() (contracts/lybra/miner/stakerewardV2pool.sol#78-84)
        - StakingRewardsV2.stake(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):
        - ClaimReward(msg.sender,reward,block.timestamp) (contracts/lybra/miner/stakerewardV2pool.sol#119)
Reentrancy in StakingRewardsV2.stake(uint256) (contracts/lybra/miner/stakerewardV2pool.sol#87-93):
        External calls:
        - stakingToken.safeTransferFrom(msg.sender,address(this),_amount) (contracts/lybra/miner/stakerewardV2pool.sol#89)
        Event emitted after the call(s):
        - StakeToken(msg.sender,_amount,block.timestamp) (contracts/lybra/miner/stakerewardV2pool.sol#92)
Reentrancy in StakingRewardsV2.withdraw(uint256) (contracts/lybra/miner/stakerewardV2pool.sol#96-102):
        External calls:
        - stakingToken.safeTransfer(msg.sender,_amount) (contracts/lybra/miner/stakerewardV2pool.sol#100)
        Event emitted after the call(s):
        - WithdrawToken(msg.sender,_amount,block.timestamp) (contracts/lybra/miner/stakerewardV2pool.sol#101)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
StakingRewardsV2.getReward() (contracts/lybra/miner/stakerewardV2pool.sol#114-121) uses timestamp for comparisons
        Dangerous comparisons:
        - reward > 0 (contracts/lybra/miner/stakerewardV2pool.sol#116)
StakingRewardsV2.setRewardsDuration(uint256) (contracts/lybra/miner/stakerewardV2pool.sol#124-128) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(finishAt < 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
        Dangerous comparisons:
        - block.timestamp >= finishAt (contracts/lybra/miner/stakerewardV2pool.sol#138)
        - require(bool,string)(rewardRatio != 0,reward ratio = 0) (contracts/lybra/miner/stakerewardV2pool.sol#145)
StakingRewardsV2._min(uint256,uint256) (contracts/lybra/miner/stakerewardV2pool.sol#152-154) uses timestamp for comparisons
        Dangerous comparisons:
        - 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/IesLBR.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#96) is not in mixedCase
```

```
Parameter StakingRewardsV2.getBoost(address)._account (contracts/lybra/miner/stakerewardV2pool.sol#104) is not in mixedCase
Parameter StakingRewardsV2.earned(address)._account (contracts/lybra/miner/stakerewardV2pool.sol#109) is not in mixedCase
Parameter StakingRewardsV2.setRewardsDuration(uint256)._duration (contracts/lybra/miner/stakerewardV2pool.sol#124) is not in mixedCase
Parameter StakingRewardsV2.setBoost(address)._boost (contracts/lybra/miner/stakerewardV2pool.sol#131) is not in mixedCase
Parameter StakingRewardsV2.notifyRewardAmount(uint256)._amount (contracts/lybra/miner/stakerewardV2pool.sol#137) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable StakingRewardsV2.constructor(address,address,address)._rewardToken (contracts/lybra/miner/stakerewardV2pool.sol#53) is too similar to StakingRewardsV2.rewardsToken (contracts/lybra/miner/stakerewardV2pool.sol#21)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
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: 1e18
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/magic_number.md
INFO:Detectors:
In a function StakingRewardsV2.notifyRewardAmount(uint256) (contracts/lybra/miner/stakerewardV2pool.sol#137-150) 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

```
INFO:Detectors:
LybraPeUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#197-214) ignores return value by PeUSD.transferFrom(_provider,address(configurator),totalFee) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#204)
LybraPeUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#197-214) ignores return value by PeUSD.transferFrom(_provider,address(configurator),amount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#210)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
LybraPeUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#123-149) performs a multiplication on the result of a division:
        - onBehalfOfCollateralRatio = (depositedAsset[onBehalfOf] * assetPrice * 100) / getBorrowedOf(onBehalfOf) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#125)
        - reducedAsset = assetAmount * onBehalfOfCollateralRatio / 1e20 (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#135)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
Reentrancy in LybraPeUSDVaultBase._mintPeUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#178-190):
        External calls:
        - configurator.refreshMintReward(_provider) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#182)
        State variables written after the call(s):
        - borrowed[_provider] += _mintAmount (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#184)
        LybraPeUSDVaultBase.borrowed (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#23) can be used in cross function reentrancies:
        - LybraPeUSDVaultBase._mintPeUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#178-190)
        - LybraPeUSDVaultBase._newFee(address) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#241-243)
        - LybraPeUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#197-214)
        - LybraPeUSDVaultBase.getBorrowedOf(address) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#257-259)
        - LybraPeUSDVaultBase.rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#160-173)
Reentrancy in LybraPeUSDVaultBase._mintPeUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#178-190):
        External calls:
        - configurator.refreshMintReward(_provider) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#182)
        - PeUSD.mint(_onBehalfOf,_mintAmount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#186)
        State variables written after the call(s):
        - poolTotalCirculation += _mintAmount (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#187)
        LybraPeUSDVaultBase.poolTotalCirculation (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#19) can be used in cross function reentrancies:
        - LybraPeUSDVaultBase._mintPeUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#178-190)
        - LybraPeUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#197-214)
        - LybraPeUSDVaultBase.getPoolTotalCirculation() (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#261-263)
Reentrancy in LybraPeUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#197-214):
        External calls:
        - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#198)
        - 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)
        State variables written after the call(s):
        - borrowed[_onBehalfOf] -= amount - totalFee (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#206)
        LybraPeUSDVaultBase.borrowed (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#23) can be used in cross function reentrancies:
        - LybraPeUSDVaultBase._mintPeUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#178-190)
        - LybraPeUSDVaultBase._newFee(address) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#241-243)
        - LybraPeUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#197-214)
        - LybraPeUSDVaultBase.getBorrowedOf(address) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#257-259)
        - LybraPeUSDVaultBase.rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#160-173)
Reentrancy in LybraPeUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#123-149):
        External calls:
        - _repay(provider,onBehalfOf,peusdAmount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#132)
                - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#198)
                - 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#212)
        State variables written after the call(s):
        - depositedAsset[onBehalfOf] -= reducedAsset (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#140)
        LybraPeUSDVaultBase.depositedAsset (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#22) can be used in cross function reentrancies:
        - LybraPeUSDVaultBase._checkHealth(address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#229-232)
        - LybraPeUSDVaultBase._withdraw(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#216-224)
        - LybraPeUSDVaultBase.depositAssetToMint(uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#59-69)
        - LybraPeUSDVaultBase.depositedAsset (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#22)
        - LybraPeUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#123-149)
        - LybraPeUSDVaultBase.rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#160-173)
Reentrancy in LybraPeUSDVaultBase.rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#160-173):
        External calls:
        - _repay(msg.sender,provider,peusdAmount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#167)
                - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#198)
                - 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#212)
        State variables written after the call(s):
        - depositedAsset[provider] -= collateralAmount (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#170)
        LybraPeUSDVaultBase.depositedAsset (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#22) can be used in cross function reentrancies:
        - LybraPeUSDVaultBase._checkHealth(address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#229-232)
        - LybraPeUSDVaultBase._withdraw(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#216-224)
        - LybraPeUSDVaultBase.depositAssetToMint(uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#59-69)
        - LybraPeUSDVaultBase.depositedAsset (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#22)
        - LybraPeUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#123-149)
        - LybraPeUSDVaultBase.rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#160-173)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
LybraPeUSDVaultBase.liquidation(address,address,uint256).reward2keeper (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#141) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
LybraPeUSDVaultBase._mintPeUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#178-190) ignores return value by PeUSD.mint(_onBehalfOf,_mintAmount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#186)
LybraPeUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#197-214) ignores return value by PeUSD.burn(_provider,amount - totalFee) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#205)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
Reentrancy in LybraPeUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#197-214):
        External calls:
        - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#198)
        State variables written after the call(s):
        - _updateFee(_onBehalfOf) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#199)
                - feeStored[user] += _newFee(user) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#236)
        - feeStored[_onBehalfOf] = 0 (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#203)
        - feeStored[_onBehalfOf] = totalFee - amount (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#209)
        - _updateFee(_onBehalfOf) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#199)
                - feeUpdatedAt[user] = block.timestamp (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#237)
Reentrancy in LybraPeUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#197-214):
        External calls:
        - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#198)
        - 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)
        State variables written after the call(s):
        - poolTotalCirculation -= amount - totalFee (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#207)
Reentrancy in LybraPeUSDVaultBase.depositAssetToMint(uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#59-69):
        External calls:
        - collateralAsset.safeTransferFrom(msg.sender,address(this),assetAmount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#61)
        State variables written after the call(s):
        - depositedAsset[msg.sender] += assetAmount (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#63)
Reentrancy in LybraRETHVault.depositEtherToMint(uint256) (contracts/lybra/pools/LybraRETHVault.sol#31-43):
        External calls:
        - IRocketDepositPool(rocketStorage.getAddress(keccak256(bytes)(abi.encodePacked(contract.address,rocketDepositPool)))).deposit{value: msg.value}() (contracts/lybra/pools/LybraRETHVault.sol#34)
        State variables written after the call(s):
        - depositedAsset[msg.sender] += balance - preBalance (contracts/lybra/pools/LybraRETHVault.sol#36)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in LybraPeUSDVaultBase._mintPeUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#178-190):
        External calls:
        - configurator.refreshMintReward(_provider) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#182)
        - PeUSD.mint(_onBehalfOf,_mintAmount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#186)
        Event emitted after the call(s):
        - Mint(_provider,_onBehalfOf,_mintAmount,block.timestamp) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#189)
Reentrancy in LybraPeUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#197-214):
        External calls:
        - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#198)
        - 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#212)
        Event emitted after the call(s):
        - Burn(_provider,_onBehalfOf,amount,block.timestamp) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#213)
Reentrancy in LybraPeUSDVaultBase._withdraw(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#216-224):
        External calls:
        - collateralAsset.safeTransfer(_onBehalfOf,_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.depositAssetToMint(uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#59-69):
        External calls:
        - collateralAsset.safeTransferFrom(msg.sender,address(this),assetAmount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#61)
        - _mintPeUSD(msg.sender,msg.sender,mintAmount,assetPrice) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#66)
                - configurator.refreshMintReward(_provider) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#182)
                - PeUSD.mint(_onBehalfOf,_mintAmount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#186)
        Event emitted after the call(s):
        - DepositAsset(msg.sender,address(collateralAsset),assetAmount,block.timestamp) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#68)
        - Mint(_provider,_onBehalfOf,_mintAmount,block.timestamp) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#189)
                - _mintPeUSD(msg.sender,msg.sender,mintAmount,assetPrice) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#66)
Reentrancy in LybraRETHVault.depositEtherToMint(uint256) (contracts/lybra/pools/lybraRETHVault.sol#31-43):
        External calls:
        - IRocketDepositPool(rocketStorage.getAddress(keccak256(bytes)(abi.encodePacked(contract.address,rocketDepositPool)))).deposit{value: msg.value}() (contracts/lybra/pools/LybraRETHVault.sol#34)
        - _mintPeUSD(msg.sender,msg.sender,mintAmount,getAssetPrice()) (contracts/lybra/pools/LybraRETHVault.sol#39)
                - etherOracle.fetchPrice() (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#209)
                - configurator.refreshMintReward(_provider) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#182)
                - PeUSD.mint(_onBehalfOf,_mintAmount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#186)
        External calls sending eth:
        - IRocketDepositPool(rocketStorage.getAddress(keccak256(bytes)(abi.encodePacked(contract.address,rocketDepositPool)))).deposit{value: msg.value}() (contracts/lybra/pools/LybraRETHVault.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,_mintAmount,block.timestamp) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#189)
                - _mintPeUSD(msg.sender,msg.sender,mintAmount,getAssetPrice()) (contracts/lybra/pools/LybraRETHVault.sol#39)
Reentrancy in LybraPeUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#123-149):
        External calls:
        - _repay(provider,onBehalfOf,peusdAmount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#132)
                - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#198)
                - 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#212)
        - collateralAsset.safeTransfer(msg.sender,reward2keeper) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#145)
        - collateralAsset.safeTransfer(provider,reducedAsset - reward2keeper) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#147)
        Event emitted after the call(s):
        - LiquidationRecord(provider,msg.sender,onBehalfOf,peusdAmount,reducedAsset,reward2keeper,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,peusdAmount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#167)
                - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#198)
                - 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#212)
        - collateralAsset.safeTransfer(msg.sender,collateralAmount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#171)
        Event emitted after the call(s):
        - RigidRedemption(msg.sender,provider,peusdAmount,collateralAmount,block.timestamp) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#172)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

INFO:Detectors:
LybraPeUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#123-149) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(onBehalfOfCollateralRatio < configurator.getBadCollateralRatio(address(this)),Borrowers collateral ratio should below badCollateralRatio) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#126)
        - onBehalfOfCollateralRatio > 1e20 && onBehalfOfCollateralRatio < 11e19 (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#134)
        - onBehalfOfCollateralRatio >= 11e19 (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#137)
        - msg.sender != provider && onBehalfOfCollateralRatio >= 1e20 + keeperRatio * 1e18 (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#143)
LybraPeUSDVaultBase.rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#160-173) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(borrowed[provider] >= peusdAmount,peusdAmount cannot surpass providers debt) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#163)
        - require(bool,string)(providerCollateralRatio >= 100 * 1e18,The provider's collateral ratio should be not less than 100%.) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#166)
LybraPeUSDVaultBase._mintPeUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#178-190) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(poolTotalCirculation + _mintAmount <= configurator.mintVaultMaxSupply(address(this)),ESL) (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:
        - amount > totalFee (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#202)
        - borrowed[_onBehalfOf] + totalFee >= _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:
        - getBorrowedOf(_provider) > 0 (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#220)
LybraPeUSDVaultBase._checkHealth(address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#229-232) uses timestamp for comparisons
        Dangerous comparisons:
        - ((depositedAsset[user] * price * 100) / getBorrowedOf(user)) < configurator.getSafeCollateralRatio(address(this)) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#230)
LybraPeUSDVaultBase._updateFee(address) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#234-239) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp > feeUpdatedAt[user] (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#235)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Function LybraRETHVault.getAssetPrice() (contracts/lybra/pools/lybraRETHVault.sol#45-47) has a dubious typecast: IRETH=address
Function LybraRETHVault.getAsset2EtherExchangeRate() (contracts/lybra/pools/LybraRETHVault.sol#49-51) has a dubious typecast: IRETH=address
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/dubious_typecast.md
INFO:Detectors:
Pragma version^0.8.17 (contracts/lybra/interfaces/IEUSD.sol#2) allows old versions
Pragma version^0.8.17 (contracts/lybra/interfaces/IPeUSD.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/pools/LybraRETHVault.sol#3) 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: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Parameter IEUSD.getSharesByMintedEUSD(uint256)._EUSDAmount (contracts/lybra/interfaces/IEUSD.sol#32) is not in mixedCase
Variable LybraPeUSDVaultBase.PeUSD (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#16) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Function LybraPeUSDVaultBase.depositAssetToMint(uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#59-69) contains magic number: 10000000000000000
Function LybraPeUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#123-149) contains magic numbers: 100, 1e20, 1e20, 11e19, 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#229-232) contains magic number: 100
Function LybraPeUSDVaultBase._newFee(address) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#241-243) contains magic number: 86
Function LybraRETHVault.depositEtherToMint(uint256) (contracts/lybra/pools/lybraRETHVault.sol#31-43) contains magic number: 10000000000000000
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/magic_number.md
INFO:Detectors:
In a function LybraRETHVault.depositEtherToMint(uint256) (contracts/lybra/pools/lybraRETHVault.sol#31-43) variable LybraPeUSDVaultBase.collateralAsset (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#17) is read multiple times
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/multiple_storage_read.md
```

LybraStETHVault.sol

```
INFO:Detectors:
LybraStETHVault.getDutchAuctionDiscountPrice() (contracts/lybra/pools/LybraStETHVault.sol#102-106) uses a weak PRNG: "time = (block.timestamp - lidoRebaseTime) % 86400 (contracts/lybra/pools/LybraStETHVault.sol#103)"
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG
INFO:Detectors:
LybraStETHVault.excessIncomeDistribution(uint256) (contracts/lybra/pools/LybraStETHVault.sol#62-96) ignores return value by collateralAsset.transfer(msg.sender,realAmount) (contracts/lybra/pools/LybraStETHVault.sol#94)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
Function LybraStETHVault.setLidoRebaseTime(uint256) (contracts/lybra/pools/LybraStETHVault.sol#25-28) is a non-protected setter lidoRebaseTime is written
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/unprotected_setter.md
INFO:Detectors:
LybraEUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#148-178) performs a multiplication on the result of a division:
        - onBehalfOfCollateralRatio = (depositedAsset[onBehalfOf] * assetPrice * 100) / borrowed[onBehalfOf] (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#150)
        - reducedAsset = assetAmount * onBehalfOfCollateralRatio / 1e20 (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#161)
LybraEUSDVaultBase.superLiquidation(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#189-213) performs a multiplication on the result of a division:
        - eusdAmount = (assetAmount * assetPrice) / 1e18 (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#195)
        - eusdAmount = (eusdAmount * 1e20) / onBehalfOfCollateralRatio (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#197)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
LybraStETHVault.excessIncomeDistribution(uint256) (contracts/lybra/pools/LybraStETHVault.sol#62-96) uses a dangerous strict equality:
        - sharesAmount == 0 (contracts/lybra/pools/LybraStETHVault.sol#77)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities
INFO:Detectors:
Reentrancy in LybraEUSDVaultBase._mintEUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#263-273):
        External calls:
        - configurator.refreshMintReward(_provider) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#265)
        - EUSD.mint(_onBehalfOf,_mintAmount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#268)
        State variables written after the call(s):
        - poolTotalCirculation += _mintAmount (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#270)
        LybraEUSDVaultBase.poolTotalCirculation (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#24) can be used in cross function reentrancies:
        - LybraEUSDVaultBase._mintEUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#263-273)
        - LybraEUSDVaultBase._newFee() (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#304-306)
        - LybraEUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#280-290)
        - LybraEUSDVaultBase.getPoolTotalCirculation() (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#319-321)
        - LybraEUSDVaultBase.superLiquidation(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#189-213)
Reentrancy in LybraEUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#280-290):
        External calls:
        - EUSD.burn(_provider,amount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#283)
        - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#284)
```

```
        State variables written after the call(s):
        - poolTotalCirculation += _mintAmount (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#270)
        LybraEUSDVaultBase.poolTotalCirculation (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#24) can be used in cross function reentrancies:
        - LybraEUSDVaultBase._mintEUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#263-273)
        - LybraEUSDVaultBase._newFee() (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#304-306)
        - LybraEUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#280-290)
        - LybraEUSDVaultBase.getPoolTotalCirculation() (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#319-321)
        - LybraEUSDVaultBase.superLiquidation(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#189-213)
Reentrancy in LybraEUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#280-290):
        External calls:
        - EUSD.burn(_provider,amount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#283)
        - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#284)
        State variables written after the call(s):
        - borrowed[_onBehalfOf] -= amount (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#286)
        LybraEUSDVaultBase.borrowed (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#27) can be used in cross function reentrancies:
        - LybraEUSDVaultBase._checkHealth(address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#295-297)
        - LybraEUSDVaultBase._mintEUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#263-273)
        - LybraEUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#280-290)
        - LybraEUSDVaultBase.getBorrowedOf(address) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#315-317)
        - LybraEUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#148-178)
        - LybraEUSDVaultBase.rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#234-248)
        - LybraEUSDVaultBase.superLiquidation(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#189-213)
        - LybraEUSDVaultBase.withdraw(address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#93-107)
Reentrancy in LybraStETHVault.excessIncomeDistribution(uint256) (contracts/lybra/pools/LybraStETHVault.sol#62-96):
        External calls:
        - payAmount = realAmount * getAssetPrice() * dutchAuctionDiscountPrice / 10_000 / 1e18 (contracts/lybra/pools/LybraStETHVault.sol#67)
                - etherOracle.fetchPrice() (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#312)
        - success = EUSD.transferFrom(msg.sender,address(configurator),income) (contracts/lybra/pools/LybraStETHVault.sol#71)
        - configurator.distributeRewards() (contracts/lybra/pools/LybraStETHVault.sol#74)
        - EUSD.burnShares(msg.sender,sharesAmount) (contracts/lybra/pools/LybraStETHVault.sol#82)
        State variables written after the call(s):
        - feeStored = 0 (contracts/lybra/pools/LybraStETHVault.sol#83)
        LybraEUSDVaultBase.feeStored (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#28) can be used in cross function reentrancies:
        - LybraEUSDVaultBase._saveReport() (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#299-302)
        - LybraStETHVault.excessIncomeDistribution(uint256) (contracts/lybra/pools/LybraStETHVault.sol#62-96)
        - LybraEUSDVaultBase.feeStored (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#28)
Reentrancy in LybraStETHVault.excessIncomeDistribution(uint256) (contracts/lybra/pools/LybraStETHVault.sol#62-96):
        External calls:
        - payAmount = realAmount * getAssetPrice() * dutchAuctionDiscountPrice / 10_000 / 1e18 (contracts/lybra/pools/LybraStETHVault.sol#67)
                - etherOracle.fetchPrice() (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#312)
        - success_scope_0 = EUSD.transferFrom(msg.sender,address(configurator),payAmount) (contracts/lybra/pools/LybraStETHVault.sol#86)
        - configurator.distributeRewards() (contracts/lybra/pools/LybraStETHVault.sol#88)
        State variables written after the call(s):
        - feeStored = income - payAmount (contracts/lybra/pools/LybraStETHVault.sol#89)
        LybraEUSDVaultBase.feeStored (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#28) can be used in cross function reentrancies:
        - LybraEUSDVaultBase._saveReport() (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#299-302)
        - LybraStETHVault.excessIncomeDistribution(uint256) (contracts/lybra/pools/LybraStETHVault.sol#62-96)
        - LybraEUSDVaultBase.feeStored (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#28)
Reentrancy in LybraStETHVault.excessIncomeDistribution(uint256) (contracts/lybra/pools/LybraStETHVault.sol#62-96):
        External calls:
        - payAmount = realAmount * getAssetPrice() * dutchAuctionDiscountPrice / 10_000 / 1e18 (contracts/lybra/pools/LybraStETHVault.sol#67)
                - etherOracle.fetchPrice() (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#312)
        - success = EUSD.transferFrom(msg.sender,address(configurator),income) (contracts/lybra/pools/LybraStETHVault.sol#71)
        - configurator.distributeRewards() (contracts/lybra/pools/LybraStETHVault.sol#74)
        - EUSD.burnShares(msg.sender,sharesAmount) (contracts/lybra/pools/LybraStETHVault.sol#82)
        - success_scope_0 = EUSD.transferFrom(msg.sender,address(configurator),payAmount) (contracts/lybra/pools/LybraStETHVault.sol#86)
        - configurator.distributeRewards() (contracts/lybra/pools/LybraStETHVault.sol#88)
        State variables written after the call(s):
        - lastReportTime = block.timestamp (contracts/lybra/pools/LybraStETHVault.sol#93)
        LybraEUSDVaultBase.lastReportTime (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#23) can be used in cross function reentrancies:
        - LybraEUSDVaultBase._newFee() (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#304-306)
        - LybraEUSDVaultBase._saveReport() (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#299-302)
        - LybraStETHVault.excessIncomeDistribution(uint256) (contracts/lybra/pools/LybraStETHVault.sol#62-96)
        - LybraEUSDVaultBase.lastReportTime (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#23)
Reentrancy in LybraEUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#148-178):
        External calls:
        - _repay(provider,onBehalfOf,eusdAmount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#157)
                - EUSD.burn(_provider,amount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#283)
                - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#284)
        State variables written after the call(s):
        - depositedAsset[onBehalfOf] -= reducedAsset (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#167)
        LybraEUSDVaultBase.depositedAsset (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#26) can be used in cross function reentrancies:
        - LybraEUSDVaultBase._checkHealth(address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#295-297)
        - LybraEUSDVaultBase.depositAssetToMint(uint256,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#70-81)
        - LybraEUSDVaultBase.depositedAsset (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#26)
        - LybraEUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#148-178)
        - LybraEUSDVaultBase.rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#234-248)
        - LybraEUSDVaultBase.superLiquidation(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#189-213)
        - LybraEUSDVaultBase.withdraw(address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#93-107)
Reentrancy in LybraEUSDVaultBase.rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#234-248):
        External calls:
        - _repay(msg.sender,provider,eusdAmount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#241)
                - EUSD.burn(_provider,amount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#283)
                - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#284)
        State variables written after the call(s):
        - depositedAsset[provider] -= collateralAmount (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#244)
        LybraEUSDVaultBase.depositedAsset (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#26) can be used in cross function reentrancies:
        - LybraEUSDVaultBase._checkHealth(address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#295-297)
        - LybraEUSDVaultBase.depositAssetToMint(uint256,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#70-81)
        - LybraEUSDVaultBase.depositedAsset (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#26)
        - LybraEUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#148-178)
        - LybraEUSDVaultBase.rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#234-248)
        - LybraEUSDVaultBase.superLiquidation(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#189-213)
        - LybraEUSDVaultBase.withdraw(address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#93-107)
Reentrancy in LybraEUSDVaultBase.superLiquidation(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#189-213):
        External calls:
        - _repay(provider,onBehalfOf,eusdAmount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#201)
                - EUSD.burn(_provider,amount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#283)
                - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#284)
        State variables written after the call(s):
        - depositedAsset[onBehalfOf] -= assetAmount (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#204)
        LybraEUSDVaultBase.depositedAsset (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#26) can be used in cross function reentrancies:
        - LybraEUSDVaultBase._checkHealth(address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#295-297)
        - LybraEUSDVaultBase.depositAssetToMint(uint256,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#70-81)
        - LybraEUSDVaultBase.depositedAsset (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#26)
        - LybraEUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#148-178)
        - LybraEUSDVaultBase.rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#234-248)
        - LybraEUSDVaultBase.superLiquidation(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#189-213)
        - LybraEUSDVaultBase.withdraw(address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#93-107)
        - totalDepositedAsset -= assetAmount (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#203)
        LybraEUSDVaultBase.totalDepositedAsset (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#22) can be used in cross function reentrancies:
        - LybraEUSDVaultBase.depositAssetToMint(uint256,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#70-81)
        - LybraEUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#148-178)
        - LybraEUSDVaultBase.rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#234-248)
        - LybraEUSDVaultBase.superLiquidation(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#189-213)
        - LybraEUSDVaultBase.totalDepositedAsset (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#22)
        - LybraEUSDVaultBase.withdraw(address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#93-107)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
LybraEUSDVaultBase.superLiquidation(address,address,uint256).reward2keeper (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#205) is a local variable never initialized
LybraEUSDVaultBase.liquidation(address,address,uint256).reward2keeper (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#169) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
LybraStETHVault.excessIncomeDistribution(uint256) (contracts/lybra/pools/LybraStETHVault.sol#62-96) ignores return value by EUSD.burnShares(msg.sender,sharesAmount) (contracts/lybra/pools/LybraStETHVault.sol#82)
LybraEUSDVaultBase._mintEUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#263-273) ignores return value by EUSD.mint(_onBehalfOf,_mintAmount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#268)
LybraEUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#280-290) ignores return value by EUSD.burn(_provider,amount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#283)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
Reentrancy in LybraEUSDVaultBase._mintEUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#263-273):
        External calls:
        - configurator.refreshMintReward(_provider) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#265)
        State variables written after the call(s):
        - borrowed[_provider] += _mintAmount (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#266)
Reentrancy in LybraEUSDVaultBase._mintEUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#263-273):
        External calls:
        - configurator.refreshMintReward(_provider) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#265)
        - EUSD.mint(_onBehalfOf,_mintAmount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#268)
        State variables written after the call(s):
        - _saveReport() (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#269)
                - feeStored += _newFee() (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#300)
        - _saveReport() (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#269)
                - lastReportTime = block.timestamp (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#301)
Reentrancy in LybraEUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#280-290):
        External calls:
        - EUSD.burn(_provider,amount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#283)
        - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#284)
        State variables written after the call(s):
        - _saveReport() (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#287)
                - feeStored += _newFee() (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#300)
        - _saveReport() (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#287)
                - lastReportTime = block.timestamp (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#301)
        - poolTotalCirculation -= amount (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#288)
Reentrancy in LybraEUSDVaultBase.depositAssetToMint(uint256,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#70-81):
        External calls:
        - collateralAsset.safeTransferFrom(msg.sender,address(this),assetAmount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#72)
        State variables written after the call(s):
        - depositedAsset[msg.sender] += assetAmount (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#74)
        - depositedTime[msg.sender] = block.timestamp (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#75)
        - totalDepositedAsset += assetAmount (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#73)
```

```
Reentrancy in LybraStETHVault.depositEtherToMint(uint256) (contracts/lybra/pools/LybraStETHVault.sol#37-52):
        External calls:
        - sharesAmount = Ilido(address(collateralAsset)).submit{value: msg.value}(address(configurator)) (contracts/lybra/pools/LybraStETHVault.sol#40)
        State variables written after the call(s):
        - depositedAsset[msg.sender] += msg.value (contracts/lybra/pools/LybraStETHVault.sol#44)
        - depositedTime[msg.sender] = block.timestamp (contracts/lybra/pools/LybraStETHVault.sol#45)
        - totalDepositedAsset += msg.value (contracts/lybra/pools/LybraStETHVault.sol#43)
Reentrancy in LybraEUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#148-178):
        External calls:
        - _repay(provider,onBehalfOf,eusdAmount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#157)
                - EUSD.burn(_provider,amount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#283)
                - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#284)
        State variables written after the call(s):
        - totalDepositedAsset -= reducedAsset (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#166)
Reentrancy in LybraEUSDVaultBase.rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#234-248):
        External calls:
        - _repay(msg.sender,provider,eusdAmount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#241)
                - EUSD.burn(_provider,amount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#283)
                - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#284)
        State variables written after the call(s):
        - totalDepositedAsset -= collateralAmount (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#245)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in LybraEUSDVaultBase._mintEUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#263-273):
        External calls:
        - configurator.refreshMintReward(_provider) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#265)
        - EUSD.mint(_onBehalfOf,_mintAmount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#268)
        Event emitted after the call(s):
        - Mint(msg.sender,_onBehalfOf,_mintAmount,block.timestamp) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#272)
Reentrancy in LybraEUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#280-290):
        External calls:
        - EUSD.burn(_provider,amount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#283)
        - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#284)
        Event emitted after the call(s):
        - Burn(_provider,_onBehalfOf,amount,block.timestamp) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#289)
Reentrancy in LybraEUSDVaultBase.depositAssetToMint(uint256,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#70-81):
        External calls:
        - collateralAsset.safeTransferFrom(msg.sender,address(this),assetAmount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#72)
        - _mintEUSD(msg.sender,msg.sender,mintAmount,getAssetPrice()) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#78)
                - configurator.refreshMintReward(_provider) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#265)
                - EUSD.mint(_onBehalfOf,_mintAmount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#268)
        Event emitted after the call(s):
        - DepositAsset(msg.sender,address(collateralAsset),assetAmount,block.timestamp) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#80)
        - Mint(msg.sender,_onBehalfOf,_mintAmount,block.timestamp) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#272)
                - _mintEUSD(msg.sender,msg.sender,mintAmount,getAssetPrice()) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#78)
Reentrancy in LybraStETHVault.depositEtherToMint(uint256) (contracts/lybra/pools/LybraStETHVault.sol#37-52):
        External calls:
        - sharesAmount = Ilido(address(collateralAsset)).submit{value: msg.value}(address(configurator)) (contracts/lybra/pools/LybraStETHVault.sol#40)
        - _mintEUSD(msg.sender,msg.sender,mintAmount,getAssetPrice()) (contracts/lybra/pools/LybraStETHVault.sol#48)
                - etherOracle.fetchPrice() (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#312)
                - configurator.refreshMintReward(_provider) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#265)
                - EUSD.mint(_onBehalfOf,_mintAmount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#268)
        External calls sending eth:
        - sharesAmount = Ilido(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,block.timestamp) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#272)
                - _mintEUSD(msg.sender,msg.sender,mintAmount,getAssetPrice()) (contracts/lybra/pools/LybraStETHVault.sol#48)
Reentrancy in LybraStETHVault.excessIncomeDistribution(uint256) (contracts/lybra/pools/LybraStETHVault.sol#62-96):
        External calls:
        - payAmount = realAmount * getAssetPrice() * dutchAuctionDiscountPrice / 10_000 / 1e18 (contracts/lybra/pools/LybraStETHVault.sol#67)
                - etherOracle.fetchPrice() (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#312)
        - success = EUSD.transferFrom(msg.sender,address(configurator),income) (contracts/lybra/pools/LybraStETHVault.sol#71)
        - configurator.distributeRewards() (contracts/lybra/pools/LybraStETHVault.sol#74)
        - EUSD.burnShares(msg.sender,sharesAmount) (contracts/lybra/pools/LybraStETHVault.sol#82)
        Event emitted after the call(s):
        - FeeDistribution(address(configurator),income,block.timestamp) (contracts/lybra/pools/LybraStETHVault.sol#84)
Reentrancy in LybraStETHVault.excessIncomeDistribution(uint256) (contracts/lybra/pools/LybraStETHVault.sol#62-96):
        External calls:
        - payAmount = realAmount * getAssetPrice() * dutchAuctionDiscountPrice / 10_000 / 1e18 (contracts/lybra/pools/LybraStETHVault.sol#67)
                - etherOracle.fetchPrice() (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#312)
        - success_scope_0 = EUSD.transferFrom(msg.sender,address(configurator),payAmount) (contracts/lybra/pools/LybraStETHVault.sol#86)
        - configurator.distributeRewards() (contracts/lybra/pools/LybraStETHVault.sol#88)
        Event emitted after the call(s):
        - FeeDistribution(address(configurator),payAmount,block.timestamp) (contracts/lybra/pools/LybraStETHVault.sol#90)
Reentrancy in LybraStETHVault.excessIncomeDistribution(uint256) (contracts/lybra/pools/LybraStETHVault.sol#62-96):
        External calls:
        - payAmount = realAmount * getAssetPrice() * dutchAuctionDiscountPrice / 10_000 / 1e18 (contracts/lybra/pools/LybraStETHVault.sol#67)
                - etherOracle.fetchPrice() (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#312)
        - success = EUSD.transferFrom(msg.sender,address(configurator),income) (contracts/lybra/pools/LybraStETHVault.sol#71)
        - configurator.distributeRewards() (contracts/lybra/pools/LybraStETHVault.sol#74)
        - EUSD.burnShares(msg.sender,sharesAmount) (contracts/lybra/pools/LybraStETHVault.sol#82)
        - success_scope_0 = EUSD.transferFrom(msg.sender,address(configurator),payAmount) (contracts/lybra/pools/LybraStETHVault.sol#86)
        - configurator.distributeRewards() (contracts/lybra/pools/LybraStETHVault.sol#88)
        - collateralAsset.transfer(msg.sender,realAmount) (contracts/lybra/pools/LybraStETHVault.sol#94)
        Event emitted after the call(s):
        - LSDValueCaptured(realAmount,payAmount,dutchAuctionDiscountPrice,block.timestamp) (contracts/lybra/pools/LybraStETHVault.sol#95)
Reentrancy in LybraEUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#148-178):
        External calls:
        - _repay(provider,onBehalfOf,eusdAmount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#157)
                - EUSD.burn(_provider,amount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#283)
                - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#284)
        - collateralAsset.safeTransfer(msg.sender,reward2keeper) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#173)
        - collateralAsset.safeTransfer(provider,reducedAsset - reward2keeper) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#175)
        Event emitted after the call(s):
        - LiquidationRecord(provider,msg.sender,onBehalfOf,eusdAmount,reducedAsset,reward2keeper,false,block.timestamp) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#177)
Reentrancy in LybraEUSDVaultBase.rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#234-248):
        External calls:
        - _repay(msg.sender,provider,eusdAmount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#241)
                - EUSD.burn(_provider,amount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#283)
                - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#284)
        - collateralAsset.safeTransfer(msg.sender,checkWithdrawal(msg.sender,collateralAmount)) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#246)
        Event emitted after the call(s):
        - RigidRedemption(msg.sender,provider,eusdAmount,collateralAmount,block.timestamp) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#247)
Reentrancy in LybraEUSDVaultBase.superLiquidation(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#189-213):
        External calls:
        - _repay(provider,onBehalfOf,eusdAmount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#201)
                - EUSD.burn(_provider,amount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#283)
                - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#284)
        - collateralAsset.safeTransfer(msg.sender,reward2keeper) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#208)
        - collateralAsset.safeTransfer(provider,assetAmount - reward2keeper) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#210)
        Event emitted after the call(s):
        - LiquidationRecord(provider,msg.sender,onBehalfOf,eusdAmount,assetAmount,reward2keeper,true,block.timestamp) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#212)
Reentrancy in LybraEUSDVaultBase.withdraw(address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#93-107):
        External calls:
        - collateralAsset.safeTransfer(onBehalfOf,withdrawal) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#102)
        Event emitted after the call(s):
        - WithdrawAsset(msg.sender,address(collateralAsset),onBehalfOf,withdrawal,block.timestamp) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#106)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
LybraStETHVault.excessIncomeDistribution(uint256) (contracts/lybra/pools/LybraStETHVault.sol#62-96) uses timestamp for comparisons
        Dangerous comparisons:
        - payAmount > income (contracts/lybra/pools/LybraStETHVault.sol#70)
        - require(bool,string)(success,TF) (contracts/lybra/pools/LybraStETHVault.sol#72)
        - sharesAmount == 0 (contracts/lybra/pools/LybraStETHVault.sol#77)
        - require(bool,string)(success_scope_0,TF) (contracts/lybra/pools/LybraStETHVault.sol#87)
LybraStETHVault.getDutchAuctionDiscountPrice() (contracts/lybra/pools/LybraStETHVault.sol#102-106) uses timestamp for comparisons
        Dangerous comparisons:
        - time < 1800 (contracts/lybra/pools/LybraStETHVault.sol#104)
LybraEUSDVaultBase.checkWithdrawal(address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#109-111) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp - 259200 >= depositedTime[user] (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#110)
Reference: 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: Ilido<=address
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/dubious_typecast.md
INFO:Detectors:
Setter function LybraStETHVault.setLidoRebaseTime(uint256) (contracts/lybra/pools/LybraStETHVault.sol#25-28) 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/IEUSD.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/pools/LybraStETHVault.sol#3) allows old versions
Pragma version^0.8.17 (contracts/lybra/pools/base/LybraEUSDVaultBase.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:
Parameter IEUSD.getSharesByMintedEUSD(uint256)._EUSDAmount (contracts/lybra/interfaces/IEUSD.sol#32) is not in mixedCase
Parameter LybraStETHVault.setLidoRebaseTime(uint256)._time (contracts/lybra/pools/LybraStETHVault.sol#25) is not in mixedCase
Variable LybraEUSDVaultBase.EUSD (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#16) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Function LybraEUSDVaultBase.depositAssetToMint(uint256,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#70-81) contains magic number: 10000000000000000
Function LybraEUSDVaultBase.checkWithdrawal(address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#109-111) contains magic numbers: 259200, 999
Function LybraEUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#148-178) contains magic numbers: 100, 1e20, 1e20, 11e19, 11, 1e20, 100
```

```
Function LybraEUSDVaultBase.superLiquidation(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#189-213) contains magic numbers: 100, 100, 125, 1e20, 1e20, 1e20
Function LybraEUSDVaultBase.rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#234-248) contains magic numbers: 100, 100
Function LybraEUSDVaultBase._checkHealth(address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#295-297) contains magic number: 100
Function LybraEUSDVaultBase._newFee() (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#304-306) contains magic number: 86
Function LybraSTETHVault.depositEtherToMint(uint256) (contracts/lybra/pools/LybraSTETHVault.sol#37-52) contains magic number: 10000000000000000
Function LybraSTETHVault.excessIncomeDistribution(uint256) (contracts/lybra/pools/LybraSTETHVault.sol#62-96) contains magic number: 10
Function LybraSTETHVault.getDutchAuctionDiscountPrice() (contracts/lybra/pools/LybraSTETHVault.sol#102-106) contains magic numbers: 86400, 1800, 10, 10
Function LybraSTETHVault.slitherConstructorVariables() (contracts/lybra/pools/LybraSTETHVault.sol#12-114) contains magic number: 43200
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/magic_number.md
INFO:Detectors:
In a function LybraSTETHVault.excessIncomeDistribution(uint256) (contracts/lybra/pools/LybraSTETHVault.sol#62-96) variable LybraEUSDVaultBase.EUSD (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#16) is read multiple times
In a function LybraSTETHVault.excessIncomeDistribution(uint256) (contracts/lybra/pools/LybraSTETHVault.sol#62-96) variable LybraEUSDVaultBase.configurator (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#18) is read multiple times
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/multiple_storage_read.md
```

LybraWbETHVault.sol

```
INFO:Detectors:
LybraPeUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#197-214) ignores return value by PeUSD.transferFrom(_provider,address(configurator),totalFee) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#204)
LybraPeUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#197-214) ignores return value by PeUSD.transferFrom(_provider,address(configurator),amount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#210)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
LybraPeUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#123-149) performs a multiplication on the result of a division:
        - onBehalfOfCollateralRatio = (depositedAsset[onBehalfOf] * assetPrice * 100) / getBorrowedOf(onBehalfOf) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#125)
        - reducedAsset = assetAmount * onBehalfOfCollateralRatio / 1e20 (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#135)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
Reentrancy in LybraPeUSDVaultBase._mintPeUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#178-190):
        External calls:
        - configurator.refreshMintReward(_provider) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#182)
        State variables written after the call(s):
        - borrowed[_provider] += _mintAmount (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#184)
        LybraPeUSDVaultBase.borrowed (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#23) can be used in cross function reentrancies:
        - LybraPeUSDVaultBase._mintPeUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#178-190)
        - LybraPeUSDVaultBase._newFee(address) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#241-243)
        - LybraPeUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#197-214)
        - LybraPeUSDVaultBase.getBorrowedOf(address) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#257-259)
        - LybraPeUSDVaultBase.rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#160-173)
Reentrancy in LybraPeUSDVaultBase._mintPeUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#178-190):
        External calls:
        - configurator.refreshMintReward(_provider) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#182)
        - PeUSD.mint(_onBehalfOf,_mintAmount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#186)
        State variables written after the call(s):
        - poolTotalCirculation += _mintAmount (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#187)
        LybraPeUSDVaultBase.poolTotalCirculation (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#19) can be used in cross function reentrancies:
        - LybraPeUSDVaultBase._mintPeUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#178-190)
        - LybraPeUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#197-214)
        - LybraPeUSDVaultBase.getPoolTotalCirculation() (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#261-263)
Reentrancy in LybraPeUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#197-214):
        External calls:
        - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#198)
        - 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)
        State variables written after the call(s):
        - borrowed[_onBehalfOf] -= amount - totalFee (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#206)
        LybraPeUSDVaultBase.borrowed (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#23) can be used in cross function reentrancies:
        - LybraPeUSDVaultBase._mintPeUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#178-190)
        - LybraPeUSDVaultBase._newFee(address) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#241-243)
        - LybraPeUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#197-214)
        - LybraPeUSDVaultBase.getBorrowedOf(address) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#257-259)
        - LybraPeUSDVaultBase.rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#160-173)
Reentrancy in LybraPeUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#123-149):
        External calls:
        - _repay(provider,onBehalfOf,pesudAmount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#132)
                - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#198)
                - 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#212)
        State variables written after the call(s):
        - depositedAsset[onBehalfOf] -= reducedAsset (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#140)
        LybraPeUSDVaultBase.depositedAsset (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#22) can be used in cross function reentrancies:
        - LybraPeUSDVaultBase._checkHealth(address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#229-232)
        - LybraPeUSDVaultBase._withdraw(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#216-224)
        - LybraPeUSDVaultBase.depositAssetToMint(uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#59-69)
        - LybraPeUSDVaultBase.depositedAsset (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#22)
        - LybraPeUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#123-149)
        - LybraPeUSDVaultBase.rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#160-173)
Reentrancy in LybraPeUSDVaultBase.rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#160-173):
        External calls:
        - _repay(msg.sender,provider,pesudAmount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#167)
                - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#198)
                - 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#212)
        State variables written after the call(s):
        - depositedAsset[provider] -= collateralAmount (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#170)
        LybraPeUSDVaultBase.depositedAsset (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#22) can be used in cross function reentrancies:
        - LybraPeUSDVaultBase._checkHealth(address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#229-232)
        - LybraPeUSDVaultBase._withdraw(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#216-224)
        - LybraPeUSDVaultBase.depositAssetToMint(uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#59-69)
        - LybraPeUSDVaultBase.depositedAsset (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#22)
        - LybraPeUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#123-149)
        - LybraPeUSDVaultBase.rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#160-173)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
LybraPeUSDVaultBase.liquidation(address,address,uint256).reward2keeper (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#141) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
LybraPeUSDVaultBase._mintPeUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#178-190) ignores return value by PeUSD.mint(_onBehalfOf,_mintAmount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#186)
LybraPeUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#197-214) ignores return value by PeUSD.burn(_provider,amount - totalFee) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#205)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
Reentrancy in LybraPeUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#197-214):
        External calls:
        - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#198)
        State variables written after the call(s):
        - _updateFee(_onBehalfOf) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#199)
                - feeStored[user] += _newFee(user) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#236)
        - feeStored[_onBehalfOf] = 0 (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#203)
        - feeStored[_onBehalfOf] = totalFee - amount (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#209)
        - _updateFee(_onBehalfOf) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#199)
                - feeUpdatedAt[user] = block.timestamp (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#237)
Reentrancy in LybraPeUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#197-214):
        External calls:
        - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#198)
        - 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)
        State variables written after the call(s):
        - poolTotalCirculation -= amount - totalFee (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#207)
Reentrancy in LybraPeUSDVaultBase.depositAssetToMint(uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#59-69):
        External calls:
        - collateralAsset.safeTransferFrom(msg.sender,address(this),assetAmount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#61)
        State variables written after the call(s):
        - depositedAsset[msg.sender] += assetAmount (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#63)
Reentrancy in LybraWBETHVault.depositEtherToMint(uint256) (contracts/lybra/pools/LybraWbETHVault.sol#21-33):
        External calls:
        - IWBETH(address(collateralAsset)).deposit{value: msg.value}(address(configurator)) (contracts/lybra/pools/LybraWbETHVault.sol#24)
        State variables written after the call(s):
        - depositedAsset[msg.sender] += balance - preBalance (contracts/lybra/pools/LybraWbETHVault.sol#26)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in LybraPeUSDVaultBase._mintPeUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#178-190):
        External calls:
        - configurator.refreshMintReward(_provider) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#182)
        - PeUSD.mint(_onBehalfOf,_mintAmount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#186)
        Event emitted after the call(s):
        - Mint(_provider,_onBehalfOf,_mintAmount,block.timestamp) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#189)
Reentrancy in LybraPeUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#197-214):
        External calls:
        - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#198)
        - 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#212)
        Event emitted after the call(s):
        - Burn(_provider,_onBehalfOf,amount,block.timestamp) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#213)
Reentrancy in LybraPeUSDVaultBase._withdraw(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#216-224):
        External calls:
        - collateralAsset.safeTransfer(_onBehalfOf,_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.depositAssetToMint(uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#59-69):
        External calls:
        - collateralAsset.safeTransferFrom(msg.sender,address(this),assetAmount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#61)
        - _mintPeUSD(msg.sender,msg.sender,mintAmount,assetPrice) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#66)
                - configurator.refreshMintReward(_provider) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#182)
                - PeUSD.mint(_onBehalfOf,_mintAmount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#186)
        Event emitted after the call(s):
        - DepositAsset(msg.sender,address(collateralAsset),assetAmount,block.timestamp) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#68)
        - Mint(_provider,_onBehalfOf,_mintAmount,block.timestamp) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#189)
                - _mintPeUSD(msg.sender,msg.sender,mintAmount,assetPrice) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#66)
Reentrancy in LybraWBETHVault.depositEtherToMint(uint256) (contracts/lybra/pools/LybraWbETHVault.sol#21-33):
        External calls:
        - IWBETH(address(collateralAsset)).deposit{value: msg.value}(address(configurator)) (contracts/lybra/pools/LybraWbETHVault.sol#24)
        - _mintPeUSD(msg.sender,msg.sender,mintAmount,getAssetPrice()) (contracts/lybra/pools/LybraWbETHVault.sol#29)
                - etherOracle.fetchPrice() (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#209)
                - configurator.refreshMintReward(_provider) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#182)
                - PeUSD.mint(_onBehalfOf,_mintAmount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#186)
        External calls sending eth:
        - IWBETH(address(collateralAsset)).deposit{value: msg.value}(address(configurator)) (contracts/lybra/pools/LybraWbETHVault.sol#24)
        Event emitted after the call(s):
        - DepositEther(msg.sender,address(collateralAsset),msg.value,balance - preBalance,block.timestamp) (contracts/lybra/pools/LybraWbETHVault.sol#32)
        - Mint(_provider,_onBehalfOf,_mintAmount,block.timestamp) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#189)
                - _mintPeUSD(msg.sender,msg.sender,mintAmount,getAssetPrice()) (contracts/lybra/pools/LybraWbETHVault.sol#29)
Reentrancy in LybraPeUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#123-149):
        External calls:
        - _repay(provider,onBehalfOf,peusdAmount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#132)
                - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#198)
                - 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#212)
        - collateralAsset.safeTransfer(msg.sender,reward2keeper) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#145)
        - collateralAsset.safeTransfer(provider,reducedAsset - reward2keeper) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#147)
        Event emitted after the call(s):
        - LiquidationRecord(provider,msg.sender,onBehalfOf,peusdAmount,reducedAsset,reward2keeper,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,peusdAmount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#167)
                - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#198)
                - 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#212)
        - collateralAsset.safeTransfer(msg.sender,collateralAmount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#171)
        Event emitted after the call(s):
        - RigidRedemption(msg.sender,provider,peusdAmount,collateralAmount,block.timestamp) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#172)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
LybraPeUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#123-149) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(onBehalfOfCollateralRatio < configurator.getBadCollateralRatio(address(this)),Borrowers collateral ratio should below badCollateralRatio) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#126)
        - onBehalfOfCollateralRatio > 1e20 && onBehalfOfCollateralRatio < 11e19 (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#134)
        - onBehalfOfCollateralRatio >= 11e19 (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#137)
        - msg.sender != provider && onBehalfOfCollateralRatio >= 1e20 + keeperRatio * 1e18 (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#143)
LybraPeUSDVaultBase.rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#160-173) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(borrowed[provider] >= peusdAmount,peusdAmount cannot surpass providers debt) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#163)
        - require(bool,string)(providerCollateralRatio >= 100 * 1e18,The provider's collateral ratio should be not less than 100%.) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#166)
LybraPeUSDVaultBase._mintPeUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#178-190) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(poolTotalCirculation + _mintAmount <= configurator.mintVaultMaxSupply(address(this)),ESL) (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:
        - amount > totalFee (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#202)
        - borrowed[_onBehalfOf] + totalFee >= _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:
        - getBorrowedOf(_provider) > 0 (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#220)
LybraPeUSDVaultBase._checkHealth(address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#229-232) uses timestamp for comparisons
        Dangerous comparisons:
        - ((depositedAsset[user] * price * 100) / getBorrowedOf(user)) < configurator.getSafeCollateralRatio(address(this)) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#230)
LybraPeUSDVaultBase._updateFee(address) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#234-239) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp > feeUpdatedAt[user] (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#235)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Function LybraWBETHVault.depositEtherToMint(uint256) (contracts/lybra/pools/LybraWbETHVault.sol#21-33) has a dubious typecast: IWBETH=address
Function LybraWBETHVault.getAssetPrice() (contracts/lybra/pools/LybraWbETHVault.sol#35-37) has a dubious typecast: IWBETH=address
Function LybraWBETHVault.getAsset2EtherExchangeRate() (contracts/lybra/pools/LybraWbETHVault.sol#38-40) has a dubious typecast: IWBETH=address
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/dubious_typecast.md
INFO:Detectors:
Pragma version^0.8.17 (contracts/lybra/interfaces/IEUSD.sol#2) allows old versions
Pragma version^0.8.17 (contracts/lybra/interfaces/IPeUSD.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/pools/LybraWbETHVault.sol#3) 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: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Parameter IEUSD.getSharesByMintedEUSD(uint256)._EUSDAmount (contracts/lybra/interfaces/IEUSD.sol#32) is not in mixedCase
Variable LybraPeUSDVaultBase.PeUSD (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#16) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Function LybraPeUSDVaultBase.depositAssetToMint(uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#59-69) contains magic number: 10000000000000000
Function LybraPeUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#123-149) contains magic numbers: 100, 1e20, 1e20, 11e19, 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#229-232) contains magic number: 100
Function LybraPeUSDVaultBase._newFee(address) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#241-243) contains magic number: 86
Function LybraWBETHVault.depositEtherToMint(uint256) (contracts/lybra/pools/LybraWbETHVault.sol#21-33) contains magic number: 10000000000000000
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/magic_number.md
INFO:Detectors:
In a function LybraWBETHVault.depositEtherToMint(uint256) (contracts/lybra/pools/LybraWbETHVault.sol#21-33) variable LybraPeUSDVaultBase.collateralAsset (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#17) is read multiple times
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/multiple_storage_read.md
```

LybraWstETHVault.sol

```
INFO:Detectors:
LybraPeUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#197-214) ignores return value by PeUSD.transferFrom(_provider,address(configurator),totalFee) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#204)
LybraPeUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#197-214) ignores return value by PeUSD.transferFrom(_provider,address(configurator),amount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#210)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
LybraPeUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#123-149) performs a multiplication on the result of a division:
        - onBehalfOfCollateralRatio = (depositedAsset[onBehalfOf] * assetPrice * 100) / getBorrowedOf(onBehalfOf) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#125)
        - reducedAsset = assetAmount * onBehalfOfCollateralRatio / 1e20 (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#135)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
Reentrancy in LybraPeUSDVaultBase._mintPeUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#178-190):
        External calls:
        - configurator.refreshMintReward(_provider) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#182)
        State variables written after the call(s):
        - borrowed[_provider] += _mintAmount (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#184)
        LybraPeUSDVaultBase.borrowed (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#23) can be used in cross function reentrancies:
        - LybraPeUSDVaultBase._mintPeUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#178-190)
        - LybraPeUSDVaultBase._newFee(address) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#241-243)
        - LybraPeUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#197-214)
        - LybraPeUSDVaultBase.getBorrowedOf(address) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#257-259)
        - LybraPeUSDVaultBase.rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#160-173)
Reentrancy in LybraPeUSDVaultBase._mintPeUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#178-190):
        External calls:
        - configurator.refreshMintReward(_provider) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#182)
        - PeUSD.mint(_onBehalfOf,_mintAmount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#186)
        State variables written after the call(s):
        - poolTotalCirculation += _mintAmount (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#187)
        LybraPeUSDVaultBase.poolTotalCirculation (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#19) can be used in cross function reentrancies:
        - LybraPeUSDVaultBase._mintPeUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#178-190)
        - LybraPeUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#197-214)
        - LybraPeUSDVaultBase.getPoolTotalCirculation() (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#261-263)
Reentrancy in LybraPeUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#197-214):
        External calls:
        - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#198)
        - 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)
```

```
        State variables written after the call(s):
        - borrowed[_onBehalfOf] -= amount - totalFee (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#206)
        LybraPeUSDVaultBase.borrowed (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#23) can be used in cross function reentrancies:
        - LybraPeUSDVaultBase._mintPeUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#178-190)
        - LybraPeUSDVaultBase._newFee(address) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#241-243)
        - LybraPeUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#197-214)
        - LybraPeUSDVaultBase.getBorrowedOf(address) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#257-259)
        - LybraPeUSDVaultBase.rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#160-173)
Reentrancy in LybraPeUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#123-149):
        External calls:
        - _repay(provider,onBehalfOf,peusdAmount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#132)
                - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#198)
                - 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#212)
        State variables written after the call(s):
        - depositedAsset[onBehalfOf] -= reducedAsset (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#140)
        LybraPeUSDVaultBase.depositedAsset (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#22) can be used in cross function reentrancies:
        - LybraPeUSDVaultBase._checkHealth(address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#229-232)
        - LybraPeUSDVaultBase._withdraw(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#216-224)
        - LybraPeUSDVaultBase.depositAssetToMint(uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#59-69)
        - LybraPeUSDVaultBase.depositedAsset (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#22)
        - LybraPeUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#123-149)
        - LybraPeUSDVaultBase.rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#160-173)
Reentrancy in LybraPeUSDVaultBase.rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#160-173):
        External calls:
        - _repay(msg.sender,provider,peusdAmount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#167)
                - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#198)
                - 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#212)
        State variables written after the call(s):
        - depositedAsset[provider] -= collateralAmount (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#170)
        LybraPeUSDVaultBase.depositedAsset (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#22) can be used in cross function reentrancies:
        - LybraPeUSDVaultBase._checkHealth(address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#229-232)
        - LybraPeUSDVaultBase._withdraw(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#216-224)
        - LybraPeUSDVaultBase.depositAssetToMint(uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#59-69)
        - LybraPeUSDVaultBase.depositedAsset (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#22)
        - LybraPeUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#123-149)
        - LybraPeUSDVaultBase.rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#160-173)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
LybraPeUSDVaultBase.liquidation(address,address,uint256).reward2keeper (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#141) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
LybraWstETHVault.depositEtherToMint(uint256) (contracts/lybra/pools/LybraWstETHVault.sol#29-40) ignores return value by lido.approve(address(collateralAsset),msg.value) (contracts/lybra/pools/LybraWstETHVault.sol#33)
LybraPeUSDVaultBase._mintPeUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#178-190) ignores return value by PeUSD.mint(_onBehalfOf,_mintAmount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#
186)
LybraPeUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#197-214) ignores return value by PeUSD.burn(_provider,amount - totalFee) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#205)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
Reentrancy in LybraPeUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#197-214):
        External calls:
        - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#198)
        State variables written after the call(s):
        - _updateFee(_onBehalfOf) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#199)
                - feeStored[user] += _newFee(user) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#236)
        - feeStored[_onBehalfOf] = 0 (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#203)
        - feeStored[_onBehalfOf] = totalFee - amount (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#209)
        - _updateFee(_onBehalfOf) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#199)
                - feeUpdatedAt[user] = block.timestamp (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#237)
Reentrancy in LybraPeUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#197-214):
        External calls:
        - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#198)
        - 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)
        State variables written after the call(s):
        - poolTotalCirculation -= amount - totalFee (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#207)
Reentrancy in LybraPeUSDVaultBase.depositAssetToMint(uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#59-69):
        External calls:
        - collateralAsset.safeTransferFrom(msg.sender,address(this),assetAmount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#61)
        State variables written after the call(s):
        - depositedAsset[msg.sender] += assetAmount (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#63)
Reentrancy in LybraWstETHVault.depositEtherToMint(uint256) (contracts/lybra/pools/LybraWstETHVault.sol#29-40):
        External calls:
        - sharesAmount = lido.submit{value: msg.value}(address(configurator)) (contracts/lybra/pools/LybraWstETHVault.sol#31)
        - lido.approve(address(collateralAsset),msg.value) (contracts/lybra/pools/LybraWstETHVault.sol#33)
        - wstETHAmount = IWstETH(address(collateralAsset)).wrap(msg.value) (contracts/lybra/pools/LybraWstETHVault.sol#34)
        External calls sending eth:
        - sharesAmount = lido.submit{value: msg.value}(address(configurator)) (contracts/lybra/pools/LybraWstETHVault.sol#31)
        State variables written after the call(s):
        - depositedAsset[msg.sender] += wstETHAmount (contracts/lybra/pools/LybraWstETHVault.sol#35)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in LybraPeUSDVaultBase._mintPeUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#178-190):
        External calls:
        - configurator.refreshMintReward(_provider) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#182)
        - PeUSD.mint(_onBehalfOf,_mintAmount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#186)
        Event emitted after the call(s):
        - Mint(_provider,_onBehalfOf,_mintAmount,block.timestamp) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#189)
Reentrancy in LybraPeUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#197-214):
        External calls:
        - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#198)
        - 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#212)
        Event emitted after the call(s):
        - Burn(_provider,_onBehalfOf,amount,block.timestamp) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#213)
Reentrancy in LybraPeUSDVaultBase._withdraw(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#216-224):
        External calls:
        - collateralAsset.safeTransfer(_onBehalfOf,_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.depositAssetToMint(uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#59-69):
        External calls:
        - collateralAsset.safeTransferFrom(msg.sender,address(this),assetAmount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#61)
        - _mintPeUSD(msg.sender,msg.sender,mintAmount,assetPrice) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#66)
                - configurator.refreshMintReward(_provider) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#182)
                - PeUSD.mint(_onBehalfOf,_mintAmount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#186)
        Event emitted after the call(s):
        - DepositAsset(msg.sender,address(collateralAsset),assetAmount,block.timestamp) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#68)
        - Mint(_provider,_onBehalfOf,_mintAmount,block.timestamp) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#189)
                - _mintPeUSD(msg.sender,msg.sender,mintAmount,assetPrice) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#66)
Reentrancy in LybraWstETHVault.depositEtherToMint(uint256) (contracts/lybra/pools/LybraWstETHVault.sol#29-40):
        External calls:
        - sharesAmount = lido.submit{value: msg.value}(address(configurator)) (contracts/lybra/pools/LybraWstETHVault.sol#31)
        - lido.approve(address(collateralAsset),msg.value) (contracts/lybra/pools/LybraWstETHVault.sol#33)
        - wstETHAmount = IWstETH(address(collateralAsset)).wrap(msg.value) (contracts/lybra/pools/LybraWstETHVault.sol#34)
        - _mintPeUSD(msg.sender,msg.sender,mintAmount,getAssetPrice()) (contracts/lybra/pools/LybraWstETHVault.sol#37)
                - etherOracle.fetchPrice() (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#249)
                - configurator.refreshMintReward(_provider) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#182)
                - PeUSD.mint(_onBehalfOf,_mintAmount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#186)
        External calls sending eth:
        - sharesAmount = lido.submit{value: msg.value}(address(configurator)) (contracts/lybra/pools/LybraWstETHVault.sol#31)
        Event emitted after the call(s):
        - DepositEther(msg.sender,address(collateralAsset),msg.value,wstETHAmount,block.timestamp) (contracts/lybra/pools/LybraWstETHVault.sol#39)
        - Mint(_provider,_onBehalfOf,_mintAmount,block.timestamp) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#189)
                - _mintPeUSD(msg.sender,msg.sender,mintAmount,getAssetPrice()) (contracts/lybra/pools/LybraWstETHVault.sol#37)
Reentrancy in LybraPeUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#123-149):
        External calls:
        - _repay(provider,onBehalfOf,peusdAmount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#132)
                - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#198)
                - 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#212)
        - collateralAsset.safeTransfer(msg.sender,reward2keeper) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#145)
        - collateralAsset.safeTransfer(provider,reducedAsset - reward2keeper) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#147)
        Event emitted after the call(s):
        - LiquidationRecord(provider,msg.sender,onBehalfOf,peusdAmount,reducedAsset,reward2keeper,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,peusdAmount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#167)
                - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#198)
                - 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#212)
        - collateralAsset.safeTransfer(msg.sender,collateralAmount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#171)
        Event emitted after the call(s):
        - RigidRedemption(msg.sender,provider,peusdAmount,collateralAmount,block.timestamp) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#172)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```

```
INFO:Detectors:
LybraPeUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#123-149) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(onBehalfOfCollateralRatio < configurator.getBadCollateralRatio(address(this)),Borrowers collateral ratio should below badCollateralRatio) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#126)
        - onBehalfOfCollateralRatio > 1e20 && onBehalfOfCollateralRatio < 11e19 (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#134)
        - onBehalfOfCollateralRatio >= 11e19 (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#137)
        - msg.sender != provider && onBehalfOfCollateralRatio >= 1e20 + keeperRatio * 1e18 (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#143)
LybraPeUSDVaultBase.rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#160-173) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(borrowed[provider] >= peusdAmount,peusdAmount cannot surpass providers debt) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#163)
        - require(bool,string)(providerCollateralRatio >= 100 * 1e18,The provider's collateral ratio should be not less than 100% ) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#166)
LybraPeUSDVaultBase._mintPeUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#178-190) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(poolTotalCirculation + _mintAmount <= configurator.mintVaultMaxSupply(address(this)),ESL) (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:
        - amount > totalFee (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#202)
        - borrowed[_onBehalfOf] + totalFee >= _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:
        - getBorrowedOf(_provider) > 0 (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#220)
LybraPeUSDVaultBase._checkHealth(address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#229-232) uses timestamp for comparisons
        Dangerous comparisons:
        - ((depositedAsset[user] * price * 100) / getBorrowedOf(user)) < configurator.getSafeCollateralRatio(address(this)) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#230)
LybraPeUSDVaultBase._updateFee(address) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#234-239) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp > feeUpdatedAt[user] (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#235)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Function LybraWstETHVault.depositEtherToMint(uint256) (contracts/lybra/pools/LybraWstETHVault.sol#29-40) has a dubious typecast: IWstETH<=address
Function LybraWstETHVault.getAssetPrice() (contracts/lybra/pools/LybraWstETHVault.sol#42-44) has a dubious typecast: IWstETH<=address
Function LybraWstETHVault.getAsset2EtherExchangeRate() (contracts/lybra/pools/LybraWstETHVault.sol#45-47) has a dubious typecast: IWstETH<=address
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/dubious_typecast.md
INFO:Detectors:
Pragma version^0.8.17 (contracts/lybra/interfaces/IEUSD.sol#2) allows old versions
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/LybraWstETHVault.sol#3) 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: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Parameter IEUSD.getSharesByMintedEUSD(uint256)._EUSDAmount (contracts/lybra/interfaces/IEUSD.sol#32) is not in mixedCase
Variable LybraPeUSDVaultBase.PeUSD (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#16) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Function LybraPeUSDVaultBase.depositAssetToMint(uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#59-69) contains magic number: 10000000000000000
Function LybraPeUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#123-149) contains magic numbers: 100, 1e20, 1e20, 11e19, 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#229-232) contains magic number: 100
Function LybraPeUSDVaultBase._newFee(address) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#241-243) contains magic number: 86
Function LybraWstETHVault.depositEtherToMint(uint256) (contracts/lybra/pools/LybraWstETHVault.sol#29-40) contains magic number: 10000000000000000
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/magic_number.md
INFO:Detectors:
In a function LybraWstETHVault.depositEtherToMint(uint256) (contracts/lybra/pools/LybraWstETHVault.sol#29-40) variable LybraPeUSDVaultBase.collateralAsset (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#17) is read multiple times
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/multiple_storage_read.md
```

LybraEUSDVaultBase.sol

```
INFO:Detectors:
LybraEUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#148-178) performs a multiplication on the result of a division:
        - onBehalfOfCollateralRatio = (depositedAsset[onBehalfOf] * assetPrice * 100) / borrowed[onBehalfOf] (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#150)
        - reducedAsset = assetAmount * onBehalfOfCollateralRatio / 1e20 (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#161)
LybraEUSDVaultBase.superLiquidation(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#189-213) performs a multiplication on the result of a division:
        - eusdAmount = (assetAmount * assetPrice) / 1e18 (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#195)
        - eusdAmount = (eusdAmount * 1e20) / onBehalfOfCollateralRatio (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#197)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
Reentrancy in LybraEUSDVaultBase._mintEUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#263-273):
        External calls:
        - configurator.refreshMintReward(_provider) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#265)
        - EUSD.mint(_onBehalfOf,_mintAmount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#268)
        State variables written after the call(s):
        - poolTotalCirculation += _mintAmount (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#270)
        LybraEUSDVaultBase.poolTotalCirculation (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#24) can be used in cross function reentrancies:
        - LybraEUSDVaultBase._mintEUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#263-273)
        - LybraEUSDVaultBase._newFee() (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#304-306)
        - LybraEUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#280-290)
        - LybraEUSDVaultBase.getPoolTotalCirculation() (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#319-321)
        - LybraEUSDVaultBase.superLiquidation(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#189-213)
Reentrancy in LybraEUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#280-290):
        External calls:
        - EUSD.burn(_provider,amount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#283)
        - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#284)
        State variables written after the call(s):
        - borrowed[_onBehalfOf] -= amount (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#286)
        LybraEUSDVaultBase.borrowed (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#27) can be used in cross function reentrancies:
        - LybraEUSDVaultBase._checkHealth(address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#295-297)
        - LybraEUSDVaultBase._mintEUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#263-273)
        - LybraEUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#280-290)
        - LybraEUSDVaultBase.getBorrowedOf(address) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#315-317)
        - LybraEUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#148-178)
        - LybraEUSDVaultBase.rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#234-248)
        - LybraEUSDVaultBase.superLiquidation(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#189-213)
        - LybraEUSDVaultBase.withdraw(address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#93-107)
Reentrancy in LybraEUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#148-178):
        External calls:
        - _repay(provider,onBehalfOf,eusdAmount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#157)
                - EUSD.burn(_provider,amount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#283)
                - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#284)
        State variables written after the call(s):
        - depositedAsset[onBehalfOf] -= reducedAsset (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#167)
        LybraEUSDVaultBase.depositedAsset (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#26) can be used in cross function reentrancies:
        - LybraEUSDVaultBase._checkHealth(address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#295-297)
        - LybraEUSDVaultBase.depositAssetToMint(uint256,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#70-81)
        - LybraEUSDVaultBase.depositedAsset (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#26)
        - LybraEUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#148-178)
        - LybraEUSDVaultBase.rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#234-248)
        - LybraEUSDVaultBase.superLiquidation(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#189-213)
        - LybraEUSDVaultBase.withdraw(address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#93-107)
Reentrancy in LybraEUSDVaultBase.rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#234-248):
        External calls:
        - _repay(msg.sender,provider,eusdAmount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#241)
                - EUSD.burn(_provider,amount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#283)
                - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#284)
        State variables written after the call(s):
        - depositedAsset[provider] -= collateralAmount (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#244)
        LybraEUSDVaultBase.depositedAsset (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#26) can be used in cross function reentrancies:
        - LybraEUSDVaultBase._checkHealth(address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#295-297)
        - LybraEUSDVaultBase.depositAssetToMint(uint256,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#70-81)
        - LybraEUSDVaultBase.depositedAsset (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#26)
        - LybraEUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#148-178)
        - LybraEUSDVaultBase.rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#234-248)
        - LybraEUSDVaultBase.superLiquidation(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#189-213)
        - LybraEUSDVaultBase.withdraw(address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#93-107)
Reentrancy in LybraEUSDVaultBase.superLiquidation(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#189-213):
        External calls:
        - _repay(provider,onBehalfOf,eusdAmount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#201)
                - EUSD.burn(_provider,amount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#283)
                - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#284)
        State variables written after the call(s):
        - depositedAsset[onBehalfOf] -= assetAmount (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#204)
        LybraEUSDVaultBase.depositedAsset (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#26) can be used in cross function reentrancies:
        - LybraEUSDVaultBase._checkHealth(address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#295-297)
        - LybraEUSDVaultBase.depositAssetToMint(uint256,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#70-81)
        - LybraEUSDVaultBase.depositedAsset (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#26)
        - LybraEUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#148-178)
        - LybraEUSDVaultBase.rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#234-248)
        - LybraEUSDVaultBase.superLiquidation(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#189-213)
        - LybraEUSDVaultBase.withdraw(address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#93-107)
        - totalDepositedAsset -= assetAmount (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#203)
        LybraEUSDVaultBase.totalDepositedAsset (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#22) can be used in cross function reentrancies:
        - LybraEUSDVaultBase.depositAssetToMint(uint256,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#70-81)
        - LybraEUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#148-178)
        - LybraEUSDVaultBase.rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#234-248)
        - LybraEUSDVaultBase.superLiquidation(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#189-213)
        - LybraEUSDVaultBase.totalDepositedAsset (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#22)
        - LybraEUSDVaultBase.withdraw(address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#93-107)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
LybraEUSDVaultBase.liquidation(address,address,uint256).reward2keeper (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#169) is a local variable never initialized
LybraEUSDVaultBase.superLiquidation(address,address,uint256).reward2keeper (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#205) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
```

```
INFO:Detectors:
LybraEUSDVaultBase._mintEUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#263-273) ignores return value by EUSD.mint(_onBehalfOf,_mintAmount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#268)
LybraEUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#280-290) ignores return value by EUSD.burn(_provider,amount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#283)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
Reentrancy in LybraEUSDVaultBase._mintEUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#263-273):
        External calls:
        - configurator.refreshMintReward(_provider) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#265)
        State variables written after the call(s):
        - borrowed[_provider] += _mintAmount (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#266)
Reentrancy in LybraEUSDVaultBase._mintEUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#263-273):
        External calls:
        - configurator.refreshMintReward(_provider) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#265)
        - EUSD.mint(_onBehalfOf,_mintAmount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#268)
        State variables written after the call(s):
        - _saveReport() (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#269)
                - feeStored += _newFee() (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#300)
        - _saveReport() (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#269)
                - lastReportTime = block.timestamp (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#301)
Reentrancy in LybraEUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#280-290):
        External calls:
        - EUSD.burn(_provider,amount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#283)
        - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#284)
        State variables written after the call(s):
        - _saveReport() (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#287)
                - feeStored += _newFee() (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#300)
        - _saveReport() (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#287)
                - lastReportTime = block.timestamp (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#301)
        - poolTotalCirculation -= amount (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#288)
Reentrancy in LybraEUSDVaultBase.depositAssetToMint(uint256,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#70-81):
        External calls:
        - collateralAsset.safeTransferFrom(msg.sender,address(this),assetAmount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#72)
        State variables written after the call(s):
        - depositedAsset[msg.sender] += assetAmount (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#74)
        - depositedTime[msg.sender] = block.timestamp (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#75)
        - totalDepositedAsset += assetAmount (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#73)
Reentrancy in LybraEUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#148-178):
        External calls:
        - _repay(provider,onBehalfOf,eusdAmount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#157)
                - EUSD.burn(_provider,amount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#283)
                - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#284)
        State variables written after the call(s):
        - totalDepositedAsset -= reducedAsset (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#166)
Reentrancy in LybraEUSDVaultBase.rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#234-248):
        External calls:
        - _repay(msg.sender,provider,eusdAmount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#241)
                - EUSD.burn(_provider,amount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#283)
                - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#284)
        State variables written after the call(s):
        - totalDepositedAsset -= collateralAmount (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#245)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in LybraEUSDVaultBase._mintEUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#263-273):
        External calls:
        - configurator.refreshMintReward(_provider) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#265)
        - EUSD.mint(_onBehalfOf,_mintAmount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#268)
        Event emitted after the call(s):
        - Mint(msg.sender,_onBehalfOf,_mintAmount,block.timestamp) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#272)
Reentrancy in LybraEUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#280-290):
        External calls:
        - EUSD.burn(_provider,amount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#283)
        - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#284)
        Event emitted after the call(s):
        - Burn(_provider,_onBehalfOf,amount,block.timestamp) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#289)
Reentrancy in LybraEUSDVaultBase.depositAssetToMint(uint256,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#70-81):
        External calls:
        - collateralAsset.safeTransferFrom(msg.sender,address(this),assetAmount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#72)
        - _mintEUSD(msg.sender,msg.sender,mintAmount,getAssetPrice()) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#78)
                - configurator.refreshMintReward(_provider) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#265)
                - EUSD.mint(_onBehalfOf,_mintAmount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#268)
        Event emitted after the call(s):
        - DepositAsset(msg.sender,address(collateralAsset),assetAmount,block.timestamp) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#80)
        - Mint(msg.sender,_onBehalfOf,_mintAmount,block.timestamp) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#272)
                - _mintEUSD(msg.sender,msg.sender,mintAmount,getAssetPrice()) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#78)
Reentrancy in LybraEUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#148-178):
        External calls:
        - _repay(provider,onBehalfOf,eusdAmount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#157)
                - EUSD.burn(_provider,amount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#283)
                - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#284)
        - collateralAsset.safeTransfer(msg.sender,reward2keeper) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#173)
        - collateralAsset.safeTransfer(provider,reducedAsset - reward2keeper) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#175)
        Event emitted after the call(s):
        - LiquidationRecord(provider,msg.sender,onBehalfOf,eusdAmount,reducedAsset,reward2keeper,false,block.timestamp) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#177)
Reentrancy in LybraEUSDVaultBase.rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#234-248):
        External calls:
        - _repay(msg.sender,provider,eusdAmount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#241)
                - EUSD.burn(_provider,amount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#283)
                - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#284)
        - collateralAsset.safeTransfer(msg.sender,checkWithdrawal(msg.sender,collateralAmount)) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#246)
        Event emitted after the call(s):
        - RigidRedemption(msg.sender,provider,eusdAmount,collateralAmount,block.timestamp) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#247)
Reentrancy in LybraEUSDVaultBase.superLiquidation(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#189-213):
        External calls:
        - _repay(provider,onBehalfOf,eusdAmount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#201)
                - EUSD.burn(_provider,amount) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#283)
                - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#284)
        - collateralAsset.safeTransfer(msg.sender,reward2keeper) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#208)
        - collateralAsset.safeTransfer(provider,assetAmount - reward2keeper) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#210)
        Event emitted after the call(s):
        - LiquidationRecord(provider,msg.sender,onBehalfOf,eusdAmount,assetAmount,reward2keeper,true,block.timestamp) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#212)
Reentrancy in LybraEUSDVaultBase.withdraw(address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#93-107):
        External calls:
        - collateralAsset.safeTransfer(onBehalfOf,withdrawal) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#102)
        Event emitted after the call(s):
        - WithdrawAsset(msg.sender,address(collateralAsset),onBehalfOf,withdrawal,block.timestamp) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#106)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
LybraEUSDVaultBase.checkWithdrawal(address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#109-111) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp - 259200 >= depositedTime[user] (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#110)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
LybraEUSDVaultBase._etherPrice() (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#311-313) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.17 (contracts/lybra/interfaces/IEUSD.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/pools/base/LybraEUSDVaultBase.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:
Parameter IEUSD.getSharesByMintedEUSD(uint256)._EUSDAmount (contracts/lybra/interfaces/IEUSD.sol#32) is not in mixedCase
Variable LybraEUSDVaultBase.EUSD (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#16) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
LybraEUSDVaultBase (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#14-333) does not implement functions:
        - LybraEUSDVaultBase.depositEtherToMint(uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#60)
        - LybraEUSDVaultBase.excessIncomeDistribution(uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#223)
        - LybraEUSDVaultBase.getAsset2EtherExchangeRate() (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#332)
        - LybraEUSDVaultBase.getAssetPrice() (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#331)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions
INFO:Detectors:
Function LybraEUSDVaultBase.depositAssetToMint(uint256,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#70-81) contains magic number: 1000000000000000000
Function LybraEUSDVaultBase.checkWithdrawal(address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#109-111) contains magic numbers: 259200, 999
Function LybraEUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#148-178) contains magic numbers: 100, 1e20, 1e20, 11e19, 11, 1e20, 100
Function LybraEUSDVaultBase.superLiquidation(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#189-213) contains magic numbers: 100, 100, 125, 1e20, 1e20, 1e20
Function LybraEUSDVaultBase.rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#234-248) contains magic numbers: 100, 100
Function LybraEUSDVaultBase._checkHealth(address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#295-297) contains magic number: 100
Function LybraEUSDVaultBase._newFee() (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#304-306) contains magic number: 86
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/magic_number.md
INFO:Detectors:
In a function LybraEUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#148-178) variable LybraEUSDVaultBase.depositedAsset (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#26) is read m
ultiple times
In a function LybraEUSDVaultBase.superLiquidation(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#189-213) variable LybraEUSDVaultBase.depositedAsset (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#26) is r
ead multiple times
In a function LybraEUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#280-290) variable LybraEUSDVaultBase.borrowed (contracts/lybra/pools/base/LybraEUSDVaultBase.sol#27) is read multiple tim
es
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/multiple_storage_read.md
```

LybraPeUSDVaultBase.sol

```
INFO:Detectors:
LybraPeUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#197-214) ignores return value by PeUSD.transferFrom(_provider,address(configurator),totalFee) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#204)
LybraPeUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#197-214) ignores return value by PeUSD.transferFrom(_provider,address(configurator),amount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#210)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
LybraPeUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#123-149) performs a multiplication on the result of a division:
        - onBehalfOfCollateralRatio = (depositedAsset[onBehalfOf] * assetPrice * 100) / getBorrowedOf(onBehalfOf) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#125)
        - reducedAsset = assetAmount * onBehalfOfCollateralRatio / 1e20 (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#135)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
Reentrancy in LybraPeUSDVaultBase._mintPeUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#178-190):
        External calls:
        - configurator.refreshMintReward(_provider) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#182)
        State variables written after the call(s):
        - borrowed[_provider] += _mintAmount (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#184)
        LybraPeUSDVaultBase.borrowed (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#23) can be used in cross function reentrancies:
        - LybraPeUSDVaultBase._mintPeUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#178-190)
        - LybraPeUSDVaultBase._newFee(address) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#241-243)
        - LybraPeUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#197-214)
        - LybraPeUSDVaultBase.getBorrowedOf(address) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#257-259)
        - LybraPeUSDVaultBase.rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#160-173)
Reentrancy in LybraPeUSDVaultBase._mintPeUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#178-190):
        External calls:
        - configurator.refreshMintReward(_provider) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#182)
        - PeUSD.mint(_onBehalfOf,_mintAmount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#186)
        State variables written after the call(s):
        - poolTotalCirculation += _mintAmount (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#187)
        LybraPeUSDVaultBase.poolTotalCirculation (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#19) can be used in cross function reentrancies:
        - LybraPeUSDVaultBase._mintPeUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#178-190)
        - LybraPeUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#197-214)
        - LybraPeUSDVaultBase.getPoolTotalCirculation() (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#261-263)
Reentrancy in LybraPeUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#197-214):
        External calls:
        - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#198)
        - 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)
        State variables written after the call(s):
        - borrowed[_onBehalfOf] -= amount - totalFee (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#206)
        LybraPeUSDVaultBase.borrowed (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#23) can be used in cross function reentrancies:
        - LybraPeUSDVaultBase._mintPeUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#178-190)
        - LybraPeUSDVaultBase._newFee(address) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#241-243)
        - LybraPeUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#197-214)
        - LybraPeUSDVaultBase.getBorrowedOf(address) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#257-259)
        - LybraPeUSDVaultBase.rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#160-173)
Reentrancy in LybraPeUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#123-149):
        External calls:
        - _repay(provider,onBehalfOf,pesudsAmount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#132)
                - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#198)
                - 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#212)
        State variables written after the call(s):
        - depositedAsset[onBehalfOf] -= reducedAsset (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#140)
        LybraPeUSDVaultBase.depositedAsset (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#22) can be used in cross function reentrancies:
        - LybraPeUSDVaultBase._checkHealth(address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#229-232)
        - LybraPeUSDVaultBase._withdraw(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#216-224)
        - LybraPeUSDVaultBase.depositAssetToMint(uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#59-69)
        - LybraPeUSDVaultBase.depositedAsset (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#22)
        - LybraPeUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#123-149)
        - LybraPeUSDVaultBase.rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#160-173)
Reentrancy in LybraPeUSDVaultBase.rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#160-173):
        External calls:
        - _repay(msg.sender,provider,pesudsAmount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#167)
                - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#198)
                - 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#212)
        State variables written after the call(s):
        - depositedAsset[provider] -= collateralAmount (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#170)
        LybraPeUSDVaultBase.depositedAsset (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#22) can be used in cross function reentrancies:
        - LybraPeUSDVaultBase._checkHealth(address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#229-232)
        - LybraPeUSDVaultBase._withdraw(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#216-224)
        - LybraPeUSDVaultBase.depositAssetToMint(uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#59-69)
        - LybraPeUSDVaultBase.depositedAsset (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#22)
        - LybraPeUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#123-149)
        - LybraPeUSDVaultBase.rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#160-173)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
LybraPeUSDVaultBase.liquidation(address,address,uint256).reward2keeper (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#141) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
LybraPeUSDVaultBase._mintPeUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#178-190) ignores return value by PeUSD.mint(_onBehalfOf,_mintAmount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#186)
LybraPeUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#197-214) ignores return value by PeUSD.burn(_provider,amount - totalFee) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#205)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
Reentrancy in LybraPeUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#197-214):
        External calls:
        - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#198)
        State variables written after the call(s):
        - _updateFee(_onBehalfOf) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#199)
                - feeStored[user] += _newFee(user) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#236)
        - feeStored[_onBehalfOf] = 0 (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#203)
        - feeStored[_onBehalfOf] = totalFee - amount (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#209)
        - _updateFee(_onBehalfOf) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#199)
                - feeUpdatedAt[user] = block.timestamp (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#237)
Reentrancy in LybraPeUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#197-214):
        External calls:
        - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#198)
        - 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)
        State variables written after the call(s):
        - poolTotalCirculation -= amount - totalFee (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#207)
Reentrancy in LybraPeUSDVaultBase.depositAssetToMint(uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#59-69):
        External calls:
        - collateralAsset.safeTransferFrom(msg.sender,address(this),assetAmount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#61)
        State variables written after the call(s):
        - depositedAsset[msg.sender] += assetAmount (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#63)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in LybraPeUSDVaultBase._mintPeUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#178-190):
        External calls:
        - configurator.refreshMintReward(_provider) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#182)
        - PeUSD.mint(_onBehalfOf,_mintAmount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#186)
        Event emitted after the call(s):
        - Mint(_provider,_onBehalfOf,_mintAmount,block.timestamp) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#189)
Reentrancy in LybraPeUSDVaultBase._repay(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#197-214):
        External calls:
        - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#198)
        - 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#212)
        Event emitted after the call(s):
        - Burn(_provider,_onBehalfOf,amount,block.timestamp) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#213)
Reentrancy in LybraPeUSDVaultBase._withdraw(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#216-224):
        External calls:
        - collateralAsset.safeTransfer(_onBehalfOf,_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.depositAssetToMint(uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#59-69):
        External calls:
        - collateralAsset.safeTransferFrom(msg.sender,address(this),assetAmount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#61)
        - _mintPeUSD(msg.sender,msg.sender,mintAmount,assetPrice) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#66)
                - configurator.refreshMintReward(_provider) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#182)
                - PeUSD.mint(_onBehalfOf,_mintAmount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#186)
        Event emitted after the call(s):
        - DepositAsset(msg.sender,address(collateralAsset),assetAmount,block.timestamp) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#68)
        - Mint(_provider,_onBehalfOf,_mintAmount,block.timestamp) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#189)
                - _mintPeUSD(msg.sender,msg.sender,mintAmount,assetPrice) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#66)
Reentrancy in LybraPeUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#123-149):
        External calls:
        - _repay(provider,onBehalfOf,pesudsAmount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#132)
                - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#198)
                - 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#212)
        - collateralAsset.safeTransfer(msg.sender,reward2keeper) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#145)
        - collateralAsset.safeTransfer(provider,reducedAsset - reward2keeper) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#147)
        Event emitted after the call(s):
        - LiquidationRecord(provider,msg.sender,onBehalfOf,pesudAmount,reducedAsset,reward2keeper,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,pesudAmount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#167)
                - configurator.refreshMintReward(_onBehalfOf) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#198)
                - 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#212)
        - collateralAsset.safeTransfer(msg.sender,collateralAmount) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#171)
        Event emitted after the call(s):
        - RigidRedemption(msg.sender,provider,pesudAmount,collateralAmount,block.timestamp) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#172)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
LybraPeUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#123-149) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(onBehalfOfCollateralRatio < configurator.getBadCollateralRatio(address(this)),Borrowers collateral ratio should below badCollateralRatio) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#126)
        - onBehalfOfCollateralRatio > 1e20 && onBehalfOfCollateralRatio < 11e19 (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#134)
        - onBehalfOfCollateralRatio >= 11e19 (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#137)
        - msg.sender != provider && onBehalfOfCollateralRatio >= 1e20 + keeperRatio * 1e18 (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#143)
LybraPeUSDVaultBase.rigidRedemption(address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#160-173) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(borrowed[provider] >= pesudAmount,pesudAmount cannot surpass providers debt) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#163)
        - require(bool,string)(providerCollateralRatio >= 100 * 1e18,The provider's collateral ratio should be not less than 100%.) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#166)
LybraPeUSDVaultBase._mintPeUSD(address,address,uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#178-190) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(poolTotalCirculation + _mintAmount <= configurator.mintVaultMaxSupply(address(this)),ESL) (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:
        - amount > totalFee (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#202)
        - borrowed[_onBehalfOf] < totalFee >= _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:
        - getBorrowedOf(_provider) > 0 (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#220)
LybraPeUSDVaultBase._checkHealth(address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#229-232) uses timestamp for comparisons
        Dangerous comparisons:
        - ((depositedAsset[user] * price * 100) / getBorrowedOf(user)) < configurator.getSafeCollateralRatio(address(this)) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#230)
LybraPeUSDVaultBase._updateFee(address) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#234-239) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp > feeUpdatedAt[user] (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#235)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
LybraPeUSDVaultBase._etherPrice() (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#248-250) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
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#3) 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: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Variable LybraPeUSDVaultBase.PeUSD (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#16) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
LybraPeUSDVaultBase (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#14-275) does not implement functions:
        - LybraPeUSDVaultBase.depositEtherToMint(uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#49)
        - LybraPeUSDVaultBase.getAsset2EtherExchangeRate() (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#274)
        - LybraPeUSDVaultBase.getAssetPrice() (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#273)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions
INFO:Detectors:
Function LybraPeUSDVaultBase.depositAssetToMint(uint256,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#59-69) contains magic number: 100000000000000000
Function LybraPeUSDVaultBase.liquidation(address,address,uint256) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#123-149) contains magic numbers: 100, 1e20, 1e20, 11e19, 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#229-232) contains magic number: 100
Function LybraPeUSDVaultBase._newFee(address) (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#241-243) contains magic number: 86
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/magic_number.md
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.PeUSD (contracts/lybra/pools/base/LybraPeUSDVaultBase.sol#16) 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: https://github.com/pessimistic-io/slitherin/blob/master/docs/multiple_storage_read.md
```

EUSD.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/EUSD.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:
Parameter EUSD.balanceOf(address)._account (contracts/lybra/token/EUSD.sol#132) is not in mixedCase
Parameter EUSD.transfer(address,uint256)._recipient (contracts/lybra/token/EUSD.sol#150) is not in mixedCase
Parameter EUSD.transfer(address,uint256)._amount (contracts/lybra/token/EUSD.sol#150) is not in mixedCase
Parameter EUSD.allowance(address,address)._owner (contracts/lybra/token/EUSD.sol#162) is not in mixedCase
Parameter EUSD.allowance(address,address)._spender (contracts/lybra/token/EUSD.sol#162) is not in mixedCase
Parameter EUSD.approve(address,uint256)._spender (contracts/lybra/token/EUSD.sol#196) is not in mixedCase
Parameter EUSD.approve(address,uint256)._amount (contracts/lybra/token/EUSD.sol#196) is not in mixedCase
Parameter EUSD.increaseAllowance(address,uint256)._spender (contracts/lybra/token/EUSD.sol#240) is not in mixedCase
Parameter EUSD.increaseAllowance(address,uint256)._addedValue (contracts/lybra/token/EUSD.sol#240) is not in mixedCase
Parameter EUSD.sharesOf(address)._account (contracts/lybra/token/EUSD.sol#281) is not in mixedCase
Parameter EUSD.getSharesByMintedEUSD(uint256)._EUSDAmount (contracts/lybra/token/EUSD.sol#288) is not in mixedCase
Parameter EUSD.getMintedEUSDByShares(uint256)._sharesAmount (contracts/lybra/token/EUSD.sol#299) is not in mixedCase
Parameter EUSD.transferShares(address,uint256)._recipient (contracts/lybra/token/EUSD.sol#320) is not in mixedCase
Parameter EUSD.transferShares(address,uint256)._sharesAmount (contracts/lybra/token/EUSD.sol#320) is not in mixedCase
Parameter EUSD.mint(address,uint256)._recipient (contracts/lybra/token/EUSD.sol#395) is not in mixedCase
Parameter EUSD.mint(address,uint256)._mintAmount (contracts/lybra/token/EUSD.sol#395) is not in mixedCase
Parameter EUSD.burn(address,uint256)._account (contracts/lybra/token/EUSD.sol#425) is not in mixedCase
Parameter EUSD.burn(address,uint256)._burnAmount (contracts/lybra/token/EUSD.sol#425) is not in mixedCase
Parameter EUSD.burnShares(address,uint256)._account (contracts/lybra/token/EUSD.sol#445) is not in mixedCase
Parameter EUSD.burnShares(address,uint256)._sharesAmount (contracts/lybra/token/EUSD.sol#445) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Function EUSD.decimals() (contracts/lybra/token/EUSD.sol#112-114) contains magic number: 18
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/magic_number.md
```

LBR.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/LBR.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:
Constant LBR.maxSupply (contracts/lybra/token/LBR.sol#14) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

LBRL2.sol

```
INFO:Detectors:
Pragma version^0.8.17 (contracts/lybra/token/LBRL2.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
```

PeUSD.sol

```
INFO:Detectors:
Pragma version^0.8.17 (contracts/lybra/token/PeUSD.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
```

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#unused-return
INFO:Detectors:
PeUSDMainnet executeFlashloan parameter from is not related to msg.sender success = EUSD.transferFrom(address(receiver),address(this),EUSD.getMintedEUSDByShares(shareAmount)) (contracts/lybra/token/PeUSDMainnet.sol#132)
Reference: https://ventral.digital/posts/2022/8/18/sznsdaos-bountyboard-unauthorized-transferfrom-vulnerability
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)
        Event emitted after the call(s):
        - ConvertToEUSD(msg.sender,peusdAmount,EUSD.getMintedEUSDByShares(share),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)
        - receiver.onFlashLoan(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):
        - Flashloaned(address(receiver),eusdAmount,burnShare,block.timestamp) (contracts/lybra/token/PeUSDMainnet.sol#138)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Function PeUSDMainnet.executeFlashloan(IFlashBorrower,uint256,bytes) (contracts/lybra/token/PeUSDMainnet.sol#127-139) has a dubious typecast: address<=address
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/dubious_typecast.md
INFO:Detectors:
Pragma version^0.8.17 (contracts/lybra/interfaces/IEUSD.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/token/PeUSDMainnet.sol#14) 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 IEUSD.getSharesByMintedEUSD(uint256)._EUSDAmount (contracts/lybra/interfaces/IEUSD.sol#32) 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
Reference: https://github.com/pessimistic-io/slitherin/blob/master/docs/magic_number.md
INFO:Detectors:
In a function PeUSDMainnet.convertToPeUSD(address,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.convertToEUSD(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/pessimistic-io/slitherin/blob/master/docs/multiple_storage_read.md
```

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
solc-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#19) 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._nonblockingLzReceive(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#unused-return
INFO:Detectors:
Reentrancy in LBRMinerFromL2._nonblockingLzReceive(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):
        - ClaimReward(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 randonmness.
165	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randonmness.

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

234	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
234	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
265	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
265	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
268	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
268	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
279	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "++" discovered
280	(SWC-110) Assert Violation	Unknown	Out of bounds array access
289	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
291	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "-" discovered
291	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "*" discovered
292	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+" discovered
292	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "/" discovered
297	(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) Floating Pragma	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) Floating Pragma	Low	A floating pragma is set.

LBR.sol

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

LBRL2.sol

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

PeUSD.sol

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

PeUSDMainnet.sol

Line	SWC Title	Severity	Short Description
14	(SWC-103) Floating Pragma	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 randonmness.
53	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randonmness.

LBRMinerFromL2.sol

Line	SWC Title	Severity	Short Description
3	(SWC-103) Floating Pragma	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