SHERLOCK

SHERLOCK SECURITY REVIEW FOR

Prepared For: Flux Protocol

Prepared By: Sherlock

Security Researchers: Pauliax, hickupHH3

Dates Reviewed: June 10th - June 16th, 2022

Report Delivered: June 18th, 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-fixes-admin (https://github.com/fluxprotocol/fpo-evm/tree/p2p-audit-fixes-admin)
Commit: a0063173a4607dc8da517cde8cf62de6b287202d (https://github.com/fluxprotocol/fpo-evm/tree/a0063173a4607dc8da517cde8cf62de6b287202d)
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.

This audit is a review of a previous audit performed. A number of changes have been made since then, such as the removal of admin permissions from FluxP2PFactory that makes oracle deployment and signer modification more permissionless, and fixes for issues raised previously.

We looked at functional correctness for the changes made. While a couple of minor issues and gas optimizations have been found, no notable flaws were uncovered.

Note: Sherlock has not viewed the fixes for this audit, as Flux has decided not to bring this product to market yet.

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.

Note that the first test fails from the new changes, where a revert happens due to 'Future timestamp'

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
10	3	0	0

Issue L-01

Same formula of roundID generation prevents multiple signer modifications in a round

Summary
Dependency of `modifySigners()` on the price feed's `roundID` prevents multiple signer modifications in between rounds.

Severity
Low

Vulnerability Detail
The same formula of `roundID` generation is used for both `transmit()` and `modifySigners()`, which is reliant on the price feed's latestRound.

```
uint256 round = FluxPriceFeed(fluxPriceFeeds[_id].priceFeed).latestRound() + 1;
```

There is therefore an indirect dependency of `modifySigners()` on `transmit()`, where only a single modification can be made (ie. only do addition or removal of a signer, but not chained or multiple modifications) until a new price feed transmission is made.

While this seems to be intended, where the message to sign "comes with an acknowledgment that a new round must occur to modify signers again", there are potentially scenarios where multiple signers need to be changed in between rounds, such as replacing a signer (adding signer A, removing signer B).

Hence, to accommodate this use case, we recommend using a separate counter.

Impact
Only 1 modification can be made after each price feed transmission.

Code Snippet
https://github.com/fluxprotocol/fpo-evm/blob/a0063173a4607dc8da517cde8cf62de6b287202d/contracts/FluxP2PFactory.sol#L205-L206

Tool used
Manual Review

Recommendation
Have a separate counter for signer modifications.

Flux Protocol Comment

Implemented the recommended change.

Sherlock Comment
TBD

Issue L-02

Incorrect revert reason

Summary
Minimum number of signers is 2, but the revert string gives the impression that it is 3.

Severity
Low

Vulnerability Detail
The minimum signers allowed is 2, so prior to the removal, the requirement would be a minimum of 3 signers. While the check is correct, the revert reason isn't. It should be changed from `>2` to `>1`.

Impact
Confusion over minimum number of signers required

Code Snippet
https://github.com/fluxprotocol/fpo-evm/blob/a0063173a4607dc8da517cde8cf62de6b287202d/contracts/FluxP2PFactory.sol#L221-L222

Tool used
Manual review

Recommendation
```
require(fluxPriceFeeds[_id].signers.length() > 2, "Need >1 signers");
```

Flux Protocol Comment
Implemented the recommended change.

Sherlock Comment
TBD

Issue L-03

Modifying signers list does not check return values

Summary
When adding or removing signers, it does not check if the operation was successful.

Severity
Low

Vulnerability Detail
EnumerableSet operations return a boolean value indicating if it succeeded or not, e.g. when adding an element:
** Returns true if the value was added to the set, that is if it was not already present.*

While in most cases this does not cause any serious issues, it makes it possible to deploy an oracle with duplicate signers bypassing min signers check. If you invoke deployOracle with 2 duplicate signers, it will add only 1 signer but set minSigners to 2 leaving it impossible to perform any action on this price feed later.

Impact
This is not a big problem, because the hash of the price feed is derived from the address of msg.sender, thus it does not have a serious impact.

Code Snippet
https://github.com/fluxprotocol/fpo-evm/blob/a0063173a4607dc8da517cde8cf62de6b287202d/contracts/FluxP2PFactory.sol#L94-L109

https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/utils/structs/EnumerableSet.sol#L229

https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/utils/structs/EnumerableSet.sol#L239

Tool used
Manual review

Recommendation
Require that add/remove operations succeed.

Flux Protocol Comment
Implemented the recommended change.

Sherlock Comment

TBD

SHERLOCK

Issue I-01

Same hash produced regardless of `msg.sender` casing

Summary
Unnecessary requirement of `_creator` to be in lowercase

Severity
Informational

Vulnerability Detail
Regardless of whether the address has been checksummed, or in lower case, the hash result will be the same. Hence, the description of the `_creator` parameter can simply be `msg.sender` of `deployOracle()`.

Impact
Redundancy

Code Snippet
https://github.com/fluxprotocol/fpo-evm/blob/a0063173a4607dc8da517cde8cf62de6b287202d/contracts/FluxP2PFactory.sol#L46

Tool used
Manual review

Recommendation
```
/// @param _creator msg.sender of `deployOracle()`
```

Flux Protocol Comment
Implemented the recommended change.

Sherlock Comment
TBD

Issue I-02

Redundant address casting of `msg.sender`

Summary
Redundant address casting

Severity
Informational

Vulnerability Detail
`msg.sender` is already of `address` type; the explicit casting has no effect and is therefore unnecessary.

Impact
Redundancy.

Code Snippet
https://github.com/fluxprotocol/fpo-evm/blob/a0063173a4607dc8da517cde8cf62de6b287202d/contracts/FluxP2PFactory.sol#L97

Tool used
Manual review

Recommendation
```
bytes32 id = hashFeedId(_pricePair, _decimals, msg.sender);
```

Flux Protocol Comment
Implemented the recommended change.

Sherlock Comment
TBD

Issue I-03

Declare and use `timestamp` earlier to cache `_timestamps[i]`

Summary
Cache `_timestamps[i]` in the `timestamp` variable as a gas optimization.

Severity
Informational

Vulnerability Detail
A gas optimization would be to declare and save `_timestamps[i]` into the `timestamp` variable.

Impact
Higher gas costs

Code Snippet
https://github.com/fluxprotocol/fpo-evm/blob/a0063173a4607dc8da517cde8cf62de6b287202d/contracts/FluxP2PFactory.sol#L151-L152

Tool used
Manual review

Recommendation
```
uint64 timestamp;
for (uint256 i = 0; i < len; ++i) {
  ...
  timestamp = _timestamps[i];
  require(timestamp > lastRoundTimestamp, "Stale timestamp");
  require(timestamp < block.timestamp + 5, "Future timestamp");
  ...
}
```

Flux Protocol Comment
Implemented the recommended change.

Sherlock Comment
TBD

Issue I-04

Declare and save `len / 2` in variable

Summary
Repeated calculation of `len / 2` is gas inefficient.

Severity
Informational

Vulnerability Detail
`len / 2` is calculated numerous times. It would be more gas efficient to save the result in a local variable.

Impact
Higher gas costs.

Code Snippet
https://github.com/fluxprotocol/fpo-evm/blob/a0063173a4607dc8da517cde8cf62de6b287202d/contracts/FluxP2PFactory.sol#L167-L173

Tool used
Manual review

Recommendation
```
// a further gas optimization would be to utilise bitshift
// because the DIV opcode uses 5 gas while the SHR opcode only uses 3
gas.
// medianIndex = len >> 1;
uint256 medianIndex = len / 2;
if (len % 2 == 0) {
  answer = ((_answers[medianIndex - 1] + _answers[medianIndex]) / 2);
  timestamp = ((_timestamps[medianIndex - 1] + _timestamps[medianIndex)
/ 2);
} else {
  answer = _answers[medianIndex];
  timestamp = _timestamps[medianIndex];
}
```

```
```

Flux Protocol Comment
Implemented the recommended change.

Sherlock Comment
TBD

Issue I-05

Spelling Error

Summary
Initial is spelt wrong.

Severity
Informational

Impact
Readability.

Code Snippet
https://github.com/fluxprotocol/fpo-evm/blob/a0063173a4607dc8da517cde8cf62de6b287202d/contracts/FluxP2PFactory.sol#L30

Tool used
Manual review

Recommendation
`intitial` → `initial`

Flux Protocol Comment
Implemented the recommended change.

Sherlock Comment
TBD

SHERLOCK

Issue I-06

Emit events with relevant data

Summary
After updating the contracts, do not forget to also update the events. For example, event NewTransmission could emit a new parameter of _timestamp.

Severity
Informational

Impact
Consistency.

Code Snippet
https://github.com/fluxprotocol/fpo-evm/blob/a0063173a4607dc8da517cde8cf62de6b287202d/contracts/FluxPriceFeed.sol#L78

Tool used
Manual review

Recommendation
Revisit events and make sure that they contain the relevant data.

Flux Protocol Comment
Implemented the recommended change.

Sherlock Comment
TBD

Issue I-07

Inconsistent logging of errors

Summary
Based on description the event Log should log all errors:

```
/// @notice logs error messages
/// @param message the error message
event Log(string message);
```

However, it does not log when _verifySignature fails.

Severity
Informational

Impact
Consistency.

Code Snippet
https://github.com/fluxprotocol/fpo-evm/blob/a0063173a4607dc8da517cde8cf62de6b287202d/contracts/FluxP2PFactory.sol#L41

https://github.com/fluxprotocol/fpo-evm/blob/a0063173a4607dc8da517cde8cf62de6b287202d/contracts/FluxP2PFactory.sol#L81

Tool used
Manual review

Recommendation
Consider either emitting this event on other failures or renaming it to something like LogTransmitError.

Flux Protocol Comment
Implemented the recommended change.

Sherlock Comment
TBD

Issue I-08

Cancel signatures

Summary
Signers cannot cancel their signatures if they change their mind.

Severity
Informational

Impact
Signers need to be extra careful when signing messages.

Tool used
Manual review

Recommendation
Consider adding a function where each signer can cancel their signatures on chain, like updating lastRoundModifySigners/lastRoundTransmit or incrementing a nonce, or something like that.

Flux Protocol Comment
Implemented the recommended change.

Sherlock Comment
TBD

SHERLOCK

Issue I-09

Signature replayability

Summary
toEthSignedMessageHash does not include the domain separator meaning the signatures can be replayed on different chains.

Severity
Informational

Impact
When identical price feeds are deployed across different chains, it will be possible to replay signature actions. However, this might be the intended behavior, sign once, sync everywhere.

Code Snippet
https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/utils/cryptography/ECDSA.sol#L202-L206

Tool used
Manual review

Recommendation
If the FluxP2PFactory is intended to be deployed on several chains, consider including a domain separator when calculating the signature hash, e.g. toTypedDataHash includes domainSeparator.

Flux Protocol Comment
Implemented the recommended change.

Sherlock Comment
TBD

Issue I-10

Repeated storage access should be cached

Summary
Cache storage variables as a gas optimization.

Severity
Informational

Vulnerability Detail
There are many inefficiencies when accessing the same storage slot several times, for example, FluxPriceFeed(fluxPriceFeeds[_id].priceFeed) is accessed twice here:
    uint256 round = FluxPriceFeed(fluxPriceFeeds[_id].priceFeed).latestRound() + 1;
    uint256 lastRoundTimestamp =
FluxPriceFeed(fluxPriceFeeds[_id].priceFeed).latestTimestamp();

Impact
Higher gas costs

Code Snippet
https://github.com/fluxprotocol/fpo-evm/blob/a0063173a4607dc8da517cde8cf62de6b287202d/contracts/FluxP2PFactory.sol#L137-L138

Tool used
Manual review

Recommendation
```
FluxPriceFeed fluxPriceFeed =
FluxPriceFeed(fluxPriceFeeds[_id].priceFeed);

uint256 round = fluxPriceFeed.latestRound() + 1;
uint256 lastRoundTimestamp = fluxPriceFeed.latestTimestamp();
```

Flux Protocol Comment
Implemented the recommended change.

Sherlock Comment
TBD