Shell Protocol

Scope

The code under review can be found within the C4 Shell Protocol repository.

Summary

Findings

ID	Issue	Severity
L-01	Use a more recent version of Solidity	Low
L-02	Missing check for duration in constructor	Low
L-03	Missing event emission for critical storage configurations	Low
L-04	Use a sentinel value in SpecifiedToken enum to represent default state	Low
L-05	Mathematical terminology should be updated to better suit the intentions of the calculations	Low
N-01	Refactor contract architecture to improve code readability and maintainability	Non-Critical

ID	Issue	Severity
N-02	Incorrect comments should be rectified for better code understandability	Non-Critical
N-03	Use <= instead of < to correctly match intention of the INT_MAX threshold	Non-Critical
N-04	Refactor code to improve code readability and maintainabiilty	Non-Critical

Findings

[L-01] Use a more recent version of Solidity

There is 1 instance of this:

Note: This finding is not included in the QA findings of the bot report.

https://github.com/code-423n4/2023-08-shell/blob/c61cf0e01bada04c3d6055acb81f61955ed600aa/src/proteus/EvolvingProteus.sol#L4

The current compiler version for EvolvingProteus.sol is =0.8.10. Consider moving to the latest version 0.8.19 since it has lesser (**currently known**) bugs than 0.8.10. View bugs for 0.8.10 and 0.8.19 in the bugs_by_version.json file provided by the Solidity team.

```
File: src/proteus/EvolvingProteus.sol
4: pragma solidity =0.8.10;
```

[L-02] Missing check for duration in constructor

There is 1 instance of this:

https://github.com/code-423n4/2023-08-shell/blob/c61cf0e01bada04c3d6055acb81f61955ed600aa/src/proteus/EvolvingProteus.sol#L262

The constructor below does not contain a check for duration to be greater than 0. It should not accept durations with 0 (in case of deployer mistake) as it may lead to improper behaviour such as reverting whenever calling function t(). This reversion occurs since duration(self) is zero and the divu() function from the ABDK library expects y parameter to never be zero. Additionally if the check is not implemented, it may lead to an unnecessary contract being deployed.

constructor:

```
File: src/proteus/EvolvingProteus.sol
243:     constructor(
244:         int128 py_init,
245:         int128 px_init,
246:         int128 py_final,
```

```
247:         int128 px_final,
248:         uint256 duration
249:     ) {
250:         // price value checks
251:         if (py_init >= MAX_PRICE_VALUE || py_final >=
MAX_PRICE_VALUE) revert MaximumAllowedPriceExceeded();
252:         if (px_init <= MIN_PRICE_VALUE || px_final <=
MIN_PRICE_VALUE) revert MinimumAllowedPriceExceeded();
253:
254:         // at all times x price should be less than y price
255:         if (py_init <= px_init) revert InvalidPrice();
256:         if (py_final <= px_final) revert InvalidPrice();
257:
258:         // max. price ratio check
259:         if (py_init.div(py_init.sub(px_init)) >
ABDKMath64x64.divu(uint(MAX_PRICE_RATIO), 1)) revert
MaximumAllowedPriceRatioExceeded();
260:         if (py_final.div(py_final.sub(px_final)) >
ABDKMath64x64.divu(uint(MAX_PRICE_RATIO), 1)) revert
MaximumAllowedPriceRatioExceeded();
261:
262:         config = LibConfig.newConfig(py_init, px_init, py_final,
px_final, duration);
263:       }
```

function t():

```
File: src/proteus/EvolvingProteus.sol
89:     function t(Config storage self) public view returns (int128) {
90:         return elapsed(self).divu(duration(self));
91:     }
```

[L-03] Missing event emission for critical storage configurations

Any changes made to storage variables should emit an event for off-chain tracking as well as deployer awareness.

There is 1 instance of this issue:

https://github.com/code-423n4/2023-08-shell/blob/c61cf0e01bada04c3d6055acb81f61955ed600aa/src/proteus/EvolvingProteus.sol#L243

The constructor sets the values for struct Config on Line 262 but does not emit an event. constructor:

```
File: src/proteus/EvolvingProteus.sol
243:     constructor(
244:         int128 py_init,
245:         int128 px_init,
246:         int128 py_final,
```

```
247:         int128 px_final,
248:         uint256 duration
249:     ) {
250:         // price value checks
251:         if (py_init >= MAX_PRICE_VALUE || py_final >=
MAX_PRICE_VALUE) revert MaximumAllowedPriceExceeded();
252:         if (px_init <= MIN_PRICE_VALUE || px_final <=
MIN_PRICE_VALUE) revert MinimumAllowedPriceExceeded();
253:
254:         // at all times x price should be less than y price
255:         if (py_init <= px_init) revert InvalidPrice();
256:         if (py_final <= px_final) revert InvalidPrice();
257:
258:         // max. price ratio check
259:         if (py_init.div(py_init.sub(px_init)) >
ABDKMath64x64.divu(uint(MAX_PRICE_RATIO), 1)) revert
MaximumAllowedPriceRatioExceeded();
260:         if (py_final.div(py_final.sub(px_final)) >
ABDKMath64x64.divu(uint(MAX_PRICE_RATIO), 1)) revert
MaximumAllowedPriceRatioExceeded();
261:
262:         config = LibConfig.newConfig(py_init, px_init, py_final,
px_final, duration);
263:       }
```

[L-04] Use a sentinel value in SpecifiedToken enum to represent default state

enum SpecifiedToken:

```
File: src/proteus/ILiquidityPoolImplementation.sol
6: enum SpecifiedToken {
7:     X,
8:     Y
9: }
```

https://github.com/code-423n4/2023-08-shell/blob/c61cf0e01bada04c3d6055acb81f61955ed600aa/src/proteus/EvolvingProteus.sol#L272

The first element in the enum SpecifiedToken should be a sentinel value such as `UNINITIALIZED`. This will prove useful in functions taking SpecifiedToken input as a parameter.

Let's take an example: If the swapGivenInputAmount function caller forgets to pass in a parameter value for SpecifiedToken inputToken (which was supposed to be Y), it will default to X and incorrectly provide an outputAmount to swap for the wrong token. Although this is the function caller's mistake, it should be addressed for better security with input mistakes. **(Note: This contract is read-only since it comprises of only view functions, therefore such input mistakes should be addressed with validation checks).**

Solution: Use a sentinel value: Instead of relying on the default value X, we can define a sentinel value (a special value of the enum) that represents an uninitialized state. Then, we can check if the provided value is

equal to the sentinel value to handle the case where the parameter was not properly set.

```
File: src/proteus/EvolvingProteus.sol
272:     function swapGivenInputAmount(
273:         uint256 xBalance,
274:         uint256 yBalance,
275:         uint256 inputAmount,
276:         SpecifiedToken inputToken
277:     ) external view returns (uint256 outputAmount) {
278:         // input amount validations against the current balance
279:         require(
280:             inputAmount < INT_MAX && xBalance < INT_MAX && yBalance <
INT_MAX
281:         );
282:
283:         _checkAmountWithBalance(
284:             (inputToken == SpecifiedToken.X) ? xBalance : yBalance,
285:             inputAmount
286:         );
287:
288:         int256 result = _swap(
289:             FEE_DOWN,
290:             int256(inputAmount),
291:             int256(xBalance),
292:             int256(yBalance),
293:             inputToken
294:         );
295:         // amount cannot be less than 0
296:         require(result < 0);
297:
298:         // output amount validations against the current balance
299:         outputAmount = uint256(-result);
300:         _checkAmountWithBalance(
301:             (inputToken == SpecifiedToken.X) ? yBalance : xBalance,
302:             outputAmount
303:         );
304:     }
```

[L-05] Mathematical terminology should be updated to better suit the intentions of the calculations

There is 1 instance of this:

https://github.com/code-423n4/2023-08-shell/blob/c61cf0e01bada04c3d6055acb81f61955ed600aa/src/proteus/EvolvingProteus.sol#L716

In mathematical terminology, $b^2 - 4ac$ is the discriminant and not $\sqrt{b^2 - 4ac}$ Thus, the variable name should be updated from disc to sqrtDisc to better match the implementation of the calculations and the usage of correct mathematical terminology.

```
File: src/proteus/EvolvingProteus.sol
716: int256 disc = int256(Math.sqrt(uint256((bQuad**2 -
(aQuad.muli(cQuad)*4)))));
```

[N-01] Refactor contract architecture to improve code readability and maintainability

There are 2 instances of this:

https://github.com/code-423n4/2023-08-shell/blob/c61cf0e01bada04c3d6055acb81f61955ed600aa/src/proteus/EvolvingProteus.sol#L44
https://github.com/code-423n4/2023-08-shell/blob/c61cf0e01bada04c3d6055acb81f61955ed600aa/src/proteus/EvolvingProteus.sol#L137

To improve code readability and maintainability of EvolvingProteus.sol file, library LibConfig should be stored in a separate file than the EvolvingProteus.sol contract.

```
File: src/proteus/EvolvingProteus.sol
44:  library LibConfig {

137: contract EvolvingProteus is ILiquidityPoolImplementation {
```

https://github.com/code-423n4/2023-08-shell/blob/c61cf0e01bada04c3d6055acb81f61955ed600aa/src/proteus/EvolvingProteus.sol#L1

The EvolvingProteus contract can be divided into two files, one with the internal functions and the other with the external functions. This maintains the code and makes it shorter, modular and easy to read rather than the current file size of 500 nSLOC.

[N-02] Incorrect comments should be rectified for better code understandability

There are 4 instances of this:

https://github.com/code-423n4/2023-08-shell/blob/c61cf0e01bada04c3d6055acb81f61955ed600aa/src/proteus/EvolvingProteus.sol#L459

This correction is being made since we use FEE_DOWN in the function withdrawGivenInputAmount().

Incorrect comment:

```
459:      * @dev We use FEE_UP because we want to increase the perceived
amount of
460:      *  reserve tokens leaving the pool and to increase the observed
amount of
461:      *  LP tokens being burned.
```

Corrected comment:

```
459:      * @dev We use FEE_DOWN because we want to decrease the perceived
amount of
460:      *  reserve tokens leaving the pool and to decrease the observed
amount of
461:      *  LP tokens being burned.
```

https://github.com/code-423n4/2023-08-shell/blob/c61cf0e01bada04c3d6055acb81f61955ed600aa/src/proteus/EvolvingProteus.sol#L120 Incorrect comment:

```
120: @notice Calculates the b variable in the curve eq which is basically
a sq. root of the inverse of x instantaneous price
```

Corrected comment:

```
120: @notice Calculates the b variable in the curve eq which is basically
a sq. root of the of x instantaneous price
```

https://github.com/code-423n4/2023-08-shell/blob/c61cf0e01bada04c3d6055acb81f61955ed600aa/src/proteus/EvolvingProteus.sol#L173

This correction is being made since min price value is 10^-8.

Incorrect comment:

```
173:  The minimum price value calculated with abdk library equivalent to
10^12(wei)
```

Corrected comment:

```
173:  The minimum price value calculated with abdk library equivalent to
10^10(wei)
```

https://github.com/code-423n4/2023-08-shell/blob/c61cf0e01bada04c3d6055acb81f61955ed600aa/src/proteus/EvolvingProteus.sol#L831C9-L833C26

The comment does not include the BASE_FEE being cut from the absoluteValue. It should include the BASE_FEE in the comments to better suit the deduction occurring in the code.

Incorrect comment:

```
831:         // FIXED_FEE * 2 because we will possibly deduct the FIXED_FEE
from
832:         // this amount, and we don't want the final amount to be less
than
833:         // the FIXED_FEE.
```

[N-03] Use <= instead of < to correctly match intention of the INT_MAX threshold

INT_MAX variable:

```
File: src/proteus/EvolvingProteus.sol
146:  uint256 constant INT_MAX = uint256(type(int256).max);
```

The INT_MAX variable represents the type(int256).max value (typecasted to uint256). This variable represents the threshold which a token balance or input amount needs to respect by not crossing it. But these token balances or input amounts can be **equal** to the threshold. Although it's unlikely we come across such a value, there is no harm in including this extra edge case. Additionally, the use of < is inconsistent among other max threshold variable checks as well. For example, Line 800 and Line 813 use <= for threshold checks while Line 842 uses < for threshold checks.

There are 25 instances of this:

https://github.com/code-423n4/2023-08-shell/blob/c61cf0e01bada04c3d6055acb81f61955ed600aa/src/proteus/EvolvingProteus.sol#L280 https://github.com/code-423n4/2023-08-shell/blob/c61cf0e01bada04c3d6055acb81f61955ed600aa/src/proteus/EvolvingProteus.sol#L320 https://github.com/code-423n4/2023-08-shell/blob/c61cf0e01bada04c3d6055acb81f61955ed600aa/src/proteus/EvolvingProteus.sol#L362C1-L365C38 https://github.com/code-423n4/2023-08-shell/blob/c61cf0e01bada04c3d6055acb81f61955ed600aa/src/proteus/EvolvingProteus.sol#L398C1-L402C11 https://github.com/code-423n4/2023-08-shell/blob/c61cf0e01bada04c3d6055acb81f61955ed600aa/src/proteus/EvolvingProteus.sol#L435C1-L438C38 https://github.com/code-423n4/2023-08-shell/blob/c61cf0e01bada04c3d6055acb81f61955ed600aa/src/proteus/EvolvingProteus.sol#L472C1-L475C38 https://github.com/code-423n4/2023-08-shell/blob/c61cf0e01bada04c3d6055acb81f61955ed600aa/src/proteus/EvolvingProteus.sol#L590 https://github.com/code-423n4/2023-08-shell/blob/c61cf0e01bada04c3d6055acb81f61955ed600aa/src/proteus/EvolvingProteus.sol#L661 https://github.com/code-423n4/2023-08-shell/blob/c61cf0e01bada04c3d6055acb81f61955ed600aa/src/proteus/EvolvingProteus.sol#L842

```
File: src/proteus/EvolvingProteus.sol
280:  inputAmount < INT_MAX && xBalance < INT_MAX && yBalance < INT_MAX

320:  outputAmount < INT_MAX && xBalance < INT_MAX && yBalance < INT_MAX

362:  depositedAmount < INT_MAX &&
363:  xBalance < INT_MAX &&
364:  yBalance < INT_MAX &&
365:  totalSupply < INT_MAX

398:  mintedAmount < INT_MAX &&
399:  xBalance < INT_MAX &&
400:  yBalance < INT_MAX &&
401:  totalSupply < INT_MAX

435:  withdrawnAmount < INT_MAX &&
436:  xBalance < INT_MAX &&
437:  yBalance < INT_MAX &&
438:  totalSupply < INT_MAX

472:  burnedAmount < INT_MAX &&
473:  xBalance < INT_MAX &&
474:  yBalance < INT_MAX &&
475:  totalSupply < INT_MAX

590:  require(result < INT_MAX);

661:  require(result < INT_MAX);

842:  require(roundedAbsoluteAmount < INT_MAX);
```

[N-04] Refactor code to improve code readability and maintainabiilty

There is 1 instance of this:

https://github.com/code-423n4/2023-08-shell/blob/c61cf0e01bada04c3d6055acb81f61955ed600aa/src/proteus/EvolvingProteus.sol#L812C4-L813C72

The if-else block follows the pattern if(A) then do X, else if(B) then do X. If either of the conditions are met, we still execute the same statement X. Thus, it would make sense to refactor the conditions in one if block with the || operator.

Instead of this:

```
File: src/proteus/EvolvingProteus.sol
812:       if (finalBalanceRatio < MIN_M) revert BoundaryError(x,y);
813:       else if (MAX_M <= finalBalanceRatio) revert BoundaryError(x,y);
```

Use this:

```
File: src/proteus/EvolvingProteus.sol
812:       if (finalBalanceRatio < MIN_M || MAX_M <= finalBalanceRatio)
revert BoundaryError(x,y);
```