Audit notes 👍
: —- always try to read the docs.
— continue checking for panic reverts()

Staking Protocol (lybra)

- Always read docs

- Always read natspec

- Lido rebases can be in the negative, meaning token balances can reduce. With lido stETH, they should always be checking for balanceOf() and not saving balances gotten at a previous time to storage. Check that the protocol doest award tokens or redeem tokens based on stETH amount sent in by user at the point of entry. It should allow redemption in a percentage calc method. Where token to be redeemed is only a representative of percentage of stETH in the system and not some 1:1 ratio/ hardcoded ratio redemption. Calc redemption amounts when a negative rebase happens and when a positive one happens. Ensure that every user can be paid back without reverts or losses.
  Example issue → https://github.com/code-423n4/2023-06-lybra-findings/issues/931

- Things like safeCollateral ratio, unsafeCollateralRatio, min and max values used in liquidation and redemption should have a way to be changed especially in extreme market conditions.
  Example issue → https://github.com/code-423n4/2023-06-lybra-findings/issues/882, although this issue is caused because to change the ratio, a static call is made to an internal function. It wont work. Error on dev part.

- Double check that staking rewards are calculated only using variables related to the staking tokens/ total amount staked by users or amounts staked by user. Tokens that signify borrowers should not be added to calculation unless explicitly specified in the docs or natspec.
  Example issue → https://github.com/code-423n4/2023-06-lybra-findings/issues/867

- If token balances of a contract are used in any calc. Check that u cannot use flashloan to increase the token balance to manipulate the calculation.

- When calculating rewards to be paid, make sure it cannot be recalculated/modified in storage via public functions.
  Example issue → https://github.com/code-423n4/2023-06-lybra-findings/issues/794 (check its eslbr mint and burn fcns and refreshReward fcn)

- Check that after staking when withdrawing, if there is a duration of staking specified by user. Withdrawals should not happen before that. Also if there is a reward for longer locks. It should be applied to the reward gotten after staking.

- Check that any executeFlashLoan fcn or flashLoan fcn can only be executed by msg.sender for itself. So that it will not be able to manipulate for another address as the other address can be a contract with a fallback()
  Example issue → **https://github.com/code-423n4/2023-06-lybra-findings/issues/769**

- Ensure that collateralRatio calculation is the same thing everywhere
  Example issue → https://github.com/code-423n4/2023-06-lybra-findings/issues/723.

- In redemptions/repayments the user debt must always include the fees, it should be borrowed amount + fees for the borrow.
  Example issue → https://github.com/code-423n4/2023-06-lybra-findings/issues/679

- Check all contracts constructors,, if a proxy implementation contract, it must not have code it its constructor or have any constructor at all.
  Lybra example → https://github.com/code-423n4/2023-06-lybra-findings/issues/673

- When u see a mint(), check that its _mint() are not called anywhere else other than in the main token code because calling _mint() outside can cause bypass of checks in mint(). If mint amount is calculated, Check that it is not mathematically possible to cause the mint amount to be ridiculously large.
  Example → https://github.com/code-423n4/2023-06-lybra-findings/issues/647

- Fees taken from repayments or burns and other actions and paid to the protocol or other actors are part of whats in circulation and accounting should cover that
  Example → https://github.com/code-423n4/2023-06-lybra-findings/issues/532

- If rewards are calculated similarly to synthetix contracts or they look like (EUSDMinningIncentives.sol, ProtocolRewardsPool.sol and stakerRewardsV2Pool.sol) and have notifyRewardAmount() and getRewards() function. Check that they dont have the vuln described here → https://github.com/code-423n4/2023-06-lybra-findings/issues/484

- Dont check use token.BalanceOf() to check balances of liquidity pool(LP) pair contracts. IT may BE MANIPULATED VIA FLASHLOANS
  Example issue → https://github.com/code-423n4/2023-06-lybra-findings/issues/442

- https://github.com/code-423n4/2023-06-lybra-findings/issues/340

- https://github.com/code-423n4/2023-06-lybra-findings/issues/290

- In governor contracts that use votingPeriod and votingDelay check that the values are set or set well
  Example issue → https://github.com/code-423n4/2023-06-lybra-findings/issues/268

- When asets are being converted, i.e from stable coin to volatile asset., there must be slippage loss protection. Check that user can set minAmount to be received and timestamp where it can be executed on or before
  Example issue → https://github.com/code-423n4/2023-06-lybra-findings/issues/221

- Block.number doesnt work right on arbitrum and optimism
  Example issue → https://github.com/code-423n4/2023-06-lybra-findings/issues/114

- totalShares does not always eq totalSupply.
  Example issue → https://github.com/code-423n4/2023-06-lybra-findings/issues/54

- When liquidations happen, if position is overcollaterized collateral ratios should be increased a bit to make them healthier.
  Example issue → https://github.com/code-423n4/2023-06-lybra-findings/issues/72

- Validate all external calls are made correctly and the fcns exist
- Check how state values are being stored and used
- Read comments well!!!!!

Amphora Findings (uses convex, cvx and crv)

- Convex rewards allow anyone to call getRewards() on behalf of any users. https://github.com/convex-eth/platform/blob/main/contracts/contracts/BaseRewardPool.sol#L263-L279. This will break the Vault `claimRewards()` functionality that depends on this ``getReward()`
- Always be wary of token transfer actions for re-eentrancy.

- Vault.claimRewards() assumes that CVX will always get minted to the Vault (if there is a CRV reward). If CVX does not get minted, the claim will fail, preventing payout of CRV and extra rewards. Looking at the Convex code there can be the case where the operator has changed, causing no mint to happen. In this case, CVX token transfer may fail.
- Check arithmetic operations very well, maths that can cause 0 value can be exploited too, especially when one of the values is an external value. Check for divisions that lead to 0 value too. Like 100/1000.
- When users can deploy new contracts using create opcode, be careful of reorgs if they only use create opcode.
  Example issue → https://github.com/code-423n4/2023-07-amphora-findings/issues/233
- When Convex pool of an asset that has ***CurveLPStakedOnConvex*** collateral type is shut down, users can no longer deposit that asset into vault since it would always revert on Convex booster `deposit()` function, called during vault's `depositERC20()` process. With lack of method to update it, protocol would the ability to accept it as collateral even as plain LP

without staking. Also users that use the asset as collateral may face risk of being liquidated. This is also applicable when Convex itself shutdown their booster contract.

Arcade Findings ()

- Check that voting cannot be manipulated using flashloans or undercollateraized loans or collateralized loans in a single block. Check if voting power depend on token balance of msg.sender for this to be possible. TWAPs over time should be used to determine voting power instead of token balance in prev block or current block.
- Check that users cannot just add to arrays in storage and no iterations can happen to arrays that users can add to with no restrictions as if the array grows too large and we try to iterate through it we can get a gas limit DOS.
- Voting power for all uses should be updated when multiplier changes
- If merkleroot can be changed, check that user that has claimed or been validated before can use the fcn again incase of a change or withdraw its increased tokens
  Example issue → https://github.com/code-423n4/2023-07-arcade-findings/issues/213.
- Check that multiple functions that dont do the same thing/are unrelated. dont change the same values in storage. Example issue → https://github.com/code-423n4/2023-07-arcade-findings/issues/85
- Verify all constants in the contracts
- Check that u can add to storage values and they deduct from them later without reverts. Example issue → https://github.com/code-423n4/2023-07-arcade-findings/issues/58

Lens Findings

- If the project is using IDs, in a blockchain reorg. The id may be removed along with the block in which it was created.
- If there are upgrade functions, check that all users states are transferred well! Things can go wrong in upgrades
- Check that if contract is paused users cannot update state.
- During migrations check that a v1 user info/state cannot be made by another user on v2 before v1 user is migrated.
- If a function uses an EIP 712, check that its array or dynamic type is encoded well. According to the specification. It should be like “ keccak256(abi.encodePacked(idsOfProfilesToUnfollow)),” not “ keccak256(idsOfProfilesToUnfollow),”
  Example issue - https://github.com/code-423n4/2023-07-lens-findings/issues/142
- Read docs and verify that code works as intended!
- Check that user cannot self follow or do on it self actions that are intended to be done by others for it.
- Check that u cannot use an upgrade function to change balances, change state or rest balances or reset any state. Check that after migration state is the same.

- Check that migration doesnt fail for any user under any edge case.

Chainlink Findings

- When fetching rewards based on snapshot, make sure that the snapshots are gotten the same way everytime the multiplier duration is calculated https://github.com/code-423n4/2023-08-chainlink-findings/issues/982
- Check that admin role cannot be transferred faster than delay period. If project implements a delay period
- Removed Operator /party should be able to remove his remaining stake capital after removal
- Check that contracts functions cannot mistakenly recognise one type of user for another type. Eg recognise community staker as operator.
- Check that pool rate, emission rate cannot be calculated to be very tiny amounts compared to other rates in protocol. It may cause reverts in other airthmetics (heck tho) https://github.com/code-423n4/2023-08-chainlink-findings/issues/774
- Check that all values in a calc may not revert when they are zero or they may not become zero ever
- Check that during migration, rewards and principal are sent to the new contract.
- Understand more about how multipliers work and how they affect reward in staking protocols
- If timelock delay is set in code, chck that it is not too short compared to other delays
- Check that storage values are well adjusted in storage in functions. I.e totalPriincipal is reduced when unstaking.
- When migrating rewards to a new vault. Check that the rates for calc in the new vault is same as rate for calc in the old vault
- Check that paused state affects all contract functions and users
- Check that rate changes does not affect the old principal. Eg staker stakes now at 1% apr rate. Reward is accrued and set so should not change. Admin changes rate to 2%, if user adds new stake. The new rate should only affect the new stake and not the old stake. https://github.com/code-423n4/2023-08-chainlink-findings/issues/500
- If two functions calculate same reward, check that they use the same logic. https://github.com/code-423n4/2023-08-chainlink-findings/issues/494
- Check thay reward buckets cannot have their funds co-mingled. If one bucket is 0, and another is 10, at migration they should not be added together and rewards splitted between their users https://github.com/code-423n4/2023-08-chainlink-findings/issues/395
- Check that stakers can take out all their principal at least

Ethena Findings -

- Take note of protocol business logic, admin should not be able to send reward to restricted/blocked users. Or add them back
- Restricted users should not be able to do actions they are restricted from doing. You must test it to prove it. Come up with different scenarios like restriction after deposit or restriction before deposit.
- Check that restricted users cannot approve another user to act(wthdraw or stake) on their behalf
- Check that any calculations don't rely on a balance of the contract check. I.e balanceOf(address(this)). It can be manipulated to DOS the next user. https://github.com/code-423n4/2023-10-ethena-findings/issues/88
- Check that admin changes to parameters dont affect withdrawal actions started before that withdrawal. https://github.com/code-423n4/2023-10-ethena-findings/issues/29

Party DAO Findings -

- Check that payable functions when called in another function specify a msg.value in the call
- 51% attack on governance mechanisms to steal tokens
- Always check the docs and make sure they always comply to the erc standards they use
- Minimum number of votes to pass a proposal should be unique/locked in to a proposal at the time of making the proposal so that malicious actor may not reduce the threshold to pass their malicious proposal later if they dont get enough votes. https://github.com/code-423n4/2023-10-party-findings/issues/295
- Check that a transfer of admin duties from one address to another doesnt count as two admins. So one admin doesnt approve then transfer his dduties to another admin to approve. This will mean two admins where as its just one person. https://github.com/code-423n4/2023-10-party-findings/issues/233
- When there is a min or max accepted amount. Check that you cannot get to a scenario where by min is surpassed and max accepted cannot be reached because user deposit is will take contributions above max. https://github.com/code-423n4/2023-10-party-findings/issues/127
- Always check math very well. See how it performs under extremes. Try to force reverts https://github.com/code-423n4/2023-10-party-findings/issues/119

Kelp DAO findings -

- Chainlink price discrepancy is possible
  https://github.com/code-423n4/2023-11-kelp-findings/issues/584
- Eigen layer can update their strategy address, if protocol code is checking eigen layer strategy balance it must take into account the old strategy balance and new strategy balance in its accounting.
  https://github.com/code-423n4/2023-11-kelp-findings/issues/197
- When exchanging one asset for the other, if assets is worth values, always have a slippage control mechanism
- If calculating amount to mint based on what user deposits, dont add “user deposit amount” to” total deposits” before using that “total deposits” in a calculation for the amount to mint to user.
  https://github.com/code-423n4/2023-11-kelp-findings/issues/62
- Check for donation attacks, 0 value transfers to users should not be allowed.
  https://github.com/code-423n4/2023-11-kelp-findings/issues/42

Revolution Findings -

- Prevent JSON injection attacks in nft tokens
  https://github.com/code-423n4/2023-12-revolutionprotocol-findings/issues/167
- Arrays should be keccack hashed before passing into a signature typehash. Check the eip-712 specification.
  https://github.com/code-423n4/2023-12-revolutionprotocol-findings/issues/77
- quoromVotes must never be bypassed
  https://github.com/code-423n4/2023-12-revolutionprotocol-findings/issues/409
- It must not break erc 721 if nfts are used. tokenURI() fcn must check that a token ID exists and revert if not.
  https://github.com/code-423n4/2023-12-revolutionprotocol-findings/issues/471
- https://github.com/code-423n4/2023-12-revolutionprotocol-findings/issues/266
- Check that DOS due to OOG doesnt happen in functions that iterate over loops or arrays in storage.
  https://github.com/code-423n4/2023-12-revolutionprotocol-findings/issues/178
- All BPS variables should have a min and max. Check that it cantr be set too high to cause a DOS
  https://github.com/code-423n4/2023-12-revolutionprotocol-findings/issues/160

WildCat Findings -

- Check for weird token behaviors - https://github.com/code-423n4/2023-10-wildcat-findings/issues/503
- Check for rounding errors - https://github.com/code-423n4/2023-10-wildcat-findings/issues/501
- Be wary of calculations involving contract balance after an asset is transferred in. https://github.com/code-423n4/2023-10-wildcat-findings/issues/500. Calculations and asset transfer in should happen in a way that doesnt inflate/deflate price or amount calculated.
- Create2 opcode must be implemented with a check for address(0) in case of deployment failing https://github.com/code-423n4/2023-10-wildcat-findings/issues/499
- `address.Codehash != bytes(0)` to check if an address is a contract or not may be risky https://github.com/code-423n4/2023-10-wildcat-findings/issues/491
- Borrower has no way to update `maxTotalSupply` of `market` or

  close market.

  https://github.com/code-423n4/2023-10-wildcat-findings/issues/431
- Check that no changes outside the min and max specified values is allowed. https://github.com/code-423n4/2023-10-wildcat-findings/issues/196
- Another break of business logic https://github.com/code-423n4/2023-10-wildcat-findings/issues/68

Tapioca Findings -

- Be careful of payloads and arbitrary calls, especially with random/permissionless addresses. https://github.com/code-423n4/2023-07-tapioca-findings/issues/1695

- Uniswap poolFee should not be hardcoded. Hardcoding it may lead to suboptimal routes with stargate https://github.com/code-423n4/2023-07-tapioca-findings/issues/1553
- LP token balance is never same as amount of underlying asset. Be watchful when projects use LP token balance to represent the underlying asset amount https://github.com/code-423n4/2023-07-tapioca-findings/issues/1520

- Dont use uniswap TWAP on L2s like optimism. It may not be always accurate. Optimism advises to avoid using it.
  https://github.com/code-423n4/2023-07-tapioca-findings/issues/1474
- **Yearn vaults integration bug** : BPT_IN_FOR_EXACT_TOKENS or yearn query logic 2 is susceptible to manipulation.
  https://github.com/code-423n4/2023-07-tapioca-findings/issues/1449
- When using yearn vaults, always specify loss the strategy is willing to take to prevent DOS
  https://github.com/code-423n4/2023-07-tapioca-findings/issues/1456
- It is always better to buy STETH from uniswap pools than buying directly from LIDO because it is cheaper in pools
  https://github.com/code-423n4/2023-07-tapioca-findings/issues/1437
- Using ```curveStEthPool.get_dy(1, 0, stEthBalance``` can cause susceptible to manipulation and view only reentrancy.
- Don’t price or use stETH price values as eth or vice versa. This can be manipulated.
  https://github.com/code-423n4/2023-07-tapioca-findings/issues/1432
- Don’t use high slippage in curve asset swaps
  https://github.com/code-423n4/2023-07-tapioca-findings/issues/1430
- Curve allows claiming rewards on behalf of other contracts. We can grief a legit contract that does logic based on amount of rewards returned from a curve reward check. Claiming on behalf of the contract before the contract does will cause the logic that relies on the reward check to not run in some instances.
  https://github.com/code-423n4/2023-07-tapioca-findings/issues/1429
  https://github.com/code-423n4/2023-07-tapioca-findings/issues/1425
- Be wary of static call usage, should not be used on functions that change values in storage
  https://github.com/code-423n4/2023-07-tapioca-findings/issues/1428
- Functions using Uniswap swap calls must always allow users to specify a deadline and not be hardcoded.
  https://github.com/code-423n4/2023-07-tapioca-findings/issues/1408
- Contracts that receive NFTs via safeTransferFrom must implement onERC721Received()
  https://github.com/code-423n4/2023-07-tapioca-findings/issues/1405
- Don't use 0 as minOut or as slippage. Let the user set it.
  https://github.com/code-423n4/2023-07-tapioca-findings/issues/1368
- Always send gas as ether when doing cross chain transactions
  https://github.com/code-423n4/2023-07-tapioca-findings/issues/1264
- Flashloan receiver must approve msg.sender
  https://github.com/code-423n4/2023-07-tapioca-findings/issues/1223
- Possible to block layer zero channel
  https://github.com/code-423n4/2023-07-tapioca-findings/issues/1220
- Can block layer zero channel to DoS and cause users to lose gas
  https://github.com/code-423n4/2023-07-tapioca-findings/issues/1207

- Check that external calls are able to return value if the implementation expects them to https://github.com/code-423n4/2023-07-tapioca-findings/issues/1184
- Do not pause repay() and liquidate() functions in a lending protocol https://github.com/code-423n4/2023-07-tapioca-findings/issues/1169
- Lack of safety buffer between liquidation threshold and LTV ratio for borrowers to prevent unfair liquidations https://github.com/code-423n4/2023-07-tapioca-findings/issues/1164
- Because reward tokens are passed in as array, attacker can pass same address as two different array elements and get rewards transferred to it https://github.com/code-423n4/2023-07-tapioca-findings/issues/1094
- Don't allow arbitrary addresses supplied by user to be used in a delegateCall https://github.com/code-423n4/2023-07-tapioca-findings/issues/1083
- When decoding calldata, arguments encoded must be same as arguments decoded if not there will be a revert https://github.com/code-423n4/2023-07-tapioca-findings/issues/1069
- If user must borrow shares, they must have allowance to do so from the lender https://github.com/code-423n4/2023-07-tapioca-findings/issues/1061
- Liquidations should not revert, in extreme market conditions https://github.com/code-423n4/2023-07-tapioca-findings/issues/1026
- When granting approvals, checck that it is the granted address that can use the approval https://github.com/code-423n4/2023-07-tapioca-findings/issues/1019
- Don't hardcode aave pool address, it is an integration flaw to do so https://github.com/code-423n4/2023-07-tapioca-findings/issues/993
- If important contract addr is changed in another contract, especially one that swaps tokens, the new contract addr must be given approval to swap too https://github.com/code-423n4/2023-07-tapioca-findings/issues/990
- Emergency withdrawal fcn with no pause for deposits or withdrawals fcns is useless https://github.com/code-423n4/2023-07-tapioca-findings/issues/989

- Tokens staked in aave + rewards for the stake can only be retrieved via calls to stakeTokenReward.redeem() https://github.com/code-423n4/2023-07-tapioca-findings/issues/943
- When integrating nfts i.e erc721 and erc1155, check for what happens if nft is erc1155 but with each token id having a maxSupply of 1. https://github.com/code-423n4/2023-07-tapioca-findings/issues/916
-