SHERLOCK

SHERLOCK SECURITY REVIEW FOR

Prepared For:	Flux Protocol
Prepared By:	Sherlock
Security Researchers:	WatchPug, Pauliax, hickupHH3
Dates Reviewed:	May 19th - May 31st, 2022
Report Delivered:	June 6th, 2022

Introduction

“Flux is the trustless data layer for web3. Flux is a cross-chain oracle that provides smart contracts with access to economically secure data feeds on anything.”

Scope

Branch: p2p-audit (https://github.com/fluxprotocol/fpo-evm/tree/p2p-audit)
Commit: 3438a823ba330a0b344cbd5078fec050cabb7b93 (https://github.com/fluxprotocol/fpo-evm/tree/3438a823ba330a0b344cbd5078fec050cabb7b93/)
Contracts:

- ExamplePriceFeedConsumer.sol
- FluxP2PFactory.sol
- FluxPriceFeed.sol

Summary

The FluxP2P contracts allow the easy and simple deployment of first-party price feeds on EVM chains. The codebase size isn’t too large and complex as it consists of 2 contracts, with the third (ExamplePriceFeedConsumer) being an example of how to utilise the price feed. The core contract is the EIP-2362 compatible FluxP2PFactory contract that handles the deployment and indexing of individual price feeds which are compatible with Chainlink's V2 and V3 aggregator interface.

For this audit, we looked at functional correctness and considered potential issues with project integrations, as well as protection against adversarial agents. Notable issues found included concerns over data freshness and possible manipulation of answers submitted by validators. Some deviations from functional correctness were reported, as well as a number of gas optimizations.

Note: For this audit, the fix review was included as part of a follow-up audit, so there are no Sherlock verifications on the fixes in this audit report.

Test Suite

Test coverage: Good
Quality of tests: The quality of tests were high, with adequate coverage on the FluxP2PFactory contract. Given the relatively small size of the contracts, it is recommended to add more tests to achieve 100% coverage, at least for the factory and price feed contracts.

SHERLOCK

Findings

Each issue has an assigned severity:

- Informational issues are subjective in nature. They are typically suggestions around best practices or readability. Code maintainers should use their own judgement as to whether to address such issues.
- Low issues are objective in nature but are not security vulnerabilities. These should be addressed unless there is a clear reason not to.
- Medium issues are security vulnerabilities that may not be directly exploitable or may require certain conditions in order to be exploited. All major issues should be addressed.
- High issues are directly exploitable security vulnerabilities that need to be fixed.

Total Issues

Informational	Low	Medium	High
11	3	3	2

Issue H-01

Timestamp of the answers should be signed by validators and checked on-chain

Summary
Answers submitted by validators could be stale, but are still aggregated and considered to be a fresh price dated at `block.timestamp`.

Severity
High

Vulnerability Detail
There could be a significant time delay from the time signers submitted their answers to the time that the transmit transaction is executed (Eg. function caller withholds the transaction, network congestion etc.). There is no guarantee on the freshness of data as a result.

Impact
Stale prices will affect protocol integrations significantly, especially money markets, as can be seen in the recent incident with LUNA.

Code Snippet
https://github.com/fluxprotocol/fpo-evm/blob/3438a823ba330a0b344cbd5078fec050cabb7b93/contracts/FluxP2PFactory.sol#L92-L150

https://github.com/fluxprotocol/fpo-evm/blob/3438a823ba330a0b344cbd5078fec050cabb7b93/contracts/FluxPriceFeed.sol#L73-L79

Tool used
Manual Review

Recommendation
Include the timestamp as part of the data to be signed by validators, and check against block.timestamp.

```
function transmit(
    bytes[] calldata _signatures,
    string calldata _pricePair,
    uint8 _decimals,
    uint32 _roundId,
    int192[] calldata _answers,
    uint64[] calldata _timestamps,
) external {
    uint256 len = _answers.length;
    require(_signatures.length == len && _timestamps.length == len, "Lengths mismatch");
    ...
    uint256 lastRoundTimestamp = priceFeed.latestTimestamp();
    for (uint256 i = 0; i < len; i++) {
        bytes32 hashedMsg = ECDSA.toEthSignedMessageHash(
            keccak256(abi.encodePacked(_pricePair, _decimals, _roundId, _answers[i], _timestamps[
        );
        (address recoveredSigner, ECDSA.RecoverError error) = ECDSA.tryRecover(hashedMsg, _signat
        ...

        // require transmitted answers to be sorted in ascending order
        if (i < len - 1) {
            require(_answers[i] <= _answers[i + 1], "Not sorted");
        }

        require(_timestamps[i] > lastRoundTimestamp, "Bad timestamp");

        ...
    }

    // calculate median of _answers
    int192 answer;
    uint256 timestamp;
    if (len % 2 == 0) {
        answer = ((_answers[(len / 2) - 1] + _answers[len / 2]) / 2);
        timestamp = ((_timestamps[(len / 2) - 1] + _timestamps[len / 2]) / 2);
    } else {
        answer = _answers[len / 2];
        timestamp = _timestamps[len / 2];
    }

    // try transmitting values to the oracle
    /* solhint-disable-next-line no-empty-blocks */
    try priceFeed.transmit(answer, uint256(timestamp)) {
        // transmission is successful, nothing to do
    } catch Error(string memory reason) {
        // catch failing revert() and require()
        emit Log(reason);
    }
}
```

SHERLOCK

```
function transmit(int192 _answer, uint256 _timestamp) external onlyRole(VALIDATOR_ROLE) {
    // Check the timestamp
    if (block.timestamp > _timestamp) {
        // reject stale price
        require(block.timestamp - _timestamp < validPeriod, "Bad timestamp");
    } else {
        // reject future timestamp (< 3s is allowed)
        require(_timestamp - block.timestamp < 3, "Bad timestamp");
        _timestamp = block.timestamp;
    }
    // Check the report contents, and record the result
    latestAggregatorRoundId++;
    transmissions[latestAggregatorRoundId] = Transmission(_answer, uint64(_timestamp));

    emit NewTransmission(latestAggregatorRoundId, _answer, msg.sender);
}
```

Flux Protocol Comment

Implemented the recommended change in follow-up audit.

Issue H-02

Possible front-running and manipulation the final median of answers

Summary

The answers and signatures can be filtered and reduced by a malicious actor down to the minimum number of signers required.

Severity

High

Vulnerability Detail

For a successful transmit, it is required that at least `minSigners` provide their answers. Should there be more signers that signed for this round, a malicious actor can watch the mempool and filter the answers, and re-submit only `minSigners` number of signatures.

This problem is further exacerbated by the fact that signature timestamps are not included, where the possibility of answers deviating from each other is increased. Another factor to consider is the difference between the number of validators and the minimum number of signers.

Proof of Concept

- Price feed has 6 signers and `minSigners` is 3
- All 6 signers submit their answers for the round, with values: [1, 5, 8, 10, 10, 10]. The median is 9.
- A malicious actor picks only minimal required signatures (3) and frontruns the tx with values: [1, 5, 8]. In this case, the median is 5.

Impact

There is the possibility of a large deviation between the expected and actual median price.

Code Snippet

https://github.com/fluxprotocol/fpo-evm/blob/3438a823ba330a0b344cbd5078fec050cabb7b93/contracts/FluxP2PFactory.sol#L92-L150

Tool used
Manual review

Recommendation
Consider making transmit() a permissioned call to only validators. This comes with the assumption that there is an existence of a penalty mechanism to penalize malicious validators.

In addition, consider ensuring that the difference between `minSigners` and number of validators for a price feed is kept to a hardcoded limit.

Flux Protocol Comment
Implemented the recommended change by reverting if msg.sender is not a signer when transmitting answers.

Issue M-01

Centralization risk of Feed contracts' VALIDATOR_ROLE

Summary

Any address that is granted the `VALIDATOR_ROLE` will bypass the aggregation and validator group signature process.

Severity

Medium

Vulnerability Detail

`DEFAULT_ADMIN_ROLE` can be granted to any address by the admin of the FluxP2PFactory, and `VALIDATOR_ROLE` can be granted by addresses with the `DEFAULT_ADMIN_ROLE`.
Any address that is given the `VALIDATOR_ROLE` can submit any answer to the price feed, thus bypassing the aggregation and validator group signature process.

Impact

The price feed can be fed malicious answers.

Code Snippet

https://github.com/fluxprotocol/fpo-evm/blob/3438a823ba330a0b344cbd5078fec050cabb7b93/contracts/FluxPriceFeed.sol#L73-L79

Tool used

Manual review

Recommendation

Consider simplifying the contract to make the `transmit()` function callable by the P2PFactory only.

Flux Protocol Comment

Mitigated by removing the admin role from the FluxP2PFactory contract.

SHERLOCK

Issue M-02

Duplicate signature should be checked in the first round as well

Summary

The duplicate signature check for the first round is skipped. This is partially caused by a mismatch in `_roundId` and `latestAggregatorRoundId`.

Severity

Medium

Vulnerability Detail

`_roundId` has to be 0 for the first round, to match `latestAggregatorRoundId = 1`. However, the duplicate check only occurs when `_roundId` is greater than zero. Thus, the check is skipped for the first round.

Impact

Duplicate signatures and answers can be submitted for the first round.

Code Snippet

https://github.com/fluxprotocol/fpo-evm/blob/3438a823ba330a0b344cbd5078fec050cabb7b93/contracts/FluxP2PFactory.sol#L128-L130

Tool used

Manual review

Recommendation

Sync the round ids. Then, the duplicate check signature becomes

```
require(priceFeed.latestRound() + 1 == _roundId, "Wrong roundId");dld
```

This also means that the minimum value of _roundId is 1, resulting in the if (_roundId > 0) condition being redundant.

Flux Protocol Comment

Implemented the recommended change.

SHERLOCK

Issue M-03

Validators will be unable to validate current round after priceFeed.transmit() fails

Summary

Validators used for the current round cannot submit for the same round should the transmission fail.

Severity

Medium

Vulnerability Detail

Should transmission fail, the transaction will not revert, but an error log is emitted. This means that the validators' last signed round will still be updated to the latest round, but price feed's latestAggregatorRoundId remains unchanged.

Existing validators who participated in the failed update will then be unable to participate in the current round because of the condition `require(fluxPriceFeeds[id].lastSignedRound[recoveredSigner] < _roundId, "Duplicate signature");`

Proof of Concept

The current implementation of `priceFeed.transmit()` will revert under the following circumstances:

- Caller does not have `VALIDATOR_ROLE`
- `latestAggregatorRoundId` increment overflows
- Insufficient gas remaining

Of the 3 cases, the first would be the most probable, as demonstrated in the test script below.

```
it.only("will have providers be unable to sign current round after
failed transmission", async function () {
  // step 0: setup
```

SHERLOCK

```
  const pricePair = this.eth_usd_str;
  const decimals = 3;
  let answers = [3000, 4000];

  // deploy oracle
  await this.proxy
    .connect(this.signers.admin)
    .deployOracle(this.eth_usd_str, decimals, [this.provider1.address,
this.provider2.address]);

  const PriceFeedContract = await
ethers.getContractFactory("FluxPriceFeed");
  const eth_usd_addr = await
this.proxy.connect(this.signers.admin).addressOfPricePair(this.eth_usd_i
d);
  const pricefeed = PriceFeedContract.attach(eth_usd_addr);
  const VALIDATOR_ROLE =
keccak256(ethers.utils.toUtf8Bytes("VALIDATOR_ROLE"));

  // transfer price feed owner to admin
  await
this.proxy.connect(this.signers.admin).transferOwner(this.eth_usd_id,
this.signers.admin.address);
  // increment price feed round by successfully transmitting an answer
  // so that duplicate signature will be checked
  await pricefeed.connect(this.signers.admin).grantRole(VALIDATOR_ROLE,
this.signers.admin.address);
  await pricefeed.connect(this.signers.admin).transmit(1);

  let round = await this.proxy.latestRoundOfPricePair(this.eth_usd_id);

  // sign answer 0 by provider1 and answer 1 by provider2
  let p1_msgHash = ethers.utils.solidityKeccak256(
    ["string", "uint8", "uint32", "int192"],
    [pricePair, decimals, round, answers[0]],
  );
  let p2_msgHash = ethers.utils.solidityKeccak256(
    ["string", "uint8", "uint32", "int192"],
    [pricePair, decimals, round, answers[1]],
  );
  let p1_sig = await this.provider1.signMessage(arrayify(p1_msgHash));
  let p2_sig = await this.provider2.signMessage(arrayify(p2_msgHash));
```

```
  let sigs = [p1_sig, p2_sig];

  // step 1: revoke proxy's VALIDATOR_ROLE from price feed
  await pricefeed.connect(this.signers.admin).revokeRole(VALIDATOR_ROLE,
this.proxy.address);

  // step 2: attempt transmission, will not revert but price feed would
not have updated
  await this.proxy.connect(this.signers.admin).transmit(sigs, pricePair,
decimals, round, answers);
  // same round
  expect(await
this.proxy.latestRoundOfPricePair(this.eth_usd_id)).to.be.eq(round);

  // step 3: re-grant proxy VALIDATOR_ROLE, re-attempt transmission
  // will revert
  await pricefeed.connect(this.signers.admin).grantRole(VALIDATOR_ROLE,
this.proxy.address);
  await expect(
    this.proxy.connect(this.signers.admin).transmit(sigs, pricePair,
decimals, round, answers),
  ).to.be.revertedWith("Duplicate signature");
});
```

SHERLOCK

Impact
A new, mutually exclusive, set of validators is required to sign the current round.

Code Snippet

https://github.com/fluxprotocol/fpo-evm/blob/3438a823ba330a0b344cbd5078fec050cabb7b93/contracts/FluxP2PFactory.sol#L104

https://github.com/fluxprotocol/fpo-evm/blob/3438a823ba330a0b344cbd5078fec050cabb7b93/contracts/FluxPriceFeed.sol#L102-L104

https://github.com/fluxprotocol/fpo-evm/blob/3438a823ba330a0b344cbd5078fec050cabb7b93/contracts/FluxPriceFeed.sol#L75

Tool used
Manual review

Recommendation
We suggest making `priceFeed.transmit()` callable by the factory only. Then, the try-catch block can be removed.

Replaying failed transactions wouldn't be a concern if data freshness is checked, as suggested in the recommendation for H-01.

Flux Protocol Comment
Mitigated by reverting in the failure condition.

Issue L-01

Inconsistent and unconventional `_roundId`

Summary
A transmission for the Nth _roundId is stored in the N+1th latestAggregatorRoundId in the price feed.

Severity
Low

Vulnerability Detail
A check is performed to ensure that the `_roundId` passed for new transmission is equal to the price feed's `latestRound()`. However, in `priceFeed.transmit(answer)`, the `latestAggregatorRoundId` is first incremented by 1, then used as the key to store the `_answer`. This is unconventional and counterintuitive.

Impact
Confusion around the correct round id to be used.

Code Snippet
https://github.com/fluxprotocol/fpo-evm/blob/3438a823ba330a0b344cbd5078fec050cabb7b93/contracts/FluxP2PFactory.sol#L104

https://github.com/fluxprotocol/fpo-evm/blob/3438a823ba330a0b344cbd5078fec050cabb7b93/contracts/FluxPriceFeed.sol#L102-L104

https://github.com/fluxprotocol/fpo-evm/blob/3438a823ba330a0b344cbd5078fec050cabb7b93/contracts/FluxPriceFeed.sol#L75

Tool used
Manual review

Recommendation
Identical to M-02.

Flux Protocol Comment
Implemented the recommended change.

Issue L-02

Misnomer for the `transferOwner` function

Summary
Summary text

Severity
Low

Vulnerability Detail
The function name gives the impression that ownership of the specified price feed will be transferred to `_newAdmin`. While its behaviour grants `_newAdmin` the `DEFAULT_ADMIN_ROLE`, it does not revoke the current admin (P2PFactory) of its role.

Impact
Confusion about the intended behaviour of the function.

Code Snippet
https://github.com/fluxprotocol/fpo-evm/blob/3438a823ba330a0b344cbd5078fec050cabb7b93/contracts/FluxP2PFactory.sol#L220-L223

Tool used
Manual review

Recommendation
If this is intended behaviour, then we suggest renaming the function to be `addOwner()`. Otherwise, revoke the contract's `DEFAULT_ADMIN_ROLE` as well. To reduce the likelihood of an error, consider making the transfer a 2-step process.

Flux Protocol Comment
Mitigated by removing the admin role from the FluxP2PFactory contract.

Issue L-03

Non-compliance with EIP2362 for initiated IDs with no value yet

Summary

A status code of 404 should be returned if the value for an id is not available yet. However, the current implementation will return a status code of 200 (success) and answer of zero.

Severity

Low

Vulnerability Detail

As per the EIP-2362 specification, "`valueFor` MUST return a status code of 404 if the value for an id is not available yet."

Impact

Zero value is assumed to be valid success status is returned.

Code Snippet

https://github.com/fluxprotocol/fpo-evm/blob/3438a823ba330a0b344cbd5078fec050cabb7b93/contracts/FluxP2PFactory.sol#L169-L174

Tool used

Manual review

Recommendation

Change the try-catch statement to

```
try priceFeed.latestRoundData() returns (uint80 roundId, int256 answer,
uint256, uint256 updatedAt, uint80) {
        if (roundId) > 0 {
          return (answer, updatedAt, 200);
        }
} catch {}
return (0, 0, 404);
```

Flux Protocol Comment

Implemented the recommended change.

Issue I-01

Add constructor initializer in FluxP2PFactory

Summary
Automatic initialization of the implementation contract.

Severity
Low

Vulnerability Detail
As per OpenZeppelin's (OZ) recommendation, "The guidelines are now to make it impossible for *anyone* to run `initialize` on an implementation contract, by adding an empty constructor with the `initializer` modifier. So the implementation contract gets initialized automatically upon deployment."

Impact
The implementation contract instance has to be manually initialized.

Tool used
Manual review

Recommendation
```
/// @custom:oz-upgrades-unsafe-allow constructor
/* solhint-disable-next-line no-empty-blocks */
constructor() {
     _disableInitializers();
 }
```

Flux Protocol Comment
Obsoleted by removing OpenZeppelin Upgrades from FluxP2PFactory contract.

Issue I-02

Redundant events in CLInterface.sol

Summary
CLInterface events are never emitted.

Severity
Informational

Vulnerability Detail
The `AnswerUpdated` and `NewRound` events are never emitted.

Impact
Redundancy.

Code Snippet
https://github.com/fluxprotocol/fpo-evm/blob/3438a823ba330a0b344cbd5078fec050cabb7b93/contracts/interface/CLInterface.sol#L15-L16

Tool used
Manual review

Recommendation
The events can be removed from the interface.

Flux Protocol Comment
Implemented the recommended change.

Issue I-03

Consider having minSigners as an input parameter

Summary
Deploying an oracle that has a different `minSigners` has to be done in 2 steps currently. Having it as an input parameter reduces this to 1.

Severity
Informational

Vulnerability Detail
An oracle that is to be deployed with `minSigners != 2` will first require the deployment to be executed, then have its value updated via `setMinSigners()`.

Impact
Higher gas costs, additional overhead on devops.

Code Snippet
https://github.com/fluxprotocol/fpo-evm/blob/3438a823ba330a0b344cbd5078fec050cabb7b93/contracts/FluxP2PFactory.sol#L61-L84

Tool used
Manual review

Recommendation
Take in `minSigners` as an input parameter.

Flux Protocol Comment
Obsoleted by internally calculating minSigners as signers.length/2 + 1.

Issue I-04

ExamplePriceFeedConsumer.sol: Consider using Ownable rather than AccessControl

Summary
Only the owner role is required; it would therefore be simpler to use `Ownable`.

Severity
Informational

Vulnerability Detail
`AccessControl` is not necessary here, `Ownable` is simpler and can increase the signal and noise ratio.

Impact
Simplicity and readability.

Code Snippet
https://github.com/fluxprotocol/fpo-evm/blob/3438a823ba330a0b344cbd5078fec050cabb7b93/contracts/ExamplePriceFeedConsumer.sol#L9-L10

Tool used
Manual review

Recommendation
Change to

```
contract ExamplePriceFeedConsumer is Ownable {
    CLV2V3Interface public priceFeed;
    ...
    constructor(address _priceFeed) {
        priceFeed = CLV2V3Interface(_priceFeed);
    }
    ...
    function setPriceFeed(address _priceFeed) public onlyOwner {
        priceFeed = CLV2V3Interface(_priceFeed);
    }
}
```

Flux Protocol Comment
Implemented the recommended change.

SHERLOCK

Issue I-05

ExamplePriceFeedConsumer.sol: Consider using latestRoundData() instead of latestAnswer()

Summary
Using `latestRoundData()` is preferred over `latestAnswer()`.

Severity
Informational

Vulnerability Detail
While `latestAnswer()` is simpler, `latestRoundData()` provides more data for validity and freshness checks, which should be preferred in most cases.

Impact
Data validation.

Code Snippet
https://github.com/fluxprotocol/fpo-evm/blob/3438a823ba330a0b344cbd5078fec050cabb7b93/contracts/ExamplePriceFeedConsumer.sol#L21-L23

Tool used
Manual review

Recommendation
Change to

```
// not preferred, the result can be invalid/stale
function getLatestPrice() public view returns (int256) {
    return priceFeed.latestAnswer();
}
// preferred, invalid/stale price is checked
function getPrice() public view returns (int256) {
    (uint80 roundID, int256 feedPrice, , uint256 timestamp, uint80
answeredInRound) = priceFeed.latestRoundData();
    require(feedPrice > 0, "price <= 0");
    require(answeredInRound >= roundID, "stale price");
    require(timestamp != 0, "round not complete");
    return feedPrice;
}
```

Flux Protocol Comment
Implemented the recommended change.

SHERLOCK

Issue I-06

latestAggregatorRoundId can be uint256 instead of uint32

Summary
There isn't a good reason why `latestAggregatorRoundId` should be of type `uint32` instead of `uint256`.

Severity
Informational

Vulnerability Detail
From what we see, `latestAggregatorRoundId` being of type `uint32` isn't required for compatibility reasons, neither is it efficiently packed, which therefore makes it less gas efficient because of type casting to it.

Impact
Higher gas cost.

Code Snippet
https://github.com/fluxprotocol/fpo-evm/blob/3438a823ba330a0b344cbd5078fec050cabb7b93/contracts/FluxPriceFeed.sol#L14

Tool used
Manual review

Recommendation
Switch `latestAggregatorRoundId` to type `uint256`

Flux Protocol Comment
Implemented the recommended change.

Issue I-07

Use _grantRole instead of _setupRole

Summary
`_setupRole()` is deprecated in favor of `_grantRole()`.

Severity
Informational

Vulnerability Detail
It is more gas-efficient to call `_grantRole()` directly because `_setupRole()` will call `_grantRole()`.
https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/access/AccessControl.sol#L203-L207

Impact
Higher gas cost.

Code Snippet

https://github.com/fluxprotocol/fpo-evm/blob/3438a823ba330a0b344cbd5078fec050cabb7b93/contracts/FluxP2PFactory.sol#L44

https://github.com/fluxprotocol/fpo-evm/blob/3438a823ba330a0b344cbd5078fec050cabb7b93/contracts/FluxPriceFeed.sol#L35-L36

https://github.com/fluxprotocol/fpo-evm/blob/3438a823ba330a0b344cbd5078fec050cabb7b93/contracts/ExamplePriceFeedConsumer.sol#L16

Tool used
Manual review

Recommendation
Change all instances of `_setupRole()` to `_grantRole()`.

Flux Protocol Comment
Implemented the recommended change.

Issue I-08

Unnecessary casting in addressOfPricePair()

Summary
Return value is unnecessarily casted from and to the `address` type.

Severity
Informational

Vulnerability Detail
`fluxPriceFeeds[_id].priceFeed` is of `address` type, but is casted to `FluxPriceFeed` and back to `address`.

Impact
Higher gas cost.

Code Snippet
https://github.com/fluxprotocol/fpo-evm/blob/3438a823ba330a0b344cbd5078fec050cabb7b93/contracts/FluxP2PFactory.sol#L185-L188

Tool used
Manual review

Recommendation
```
function addressOfPricePair(bytes32 _id) external view returns (address) {
  return fluxPriceFeeds[_id].priceFeed);
}
```

Flux Protocol Comment
Implemented the recommended change.

Issue I-09

Use Prefix Increment instead of Postfix Increment

Summary
Where possible, it would be advisable to use the prefix increment instead of the postfix increment.

Severity
Informational

Vulnerability Detail
The difference between the prefix increment and postfix increment expression lies in the return value of the expression. The prefix increment expression (++i) returns the **updated** value after it's incremented. The postfix increment expression (i++) returns the **original** value.

The prefix increment expression is cheaper in terms of gas.
https://github.com/byterocket/c4-common-issues/blob/main/0-Gas-Optimizations.md/#g012---use-prefix-increment-instead-of-postfix-increment-if-possible

Impact
Higher gas cost.

Code Snippet

https://github.com/fluxprotocol/fpo-evm/blob/3438a823ba330a0b344cbd5078fec050cabb7b93/contracts/FluxP2PFactory.sol#L78

https://github.com/fluxprotocol/fpo-evm/blob/3438a823ba330a0b344cbd5078fec050cabb7b93/contracts/FluxP2PFactory.sol#L111

https://github.com/fluxprotocol/fpo-evm/blob/3438a823ba330a0b344cbd5078fec050cabb7b93/contracts/FluxPriceFeed.sol#L75-L76

Tool used
Manual review

Recommendation
Replace variable++ with ++variable. For instance:

```
transmissions[++latestAggregatorRoundId] = Transmission(_answer,
uint64(block.timestamp));
```

Flux Protocol Comment
Implemented the recommended change.

SHERLOCK

Issue I-10

Turn on optimizer & increase no. of solc runs

Summary
The optimizer isn't switched on, and no. of solc runs used can be increased further.

Severity
Informational

Vulnerability Detail
The optimizer isn't turned on. We highly recommend doing so because the FluxP2PFactory, as it currently stands, is at 24.524 KB, which is very close to the bytecode size limit.
Turning on the optimizer and increasing the number of solc runs will help to both reduce the bytecode size by about 1/3, and at the same time, make function calls cheaper. More information about solc runs can be found in the solidity docs.

Impact
Higher gas cost.

Code Snippet
https://github.com/fluxprotocol/fpo-evm/blob/3438a823ba330a0b344cbd5078fec050cabb7b93/hardhat.config.ts#L263

Tool used
Manual review

Recommendation
```
optimizer: {
enabled: true,
runs: 1000000,
}
```

Flux Protocol Comment
Implemented the recommended change.

Issue I-11

Spelling Error

Summary
Compatibility is spelt wrong.

Severity
Informational

Impact
Readability.

Code Snippet
https://github.com/fluxprotocol/fpo-evm/blob/3438a823ba330a0b344cbd5078fec050cabb7b93/contracts/FluxPriceFeed.sol#L203

Tool used
Manual review

Recommendation
`compatability` → `compatibility`

Flux Protocol Comment
Implemented the recommended change.