ABDK CONSULTING

SMART CONTRACT AUDIT

DeFiedge

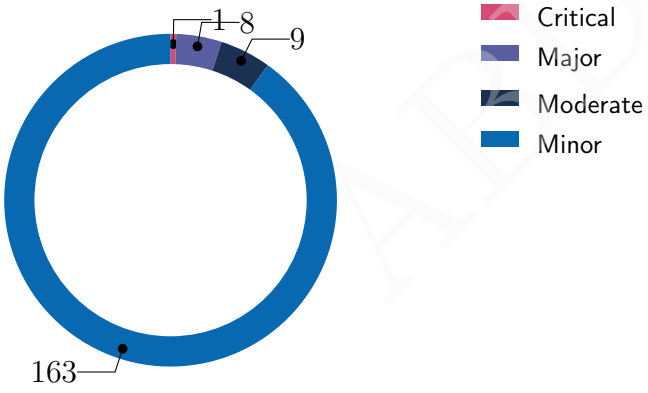
Solidity

abdk.consulting

SMART CONTRACT AUDIT CONCLUSION

by Mikhail Vladimirov and Dmitry Khovratovich
26th April 2022

We've been asked to review 17 files in a Github repository. We found 1 critical, 8 major, and a few less important issues. All critical and major fixes have been fixed.



Findings

ID	Severity	Category	Status
CVF-1	Minor	Procedural	Fixed
CVF-2	Minor	Readability	Fixed
CVF-3	Minor	Bad datatype	Fixed
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	Bad datatype	Fixed
CVF-9	Minor	Bad datatype	Fixed
CVF-10	Minor	Readability	Fixed
CVF-11	Minor	Suboptimal	Fixed
CVF-12	Minor	Overflow/Underflow	Fixed
CVF-13	Minor	Suboptimal	Fixed
CVF-14	Minor	Unclear behavior	Fixed
CVF-15	Minor	Suboptimal	Fixed
CVF-16	Minor	Procedural	Fixed
CVF-17	Minor	Procedural	Fixed
CVF-18	Minor	Bad datatype	Fixed
CVF-19	Minor	Bad datatype	Fixed
CVF-20	Moderate	Unclear behavior	Fixed
CVF-21	Minor	Suboptimal	Fixed
CVF-22	Minor	Suboptimal	Fixed
CVF-23	Minor	Suboptimal	Fixed
CVF-24	Minor	Suboptimal	Fixed
CVF-25	Minor	Procedural	Fixed
CVF-26	Minor	Bad datatype	Fixed
CVF-27	Minor	Bad datatype	Fixed

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

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

ID	Severity	Category	Status
CVF-88	Minor	Suboptimal	Info
CVF-89	Minor	Procedural	Info
CVF-90	Moderate	Overflow/Underflow	Fixed
CVF-91	Moderate	Flaw	Fixed
CVF-92	Minor	Suboptimal	Fixed
CVF-93	Minor	Bad datatype	Fixed
CVF-94	Minor	Suboptimal	Info
CVF-95	Minor	Suboptimal	Fixed
CVF-96	Minor	Suboptimal	Fixed
CVF-97	Minor	Overflow/Underflow	Fixed
CVF-98	Minor	Bad datatype	Fixed
CVF-99	Moderate	Overflow/Underflow	Fixed
CVF-100	Moderate	Flaw	Fixed
CVF-101	Minor	Suboptimal	Fixed
CVF-102	Minor	Suboptimal	Fixed
CVF-103	Minor	Overflow/Underflow	Fixed
CVF-104	Minor	Suboptimal	Fixed
CVF-105	Minor	Bad datatype	Fixed
CVF-106	Minor	Flaw	Fixed
CVF-107	Minor	Suboptimal	Info
CVF-108	Minor	Readability	Fixed
CVF-109	Minor	Unclear behavior	Info
CVF-110	Minor	Procedural	Fixed
CVF-111	Minor	Suboptimal	Fixed
CVF-112	Minor	Suboptimal	Fixed
CVF-113	Minor	Unclear behavior	Info
CVF-114	Minor	Bad datatype	Fixed
CVF-115	Minor	Bad naming	Fixed
CVF-116	Minor	Documentation	Fixed
CVF-117	Minor	Procedural	Fixed

ID	Severity	Category	Status
CVF-118	Minor	Bad datatype	Fixed
CVF-119	Minor	Suboptimal	Fixed
CVF-120	Minor	Suboptimal	Fixed
CVF-121	Minor	Suboptimal	Fixed
CVF-122	Minor	Bad datatype	Fixed
CVF-123	Minor	Suboptimal	Fixed
CVF-124	Minor	Unclear behavior	Fixed
CVF-125	Minor	Unclear behavior	Fixed
CVF-126	Minor	Unclear behavior	Fixed
CVF-127	Minor	Procedural	Fixed
CVF-128	Major	Unclear behavior	Fixed
CVF-129	Major	Suboptimal	Fixed
CVF-130	Minor	Suboptimal	Fixed
CVF-131	Minor	Suboptimal	Fixed
CVF-132	Minor	Procedural	Fixed
CVF-133	Minor	Bad datatype	Fixed
CVF-134	Minor	Bad datatype	Fixed
CVF-135	Minor	Bad datatype	Fixed
CVF-136	Minor	Procedural	Fixed
CVF-137	Minor	Suboptimal	Info
CVF-138	Minor	Unclear behavior	Fixed
CVF-139	Major	Suboptimal	Fixed
CVF-140	Minor	Bad naming	Fixed
CVF-141	Minor	Bad naming	Fixed
CVF-142	Minor	Bad datatype	Fixed
CVF-143	Minor	Procedural	Info
CVF-144	Minor	Suboptimal	Fixed
CVF-145	Major	Suboptimal	Fixed
CVF-146	Minor	Bad datatype	Fixed
CVF-147	Minor	Bad naming	Fixed

ID	Severity	Category	Status
CVF-148	Minor	Bad datatype	Fixed
CVF-149	Minor	Overflow/Underflow	Fixed
CVF-150	Minor	Bad datatype	Fixed
CVF-151	Minor	Suboptimal	Fixed
CVF-152	Major	Flaw	Fixed
CVF-153	Moderate	Unclear behavior	Fixed
CVF-154	Minor	Unclear behavior	Fixed
CVF-155	Minor	Suboptimal	Fixed
CVF-156	Minor	Suboptimal	Fixed
CVF-157	Minor	Suboptimal	Fixed
CVF-158	Critical	Flaw	Fixed
CVF-159	Minor	Suboptimal	Fixed
CVF-160	Minor	Suboptimal	Fixed
CVF-161	Major	Flaw	Fixed
CVF-162	Minor	Procedural	Fixed
CVF-163	Minor	Suboptimal	Fixed
CVF-164	Minor	Suboptimal	Fixed
CVF-165	Minor	Bad datatype	Info
CVF-166	Minor	Procedural	Fixed
CVF-167	Minor	Procedural	Fixed
CVF-168	Minor	Bad naming	Fixed
CVF-169	Minor	Bad datatype	Fixed
CVF-170	Minor	Readability	Fixed
CVF-171	Minor	Documentation	Fixed
CVF-172	Minor	Bad datatype	Fixed
CVF-173	Minor	Documentation	Fixed
CVF-174	Minor	Documentation	Fixed
CVF-175	Minor	Bad naming	Fixed
CVF-176	Minor	Bad datatype	Fixed
CVF-177	Minor	Bad datatype	Fixed

ID	Severity	Category	Status
CVF-178	Minor	Documentation	Fixed
CVF-179	Minor	Bad datatype	Fixed
CVF-180	Minor	Bad datatype	Fixed
CVF-181	Minor	Bad datatype	Fixed

Contents

1 Document properties

Version

Version	Date	Author	Description
0.1	April 25, 2022	D. Khovratovich	Initial Draft
0.2	April 26, 2022	D. Khovratovich	Minor revision
1.0	April 26, 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 9776438 commit:

- base/StrategyBase.sol
- base/StrategyManager.sol
- base/UniswapV3LiquidityManager.sol
- interfaces/IERC20.sol
- interfaces/IOneInch.sol
- interfaces/IStrategy.sol
- interfaces/IStrategyFactory.sol
- interfaces/IStrategyManager.sol
- libraries/DateTimeLibrary.sol
- libraries/LiquidityHelper.sol
- libraries/OneInchHelper.sol
- libraries/OracleLibrary.sol
- libraries/ShareHelper.sol
- DefiEdgeStrategy.sol
- DefiEdgeStrategyDeployer.sol
- DefiEdgeStrategyFactory.sol
- ERC20.sol

The fixes were provided in a 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** Fixed
- **Category** Procedural
- **Source** DefiEdgeStrategy.sol

Recommendation Should be "^0.7.0" or "^0.7.6" if there is something special about this particular version. Requiring a specific compiler version makes it harder to migrate to new compiler versions. Also relevant for the following files: OneInchHelper.sol, DefiEdgeStrategyFactory.sol, DefiEdgeStrategyDeployer.sol, LiquidityHelper.sol, ERC20.sol, ShareHelper.sol, OracleLibrary.sol, StrategyManager.sol, StrategyBase.sol, DateTimeLibrary.sol, UniswapV3LiquidityManager.sol, IERC20.sol, IStrategy.sol, IStrategyManager.sol, IStrategyFactory.sol.

Listing 1:

```
3 pragma solidity =0.7.6;
```

3.2 CVF-2

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

Recommendation The "user" parameter should be indexed.

Listing 2:

```
18 event Mint(address user, uint256 share, uint256 amount0, uint256
   ↪ amount1);
   event Burn(address user, uint256 share, uint256 amount0, uint256
   ↪ amount1);
```

3.3 CVF-3

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

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

Listing 3:

```
41 address _factory,
```

3.4 CVF-4

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

Recommendation The type of this argument should be "IUniswapV3Pool".

Listing 4:

```
42 address _pool,
```

3.5 CVF-5

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

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

Listing 5:

```
43 address _oneInchRouter,
```

3.6 CVF-6

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

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

Listing 6:

```
44 address _chainlinkRegistry,
```

3.7 CVF-7

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

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

Listing 7:

```
45 address _manager,
```

3.8 CVF-8

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

Recommendation The type of this argument should be "bool[2]".

Listing 8:

```
46 bool[] memory _usdAsBase,
```

3.9 CVF-9

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

Recommendation The value "5" should be a named constant.

Listing 9:

```
51  require(_ticks.length <= 5, "ITL");
285 require(ticks.length <= 5, "ITL");
```

3.10 CVF-10

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

Recommendation These assignments should be moved to the "else" branch of the subsequent conditional statement.

Listing 10:

```
92 amount0 = _amount0;
   amount1 = _amount1;
```

3.11 CVF-11

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

Description The expression "ticks[0]" is calculated several times.
Recommendation Consider calculating once and reusing.

Listing 11:

```
98      ticks[0].tickLower,
        ticks[0].tickUpper,

106 ticks[0].amount0 = ticks[0].amount0.add(amount0);
    ticks[0].amount1 = ticks[0].amount1.add(amount1);
```

3.12 CVF-12

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

Description Phantom overflow is possible here.
Recommendation Consider using the "mulDiv" function or some other approach resistant to phantom overflows.

Listing 12:

```
174 collect0 = collect0.mul(_shares).div(_totalSupply);

178 collect1 = collect1.mul(_shares).div(_totalSupply);
```

3.13 CVF-13

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

Recommendation The check is redundant, as the "for" loop inside will anyway do it implicitly.

Listing 13:

```
182 if (ticks.length != 0) {
```

3.14 CVF-14

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

Recommendation Should probably be zero instead of "tick.amountN".

Listing 14:

```
202 : tick.amount0;

205 : tick.amount1;
```

3.15 CVF-15

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

Description When "_burnAll" is true, "_existingTicks" contents doesn't matter.
Recommendation Consider requiring "existingTicks" to be empty in such a case.

Listing 15:

```
235 PartialTick[] memory _existingTicks,

237 bool _burnAll
```

3.16 CVF-16

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

Description This comment is confusing.
Recommendation Remove it or resolve.
Client Comment Removed the comment.

Listing 16:

```
313 // TODO: Make this function work correctly
```

3.17 CVF-17

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

Recommendation This commented out functions should be either removed or uncommented and fixed.
Client Comment Removed the comment.

Listing 17:

```
325 // // TODO: Make this function work correctly
    // function emergencyBurn(

343 // }
```

3.18 CVF-18

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

Recommendation The type of the token arguments should be 'IERC20".

Listing 18:

```
16 function decodeData(address token0, address token1, bytes
     ↪ calldata data)
```

3.19 CVF-19

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

Recommendation The type of the token returned values should be "IERC20".

Listing 19:

```
20 address srcToken,
   address dstToken,
```

3.20 CVF-20

- **Severity** Moderate
- **Status** Fixed
- **Category** Unclear behavior
- **Source** OneInchHelper.sol

Description Only the first byte of data is used to select function, while the other bytes are ignored.
Recommendation Consider using all selector four bytes. It is possible to efficiently extract selector from data: require (data.length >= 4); bytes4 selector; assembly { selector := mload (add (data, 0x20)); }

Listing 20:

```
26 if(data[0] == 0x7c) {
34 } else if(data[0] == 0x2e){
43 } else if(data[0] == 0xe4){
```

3.21 CVF-21

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

Description The "_amount" variable is redundant.
Recommendation Just use "amount" instead.

Listing 21:

```
45 (uint256 _amount, ,uint256[] memory pools) = abi.decode(
```

3.22 CVF-22

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

Recommendation This could be optimized as: bool zeroForOne == pools[0] » 255 == 0;

Listing 22:

```
50 bool zeroForOne = pools[0] & 1 << 255 == 0;
```

3.23 CVF-23

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

Description The expression "pools[0]" is calculated several times.
Recommendation Consider calculating once and reusing.

Listing 23:

```
50 bool zeroForOne = pools[0] & 1 << 255 == 0;

52 srcToken = zeroForOne ? IUniswapV3Pool(pools[0]).token0() :
     ↪ IUniswapV3Pool(pools[0]).token1();
```

3.24 CVF-24

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

Description If the function selector is not recognized, the function silently does nothing.
Recommendation Consider reverting in such a case.

Listing 24:

```
56 }
```

3.25 CVF-25

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

Recommendation This interface should be moved to a separate file named "IDefiEdgeStrategyDeployer.sol".

Listing 25:

```
10 interface IDefiEdgeStrategyDeployer {
```

3.26 CVF-26

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

Recommendation The types of these arguments should be "IStrategyFactory", "IUniswapV3Pool", "IOneInchRouter", "FeedRegistryInterface", and "IStrategyManager".

Listing 26:

```
12 address _factory,
address _pool,
address _swapRouter,
address _chainlinkRegistry,
address _manager,
```

3.27 CVF-27

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

Recommendation The type of the argument should be "bool[2]".

Listing 27:

```
17 bool[] memory _usdAsBase,
```

3.28 CVF-28

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** DefiEdgeStrategyFactory.sol

Recommendation The return type should be "DefiEdgeStrategy".
Client Comment It doesn't produce any vulnerability, for the other code dependecies we'll like to keep it as it is.

Listing 28:

```
19 ) external returns (address);
```

3.29 CVF-29

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

Description This contract should implement the "IStrategyFactory" interface. Otherwise compiler cannot check that the contract and the interface have the same API.

Listing 29:

```
22 contract DefiEdgeStrategyFactory {
```

3.30 CVF-30

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** DefiEdgeStrategyFactory.sol

Recommendation The type of the "strategy" parameter should be "DefiEdgeStrategy".
Client Comment It doesn't produce any vulnerability, for the other code dependecies we'll like to keep it as it is.

Listing 30:

```
25 event NewStrategy(address indexed strategy, address indexed
   ↪ creater);
```

3.31 CVF-31

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** DefiEdgeStrategyFactory.sol

Recommendation The value type should be "DefiEdgeStrategy".
Client Comment It doesn't produce any vulnerability, for the other code dependecies we'll like to keep it as it is.

Listing 31:

```
27 mapping(uint256 => address) public strategyByIndex; // map
   ↪ strategies by index
```

3.32 CVF-32

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** DefiEdgeStrategyFactory.sol

Recommendation The key type should be "DefiEdgeStrategy".
Client Comment It doesn't produce any vulnerability, for the other code dependecies we'll like to keep it as it is.

Listing 32:

```
28 mapping(address => bool) public isValid; // make strategy valid
     ↪ when deployed

55 mapping(address => bool) public denied;
```

3.33 CVF-33

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** DefiEdgeStrategyFactory.sol

Recommendation The key type should be "IStrategyManager", the value type should be "DefiEdgeStrategy".
Client Comment It doesn't produce any vulnerability, for the other code dependecies we'll like to keep it as it is.

Listing 33:

```
30 mapping(address => address) public strategyByManager; //
     ↪ strategy manager contracts linked with strategies
```

3.34 CVF-34

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

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

Listing 34:

```
48 address public deployerProxy;
```

3.35 CVF-35

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

Recommendation The type of this variable should be "IUniswapV3Factory".

Listing 35:

```
49 address public uniswapV3Factory; // Uniswap V3 pool factory
```

3.36 CVF-36

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

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

Listing 36:

```
50 address public chainlinkRegistry; // Chainlink registry
```

3.37 CVF-37

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

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

Listing 37:

```
52 address public oneInchRouter;
```

3.38 CVF-38

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

Recommendation The type of this field should be "IUniswapV3Pool".

Listing 38:

```
63 address pool;
```

3.39 CVF-39

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

Recommendation The type of this field should be "bool[2]" or a bit mask of two bits.

Listing 39:

```
64 bool[] usdAsBase;
```

3.40 CVF-40

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

Recommendation The types of these arguments should be: "IDefyEdgeStrageryDeployer", "FeedRegistryInterface", "IUniswapV3Factory", "IOneInchRouter".

Listing 40:

```
76 address _deployerProxy,
   address _chainlinkRegistry,
   address _uniswapV3factory,
   address _oneInchRouter,
```

3.41 CVF-41

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

Description There are no range checks for these arguments.
Recommendation Consider adding appropriate checks.

Listing 41:

```
80 uint256 _allowedSlippage,
   uint256 _allowedDeviation
```

3.42 CVF-42

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

Description This limitation looks arbitrary.
Recommendation Consider adding support for tokens with more decimals.
Client Comment We don't want to change it for now.

Listing 42:

```
103 IERC20Minimal(pool.token0()).decimals() <= 18 &&
        IERC20Minimal(pool.token1()).decimals() <= 18,
```

3.43 CVF-43

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

Recommendation The value "18" should be a named constant.

Listing 43:

```
103 IERC20Minimal(pool.token0()).decimals() <= 18 &&
        IERC20Minimal(pool.token1()).decimals() <= 18,
```

3.44 CVF-44

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

Description There are no range checks for the arguments.
Recommendation Consider adding appropriate checks.

Listing 44:

```
152 function changeDefaultAllowedDeviation(uint256 _allowedDeviation
    ↪ )

159 function changeAllowedSlippage(uint256 _allowedSlippage)

170 function changeFee(uint256 _fee) external onlyGovernance {
```

3.45 CVF-45

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

Description These functions should emit some events.

Listing 45:

```
152 function changeDefaultAllowedDeviation(uint256 _allowedDeviation
    ↪ )

159 function changeAllowedSlippage(uint256 _allowedSlippage)

170 function changeFee(uint256 _fee) external onlyGovernance {
```

3.46 CVF-46

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

Description The name "PROTOCOL_FEE" looks like the name of a constant, while actually its value could be changed.
Recommendation Consider renaming inCamelCase.

Listing 46:

```
171 PROTOCOL_FEE = _fee;
```

3.47 CVF-47

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

Description This check makes it impossible to cancel a pending governance transfer.
Recommendation Consider removing this check.

Listing 47:

```
187 require(_governance != address(0));
```

3.48 CVF-48

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

Description This function flip the deny status of a strategy, which is error-prone, as two consequent invocation with the same argument effectively do nothing.
Recommendation Consider passing the desired deny status as a separate argument of type "bool".

Listing 48:

```
203 function deny(address _strategy) external onlyGovernance {
```

3.49 CVF-49

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

Description This function should emit some event.

Listing 49:

```
203 function deny(address _strategy) external onlyGovernance {
```

3.50 CVF-50

- **Severity** Major
- **Status** Fixed
- **Category** Procedural
- **Source** DefiEdgeStrategyDeployer.sol

Description This contract should implement the "IDefiEdgeStrategyDeployer" interface. Otherwise compiler cannot check that the contract and the interface have the same API.

Listing 50:

```
8 contract DefiEdgeStrategyDeployer {
```

3.51 CVF-51

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

Description This function should emit some event.

Listing 51:

```
9 function createStrategy(
```

3.52 CVF-52

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

Recommendation The types of these arguments should be "IStrategyFactory", "IUniswapV3Pool", "IOneInchRouter", "FeedRegistryInterface", and "IStrategyManager".

Listing 52:

```
10 address _factory,
   address _pool,
   address _swapRouter,
   address _chainlinkRegistry,
   address _manager,
```

3.53 CVF-53

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

Recommendation The type of the argument should be "bool[2]".

Listing 53:

```
15 bool[] memory _usdAsBase,
```

3.54 CVF-54

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** DefiEdgeStrategyDeployer.sol

Recommendation The return type should be "DefiEdgeStrategy".
Client Comment It doesn't produce any vulnerability, for the other code dependecies we'll like to keep it as it is.

Listing 54:

```
17 ) external returns (address strategy) {
```

3.55 CVF-55

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

Description Exactly the same structure is defined in the "IStrategy" interface.
Recommendation Consider defining in one place.

Listing 55:

```
22 struct Tick {
       uint256 amount0;
       uint256 amount1;
       int24 tickLower;
       int24 tickUpper;
   }
```

3.56 CVF-56

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

Recommendation The type of the "_pool" argument should be "IUniswapV3Pool".

Listing 56:

```
39 address _pool,

68 address _pool,
```

3.57 CVF-57

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

Recommendation This commented out function should be removed.

Listing 57:

```
87 // /**
   //  * @dev Replaces old ticks with new ticks
   //  * @param _ticks New ticks
90 //  */
   // function invalidTicks(DefiEdgeStrategy.Tick[] memory _ticks)

113 // }
```

3.58 CVF-58

- **Severity** Minor
- **Status** Fixed
- **Category** Documentation
- **Source** ERC20.sol

Description The semantics of the keys in this mapping is unclear.
Recommendation Consider documenting.

Listing 58:

```
38 mapping(address => mapping(address => uint256)) public override
   ↪ allowance;
```

3.59 CVF-59

- **Severity** Major
- **Status** Fixed
- **Category** Bad datatype
- **Source** ERC20.sol

Recommendation These variables should be turned into named constants.

Listing 59:

```
42 bytes32 public name = "DefiEdge Share";
   bytes32 public symbol = "DEShare";
   uint8 public decimals = 18;
```

3.60 CVF-60

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** ERC20.sol

Description This function would not be necessary if the corresponding mapping would be made public and named appropriately.
Client Comment We'll like to go with OpenZepellin's implementation of ERC20.

Listing 60:

```
49 function balanceOf(address account)
```

3.61 CVF-61

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** ERC20.sol

Recommendation This commented out code should be removed.

Listing 61:

```
77 // /**
   //  * @dev See {IERC20-allowance}.
   //  */
80 // function allowance(address owner, address spender)
   //     public
   //     view
   //     virtual
   //     override
   //     returns (uint256)
   // {
   //     return allowance[owner][spender];
   // }
```

3.62 CVF-62

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** ERC20.sol

Description This function emits the "Approval" event, which is not expected to be emitted from the "transferFrom" function according to ERC-20.
Recommendation Consider refactoring the code to not emit unexpected events.
Client Comment We'll like to go with OpenZepellin's implementation of ERC20.

Listing 62:

```
126 _approve(
```

3.63 CVF-63

- **Severity** Minor
- **Status** Info
- **Category** Suboptimal
- **Source** ERC20.sol

Recommendation This code could be replaced with a "decreaseAllowance" call.
Client Comment We'll like to go with OpenZepellin's implementation of ERC20.

Listing 63:

```
126 _approve(
        sender,
        _msgSender(),
        allowance[sender][_msgSender()].sub(amount, "a")
130 );
```

3.64 CVF-64

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** ERC20.sol

Description Here underflow protection is used to enforce a business-level constraint. This is a bad practice as it makes the code harder to read and more error-prone.
Recommendation Consider checking business-level constraints (such as whether a balance is sufficient for an operation) via explicit "require" statements.
Client Comment We'll like to go with OpenZepellin's implementation of ERC20.

Listing 64:

```
129     allowance[sender][_msgSender()].sub(amount, "a")

181     allowance[_msgSender()][spender].sub(subtractedValue, "a")

210 _balances[sender] = _balances[sender].sub(

253 _balances[account] = _balances[account].sub(amount, "b");
```

3.65 CVF-65

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** ERC20.sol

Description These checks are redundant. They consume gas without providing any benefits. Some of these checks may never be positive in practice.
Recommendation Consider removing them.

Listing 65:

```
205 require(sender != address(0), "0");
    require(recipient != address(0), "0");

228 require(account != address(0), "0");

249 require(account != address(0), "0");

276 require(owner != address(0), "0");
    require(spender != address(0), "0");
```

3.66 CVF-66

- **Severity** Minor
- **Status** Fixed
- **Category** Unclear behavior
- **Source** ERC20.sol

Description Transferring tokens to the zero address is quite common practice used by smart contracts to "burn" tokens. This checks makes the token incompatible with such contracts.

Listing 66:

```
206 require(recipient != address(0), "0");
```

3.67 CVF-67

- **Severity** Minor
- **Status** Info
- **Category** Procedural
- **Source** ERC20.sol

Recommendation It is a good practice to put a comment into an empty block to explain why the block is empty.
Client Comment We'll like to go with OpenZepellin's implementation of ERC20

Listing 67:

```
301 ) internal virtual {}
```

3.68 CVF-68

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

Description This interface is imported twice.
Recommendation Remove one import statement.

Listing 68:

```
5  import "@uniswap/v3-core/contracts/interfaces/IUniswapV3Pool.sol
   ↪ ";

10 import "@uniswap/v3-core/contracts/interfaces/IUniswapV3Pool.sol
   ↪ ";
```

3.69 CVF-69

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

Description This constant is defined in multiple places.
Recommendation Consider defining in one place.

Listing 69:

```
17 uint256 public constant BASE = 1e18;
```

3.70 CVF-70

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

Description Identical function is defined in "OracleLibrary".
Recommendation Consider defining in one place and using everywhere,

Listing 70:

```
24 function normalise(address _token, uint256 _amount)
```

3.71 CVF-71

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

Description Declaring a library function as "public" means that it will be called via the "DELEGATECALL" opcode, which consumes quite a lot of gas.
Recommendation Consider declaring simple library functions as "internal" unless the goal is to reduce the size of a contract.
Client Comment Our goal is to reduce contract size.

Listing 71:

```
25     public

52 ) public view returns (uint256 share) {

102     public
```

3.72 CVF-72

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

Description The optional "decimals" property in ERC-20 is used by UI to render token amounts in a human-readable way. Using this property in smart contracts is discouraged.
Recommendation Consider treating all token amounts as integers.
Client Comment It doens't create any vulnerability. In case a token doesn't implement the decimals() function, the contracts won't be able to support that specific token.

Listing 72:

```
29 return uint256(_amount) * (10**(18 - IERC20Minimal(_token).
   ↪ decimals()));
```

3.73 CVF-73

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

Description Overflow and underflow are possible here.
Recommendation Consider using the "SafeMath" library.

Listing 73:

```
29 return uint256(_amount) * (10**(18 - IERC20Minimal(_token).
   ↪ decimals()));
```

3.74 CVF-74

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

Description This formula doesn't support tokens with more than 18 decimals.
Recommendation Consider implementing proper support for such tokens.

Listing 74:

```
29 return uint256(_amount) * (10**(18 - IERC20Minimal(_token).
   ↪ decimals()));
```

3.75 CVF-75

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

Description This formula is suboptimal for tokens with exactly 18 decimals, which is the most common case.
Recommendation Consider optimizing for this case.

Listing 75:

```
29 return uint256(_amount) * (10**(18 - IERC20Minimal(_token).
   ↪ decimals()));
```

3.76 CVF-76

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

Recommendation The type of the "_registry" argument should be "FeedRegistryInterface".

Listing 76:

```
44 address _registry,
```

3.77 CVF-77

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

Recommendation The type of the "_pool" argument should be "IUniswapV3Pool".

Listing 77:

```
45 address _pool,
```

3.78 CVF-78

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

Recommendation The type of this argument should be "bool[2]".

Listing 78:

```
46 bool[] memory _isBase,
```

3.79 CVF-79

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

Description There is no length check for this argument.
Recommendation Consider adding an appropriate check.

Listing 79:

```
46 bool[] memory _isBase,
```

3.80 CVF-80

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

Description Phantom overflow is possible here, i.e. a situation when the final result would fit into the destination type while some intermediary calculation overflows.
Recommendation Consider using the "mulDiv" function or some other approach resistant to phantom overflow.

Listing 80:

```
82     share = numerator.mul(_totalShares).div(denominator);

84     share = ((token0Price.mul(_amount0)).add(token1Price.mul(
         ↪ _amount1)))
           .div(100 * 1e18);

119    managementProtocolShare = _accManagementFee.mul(

121    ).div(1e8);

128 if (_accPerformanceFee > 0) {
```

3.81 CVF-81

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

Recommendation This value could be rendered as 100e18.

Listing 81:

```
85 .div(100 * 1e18);
```

3.82 CVF-82

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

Recommendation The value "100 * 1e18" should be a named constant.

Listing 82:

```
85 .div(100 * 1e18);
```

3.83 CVF-83

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

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

Listing 83:

```
97 address _factory,
```

3.84 CVF-84

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

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

Listing 84:

```
98 address _manager,
```

3.85 CVF-85

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

Description The number formats of these values is unclear.
Recommendation Consider documenting.

Listing 85:

```
99  uint256 _accManagementFee,
100 uint256 _accPerformanceFee

107     uint256 managerShare,
        uint256 protocolShare
```

3.86 CVF-86

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

Recommendation This interface should be moved to a separate file named "IERC20Minimal".

Listing 86:

```
21 interface IERC20Minimal {
```

3.87 CVF-87

- **Severity** Moderate
- **Status** Fixed
- **Category** Procedural
- **Source** OracleLibrary.sol

Recommendation The return type should be "uint8" as defined in ERC-20.

Listing 87:

```
22 function decimals() external view returns (uint256);
```

3.88 CVF-88

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

Description Declaring a library function as "public" means that it will be called via the "DELEGATECALL" opcode, which consumes quite a lot of gas.
Recommendation Consider declaring simple library functions as "internal" unless the goal is to reduce the size of a contract.
Client Comment Goal is to reduce contract size.

Listing 88:

```
31     public
43     public
78 ) public view returns (uint256 price) {
98 ) public view returns (uint256 price) {
127 ) public view returns (bool) {
224 ) public view returns (bool) {
292 ) public view returns (bool) {
```

3.89 CVF-89

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

Description The optional "decimals" property in ERC-20 is used by UI to render token amounts in a human-readable way. Using this property in smart contracts is discouraged.
Recommendation Consider treating all token amounts as integers.
Client Comment It doens't create any vulnerability. In case a token doesn't implement the decimals() function, the contracts won't be able to support that specific token.

Listing 89:

```
35 return uint256(_amount) * (10**(18 - IERC20Minimal(_token).
   ↪ decimals()));

53 uint256 token0Decimals = IERC20Minimal(pool.token0()).decimals()
   ↪ ;
   uint256 token1Decimals = IERC20Minimal(pool.token1()).decimals()
   ↪ ;
```

3.90 CVF-90

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

Description Overflow and underflow are possible here.
Recommendation Consider using the "SafeMath" library.

Listing 90:

```
35 return uint256(_amount) * (10**(18 - IERC20Minimal(_token).
   ↪ decimals()));
```

3.91 CVF-91

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

Description This formula doesn't support tokens with more than 18 decimals.
Recommendation Consider implementing proper support for such tokens.

Listing 91:

```
35 return uint256(_amount) * (10**(18 - IERC20Minimal(_token).
   ↪ decimals()));
```

3.92 CVF-92

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

Description This formula is suboptimal for tokens with exactly 18 decimals, which is the most common case.
Recommendation Consider optimizing for this case.

Listing 92:

```
35 return uint256(_amount) * (10**(18 - IERC20Minimal(_token).
   ↪ decimals()));
```

3.93 CVF-93

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

Recommendation The type of the "_pool" argument should be "IUniswapV3Pool".

Listing 93:

```
42  function getUniswapPrice(address _pool)
123     address _pool,
216     address _pool,
265         deviation = BASE.sub(diff);
```

3.94 CVF-94

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

Description When the denominator is a power of two known an compile time, there is more efficient alternatives for the "mulDiv" function that use shift instead of division.
Recommendation Consider using these alternatives.
Client Comment We're using Uniswap's mulDiv implemenation. We'll like to stick with it.

Listing 94:

```
51 price = FullMath.mulDiv(priceX192, BASE, 1 << 192);
```

3.95 CVF-95

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

Description The condition "token0Decimals >= token1Decimals" is calculated twice.
Recommendation Consider refactoring like this: price = token0Decimals >= token1Decimals ? price.mul (10**(token0Decimals - token1Decimals) : price.div (10**(token1Decimals - token0Decimals);

Listing 95:

```
56 uint256 decimalsDelta = token0Decimals >= token1Decimals

61 if (token0Decimals >= token1Decimals) {
```

3.96 CVF-96

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

Description This formula is suboptimal in the most common case when both tokens have the same number of decimals.
Recommendation Consider optimizing for this case.

Listing 96:

```
61 if (token0Decimals >= token1Decimals) {
       price = price.mul(10**(decimalsDelta));
   } else {
       price = price.div(10**(decimalsDelta));
   }
```

3.97 CVF-97

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

Description Overflow is possible here.
Recommendation Consider using safe power function.

Listing 97:

```
62 price = price.mul(10**(decimalsDelta));

64 price = price.div(10**(decimalsDelta));
```

3.98 CVF-98

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

Recommendation The type of the "_registry" argument should be "FeedRegistryInterface".

Listing 98:

```
75  address _registry,
95  address _registry,
124 address _registry,
217 address _registry,
```

3.99 CVF-99

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

Description Overflow and underflow are possible here.
Recommendation Consider using the "SafeMath" library.

Listing 99:

```
83  price = uint256(_price) * (10**(18 - registry.decimals(_base,
      ↪ _quote)));
```

3.100 CVF-100

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

Description This formula doesn't support tokens with more than 18 decimals.
Recommendation Consider implementing proper support for such tokens.

Listing 100:

```
83  price = uint256(_price) * (10**(18 - registry.decimals(_base,
      ↪ _quote)));
```

3.101 CVF-101

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

Description This formula is suboptimal for tokens with exactly 18 decimals, which is the most common case.
Recommendation Consider optimizing for this case.

Listing 101:

```
83 price = uint256(_price) * (10**(18 - registry.decimals(_base,
   ↪ _quote)));
```

3.102 CVF-102

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

Description This assignment is redundant.
Recommendation Just return the expression value.

Listing 102:

```
83 price = uint256(_price) * (10**(18 - registry.decimals(_base,
   ↪ _quote)));
```

3.103 CVF-103

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

Description Phantom overflow is possible here, i.e. a situation when the final result would fit into the destination type, while some intermediary calculation overflows.
Recommendation Consider using the "mulDiv" function or some other approach that is resistant to phantom overflows.

Listing 103:

```
104             .mul(

110             )
                .div(BASE);

132         .mul(getPriceInUSD(_registry, pool.token1(), _usdAsBase[1]))
            .div(BASE);

258 diff = amountInUSD.mul(BASE).div(amountOutUSD);

326 diff = amountInUSD.mul(BASE).div(amountOutUSD);
```

3.104 CVF-104

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

Recommendation Consider extracting scaled multiplication and division into a utility functions.

Listing 104:

```
104             .mul(

110             )
                .div(BASE);

132         .mul(getPriceInUSD(_registry, pool.token1(), _usdAsBase[1]))
            .div(BASE);

258 diff = amountInUSD.mul(BASE).div(amountOutUSD);

326 diff = amountInUSD.mul(BASE).div(amountOutUSD);
```

3.105 CVF-105

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

Recommendation The type of this argument should be "bool[2]".

Listing 105:

```
125 bool[] memory _usdAsBase,
```

3.106 CVF-106

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

Description There is no length check for this argument.
Recommendation Consider adding an appropriate check.

Listing 106:

```
125 bool[] memory _usdAsBase,
```

3.107 CVF-107

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

Description In case one of the pool tokens is ETH or both tokens are quoted against ETH, rather than USD, it would be more efficient to use ETH denominated prices instead of USD denominated.
Recommendation Consider using ETH denominated prices where appropriate.
Client Comment Require lots of changes.

Listing 107:

```
132     .mul(getPriceInUSD(_registry, pool.token1(), _usdAsBase[1]))

136 uint256 chainlinkPriceInUSD = getPriceInUSD(
```

3.108 CVF-108

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

Recommendation This could be simplified as: return diff > BASE.add (_allowedDeviation) || diff < BASE.sub (_allowedDeviation);

Listing 108:

```
149 if (
150     diff > (BASE.add(_allowedDeviation)) ||
        diff < (BASE.sub(_allowedDeviation))
    ) {
        return true;
    }

156 return false;
```

3.109 CVF-109

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

Recommendation It would be more logical to use (1+x) and 1/(1+x) as the boundaries for allowed deviation, rather than (1+x) and (1-x).

Listing 109:

```
150 diff > (BASE.add(_allowedDeviation)) ||
    diff < (BASE.sub(_allowedDeviation))

330 diff > (BASE.add(factory.allowedSlippage())) ||
    diff < (BASE.sub(factory.allowedSlippage()))
```

3.110 CVF-110

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

Recommendation This commented function should be removed.

Listing 110:

```
159 // /**
160 //  * @notice Checks the if swap exceed allowed swap deviation
    ↪ or not
//  * @param _pool Address of the pool
//  * @param _registry Chainlink registry
//  * @param _usdAsBase checks if pegged to USD
//  * @param _manager Manager contract address to check allowed
    ↪ deviation
//  */
// function isSwapExceedDeviation(

202 // }
```

3.111 CVF-111

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

Description The call to "pool.token1()" is redundant.
Recommendation Consider simplifying like this: (bool usdAsBaseAmountIn, bool usdAsBaseAmountOut) = pool.token0() == _tokenIn ? (_usdAsBase[0], _usdAsBase[1]) : (_usdAsBase[1], _usdAsBase[0]);

Listing 111:

```
230 bool usdAsBaseAmountIn = pool.token0() == _tokenIn
        ? _usdAsBase[0]
        : _usdAsBase[1];

234 bool usdAsBaseAmountOut = pool.token1() == _tokenOut
        ? _usdAsBase[1]
        : _usdAsBase[0];

298 bool usdAsBaseAmountIn = IUniswapV3Pool(_pool).token0() ==
    ↪ _tokenIn
        ? _isBase[0]
300     : _isBase[1];

302 bool usdAsBaseAmountOut = IUniswapV3Pool(_pool).token1() ==
    ↪ _tokenOut
        ? _isBase[1]
        : _isBase[0];
```

3.112 CVF-112

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

Description The "amountInUSD" value here is already multiplied by BASE. Multiplying it once again leaves little room for significant digits.
Recommendation Consider dividing "amountOutUSD" by "BASE" instead.

Listing 112:

```
258 diff = amountInUSD.mul(BASE).div(amountOutUSD);

326 diff = amountInUSD.mul(BASE).div(amountOutUSD);
```

3.113 CVF-113

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

Recommendation It would be more logical to use (diff - 1) and (1/diff - 1) as deviation measurement instead of (diff - 1) and (1 - diff).
Client Comment We think (1+x) & (1-x) is more accurate to use than (1+x) & (1/1+x). We would also like to keep same flow for all the operators.

Listing 113:

```
263 deviation = diff.sub(BASE);

265 deviation = BASE.sub(diff);
```

3.114 CVF-114

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

Recommendation The type of this argument should be 'IUniswapV3Factory".

Listing 114:

```
286 address _factory,
```

3.115 CVF-115

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

Recommendation Events are usually named via nouns, such as "Fee", "Operator", "Limit" etc.

Listing 115:

```
13 import "@openzeppelin/contracts/access/AccessControl.sol";

18     event ChangeFee(uint256 tier);
       event ChangeOperator(address indexed operator);
20     event ChangeLimit(uint256 limit);
       event ChangeAllowedDeviation(uint256 deviation);
       event ClaimFee(uint256 managerFee, uint256 protocolFee);
       event ChangePerformanceFee(uint256 performanceFee);
```

3.116 CVF-116

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

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

Listing 116:

```
42 uint256 public managementFee;
```

3.117 CVF-117

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

Description These variables are not initialized.
Recommendation Consider explicitly initializing them to 0.
Client Comment By default vault of these variables is zero.

Listing 117:

```
52 uint256 public maxAllowedSwap;

55 uint256 public swapCounter;

58 uint256 public lastSwapTimestamp;
```

3.118 CVF-118

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

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

Listing 118:

```
65 address _factory,
```

3.119 CVF-119

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

Description There are no range checks for these arguments.
Recommendation Consider adding appropriate checks.

Listing 119:

```
68  uint256 _managementFee,
    uint256 _performanceFee,
70  uint256 _limit,
    uint256 _allowedDeviation
```

3.120 CVF-120

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

Recommendation This could be simplified to: return hasRole(ADMIN_ROLE, _account) || hasRote(MANAGER_ROLE, _account);

Listing 120:

```
107 if (hasRole(ADMIN_ROLE, _account) || hasRole(MANAGER_ROLE,
    ↪ _account)) {
        return true;
    } else {
110     return false;
    }
```

3.121 CVF-121

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

Recommendation This could be simplified to: return hasRole(ADMIN_ROLE, _account) || hasRole (MANAGER_ROLE, _account) || hasRole (BURNER_ROLE, _account);

Listing 121:

```
115 if (
        hasRole(ADMIN_ROLE, _account) ||
        hasRole(MANAGER_ROLE, _account) ||
        hasRole(BURNER_ROLE, _account)
    ) {
120     return true;
    } else {
        return false;
    }
```

3.122 CVF-122

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

Recommendation The return type should be 'IStrategy".

Listing 122:

```
126 function strategy() public view returns (address) {
```

3.123 CVF-123

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

Description There are no range checks for the arguments.
Recommendation Consider adding appropriate checks.

Listing 123:

```
135 function changeFee(uint256 _fee) public onlyOperator {

172 function changeLimit(uint256 _limit) external onlyOperator {

200 function changeAllowedDeviation(uint256 _allowedDeviation)

259 function changeMaxSwapLimit(uint256 _limit) external
      ↪ onlyGovernance {
```

3.124 CVF-124

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

Description These events are emitted even when nothing actually changed.

Listing 124:

```
137 emit ChangeFee(managementFee);

165 emit ChangeOperator(operator);

186 emit ChangePerformanceFee(_performanceFee);

205 emit ChangeAllowedDeviation(_allowedDeviation);
```

3.125 CVF-125

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

Description These functions should emit some event.

Listing 125:

```
144 function changeFeeTo(address _newFeeTo) external onlyOperator {
152 function changeOperator(address _operator) external onlyOperator
    ↪ {
172 function changeLimit(uint256 _limit) external onlyOperator {
192 function freezeEmergencyFunctions() external onlyGovernance {
259 function changeMaxSwapLimit(uint256 _limit) external
    ↪ onlyGovernance {
```

3.126 CVF-126

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

Description This check makes it impossible to cancel a pending operator change.
Recommendation Consider removing this check.

Listing 126:

```
153 require(_operator != address(0));
```

3.127 CVF-127

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

Recommendation This constant must be named.

Listing 127:

```
184 require(_performanceFee <= 20 * 1e6);
```

3.128 CVF-128

- **Severity** Major
- **Status** Fixed
- **Category** Unclear behavior
- **Source** StrategyManager.sol

Description This check doesn't guarantee that the allowed swap deviation will always be less than the allowed deviation, as it is still possible to set the allowed deviation below the allowed swap deviation.
Recommendation Consider adding a similar check to the "changeAllowedDeviation" function.

Listing 128:

```
216 require(_allowedSwapDeviation < allowedDeviation, "ID");
```

3.129 CVF-129

- **Severity** Major
- **Status** Fixed
- **Category** Suboptimal
- **Source** StrategyManager.sol

Description Converting timestamps to calendar dates just to check whether two timestamps belong to the same day is overkill. See the next comment for suggestion of a more efficient and much simpler approach.

Listing 129:

```
225 (
        uint256 currentYear,
        uint256 currentMonth,
        uint256 currentDay
    ) = DateTimeLibrary.timestampToDate(block.timestamp);
230 (uint256 swapYear, uint256 swapMonth, uint256 swapDay) =
    ↪ DateTimeLibrary
      .timestampToDate(lastSwapTimestamp);
```

3.130 CVF-130

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

Recommendation This check could be simplified as: if (block.timestamp / 1 days == lastSwapTimestamp / 1 days)

Listing 130:

```
233 if (
        currentYear == swapYear &&
        currentMonth == swapMonth &&
        currentDay == swapDay
    ) {
```

3.131 CVF-131

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

Description The function always returns true.
Recommendation Consider returning nothing.

Listing 131:

```
246 return true;

251 return true;
```

3.132 CVF-132

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

Recommendation This interface should be moved to a separate file named "IOneInchRouter.sol".

Listing 132:

```
18 interface IOneInchRouter {
```

3.133 CVF-133

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

Recommendation The type of these variables should be "IERC20".

Listing 133:

```
52 address internal token0;
   address internal token1;
```

3.134 CVF-134

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

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

Listing 134:

```
56 address internal chainlinkRegistry;
```

3.135 CVF-135

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

Recommendation The type of this variable should be "bool[2]" or just a bit mask of two bits.

Listing 135:

```
60 bool[] public usdAsBase; // for Chainlink oracle
```

3.136 CVF-136

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

Recommendation This commented out modifier should be removed.

Listing 136:

```
68 // modifier onlyOperatorAndBurner() {
   //     require(
70 //         manager.hasRole(manager.ADMIN_ROLE(), msg.sender) ||
   //             manager.hasRole(manager.MANAGER_ROLE(), msg.
      ↪ sender) ||
   //             manager.hasRole(manager.BURNER_ROLE(), msg.sender
      ↪ ),
   //         "N"
   //     );
   //     _;
   // }
```

3.137 CVF-137

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

Description This function could be made much more efficient (O(n) instead of O(n^2)) is would require the ticks to be sorted, e.g. by the lower tick and by the upper tick. In such a case it would be enough to check that adjacent ticks are not the same and are in proper order.
Client Comment It doesn't produce any security vulnerability. We would like to keep it as per the current implementation

Listing 137:

```
82 function isInvalidTicks(Tick[] memory _ticks)
```

3.138 CVF-138

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

Description This condition is always true, as the loop condition is "j < i".

Listing 138:

```
93 if (i != j) {
```

3.139 CVF-139

- **Severity** Major
- **Status** Fixed
- **Category** Suboptimal
- **Source** StrategyBase.sol

Recommendation Should "break" or "return" after this. No reason to proceed with the loop.

Listing 139:

```
96 invalid = true;
```

3.140 CVF-140

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

Description The name of this modified looks like a function name.
Recommendation Consider renaming to "onlyValidStrategy".

Listing 140:

```
108 modifier isValidStrategy() {
```

3.141 CVF-141

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

Description The name of this modifier looks like a function name.
Recommendation Consider renaming to "onlyHasDeviation".

Listing 141:

```
117 modifier hasDeviation() {
```

3.142 CVF-142

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

Recommendation 1e8 should be a named constant

Listing 142:

```
163 managerShare = share.mul(managementFee).div(1e8);
```

3.143 CVF-143

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

Description The value returned by this function includes not yet minted tokens, which is hot now token contracts usually work.
Recommendation Consider minting fee tokens when fee is accrued, rather than when it is claimed. Alternatively, consider not including not yet minted tokens into the value returned by this function.
Client Comment Minting fee tokens everytime will cost gas. To save the gas we chose to mint the fee tokens only when it's claimed. claimFee() function can be called by anyone.

Listing 143:

```
171 function totalSupply() public view override returns (uint256) {
```

3.144 CVF-144

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

Recommendation This constant is redundant, as it is equivalent to the "1 days" expression.

Listing 144:

```
5 uint256 constant SECONDS_PER_DAY = 24 * 60 * 60;
```

3.145 CVF-145

- **Severity** Major
- **Status** Fixed
- **Category** Suboptimal
- **Source** DateTimeLibrary.sol

Description The formula used by this function tries to support negative timestamps down to thousands years BC. Removing such ability would make the function simpler, more efficient and easier to read.
Recommendation Consider removing such ability.

Listing 145:

```
14 function timestampToDate(uint256 _timestamp)
```

3.146 CVF-146

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

Recommendation Numbers used here should be named constants with explanations about their meaning.

Listing 146:

```
25 int256 L = _days + 68569 + OFFSET19700101;
   int256 N = (4 * L) / 146097;
   L = L - (146097 * N + 3) / 4;
   int256 _year = (4000 * (L + 1)) / 1461001;
   L = L - (1461 * _year) / 4 + 31;
30 int256 _month = (80 * L) / 2447;
   int256 _day = L - (2447 * _month) / 80;
   L = _month / 11;
   _month = _month + 2 - 12 * L;
   _year = 100 * (N - 49) + _year + L;
```

3.147 CVF-147

- **Severity** Minor
- **Status** Fixed
- **Category** Bad naming
- **Source** UniswapV3LiquidityManager.sol

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

Listing 147:

```
24 event FeesClaimed(
```

3.148 CVF-148

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** UniswapV3LiquidityManager.sol

Recommendation The type of this parameter should be "IUniswapV3Pool".

Listing 148:

```
32 address pool;
```

3.149 CVF-149

- **Severity** Minor
- **Status** Fixed
- **Category** Overflow/Underflow
- **Source** UniswapV3LiquidityManager.sol

Description Phantom overflow is possible here.
Recommendation Consider using the "mulDiv" function or other approach resistant to phantom overflow.

Listing 149:

```
96 uint256 liquidity = uint256(_currentLiquidity).mul(_shares).div(
       totalSupply()
   );

149 _fee0.mul(performanceFee).div(1e8),
150 _fee1.mul(performanceFee).div(1e8),
```

3.150 CVF-150

- **Severity** Minor
- **Status** Fixed
- **Category** Bad datatype
- **Source** UniswapV3LiquidityManager.sol

Recommendation The "1e8" value should be a named constant.

Listing 150:

```
149 _fee0.mul(performanceFee).div(1e8),
150 _fee1.mul(performanceFee).div(1e8),
```

3.151 CVF-151

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** UniswapV3LiquidityManager.sol

Description This function performs the "hasDeviation" check and the "isAllowedToBurn" check on every invocation, i.e. for every tick.
Recommendation Consider refactoring the code to perform these checks only once, before the loop.

Listing 151:

```
165 burnLiquiditySingle(i);
```

3.152 CVF-152

- **Severity** Major
- **Status** Fixed
- **Category** Flaw
- **Source** UniswapV3LiquidityManager.sol

Description The returned values are ignored.
Recommendation Consider aggregating them and returning to the caller.

Listing 152:

```
165 burnLiquiditySingle(i);
```

3.153 CVF-153

- **Severity** Moderate
- **Status** Fixed
- **Category** Unclear behavior
- **Source** UniswapV3LiquidityManager.sol

Recommendation Should probably be "0", not "tick.amountN".

Listing 153:

```
202       : tick.amount0;

205   : tick.amount1;
```

3.154 CVF-154

- **Severity** Minor
- **Status** Fixed
- **Category** Unclear behavior
- **Source** UniswapV3LiquidityManager.sol

Description These checks allow cases when srcToken = dstToken.
Recommendation Consider rewriting like this: require (srcToken == token0 && dstToken == token1 || srcToken == token1 && dstToken == token0);

Listing 154:

```
217 require(srcToken == token0 || srcToken == token1, "IT");
    require(dstToken == token0 || dstToken == token1, "IT");
```

3.155 CVF-155

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** UniswapV3LiquidityManager.sol

Description The expression "srcToken == token0" is calculated twice.
Recommendation Consider calculating once and reusing.

Listing 155:

```
217 require(srcToken == token0 || srcToken == token1, "IT");

220 address tokenIn = srcToken == token0 ? token0 : token1;
```

3.156 CVF-156

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** UniswapV3LiquidityManager.sol

Description The expression "dstToken == token0" is calculated twice.
Recommendation Consider calculating once and reusing.

Listing 156:

```
218 require(dstToken == token0 || dstToken == token1, "IT");

221 address tokenOut = dstToken == token0 ? token0 : token1;
```

3.157 CVF-157

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** UniswapV3LiquidityManager.sol

Description The "tokenIn" and "tokenOut" variables are redundant, as they are just aliases for "srcToken" and "dstToken".
Recommendation Consider removing these variables.

Listing 157:

```
220 address tokenIn = srcToken == token0 ? token0 : token1;
    address tokenOut = dstToken == token0 ? token0 : token1;
```

3.158 CVF-158

- **Severity** Critical
- **Status** Fixed
- **Category** Flaw
- **Source** UniswapV3LiquidityManager.sol

Description The content of the "data" array is not validated. In particular, function selector is not validated against the list of allowed selectors. This allows calling arbitrary functions on "oneInchRouter" on behalf of this smart contract.
Recommendation Consider validating the content of the "data" array.

Listing 158:

```
229 (bool success, bytes memory returnData) = address(oneInchRouter)
    ↪ .call{
230     value: 0
}(data);
```

3.159 CVF-159

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** UniswapV3LiquidityManager.sol

Recommendation Consider passing the returndata as is as it can contain useful information.

Listing 159:

```
237 revert("swap");
```

3.160 CVF-160

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** UniswapV3LiquidityManager.sol

Description This unwraps the returns revert reason and then wraps it again, which is waste of gas.
Recommendation Consider just passing the returned data as is: assembly { returndatacopy (0, 0, returndatasize ()) revert (0, returndatasize ()) }

Listing 160:

```
240 assembly {
        returnData := add(returnData, 0x04)
}
revert(abi.decode(returnData, (string)));
```

3.161 CVF-161

- **Severity** Major
- **Status** Fixed
- **Category** Flaw
- **Source** UniswapV3LiquidityManager.sol

Description This screws up the "returnData" length.
Recommendation See the following answer for details: https://ethereum.stackexchange.com/a/110428/4955

Listing 161:

```
241 returnData := add(returnData, 0x04)
```

3.162 CVF-162

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** UniswapV3LiquidityManager.sol

Recommendation This commented out function should be removed.

Listing 162:

```
283 // /**
    //  * @notice Swaps through the path calculated by Uniswap auto-
    ↪ router
    //  */
    // function swap(

372 // }
```

3.163 CVF-163

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** UniswapV3LiquidityManager.sol

Recommendation This check should be performed at the very beginning of the function.

Listing 163:

```
384 require(msg.sender == address(pool));
```

3.164 CVF-164

- **Severity** Minor
- **Status** Fixed
- **Category** Suboptimal
- **Source** UniswapV3LiquidityManager.sol

Description On every loop iteration, "tokensOwedN" values are added to both, "totalFeeN" and "amountN".
Recommendation Consider just adding to "totalFeeN" insidie the loop and then adding "totalFeeN" to "amountN" after the loop.

Listing 164:

```
468 totalFee0 = totalFee0.add(tokensOwed0);
    totalFee1 = totalFee1.add(tokensOwed1);

471 amount0 = amount0.add(tokensOwed0).add(position0);
    amount1 = amount1.add(tokensOwed1).add(position1);
```

3.165 CVF-165

- **Severity** Minor
- **Status** Info
- **Category** Bad datatype
- **Source** IOneInch.sol

Recommendation The type of these fields should be "IERC20".
Client Comment Struct is from one inch contract, better to not change it.

Listing 165:

```
7 address srcToken;
  address dstToken;
```

3.166 CVF-166

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** IERC20.sol

Description Some contracts use "IERC20" interface from OpenZeppelin, while some other contracts use this interface.
Recommendation Consider using the same interface everywhere.

Listing 166:

```
3 interface IERC20 {
```

3.167 CVF-167

- **Severity** Minor
- **Status** Fixed
- **Category** Procedural
- **Source** IERC20.sol

Recommendation This function is not defined by ERC20 and should be moved to some other interface.

Listing 167:

```
23 function permit(
```

3.168 CVF-168

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

Description Despite the name, this function return a single tick.
Recommendation Consider renaming to "getTick" or "tick".

Listing 168:

```
14 function ticks(uint256 index) external view returns (Tick memory
   ↪ );
```

3.169 CVF-169

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

Recommendation The return type should be "IUniswapV3Pool".

Listing 169:

```
17 function pool() external view returns (address);
```

3.170 CVF-170

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

Recommendation The "user" parameter should be indexed.

Listing 170:

```
48 event Mint(address user, uint256 share, uint256 amount0, uint256
   ↪ amount1);
   event Burn(address user, uint256 share, uint256 amount0, uint256
   ↪ amount1);
```

3.171 CVF-171

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

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

Listing 171:

```
 9 function managementFee() external view returns (uint256);

11 function performanceFee() external view returns (uint256);

19 function allowedDeviation() external view returns (uint256);

21 function allowedSwapDeviation() external view returns (uint256);
```

3.172 CVF-172

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

Recommendation The returned type should be "IStrategyFactotry".

Listing 172:

```
25 function factory() external view returns (address);
```

3.173 CVF-173

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

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

Listing 173:

```
27 function increamentSwapCounter() external returns (bool);
```

3.174 CVF-174

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

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

Listing 174:

```
5 function allowedSlippage() external view returns (uint256);
```

3.175 CVF-175

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

Description The name is too generic.
Recommendation Consider renaming to "isValidStrategy".

Listing 175:

```
7 function isValid(address) external view returns (bool);
```

3.176 CVF-176

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

Recommendation The argument type should be 'IStrategy".

Listing 176:

```
7 function isValid(address) external view returns (bool);
```

3.177 CVF-177

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

Recommendation The argument type should be "IStrategyManager", the return type should be "IStrategy".

Listing 177:

```
9 function strategyByManager(address) external view returns (
    ↪ address);
```

3.178 CVF-178

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

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

Listing 178:

```
15 function PROTOCOL_FEE() external view returns (uint256);
```

3.179 CVF-179

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

Recommendation The return type should be "IUniswapV3Factory".

Listing 179:

```
19 function uniswapV3Factory() external view returns (address);
```

3.180 CVF-180

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

Recommendation The return type should be "FeedRegistryInterface".

Listing 180:

```
21 function chainlinkRegistry() external view returns (address);
```

3.181 CVF-181

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

Recommendation The return type should be 'ISwapRouter".

Listing 181:

```
23 function swapRouter() external view returns (address);
```