



Ocean Protocol

Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **October 24th, 2021 – November 8th, 2021**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY	7
CONTACTS	8
1 EXECUTIVE OVERVIEW	9
1.1 INTRODUCTION	10
1.2 AUDIT SUMMARY	10
1.3 TEST APPROACH & METHODOLOGY	11
RISK METHODOLOGY	12
1.4 SCOPE	14
2 ASSESSMENT SUMMARY & FINDINGS OVERVIEW	16
3 FINDINGS & TECH DETAILS	18
3.1 (HAL-01) MINT ATTACK AFTER NFT TRANSFER - HIGH	20
Description	20
Code Location	20
Risk Level	20
Recommendation	20
Remediation Plan	20
3.2 (HAL-02) MINT ATTACK WITH WORTHLESS TOKEN - HIGH	21
Description	21
Risk Level	21
Recommendation	21
Remediation Plan	21
3.3 (HAL-03) DT TOKEN STAKE NOT CALCULATED SUCCESSFULLY - HIGH	22
Description	22
Code Location	22

Risk Level	24
Recommendation	24
Remediation Plan	24
3.4 (HAL-04) UNCHECKED TRANSFER - MEDIUM	25
Description	25
Code Location	25
Risk Level	33
Recommendation	33
Remediation Plan	33
3.5 (HAL-05) MULTIPLE EXTERNAL CALLS WITHIN LOOP MAY LEAD TO DENIAL OF SERVICE(DOS) - MEDIUM	34
Description	34
Code Location	34
Risk Level	39
Recommendation	39
Reference	39
Remediation Plan	39
3.6 (HAL-06) MISSING RE-ENTRANCY PROTECTION - LOW	40
Description	40
Code Location	41
Risk Level	58
Recommendation	58
References	58
Remediation Plan	58
3.7 (HAL-07) IGNORED RETURN VALUES - LOW	59
Description	59

Code Location	59
Risk Level	63
Recommendation	63
Remediation Plan	64
3.8 (HAL-08) MISSING ZERO-ADDRESS CHECK - LOW	65
Description	65
Code Location	65
Risk Level	68
Recommendation	68
Remediation Plan	68
3.9 (HAL-09) DIVIDE BEFORE MULTIPLY - LOW	69
Description	69
Code Location	69
Risk Level	72
Recommendation	72
Remediation Plan	72
3.10 (HAL-10) USE OF BLOCK-TIMESTAMP - LOW	73
Description	73
Code Location	73
Risk Level	74
Recommendation	74
Remediation Plan	74
3.11 (HAL-11) EXPERIMENTAL FEATURES ENABLED - LOW	75
Description	75

Reference	76
Code Location	76
Risk Level	76
Recommendation	76
Remediation Plan	77
3.12 (HAL-12) FLOATING PRAGMA - LOW	78
Description	78
Code Location	78
Risk Level	78
Recommendation	79
Remediation Plan	79
3.13 (HAL-13) OUTDATED DEPENDENCIES - LOW	80
Description	80
Code Location	80
Risk Level	80
Recommendation	80
References	81
Remediation Plan	81
3.14 (HAL-14) PRAGMA VERSION DEPRECATED - LOW	82
Description	82
Code Location	82
Risk Level	83
Recommendation	83
Remediation Plan	84
3.15 (HAL-15) MULTIPLE PRAGMA DEFINITIONS - LOW	85
Description	85

Code Location	85
Risk Level	87
Recommendation	87
Remediation Plan	87
3.16 (HAL-16) MISSING EVENTS EMISSION - INFORMATIONAL	88
Description	88
Code Location	88
Risk Level	89
Recommendation	89
Remediation Plan	89
3.17 (HAL-17) REDUNDANT BOOLEAN COMPARISON - INFORMATIONAL	90
Description	90
Code Location	90
Risk Level	104
Recommendation	104
Remediation Plan	104
3.18 (HAL-18) USE OF INLINE ASSEMBLY - INFORMATIONAL	105
Description	105
Code Location	105
Risk Level	107
Recommendation	107
Remediation Plan	107
3.19 (HAL-19) REDUNDANT VARIABLES - INFORMATIONAL	108
Description	108
Code Location	108
Risk Level	108

Recommendation	108
Remediation Plan	109
3.20 (HAL-20) POSSIBLE MISUSE OF PUBLIC FUNCTIONS - INFORMATIONAL	
110	
Description	110
Code Location	110
Risk Level	111
Recommendation	111
Remediation Plan	111
3.21 (HAL-21) POTENTIAL UNSAFE CALCULATION - INFORMATIONAL	113
Description	113
Code Location	113
Recommendation	113
Remediation Plan	113
3.22 (HAL-22) GAS OPTIMIZATIONS - INFORMATIONAL	114
Description	114
Code Locations	114
Recommendation	114
Remediation Plan	114
4 AUTOMATED TESTING	115
4.1 STATIC ANALYSIS REPORT	116
Description	116
Results	116
4.2 AUTOMATED SECURITY SCAN	121
Description	121
Results	121

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	10/25/2021	Alessandro Cara
0.2	Document Updates	11/02/2021	Juned Ansari
0.3	Document Updates	11/03/2021	Juned Ansari
0.4	Document Draft Release	11/12/2021	Alessandro Cara
0.5	Draft Review	11/16/2021	Gabi Urrutia
1.0	Remediation Plan	12/16/2021	Alessandro Cara
1.1	Remediation Plan Review	12/16/2021	Gabi Urrutia
1.2	Final Updates	03/23/2022	Alessandro Cara
1.3	Release Review	03/28/2022	Gabi Urrutia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
Juned Ansari	Halborn	Juned.Anṣari@halborn.com
Alessandro Cara	Halborn	alessandro.cara@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Ocean Protocol engaged Halborn to conduct a security assessment on their smart contracts v4main branch beginning on October 24th, 2021 and ending November 12th, 2021. This security assessment was scoped to the smart contracts v4main branch code in Solidity. In March 2022, Halborn reviewed the code changes that were applied since the original assessment, to ensure that the release product functioned correctly and did not present additional security concerns.

Halborn recommends that the following corrective actions are implemented to mitigate the identified security issues:

- Add checks to make sure token transfers have been successful
- Add zero address and zero amount checks where missing
- Make sure external dependencies are up-to-date
- Implement re-entrancy prevention where missing
- Lock the pragma version and be consistent across the entire set of contracts.
- Replace the use of `block.timestamp` with `block.number` or use oracles instead

1.2 AUDIT SUMMARY

The team at Halborn was provided three weeks for the engagement and assigned two full time security engineer to audit the security of the smart contract. A security engineer is a blockchain and smart-contract security expert with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The purpose of this audit was to achieve the following:

- Ensure that all v4main Contract functions are intended
- Identify potential security issues with the assets in scope

In summary, Halborn identified some security risks that were mostly addressed by the Ocean Protocol team.

In March 2022, Halborn reviewed the final changes to the smart contracts before launch. The changes included a restructuring of access control to ensure that the contract code is simpler and more secure. In more details, changes have been applied to the ownership of `FixedRateExchange` and `Dispenser`. At the time of writing, these contracts would be owned by users with the `ERC20Deployer` role or the owner of the NFT #1. This prevents permission issues with transfers of NFT token #1 to a new user. Additionally, further controls were added to prevent creating pools with the same datatoken as a pair, and proper transfer of datatokens and fees prior to NFT ownership transfer.

Finally, the NFT contract code was restructured to implement EIP165 and registry interfaces to ensure that it complies with ERC721 standards.

1.3 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of this audit. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of the Ocean Protocol contract solidity code and can quickly identify items that do not follow security best practices. The following phases and associated tools were used throughout the term of the audit:

- Research into architecture and purpose.
- Smart contract manual code review and walkthrough.
- Graphing out functionality and contract logic/connectivity/functions (`solgraph`)
- Manual assessment of use and safety for the critical Solidity variables and functions in scope to identify any arithmetic related vulnerability classes.
- Manual testing by custom scripts.

- Scanning of solidity files for vulnerabilities, security hotspots or bugs. ([MythX](#))
- Static Analysis of security for scoped contract, and imported functions. ([Slither](#))
- Testnet deployment ([Remix IDE](#))

RISK METHODOLOGY:

Vulnerabilities or issues observed by Halborn are ranked based on the risk assessment methodology by measuring the **LIKELIHOOD** of a security incident and the **IMPACT** should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. The quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that were used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of **5** to **1** with **5** being the highest likelihood or impact.

RISK SCALE - LIKELIHOOD

- 5 - Almost certain an incident will occur.
- 4 - High probability of an incident occurring.
- 3 - Potential of a security incident in the long term.
- 2 - Low probability of an incident occurring.
- 1 - Very unlikely issue will cause an incident.

RISK SCALE - IMPACT

- 5 - May cause devastating and unrecoverable impact or loss.
- 4 - May cause a significant level of impact or loss.
- 3 - May cause a partial impact or loss to many.
- 2 - May cause temporary impact or loss.
- 1 - May cause minimal or un-noticeable impact.

The risk level is then calculated using a sum of these two values, creating a value of **10** to **1** with **10** being the highest level of security risk.

EXECUTIVE OVERVIEW



10 - CRITICAL

9 - 8 - HIGH

7 - 6 - MEDIUM

5 - 4 - LOW

3 - 1 - VERY LOW AND INFORMATIONAL

1.4 SCOPE

IN-SCOPE : Ocean Protocol v4main contracts GitHub repository

POST AUDIT REPOSITORY : Ocean Protocol v4main postaudit GitHub repository

The security assessment was scoped to the following smart contracts:

```
1 contracts/ERC721Factory.sol
2 contracts/templates/ERC20Template.sol
3 contracts/templates/ERC20TemplateEnterprise.sol
4 contracts/templates/ERC721Template.sol
5 contracts/utils/ERC721RolesAddress.sol
6 contracts/utils/UtilsLib.sol
7 contracts/utils/Ownable.sol
8 contracts/utils/Deployer.sol
9 contracts/utils/ERC20Roles.sol
10 contracts/utils/ERC721/IERC721Enumerable.sol
11 contracts/utils/ERC721/IERC721Receiver.sol
12 contracts/utils/ERC721/IERC721Metadata.sol
13 contracts/utils/ERC721/ERC721.sol
14 contracts/utils/ERC721/Context.sol
15 contracts/utils/ERC721/IERC721.sol
16 contracts/utils/ERC721/Address.sol
17 contracts/utils/ERC721/Strings.sol
18 contracts/utils/ERC725/ERC725Ocean.sol
19 contracts/pools/ssContracts/SideStaking.sol
20 contracts/pools/dispenser/Dispenser.sol
21 contracts/pools/FactoryRouter.sol
22 contracts/pools/balancer/BConst.sol
23 contracts/pools/balancer/BFactory.sol
24 contracts/pools/balancer/BMath.sol
25 contracts/pools/balancer/BToken.sol
26 contracts/pools/balancer/BPool.sol
27 contracts/pools/balancer/BNum.sol
28 contracts/pools/fixedRate/FixedRateExchange.sol
29 contracts/interfaces/ISideStaking.sol
30 contracts/interfaces/IFactoryRouter.sol
31 contracts/interfaces/IFactory.sol
32 contracts/interfaces/IMetadata.sol
33 contracts/interfaces>IDispenser.sol
```

```
34 contracts/interfaces/IERC20Template.sol  
35 contracts/interfaces/IERC721Template.sol  
36 contracts/interfaces/IFixedRateExchange.sol  
37 contracts/interfaces/IERC725X.sol  
38 contracts/interfaces/IERC725Y.sol  
39 contracts/interfaces/IERC20.sol  
40 contracts/interfaces/IV3Factory.sol  
41 contracts/interfaces/IERC1271.sol  
42 contracts/interfaces/IV3ERC20.sol  
43 contracts/interfaces/IPool.sol  
44 contracts/communityFee/OPFCommunityFeeCollector.sol
```

Commit-ID : 0ecfd02598901ffa86a1aff71e7bfe92513a23a1

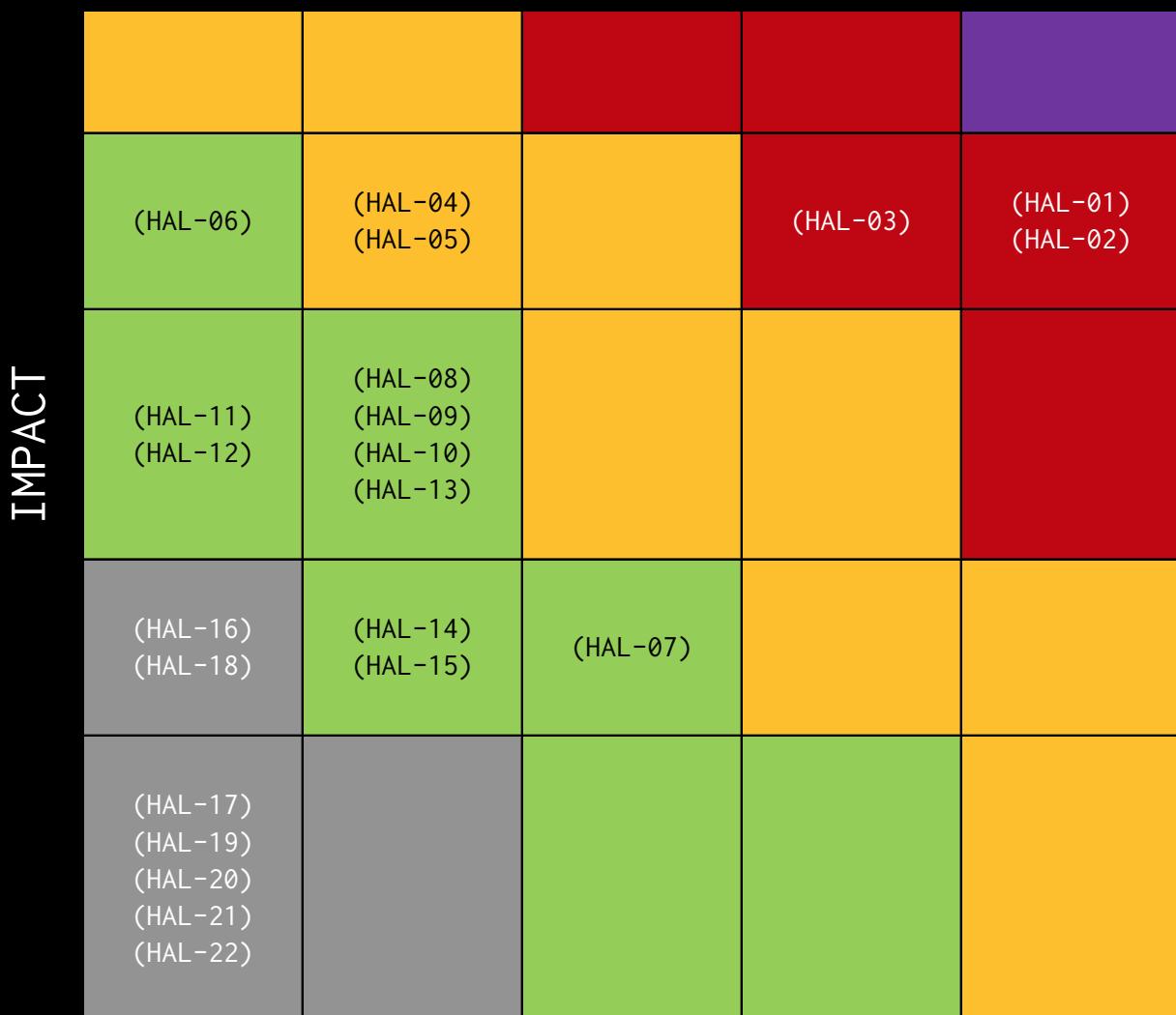
OUT-OF-SCOPE : External libraries and economics attacks

Halborn eventually reviewed the final version of the code to ensure that audit fixes carried over and that the new commits did not affect the security of the product. Code changes were reviewed in the `v1.0.0-rc1` branch, which was eventually merged into `v4main` which can be found at [GitHub](#).

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	3	2	10	7

LIKELIHOOD



EXECUTIVE OVERVIEW

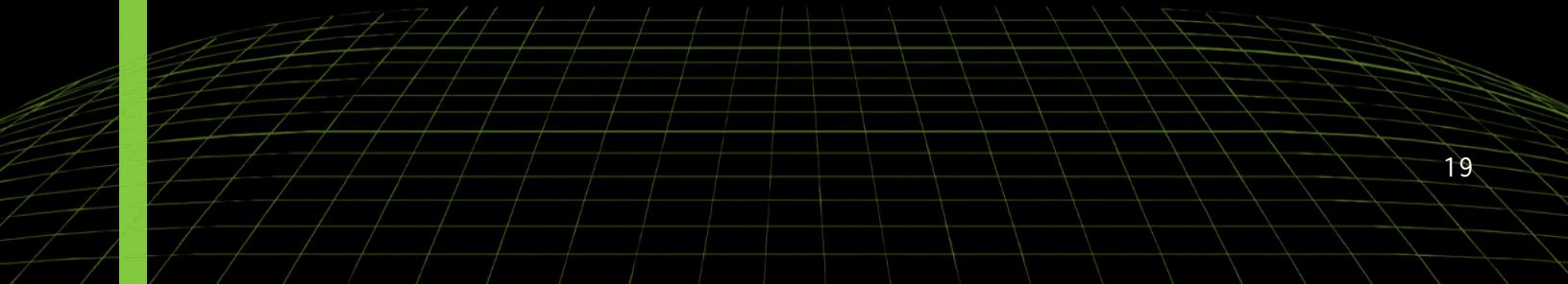
SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL01 - MINT ATTACK AFTER NFT TRANSFER	High	SOLVED - 03/23/2022
HAL02 - MINT ATTACK WITH WORTHLESS TOKEN	High	SOLVED - 03/23/2022
HAL03 - DT TOKEN STAKING NOT CALCULATED CORRECTLY	High	SOLVED - 03/24/2022
HAL04 - UNCHECKED TRANSFER	Medium	SOLVED - 12/13/2021
HAL05 - MULTIPLE EXTERNAL CALLS WITHIN LOOP MAY LEAD TO DENIAL OF SERVICE(DOS)	Medium	PARTIALLY SOLVED - 12/13/2021
HAL06 - RE-ENTRANCY PROTECTION	Low	PARTIALLY SOLVED - 12/13/2021
HAL07 - IGNORED RETURN VALUES	Low	PARTIALLY SOLVED - 12/13/2021
HAL08 - MISSING ZERO-ADDRESS CHECK	Low	SOLVED - 12/13/2021
HAL09 - DIVIDE BEFORE MULTIPLY	Low	SOLVED - 12/13/2021
HAL10 - USE OF BLOCK-TIMESTAMP	Low	RISK ACCEPTED
HAL11 - EXPERIMENTAL FEATURES ENABLED	Low	SOLVED - 12/13/2021
HAL12 - FLOATING PRAGMA	Low	SOLVED - 12/13/2021
HAL13 - OUTDATED DEPENDENCIES	Low	SOLVED - 12/13/2021
HAL14 - PRAGMA VERSION DEPRECATED	Low	SOLVED - 12/13/2021
HAL15 - MULTIPLE PRAGMA DEFINITIONS	Low	SOLVED - 12/13/2021
HAL16 - REDUNDANT BOOLEAN COMPARISON	Informational	SOLVED - 12/13/2021
HAL17 - USE OF INLINE ASSEMBLY	Informational	ACKNOWLEDGED

EXECUTIVE OVERVIEW

HAL18 - REDUNDANT VARIABLES	Informational	SOLVED - 12/13/2021
HAL19 - POSSIBLE MISUSE OF PUBLIC FUNCTIONS	Informational	PARTIALLY SOLVED - 12/13/2021
HAL20 - POTENTIAL UNSAFE CALCULATION	Informational	SOLVED - 12/13/2021
HAL21 - GAS OPTIMIZATIONS	Informational	SOLVED - 02/23/2022



FINDINGS & TECH DETAILS



3.1 (HAL-01) MINT ATTACK AFTER NFT TRANSFER - **HIGH**

Description:

In the `FixedRateExchange` and `Dispenser` contracts, users can transfer the ownership of NFT #1 to another user after having created a fixed rate exchange and dispenser instance. After transferring ownership, the roles were cleaned up; however, after the new owner created another fixed-rate exchange/dispenser instance, the original contract would once again have minting rights, allowing the first user to mint new tokens and collect base tokens.

Code Location:

The issues were present in the `FixedRateExchange.sol` and `Dispenser.sol` contracts.

Risk Level:

Likelihood - 5

Impact - 4

Recommendation:

As this was due to the permissions set out in the fixed-rate exchange and dispenser contracts, the issue could be remediated by enforcing ownership of the accounts instead of the contract.

Remediation Plan:

SOLVED: The issue was fixed by the team after refactoring the permissions of the `FixedRateExchanges` and `Dispenser` contracts to accept as owners only the owner of NFT #1 and users with `ERC20Deployer` roles.

3.2 (HAL-02) MINT ATTACK WITH WORTHLESS TOKEN - **HIGH**

Description:

In the `FixedRateExchange` contract, users can create a new fixed-rate exchange associated with a datatoken using a worthless ERC20 token as the base token. The attacker would be able to swap their tokens for datatokens and then sell the datatokens to the first exchange.

Risk Level:

Likelihood - 5

Impact - 4

Recommendation:

Since this was because any user could create a new fixed rate exchange, the issue could be solved by restricting unauthorized users from creating a fixed rate exchange.

Remediation Plan:

SOLVED: The issue was fixed by the team after refactoring the creation of the fixed-rate exchange functionality to only allow users with the role `ERC20Deployer` or the owner of NFT #1.

3.3 (HAL-03) DT TOKEN STAKE NOT CALCULATED SUCCESSFULLY - HIGH

Description:

Within the `BPool` contract, the order of instructions resulted in wrong calculations. After the users called the `joinSwapExternAmountIn` function, calculations were made on the provided supply and the pool share was minted. Later, the side staking bot (`ssBot`) would provide their share. However, the calculations were made on the `totalSupply` without taking into account that this would have changed after the shares were minting. This ultimately resulted in miscalculations in the future.

Code Location:

The issue was present in the `Bpool.sol` contract on line 1053, 1064 and 1042.

Listing 1

```

1  function joinswapExternAmountIn(
2      uint256 tokenAmountIn,
3      uint256 minPoolAmountOut
4  ) external _lock_ returns (uint256 poolAmountOut) {
5      //tokenIn = _baseTokenAddress;
6      require(_finalized, "ERR_NOT_FINALIZED");
7      _checkBound(_baseTokenAddress);
8      require(
9          tokenAmountIn <= bmul(_records[_baseTokenAddress].
L balance, MAX_IN_RATIO),
10         "ERR_MAX_IN_RATIO"
11     );
12     //ask ssContract
13     Record storage inRecord = _records[_baseTokenAddress];
14
15     poolAmountOut = calcPoolOutGivenSingleIn(
16         inRecord.balance,
17         inRecord.denorm,
18         _totalSupply,
19         _totalWeight,
```

```
20             tokenAmountIn
21         );
22
23         require(poolAmountOut >= minPoolAmountOut, "ERR_LIMIT_OUT"
24     );
25
26         inRecord.balance = badd(inRecord.balance, tokenAmountIn);
27
28         emit LOG_JOIN(msg.sender, _baseTokenAddress, tokenAmountIn
29             , block.timestamp);
29         emit LOG_BPT(poolAmountOut);
30         _mintPoolShare(poolAmountOut);
31         _pushPoolShare(msg.sender, poolAmountOut);
32
33         _pullUnderlying(_baseTokenAddress, msg.sender,
34             tokenAmountIn);
35
36         //ask the ssContract to stake as well
37         //calculate how much should the lss stake
38         Record storage ssInRecord = _records[_datatokenAddress];
39         uint256 ssAmountIn = calcSingleInGivenPoolOut(
40             ssInRecord.balance,
41             ssInRecord.denorm,
42             _totalSupply,
43             _totalWeight,
44             poolAmountOut
45         );
46         if (ssContract.canStake(_datatokenAddress, ssAmountIn)) {
47             //call lss to approve
48             ssContract.Stake(_datatokenAddress, ssAmountIn);
49             // follow the same path
50             ssInRecord.balance = badd(ssInRecord.balance,
51             ssAmountIn);
52             emit LOG_JOIN(
53                 _controller,
54                 _datatokenAddress,
55                 ssAmountIn,
56                 block.timestamp
57             );
58             emit LOG_BPT_SS(poolAmountOut);
59             _mintPoolShare(poolAmountOut);
60             _pushPoolShare(_controller, poolAmountOut);
61             _pullUnderlying(_datatokenAddress, _controller,
62                 ssAmountIn);
```

```
59         }
60         return poolAmountOut;
61     }
```

Risk Level:

Likelihood - 4

Impact - 4

Recommendation:

The issue could be solved by minting shares, after the ssBot provides its part to ensure the correct `totalSupply` amount is considered in the calculations.

Remediation Plan:

SOLVED: Ocean protocol amended the affected lines to ensure that the `totalSupply` argument used in the calculations matched between the user's staking data tokens and the side staking bot - [GitHub pull request fix](#).

3.4 (HAL-04) UNCHECKED TRANSFER - MEDIUM

Description:

In the `ERC20Tempalte.sol`, `FactoryRouter.sol` contracts, `Dispenser.sol`, `SideStaking.sol`, `BPool.sol`, `OPFCommunityFeeCollector.sol` and `FixedRateExchange.sol`, the return values of the external transfer calls are not checked. It should be noted that token is not reverted in case of failure and returns false. If one of these tokens is used, a deposit would not be reverted if the transfer fails, and an attacker could deposit tokens for free.

Code Location:

`FactoryRouter`

Listing 2: `FactoryRouter.sol` (Lines 275,299)

```
262     function buyDTBatch(
263         Operations[] calldata _operations
264     )
265     external {
266
267         for (uint i= 0; i< _operations.length; i++) {
268
269             [Redacted for brevity]
270             } else if (_operations[i].operation ==
271             ↳ operationType.SwapExactOut){
272                 // calculate how much amount In we need for
273                 ↳ exact Out
274                 uint amountIn = IPool(_operations[i].source)
275                 .getAmountInExactOut(_operations[i].tokenIn,
276                 ↳ _operations[i].tokenOut,_operations[i].amountsOut);
277                 // pull amount In from user
278                 IERC20(_operations[i].tokenIn).transferFrom(
279                 ↳ msg.sender,address(this),amountIn);
280                 // we approve pool to pull token from router
281                 IERC20(_operations[i].tokenIn).approve(
282                 ↳ _operations[i].source,amountIn);
```

```

278          // perform swap
279          IPool(_operations[i].source)
280          .swapExactAmountOut(_operations[i].tokenIn,
281          _operations[i].amountsIn,
282          _operations[i].tokenOut,
283          _operations[i].amountsOut,
284          _operations[i].maxPrice);
285          // send amount out back to user
286          require(IERC20(_operations[i].tokenOut)
287          .transfer(msg.sender, _operations[i].amountsOut
288          ), 'Failed MultiSwap');
289
290          } else if (_operations[i].operation ==
291          operationType.FixedRate) {
292              // get datatoken address
293              (, address datatoken, , , , , , ) =
294              IFixedRateExchange(_operations[i].source).
295              getExchange(_operations[i].exchangeIds);
296              // get tokenIn amount required for dt out
297              (uint baseTokenAmount, , , ) =
298              IFixedRateExchange(_operations[i].source).
299              calcBaseInGivenOutDT(_operations[i].
300              exchangeIds, _operations[i].amountsOut);
301
302              // pull tokenIn amount
303              IERC20(_operations[i].tokenIn).transferFrom(
304              msg.sender, address(this), baseTokenAmount);
305              // we approve pool to pull token from router
306              IERC20(_operations[i].tokenIn).approve(
307              _operations[i].source, baseTokenAmount);
308              // perform swap
309              IFixedRateExchange(_operations[i].source)
310              .buyDT(_operations[i].exchangeIds, _operations[
311              i].amountsOut, _operations[i].amountsIn);
312              // send dt out to user
313              IERC20(datatoken).transfer(msg.sender,
314              _operations[i].amountsOut);
315
316          } else {
317              IDispenser(_operations[i].source)
318              .dispense(_operations[i].tokenOut, _operations[
319              i].amountsOut, msg.sender);
320          }

```

FINDINGS & TECH DETAILS

```
313 }  
314 }  
315 }
```

Dispenser

Listing 3: Dispenser.sol (Line 248)

```

236     function ownerWithdraw(address datatoken) external{
237         require(
238             datatoken != address(0),
239             'Invalid token contract address'
240         );
241         require(
242             datatokens[datatoken].owner == msg.sender,
243             'Invalid owner'
244         );
245         IERC20Template tokenInstance = IERC20Template(datatoken);
246         uint256 ourBalance = tokenInstance.balanceOf(address(this))
247     );
248     if(ourBalance>0){
249         tokenInstance.transfer(msg.sender,ourBalance);
250         emit OwnerWithdrawed(datatoken, msg.sender, ourBalance
251     );
252 }

```

Dispenser

Listing 4: Dispenser.sol (Line 227)

```

187     function dispense(address datatoken, uint256 amount, address
188 destination) external payable{
189         require(
190             datatoken != address(0),
191             'Invalid token contract address'
192         );
193         require(
194             datatokens[datatoken].active == true,
195             'Dispenser not active'
196         );
197         require(
198             amount > 0,
199             'Invalid zero amount'
200         );
201         require(
202             datatokens[datatoken].maxTokens >= amount,
203             'Amount too high'
204         );
205 }

```

```

203     );
204     if(datatokens[datatoken].allowedSwapper != address(0)){
205         require(
206             datatokens[datatoken].allowedSwapper == msg.sender
207         ,
208             "This address is not allowed to request DT"
209         );
210     }
211
212     IERC20Template tokenInstance = IERC20Template(datatoken);
213     uint256 callerBalance = tokenInstance.balanceOf(
214         destination);
215     require(
216         callerBalance<datatokens[datatoken].maxBalance ,
217         'Caller balance too high'
218     );
219     uint256 ourBalance = tokenInstance.balanceOf(address(this))
220 );
221     if(ourBalance<amount && tokenInstance.isMinter(address(
222         this))){
223         //we need to mint the difference if we can
224         tokenInstance.mint(address(this),amount - ourBalance);
225         ourBalance = tokenInstance.balanceOf(address(this));
226     }
227     require(
228         ourBalance>=amount ,
229         'Not enough reserves'
230     );
231     tokenInstance.transfer(destination,amount);
232     emit TokensDispensed(datatoken, destination, amount);
233 }
```

FixedRateExchange

Listing 5: FixedRateExchange.sol (Lines 397,398,399,400)

```

394     } else {
395         exchanges[exchangeId].dtBalance = (exchanges[
396             exchangeId].dtBalance)
397             .sub(dataTokenAmount);
398         IERC20Template(exchanges[exchangeId].dataToken).
399         transfer(
400             msg.sender ,
```

```

399         dataTokenAmount
400     );
401 }
```

FixedRateExchange

Listing 6: FixedRateExchange.sol (Lines 477,478,479,480)

```

474     } else {
475         exchanges[exchangeId].btBalance = (exchanges[
476             exchangeId].btBalance)
477             .sub(baseTokenAmountBeforeFee);
478         IERC20Template(exchanges[exchangeId].baseToken).  

479         transfer(
480             msg.sender,
481             baseTokenAmount
482         );
483     }
```

FixedRateExchange

Listing 7: FixedRateExchange.sol (Lines 500,501,502,503)

```

496     onlyExchangeOwner(exchangeId)
497     {
498         uint256 amount = exchanges[exchangeId].btBalance;
499         exchanges[exchangeId].btBalance = 0;
500         IERC20Template(exchanges[exchangeId].baseToken).transfer(
501             exchanges[exchangeId].exchangeOwner,
502             amount
503         );
504     }
```

FixedRateExchange

Listing 8: FixedRateExchange.sol (Lines 519,520,521,522)

```

515     onlyExchangeOwner(exchangeId)
516     {
517         uint256 amount = exchanges[exchangeId].dtBalance;
518         exchanges[exchangeId].dtBalance = 0;
```

```

519         IERC20Template(exchanges[exchangeId].dataToken).transfer(
520             exchanges[exchangeId].exchangeOwner ,
521             amount
522         );
523

```

FixedRateExchange

Listing 9: FixedRateExchange.sol (Lines 536,537,538,539)

```

532     function collectMarketFee(bytes32 exchangeId) external {
533         // anyone call call this function, because funds are sent
534         ↳ to the correct address
535         uint256 amount = exchanges[exchangeId].marketFeeAvailable;
536         exchanges[exchangeId].marketFeeAvailable = 0;
537         IERC20Template(exchanges[exchangeId].baseToken).transfer(
538             exchanges[exchangeId].marketFeeCollector ,
539             amount
540         );
541         emit MarketFeeCollected(

```

FixedRateExchange

Listing 10: FixedRateExchange.sol (Lines 551,552,553,554)

```

548         // anyone call call this function, because funds are sent
549         ↳ to the correct address
550         uint256 amount = exchanges[exchangeId].oceanFeeAvailable;
551         exchanges[exchangeId].oceanFeeAvailable = 0;
552         IERC20Template(exchanges[exchangeId].baseToken).transfer(
553             opfCollector ,
554             amount
555         );
556         emit OceanFeeCollected(

```

SideStaking

Listing 11: SideStaking.sol (Lines 350,351,352,353)

```

345         IERC20Template lPTokens = IERC20Template(
346             _datatokens[datatokenAddress].poolAddress

```

```

347     );
348     uint256 lpBalance = lPTokens.balanceOf(address(this));
349     // uint256 balanceToTransfer = lpBalance.div(2);
350     lPTokens.transfer(
351         _datatokens[datatokenAddress].publisherAddress,
352         lpBalance.div(2)
353     );
354 }
```

SideStaking

Listing 12: SideStaking.sol (Line 390)

```

385             amount > 0 &&
386             _datatokens[datatokenAddress].datatokenBalance >=
387             amount
388             ) {
389                 IERC20Template dt = IERC20Template(datatokenAddress);
390                 _datatokens[datatokenAddress].vestingLastBlock = block
391                 .number;
392                 dt.transfer(_datatokens[datatokenAddress].
393                 publisherAddress, amount);
394                 _datatokens[datatokenAddress].datatokenBalance -=
395                 amount;
396                 _datatokens[datatokenAddress].vestingAmountSoFar +=
397                 amount;
398             }
399 }
```

BPool

Listing 13: BPool.sol (Line 257)

```

252     function collectOPF() external {
253         address[] memory tokens = getFinalTokens();
254         for (uint256 i = 0; i < tokens.length; i++) {
255             uint256 amount = communityFees[tokens[i]];
256             communityFees[tokens[i]] = 0;
257             IERC20(tokens[i]).transfer(_opfCollector, amount);
258         }
259     }
```

BPool

Listing 14: BPool.sol (Line 268)

```
261     function collectMarketFee(address to) external {
262         require(_marketCollector == msg.sender, "ONLY MARKET
263             COLLECTOR");
264
265         address[] memory tokens = getFinalTokens();
266         for (uint256 i = 0; i < tokens.length; i++) {
267             uint256 amount = marketFees[tokens[i]];
268             marketFees[tokens[i]] = 0;
269             IERC20(tokens[i]).transfer(to, amount);
270         }
271     }
```

Risk Level:

Likelihood - 2

Impact - 4

Recommendation:

It is recommended to use `SafeERC20`, or to ensure that the transfer return value is checked.

Remediation Plan:

SOLVED: The Ocean Protocol team amended the smart contracts to use the `safeTransfer` functions instead.

3.5 (HAL-05) MULTIPLE EXTERNAL CALLS WITHIN LOOP MAY LEAD TO DENIAL OF SERVICE(DOS) - MEDIUM

Description:

External calls within a loop increase Gas usage or can lead to a denial-of-service attack. In some discovered functions, there is a for loop where multiple external calls are executed. If the integer index variable within the loop evaluates at large numbers, a denial of service could occur.

Code Location:

ERC721Factory

Listing 15: ERC721Factory.sol (Lines 473,474,479,480,481,482,483,484,489,490,491,492,493,494,497,498,499,500,501,503,504,505,506,507,508,509,510)

```
466     function startMultipleTokenOrder(
467         tokenOrder[] memory orders
468     ) external {
469         uint256 ids = orders.length;
470         // TO DO. We can do better here , by grouppping
471         // publishMarketFeeTokens and consumeFeeTokens and have a single
472         for (uint256 i = 0; i < ids; i++) {
473             (address publishMarketFeeAddress, address
474             publishMarketFeeToken, uint256 publishMarketFeeAmount)
475             = IERC20Template(orders[i].tokenAddress).
476             getPublishingMarketFee();
477             // check if we have publishFees, if so transfer them
478             // to us and approve dttemplate to take them
479             if (publishMarketFeeAmount > 0 &&
480                 publishMarketFeeToken!=address(0)
481                 && publishMarketFeeAddress!=address(0)) {
```

```
479             require(IERC20Template(publishMarketFeeToken).  
480     transferFrom(  
481                 msg.sender,  
482                 address(this),  
483                 publishMarketFeeAmount  
484             ),'Failed to transfer publishFee');  
485             IERC20Template(publishMarketFeeToken).approve(  
486                 orders[i].tokenAddress, publishMarketFeeAmount);  
487             }  
488             // check if we have consumeFees , if so transfer them  
489             to us and approve dttemplate to take them  
490             if (orders[i].consumeFeeAmount > 0 && orders[i].  
491             consumeFeeToken!=address(0)  
492             && orders[i].consumeFeeAddress!=address(0)) {  
493             require(IERC20Template(orders[i].consumeFeeToken).  
494     transferFrom(  
495                 msg.sender,  
496                 address(this),  
497                 orders[i].consumeFeeAmount  
498             ),'Failed to transfer consumeFee');  
499             IERC20Template(orders[i].consumeFeeToken).approve(  
500                 orders[i].tokenAddress, orders[i].consumeFeeAmount);  
501             }  
502             // transfer erc20 datatoken from consumer to us  
503             require(IERC20Template(orders[i].tokenAddress).  
504     transferFrom(  
505                 msg.sender,  
506                 address(this),  
507                 orders[i].amount  
508             ),'Failed to transfer datatoken');  
509             IERC20Template(orders[i].tokenAddress).startOrder(  
510                 orders[i].consumer,  
511                 orders[i].amount,  
512                 orders[i].serviceId,  
513                 orders[i].consumeFeeAddress,  
514                 orders[i].consumeFeeToken,  
515                 orders[i].consumeFeeAmount  
516             );  
517         }  
518     }
```

Listing 16: FactoryRouter.sol (Lines 271,273,275,276,277,278,279,280,281,284,287,288,290,292,294,295,296,297,298,299,301,302,306,307,309,310,311,314,316,318,319,321,324,325)

```

262     function buyDTBatch(
263         Operations[] calldata _operations
264     )
265     external {
266
267         for (uint i= 0; i< _operations.length; i++) {
268
269             if(_operations[i].operation == operationType.
270             ↳ SwapExactIn) {
271                 // Get amountIn from user to router
272                 IERC20(_operations[i].tokenIn).transferFrom(
273                   ↳ msg.sender, address(this), _operations[i].amountsIn);
274                 // we approve pool to pull token from router
275                 IERC20(_operations[i].tokenIn).approve(
276                   ↳ _operations[i].source, _operations[i].amountsIn);
277                 // Perform swap
278                 (uint amountReceived,) =
279                   IPool(_operations[i].source)
280                     .swapExactAmountIn(_operations[i].tokenIn,
281                         _operations[i].amountsIn,
282                         _operations[i].tokenOut,
283                         _operations[i].amountsOut,
284                         _operations[i].maxPrice);
285                 // transfer token swapped to user
286
287                 require(IERC20(_operations[i].tokenOut)).
288                   ↳ transfer(msg.sender, amountReceived), 'Failed MultiSwap');
289             } else if (_operations[i].operation ==
290             ↳ operationType.SwapExactOut){
291                 // calculate how much amount In we need for
292                 ↳ exact Out
293                 uint amountIn = IPool(_operations[i].source)
294                   .getAmountInExactOut(_operations[i].tokenIn,
295                     ↳ _operations[i].tokenOut, _operations[i].amountsOut);
296                 // pull amount In from user
297                 IERC20(_operations[i].tokenIn).transferFrom(
298                   ↳ msg.sender, address(this), amountIn);
299                 // we approve pool to pull token from router
300                 IERC20(_operations[i].tokenIn).approve(
301                   ↳ _operations[i].source, amountIn);

```

```
293                         // perform swap
294                         IPool(_operations[i].source)
295                         .swapExactAmountOut(_operations[i].tokenIn,
296                         _operations[i].amountsIn,
297                         _operations[i].tokenOut,
298                         _operations[i].amountsOut,
299                         _operations[i].maxPrice);
300                         // send amount out back to user
301                         require(IERC20(_operations[i].tokenOut)
302                         .transfer(msg.sender, _operations[i].amountsOut
↳ ), 'Failed MultiSwap');
303
304             } else if (_operations[i].operation ==
305             operationType.FixedRate) {
306                 // get datatoken address
307                 (, address datatoken, , , , , , , ) =
308                 IFixedRateExchange(_operations[i].source).
309                 getExchange(_operations[i].exchangeIds);
310                 // get tokenIn amount required for dt out
311                 (uint baseTokenAmount, , , ) =
312                 IFixedRateExchange(_operations[i].source).
313                 calcBaseInGivenOutDT(_operations[i].
314                 exchangeIds, _operations[i].amountsOut);
315
316                 // pull tokenIn amount
317                 IERC20(_operations[i].tokenIn).transferFrom(
318                 msg.sender, address(this), baseTokenAmount);
319                 // we approve pool to pull token from router
320                 IERC20(_operations[i].tokenIn).approve(
321                 _operations[i].source, baseTokenAmount);
322                 // perform swap
323                 IFixedRateExchange(_operations[i].source)
324                 .buyDT(_operations[i].exchangeIds, _operations[
325                 i].amountsOut, _operations[i].amountsIn);
326                 // send dt out to user
327                 IERC20(datatoken).transfer(msg.sender,
328                 _operations[i].amountsOut);
329
330             } else {
331                 IDispenser(_operations[i].source)
332                 .dispense(_operations[i].tokenOut, _operations[
333                 i].amountsOut, msg.sender);
334
335         }
```

```

328 }
329
330 }
```

ERC721Template

Listing 17: ERC721Template.sol (Line 450)

```

448     function _cleanERC20Permissions(uint256 length) internal {
449         for (uint256 i = 0; i < length; i++) {
450             IERC20Template(deployedERC20List[i]).cleanFrom721();
451         }
452     }
```

BPool

Listing 18: BPool.sol (Line 257)

```

252     function collectOPF() external {
253         address[] memory tokens = getFinalTokens();
254         for (uint256 i = 0; i < tokens.length; i++) {
255             uint256 amount = communityFees[tokens[i]];
256             communityFees[tokens[i]] = 0;
257             IERC20(tokens[i]).transfer(_opfCollector, amount);
258         }
259     }
```

Listing 19: BPool.sol (Line 268)

```

261     function collectMarketFee(address to) external {
262         require(_marketCollector == msg.sender, "ONLY MARKET
263             COLLECTOR");
264
265         address[] memory tokens = getFinalTokens();
266         for (uint256 i = 0; i < tokens.length; i++) {
267             uint256 amount = marketFees[tokens[i]];
268             marketFees[tokens[i]] = 0;
269             IERC20(tokens[i]).transfer(to, amount);
270         }
271     }
```

Risk Level:

Likelihood - 2

Impact - 4

Recommendation:

If possible, use pull over push strategy for external calls. Please refer to the reference section for more details.

Reference:

[External Calls Recommendation](#)

Remediation Plan:

PARTIALLY SOLVED: The Ocean Protocol team amended loops to include a maximum amount of iterations for certain functions that can be called externally. The following functions were not modified:

- `CollectMarketFee` and `CollectOPF` in the `BPool` contract.
- `CleanERC20Permissions` in the `ERC721Template` contract.

3.6 (HAL-06) MISSING RE-ENTRANCY PROTECTION - LOW

Description:

Testing revealed that within some in-scope contracts, the non-reentrancy guard was missing for functions that handled token transfers. The following list presents the contracts and functions where the guard was missing.

- `BPool.sol`:
 - `exitswapExternAmountOut`, `exitswapPoolAmountIn`, `joinswapExternAmountIn`, `joinswapPoolAmountOut`, `setup`
- `Dispenser.sol`:
 - `dispense`, `ownerWithdraw`
- `ERC20Template.sol`:
 - `deployPool`
- `ERC721Factory.sol`:
 - `createNftErcWithDispenser`, `createNftErcWithFixedRate`, `createNftErcWithPool`, `createNftWithErc`, `deployERC721Contract`
- `ERC721Template.sol`:
 - `createERC20`
- `FixedRateExchange.sol`:
 - `buyDT`, `collectBT`, `collectDT`, `collectMarketFee`, `collectOceanFee`, `sellDT`
- `SideStaking.sol`:
 - `getVesting`, `newDataTokenCreated`, `notifyFinalize`, and `Stake`

Furthermore, in these function, persistent state reading and writing followed external calls, as well as emitting events after the call, making it vulnerable to a Reentrancy attack.

Code Location:

Listing 20: FixedRateExchange.sol (Lines 364,365,366,367,368,369,370,371,373,374,375,395,396)

```
333     function buyDT(bytes32 exchangeId, uint256 dataTokenAmount,
334         uint256 maxBaseTokenAmount)
335         external
336         onlyActiveExchange(exchangeId)
337     {
338         require(
339             dataTokenAmount != 0,
340             "FixedRateExchange: zero data token amount"
341         );
342         if(exchanges[exchangeId].allowedSwapper != address(0)){
343             require(
344                 exchanges[exchangeId].allowedSwapper == msg.sender
345             ,
346                 "FixedRateExchange: This address is not allowed to
347 swap"
348             );
349         }
350         (
351             uint256 baseTokenAmount,
352             uint256 baseTokenAmountBeforeFee,
353             uint256 oceanFeeAmount,
354             uint256 marketFeeAmount
355         ) = calcBaseInGivenOutDT(exchangeId, dataTokenAmount);
356         require(
357             baseTokenAmount <= maxBaseTokenAmount,
358             "FixedRateExchange: Too many base tokens"
359         );
360         // we account fees , fees are always collected in
361         basetoken
362         exchanges[exchangeId].oceanFeeAvailable = exchanges[
363             exchangeId
364             .oceanFeeAvailable
365             .add(oceanFeeAmount);
366         exchanges[exchangeId].marketFeeAvailable = exchanges[
367             exchangeId
368             .marketFeeAvailable
369             .add(marketFeeAmount);
370         require(
371             IERC20Template(exchanges[exchangeId].baseToken).
```

```
L  transferFrom(
366             msg.sender,
367             address(this), // we send basetoken to this
L  address, then exchange owner can withdraw
368             baseTokenAmount
369         ),
370         "FixedRateExchange: transferFrom failed in the
L  baseToken contract"
371     );
372
373     exchanges[exchangeId].btBalance = (exchanges[exchangeId].
L  btBalance).add(
374         baseTokenAmountBeforeFee
375     );
376
377     if (dataTokenAmount > exchanges[exchangeId].dtBalance) {
378         //first, let's try to mint
379         if(exchanges[exchangeId].withMint
380             && IERC20Template(exchanges[exchangeId].dataToken).
L  isMinter(address(this)))
381             {
382                 IERC20Template(exchanges[exchangeId].dataToken).
L  mint(msg.sender, dataTokenAmount);
383             }
384             else{
385                 require(
386                     IERC20Template(exchanges[exchangeId].dataToken
L  ).transferFrom(
387                         exchanges[exchangeId].exchangeOwner,
388                         msg.sender,
389                         dataTokenAmount
390                     ),
391                     "FixedRateExchange: transferFrom failed in the
L  dataToken contract"
392                 );
393             }
394         } else {
395             exchanges[exchangeId].dtBalance = (exchanges[
L  exchangeId].dtBalance)
396                 .sub(dataTokenAmount);
397             IERC20Template(exchanges[exchangeId].dataToken).
L  transfer(
398                 msg.sender,
399                 dataTokenAmount
```

```

400      );
401    }
402
403    emit Swapped(
404      exchangeId,
405      msg.sender,
406      baseTokenAmount,
407      dataTokenAmount,
408      exchanges[exchangeId].dataToken,
409      marketFeeAmount,
410      oceanFeeAmount
411    );
412  }

```

Listing 21: FixedRateExchange.sol (Lines 452,453,454,455,456,457,458,459,461,462,463,475,476)

```

421   function sellDT(bytes32 exchangeId, uint256 dataTokenAmount,
422   ↳ uint256 minBaseTokenAmount)
423     external
424     onlyActiveExchange(exchangeId)
425   {
426     require(
427       dataTokenAmount != 0,
428       "FixedRateExchange: zero data token amount"
429     );
430     if(exchanges[exchangeId].allowedSwapper != address(0)){
431       require(
432         exchanges[exchangeId].allowedSwapper == msg.sender
433       ,
434         "FixedRateExchange: This address is not allowed to
435       ↳ swap"
436       );
437     }
438     (
439       uint256 baseTokenAmount,
440       uint256 baseTokenAmountBeforeFee,
441       uint256 oceanFeeAmount,
442       uint256 marketFeeAmount
443     ) = calcBaseOutGivenInDT(exchangeId, dataTokenAmount);
444     require(
445       baseTokenAmount >= minBaseTokenAmount,
446       "FixedRateExchange: Too few base tokens"
447     );

```

```
444     );
445     // we account fees , fees are always collected in
446     basetoken
447     exchanges[exchangeId].oceanFeeAvailable = exchanges[
448     exchangeId]
449         .oceanFeeAvailable
450         .add(oceanFeeAmount);
451     exchanges[exchangeId].marketFeeAvailable = exchanges[
452     exchangeId]
453         .marketFeeAvailable
454         .add(marketFeeAmount);
455     require(
456         IERC20Template(exchanges[exchangeId].dataToken).  

457     transferFrom(
458         msg.sender,
459         address(this),
460         dataTokenAmount
461     ),
462         "FixedRateExchange: transferFrom failed in the
463     dataToken contract"
464     );
465     exchanges[exchangeId].dtBalance = (exchanges[exchangeId].  

466     dtBalance).add(
467         dataTokenAmount
468     );
469     if (baseTokenAmount > exchanges[exchangeId].btBalance) {
470         require(
471             IERC20Template(exchanges[exchangeId].baseToken).  

472     transferFrom(
473         exchanges[exchangeId].exchangeOwner,
474         msg.sender,
475         baseTokenAmount
476     ),
477         "FixedRateExchange: transferFrom failed in the
478     baseToken contract"
479     );
480     } else {
481         exchanges[exchangeId].btBalance = (exchanges[  

482     exchangeId].btBalance)
483             .sub(baseTokenAmountBeforeFee);
484         IERC20Template(exchanges[exchangeId].baseToken).  

485     transfer(
```

```

478         msg.sender,
479         baseTokenAmount
480     );
481 }
482
483 emit Swapped(
484     exchangeId,
485     msg.sender,
486     baseTokenAmount,
487     dataTokenAmount,
488     exchanges[exchangeId].baseToken,
489     marketFeeAmount,
490     oceanFeeAmount
491 );
492 }
```

Listing 22: SideStaking.sol (Lines 258,259)

```

245 function Stake(
246     address datatokenAddress,
247     address stakeToken,
248     uint256 amount
249 ) public {
250     if (_datatokens[datatokenAddress].bound != true) return;
251     require(
252         msg.sender == _datatokens[datatokenAddress].
253         poolAddress,
254         "ERR: Only pool can call this"
255     );
256     bool ok = canStake(datatokenAddress, stakeToken, amount);
257     if (ok != true) return;
258     IERC20Template dt = IERC20Template(datatokenAddress);
259     dt.approve(_datatokens[datatokenAddress].poolAddress,
260     amount);
261     _datatokens[datatokenAddress].datatokenBalance -= amount;
262 }
```

Listing 23: SideStaking.sol (Lines 390,391,392)

```

357     function getVesting(address datatokenAddress) public {
358         require(
359             _datatokens[datatokenAddress].bound == true,
360             "ERR: Invalid datatoken"
```

```
361     );
362     // is this needed?
363     // require(msg.sender == _datatokens[datatokenAddress].
364     ↳ publisherAddress,'ERR: Only publisher can call this');
365     //calculate how many tokens we need to vest to publisher<<
366     uint256 blocksPassed;
367
368     if (_datatokens[datatokenAddress].vestingEndBlock < block.
369     ↳ number) {
370         blocksPassed =
371             _datatokens[datatokenAddress].vestingEndBlock -
372             _datatokens[datatokenAddress].vestingLastBlock;
373     } else {
374         blocksPassed =
375             block.number -
376             _datatokens[datatokenAddress].vestingLastBlock;
377     }
378
379     uint256 vestPerBlock = _datatokens[datatokenAddress].
380     ↳ vestingAmount.div(
381         _datatokens[datatokenAddress].vestingEndBlock -
382             _datatokens[datatokenAddress].blockDeployed
383     );
384     if (vestPerBlock == 0) return;
385     uint256 amount = blocksPassed.mul(vestPerBlock);
386     if (
387         amount > 0 &&
388         _datatokens[datatokenAddress].datatokenBalance >=
389         ↳ amount
390     ) {
391         IERC20Template dt = IERC20Template(datatokenAddress);
392         _datatokens[datatokenAddress].vestingLastBlock = block.
393         ↳ number;
394         dt.transfer(_datatokens[datatokenAddress].
395         ↳ publisherAddress, amount);
396         _datatokens[datatokenAddress].datatokenBalance -=
397         ↳ amount;
398         _datatokens[datatokenAddress].vestingAmountSoFar +=
399         ↳ amount;
400     }
401 }
```

Listing 24: SideStaking.sol (Lines 322,326,327,328,329,333,334,335,336,337,338,339,340,342,343)

```
302     function notifyFinalize(address datatokenAddress, uint256
303         decimals)
304     internal
305     {
306         if (_datatokens[datatokenAddress].bound != true) return;
307         if (_datatokens[datatokenAddress].poolFinalized == true)
308             return;
309         _datatokens[datatokenAddress].poolFinalized = true;
310         uint256 baseTokenWeight = 5 * BASE; //pool weight: 50-50
311         uint256 dataTokenWeight = 5 * BASE; //pool weight: 50-50
312         uint256 baseTokenAmount = _datatokens[datatokenAddress]
313             .basetokenBalance;
314         //given the price, compute dataTokenAmount
315
316         uint256 dataTokenAmount = ((_datatokens[datatokenAddress].
317             rate *
318                 baseTokenAmount *
319                     dataTokenWeight) /
320                         baseTokenWeight /
321                             BASE) * (10**18 - decimals);
322
323         //approve the tokens and amounts
324         IERC20Template dt = IERC20Template(datatokenAddress);
325         dt.approve(_datatokens[datatokenAddress].poolAddress,
326             dataTokenAmount);
327         IERC20Template dtBase = IERC20Template(
328             _datatokens[datatokenAddress].basetokenAddress
329         );
330
331         // call the pool, bind the tokens, set the price, finalize
332         pool
333             IPool pool = IPool(_datatokens[datatokenAddress].
334             poolAddress);
335             pool.setup(
336                 datatokenAddress,
337                 dataTokenAmount,
338                 dataTokenWeight,
339                 _datatokens[datatokenAddress].basetokenAddress,
```

```

338         baseTokenAmount ,
339         baseTokenWeight
340     );
341     //subtract
342     _datatokens[datatokenAddress].basetokenBalance -=
↳ baseTokenAmount;
343     _datatokens[datatokenAddress].datatokenBalance -=
↳ dataTokenAmount;
344     // send 50% of the pool shares back to the publisher
345     IERC20Template lPTokens = IERC20Template(
346         _datatokens[datatokenAddress].poolAddress
347     );
348     uint256 lpBalance = lPTokens.balanceOf(address(this));
349     // uint256 balanceToTransfer = lpBalance.div(2);
350     lPTokens.transfer(
351         _datatokens[datatokenAddress].publisherAddress ,
352         lpBalance.div(2)
353     );
354 }
```

Listing 25: Dispenser.sol (Lines 220,227,228)

```

187     function dispense(address datatoken, uint256 amount, address
↳ destination) external payable{
188         require(
189             datatoken != address(0),
190             'Invalid token contract address'
191         );
192         require(
193             datatokens[datatoken].active == true,
194             'Dispenser not active'
195         );
196         require(
197             amount > 0,
198             'Invalid zero amount'
199         );
200         require(
201             datatokens[datatoken].maxTokens >= amount,
202             'Amount too high'
203         );
204         if(datatokens[datatoken].allowedSwapper != address(0)){
205             require(
206                 datatokens[datatoken].allowedSwapper == msg.sender
↳ ,
```

```

207             "This address is not allowed to request DT"
208         );
209     }
210
211     IERC20Template tokenInstance = IERC20Template(datatoken);
212     uint256 callerBalance = tokenInstance.balanceOf(
213         destination);
214     require(
215         callerBalance<datatokens[datatoken].maxBalance ,
216         'Caller balance too high'
217     );
218     uint256 ourBalance = tokenInstance.balanceOf(address(this))
219     );
220     if(ourBalance<amount && tokenInstance.isMinter(address(
221         this))){
222         //we need to mint the difference if we can
223         tokenInstance.mint(address(this),amount - ourBalance);
224         ourBalance = tokenInstance.balanceOf(address(this));
225     }
226     require(
227         ourBalance>=amount ,
228         'Not enough reserves'
229     );
230     tokenInstance.transfer(destination ,amount);
231     emit TokensDispensed(datatoken , destination , amount);
232 }
```

Listing 26: Dispenser.sol (Lines 248,249)

```

236     function ownerWithdraw(address datatoken) external{
237         require(
238             datatoken != address(0),
239             'Invalid token contract address'
240         );
241         require(
242             datatokens[datatoken].owner == msg.sender ,
243             'Invalid owner'
244         );
245         IERC20Template tokenInstance = IERC20Template(datatoken);
246         uint256 ourBalance = tokenInstance.balanceOf(address(this))
247     );
248     if(ourBalance>0){
249         tokenInstance.transfer(msg.sender ,ourBalance);
250         emit OwnerWithdrawed(datatoken , msg.sender , ourBalance
251     );
```

```

↳ );
250     }
251 }
```

Listing 27: FixedRateExchange.sol (Lines 364,365,366,367,368,369,370,371,382,385,386,387,388,389,390,391,392,397,398,399,400,403,404,405,406,407,408,409,410,411)

```

333     function buyDT(bytes32 exchangeId, uint256 dataTokenAmount,
↳ uint256 maxBaseTokenAmount)
334         external
335         onlyActiveExchange(exchangeId)
336     {
337         require(
338             dataTokenAmount != 0,
339             "FixedRateExchange: zero data token amount"
340         );
341         if(exchanges[exchangeId].allowedSwapper != address(0)){
342             require(
343                 exchanges[exchangeId].allowedSwapper == msg.sender
↳ ,
344                 "FixedRateExchange: This address is not allowed to
↳ swap"
345             );
346         }
347         (
348             uint256 baseTokenAmount,
349             uint256 baseTokenAmountBeforeFee,
350             uint256 oceanFeeAmount,
351             uint256 marketFeeAmount
352         ) = calcBaseInGivenOutDT(exchangeId, dataTokenAmount);
353         require(
354             baseTokenAmount <= maxBaseTokenAmount,
355             "FixedRateExchange: Too many base tokens"
356         );
357         // we account fees , fees are always collected in
↳ basetoken
358         exchanges[exchangeId].oceanFeeAvailable = exchanges[
↳ exchangeId]
359             .oceanFeeAvailable
360             .add(oceanFeeAmount);
361         exchanges[exchangeId].marketFeeAvailable = exchanges[
↳ exchangeId]
```

```
362         .marketFeeAvailable
363         .add(marketFeeAmount);
364     require(
365         IERC20Template(exchanges[exchangeId].baseToken) .
366     transferFrom(
367         msg.sender,
368         address(this), // we send basetoken to this
369         baseTokenAmount
370     ),
371         "FixedRateExchange: transferFrom failed in the
372     baseToken contract"
373 );
374
375     exchanges[exchangeId].btBalance = (exchanges[exchangeId] .
376     btBalance).add(
377         baseTokenAmountBeforeFee
378     );
379
380     if (dataTokenAmount > exchanges[exchangeId].dtBalance) {
381         //first, let's try to mint
382         if(exchanges[exchangeId].withMint
383             && IERC20Template(exchanges[exchangeId].dataToken) .
384         isMinter(address(this)))
385         {
386             IERC20Template(exchanges[exchangeId].dataToken) .
387         mint(msg.sender, dataTokenAmount);
388         }
389         else{
390             require(
391                 IERC20Template(exchanges[exchangeId].dataToken) .
392         ).transferFrom(
393             exchanges[exchangeId].exchangeOwner,
394             msg.sender,
395             dataTokenAmount
396         ),
397             "FixedRateExchange: transferFrom failed in the
398         dataToken contract"
399         );
400     }
401 }
```

```
397             IERC20Template(exchanges[exchangeId].dataToken).  
398     transfer(  
399             msg.sender,  
400             dataTokenAmount  
401         );  
402     }  
403     emit Swapped(  
404         exchangeId,  
405         msg.sender,  
406         baseTokenAmount,  
407         dataTokenAmount,  
408         exchanges[exchangeId].dataToken,  
409         marketFeeAmount,  
410         oceanFeeAmount  
411     );  
412 }
```

Listing 28: FixedRateExchange.sol (Lines 500,501,502,503,505,506,507,508,509,510)

```
494     function collectBT(bytes32 exchangeId)  
495         external  
496         onlyExchangeOwner(exchangeId)  
497     {  
498         uint256 amount = exchanges[exchangeId].btBalance;  
499         exchanges[exchangeId].btBalance = 0;  
500         IERC20Template(exchanges[exchangeId].baseToken).transfer(  
501             exchanges[exchangeId].exchangeOwner,  
502             amount  
503         );  
504         emit TokenCollected(  
505             exchangeId,  
506             exchanges[exchangeId].exchangeOwner,  
507             exchanges[exchangeId].baseToken,  
508             amount  
509         );  
510     }  
511 }
```

Listing 29: FixedRateExchange.sol (Lines 519,520,521,522,524,525,526,527,528,529)

```
513     function collectDT(bytes32 exchangeId)
514         external
515             onlyExchangeOwner(exchangeId)
516     {
517         uint256 amount = exchanges[exchangeId].dtBalance;
518         exchanges[exchangeId].dtBalance = 0;
519         IERC20Template(exchanges[exchangeId].dataToken).transfer(
520             exchanges[exchangeId].exchangeOwner,
521             amount
522         );
523
524         emit TokenCollected(
525             exchangeId,
526             exchanges[exchangeId].exchangeOwner,
527             exchanges[exchangeId].dataToken,
528             amount
529         );
530     }
```

Listing 30: FixedRateExchange.sol (Lines 536,537,538,539,540,541,542,543,544)

```
532     function collectMarketFee(bytes32 exchangeId) external {
533         // anyone can call this function, because funds are sent
534         // to the correct address
535         uint256 amount = exchanges[exchangeId].marketFeeAvailable;
536         exchanges[exchangeId].marketFeeAvailable = 0;
537         IERC20Template(exchanges[exchangeId].baseToken).transfer(
538             exchanges[exchangeId].marketFeeCollector,
539             amount
540         );
541         emit MarketFeeCollected(
542             exchangeId,
543             exchanges[exchangeId].baseToken,
544             amount
545         );
546     }
```

Listing 31: FixedRateExchange.sol (Lines 551,552,553,554,555,556,557,558,559)

```

547     function collectOceanFee(bytes32 exchangeId) external {
548         // anyone can call this function, because funds are sent
549         // to the correct address
550         uint256 amount = exchanges[exchangeId].oceanFeeAvailable;
551         exchanges[exchangeId].oceanFeeAvailable = 0;
552         IERC20Template(exchanges[exchangeId].baseToken).transfer(
553             opfCollector,
554             amount
555         );
556         emit OceanFeeCollected(
557             exchangeId,
558             exchanges[exchangeId].baseToken,
559             amount
560         );

```

Listing 32: FixedRateExchange.sol (Lines 452,453,454,455,456,457,458,459,466,467,468,469,470,471,472,473,477,478,479,480,483,484,485,486,487,488,489,490,491)

```

421     function sellDT(bytes32 exchangeId, uint256 dataTokenAmount,
422         uint256 minBaseTokenAmount)
423         external
424         onlyActiveExchange(exchangeId)
425     {
426         require(
427             dataTokenAmount != 0,
428             "FixedRateExchange: zero data token amount"
429         );
430         if(exchanges[exchangeId].allowedSwapper != address(0)){
431             require(
432                 exchanges[exchangeId].allowedSwapper == msg.sender
433             ,
434                 "FixedRateExchange: This address is not allowed to
435                 swap"
436             );
437         }

```

```
438         uint256 oceanFeeAmount,
439         uint256 marketFeeAmount
440     ) = calcBaseOutGivenInDT(exchangeId, dataTokenAmount);
441     require(
442         baseTokenAmount >= minBaseTokenAmount,
443         "FixedRateExchange: Too few base tokens"
444     );
445     // we account fees , fees are always collected in
446     basetoken
447     exchanges[exchangeId].oceanFeeAvailable = exchanges[
448     exchangeId]
449         .oceanFeeAvailable
450         .add(oceanFeeAmount);
451     exchanges[exchangeId].marketFeeAvailable = exchanges[
452     exchangeId]
453         .marketFeeAvailable
454         .add(marketFeeAmount);
455     require(
456         IERC20Template(exchanges[exchangeId].dataToken).  

457     transferFrom(
458         msg.sender,
459         address(this),
460         dataTokenAmount
461     ),
462         "FixedRateExchange: transferFrom failed in the
463     dataToken contract"
464 );
465     exchanges[exchangeId].dtBalance = (exchanges[exchangeId].  

466     dtBalance).add(
467         dataTokenAmount
468 );
469     if (baseTokenAmount > exchanges[exchangeId].btBalance) {
470         require(
471             IERC20Template(exchanges[exchangeId].baseToken).  

472     transferFrom(
473         exchanges[exchangeId].exchangeOwner,
474         msg.sender,
475         baseTokenAmount
476     ),
477         "FixedRateExchange: transferFrom failed in the
478     baseToken contract"
479 );
```

```

474         } else {
475             exchanges[exchangeId].btBalance = (exchanges[
476 ↳ exchangeId].btBalance)
477                 .sub(baseTokenAmountBeforeFee);
478             IERC20Template(exchanges[exchangeId].baseToken).  

479             transfer(  

480                 msg.sender,  

481                 baseTokenAmount  

482             );
483
484             emit Swapped(  

485                 exchangeId,  

486                 msg.sender,  

487                 baseTokenAmount,  

488                 dataTokenAmount,  

489                 exchanges[exchangeId].baseToken,  

490                 marketFeeAmount,  

491                 oceanFeeAmount  

492             );
493         }

```

Listing 33: SideStaking.sol (Lines 104,109,110,111,112,113,114,115,116
,117,118,119,120,121,122,123,124,125)

```

74     function newDataTokenCreated(
75         address datatokenAddress,
76         address basetokenAddress,
77         address poolAddress,
78         address publisherAddress,
79         uint256[] memory ssParams
80     ) external onlyRouter returns (bool) {
81         //check if we are the controller of the pool
82         require(poolAddress != address(0), "Invalid poolAddress");
83         IPool bpool = IPool(poolAddress);
84         require(
85             bpool.getController() == address(this),
86             "We are not the pool controller"
87         );
88         //check if the tokens are bound
89         require(
90             bpool.getDataTokenAddress() == datatokenAddress,
91             "DataToken address mismatch"

```

FINDINGS & TECH DETAILS

```
92     );
93     require(
94         bpool.getBaseTokenAddress() == basetokenAddress ,
95         "BaseToken address mismatch"
96     );
97     // check if we are the minter of DT
98     IERC20Template dt = IERC20Template(datatokenAddress);
99     require(
100         (dt.permissions(address(this))).minter == true ,
101         "BaseToken address mismatch"
102     );
103    // get cap and mint it..
104    dt.mint(address(this), dt.cap());
105
106    require(dt.balanceOf(address(this)) == dt.totalSupply(), "
107 Mint failed");
107    require(dt.totalSupply().div(10) >= ssParams[2], "Max
108 vesting 10%");
108    //we are rich :)let's setup the records and we are good to
109 go
109    _datatokens[datatokenAddress] = Record({
110        bound: true ,
111        basetokenAddress: basetokenAddress ,
112        poolAddress: poolAddress ,
113        poolFinalized: false ,
114        datatokenBalance: dt.totalSupply() - ssParams[2], //
115 We need to remove the vesting amount from that
115        datatokenCap: dt.cap(),
116        basetokenBalance: ssParams[4],
117        lastPrice: 0,
118        rate: ssParams[0],
119        publisherAddress: publisherAddress ,
120        blockDeployed: block.number ,
121        vestingEndBlock: block.number + ssParams[3],
122        vestingAmount: ssParams[2],
123        vestingLastBlock: block.number ,
124        vestingAmountSoFar: 0
125    });
126
127    notifyFinalize(datatokenAddress, ssParams[1]);
128
129    return (true);
130 }
```

Risk Level:

Likelihood - 1

Impact - 4

Recommendation:

To protect against cross-functional reentrancy attacks, it may be necessary to use a mutex. By using this lock, an attacker can no longer exploit the function with a recursive call. OpenZeppelin has its own mutex implementation called ReentrancyGuard which provides a modifier to any function called `nonReentrant` that guards the function with a mutex against the Reentrancy attacks.

Ocean Protocol should consider refactoring the code to follow the checks-effects-interactions pattern and use the `nonReentrant` modifier. See the references' section for more information.

References:

[Checks-Effect-Interactions pattern](#)

Remediation Plan:

PARTIALLY SOLVED: The Ocean Protocol team added the reentrancy protection guard on all external functions that any user could call. It was decided not to implement protection on internal functions and functions that can be called by a privileged user. An additional function where guard was not implemented was `deployErc721Contract` within the `ERC721Factory` contract.

3.7 (HAL-07) IGNORED RETURN VALUES - LOW

Description:

In the `ERC721Factory.sol`, `SideStaking.sol`, `BFactory.sol` and `FactoryRouter.sol` contract, there are instances where external methods are called, and the return values are not stored in a local or state variable.

Code Location:

`ERC721Factory`

Listing 34: ERC721Factory.sol (Line 484)

```
479         require(IERC20Template(publishMarketFeeToken).  
480             transferFrom(  
481                 msg.sender,  
482                 address(this),  
483                 publishMarketFeeAmount  
484             ),'Failed to transfer publishFee');  
484             IERC20Template(publishMarketFeeToken).approve(  
485             orders[i].tokenAddress, publishMarketFeeAmount);  
485         }
```

`ERC721Factory`

Listing 35: ERC721Factory.sol (Line 494)

```
489         require(IERC20Template(orders[i].consumeFeeToken).  
490             transferFrom(  
491                 msg.sender,  
492                 address(this),  
493                 orders[i].consumeFeeAmount  
494             ),'Failed to transfer consumeFee');  
494             IERC20Template(orders[i].consumeFeeToken).approve(  
495             orders[i].tokenAddress, orders[i].consumeFeeAmount);  
495         }
```

ERC721Factory

Listing 36: ERC721Factory.sol (Line 598)

```

597         // allow router to take the liquidity
598         IERC20Template(_PoolData.addresses[1]).approve(router,
599         ↳ _PoolData.ssParams[4]);
600
601         poolAddress = IERC20Template(erc20Address).deployPool(
602             _PoolData.ssParams,
603             _PoolData.swapFees,
604             _PoolData.addresses
605         );

```

FactoryRouter

Listing 37: FactoryRouter.sol (Lines 273,292,294,295,296,297,298,299,316,321)

```

262     function buyDTBatch(
263         Operations[] calldata _operations
264     )
265     external {
266
267         for (uint i= 0; i< _operations.length; i++) {
268
269             if(_operations[i].operation == operationType.
270             ↳ SwapExactIn) {
271
272                 // Get amountIn from user to router
273                 IERC20(_operations[i].tokenIn).transferFrom(
274                 ↳ msg.sender, address(this), _operations[i].amountsIn);
275
276                 // we approve pool to pull token from router
277                 IERC20(_operations[i].tokenIn).approve(
278                 ↳ _operations[i].source, _operations[i].amountsIn);
279
280                 // Perform swap
281                 (uint amountReceived,) =
282                 IPool(_operations[i].source)
283                     .swapExactAmountIn(_operations[i].tokenIn,
284                         _operations[i].amountsIn,
285                         _operations[i].tokenOut,
286                         _operations[i].amountsOut,
287                         _operations[i].maxPrice);
288
289                 // transfer token swapped to user

```

```

283
284             require(IERC20(_operations[i].tokenOut).
285     transfer(msg.sender, amountReceived), 'Failed MultiSwap');
286         } else if (_operations[i].operation ==
287             operationType.SwapExactOut){
288             // calculate how much amount In we need for
289             // exact Out
290             uint amountIn = IPool(_operations[i].source)
291                 .getAmountInExactOut(_operations[i].tokenIn,
292                     _operations[i].tokenOut, _operations[i].amountsOut);
293             // pull amount In from user
294             IERC20(_operations[i].tokenIn).transferFrom(
295                 msg.sender, address(this), amountIn);
296             // we approve pool to pull token from router
297             IERC20(_operations[i].tokenIn).approve(
298                 _operations[i].source, amountIn);
299             // perform swap
300             IPool(_operations[i].source)
301                 .swapExactAmountOut(_operations[i].tokenIn,
302                     _operations[i].amountsIn,
303                     _operations[i].tokenOut,
304                     _operations[i].amountsOut,
305                     _operations[i].maxPrice);
306             // send amount out back to user
307             require(IERC20(_operations[i].tokenOut)
308                 .transfer(msg.sender, _operations[i].amountsOut
309 ), 'Failed MultiSwap');

310         } else if (_operations[i].operation ==
311             operationType.FixedRate) {
312             // get datatoken address
313             (, address datatoken, , , , , , ) =
314                 IFixedRateExchange(_operations[i].source).
315                 getExchange(_operations[i].exchangeIds);
316             // get tokenIn amount required for dt out
317             (uint baseTokenAmount, , , ) =
318                 IFixedRateExchange(_operations[i].source).
319                 calcBaseInGivenOutDT(_operations[i].
320                     exchangeIds, _operations[i].amountsOut);
321
322             // pull tokenIn amount
323             IERC20(_operations[i].tokenIn).transferFrom(
324                 msg.sender, address(this), baseTokenAmount);
325             // we approve pool to pull token from router

```

```

316             IERC20(_operations[i].tokenIn).approve(
317             _operations[i].source, baseTokenAmount);
318             // perform swap
319             IFixedRateExchange(_operations[i].source)
320             .buyDT(_operations[i].exchangeIds, _operations[
321             i].amountsOut, _operations[i].amountsIn);
322             // send dt out to user
323             IERC20(datatoken).transfer(msg.sender,
324             _operations[i].amountsOut);
325
326             } else {
327             IDispenser(_operations[i].source)
328             .dispense(_operations[i].tokenOut, _operations[
329             i].amountsOut, msg.sender);
330         }

```

SideStaking

Listing 38: SideStaking.sol (Line 258)

```

245     function Stake(
246         address datatokenAddress,
247         address stakeToken,
248         uint256 amount
249     ) public {
250         if (_datatokens[datatokenAddress].bound != true) return;
251         require(
252             msg.sender == _datatokens[datatokenAddress].
253             poolAddress,
254             "ERR: Only pool can call this"
255         );
256         bool ok = canStake(datatokenAddress, stakeToken, amount);
257         if (ok != true) return;
258         IERC20Template dt = IERC20Template(datatokenAddress);
259         dt.approve(_datatokens[datatokenAddress].poolAddress,
260             amount);
261         _datatokens[datatokenAddress].datatokenBalance -= amount;
262     }

```

SideStaking

Listing 39: SideStaking.sol (Lines 322,326,327,328,329)

```
320      //approve the tokens and amounts
321      IERC20Template dt = IERC20Template(datatokenAddress);
322      dt.approve(_datatokens[datatokenAddress].poolAddress,
323      ↳ dataTokenAmount);
323      IERC20Template dtBase = IERC20Template(
324          _datatokens[datatokenAddress].basetokenAddress
325      );
326      dtBase.approve(
327          _datatokens[datatokenAddress].poolAddress,
328          baseTokenAmount
329      );
```

BFactory

Listing 40: BFactory.sol (Lines 117,118,119,120,121,122)

```
116      // requires approval first from basetokenSender
117      ISideStaking(addresses[0]).newDataTokenCreated(
118          tokens[0],
119          tokens[1],
120          bpool,
121          addresses[3], //publisherAddress
122          ssParams);
123
124      return bpool;
```

Risk Level:

Likelihood - 3

Impact - 2

Recommendation:

It is recommended to add return values check to avoid unexpected contract crashes. This can be implemented with a `require` statement and will help handle exceptions more comprehensively.

Remediation Plan:

PARTIALLY SOLVED: The Ocean Protocol team amended certain functionalities to include return value checking; however, the following functions were called without checking the return value:

- newDataTokenCreated: BFactory contract
- swapExactAmountOut function called within the buyDTBatch function : FactoryRouter contract

3.8 (HAL-08) MISSING ZERO-ADDRESS CHECK - LOW

Description:

There are several instances where address validation is missing. For instance, zero address validation failure has been found when assigning user-supplied address values to state variables directly.

The following list shows all the instances where zero address check failure was identified:

- ERC721Factory.sol: variable `_router`
- FactoryRouter.sol: variables `_routerOwner`, `_opfCollector` within the constructor, and variable `_factory` within function `addFactory`
- ERC20Template.sol, variables `_publishMarketFeeAddress` and `_publishMarketFeeToken` within function `setPublishingMarketFee`; variable `_newFeeCollector` within function `setFeeCollector`
- SideStaking.sol, variable `_router` within the constructor
- BPool.sol variable `_newCollector` within `updateMarketFeeCollector`

Code Location:

ERC721Factory

Listing 41: ERC721Factory.sol (Lines 83,93)

```
79     constructor(
80         address _template721,
81         address _template,
82         address _collector,
83         address _router
84     ) {
85         require(
86             _template != address(0) &&
87             _collector != address(0) &&
88             _template721 != address(0),
```

```
89         "ERC721DTFactory: Invalid template token/community fee
L  collector address"
90     );
91     add721TokenTemplate(_template721);
92     addTokenTemplate(_template);
93     router = _router;
94     communityFeeCollector = _collector;
95 }
```

FactoryRouter

Listing 42: FactoryRouter.sol (Lines 41,42)

```
34     constructor(
35         address _routerOwner,
36         address _oceanToken,
37         address _bpoolTemplate,
38         address _opfCollector,
39         address[] memory _preCreatedPools
40     ) public BFactory(_bpoolTemplate, _opfCollector,
L  _preCreatedPools) {
41         routerOwner = _routerOwner;
42         opfCollector = _opfCollector;
43         oceanTokens[_oceanToken] = true;
44     }
```

FactoryRouter

Listing 43: FactoryRouter.sol (Line 68)

```
66     function addFactory(address _factory) external onlyRouterOwner
L  {
67         require(factory == address(0), "FACTORY ALREADY SET");
68         factory = _factory;
69     }
```

ERC20Template.sol

Listing 44: ERC20Template.sol (Lines 555,556,558,559)

```

554     function setPublishingMarketFee(
555         address _publishMarketFeeAddress ,
556         address _publishMarketFeeToken ,
557         uint256 _publishMarketFeeAmount) external
↳ onlyPublishingMarketFeeAddress {
558     publishMarketFeeAddress = _publishMarketFeeAddress;
559     publishMarketFeeToken = _publishMarketFeeToken;
560     publishMarketFeeAmount = _publishMarketFeeAmount;
561 }
```

ERC20Template.sol

Listing 45: ERC20Template.sol (Lines 527,532)

```

527     function setFeeCollector(address _newFeeCollector) external {
528         require(
529             permissions[msg.sender].feeManager == true ,
530             "ERC20Template: NOT FEE MANAGER"
531         );
532         feeCollector = _newFeeCollector;
533     }
```

SideStaking.sol

Listing 46: SideStaking.sol (Line 61)

```

60     constructor(address _router) public {
61         router = _router;
62     }
```

Listing 47: BPool.sol (Line 274)

```

272     function updateMarketFeeCollector(address _newCollector)
↳ external {
273         require(_marketCollector == msg.sender , "ONLY MARKET
↳ COLLECTOR");
274         _marketCollector = _newCollector;
275     }
```

Risk Level:

Likelihood - 2

Impact - 3

Recommendation:

Although administrative restrictions are imposed on this function due to the role-based access controls (RBAC) it is recommended to add proper address validation when assigning user supplied input to a variable. This could be as simple as using the following statement:

Listing 48

```
1 require(address_input != 0, "Address is zero")
```

Remediation Plan:

SOLVED: The Ocean Protocol team implemented the recommended fix to ensure addresses are not equal to zero.

3.9 (HAL-09) DIVIDE BEFORE MULTIPLY - LOW

Description:

Solidity's integer division could be truncated. As a result, loss of precision can sometimes be avoided by multiplying before dividing, although the manual implementation of the precision/decimal calculation is taken care of by the developer. In the smart contracts set, there are multiple instances where division is performed before multiplication.

Code Location:

FixedRateExchange

Listing 49: FixedRateExchange.sol (Lines 259,260,261,262,263,268,269,270,273,274,275)

```
248     function calcBaseInGivenOutDT(bytes32 exchangeId, uint256
249         ↳ dataTokenAmount)
250         public
251         view
252         onlyActiveExchange(exchangeId)
253         returns (
254             uint256 baseTokenAmount,
255             uint256 baseTokenAmountBeforeFee,
256             uint256 oceanFeeAmount,
257             uint256 marketFeeAmount
258         )
259     {
260         baseTokenAmountBeforeFee = dataTokenAmount
261             .mul(exchanges[exchangeId].fixedRate)
262             .div(BASE)
263             .mul(10**exchanges[exchangeId].btDecimals)
264             .div(10**exchanges[exchangeId].dtDecimals);
265
266         oceanFeeAmount;
267         if (getOPFFee(exchanges[exchangeId].baseToken) != 0) {
268             oceanFeeAmount = baseTokenAmountBeforeFee
```

```

269             .mul(getOPFFee(exchanges[exchangeId].baseToken))
270             .div(BASE);
271         }
272
273         marketFeeAmount = baseTokenAmountBeforeFee
274             .mul(exchanges[exchangeId].marketFee)
275             .div(BASE);
276
277
278         baseTokenAmount = baseTokenAmountBeforeFee.add(
279             marketFeeAmount).add(
280                 oceanFeeAmount
281             );
282     }

```

FixedRateExchange

Listing 50: FixedRateExchange.sol (Lines 301,302,303,304,305,310,311,312,315,316,317)

```

290     function calcBaseOutGivenInDT(bytes32 exchangeId, uint256
291         dataTokenAmount)
292         public
293         view
294         onlyActiveExchange(exchangeId)
295         returns (
296             uint256 baseTokenAmount,
297             uint256 baseTokenAmountBeforeFee,
298             uint256 oceanFeeAmount,
299             uint256 marketFeeAmount
300         )
301     {
302         baseTokenAmountBeforeFee = dataTokenAmount
303             .mul(exchanges[exchangeId].fixedRate)
304             .div(BASE)
305             .mul(10**exchanges[exchangeId].btDecimals)
306             .div(10**exchanges[exchangeId].dtDecimals);
307
308         oceanFeeAmount;
309         if (getOPFFee(exchanges[exchangeId].baseToken) != 0) {
310             oceanFeeAmount = baseTokenAmountBeforeFee

```

```

311         .mul(getOPFFee(exchanges[exchangeId].baseToken))
312             .div(BASE);
313     }
314
315     marketFeeAmount = baseTokenAmountBeforeFee
316         .mul(exchanges[exchangeId].marketFee)
317         .div(BASE);
318
319
320     baseTokenAmount = baseTokenAmountBeforeFee.sub(
321     ↳ marketFeeAmount).sub(
322         oceanFeeAmount
323     );
324 }
```

SideStaking

Listing 51: SideStaking.sol (Lines 314,315,316,317,318)

```

310     uint256 baseTokenAmount = _datatokens[datatokenAddress]
311         .basetokenBalance;
312     //given the price, compute dataTokenAmount
313
314     uint256 dataTokenAmount = ((_datatokens[datatokenAddress].
315     ↳ rate *
316         baseTokenAmount *
317             dataTokenWeight) /
318             baseTokenWeight /
319                 BASE) * (10**18 - decimals));
```

SideStaking

Listing 52: SideStaking.sol (Lines 378,379,380,381,383)

```

375         _datatokens[datatokenAddress].vestingLastBlock;
376     }
377
378     uint256 vestPerBlock = _datatokens[datatokenAddress].
379     ↳ vestingAmount.div(
380         _datatokens[datatokenAddress].vestingEndBlock -
381             _datatokens[datatokenAddress].vestingLastBlock);
382 }
```

```
380             _datatokens[datatokenAddress].blockDeployed  
381         );  
382         if (vestPerBlock == 0) return;  
383         uint256 amount = blocksPassed.mul(vestPerBlock);  
384         if (  
385             amount > 0 &&
```

Risk Level:

Likelihood - 2

Impact - 3

Recommendation:

Consider performing multiplications before divisions to ensure precision in results when using non-floating-point data types.

Remediation Plan:

SOLVED: The Ocean Protocol team amended smart contracts to perform multiplication operations before division.

3.10 (HAL-10) USE OF BLOCK-TIMESTAMP - LOW

Description:

During a manual review, the use of `block.timestamp` in the `ERC20Template.sol` contract was observed. Contract developers should note that `block.timestamp` does not mean the current time. Miners can influence the value of `block.timestamp` to a certain degree, so developers should be warned that this may be at some risk if miners collude in time manipulation to influence oracle pricing. Miners can influence the timestamp with a tolerance of 15 seconds.

Code Location:

Listing 53: `ERC20Template.sol` (Line 638)

```
629     function permit(
630         address owner,
631         address spender,
632         uint256 value,
633         uint256 deadline,
634         uint8 v,
635         bytes32 r,
636         bytes32 s
637     ) external {
638         require(deadline >= block.timestamp, "ERC20DT: EXPIRED");
639         bytes32 digest = keccak256(
640             abi.encodePacked(
641                 "\x19\x01",
642                 DOMAIN_SEPARATOR,
643                 keccak256(
644                     abi.encode(
645                         PERMIT_TYPEHASH,
```

Risk Level:

Likelihood - 2

Impact - 3

Recommendation:

Use `block.number` instead of `block.timestamp` or `now` to reduce the risk of MEV attacks. Check if the project timescale occurs across years, days, and months rather than seconds. If possible, it is recommended to use Oracles.

Remediation Plan:

ACKNOWLEDGED: The [Ocean Protocol](#) team decided not to modify the highlighted occurrence of `block.timestamp`.

3.11 (HAL-11) EXPERIMENTAL FEATURES ENABLED - LOW

Description:

The use of experimental features could be dangerous on live deployments. The experimental ABI encoder does not handle non-integer values shorter than 32 bytes properly. This applies to bytesNN types, bool, enum and other types when they are part of an array or a struct and encoded directly from storage. This means these storage references have to be used directly inside `abi.encode(. . .)` as arguments in external function calls or in event data without prior assignment to a local variable. Using `return` does not trigger the bug. The types bytesNN and bool will result in corrupted data while enum might lead to an invalid revert.

Furthermore, arrays with elements shorter than 32 bytes may not be handled correctly, even if the base type is an integer type. Encoding such arrays in the way described above can lead to other data in the encoding being overwritten if the number of elements encoded is not a multiple of the number of elements that fit a single slot. If nothing follows the array in the encoding (note that dynamically-sized arrays are always encoded after statically-sized arrays with statically-sized content), or if only a single array is encoded, no other data is overwritten. There are known bugs that are publicly released while using this feature. However, the bug only manifests itself when all the following conditions are met:

1. Storage data involving arrays or structs is sent directly to an external function call, to `abi.encode` or to event data without prior assignment to a local (memory) variable.
2. There is an array that contains elements with size less than 32 bytes or a struct that has elements that share a storage slot or members of type bytesNN shorter than 32 bytes.

In addition to that, in the following situations, your code is NOT affected:

1. If all the structs or arrays only use uint256 or int256 types.
2. If only integer types (that may be shorter) are used and only encode at most one array at a time.
3. If only such data is returned and is not used in abi.encode, external calls or event data.

ABIEncoderV2 is enabled to be able to pass a struct type into a function, both web3 and in another contract. Naturally, any bug can have wildly varying consequences depending on the program control flow, but it is expected that this is more likely to lead to malfunction than exploitability. The bug, when triggered, will under certain circumstances send corrupt parameters on method invocations to other contracts.

Reference:

[Solidity Optimizer and ABIEncoderV2 Bug](#)

Code Location:

Listing 54

```
1 ERC721Factory.sol:2:pragma experimental ABIEncoderV2;
2 FactoryRouter.sol:5:pragma experimental ABIEncoderV2;
3 IFactory.sol:2:pragma experimental ABIEncoderV2;
4 IERC20Template.sol:2:pragma experimental ABIEncoderV2;
```

Risk Level:

Likelihood - 1

Impact - 3

Recommendation:

When possible, do not use experimental features in the final live deployment. Validate and check that all the above conditions are true for integers and arrays (i.e., all using uint256).

FINDINGS & TECH DETAILS

Remediation Plan:

SOLVED: Experimental features have been removed from the smart contracts set.

3.12 (HAL-12) FLOATING PRAGMA - LOW

Description:

Some OceanProtocol v4main contracts use the floating pragma ^0.8.0. Contracts should be deployed with the same compiler version and flags with which they were thoroughly tested. Locking the pragma ensures that contracts are not accidentally get deployed using an outdated compiler version. New pragmas could introduce bugs that negatively affect the contract system or a pragma version too new and has not been extensively tested.

Code Location:

Listing 55: v4main-contract

```
1 (contracts/utils/Ownable.sol:3) pragma solidity ^0.8.0;
2 (contracts/utils/ERC721/IERC721Enumerable.sol:1) pragma solidity
↳ ^0.8.0;
3 (contracts/utils/ERC721/IERC721Receiver.sol:1) pragma solidity
↳ ^0.8.0;
4 (contracts/utils/ERC721/IERC721Metadata.sol:1) pragma solidity
↳ ^0.8.0;
5 (contracts/utils/ERC721/ERC721.sol:1) pragma solidity ^0.8.0;
6 (contracts/utils/ERC721/Context.sol:2) pragma solidity ^0.8.0;
7 (contracts/utils/ERC721/IERC721.sol:5) pragma solidity ^0.8.0;
8 (contracts/utils/ERC721/Address.sol:2) pragma solidity ^0.8.0;
9 (contracts/utils/ERC721/Strings.sol:3) pragma solidity ^0.8.0;
10 (contracts/utils/ERC725/ERC725Ocean.sol:2) pragma solidity ^0.8.0;
11 (contracts/interfaces/IV3ERC20.sol:1) pragma solidity ^0.8.0;
```

Risk Level:

Likelihood - 1

Impact - 3

Recommendation:

Consider locking the pragma version into a secure and stable compiler version. When possible, do not use floating pragma in the final live deployment. Specifying a fixed version of the compiler ensures that the bytecode produced does not vary between builds. This is especially important if the solution relies on code verification at the bytecode level.

Remediation Plan:

SOLVED: Floating pragma were removed from the smart contracts set.

3.13 (HAL-13) OUTDATED DEPENDENCIES - LOW

Description:

The `4.2.0` version of `openzeppelin-contracts` is used in the `v4main` smart contracts. The latest version is `4.3.2` where a vulnerability in `UUPSUpgradeable` is fixed. Although this is not used in `v4main` contracts, it is a good security practice to keep all libraries up-to-date.

Code Location:

Listing 56: v4main/package.json (Line 29)

```
27 "dependencies": {  
28   "@balancer-labs/v2-pool-utils": "^1.0.0",  
29   "@openzeppelin/contracts": "^4.2.0",  
30   "@openzeppelin/test-helpers": "^0.5.10",  
31   "dotenv": "^10.0.0",  
32   "eth-permit": "^0.1.10",  
33   "ethereumjs-util": "^7.0.10",  
34   "hardhat-contract-sizer": "^2.0.3",  
35   "solidity-bytes-utils": "^0.8.0"  
36 }
```

Risk Level:

Likelihood - 2

Impact - 3

Recommendation:

Although `UUPSUpgradeable` is not used directly within `v4main` contracts, it is always important to keep all libraries up-to-date. See references' section for a more detailed description of OpenZeppelin contract issues.

FINDINGS & TECH DETAILS

References:

[Open Zeppelin Advisory](#)
[UUPS Implementation Workaround](#)

Remediation Plan:

SOLVED: The [Ocean Protocol team](#) updated the external dependency version to v4.3.3.

3.14 (HAL-14) PRAGMA VERSION DEPRECATED - LOW

Description:

In some in-scope contracts, the current pragma versions in use are `pragma >=0.5.0`, `>=0.5.7`, `>=0.6.0` etc. While these versions are still functional, and some security issues are mitigated by implementing contracts such as `SafeMath.sol`, it increases the risk to long-term sustainability and Solidity code integrity.

Code Location:

Listing 57: v4main-contract

```
1 (contracts/ERC721Factory.sol:1) pragma solidity >=0.6.0;
2 (contracts/templates/ERC20Template.sol:1) pragma solidity >=0.6.0;
3 (contracts/templates/ERC20TemplateEnterprise.sol:1) pragma
↳ solidity >=0.6.0;
4 (contracts/templates/ERC721Template.sol:1) pragma solidity
↳ >=0.6.0;
5 (contracts/utils/ERC721RolesAddress.sol:1) pragma solidity
↳ >=0.6.0;
6 (contracts/utils/UtilsLib.sol:9) pragma solidity >=0.5.0 <0.9.0;
7 (contracts/utils/Deployer.sol:1) pragma solidity >=0.5.7;
8 (contracts/utils/ERC20Roles.sol:1) pragma solidity >=0.6.0;
9 (contracts/pools/sscontracts/SideStaking.sol:1) pragma solidity
↳ >=0.6.0;
10 (contracts/pools/dispenser/Dispenser.sol:1) pragma solidity
↳ >=0.5.7;
11 (contracts/pools/FactoryRouter.sol:4) pragma solidity >=0.5.7;
12 (contracts/pools/balancer/BConst.sol:14) pragma solidity >=0.5.7;
13 (contracts/pools/balancer/BFactory.sol:1) pragma solidity >=0.5.7;
14 (contracts/pools/balancer/BMath.sol:14) pragma solidity >=0.5.7;
15 (contracts/pools/balancer/BToken.sol:14) pragma solidity >=0.5.7;
16 (contracts/pools/balancer/BPool.sol:1://pragma solidity 0.5.7;
17 (contracts/pools/balancer/BPool.sol:2) pragma solidity >=0.6.0;
18 (contracts/pools/balancer/BNum.sol:14) pragma solidity >=0.5.7;
19 (contracts/pools/fixedRate/FixedRateExchange.sol:1) pragma
↳ solidity >=0.5.7;
```

```
20 (contracts/interfaces/ISideStaking.sol:9) pragma solidity >=0.5.7;
21 (contracts/interfaces/IFactoryRouter.sol:1) pragma solidity
↳ >=0.5.7;
22 (contracts/interfaces/IFactory.sol:1) pragma solidity >=0.5.7;
23 (contracts/interfaces/IMetadata.sol:3) pragma solidity >=0.6.2
↳ <0.9.0;
24 (contracts/interfaces/IDispenser.sol:1) pragma solidity >=0.5.7;
25 (contracts/interfaces/IERC20Template.sol:1) pragma solidity
↳ >=0.5.0;
26 (contracts/interfaces/IERC721Template.sol:3) pragma solidity
↳ >=0.6.2 <0.9.0;
27 (contracts/interfaces/IFixedRateExchange.sol:1) pragma solidity
↳ >=0.5.7;
28 (contracts/interfaces/IERC725X.sol:2) pragma solidity >=0.5.0
↳ <0.9.0;
29 (contracts/interfaces/IERC725Y.sol:2) pragma solidity >=0.5.0
↳ <0.9.0;
30 (contracts/interfaces/IERC20.sol:3) pragma solidity >=0.5.7;
31 (contracts/interfaces/IV3Factory.sol:1) pragma solidity >=0.6.0;
32 (contracts/interfaces/IERC1271.sol:2) pragma solidity >=0.5.0
↳ <0.79.0;
33 (contracts/interfaces/IPool.sol:10) pragma solidity >=0.5.7;
34 (contracts/communityFee/OPFCommunityFeeCollector.sol:1) pragma
↳ solidity >=0.6.0;
```

Risk Level:

Likelihood - 2

Impact - 2

Recommendation:

At the time of this audit, the current solidity version is already at 0.8.X. When possible, it is recommended to use an updated pragma version to take advantage of new features, for example, after **Solidity version 0.8.0**, arithmetic operations revert to underflow and overflow by default. Using this version, utility contracts like **SafeMath.sol** will not be needed.

FINDINGS & TECH DETAILS

Remediation Plan:

SOLVED: All smart contracts have been upgraded to use the compiler version 0.8.10.

3.15 (HAL-15) MULTIPLE PRAGMA DEFINITIONS - LOW

Description:

OceanProtocol v4main contract uses a complex pragma version (such as `>=0.5.0 <0.9.0`, `>=0.5.7`, `>=0.6.0`, `>=0.6.2 <0.9.0`). Contracts should be deployed with the same compiler version and flags with which they were thoroughly tested. Locking the pragma helps to ensure that contracts are not accidentally get deployed using another pragma, for example, either an outdated pragma version that could introduce bugs that negatively affect the contract system or a recently released pragma version that has not been extensively tested. The latest pragma version (0.8.9) was released in September 2021. Many pragma versions have been lately released, going from version 0.7.x to the recently released version 0.8.x. in just 8 months.

Reference: <https://github.com/ethereum/solidity/releases>

In the Solidity Github repository, there is a json file where are all the bugs found in the different compiler versions. It should be noted that pragma 0.6.12 and 0.7.6 are widely used by Solidity developers and have been extensively tested in many security audits.

Reference: https://github.com/ethereum/solidity/blob/develop/docs/bugs_by_version.json

Code Location:

Listing 58: v4main-contract

```
1 (contracts/ERC721Factory.sol:1) pragma solidity >=0.6.0;
2 (contracts/templates/ERC20Template.sol:1) pragma solidity >=0.6.0;
3 (contracts/templates/ERC20TemplateEnterprise.sol:1) pragma
↳ solidity >=0.6.0;
4 (contracts/templates/ERC721Template.sol:1) pragma solidity
↳ >=0.6.0;
```

```
5 (contracts/utils/ERC721RolesAddress.sol:1) pragma solidity
↳ >=0.6.0;
6 (contracts/utils/UtilsLib.sol:9) pragma solidity >=0.5.0 <0.9.0;
7 (contracts/utils/Ownable.sol:3) pragma solidity ^0.8.0;
8 (contracts/utils/Deployer.sol:1) pragma solidity >=0.5.7;
9 (contracts/utils/ERC20Roles.sol:1) pragma solidity >=0.6.0;
10 (contracts/utils/ERC721/IERC721Enumerable.sol:1) pragma solidity
↳ ^0.8.0;
11 (contracts/utils/ERC721/IERC721Receiver.sol:1) pragma solidity
↳ ^0.8.0;
12 (contracts/utils/ERC721/IERC721Metadata.sol:1) pragma solidity
↳ ^0.8.0;
13 (contracts/utils/ERC721/ERC721.sol:1) pragma solidity ^0.8.0;
14 (contracts/utils/ERC721/Context.sol:2) pragma solidity ^0.8.0;
15 (contracts/utils/ERC721/IERC721.sol:5) pragma solidity ^0.8.0;
16 (contracts/utils/ERC721/Address.sol:2) pragma solidity ^0.8.0;
17 (contracts/utils/ERC721/Strings.sol:3) pragma solidity ^0.8.0;
18 (contracts/utils/ERC725/ERC725Ocean.sol:2) pragma solidity ^0.8.0;
19 (contracts/pools/sscontracts/SideStaking.sol:1) pragma solidity
↳ >=0.6.0;
20 (contracts/pools/dispenser/Dispenser.sol:1) pragma solidity
↳ >=0.5.7;
21 (contracts/pools/FactoryRouter.sol:4) pragma solidity >=0.5.7;
22 (contracts/pools/balancer/BConst.sol:14) pragma solidity >=0.5.7;
23 (contracts/pools/balancer/BFactory.sol:1) pragma solidity >=0.5.7;
24 (contracts/pools/balancer/BMath.sol:14) pragma solidity >=0.5.7;
25 (contracts/pools/balancer/BToken.sol:14) pragma solidity >=0.5.7;
26 (contracts/pools/balancer/BPool.sol:1://pragma solidity 0.5.7;
27 (contracts/pools/balancer/BPool.sol:2) pragma solidity >=0.6.0;
28 (contracts/pools/balancer/BNum.sol:14) pragma solidity >=0.5.7;
29 (contracts/pools/fixedRate/FixedRateExchange.sol:1) pragma
↳ solidity >=0.5.7;
30 (contracts/interfaces/ISideStaking.sol:9) pragma solidity >=0.5.7;
31 (contracts/interfaces/IFactoryRouter.sol:1) pragma solidity
↳ >=0.5.7;
32 (contracts/interfaces/IFactory.sol:1) pragma solidity >=0.5.7;
33 (contracts/interfaces/IMetadata.sol:3) pragma solidity >=0.6.2
↳ <0.9.0;
34 (contracts/interfaces/IDispenser.sol:1) pragma solidity >=0.5.7;
35 (contracts/interfaces/IERC20Template.sol:1) pragma solidity
↳ >=0.5.0;
36 (contracts/interfaces/IERC721Template.sol:3) pragma solidity
↳ >=0.6.2 <0.9.0;
37 (contracts/interfaces/IFixedRateExchange.sol:1) pragma solidity
```

```
↳ >=0.5.7;
38 (contracts/interfaces/IERC725X.sol:2) pragma solidity >=0.5.0
↳ <0.9.0;
39 (contracts/interfaces/IERC725Y.sol:2) pragma solidity >=0.5.0
↳ <0.9.0;
40 (contracts/interfaces/IERC20.sol:3) pragma solidity >=0.5.7;
41 (contracts/interfaces/IV3Factory.sol:1) pragma solidity >=0.6.0;
42 (contracts/interfaces/IERC1271.sol:2) pragma solidity >=0.5.0
↳ <0.79.0;
43 (contracts/interfaces/IV3ERC20.sol:1) pragma solidity ^0.8.0;
44 (contracts/interfaces/IPool.sol:10) pragma solidity >=0.5.7;
45 (contracts/communityFee/OPFCommunityFeeCollector.sol:1) pragma
↳ solidity >=0.6.0;
```

Risk Level:

Likelihood - 2

Impact - 2

Recommendation:

Consider locking and using a single pragma version with no known bugs for the compiler version. If possible, consider using the latest stable version of pragma that has been thoroughly tested for potential undiscovered vulnerabilities, such as a pragma between `0.6.12 - 0.7.6`, or the latest pragma. For example, after the `Solidity v 0.8.0`, arithmetic operations revert to underflow and overflow by default. Using this version, utility contracts like `SafeMath.sol` will not be needed.

Remediation Plan:

SOLVED: All smart contracts have been upgraded to use compiler version `0.8.10`.

3.16 (HAL-16) MISSING EVENTS EMISSION - INFORMATIONAL

Description:

It has been observed that important functionality is not to emit events. This was observed in the `FactoryRouter.sol`, `BPool.sol`, and `ERC20Template.sol` contracts. Events are a method of informing the transaction initiator about the actions performed by the called function. Event logs emit parameters in a specific log history, which can be accessed outside the contracts using some filter parameters.

Code Location:

Factory Router

Listing 59: FactoryRouter.sol (Line 51)

```
46     function changeRouterOwner(address _routerOwner) public
47       onlyRouterOwner {
48         require(
49           _routerOwner != address(0),
50           'Invalid new router owner'
51         );
52         routerOwner = _routerOwner;
53     }
```

Factory Router

Listing 60: FactoryRouter.sol (Line 86)

```
84     function updateOPFee(uint256 _newSwapOceanFee) external
85       onlyRouterOwner {
86         // TODO: add a maximum? how much? add event?
87         swapOceanFee = _newSwapOceanFee;
88     }
```

ERC20Template.sol

Listing 61: ERC20Template.sol (Lines 558,560)

```
554     function setPublishingMarketFee(
555         address _publishMarketFeeAddress ,
556         address _publishMarketFeeToken ,
557         uint256 _publishMarketFeeAmount) external
558     ↳ onlyPublishingMarketFeeAddress {
559         publishMarketFeeAddress = _publishMarketFeeAddress;
560         publishMarketFeeToken = _publishMarketFeeToken;
561         publishMarketFeeAmount = _publishMarketFeeAmount;
562     }
```

BPool.sol**Listing 62: BPool.sol (Line 339)**

```
334     function setSwapFee(uint256 swapFee) public {
335         require(!_finalized, "ERR_IS_FINALIZED");
336         require(msg.sender == _controller, "ERR_NOT_CONTROLLER");
337         require(swapFee >= MIN_FEE, "ERR_MIN_FEE");
338         require(swapFee <= MAX_FEE, "ERR_MAX_FEE");
339         ↳ _swapFee = swapFee;
340     }
```

Risk Level:

Likelihood - 1

Impact - 2

Recommendation:

For best security practices consider, as much as possible, to declare events at the end of a function. Events can be used to detect the end of an operation.

Remediation Plan:

SOLVED: The [Ocean Protocol](#) team added events for critical functionality within the highlighted functions.

3.17 (HAL-17) REDUNDANT BOOLEAN COMPARISON - INFORMATIONAL

Description:

In the solidity language, Boolean constants can be used directly and do not need to be compared to true or false. In the `ERC721Factory.sol`, `ERC20Template.sol`, `ERC721Template.sol`, `FixedRateExchange.sol`, `Dispenser.sol`, `BFactory.sol`, `BPool.sol` and `FactoryRouter.sol` contracts, boolean constants are compared to `true` or `false`.

Code Location:

`ERC721Factory.sol`

Listing 63: `ERC721Factory.sol` (Line 121)

```
120     require(
121         tokenTemplate.isActive == true,
122         "ERC721DTFactory: ERC721Token Template disabled"
123     );
```

`ERC721Factory.sol`

Listing 64: `ERC721Factory.sol` (Line 317)

```
316     require(
317         tokenTemplate.isActive == true,
318         "ERC20Factory: ERC721Token Template disabled"
319     );
```

`FactoryRouter.sol`

Listing 65: `FactoryRouter.sol` (Line 79)

```
78     function getOPFee(address baseToken) public view returns (
79         uint256) {
80         if (oceanTokens[baseToken] == true) {
```

```
80         return 0;
81     } else return swapOceanFee;
82 }
```

FactoryRouter.sol

Listing 66: FactoryRouter.sol (Lines 127,131,138)

```
126     require(
127         IFactory(factory).erc20List(msg.sender) == true,
128         "FACTORY ROUTER: NOT ORIGINAL ERC20 TEMPLATE"
129     );
130     require(
131         ssContracts[addresses[0]] == true,
132         "FACTORY ROUTER: invalid ssContract"
133     );
134     require(ssParams[1] > 0, "Wrong decimals");
135
136     // TODO: do we need this? used only for the event?
137     bool flag;
138     if (oceanTokens[tokens[1]] == true) {
139         flag = true;
140     }
```

FactoryRouter.sol

Listing 67: FactoryRouter.sol (Lines 183,188)

```
176     function deployFixedRate(
177         address fixedPriceAddress ,
178         address[] calldata addresses ,
179         uint[] calldata uints
180
181     ) external returns (bytes32 exchangeId) {
182     require(
183         IFactory(factory).erc20List(msg.sender) == true,
184         "FACTORY ROUTER: NOT ORIGINAL ERC20 TEMPLATE"
185     );
186
187     require(
188         fixedPrice[fixedPriceAddress] == true ,
189         "FACTORY ROUTER: Invalid FixedPriceContract"
```

```
190 );
```

FactoryRouter.sol

Listing 68: FactoryRouter.sol (Lines 221,226)

```
212     function deployDispenser(
213         address _dispenser,
214         address datatoken,
215         uint256 maxTokens,
216         uint256 maxBalance,
217         address owner,
218         address allowedSwapper
219     ) external {
220         require(
221             IFactory(factory).erc20List(msg.sender) == true,
222             "FACTORY ROUTER: NOT ORIGINAL ERC20 TEMPLATE"
223         );
224
225         require(
226             dispenser[_dispenser] == true,
227             "FACTORY ROUTER: Invalid DispenserContract"
228         );

```

Dispenser.sol

Listing 69: Dispenser.sol (Lines 192,193,194,195)

```
190         'Invalid token contract address'
191     );
192     require(
193         datatokens[datatoken].active == true,
194         'Dispenser not active'
195     );
196     require(

```

FixedRateExchange.sol

Listing 70: FixedRateExchange.sol (Lines 51,52,53,54,55)

```

50     modifier onlyActiveExchange(bytes32 exchangeId) {
51         require(
52             //exchanges[exchangeId].fixedRate != 0 &&
53             exchanges[exchangeId].active == true,
54             "FixedRateExchange: Exchange does not exist!"
55         );
56         _;
57     }

```

FixedRateExchange.sol

Listing 71: FixedRateExchange.sol (Lines 675,676,677)

```

673         returns (uint256 supply)
674     {
675         if (exchanges[exchangeId].active == false) supply = 0;
676         else if (exchanges[exchangeId].withMint == true
677             && IERC20Template(exchanges[exchangeId].dataToken).
678             ↳ isMinter(address(this))){
679                 supply = IERC20Template(exchanges[exchangeId].
680             ↳ dataToken).cap()
681                 - IERC20Template(exchanges[exchangeId].dataToken).
682             ↳ totalSupply();

```

FixedRateExchange.sol

Listing 72: FixedRateExchange.sol (Line 704)

```

700     public
701     view
702     returns (uint256 supply)
703     {
704         if (exchanges[exchangeId].active == false) supply = 0;
705         else {
706             uint256 balance = IERC20Template(exchanges[exchangeId
707             ↳ ].baseToken)

```

ERC721Template.sol

Listing 73: ERC721Template.sol (Lines 159,160,161,162,166)

```

156     function setMetaData(uint8 _metaDataState, string calldata
157         _metaDataDecryptorUrl
158             , string calldata _metaDataDecryptorAddress, bytes
159                 calldata flags,
160                     bytes calldata data,bytes calldata _metaDataHash) external
161             {
162                 require(
163                     permissions[msg.sender].updateMetadata == true,
164                         "ERC721Template: NOT METADATA_ROLE"
165                     );
166                 metaDataState = _metaDataState;
167                 metaDataDecryptorUrl = _metaDataDecryptorUrl;
168                 metaDataDecryptorAddress = _metaDataDecryptorAddress;
169                 if(hasMetaData == false){
170                     emit MetadataCreated(msg.sender, _metaDataState,
171                         _metaDataDecryptorUrl,
172                             flags, data, _metaDataHash,
173                                 /* solium-disable-next-line */
174                                     block.timestamp,
175                                         block.number);
176                     hasMetaData = true;
177                 }
178                 else
179                     emit MetadataUpdated(msg.sender, metaDataState,
180                         _metaDataDecryptorUrl,
181                             flags, data, _metaDataHash,
182                                 /* solium-disable-next-line */
183                                     block.timestamp,
184                                         block.number);
185             }

```

ERC721Template.sol

Listing 74: ERC721Template.sol (Lines 222,223,224,225)

```

220         bytes[] calldata bytess
221     ) external returns (address ) {
222         require(
223             permissions[msg.sender].deployERC20 == true,
224                 "ERC721Template: NOT ERC20DEPLOYER_ROLE"
225             );
226

```

ERC721Template.sol

Listing 75: ERC721Template.sol (Lines 336,337,338,339)

```
334     function setNewData(bytes32 _key, bytes calldata _value)
335     external {
336         require(
337             permissions[msg.sender].store == true,
338             "ERC721Template: NOT STORE UPDATER"
339         );
340         setData(_key, _value);
```

ERC721Template.sol

Listing 76: ERC721Template.sol (Lines 353,354,355,356)

```
352     function setDataERC20(bytes32 _key, bytes calldata _value)
353     public {
354         require(
355             deployedERC20[msg.sender] == true,
356             "ERC721Template: NOT ERC20 Contract"
357         );
358         setData(_key, _value);
359     }
```

ERC20Template.sol

Listing 77: ERC20Template.sol (Line 313)

```
310         IFactoryRouter(router).deployDispenser(
311             _dispenser, address(this), maxTokens, maxBalance, msg.
312             sender, allowedSwapper );
313         // add FixedPriced contract as minter if withMint == true
314         if (withMint == true)
315             _addMinter(_dispenser);
316     }
```

ERC20Template.sol

Listing 78: ERC20Template.sol (Lines 326,327,328,329)

```
325     function mint(address account, uint256 value) external {
326         require(
327             permissions[msg.sender].minter == true,
328             "ERC20Template: NOT MINTER"
329         );
330         require(
331             totalSupply().add(value) <= _cap,
332             "DataTokenTemplate: cap exceeded"
333         );
334         _mint(account, value);
335     }
```

ERC20Template.sol

Listing 79: ERC20Template.sol (Lines 102,103,104,105,106,107)

```
101     modifier onlyERC20Deployer() {
102         require(
103             IERC721Template(_erc721Address)
104                 .getPermissions(msg.sender)
105                 .deployERC20 == true,
106             "ERC20Template: NOT DEPLOYER ROLE"
107         );
108         _;
109     }
```

ERC20Template.sol

Listing 80: ERC20Template.sol (Lines 528,529,530,531)

```
527     function setFeeCollector(address _newFeeCollector) external {
528         require(
529             permissions[msg.sender].feeManager == true,
530             "ERC20Template: NOT FEE MANAGER"
531         );
532         feeCollector = _newFeeCollector;
533     }
```

SideStaking.sol

Listing 81: SideStaking.sol (Lines 99,100,101,102)

```

97      // check if we are the minter of DT
98      IERC20Template dt = IERC20Template(datatokenAddress);
99      require(
100          (dt.permissions(address(this))).minter == true,
101          "BaseToken address mismatch"
102      );
103      // get cap and mint it..
104      dt.mint(address(this), dt.cap());
105

```

SideStaking.sol

Listing 82: SideStaking.sol (Lines 138,148,157,166,175,184,193,202
,211,220,234,250,256,269,291,297,305,306)

```

137  {
138      if (_datatokens[datatokenAddress].bound != true) return
139      (0);
140      return (_datatokens[datatokenAddress].datatokenCap -
141          _datatokens[datatokenAddress].datatokenBalance);
142  }
143
144  function getPublisherAddress(address datatokenAddress)
145      public
146      view
147      returns (address)
148  {
149      if (_datatokens[datatokenAddress].bound != true) return (
150      address(0));
151      return (_datatokens[datatokenAddress].publisherAddress);
152  }
153
154  function getBaseTokenAddress(address datatokenAddress)
155      public
156      view
157      returns (address)
158  {
159      if (_datatokens[datatokenAddress].bound != true) return (
160      address(0));

```

```
158         return (_datatokens[datatokenAddress].basetokenAddress);
159     }
160
161     function getPoolAddress(address datatokenAddress)
162         public
163         view
164         returns (address)
165     {
166         if (_datatokens[datatokenAddress].bound != true) return (
167             address(0));
168         return (_datatokens[datatokenAddress].poolAddress);
169     }
170
171     function getBaseTokenBalance(address datatokenAddress)
172         public
173         view
174         returns (uint256)
175     {
176         if (_datatokens[datatokenAddress].bound != true) return
177             (0);
178         return (_datatokens[datatokenAddress].basetokenBalance);
179     }
180
181     function getDataTokenBalance(address datatokenAddress)
182         public
183         view
184         returns (uint256)
185     {
186         if (_datatokens[datatokenAddress].bound != true) return
187             (0);
188         return (_datatokens[datatokenAddress].datatokenBalance);
189     }
190
191     function getvestingEndBlock(address datatokenAddress)
192         public
193         view
194         returns (uint256)
195     {
196         if (_datatokens[datatokenAddress].bound != true) return
197             (0);
198         return (_datatokens[datatokenAddress].vestingEndBlock);
199     }
200
201     function getvestingAmount(address datatokenAddress)
```

```
198     public
199     view
200     returns (uint256)
201     {
202         if (_datatokens[datatokenAddress].bound != true) return
203         (0);
204         return (_datatokens[datatokenAddress].vestingAmount);
205     }
206
207     function getvestingLastBlock(address datatokenAddress)
208     public
209     view
210     returns (uint256)
211     {
212         if (_datatokens[datatokenAddress].bound != true) return
213         (0);
214         return (_datatokens[datatokenAddress].vestingLastBlock);
215     }
216
217     function getvestingAmountSoFar(address datatokenAddress)
218     public
219     view
220     returns (uint256)
221     {
222         if (_datatokens[datatokenAddress].bound != true) return
223         (0);
224         return (_datatokens[datatokenAddress].vestingAmountSoFar);
225     }
226
227     //called by pool to confirm that we can stake a token (add
228     //pool liquidity). If true, pool will call Stake function
229     function canStake(
230         address datatokenAddress,
231         address stakeToken,
232         uint256 amount
233     ) public view returns (bool) {
234         require(
235             msg.sender == _datatokens[datatokenAddress].
236             poolAddress,
237             "ERR: Only pool can call this"
238         );
239         if (_datatokens[datatokenAddress].bound != true) return (
240             false);
241         if (_datatokens[datatokenAddress].basetokenAddress ==
```

```
↳ stakeToken)
236         return (false);
237
238     //check balances
239     if (_datatokens[datatokenAddress].datatokenBalance >=
240         amount)
241         return (true);
242     return (false);
243 }
244
245     //called by pool so 1ss will stake a token (add pool liquidity)
246     // Function only needs to approve the amount to be spent by the
247     // pool, pool will do the rest
248     function Stake(
249         address datatokenAddress,
250         address stakeToken,
251         uint256 amount
252     ) public {
253         if (_datatokens[datatokenAddress].bound != true) return;
254         require(
255             msg.sender == _datatokens[datatokenAddress].
256             poolAddress,
257             "ERR: Only pool can call this"
258         );
259         bool ok = canStake(datatokenAddress, stakeToken, amount);
260         if (ok != true) return;
261         IERC20Template dt = IERC20Template(datatokenAddress);
262         dt.approve(_datatokens[datatokenAddress].poolAddress,
263             amount);
264         _datatokens[datatokenAddress].datatokenBalance -= amount;
265     }
266
267     //called by pool to confirm that we can stake a token (add
268     // pool liquidity). If true, pool will call Unstake function
269     function canUnStake(
270         address datatokenAddress,
271         address stakeToken,
```

```

↳ poolAddress,
272         "ERR: Only pool can call this"
273     );
274     //check balances, etc and issue true or false
275     if (_datatokens[datatokenAddress].basetokenAddress ==
276 stakeToken)
277         return (false);
278
279         // we check LPT balance TODO: review this part
280         if (IERC20Template(msg.sender).balanceOf(address(this)) >=
281 lptIn) {
282             return true;
283         }
284
285         //called by pool so 1ss will unstake a token (remove pool
286 liquidity). In our case the balancer pool will handle all, this is
287 just a notifier so 1ss can handle internal kitchen
288     function UnStake(
289         address datatokenAddress,
290         address stakeToken,
291         uint256 amount
292     ) public {
293         if (_datatokens[datatokenAddress].bound != true) return;
294         require(
295             msg.sender == _datatokens[datatokenAddress].
296 poolAddress,
297             "ERR: Only pool can call this"
298         );
299         bool ok = canUnStake(datatokenAddress, stakeToken, amount)
300 ;
301         if (ok != true) return;
302         _datatokens[datatokenAddress].datatokenBalance += amount;
303     }
304
305     //called by the pool (or by us) when we should finalize the
306 pool
307     function notifyFinalize(address datatokenAddress, uint256
308 decimals)
309         internal
310     {
311         if (_datatokens[datatokenAddress].bound != true) return;
312         if (_datatokens[datatokenAddress].poolFinalized == true)

```

```
↳ return;
307     _datatokens[datatokenAddress].poolFinalized = true;
```

SideStaking.sol

Listing 83: SideStaking.sol (Lines 358,359,360,361)

```
357     function getVesting(address datatokenAddress) public {
358         require(
359             _datatokens[datatokenAddress].bound == true,
360             "ERR: Invalid datatoken"
361         );
362         // is this needed?
```

BPool.sol

Listing 84: BPool.sol (Lines 735,736)

```
734     if (
735         ssContract.canStake(_datatokenAddress, ssStakeToken,
736         ssAmountIn) ==
737         true
738     ) {
```

BPool.sol

Listing 85: BPool.sol (Lines 816,817)

```
815     if (
816         ssContract.canStake(_datatokenAddress, ssStakeToken,
817         ssAmountIn) ==
818         true
819     ) {
```

BPool.sol

Listing 86: BPool.sol (Lines 899,900,901,902,903)

```
897     if (
```

```
899         ssContract.canUnStake(
900             _datatokenAddress,
901             ssStakeToken,
902             poolAmountIn
903         ) == true
904     } {
```

BPool.sol

Listing 87: BPool.sol (Lines 980,981,982,983,984)

```
978     );
979     if (
980         ssContract.canUnStake(
981             _datatokenAddress,
982             ssStakeToken,
983             ssAmountOut
984         ) == true
985     } {
```

BPool.sol

Listing 88: BFactory.sol (Line 88)

```
79     function newBPool(
80         address[2] memory tokens,
81         uint256[] memory ssParams,
82         uint256[] memory swapFees,
83         address[] memory addresses
84     )
85     internal
86     returns (address bpool)
87     {
88         require(poolTemplates[addresses[5]] == true, 'BFactory:
L Wrong Pool Template');
89         address[2] memory feeCollectors = [addresses[4],
L opfCollector];
90     }
```

Risk Level:

Likelihood - 1

Impact - 1

Recommendation:

It is recommended to remove the redundant boolean comparison within `if` and `require` statements.

Remediation Plan:

SOLVED: The Ocean Protocol team removed the redundant boolean comparisons with commit id `c428e0cff8356778f9643954b86a31e511fd6e06`.

3.18 (HAL-18) USE OF INLINE ASSEMBLY - INFORMATIONAL

Description:

Inline assembly is a way to access the Ethereum Virtual Machine at a low level. This discards several important safety features in Solidity and could incur in some risks should they used incorrectly.

Code Location:

ERC721Factory.sol

Listing 89: ERC721Factory.sol (Line 254)

```
254 assembly {
255     size := extcodesize(account)
256 }
```

ERC20Factory.sol

Listing 90: ERC20Template.sol (Line 207)

```
207 assembly {
208     chainId := chainid()
209 }
```

ERC20TemplateEnterprise.sol

Listing 91: ERC20TemplateEnterprise.sol (Line 213)

```
213 assembly {
214     chainId := chainid()
215 }
```

UtilsLib.sol

Listing 92: UtilsLib.sol (Line 22)

```
22 assembly {
23     result := gt(extcodesize(_target), 0)
24 }
25
```

Deployer.sol

Listing 93: Deployer.sol (Line 39)

ERC7250cean.sol

Listing 94: ERC725Ocean.sol (Line 104)

```
104 assembly {
105     success := call(txGas, to, value, add(data, 0x20), mload(data)
106     , 0, 0)
107 }
```

ERC7250cean.sol

Listing 95: ERC725Ocean.sol (Line 116)

```
116 assembly {
117     success := delegatecall(txGas, to, add(data, 0x20), mload(data
118     ), 0, 0)
119 }
```

ERC7250cean.sol

Listing 96: ERC725Ocean.sol (Line 128)

```
128 assembly {
129     newContract := create(value, add(deploymentData, 0x20), mload(
130         deploymentData))
130 }
```

Risk Level:

Likelihood - 1

Impact - 2

Recommendation:

When possible, it is not recommended to use inline assembly because it allows access to the Ethereum Virtual Machine (EVM) at a low level. An attacker could bypass many important safety features of Solidity.

Remediation Plan:

ACKNOWLEDGED: The [Ocean Protocol](#) team has not removed inline assembly statements, as they are required for the functioning of the contracts.

3.19 (HAL-19) REDUNDANT VARIABLES - INFORMATIONAL

Description:

During the manual code review, it has been observed that `oceanFeeAmount` variable is unnecessarily declared in the contract. By deleting this variable, it is possible to optimize the use of gas.

Code Location:

FixedRateExchange

Listing 97: FixedRateExchange.sol (Line 266)

```
266     oceanFeeAmount;  
267     if (getOPFFee(exchanges[exchangeId].baseToken) != 0) {
```

FixedRateExchange

Listing 98: FixedRateExchange.sol (Line 308)

```
308     oceanFeeAmount;  
309     if (getOPFFee(exchanges[exchangeId].baseToken) != 0) {
```

Risk Level:

Likelihood - 1

Impact - 1

Recommendation:

It is recommended to remove unused state variables within the code base.

Remediation Plan:

SOLVED: The [Ocean Protocol](#) team implemented the suggested fix and removed the redundant variable declaration.

3.20 (HAL-20) POSSIBLE MISUSE OF PUBLIC FUNCTIONS - INFORMATIONAL

Description:

In public functions, the array arguments are immediately copied into memory, while external functions can read directly from `calldata` which is cheaper in terms of gas. Public functions need to write the arguments to memory because public functions can be called internally using pointers to memory locations. Therefore, the function expects its arguments being located in memory when the compiler generates the code for an internal function.

Additionally, methods do not necessarily have to be public if they are only called within the contract, in which case they should be marked `internal`.

Code Location:

Below are smart contracts and their corresponding functions affected:

ERC721Factory:

`createToken(uint256, string[], address[], uint256[], bytes[])`

FactoryRouter:

`changeRouterOwner(address)` `changeRouterOwner(address)` `addOceanTo-`
`ken(address)` `removeOceanToken(address)` `getOPFFee(address)`

FixedRateExchange.sol:

`createWithDecimals(address, address[], uint256[])`

SideStaking.sol:

`getDataTokenCirculatingSupply(address)` `getPublisherAddress(address)`
`getBaseTokenAddress(address)` `getPoolAddress(address)` `getBaseTo-`
`kenBalance(address)` `getDataTokenBalance(address)` `getvestingEnd-`
`Block(address)` `getvestingAmount(address)` `getvestingLastBlock(address)`
`getvestingAmountSoFar(address)` `Stake(address, address, uint256)` `Un-`

```
Stake(address,address,uint256) getVesting(address)
```

ERC721Template.sol:

```
isERC20Deployer(address) isInitialized() setDataERC20(bytes32,bytes)
```

ERC20Template.sol:

```
isMinter(address)
```

BToken.sol:

```
name() symbol() decimals() totalSupply()
```

BPool.sol:

```
setSwapFee(uint256)
```

Risk Level:

Likelihood - 1

Impact - 1

Recommendation:

It is recommended to declare external variables instead of public functions and variables as often as possible. As for best practice, use external if the function is expected to be called only externally, and use public if the function needs to be called internally. To sum up, public functions can be accessed by everyone, external functions can only be accessed externally, and internal functions can only be called within the contract itself.

Remediation Plan:

PARTIALLY SOLVED: The [Ocean Protocol](#) team modified the function visibility to be external instead of public for all the above mentioned functions apart from the following:

- `getVestingAmountSoFar: (SideStaking contract)`
- `canStake: (SideStaking contract)`

FINDINGS & TECH DETAILS

- `canUnstake`: (`SideStaking` contract)
- `isERC20Deployer`: (`ERC20Template` contract)
- `totalSupply`: (`BToken` contract)
- `setSwapFee`: (`BPool` contract)

3.21 (HAL-21) POTENTIAL UNSAFE CALCULATION - INFORMATIONAL

Description:

Within the `SideStaking` contract, calculations were performed without the use of a secure library such as `SafeMath`. While operations with Solidity 0.8.0 are safe, the current pragma might be set to a lower one. It should be noted that when using Solidity 0.8.0, the `unchecked` statement can be used to remove arithmetic safety checks in the event that an overflow/underflow does not occur. This would help save gas.

Code Location:

`SideStaking` Lines# 314-318

Listing 99

```
1     uint256 dataTokenAmount = ((_datatokens[datatokenAddress].  
↳ rate *  
2         baseTokenAmount *  
3             dataTokenWeight) /  
4             baseTokenWeight /  
5             BASE) * (10**18 - decimals);
```

Recommendation:

It is recommended to lock the pragma to version 0.8.0 where the calculations are performed safely or directly use a library like `SafeMath`.

Remediation Plan:

SOLVED: The compiler version was updated to version 0.8.10.

3.22 (HAL-22) GAS OPTIMIZATIONS - INFORMATIONAL

Description:

Within the code, the most expensive variable computation usage was identified. While this does not pose any security risks to smart contracts, it does increase deployment costs.

Code Locations:

ERC20Template Line 46

Listing 100

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

Recommendation:

It is recommended to change the expression to `uint256 public constant BASE = 1e18.`

Remediation Plan:

SOLVED: The Ocean Protocol team amended the code to perform a more optimized computation.

AUTOMATED TESTING

4.1 STATIC ANALYSIS REPORT

Description:

Halborn used automated testing techniques to enhance coverage of certain areas of the in-scope contracts. After Halborn verified all the contracts in the repository and was able to compile them correctly into their application binary interface (ABI) and binary formats, the tool Slither was used. Slither is a Solidity static analysis framework. This tool can statically verify mathematical relationships between Solidity variables to detect invalid or inconsistent usage of the contracts' application programming interfaces (APIs) across the entire code-base.

Results:

```

ERC721Factory._createToken(uint256,string[],address[],uint256[],bytes[],address).tokenData (contracts/ERC721Factory.sol#329) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

ERC721Factory.startMultipleTokenOrder(ERC721Factory.tokenOrder[]) (contracts/ERC721Factory.sol#466-512) ignores return value by IERC20Template(publishMarketFeeToken).approve(orders[i].tokenAddress,publishMarketFeeAmount) (contracts/ERC721Factory.sol#484)
ERC721Factory.startMultipleTokenOrder(ERC721Factory.tokenOrder[]) (contracts/ERC721Factory.sol#466-512) ignores return value by IERC20Template(orders[i].consumeFeeToken).approve(orders[i].tokenAddress,orders[i].consumeFeeAmount) (contracts/ERC721Factory.sol#494)
ERC721Factory.createNftWithPool(ERC721Factory.NftCreateData,ERC721Factory.ErcCreateData,ERC721Factory.PoolData) (contracts/ERC721Factory.sol#573-606) ignores return value by IERC20Template(_PoolData.addresses[1]).approve(router,_PoolData.ssParams[4]) (contracts/ERC721Factory.sol#598)

ERC721Factory.constructor(address,address,address,address,_router) (contracts/ERC721Factory.sol#83) lacks a zero-check on :
    - router = _router (contracts/ERC721Factory.sol#93)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

ERC721Factory.startMultipleTokenOrder(ERC721Factory.tokenOrder[]) (contracts/ERC721Factory.sol#466-512) has external calls inside a loop: (publishMarketFeeAddress,publishMarketFeeToken,publishMarketFeeAmount) = IERC20Template(orders[i].tokenAddress).getPublishingMarketFee() (contracts/ERC721Factory.sol#473-474)
ERC721Factory.startMultipleTokenOrder(ERC721Factory.tokenOrder[]) (contracts/ERC721Factory.sol#466-512) has external calls inside a loop: require(bool,string)(IERC20Template(publishMarketFeeToken).transferFrom(msg.sender,address(this),publishMarketFeeAmount),Failed to transfer publishFee) (contracts/ERC721Factory.sol#479-483)
ERC721Factory.startMultipleTokenOrder(ERC721Factory.tokenOrder[]) (contracts/ERC721Factory.sol#466-512) has external calls inside a loop: (publishMarketFeeToken).approve(orders[i].tokenAddress,publishMarketFeeAmount) (contracts/ERC721Factory.sol#484)
ERC721Factory.startMultipleTokenOrder(ERC721Factory.tokenOrder[]) (contracts/ERC721Factory.sol#466-512) has external calls inside a loop: require(bool,string)(IERC20Template(orders[i].consumeFeeToken).transferFrom(msg.sender,address(this),orders[i].consumeFeeAmount),Failed to transfer consumeFee) (contracts/ERC721Factory.sol#489-493)
ERC721Factory.startMultipleTokenOrder(ERC721Factory.tokenOrder[]) (contracts/ERC721Factory.sol#466-512) has external calls inside a loop: IERC20Template(orders[i].consumeFeeToken).approve(orders[i].tokenAddress,orders[i].consumeFeeAmount) (contracts/ERC721Factory.sol#494)
ERC721Factory.startMultipleTokenOrder(ERC721Factory.tokenOrder[]) (contracts/ERC721Factory.sol#466-512) has external calls inside a loop: require(bool,string)(IERC20Template(orders[i].tokenAddress).transferFrom(msg.sender,address(this),orders[i].amount),Failed to transfer datatoken) (contracts/ERC721Factory.sol#497-501)
ERC721Factory.startMultipleTokenOrder(ERC721Factory.tokenOrder[]) (contracts/ERC721Factory.sol#466-512) has external calls inside a loop: IERC20Template(orders[i].tokenAddress).startOrder(orders[i].consumer,orders[i].amount,orders[i].serviceId,orders[i].consumeFeeAddress,orders[i].consumeFeeToken,orders[i].consumeFeeAmount) (contracts/ERC721Factory.sol#503-510)
Reentrancy in ERC721Factory.createNftWithErc(ERC721Factory.NftCreateData,ERC721Factory.ErcCreateData) (contracts/ERC721Factory.sol#540-557):
    External calls:
        - erc721Address = deployERC721Contract(_NftCreateData.name,_NftCreateData.symbol,_NftCreateData.templateIndex,address(0),_NftCreateData.baseURI) (contracts/ERC721Factory.sol#544-549)
            - require(bool,string)(tokenInstance.initialize(msg.sender,name,symbol,address(this),additionalERC20Deployer,baseURI),ERC721DTFactory: Unable to initialize token instance) (contracts/ERC721Factory.sol#135-145)
        - erc20Address = _createToken(_ErcCreateData.templateIndex,_ErcCreateData.strings,_ErcCreateData.addresses,_ErcCreateData.uints,_ErcCreateData.bytes,erc721Address) (contracts/ERC721Factory.sol#550-556)
            - require(bool,string)(tokenInstance.initialize(tokenData.strings,tokenData.addresses,factoryAddresses,tokenData.uints,tokenData.bytes),ERC20Factory: Unable to initialize token instance) (contracts/ERC721Factory.sol#348-357)
    State variables written after the call(s):
        - erc20Address = _createToken(_ErcCreateData.templateIndex,_ErcCreateData.strings,_ErcCreateData.addresses,_ErcCreateData.uints,_ErcCreateData.bytes,erc721Address) (contracts/ERC721Factory.sol#550-556)
        - erc20List[token] = true (contracts/ERC721Factory.sol#321)
Reentrancy in ERC721Factory.deployERC721Contract(string,string,uint256,address,string) (contracts/ERC721Factory.sol#107-149):
    External calls:
        - require(bool,string)(tokenInstance.initialize(msg.sender,name,symbol,address(this),additionalERC20Deployer,baseURI),ERC721DTFactory: Unable to initialize token instance) (contracts/ERC721Factory.sol#135-145)
    State variables written after the call(s):
        - currentNFTCount += 1 (contracts/ERC721Factory.sol#148)
Event emitted after the call(s):
    - NFTCreated(token,tokenTemplate.templateAddress,name,msg.sender) (contracts/ERC721Factory.sol#147)

Reentrancy in ERC721Factory.deployERC721Contract(string,string,uint256,address,string) (contracts/ERC721Factory.sol#107-149):
    External calls:
        - require(bool,string)(tokenInstance.initialize(msg.sender,name,symbol,address(this),additionalERC20Deployer,baseURI),ERC721DTFactory: Unable to initialize token instance) (contracts/ERC721Factory.sol#135-145)
    Event emitted after the call(s):
        - NFTCreated(token,tokenTemplate.templateAddress,name,msg.sender) (contracts/ERC721Factory.sol#147)

```

```

ERC721Factory.isContract(address) (contracts/ERC721Factory.sol#247-258) uses assembly
  - INLINE ASM (contracts/ERC721Factory.sol#254-256)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

ERC721Factory.deployERC721Contract(string,string,uint256,address,string) (contracts/ERC721Factory.sol#107-149) compares to a boolean constant:
  -require(bool,string)(tokenTemplate.isActive == true,ERC721DFactory: ERC721Token Template disabled) (contracts/ERC721Factory.sol#120-123)
ERC721Factory._createToken(uint256,string[],address[],uint256[],bytes[],address) (contracts/ERC721Factory.sol#301-336) compares to a boolean constant:
  -require(bool,string)(tokenTemplate.isActive == true,ERC20Factory: ERC721Token Template disabled) (contracts/ERC721Factory.sol#316-319)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Pragma version>=0.6.0 (contracts/ERC721Factory.sol#1) allows old versions
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

ERC721Factory.erc20Factory (contracts/ERC721Factory.sol#25) is never used in ERC721Factory (contracts/ERC721Factory.sol#22-689)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

createToken(uint256,string[],address[],uint256[],bytes[]) should be declared external:
  - ERC721Factory.createToken(uint256,string[],address[],uint256[],bytes[]) (contracts/ERC721Factory.sol#287-300)

OPFCommunityFeeCollector.withdrawETH() (contracts/communityFee/OPFCommunityFeeCollector.sol#51-56) sends eth to arbitrary user
  Dangerous calls:
    - collector.transfer(address(this).balance) (contracts/communityFee/OPFCommunityFeeCollector.sol#55)

FactoryRouter.buyDTBatch.FactoryRouter.Operations[] (contracts/pools/FactoryRouter.sol#262-330) ignores return value by IERC20(_operations[i].tokenIn).transferFrom(msg.sender,addr
ess(this),_operations[i].amountIn) (contracts/pools/FactoryRouter.sol#271)
FactoryRouter.buyDTBatch.FactoryRouter.Operations[] (contracts/pools/FactoryRouter.sol#262-330) ignores return value by IERC20(_operations[i].tokenIn).transferFrom(msg.sender,addr
ess(this),amountIn) (contracts/pools/FactoryRouter.sol#298)
FactoryRouter.buyDTBatch.FactoryRouter.Operations[] (contracts/pools/FactoryRouter.sol#262-330) ignores return value by IERC20(_operations[i].tokenIn).transferFrom(msg.sender,addr
ess(this),baseTokenAmount) (contracts/pools/FactoryRouter.sol#314)
FactoryRouter.buyDTBatch.FactoryRouter.Operations[] (contracts/pools/FactoryRouter.sol#262-330) ignores return value by IERC20(dataToken).transfer(msg.sender,_operations[i].amount
Out) (contracts/pools/FactoryRouter.sol#321)
BPool.collectOPF (contracts/pools/balancer/BPool.sol#253-260) ignores return value by IERC20(tokens[i]).transfer(opfCollector,amount) (contracts/pools/balancer/BPool.sol#258)
BPool.collectMarketFee(address) (contracts/pools/balancer/BPool.sol#262-271) ignores return value by IERC20(tokens[i]).transfer(to,amount) (contracts/pools/balancer/BPool.sol#269)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

Reentrancy in BPool.existswapExternAmountOut(address,uint256,uint256) (contracts/pools/balancer/BPool.sol#25-1004):
  External calls:
    - _pushUnderlying(tokenOut,msg.sender,tokenAmountIn) (contracts/pools/balancer/BPool.sol#970)
      - xfer = IERC20(erc20).transferTo(amount) (contracts/pools/balancer/BPool.sol#1027)
  State variables written after the call(s):
    - _pullPoolShare_controller.poolAmountIn (contracts/pools/balancer/BPool.sol#994)
      - _balance[src] = bsub(balance[src],amt) (contracts/pools/balancer/BToken.sol#63)
      - _balance[dst] = badd(balance[dst],amt) (contracts/pools/balancer/BToken.sol#64)
    - _burnPoolShare(bsub(poolAmountIn,exitFee)) (contracts/pools/balancer/BPool.sol#997)
      - _balance[address(this)] = bsub(balance[address(this)],amt) (contracts/pools/balancer/BToken.sol#56)
    - _pushPoolShare_factory.exitFee (contracts/pools/balancer/BPool.sol#998)
      - _balance[src] = bsub(balance[src],amt) (contracts/pools/balancer/BToken.sol#63)
      - _balance[dst] = badd(balance[dst],amt) (contracts/pools/balancer/BToken.sol#64)
    - ssOutRecord.balance = bsusb(ssOutRecord.balance,ssAmountOut) (contracts/pools/balancer/BPool.sol#988)
    - _burnPoolShare(bsub(poolAmountIn,exitFee)) (contracts/pools/balancer/BPool.sol#997)
    - _pushPoolShare_totalSupply = bsub(_totalSupply,amt) (contracts/pools/balancer/BToken.sol#57)
    - _burnPoolShareSupply = bsub(_totalSupply,amt) (contracts/pools/balancer/BToken.sol#57)
  Reentrancy in BPool.existswapPoolAmountIn(address,uint256,uint256) (contracts/pools/balancer/BPool.sol#838-923):
  External calls:
    - _pushUnderlying(tokenOut,msg.sender,tokenAmountOut) (contracts/pools/balancer/BPool.sol#887)
      - xfer = IERC20(erc20).transferTo(amount) (contracts/pools/balancer/BPool.sol#1027)
  State variables written after the call(s):
    - _pullPoolShare_controller.poolAmountIn (contracts/pools/balancer/BPool.sol#195)
      - _balance[src] = bsub(balance[src],amt) (contracts/pools/balancer/BToken.sol#63)
      - _balance[dst] = badd(balance[dst],amt) (contracts/pools/balancer/BToken.sol#64)
    - _burnPoolShare(bsub(poolAmountIn,exitFee)) (contracts/pools/balancer/BPool.sol#916)
      - _balance[src] = bsub(balance[src],amt) (contracts/pools/balancer/BToken.sol#56)
    - _pushPoolShare_factory.exitFee (contracts/pools/balancer/BPool.sol#917)
      - _balance[src] = bsub(balance[src],amt) (contracts/pools/balancer/BToken.sol#63)
      - _balance[dst] = badd(balance[dst],amt) (contracts/pools/balancer/BToken.sol#64)
  Reentrancy in BPool.setupAddress(uint256,uint256,address,uint256,uint256) (contracts/pools/balancer/BPool.sol#175-213):
  External calls:
    - bind(dataTokenAddress,dataTokenAmount,dataTokenHeight) (contracts/pools/balancer/BPool.sol#195)
      - xfer = IERC20(erc20).transferFrom(from,dest(this),amount) (contracts/pools/balancer/BPool.sol#1016)
        - xfer = IERC20(erc20).transferTo(amount) (contracts/pools/balancer/BPool.sol#1027)
    - bind(baseTokenAddress,baseTokenAmount,baseTokenHeight) (contracts/pools/balancer/BPool.sol#204)
      - xfer = IERC20(erc20).transferFrom(from,dest(this),amount) (contracts/pools/balancer/BPool.sol#1016)
        - xfer = IERC20(erc20).transferTo(amount) (contracts/pools/balancer/BPool.sol#1027)
  State variables written after the call(s):
    - finalized() (contracts/pools/balancer/BPool.sol#12)
      - _finalized = true (contracts/pools/balancer/BPool.sol#36)
    - bind(baseTokenAddress,baseTokenAmount,baseTokenHeight) (contracts/pools/balancer/BPool.sol#204)
      - _records[token].denom = bsub(balance[token],records[token].balance) (contracts/pools/balancer/BPool.sol#364-369)
      - _records[token].balance = balance (contracts/pools/balancer/BPool.sol#305)
    - bind(baseTokenAddress,baseTokenAmount,baseTokenHeight) (contracts/pools/balancer/BPool.sol#204)
      - _tokens.push(token) (contracts/pools/balancer/BPool.sol#370)
    - bind(baseTokenAddress,baseTokenAmount,baseTokenHeight) (contracts/pools/balancer/BPool.sol#204)
      - _totalWeight = badd(_totalWeight,bsub(oldWeight,denom)) (contracts/pools/balancer/BPool.sol#386)
      - _totalWeight = bsub(_totalWeight,bsub(oldWeight,denom)) (contracts/pools/balancer/BPool.sol#389)
  Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

FactoryRouter.deployPool(address[],uint256[],uint256[],address[],flag) (contracts/pools/FactoryRouter.sol#137) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

FactoryRouter.buyDTBatch.FactoryRouter.Operations[] (contracts/pools/FactoryRouter.sol#262-330) ignores return value by IERC20(_operations[i].tokenIn).approve(_operations[i].source
e,_operations[i].amountIn) (contracts/pools/FactoryRouter.sol#273)
FactoryRouter.buyDTBatch.FactoryRouter.Operations[] (contracts/pools/FactoryRouter.sol#262-330) ignores return value by IERC20(_operations[i].tokenIn).approve(_operations[i].source
e,_amountIn) (contracts/pools/FactoryRouter.sol#298)
FactoryRouter.buyDTBatch.FactoryRouter.Operations[] (contracts/pools/FactoryRouter.sol#262-330) ignores return value by IPool(_operations[i].source).swapExactAmountOut(_operations
[i].tokenIn,_operations[i].amountIn,_operations[i].operations[i].amountOut,_operations[i].maxPrice) (contracts/pools/FactoryRouter.sol#294-299)
FactoryRouter.buyDTBatch.FactoryRouter.Operations[] (contracts/pools/FactoryRouter.sol#262-330) ignores return value by IERC20(_operations[i].tokenIn).approve(_operations[i].source
e,baseTokenAmount) (contracts/pools/FactoryRouter.sol#316)
BFactory.newBPool(address[],uint256[],uint256[],address[],) (contracts/pools/balancer/BFactory.sol#79-127) ignores return value by ISideStaking(addresses[0]).newDataTokenCreated(to
kens[0],tokens[1],bpool.addresses[3],ssParams) (contracts/pools/balancer/BFactory.sol#117-122)
  
```

```

FactoryRouter.changeRouterOwner(address) (contracts/pools/FactoryRouter.sol#46-52) should emit an event for:
- routerOwner = _routerOwner (contracts/pools/FactoryRouter.sol#51)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control

FactoryRouter.updateOPFee(uint256) (contracts/pools/FactoryRouter.sol#84-87) should emit an event for:
- swapOceanFee = _newSwapOceanFee (contracts/pools/FactoryRouter.sol#86)
BPool.setSwapFee(uint256) (contracts/pools/balancer/BPool.sol#335-341) should emit an event for:
- _swapFee = swapFee (contracts/pools/balancer/BPool.sol#340)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

FactoryRouter.constructor(address,address,address,address[]) (contracts/pools/FactoryRouter.sol#35) lacks a zero-check on :
- routerOwner = _routerOwner (contracts/pools/FactoryRouter.sol#31)
FactoryRouter.constructor(address,address,address,address[],uint256) (contracts/pools/FactoryRouter.sol#42) lacks a zero-check on :
- opfCollector = _opfCollector (contracts/pools/FactoryRouter.sol#42)
FactoryRouter.addFactoryAddress(_factory) (contracts/pools/FactoryRouter.sol#66) lacks a zero-check on :
- _factory = factory (contracts/pools/FactoryRouter.sol#68)
BPool.updateMarketFeeCollector(address) (contracts/pools/balancer/BPool.sol#273) lacks a zero-check on :
- _marketFeeCollector = _newCollector (contracts/pools/balancer/BPool.sol#275)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

FactoryRouter.buyDTBatch(FactoryRouter.Operations[]) (contracts/pools/FactoryRouter.sol#262-330) has external calls inside a loop: IERC20(_operations[i].tokenIn).transferFrom(msg.sender,address(this),_operations[i].amountIn) (contracts/pools/FactoryRouter.sol#271)
FactoryRouter.buyDTBatch(FactoryRouter.Operations[]) (contracts/pools/FactoryRouter.sol#262-330) has external calls inside a loop: IERC20(_operations[i].tokenIn).approve(_operations[i].source,_operations[i].amountOut) (contracts/pools/FactoryRouter.sol#273)
FactoryRouter.buyDTBatch(FactoryRouter.Operations[]) (contracts/pools/FactoryRouter.sol#262-330) has external calls inside a loop: (amountReceived) = IPool(_operations[i].source).swapExactAmountIn(_operations[i].tokenIn,_operations[i].amountIn,_operations[i].amountOut,_operations[i].maxPrice) (contracts/pools/FactoryRouter.sol#275-281)
FactoryRouter.buyDTBatch(FactoryRouter.Operations[]) (contracts/pools/FactoryRouter.sol#262-330) has external calls inside a loop: require(bool,string)(IERC20(_operations[i].tokenOut).transfer(msg.sender,amountReceived),Failed MultiSwap) (contracts/pools/FactoryRouter.sol#284)
FactoryRouter.buyDTBatch(FactoryRouter.Operations[]) (contracts/pools/FactoryRouter.sol#262-330) has external calls inside a loop: amountIn = IPool(_operations[i].source).getAmountIn(address(this),amountIn) (contracts/pools/FactoryRouter.sol#285)
FactoryRouter.buyDTBatch(FactoryRouter.Operations[]) (contracts/pools/FactoryRouter.sol#262-330) has external calls inside a loop: IERC20(_operations[i].tokenIn).transferFrom(msg.sender,address(this),amountIn) (contracts/pools/FactoryRouter.sol#290)
FactoryRouter.buyDTBatch(FactoryRouter.Operations[]) (contracts/pools/FactoryRouter.sol#262-330) has external calls inside a loop: IERC20(_operations[i].tokenIn).approve(_operations[i].source,amountIn) (contracts/pools/FactoryRouter.sol#292)
FactoryRouter.buyDTBatch(FactoryRouter.Operations[]) (contracts/pools/FactoryRouter.sol#262-330) has external calls inside a loop: IPool(_operations[i].source).swapExactAmountOut(_operations[i].tokenIn,_operations[i].amountIn,_operations[i].amountOut,_operations[i].maxPrice) (contracts/pools/FactoryRouter.sol#294-299)
FactoryRouter.buyDTBatch(FactoryRouter.Operations[]) (contracts/pools/FactoryRouter.sol#262-330) has external calls inside a loop: require(bool,string)(IERC20(_operations[i].tokenOut).transfer(msg.sender,operations[i].amountOut),Failed MultiSwap) (contracts/pools/FactoryRouter.sol#301-302)
FactoryRouter.buyDTBatch(FactoryRouter.Operations[]) (contracts/pools/FactoryRouter.sol#262-330) has external calls inside a loop: (dataToken) = IFixedRateExchange(_operations[i].source).getExchange(_operations[i].exchangeId) (contracts/pools/FactoryRouter.sol#306-307)
FactoryRouter.buyDTBatch(FactoryRouter.Operations[]) (contracts/pools/FactoryRouter.sol#262-330) has external calls inside a loop: (baseTokenAmount) = IFixedRateExchange(_operations[i].source).calcBaseInGivenOut(_operations[i].exchangeId,_operations[i].amountsOut) (contracts/pools/FactoryRouter.sol#309-311)

BPool.collectMarketFee(address) (contracts/pools/balancer/BPool.sol#262-271) has external calls inside a loop: IERC20(tokens[i]).transfer(to,amount) (contracts/pools/balancer/BPool.sol#269)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#calls-inside-a-loop

Reentrancy in BPool.setup(address,uint256,uint256,address,uint256,uint256) (contracts/pools/balancer/BPool.sol#175-213):
External calls:
- bind(dataTokenAddress,dataTokenAmount,dataTokenWeight) (contracts/pools/balancer/BPool.sol#195)
  - xfer = IERC20(erc20).transferFrom(from,address(this),amount) (contracts/pools/balancer/BPool.sol#1016)
  - xfer = IERC20(erc20).transferTo(amount) (contracts/pools/balancer/BPool.sol#1027)
- bind(basicPoolShare,ssBalance,ssTokenWeight) (contracts/pools/balancer/BPool.sol#204)
  - xfer = IERC20(erc20).transferFrom(from,address(this),amount) (contracts/pools/balancer/BPool.sol#1016)
  - xfer = IERC20(erc20).transferTo(amount) (contracts/pools/balancer/BPool.sol#1027)
State variables written after the call(s):
- finalize() (contracts/pools/balancer/BPool.sol#212)
  - _balance(address(this)) = badd(_balance(address(this)),amt) (contracts/pools/balancer/BToken.sol#46)
  - _balance[src] = bsub(_balance[src],amt) (contracts/pools/balancer/BTken.sol#63)
  - _balance[dst] = badd(_balance[dst],amt) (contracts/pools/balancer/BTken.sol#64)
- finalize() (contracts/pools/balancer/BPool.sol#212)
  - _pullSwap = true (contracts/pools/balancer/BPool.sol#347)
- finalize() (contracts/pools/balancer/BPool.sol#212)
  - _totalSupply = badd(_totalSupply,amt) (contracts/pools/balancer/BTken.sol#47)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2

Reentrancy in BPool.exitSwapExternalAmountOut(address,uint256,uint256) (contracts/pools/balancer/BPool.sol#925-1004):
External calls:
- _pushUnderlying((tokenOut,msg.sender,tokenAmountOut)) (contracts/pools/balancer/BPool.sol#970)
  - xfer = IERC20(erc20).transferTo(amount) (contracts/pools/balancer/BPool.sol#1027)
Event emitted after the call(s):
- LOG_EXIT(controller,ssStakeToken,ssAmountOut.block.timestamp) (contracts/pools/balancer/BPool.sol#998-995)
- Transfer(address(this),address(),amt) (contracts/pools/balancer/BTken.sol#58)
  - _burnPoolShare(bsub(poolAmountIn,exitFee)) (contracts/pools/balancer/BPool.sol#8997)
- Transfer(src,dst,amt) (contracts/pools/balancer/BTken.sol#65)
  - _pushPoolShare(_factory,exitFee) (contracts/pools/balancer/BPool.sol#998)
- Transfer(src,dst,amt) (contracts/pools/balancer/BTken.sol#65)
  - _pushPoolShare(_factory,exitFee) (contracts/pools/balancer/BPool.sol#998)
Reentrancy in BPool.exitSwapPoolAmountIn(address,uint256,uint256) (contracts/pools/balancer/BPool.sol#838-923):
External calls:
- _pushUnderlying((tokenOut,msg.sender,tokenAmountOut)) (contracts/pools/balancer/BPool.sol#887)
  - xfer = IERC20(erc20).transferTo(amount) (contracts/pools/balancer/BPool.sol#1027)
Event emitted after the call(s):
- LOG_EXIT(controller,ssStakeToken,ssAmountOut.block.timestamp) (contracts/pools/balancer/BPool.sol#899-914)
- Transfer(addr(this),addr(),amt) (contracts/pools/balancer/BTken.sol#58)
  - _burnPoolShare(bsub(poolAmountIn,exitFee)) (contracts/pools/balancer/BPool.sol#916)
FactoryRouter.getOPFee(address) (contracts/pools/FactoryRouter.sol#78-82) compares to a boolean constant:
-oceanTokens(baseToken) == true (contracts/pools/FactoryRouter.sol#879)
FactoryRouter.deployPool(address[2],uint256[],uint256[],address[]) (contracts/pools/FactoryRouter.sol#117-159) compares to a boolean constant:
-require(bool,string)(IFactory(factory).erc20List(msg.sender) == true,FACTORY ROUTER: NOT ORIGINAL ERC20 TEMPLATE) (contracts/pools/FactoryRouter.sol#126-129)
FactoryRouter.deployPool(address[2],uint256[],uint256[],address[]) (contracts/pools/FactoryRouter.sol#117-159) compares to a boolean constant:
-require(bool,string)(ssContracts[addresses[]] == true,FACTORY ROUTER: invalid ssContract) (contracts/pools/FactoryRouter.sol#130-133)
FactoryRouter.deployPool(address[2],uint256[],uint256[],address[]) (contracts/pools/FactoryRouter.sol#117-159) compares to a boolean constant:
-oceanTokens(baseToken[1]) == true (contracts/pools/FactoryRouter.sol#138)
FactoryRouter.deployFixedRate(address,uint256[],uint256[],address[]) (contracts/pools/FactoryRouter.sol#176-197) compares to a boolean constant:
-require(bool,string)(IFactory(factory).erc20List(msg.sender) == true,FACTORY ROUTER: NOT ORIGINAL ERC20 TEMPLATE) (contracts/pools/FactoryRouter.sol#182-185)
FactoryRouter.deployFixedRate(address,uint256[],uint256[],address[]) (contracts/pools/FactoryRouter.sol#176-197) compares to a boolean constant:
-require(bool,string)(fixedPrice(fixedPriceAddress) == true,FACTORY ROUTER: Invalid FixedPriceContract) (contracts/pools/FactoryRouter.sol#187-198)
FactoryRouter.deployDispenser(address,uint256,uint256,address,address) (contracts/pools/FactoryRouter.sol#212-230) compares to a boolean constant:
-require(bool,string)(dispenser._dispenser == true,FACTORY ROUTER: Invalid DispenserContract) (contracts/pools/FactoryRouter.sol#225-228)
FactoryRouter.deployDispenser(address,uint256,uint256,address,address) (contracts/pools/FactoryRouter.sol#212-230) compares to a boolean constant:
-require(bool,string)(IFactory(factory).erc20List(msg.sender) == true,FACTORY ROUTER: NOT ORIGINAL ERC20 TEMPLATE) (contracts/pools/FactoryRouter.sol#228-223)
BFactory.newBPool(address[2],uint256[],uint256[],address[]) (contracts/pools/balancer/BFactory.sol#127) compares to a boolean constant:
-require(bool,string)(poolTemplates[addresses[]] == true,BFactory: Wrong Pool Template) (contracts/pools/balancer/BFactory.sol#88)
BPool.joinSwapExternalIn(address,uint256,uint256,uint256) (contracts/pools/balancer/BPool.sol#78-82) compares to a boolean constant:
-ssContract.canStakeOut(address,uint256,uint256) (contracts/pools/balancer/BPool.sol#76-80) compares to a boolean constant:
-ssContract.canStake(dataTokenAddress,ssStakeToken,ssAmountOut) == true (contracts/pools/balancer/BPool.sol#81-85)
BPool.exitSwapExternalOut(address,uint256,uint256,uint256) (contracts/pools/balancer/BPool.sol#838-922) compares to a boolean constant:
-ssContract.canUnStake(dataTokenAddress,ssStakeToken,poolAmountIn) == true (contracts/pools/balancer/BPool.sol#991-995)
BPool.exitSwapExternalOut(address,uint256,uint256,uint256) (contracts/pools/balancer/BPool.sol#925-1004) compares to a boolean constant:
-ssContract.canUnStake(dataTokenAddress,ssStakeToken,ssAmountOut) == true (contracts/pools/balancer/BPool.sol#982-986)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

FactoryRouter.getLength(IERC20[]) (contracts/pools/FactoryRouter.sol#161-163) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Praga version>=0.5.7 (contracts/pools/FactoryRouter.sol#4) allows old versions
Praga version>=0.5.7 (contracts/pools/balancer/BConst.sol#14) allows old versions
Praga version>=0.5.7 (contracts/pools/balancer/BFactory.sol#1) allows old versions
Praga version>=0.5.7 (contracts/pools/balancer/BMath.sol#14) allows old versions
Praga version>=0.5.7 (contracts/pools/balancer/BNum.sol#14) allows old versions
Praga version>=0.6.0 (contracts/pools/balancer/BPool.sol#2) allows old versions
Praga version>=0.5.7 (contracts/pools/balancer/BTken.sol#14) allows old versions
solc>=0.6.0 is not recommended for deployment

```

```

Dispenser.dispense(address,uint256,address) (contracts/pools/dispenser/Dispenser.sol#107-229) ignores return value by tokenInstance.transfer(destination,amount) (contracts/pools/dispenser/Dispenser.sol#227)
Dispenser.ownerWithdraw(address) (contracts/pools/dispenser/Dispenser.sol#236-251) ignores return value by tokenInstance.transfer(msg.sender,ourBalance) (contracts/pools/dispenser/Dispenser.sol#248)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

Contract locking ether found:
  Contract Dispenser (contracts/pools/dispenser/Dispenser.sol#10-252) has payable functions:
    - Dispenser.dispense(address,uint256,address) (contracts/pools/dispenser/Dispenser.sol#107-229)
      But does not have a function to withdraw the ether

FixedRateExchange.buyDT(bytes32,uint256,uint256) (contracts/pools/fixedRate/FixedRateExchange.sol#33-412) ignores return value by IERC20Template(exchanges[exchangeId].dataToken).transfer(msg.sender,dataTokenAmount) (contracts/pools/fixedRate/FixedRateExchange.sol#33-412)
FixedRateExchange.getDTSupply(bytes32) (contracts/pools/fixedRate/FixedRateExchange.sol#421-492) ignores return value by IERC20Template(exchanges[exchangeId].baseToken).transfer(msg.sender,baseTokenAmount) (contracts/pools/fixedRate/FixedRateExchange.sol#421-492)
FixedRateExchange.collectDT(bytes32) (contracts/pools/fixedRate/FixedRateExchange.sol#494-511) ignores return value by IERC20Template(exchanges[exchangeId].baseToken).transfer(exchangeOwner,amount) (contracts/pools/fixedRate/FixedRateExchange.sol#500-503)
FixedRateExchange.collectDT(bytes32) (contracts/pools/fixedRate/FixedRateExchange.sol#513-530) ignores return value by IERC20Template(exchanges[exchangeId].dataToken).transfer(exchangeOwner,amount) (contracts/pools/fixedRate/FixedRateExchange.sol#519-522)
FixedRateExchange.collectMarketFee(bytes32) (contracts/pools/fixedRate/FixedRateExchange.sol#532-545) ignores return value by IERC20Template(exchanges[exchangeId].baseToken).transfer(exchangeCollector,amount) (contracts/pools/fixedRate/FixedRateExchange.sol#536-539)
FixedRateExchange.collectOceanFee(bytes32) (contracts/pools/fixedRate/FixedRateExchange.sol#547-560) ignores return value by IERC20Template(exchanges[exchangeId].baseToken).transfer(opfCollector,amount) (contracts/pools/fixedRate/FixedRateExchange.sol#551-554)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

FixedRateExchange.calcBaseInGivenOutDT(bytes32,uint256) (contracts/pools/fixedRate/FixedRateExchange.sol#248-287) performs a multiplication on the result of a division:
  - baseTokenAmountBeforeFee = dataTokenAmount.mul(exchanges[exchangeId].fixedRate).div(BASE).mul(10 ** exchanges[exchangeId].dtDecimals) (contracts/pools/fixedRate/FixedRateExchange.sol#259-263)
FixedRateExchange.calcBaseInGivenOutDT(bytes32,uint256) (contracts/pools/fixedRate/FixedRateExchange.sol#248-282) performs a multiplication on the result of a division:
  - baseTokenAmountBeforeFee = dataTokenAmount.mul(exchanges[exchangeId].fixedRate).div(BASE).mul(10 ** exchanges[exchangeId].dtDecimals) (contracts/pools/fixedRate/FixedRateExchange.sol#259-263)
  - oceanFeeAmount = baseTokenAmountBeforeFee.mul(getOpFee(exchanges[exchangeId].baseToken)).div(BASE) (contracts/pools/fixedRate/FixedRateExchange.sol#268-270)
FixedRateExchange.calcBaseOutGivenInDT(bytes32,uint256) (contracts/pools/fixedRate/FixedRateExchange.sol#248-282) performs a multiplication on the result of a division:
  - baseTokenAmountBeforeFee = dataTokenAmount.mul(exchanges[exchangeId].fixedRate).div(BASE).mul(10 ** exchanges[exchangeId].dtDecimals) (contracts/pools/fixedRate/FixedRateExchange.sol#259-263)
  - marketFeeAmount = baseTokenAmountBeforeFee.mul(exchanges[exchangeId].marketFee).div(BASE) (contracts/pools/fixedRate/FixedRateExchange.sol#273-275)
FixedRateExchange.calcBaseOutGivenInDT(bytes32,uint256) (contracts/pools/fixedRate/FixedRateExchange.sol#290-324) performs a multiplication on the result of a division:
  - baseTokenAmountBeforeFee = dataTokenAmount.mul(exchanges[exchangeId].fixedRate).div(BASE).mul(10 ** exchanges[exchangeId].dtDecimals) (contracts/pools/fixedRate/FixedRateExchange.sol#301-305)
FixedRateExchange.calcBaseOutGivenInDT(bytes32,uint256) (contracts/pools/fixedRate/FixedRateExchange.sol#290-324) performs a multiplication on the result of a division:
  - baseTokenAmountBeforeFee = dataTokenAmount.mul(exchanges[exchangeId].fixedRate).div(BASE).mul(10 ** exchanges[exchangeId].dtDecimals) (contracts/pools/fixedRate/FixedRateExchange.sol#301-312)
  - oceanFeeAmount = baseTokenAmountBeforeFee.mul(getOpFee(exchanges[exchangeId].baseToken)).div(BASE) (contracts/pools/fixedRate/FixedRateExchange.sol#310-312)
FixedRateExchange.calcBaseOutGivenInDT(bytes32,uint256) (contracts/pools/fixedRate/FixedRateExchange.sol#290-324) performs a multiplication on the result of a division:
  - baseTokenAmountBeforeFee = dataTokenAmount.mul(exchanges[exchangeId].fixedRate).div(BASE).mul(10 ** exchanges[exchangeId].dtDecimals) (contracts/pools/fixedRate/FixedRateExchange.sol#301-305)
  - marketFeeAmount = baseTokenAmountBeforeFee.mul(exchanges[exchangeId].marketFee).div(BASE) (contracts/pools/fixedRate/FixedRateExchange.sol#315-317)

Reentrancy in Dispenser.dispense(address,uint256,address) (contracts/pools/dispenser/Dispenser.sol#187-229):
  External calls:
    - tokenInstance.mint(address(this),amount - ourBalance) (contracts/pools/dispenser/Dispenser.sol#220)
    - tokenInstance.transfer(destination,amount) (contracts/pools/dispenser/Dispenser.sol#227)
  Event emitted after the call(s):
    - TokensDispensed(datatoken,destination,amount) (contracts/pools/dispenser/Dispenser.sol#228)

Reentrancy in Dispenser.ownerWithdraw(address) (contracts/pools/dispenser/Dispenser.sol#236-251):
  External calls:
    - tokenInstance.transfer(msg.sender,ourBalance) (contracts/pools/dispenser/Dispenser.sol#248)
  Event emitted after the call(s):
    - OwnerWithdrawn(datatoken,msg.sender,ourBalance) (contracts/pools/dispenser/Dispenser.sol#249)

Reentrancy in FixedRateExchange.buyDT(bytes32,uint256,uint256) (contracts/pools/fixedRate/FixedRateExchange.sol#333-412):
  External calls:
    - require(bool,string)(IERC20Template(exchanges[exchangeId].baseToken).transferFrom(msg.sender,address(this)),baseTokenAmount),FixedRateExchange: transferFrom failed in the baseToken contract) (contracts/pools/fixedRate/FixedRateExchange.sol#334-371)
    - IERC20Template(exchanges[exchangeId].dataToken).mint(msg.sender,dataTokenAmount) (contracts/pools/fixedRate/FixedRateExchange.sol#382)
    - require(bool,string)(IERC20Template(exchanges[exchangeId].dataToken).transferFrom(exchangeOwner,msg.sender,dataTokenAmount),FixedRateExchange: transferFrom failed in the dataToken contract) (contracts/pools/fixedRate/FixedRateExchange.sol#385-392)
    - IERC20Template(exchanges[exchangeId].dataToken).transfer(msg.sender,dataTokenAmount) (contracts/pools/fixedRate/FixedRateExchange.sol#397-408)
  Event emitted after the call(s):
    - Swaped(exchangeId,msg.sender,baseTokenAmount,dataTokenAmount,exchanges[exchangeId].dataToken,marketFeeAmount,oceanFeeAmount) (contracts/pools/fixedRate/FixedRateExchange.sol#403-412)

Reentrancy in FixedRateExchange.collectDT(bytes32) (contracts/pools/fixedRate/FixedRateExchange.sol#494-511):
  External calls:
    - IERC20Template(exchanges[exchangeId].baseToken).transfer(exchanges[exchangeId].exchangeOwner,amount) (contracts/pools/fixedRate/FixedRateExchange.sol#500-503)
  Event emitted after the call(s):
    - TokenCollected(exchangeId,exchanges[exchangeId].exchangeOwner,exchanges[exchangeId].baseToken,amount) (contracts/pools/fixedRate/FixedRateExchange.sol#505-510)

Reentrancy in FixedRateExchange.collectDT(bytes32) (contracts/pools/fixedRate/FixedRateExchange.sol#513-530):
  External calls:
    - IERC20Template(exchanges[exchangeId].dataToken).transfer(exchanges[exchangeId].exchangeOwner,amount) (contracts/pools/fixedRate/FixedRateExchange.sol#519-522)
  Event emitted after the call(s):
    - TokenCollected(exchangeId,exchanges[exchangeId].exchangeOwner,exchanges[exchangeId].dataToken,amount) (contracts/pools/fixedRate/FixedRateExchange.sol#524-529)

Reentrancy in FixedRateExchange.collectMarketFee(bytes32) (contracts/pools/fixedRate/FixedRateExchange.sol#532-545):
  External calls:
    - IERC20Template(exchanges[exchangeId].baseToken).transfer(exchanges[exchangeId].marketFeeCollector,amount) (contracts/pools/fixedRate/FixedRateExchange.sol#536-539)
  Event emitted after the call(s):
    - MarketFeeCollected(exchangeId,exchanges[exchangeId].baseToken,amount) (contracts/pools/fixedRate/FixedRateExchange.sol#540-544)

Reentrancy in FixedRateExchange.collectOpFee(bytes32) (contracts/pools/fixedRate/FixedRateExchange.sol#551-554):
  External calls:
    - IERC20Template(exchanges[exchangeId].baseToken).transfer(opfCollector,amount) (contracts/pools/fixedRate/FixedRateExchange.sol#551-554)
  Event emitted after the call(s):
    - OpFeeCollected(exchangeId,exchanges[exchangeId].baseToken,amount) (contracts/pools/fixedRate/FixedRateExchange.sol#555-559)

Reentrancy in FixedRateExchange.collectDT(bytes32,uint256,uint256) (contracts/pools/fixedRate/FixedRateExchange.sol#421-492):
  External calls:
    - require(bool,string)(IERC20Template(exchanges[exchangeId].dataToken).transferFrom(msg.sender,address(this)),dataTokenAmount),FixedRateExchange: transferFrom failed in the dataToken contract) (contracts/pools/fixedRate/FixedRateExchange.sol#452-459)
    - require(bool,string)(IERC20Template(exchanges[exchangeId].baseToken).transferFrom(exchangeOwner,msg.sender,baseTokenAmount),FixedRateExchange: transferFrom failed in the baseToken contract) (contracts/pools/fixedRate/FixedRateExchange.sol#466-473)
    - IERC20Template(exchanges[exchangeId].baseToken).transfer(msg.sender,baseTokenAmount) (contracts/pools/fixedRate/FixedRateExchange.sol#477-480)

Dispenser.dispense(address,uint256,address) (contracts/pools/dispenser/Dispenser.sol#107-229) compares to a boolean constant:
  - require(bool,string)(datatokens[datatoken].active == true,Dispenser not active) (contracts/pools/dispenser/Dispenser.sol#192-195)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Pragma version>=0.5.7 (contracts/pools/dispenser/Dispenser.sol#1) allows old versions
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

FixedRateExchange.getDTSupply(bytes32) (contracts/pools/fixedRate/FixedRateExchange.sol#670-690) compares to a boolean constant:
  - exchanges[exchangeId].active == false (contracts/pools/fixedRate/FixedRateExchange.sol#675)
FixedRateExchange.getDTSupply(bytes32) (contracts/pools/fixedRate/FixedRateExchange.sol#670-690) compares to a boolean constant:
  - exchanges[exchangeId].withMint == true && true IERC20Template(exchanges[exchangeId].dataToken).isMinter(address(this)) (contracts/pools/fixedRate/FixedRateExchange.sol#676-677)
FixedRateExchange.getDTSupply(bytes32) (contracts/pools/fixedRate/FixedRateExchange.sol#699-714) compares to a boolean constant:
  - exchanges[exchangeId].active == false (contracts/pools/fixedRate/FixedRateExchange.sol#708)
FixedRateExchange.onlyActiveExchange(bytes32) (contracts/pools/fixedRate/FixedRateExchange.sol#50-57) compares to a boolean constant:
  - require(bool,string)(exchanges[exchangeId].active == true,FixedRateExchange: Exchange does not exist!) (contracts/pools/fixedRate/FixedRateExchange.sol#51-55)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Pragma version>=0.5.7 (contracts/pools/fixedRate/FixedRateExchange.sol#1) allows old versions
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Redundant expression "oceanFeeAmount (contracts/pools/fixedRate/FixedRateExchange.sol#266)" InFixedRateExchange (contracts/pools/fixedRate/FixedRateExchange.sol#21-814)
Redundant expression "oceanFeeAmount (contracts/pools/fixedRate/FixedRateExchange.sol#308)" InFixedRateExchange (contracts/pools/fixedRate/FixedRateExchange.sol#21-814)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

createWithDecimals(address[],uint256[]]) should be declared external:
  - FixedRateExchange.createWithDecimals(address[],uint256[]) (contracts/pools/fixedRate/FixedRateExchange.sol#166-225)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#multiline-function-that-could-be-declared-external

```

```

ERC20Template..initialize(string[],address[],address[],uint256[],bytes[]) (contracts/templates/ERC20Template.sol#173-223) uses assembly
  - INLINE ASM (contracts/templates/ERC20Template.sol#207-209)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-use

ERC20Template.createDispenser(address,uint256,uint256,pool,address) (contracts/templates/ERC20Template.sol#303-316) compares to a boolean constant:
  - withMint == true (contracts/templates/ERC20Template.sol#813)
ERC20Template.mint(address,uint256) (contracts/templates/ERC20Template.sol#325-335) compares to a boolean constant:
  - require(bool,string)(permissions[msg.sender].minter == true,ERC20Template: NOT MINTER) (contracts/templates/ERC20Template.sol#326-329)
ERC20Template.setFeeCollector(address) (contracts/templates/ERC20Template.sol#527-533) compares to a boolean constant:
  - require(bool,string)(permissions[msg.sender].feeManager == true,ERC20Template: NOT FEE MANAGER) (contracts/templates/ERC20Template.sol#528-531)
ERC20Template.onlyERC20Deployer() (contracts/templates/ERC20Template.sol#101-109) compares to a boolean constant:
  - require(bool,string)(IERC721Template_erc721Address.getPermissions(msg.sender).deployERC20 == true,ERC20Template: NOT DEPLOYER ROLE) (contracts/templates/ERC20Template.sol#102-107)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

ERC20Template.getAddressLength(address[]) (contracts/templates/ERC20Template.sol#678-676) is never used and should be removed
ERC20Template.getByteLength(bytes32[]) (contracts/templates/ERC20Template.sol#700-706) is never used and should be removed
ERC20Template.getUIntLength(uint256[]) (contracts/templates/ERC20Template.sol#685-691) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version>=0.6.0 (contracts/templates/ERC20Template.sol#1) allows old versions
Solv<0.6.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

isMinter(address) should be declared external:
  - ERC20Template.isMinter(address) (contracts/templates/ERC20Template.sol#342-344)
Dangerous call(s):
  - address(IERC4626Collector()).transfer(address(this),balance) (contracts/templates/ERC20Template.sol#737)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations

ERC20Template.setPublishingMarketFee(address,address,uint256) (contracts/templates/ERC20Template.sol#554-561) should emit an event for:
  - _publishMarketFeeAddress = _publishMarketFeeAddress (contracts/templates/ERC20Template.sol#558)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control

ERC20Template.setPublishingMarketFee(address,address,uint256) (contracts/templates/ERC20Template.sol#554-561) should emit an event for:
  - _publishMarketFeeAmount = _publishMarketFeeAmount (contracts/templates/ERC20Template.sol#560)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic

ERC20Template.setFeeCollector(address) (contracts/templates/ERC20Template.sol#527) lacks a zero-check on:
  - feeCollector = _newFeeCollector (contracts/templates/ERC20Template.sol#532)
ERC20Template.setPublishingMarketFee(address,address,uint256), publishMarketFeeAddress (contracts/templates/ERC20Template.sol#555) lacks a zero-check on:
  - _publishMarketFeeAddress = _publishMarketFeeAddress (contracts/templates/ERC20Template.sol#558)
ERC20Template.setPublishingMarketFee(address,address,uint256), _publishMarketFeeToken (contracts/templates/ERC20Template.sol#556) lacks a zero-check on:
  - _publishMarketFeeToken = _publishMarketFeeToken (contracts/templates/ERC20Template.sol#559)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in ERC20Template.deployPool(uint256[],uint256[],address[])
External calls:
  - pool = IFactoryRouter(router).deployPool(tokens,ssParams,swapFees,addresses) (contracts/templates/ERC20Template.sol#260-265)
  - onlyERC20Deployer() (contracts/templates/ERC20Template.sol#253)
    - require(bool,string)(IERC721Template_erc721Address.getPermissions(msg.sender).deployERC20 == true,ERC20Template: NOT DEPLOYER ROLE) (contracts/templates/ERC20Template.sol#102-107)
    Event emitted after the call(s):
      - NewPool(pool,addresses[0],addresses[1]) (contracts/templates/ERC20Template.sol#267)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

ERC20Template.permit(address,address,uint256,uint256,uint8,bytes32,bytes32) (contracts/templates/ERC20Template.sol#629-661) uses timestamp for comparisons
Dangerous comparisons:
  - require(bool,string)(deadline >= block.timestamp,ERC20PT: EXPIRED) (contracts/templates/ERC20Template.sol#638)

ERC721Template.setMetadata(uint8_,string,string,bytes,bytes,bytes) (contracts/templates/ERC721Template.sol#156-180) compares to a boolean constant:
  - require(bool,string)(permissions[msg.sender].updateMetadata == true,ERC721Template: NOT METADATA_ROLE) (contracts/templates/ERC721Template.sol#159-162)
ERC721Template.setURI(string,bytes) (contracts/templates/ERC721Template.sol#156-180) compares to a boolean constant:
  - hashURI == false (contracts/templates/ERC721Template.sol#156-180)
ERC721Template.createERC20(uint256,string[],address[],uint256[],bytes[]) (contracts/templates/ERC721Template.sol#215-239) compares to a boolean constant:
  - require(bool,string)(permissions[msg.sender].deployERC20 == true,ERC721Template: NOT ERC20DEPLOYER_ROLE) (contracts/templates/ERC721Template.sol#222-225)
ERC721Template.setNewData(bytes32,bytes) (contracts/templates/ERC721Template.sol#335-341) compares to a boolean constant:
  - require(bool,string)(permissions[msg.sender].store == true,ERC721Template: NOT STORE_UPDATER) (contracts/templates/ERC721Template.sol#336-339)
ERC721Template.setDataERC20(bytes32,bytes) (contracts/templates/ERC721Template.sol#352-358) compares to a boolean constant:
  - require(bool,string)(deployedERC20[msg.sender] == true,ERC721Template: NOT ERC20 Contract) (contracts/templates/ERC721Template.sol#353-356)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

Pragma version>=0.6.0 (contracts/templates/ERC721Template.sol#1) allows old versions
Solv<0.8. is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

isERC20Deployer(address) should be declared external:
  - ERC721Template.isERC20Deployer(address) (contracts/templates/ERC721Template.sol#245-247)
isInitialized() should be declared external:
  - ERC721Template.isInitialized() (contracts/templates/ERC721Template.sol#273-275)
setDataERC20(bytes32,bytes) should be declared external:
  - ERC721Template.setDataERC20(bytes32,bytes) (contracts/templates/ERC721Template.sol#352-358)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external

```

According to the test results, some findings found by these tools were considered false positives, while some of these findings were real security concerns. All relevant findings were reviewed by the auditors and addressed in the report as security concerns.

4.2 AUTOMATED SECURITY SCAN

Description:

Halborn used automated security scanners to assist with detection of well-known security issues, and to identify low-hanging fruits on the target contracts. Among the tools used was MythX, a security analysis service for Ethereum smart contracts. MythX performed a scan on the testers machine and sent the compiled results to the analyzers to locate any vulnerabilities. Only security-related findings are shown below.

Results:

ERC20Template.sol

Report for contracts/templates/ERC20Template.sol
<https://dashboard.mythx.io/#/console/analyses/a137c08a-4243-41de-81d3-9175b197b61e>

Line	SWC Title	Severity	Short Description
1	(SWC-103) Floating Pragma	Low	A floating pragma is set.
428	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randomness.

ERC721Template.sol

Report for contracts/templates/ERC721Template.sol
<https://dashboard.mythx.io/#/console/analyses/91f9850a-64be-47a2-be0b-1f153a68a094>

Line	SWC Title	Severity	Short Description
1	(SWC-103) Floating Pragma	Low	A floating pragma is set.
171	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randomness.
179	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randomness.

ERC721Factory.sol

Report for contracts/ERC721Factory.sol
<https://dashboard.mythx.io/#/console/analyses/b6d7a374-548e-4344-9087-7f0bb49d0aeb>

Line	SWC Title	Severity	Short Description
1	(SWC-103) Floating Pragma	Low	A floating pragma is set.
148	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
190	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
309	(SWC-110) Assert Violation	Unknown	Out of bounds array access
327	(SWC-110) Assert Violation	Unknown	Out of bounds array access
328	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "+=" discovered
342	(SWC-110) Assert Violation	Unknown	Out of bounds array access

OPFCommunityFeeCollector.sol

Report for contracts/communityFee/OPFCommunityFeeCollector.sol

<https://dashboard.mythx.io/#/console/analyses/f5f9d124-7e28-4694-97a3-4932797508c4>

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

FactoryRouter.sol

Report for pools/FactoryRouter.sol

<https://dashboard.mythx.io/#/console/analyses/15a467b1-ea97-4d7b-bb1c-66ccb06d7de2>

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

Dispenser.sol

Report for contracts/pools/Dispenser.sol

<https://dashboard.mythx.io/#/console/analyses/54cd37b6-83e4-44d9-9869-370a1141c5fa>

Line	SWC Title	Severity	Short Description
1	(SWC-103) Floating Pragma	Low	A floating pragma is set.
22	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.
23	(SWC-110) Assert Violation	Unknown	Public state variable with array type causing reacheable exception by default.
220	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "--" discovered

FixedRateExchange.sol

Report for contracts/pools/fixedRate/FixedRateExchange.sol

<https://dashboard.mythx.io/#/console/analyses/76e10cdb-4312-49a2-b1c1-c9fde30b4a25>

Line	SWC Title	Severity	Short Description
1	(SWC-103) Floating Pragma	Low	A floating pragma is set.
23	(SWC-101) Integer Overflow and Underflow	Unknown	Arithmetic operation "<<" discovered
173	(SWC-110) Assert Violation	Unknown	Out of bounds array access

BFactory.sol

Report for pools/balancer/BFactory.sol

<https://dashboard.mythx.io/#/console/analyses/a5fc8e43-6536-461a-be68-9e52a91151f0>

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

BPool.sol

Report for pools/balancer/BPool.sol

<https://dashboard.mythx.io/#/console/analyses/31a96126-7a8e-45c3-96ef-a978b4af6a42>

Line	SWC Title	Severity	Short Description
2	(SWC-103) Floating Pragma	Low	A floating pragma is set.
94	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.

All relevant valid findings were identified in the manual code review.

THANK YOU FOR CHOOSING
HALBORN