ABDK CONSULTING

SMART CONTRACT AUDIT

Sudoswap

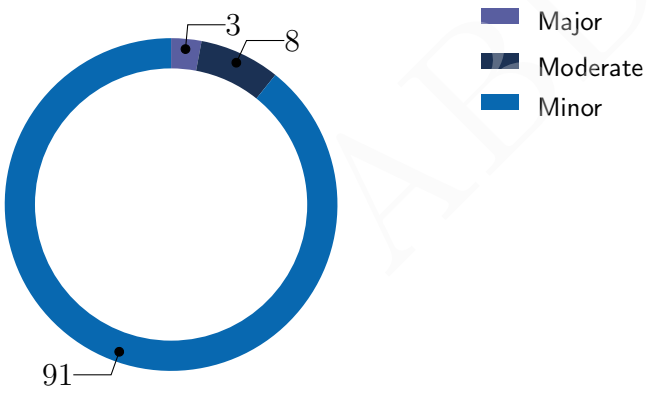
Solidity

abdk.consulting

SMART CONTRACT AUDIT CONCLUSION

by Mikhail Vladimirov and Dmitry Khovratovich
25th April 2022

We've been asked to review the 18 files in a Github repository. We found 3 major, and a few less important issues. All identified major issues have been fixed or otherwise addressed in collaboration with the client.



Findings

ID	Severity	Category	Status
CVF-1	Minor	Procedural	Info
CVF-2	Minor	Procedural	Info
CVF-3	Minor	Procedural	Info
CVF-4	Minor	Bad datatype	Fixed
CVF-5	Minor	Bad datatype	Fixed
CVF-6	Minor	Bad datatype	Fixed
CVF-7	Minor	Bad datatype	Fixed
CVF-8	Minor	Documentation	Fixed
CVF-9	Minor	Bad naming	Fixed
CVF-10	Minor	Readability	Info
CVF-11	Minor	Readability	Info
CVF-12	Minor	Bad datatype	Fixed
CVF-13	Minor	Bad datatype	Fixed
CVF-14	Minor	Bad datatype	Fixed
CVF-15	Minor	Bad datatype	Fixed
CVF-16	Minor	Suboptimal	Fixed
CVF-17	Minor	Procedural	Fixed
CVF-18	Minor	Flaw	Info
CVF-19	Minor	Procedural	Info
CVF-20	Moderate	Flaw	Fixed
CVF-21	Minor	Overflow/Underflow	Fixed
CVF-22	Minor	Flaw	Info
CVF-23	Minor	Suboptimal	Info
CVF-24	Minor	Documentation	Fixed
CVF-25	Major	Overflow/Underflow	Info
CVF-26	Minor	Flaw	Info
CVF-27	Minor	Procedural	Info

ID	Severity	Category	Status
CVF-28	Moderate	Unclear behavior	Info
CVF-29	Moderate	Suboptimal	Info
CVF-30	Minor	Suboptimal	Info
CVF-31	Moderate	Flaw	Fixed
CVF-32	Minor	Unclear behavior	Info
CVF-33	Moderate	Flaw	Fixed
CVF-34	Minor	Suboptimal	Fixed
CVF-35	Minor	Suboptimal	Info
CVF-36	Minor	Unclear behavior	Info
CVF-37	Minor	Suboptimal	Info
CVF-38	Major	Suboptimal	Info
CVF-39	Minor	Unclear behavior	Info
CVF-40	Minor	Suboptimal	Info
CVF-41	Minor	Suboptimal	Info
CVF-42	Minor	Readability	Info
CVF-43	Minor	Bad naming	Fixed
CVF-44	Minor	Documentation	Fixed
CVF-45	Minor	Documentation	Fixed
CVF-46	Minor	Bad naming	Fixed
CVF-47	Minor	Bad datatype	Fixed
CVF-48	Minor	Suboptimal	Info
CVF-49	Minor	Readability	Fixed
CVF-50	Minor	Flaw	Fixed
CVF-51	Minor	Suboptimal	Info
CVF-52	Minor	Procedural	Info
CVF-53	Minor	Procedural	Fixed
CVF-54	Moderate	Overflow/Underflow	Fixed
CVF-55	Minor	Suboptimal	Fixed
CVF-56	Minor	Suboptimal	Info
CVF-57	Minor	Bad datatype	Fixed

ID	Severity	Category	Status
CVF-58	Minor	Bad datatype	Fixed
CVF-59	Minor	Suboptimal	Fixed
CVF-60	Minor	Suboptimal	Fixed
CVF-61	Minor	Suboptimal	Fixed
CVF-62	Minor	Suboptimal	Info
CVF-63	Minor	Suboptimal	Fixed
CVF-64	Minor	Suboptimal	Fixed
CVF-65	Minor	Unclear behavior	Fixed
CVF-66	Minor	Bad datatype	Fixed
CVF-67	Minor	Documentation	Fixed
CVF-68	Minor	Procedural	Fixed
CVF-69	Minor	Procedural	Info
CVF-70	Moderate	Flaw	Info
CVF-71	Moderate	Suboptimal	Fixed
CVF-72	Major	Flaw	Fixed
CVF-73	Minor	Bad datatype	Fixed
CVF-74	Minor	Documentation	Fixed
CVF-75	Minor	Procedural	Info
CVF-76	Minor	Suboptimal	Info
CVF-77	Minor	Suboptimal	Info
CVF-78	Minor	Unclear behavior	Info
CVF-79	Minor	Procedural	Info
CVF-80	Minor	Procedural	Fixed
CVF-81	Minor	Unclear behavior	Info
CVF-82	Minor	Readability	Info
CVF-83	Minor	Suboptimal	Info
CVF-84	Minor	Documentation	Fixed
CVF-85	Minor	Suboptimal	Info
CVF-86	Minor	Procedural	Info
CVF-87	Minor	Procedural	Fixed

ID	Severity	Category	Status
CVF-88	Minor	Suboptimal	Info
CVF-89	Minor	Readability	Info
CVF-90	Minor	Suboptimal	Info
CVF-91	Minor	Procedural	Fixed
CVF-92	Minor	Procedural	Info
CVF-93	Minor	Suboptimal	Fixed
CVF-94	Minor	Procedural	Info
CVF-95	Minor	Readability	Fixed
CVF-96	Minor	Suboptimal	Fixed
CVF-97	Minor	Documentation	Fixed
CVF-98	Minor	Documentation	Fixed
CVF-99	Minor	Documentation	Fixed
CVF-100	Minor	Suboptimal	Info
CVF-101	Minor	Documentation	Fixed
CVF-102	Minor	Procedural	Info

Contents

1 Document properties

Version

Version	Date	Author	Description
0.1	April 21, 2022	D. Khovratovich	Initial Draft
0.2	April 25, 2022	D. Khovratovich	Minor revision
1.0	April 25, 2022	D. Khovratovich	Release

Contact

D. Khovratovich

khovratovich@gmail.com

2 Introduction

The following document provides the result of the audit performed by ABDK Consulting at the customer request. The audit goal is a general review of the smart contracts structure, critical/major bugs detection and issuing the general recommendations.
We have reviewed the contracts at the 7eb85bb commit:

- bonding-curves/CurveErrorCodes.sol
- bonding-curves/ExponentialCurve.sol
- bonding-curves/ICurve.sol
- bonding-curves/LinearCurve.sol
- lib/LSSVMPairCloner.sol
- lib/Ownable.sol
- LSSVMPair.sol
- LSSVMPairERC20.sol
- LSSVMPairETH.sol
- LSSVMPairEnumerable.sol
- LSSVMPairEnumerableERC20.sol
- LSSVMPairEnumerableETH.sol
- LSSVMPairFactory.sol
- LSSVMPairFactoryLike.sol
- LSSVMPairMissingEnumerable.sol
- LSSVMPairMissingEnumerableERC20.sol
- LSSVMPairMissingEnumerableETH.sol
- LSSVMRouter.sol

The fixes were provided in the new commit.

2.1 About ABDK

ABDK Consulting, established in 2016, is a leading service provider in the space of blockchain development and audit. It has contributed to numerous blockchain projects, and co-authored some widely known blockchain primitives like Poseidon hash function. The ABDK Audit Team, led by Mikhail Vladimirov and Dmitry Khovratovich, has conducted over 40 audits of blockchain projects in Solidity, Rust, Circom, C++, JavaScript, and other languages.

2.2 Disclaimer

Note that the performed audit represents current best practices and smart contract standards which are relevant at the date of publication. After fixing the indicated issues the smart contracts should be re-audited.

2.3 Methodology

The methodology is not a strict formal procedure, but rather a collection of methods and tactics that combined differently and tuned for every particular project, depending on the project structure and and used technologies, as well as on what the client is expecting from the audit. In current audit we use:

- **General Code Assessment**. The code is reviewed for clarity, consistency, style, and for whether it follows code best practices applicable to the particular programming language used. We check indentation, naming convention, commented code blocks, code duplication, confusing names, confusing, irrelevant, or missing comments etc. At this phase we also understand overall code structure.
- **Entity Usage Analysis**. Usages of various entities defined in the code are analysed. This includes both: internal usages from other parts of the code as well as potential external usages. We check that entities are defined in proper places and that their visibility scopes and access levels are relevant. At this phase we understand overall system architecture and how different parts of the code are related to each other.
- **Access Control Analysis**. For those entities, that could be accessed externally, access control measures are analysed. We check that access control is relevant and is done properly. At this phase we understand user roles and permissions, as well as what assets the system ought to protect.
- **Code Logic Analysis**. The code logic of particular functions is analysed for correctness and efficiency. We check that code actually does what it is supposed to do, that algorithms are optimal and correct, and that proper data types are used. We also check that external libraries used in the code are up to date and relevant to the tasks they solve in the code. At this phase we also understand data structures used and the purposes they are used for.

3 Detailed Results

3.1 CVF-1

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** LSSVMPairFactory.sol

Description Other contracts in this project use "Ownable" contract from the "lib" directory, while this contract uses the OpenZeppelin's "Ownable" implementation.
Recommendation Consider using the same implementation everywhere.
Client Comment Acknowledged, but no change.

Listing 1:

```
4 import {Ownable} from "@openzeppelin/contracts/access/Ownable.
    ↪ sol";
```

3.2 CVF-2

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** LSSVMPairFactory.sol

Description We didn't review these files.
Client Comment Acknowledged

Listing 2:

```
11 import {ERC20} from "solmate/tokens/ERC20.sol";
   import {SafeTransferLib} from "solmate/utils/SafeTransferLib.sol
    ↪ ";
```

3.3 CVF-3

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** LSSVMPairFactory.sol

Recommendation Consider deriving this value from the "MAX_FEE" constant, or vice versa.
Client Comment Acknowledged.

Listing 3:

```
30 uint256 internal constant MAX_PROTOCOL_FEE = 0.10e18; // 10%,
    ↪ must <= 1 - MAX_FEE
```

3.4 CVF-4

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** LSSVMPairFactory.sol

Recommendation The type of this variable should be "LSSVMPairEnumerableETH".

Listing 4:

```
32 LSSVMPairETH public immutable enumerableETHTemplate;
```

3.5 CVF-5

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** LSSVMPairFactory.sol

Recommendation The type of this variable should be "LSSVMPairMissingEnumerableETH".

Listing 5:

```
33 LSSVMPairETH public immutable missingEnumerableETHTemplate;
```

3.6 CVF-6

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** LSSVMPairFactory.sol

Recommendation The type of this variable should be "LSSVMPairEnumerableERC20".

Listing 6:

```
34 LSSVMPairERC20 public immutable enumerableERC20Template;
```

3.7 CVF-7

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** LSSVMPairFactory.sol

Recommendation The type of this variable should be "LSSVMPairMissingEnumerableERC20".

Listing 7:

```
35 LSSVMPairERC20 public immutable missingEnumerableERC20Template;
```

3.8 CVF-8

- **Severity** Minor
- **Status** Fixed
- **Category** Documentation
- **Source** LSSVMPairFactory.sol

Description The number format of this variable is unclear.
Recommendation Consider documenting.

Listing 8:

```
37 uint256 public override protocolFeeMultiplier;
```

3.9 CVF-9

- **Severity** Minor
- **Status** Fixed
- **Category** Bad naming
- **Source** LSSVMPairFactory.sol

Recommendation Events are usually named via nouns, such as "NewPool".

Listing 9:

```
47 event PairCreated(address poolAddress, address nft);
```

3.10 CVF-10

- **Severity** Minor
- **Status** Info
- **Category** Readability
- **Source** LSSVMPairFactory.sol

Recommendation Both parameters should be indexed.
Client Comment Acknowledged, indexers notified of new pool creation can query on-chain for more parameters.

Listing 10:

```
47 event PairCreated(address poolAddress, address nft);
```

3.11 CVF-11

- **Severity** Minor
- **Status** Info
- **Category** Readability
- **Source** LSSVMPairFactory.sol

Recommendation This event should have more parameters describing the pair, such as pool type, pair variant, token, fee, delta, curve etc.
Client Comment Acknowledged, indexers notified of new pool creation can query on-chain for more parameters.

Listing 11:

```
47 event PairCreated(address poolAddress, address nft);
```

3.12 CVF-12

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** LSSVMPairFactory.sol

Recommendation The type of this argument should be "LSSVMPairEnumerableETH".

Listing 12:

```
50 LSSVMPairETH _enumerableETHTemplate,
```

3.13 CVF-13

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** LSSVMPairFactory.sol

Recommendation The type of this argument should be "LSSVMPairMissingEnumerableETH".

Listing 13:

```
51 LSSVMPairETH _missingEnumerableETHTemplate,
```

3.14 CVF-14

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** LSSVMPairFactory.sol

Recommendation The type of this argument should be "LSSVMPairEnumerableERC20".

Listing 14:

```
52 LSSVMPairERC20 _enumerableERC20Template,
```

3.15 CVF-15

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** LSSVMPairFactory.sol

Recommendation The type of this argument should be "LSSVMPairMissingEnumerableERC20".

Listing 15:

```
53 LSSVMPairERC20 _missingEnumerableERC20Template,
```

3.16 CVF-16

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LSSVMPairFactory.sol

Description These checks are redundant. It is anyway possible to pass dead or invalid addresses here. Checking that the template addresses belong to contracts would make a bit more sense, but still would not guarantee anything.

Listing 16:

```
58     address(_enumerableETHTemplate) != address(0),
64     address(_missingEnumerableETHTemplate) != address(0),
70     address(_enumerableERC20Template) != address(0),
76     address(_missingEnumerableERC20Template) != address(0),
81 require(_protocolFeeRecipient != address(0), "0 recipient
     ↪ address");
```

3.17 CVF-17

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** LSSVMPairFactory.sol

Recommendation Consider using different error messages.
Client Comment Removed.

Listing 17:

```
59 "0 template address"

65 "0 template address"

71 "0 template address"

77 "0 template address"
```

3.18 CVF-18

- **Severity** Minor
- **Status** Info
- **Category** Flaw
- **Source** LSSVMPairFactory.sol

Description There are no range checks for these arguments.
Recommendation Consider adding appropriate checks.
Client Comment Users should choose appropriate ranges.

Listing 18:

```
112 uint256 _delta,
    uint256 _fee,
    uint256 _spotPrice,

174 uint256 delta;
    uint256 fee;
    uint256 spotPrice;
```

3.19 CVF-19

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** LSSVMPairFactory.sol

Description ERC-165 defines the standard way to know whether a contract implements an interface.
Recommendation Consider using it instead of a custom approach.
Client Comment Acknowledged, but no change.

Listing 19:

```
229 function isPair(address potentialPair, PairVariant variant)
```

3.20 CVF-20

- **Severity** Moderate
- **Status** Fixed
- **Category** Flaw
- **Source** LSSVMPairFactory.sol

Description This function doesn't allow specifying the token amount to be withdrawn and always tries to transfer the whole balance. Such approach could be incompatible with tokens that charge a transfer fee on top of the transfer amount.
Recommendation Consider adding an "amount" argument to the function.

Listing 20:

```
290 function withdrawERC20ProtocolFees(ERC20 token) external
    ↪ onlyOwner {
```

3.21 CVF-21

- **Severity** Minor
- **Status** Fixed
- **Category** Overflow/Underflow
- **Source** LSSVMPairFactory.sol

Description These functions should emit some events.

Listing 21:

```
301 function changeProtocolFeeRecipient(address payable
    ↪ _protocolFeeRecipient)

313 function changeProtocolFeeMultiplier(uint256
    ↪ _protocolFeeMultiplier)

326 function setBondingCurveAllowed(ICurve bondingCurve, bool
    ↪ isAllowed)

339 function setCallAllowed(address payable target, bool isAllowed)

359 function setRouterAllowed(LSSVMRouter _router, bool isAllowed)
```

3.22 CVF-22

- **Severity** Minor
- **Status** Info
- **Category** Flaw
- **Source** LSSVMPairFactory.sol

Recommendation This field should be set to true only when "isAllowed" is true, otherwise should be left unchanged.
Client Comment Acknowledged that it is semantically weird (e.g. if we set a pool to be disallowed, wasEverAllowed is still true), but does not impact the functionality (it's meant to safeguard against gov maliciously allowing/disallowing a pool that was once a router).

Listing 22:

```
369 wasEverAllowed: true
```

3.23 CVF-23

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** LSSVMPairFactory.sol

Description These functions are very similar.
Recommendation Consider merging them together of doing some other refactoring to reduce code duplication.
Client Comment Acknowledged.

Listing 23:

```
377 function _initializePairETH(

402 function _initializePairERC20(
```

3.24 CVF-24

- **Severity** Minor
- **Status** Fixed
- **Category** Documentation
- **Source** LSSVMPairFactory.sol

Recommendation The word "initial" is misleading here, as this function could be used to deposit NFTs at any time.

Listing 24:

```
441 // transfer initial NFTs from caller to recipient
```

3.25 CVF-25

- **Severity** Major
- **Status** Info
- **Category** Overflow/Underflow
- **Source** LSSVMPairMissingEnumerable.sol

Description Underflow is possible here.
Recommendation Consider checking that "numNFTs" doesn't exceed the total number of NFTs owned by the pair.
Client Comment Already checked in LSSVMPair::swapTokenForAnyNFTs.

Listing 25:

```
32 uint256 nftId = idSet.at(lastIndex--);
```

3.26 CVF-26

- **Severity** Minor
- **Status** Info
- **Category** Flaw
- **Source** LSSVMPairMissingEnumerable.sol

Description This function permits receiving arbitrary NFTs, not only NTFs handled by the "_nft" contract.
Recommendation Consider forbidding reception of unknown NFTs.
Client Comment Pool operator can always withdraw other NFTs sent into the pool, and this enables some edge cases (e.g. the pool operator minting NFTs from the pool contract if e.g. they are whitelisted to mint some NFT).

Listing 26:

```
68 function onERC721Received(
```

3.27 CVF-27

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** LSSVMRouter.sol

Description We didn't review these files.
Client Comment Acknowledged.

Listing 27:

```
5 import {ERC20} from "solmate/tokens/ERC20.sol";
  import {SafeTransferLib} from "solmate/utils/SafeTransferLib.sol
    ↪ ";
```

3.28 CVF-28

- **Severity** Moderate
- **Status** Info
- **Category** Unclear behavior
- **Source** LSSVMRouter.sol

Description The "inputAmount" argument and the "remainingValue" retruned value don't make sense in case different individual swaps use different ERC20 tokens.
Recommendation Consider passing the input amounts separately for each individual swap and returning the used amounts per swap.
Client Comment Acknowledged, however no change in existing router for now. Current version of router is intended for 1 token type at a time.

Listing 28:

```
202     uint256 inputAmount,

205 ) external checkDeadline(deadline) returns (uint256
    ↪ remainingValue) {

219     uint256 inputAmount,

222 ) external checkDeadline(deadline) returns (uint256
    ↪ remainingValue) {

732     uint256 inputAmount,

734 ) internal returns (uint256 remainingValue) {

764     uint256 inputAmount,

766 ) internal returns (uint256 remainingValue) {
```

3.29 CVF-29

- **Severity** Moderate
- **Status** Info
- **Category** Suboptimal
- **Source** LSSVMRouter.sol

Description The "minOutput" argument and the "outputAmount" returned value don't make sense in case different individual swaps use different tokens.
Recommendation Consider specifying the minimum output and returned the actual output separately for each individual swap.
Client Comment Acknowledged, however no change in existing router for now. Current version of router is intended for 1 token type at a time.

Listing 29:

```
236     uint256 minOutput,

239 ) external checkDeadline(deadline) returns (uint256 outputAmount
    ↪ ) {

794     uint256 minOutput,

796 ) internal returns (uint256 outputAmount) {
```

3.30 CVF-30

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** LSSVMRouter.sol

Description These functions do make sense only when all the swap use the same token.
Recommendation Consider enforcing this.
Client Comment Acknowledged, however no change in existing router for now.

Listing 30:

```
253 function swapNFTsForAnyNFTsThroughERC20(

292 function swapNFTsForSpecificNFTsThroughERC20(
```

3.31 CVF-31

- **Severity** Moderate
- **Status** Fixed
- **Category** Flaw
- **Source** LSSVMRouter.sol

Recommendation The "minOutput" value should be added to the final "outputAmount" value.

Listing 31:

```
275 outputAmount = _swapERC20ForAnyNFTs(
        trade.tokenToNFTTrades,
        outputAmount - minOutput,
        nftRecipient
    );

314 outputAmount = _swapERC20ForSpecificNFTs(
        trade.tokenToNFTTrades,
        outputAmount - minOutput,
        nftRecipient
    );
```

3.32 CVF-32

- **Severity** Minor
- **Status** Info
- **Category** Unclear behavior
- **Source** LSSVMRouter.sol

Description These functions should return a bit mask or a bool array telling which swaps were actually performed. Alternatively it may return an array telling the exact ETH amount used for each swap (zero means the swap was skipped).
Client Comment Acknowledged, however no change in existing router for now.

Listing 32:

```
337 function robustSwapETHForAnyNFTs(

388 function robustSwapETHForSpecificNFTs(
```

3.33 CVF-33

- **Severity** Moderate
- **Status** Fixed
- **Category** Flaw
- **Source** LSSVMRouter.sol

Description There is no check to ensure that the lengths of these arrays are the same. If the "maxCostPerPairSwap" array is longer than the "swapList" array, extra element are silently ignored.
Recommendation Consider adding appropriate check.
Client Comment Fixed by merging maxCost/minOutput into the struct.

Listing 33:

```
338 PairSwapAny[] calldata swapList,
    uint256[] memory maxCostPerPairSwap,

389 PairSwapSpecific[] calldata swapList,
390 uint256[] memory maxCostPerPairSwap,

441 PairSwapAny[] calldata swapList,

443 uint256[] memory maxCostPerPairSwap,

483 PairSwapSpecific[] calldata swapList,

485 uint256[] memory maxCostPerPairSwap,

529 PairSwapSpecific[] calldata swapList,
530 uint256[] memory minOutputPerSwapPair,
```

3.34 CVF-34

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LSSVMRouter.sol

Recommendation It would be more efficient to pass a single array of structs with two fields rather than two parallel arrays. Such approach would also make the length check unnecessary.

Listing 34:

```
338 PairSwapAny[] calldata swapList,
    uint256[] memory maxCostPerPairSwap,

389 PairSwapSpecific[] calldata swapList,
390 uint256[] memory maxCostPerPairSwap,

441 PairSwapAny[] calldata swapList,

443 uint256[] memory maxCostPerPairSwap,

483 PairSwapSpecific[] calldata swapList,

485 uint256[] memory maxCostPerPairSwap,

529 PairSwapSpecific[] calldata swapList,
530 uint256[] memory minOutputPerSwapPair,
```

3.35 CVF-35

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** LSSVMRouter.sol

Description These functions are very similar.
Recommendation Consider extracting the common parts into utility functions to avoid code duplication.
Client Comment Acknowledged.

Listing 35:

```
337 function robustSwapETHForAnyNFTs(

388 function robustSwapETHForSpecificNFTs(
```

3.36 CVF-36

- **Severity** Minor
- **Status** Info
- **Category** Unclear behavior
- **Source** LSSVMRouter.sol

Description These functions should return a bit mask or a bool array telling which swaps were actually performed. Alternatively it may return an array telling the exact token amount used for each swap (zero means the swap was skipped).
Client Comment Acknowledged, however no change in existing router for now.

Listing 36:

```
440 function robustSwapERC20ForAnyNFTs(

482 function robustSwapERC20ForSpecificNFTs(
```

3.37 CVF-37

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** LSSVMRouter.sol

Description These functions are very similar.
Recommendation Consider extracting the common parts into utility functions to avoid code duplication.
Client Comment Acknowledged

Listing 37:

```
440 function robustSwapERC20ForAnyNFTs(

482 function robustSwapERC20ForSpecificNFTs(
```

3.38 CVF-38

- **Severity** Major
- **Status** Info
- **Category** Suboptimal
- **Source** LSSVMRouter.sol

Description The "inputAmount" argument and the "remainingValue" retruned value don't make sense in case different ERC20 tokens are used for different swaps.
Recommendation Consider removing the "inputAmount" argument and returning separately the token amounts consumed by the individual swaps.
Client Comment Acknowledged, however no change in existing router for now. Current version of router is intended for 1 token type at a time.

Listing 38:

```
442     uint256 inputAmount,

446 ) external checkDeadline(deadline) returns (uint256
    ↪ remainingValue) {

484     uint256 inputAmount,

492     returns (uint256 remainingValue)
```

3.39 CVF-39

- **Severity** Minor
- **Status** Info
- **Category** Unclear behavior
- **Source** LSSVMRouter.sol

Recommendation This function should return a bit mask or a bool array telling which swaps were actually performed. Alternatively it may return an array telling the exact token amount output for each swap (zero means the swap was skipped).
Client Comment Acknowledged, however no change in existing router for now.

Listing 39:

```
528 function robustSwapNFTsForToken(
```

3.40 CVF-40

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** LSSVMRouter.sol

Description The "outputAmount" returned value doesn't make sense in case different swaps use different ERC20 tokens.
Recommendation Consider returning separately the output amounts for the individual swaps.
Client Comment Acknowledged, however no change in existing router for now.

Listing 40:

```
533 ) external checkDeadline(deadline) returns (uint256 outputAmount
    ↪ ) {
```

3.41 CVF-41

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** LSSVMRouter.sol

Description These calls are redundant.
Recommendation Just pass all the remaining value to the swaps.
Client Comment This is intentional. In gas profiling, pre-calculating the amount to send for ETH is actually cheaper, gas-wise, as we don't have to spend gas doing another transfer back to the caller of leftover tokens.

Listing 41:

```
657 (error, , pairCost, ) = swapList[i].pair.getBuyNFTQuote(
        swapList[i].numItems
    );

699 (error, , pairCost, ) = swapList[i].pair.getBuyNFTQuote(
700     swapList[i].nftIds.length
    );
```

3.42 CVF-42

- **Severity** Minor
- **Status** Info
- **Category** Readability
- **Source** LSSVMRouter.sol

Description Using underflow checks to enforce business-level constraints is a bad practice, as it makes the code harder to read.
Recommendation Consider using an explicit "require" statement.
Client Comment Acknowledged, no change.

Listing 42:

```
664 // Total ETH taken from sender cannot exceed inputAmount
    // because otherwise the deduction from remainingValue will fail
```

3.43 CVF-43

- **Severity** Minor
- **Status** Fixed
- **Category** Bad naming
- **Source** LSSVMPairFactoryLike.sol

Description Names of some other interfaces in this project use the "I" prefix,but not the name of this interface.
Recommendation Consider using a consistent naming policy.

Listing 43:

```
6 interface LSSVMPairFactoryLike {
```

3.44 CVF-44

- **Severity** Minor
- **Status** Fixed
- **Category** Documentation
- **Source** LSSVMPairFactoryLike.sol

Description The number format of the returned value is unclear.
Recommendation Consider documenting.

Listing 44:

```
14 function protocolFeeMultiplier() external view returns (uint256)
    ↪ ;
```

3.45 CVF-45

- **Severity** Minor
- **Status** Fixed
- **Category** Documentation
- **Source** LSSVMPair.sol

Description The number format of these variables in unclear.
Recommendation Consider documenting.

Listing 45:

```
29 uint128 public spotPrice;

35 uint96 public fee;
```

3.46 CVF-46

- **Severity** Minor
- **Status** Fixed
- **Category** Bad naming
- **Source** LSSVMPair.sol

Recommendation Events are usually named via nouns, such as "SpotPriceUpdate" or "SpotPrice", "TokenUpdate", etc.

Listing 46:

```
52 event SpotPriceUpdated(uint128 newSpotPrice);
   event TokenDeposited(uint256 amount);
   event TokenWithdrawn(uint256 amount);
   event DeltaUpdated(uint128 newDelta);
   event FeeUpdated(uint96 newFee);
```

3.47 CVF-47

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** LSSVMPair.sol

Recommendation The type of this argument should be "uint96".

Listing 47:

```
73 uint256 _fee,
```

3.48 CVF-48

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** LSSVMPair.sol

Description This check relies on a feature of the "Ownable" smart contract that forbids zero owner address.
Recommendation Consider preventing double initialization in a more generic and more self-contained way. For example use a separate "initialized" flag.
Client Comment Acknowledged, but no change.

Listing 48:

```
76 require(owner() == address(0), "Initialized");
```

3.49 CVF-49

- **Severity** Minor
- **Status** Fixed
- **Category** Readability
- **Source** LSSVMPair.sol

Recommendation Should be "else if" for readability.

Listing 49:

```
87 if (_poolType == PoolType.TRADE) {
```

3.50 CVF-50

- **Severity** Minor
- **Status** Fixed
- **Category** Flaw
- **Source** LSSVMPair.sol

Recommendation Should be "<=" according to the comment above.

Listing 50:

```
88     require(_fee < MAX_FEE, "Trade fee must be less than 90%");
```

```
654 require(newFee < MAX_FEE, "Trade fee must be less than 90%");
```

3.51 CVF-51

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** LSSVMPair.sol

Description These functions are very similar.
Recommendation Consider extracting common parts into utility functions to avoid code duplication.
Client Comment Acknowledged.

Listing 51:

```
122 function swapTokenForAnyNFTs(

193 function swapTokenForSpecificNFTs(
```

3.52 CVF-52

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** LSSVMPair.sol

Recommendation Declarations of the "error" and "newSpotPrice" variables could be moved into the parenthesis.
Client Comment Acknowledged, no change.

Listing 52:

```
148 CurveErrorCodes.Error error;
    uint256 newSpotPrice;
150 (error, newSpotPrice, inputAmount, protocolFee) = _bondingCurve
```

3.53 CVF-53

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** LSSVMPair.sol

Description The actual error code is lost here.
Recommendation Consider returning it inside the revert message.

Listing 53:

```
158 require(error == CurveErrorCodes.Error.OK, "Bonding curve error
    ↪ ");
```

3.54 CVF-54

- **Severity** Moderate
- **Status** Fixed
- **Category** Overflow/Underflow
- **Source** LSSVMPair.sol

Description Overflow is possible here.
Recommendation Consider using the "uint256" type for spot price.

Listing 54:

```
161 spotPrice = uint128(newSpotPrice);
    emit SpotPriceUpdated(uint128(newSpotPrice));

233 spotPrice = uint128(newSpotPrice);
    emit SpotPriceUpdated(uint128(newSpotPrice));

298 spotPrice = uint128(newSpotPrice);
    emit SpotPriceUpdated(uint128(newSpotPrice));
```

3.55 CVF-55

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LSSVMPair.sol

Description This check is redundant, as it is anyway possible to specify NFTs not owned by the contract.
Recommendation Consider removing this check.

Listing 55:

```
212 (nftIds.length <= _nft.balanceOf(address(this))),
```

3.56 CVF-56

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** LSSVMPair.sol

Description This function doesn't scale.
Recommendation Consider implementing an ability to query NFT IDs list in chunks.
Client Comment This function should only be called off-chain.

Listing 56:

```
372 function getAllHeldIds() external view virtual returns (uint256
    ↪ [] memory);
```

3.57 CVF-57

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** LSSVMPair.sol

Recommendation The type of the "a" argument should be "IERC721".

Listing 57:

```
606 function withdrawERC721(address a, uint256[] calldata nftIds)
```

3.58 CVF-58

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** LSSVMPair.sol

Recommendation The type of the "a" argument should be "IERC20".

Listing 58:

```
615 function withdrawERC20(address a, uint256 amount) external
    ↪ virtual;
```

3.59 CVF-59

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LSSVMPair.sol

Description These events are emitted even if nothing actually changed.

Listing 59:

```
628 emit SpotPriceUpdated(newSpotPrice);

642 emit DeltaUpdated(newDelta);

656 emit FeeUpdated(newFee);
```

3.60 CVF-60

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LSSVMPair.sol

Description These operations should be performed only if newdelta!=delta.

Listing 60:

```
636 ICurve _bondingCurve = bondingCurve();
    require(
        _bondingCurve.validateDelta(newDelta),
        "Invalid delta for curve"
640 );
    delta = newDelta;
```

3.61 CVF-61

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LSSVMPair.sol

Description These operations should be performed only if fee != newFee.

Listing 61:

```
652 PoolType _poolType = poolType();
    require(_poolType == PoolType.TRADE, "Only for Trade pools");
    require(newFee < MAX_FEE, "Trade fee must be less than 90%");
    fee = newFee;
```

3.62 CVF-62

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** LSSVMPair.sol

Description The returned data is ignored.
Recommendation Consider forwarding the returned data to the called.
Client Comment Acknowledged. The arbitrary call is intended for situations like allowing pool owners to perform various actions that check NFT ownership, not for full composability, i.e. the intent is for EOAs to be making the call, not contracts.

Listing 62:

```
684 (bool result, ) = target.call{value: 0}(data);
```

3.63 CVF-63

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LSSVMPair.sol

Description This variable is not used.
Recommendation Consider removing it.

Listing 63:

```
692 uint256 public unlockTime;
```

3.64 CVF-64

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LSSVMPair.sol

Description This function does nothing.
Recommendation Consider removing it.

Listing 64:

```
694 function lockPool(uint256) external {}
```

3.65 CVF-65

- **Severity** Minor
- **Status** Fixed
- **Category** Unclear behavior
- **Source** LSSVMPairETH.sol

Description This check should be done after rounding the protocol fee down to the actual ETH balance of the contract. Otherwise it is still possible to initiate a zero ether transfer.

Listing 65:

```
36 if (protocolFee > 0) {
```

```
59 if (protocolFee > 0) {
```

3.66 CVF-66

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** LSSVMPairETH.sol

Recommendation This value should be a named constant.

Listing 66:

```
81 return 61;
```

3.67 CVF-67

- **Severity** Minor
- **Status** Fixed
- **Category** Documentation
- **Source** LSSVMPairETH.sol

Recommendation Consider explaining in a comment how this value was calculated.

Listing 67:

```
81 return 61;
```

3.68 CVF-68

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** LSSVMPairETH.sol

Description In the former case the recipient is "owner()" while in the latter case it is "msg.sender". These two values guaranteed to be the same.
Recommendation Consider using a consistent approach to improve readability.

Listing 68:

```
99 payable(owner()).safeTransferETH(amount);
```

```
111 ERC20(a).safeTransfer(msg.sender, amount);
```

3.69 CVF-69

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** LSSVMPairERC20.sol

Description We didn't review this file.
Client Comment Acknowledged.

Listing 69:

```
5 import {ERC20} from "solmate/tokens/ERC20.sol";
```

3.70 CVF-70

- **Severity** Moderate
- **Status** Info
- **Category** Flaw
- **Source** LSSVMPairERC20.sol

Description This check is not perfect. It will not work in case "_assetRecipient" is a contract that handles incoming token transfers via a callback (such as defined in ERC-777) and forwards them somewhere. Also, there could be ways to increase the token balance of "_assetRecipient" without sending tokens to it. For example, lets consider a situation when "_assetRecipient" participates in some DeFi activity and is eligible to receive some reward tokens from it. Many DeFi allows anybody to initiate reward claim. Such operation would increase the token balance of the reward recipient.
Recommendation Consider allowing only trusted routers and transferring tokens through this contract, i.e.: 1. The router transfers tokens to this contract 2. This contract verifies token reception 3. This contract transfers tokens to the ultimate recipient.
Client Comment Risks acknowledged.

Listing 70:

```
68 // Verify token transfer (protect pair against malicious router)
   require(
70     _token.balanceOf(_assetRecipient) - beforeBalance ==
         inputAmount - protocolFee,
       "ERC20 not transferred in"
   );
```

3.71 CVF-71

- **Severity** Moderate
- **Status** Fixed
- **Category** Suboptimal
- **Source** LSSVMPairERC20.sol

Description Checking the NFT balance doesn't prevent reentrancy as a reentrant call, may not move any NFTs at all or may preserve the total NFT balance.
Recommendation Consider using a more straightforward way to protect against reentrancy, such as "ReentrancyGuard" from OpenZeppelin.
Client Comment ReentrancyGuard is now used.

Listing 71:

```
86 // Check NFT balance again to prevent against reentrancy
```

3.72 CVF-72

- **Severity** Major
- **Status** Fixed
- **Category** Flaw
- **Source** LSSVMPairERC20.sol

Description This check should be done after rounding the protocol fee down to the actual token balance. Otherwise it is still possible to initialize a transfer of zero tokens.
Client Comment Added additional check that protocolFee is nonzero after rounding down.

Listing 72:

```
121 if (protocolFee > 0) {
```

3.73 CVF-73

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** LSSVMPairERC20.sol

Recommendation This value should be a named constant.

Listing 73:

```
146 return 81;
```

3.74 CVF-74

- **Severity** Minor
- **Status** Fixed
- **Category** Documentation
- **Source** LSSVMPairERC20.sol

Recommendation Consider explaining in a comment how this value was calculated.

Listing 74:

```
146 return 81;
```

3.75 CVF-75

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** LSSVMPairERC20.sol

Description Why only the pair token withdrawals are logged?
Recommendation Consider logging all withdrawals and including the token address as an event parameter.
Client Comment Only pair tokens would affect relevant balances for swap operations.

Listing 75:

```
158 // emit event since it is the pair token
    emit TokenWithdrawn(amount);
```

3.76 CVF-76

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** Ownable.sol

Description This parameter is redundant as its value could be derived from the previous events.
Client Comment Acknowledged.

Listing 76:

```
12 address indexed previousOwner,
```

3.77 CVF-77

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** Ownable.sol

Description Transferring ownership to the zero address is a common way to denounce ownership.
Recommendation Consider not forbidding this.
Client Comment As a minor gas optimization, we use checking owner to be address(0) to know if the pool has already been initialized. Users wishing to renounce ownership can use something like 0x0...dead as an alternative.

Listing 77:

```
35 if (newOwner == address(0)) revert Ownable_NewOwnerZeroAddress()
   ↪ ;
```

3.78 CVF-78

- **Severity** Minor
- **Status** Info
- **Category** Unclear behavior
- **Source** Ownable.sol

Description This event is emitted even if the owner didn't actually change.
Client Comment Acknowledged.

Listing 78:

```
44 emit OwnershipTransferred(oldOwner, newOwner);
```

3.79 CVF-79

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** ExponentialCurve.sol

Description We didn't review this file.
Client Comment Acknowledged.

Listing 79:

```
6 import {FixedPointMathLib} from "solmate/utils/FixedPointMathLib
    ↪ .sol";
```

3.80 CVF-80

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** ExponentialCurve.sol

Description A comment in "ICurve;sol" says that for exponential curve, delta has to be "at least 1", which meant delta >= 1, while here the condition is delta > 1.
Recommendation Consider making the actual condition and the comment consistent with each other.

Listing 80:

```
27 return delta > FixedPointMathLib.WAD;
```

3.81 CVF-81

- **Severity** Minor
- **Status** Info
- **Category** Unclear behavior
- **Source** ExponentialCurve.sol

Description All these expressions seem to round the result down, i.e. towards the user.
Recommendation A common best practice is to round towards the protocol to prevent abusing rounding errors.
Client Comment Acknowledged, no change.

Listing 81:

```
68 uint256 deltaPowN = delta.fpow(numItems, FixedPointMathLib.WAD);

71 newSpotPrice = spotPrice.fmul(deltaPowN, FixedPointMathLib.WAD);

79 uint256 buySpotPrice = spotPrice.fmul(delta, FixedPointMathLib.
   ↪ WAD);

84 inputValue = buySpotPrice.fmul(

93 protocolFee = inputValue.fmul(

99 inputValue += inputValue.fmul(feeMultiplier, FixedPointMathLib.
   ↪ WAD);
```

3.82 CVF-82

- **Severity** Minor
- **Status** Info
- **Category** Readability
- **Source** ExponentialCurve.sol

Description These fixed-point operations always take the same second argument.
Recommendation Consider wrapping such calls into single-argument function for readability.
Client Comment Acknowledged.

Listing 82:

```
68 uint256 deltaPowN = delta.fpow(numItems, FixedPointMathLib.WAD);
```

3.83 CVF-83

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** ExponentialCurve.sol

Description Calculating the new spot price by adjusting the previous spot price may accumulate errors.
Recommendation Better way would be to calculate the spot price from the initial spot price, the total number of tokens sold since that time and the total number of tokens bought since that time. Such approach would require changing the "ICurve" API.
Client Comment Acknowledged.

Listing 83:

```
71  newSpotPrice = spotPrice.fmul(deltaPowN, FixedPointMathLib.WAD);

143 newSpotPrice = spotPrice.fmul(invDeltaPowN, FixedPointMathLib.
    ↪ WAD);
```

3.84 CVF-84

- **Severity** Minor
- **Status** Fixed
- **Category** Documentation
- **Source** ExponentialCurve.sol

Description The code implicitly assumes that delta is greater than 1.
Recommendation Consider making an explicit assert about it.

Listing 84:

```
81  // If the user buys n items, then the total cost is equal to:
    // buySpotPrice + (delta * buySpotPrice) + (delta^2 *
       ↪ buySpotPrice) + ... (delta^(numItems - 1) * buySpotPrice)
    // This is equal to buySpotPrice * (delta^n - 1) / (delta - 1)
    inputValue = buySpotPrice.fmul(
        (deltaPowN - FixedPointMathLib.WAD).fdiv(
            delta - FixedPointMathLib.WAD,
```

3.85 CVF-85

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** ExponentialCurve.sol

Description Values between 0 and 1 seem to loss precision faster when exponentiating, compared to bigger values.
Recommendation Consider exponentiating first, and then invert.
Client Comment Acknowledged.

Listing 85:

```
140 uint256 invDeltaPowN = invDelta.fpow(numItems, FixedPointMathLib
    ↪ .WAD);
```

3.86 CVF-86

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** ExponentialCurve.sol

Description This value is calculated rounding down, i.e.towards the user.
Recommendation A common best practice is to round towards the protocol to prevent abusing rounding errors.
Client Comment Acknowledged, no change.

Listing 86:

```
160 protocolFee = outputValue.fmul(
```

3.87 CVF-87

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** LSSVMPairCloner.sol

Description We didn't review this file.
Client Comment Acknowledged.

Listing 87:

```
7 import {ERC20} from "solmate/tokens/ERC20.sol";
```

3.88 CVF-88

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** LSSVMPairCloner.sol

Description These functions are very similar.
Recommendation Consider extracting the common parts into utility functions to avoid code duplication.
Client Comment Acknowledged.

Listing 88:

```
21  function cloneETHPair(

111 function cloneERC20Pair(
```

3.89 CVF-89

- **Severity** Minor
- **Status** Info
- **Category** Readability
- **Source** LSSVMPairCloner.sol

Description Most of the assembly code inside these assembly blocks could be rewritten in plain Solidity using "abi.encodePacked".
Recommendation Consider using as little assembly as possible to improve code readability.
Client Comment Acknowledged.

Listing 89:

```
28  assembly {

119 assembly {

209 assembly {

248 assembly {
```

3.90 CVF-90

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** LSSVMPairCloner.sol

Description It is unclear what "extra" means here.
Recommendation Consider explaining that it is just the extra data size.
Client Comment Acknowledged.

Listing 90:

```
58 // 60 extra     | PUSH1 extra            | extra 0 0 0 0
   ↪              | [0, cds) = calldata

63 // 60 extra     | PUSH1 extra            | extra cds 0 0 0 0
   ↪          | [0, cds) = calldata, [cds, cds+0x35) = extraData
```

3.91 CVF-91

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** LinearCurve.sol

Description We didn't review this file.
Client Comment Acknowledged.

Listing 91:

```
6 import {FixedPointMathLib} from "solmate/utils/FixedPointMathLib
  ↪ .sol";
```

3.92 CVF-92

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** LinearCurve.sol

Description These values are calculated rounding down, i.e. towards the user.
Recommendation A common best practice is to round towards the protocol to prevent abusing rounding errors.
Client Comment Acknowledged, no change.

Listing 92:

```
75 inputValue =

82 protocolFee = inputValue.fmul(

88 inputValue += inputValue.fmul(feeMultiplier, FixedPointMathLib.
  ↪ WAD);
```

3.93 CVF-93

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** LinearCurve.sol

Description Thee formulas are basically the same. The only difference is what variable is used as the number of items sold.
Recommendation Consider placing this calculation after the conditional statement, and setting "numItems" variable to the "numItemsTillZeroPrice" value in case the new spot price is zero.

Listing 93:

```
140 outputValue =
        numItemsTillZeroPrice *
        spotPrice -
        (numItemsTillZeroPrice * (numItemsTillZeroPrice - 1) * delta
            ↪ ) /
        2;

155 outputValue =
        numItems *
        spotPrice -
        (numItems * (numItems - 1) * delta) /
        2;
```

3.94 CVF-94

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** LinearCurve.sol

Description This value is rounded down, i.e. towards the user.
Recommendation A common best practice is to round towards the protocol to prevent abusing rounding errors.
Client Comment Acknowledged, no change.

Listing 94:

```
163 protocolFee = outputValue.fmul(
```

3.95 CVF-95

- **Severity** Minor
- **Status** Fixed
- **Category** Readability
- **Source** ICurve.sol

Recommendation Remove "is"

Listing 95:

```
9 validity is can be different for each type of curve, for
    ↪ instance ExponentialCurve
```

3.96 CVF-96

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** ICurve.sol

Description The "pure" modifier implies that the functions doesn't depend on the storage state.
Recommendation Consider changing to "view" to allow depending on the storage state.

Listing 96:

```
14 function validateDelta(uint256 delta) external pure returns (
    ↪ bool valid);

23     pure

47     pure

76     pure
```

3.97 CVF-97

- **Severity** Minor
- **Status** Fixed
- **Category** Documentation
- **Source** ICurve.sol

Description The number format and the semantics of the "delta" argument is unclear.
Recommendation Consider documenting.

Listing 97:

```
14 function validateDelta(uint256 delta) external pure returns (
    ↪ bool valid);

41     uint256 delta,

70     uint256 delta,
```

3.98 CVF-98

- **Severity** Minor
- **Status** Fixed
- **Category** Documentation
- **Source** ICurve.sol

Description The number format of a spot price is unclear.
Recommendation Consider documenting.

Listing 98:

```
21 function validateSpotPrice(uint256 newSpotPrice)

40     uint256 spotPrice,

50         uint256 newSpotPrice,

69     uint256 spotPrice,

79         uint256 newSpotPrice,
```

3.99 CVF-99

- **Severity** Minor
- **Status** Fixed
- **Category** Documentation
- **Source** ICurve.sol

Description The number format of fee multipliers is unclear.
Recommendation Consider documenting.

Listing 99:

```
43 uint256 feeMultiplier,
   uint256 protocolFeeMultiplier

72 uint256 feeMultiplier,
   uint256 protocolFeeMultiplier
```

3.100 CVF-100

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** ICurve.sol

Description Returning error codes is discouraged.
Recommendation Consider reverting on error.
Client Comment Acknowledged.

Listing 100:

```
49 CurveErrorCodes.Error error,

78 CurveErrorCodes.Error error,
```

3.101 CVF-101

- **Severity** Minor
- **Status** Fixed
- **Category** Documentation
- **Source** ICurve.sol

Description The number format of fees is unclear.
Recommendation Consider documenting.

Listing 101:

```
52 uint256 protocolFee

81 uint256 protocolFee
```

3.102 CVF-102

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** CurveErrorCodes.sol

Recommendation Consider using type-safe named errors instead of error codes enum. See documentation for details: https://docs.soliditylang.org/en/v0.8.12/structure-of-a-contract.html#errors
Client Comment Acknowledged, no change.

Listing 102:

```
5 enum Error {
```