SHERLOCK

# SHERLOCK SECURITY REVIEW FOR

| | |
|---|---|
| Prepared for: | Iron Bank |
| Prepared by: | Sherlock |
| Lead Security Expert: | 0x52 |
| Dates Audited: | May 29 - June 11, 2023 |
| Prepared on: | July 21, 2023 |

## Introduction

Iron Bank is a decentralized lending platform focused on capital efficiency allowing protocols and individuals to supply and borrow cryptoassets.

### Scope

Repository: ibdotxyz/ib-v2

Branch: eth

Commit: 66c70f3f58a1dc1e07908b4dae2f55c30e3b7edd

---

For the detailed scope, see the contest details.

### Findings

Each issue has an assigned severity:

- 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.

### Issues found

| Medium | High |
|---|---|
| 5 | 1 |

### Issues not fixed or acknowledged

| Medium | High |
|---|---|
| 0 | 0 |

### Security experts who found valid issues

| | | |
|---|---|---|
| sl1 | rvierdiiev | tsvetanovv |
| harisnabeel | kutugu | BugBusters |
| p0wd3r | n1punp | BenRai |
| IceBear | thekmj | 0xHati |

| | | |
|---|---|---|
| SovaSlava | 0x8chars | Arabadzhiev |
| MohammedRizwan | Norah | Pheonix |
| Kodyvim | shaka | devScrooge |
| HexHackers | vagrant | Hama |
| deadrxsezzz | Bozho | bitsurfer |
| berlin-101 | Jaraxxus | bin2chen |
| lil.eth | Proxy | santipu_ |
| Ruhum | qbs | Aymen0909 |
| turvec | Madalad | saidam017 |
| 3agle | Diana | martin |
| Brenzee | Schpiel | kn0t |
| shealtielanz | Ignite | ni8mare |
| Breeje | 0x3b | holyhansss |
| branch_indigo | Ocean_Sky | evilakela |
| 0xStalin | Angry_Mustache_Man | Kose |
| sufi | 0x52 | gkrastenov |
| ast3ros | Arz | josephdara |
| toshii | CMierez | 0xMAKEOUTHILL |
| sashik_eth | plainshift-2 | jayphbee |
| R-Nemes | n33k | 0xpinky |
| Bauchibred | qpzm | simon135 |
| tnquanghuy0512 | Delvir0 | anthony |

## Issue H-1: supplyNativeToken will strand ETH in contract if called after ACTION_DEFER_LIQUIDITY_CHECK

Source: https://github.com/sherlock-audit/2023-05-ironbank-judging/issues/361

### Found by

0x52, CMierez, evilakela

### Summary

supplyNativeToken deposits msg.value to the WETH contract. This is very problematic if it is called after ACTION_DEFER_LIQUIDITY_CHECK. Since onDeferredLiqudityCheck creates a new context msg.value will be 0 and no ETH will actually be deposited for the user, causing funds to be stranded in the contract.

### Vulnerability Detail

TxBuilderExtension.sol#L252-L256

```
function supplyNativeToken(address user) internal nonReentrant {
    WethInterface(weth).deposit{value: msg.value}();
    IERC20(weth).safeIncreaseAllowance(address(ironBank), msg.value);
    ironBank.supply(address(this), user, weth, msg.value);
}
```

supplyNativeToken uses the context sensitive msg.value to determine how much ETH to send to convert to WETH. After ACTION_DEFER_LIQUIDITY_CHECK is called, it enters a new context in which msg.value is always 0. We can outline the execution path to see where this happens:

```
execute > executeInteral > deferLiquidityCheck >
ironBank.deferLiquidityCheck > onDeferredLiquidityCheck (new context) >
executeInternal > supplyNativeToken
```

When IronBank makes it's callback to TxBuilderExtension it creates a new context. Since the ETH is not sent along to this new context, msg.value will always be 0. Which will result in no ETH being deposited and the sent ether is left in the contract.

Although these funds can be recovered by the admin, it may can easily cause the user to be unfairly liquidated in the meantime since a (potentially significant) portion of their collateral hasn't been deposited. Additionally in conjunction with my other submission on ownable not being initialized correctly, the funds would be completely unrecoverable due to lack of owner.

## Impact

User funds are indefinitely (potentially permanently) stuck in the contract. Users may be unfairly liquidated due to their collateral not depositing.

## Code Snippet

TxBuilderExtension.sol#L252-L256

## Tool used

Manual Review

## Recommendation

msg.value should be cached at the beginning of the function to preserve it across contexts

## Discussion

**ibsunhub**

Confirm the issue!

However, we believe the correct modification is to pass `msg.value` through the whole external call and make `deferLiquidityCheck` function payable.

**0xffff11**

Valid high. I also think the fix from sponsor is most accurate.

**IAm0x52**

Passing through `msg.value` will result in it being nonfunctional in the event that `supplyNativeToken` is called before `ACTION_DEFER_LIQUIDITY_CHECK` since it will deposit msg.value into WETH. Then when it calls `deferLiquidityCheck` it would again attempt to send `msg.value` which would cause it to revert due to lack of funds.

My suggestion would be to cache msg.value as an internal storage variable (i.e. _msgValue) at the beginning of execute. Adjust supplyNativeToken to use that storage variable rather than `msg.value`. After the supply to ironBank reset this variable to 0. This allows you to preserve the msg.value across all contexts

**ibsunhub**

The situation mentioned is same with #192. The solution sounds viable and better. Will work on a fix according to the recommendation.

**0xffff11**

@ibsunhub You mean that #192 should be a dup of this one?

**ibsunhub**

No, just think they are related.

**iamjakethehuman**

Escalate for 10 USDC I believe this issue to be of Medium Severity. Here's why:

1. It is valid for one asset only (ETH)
2. It requires to call two specific operations and in a specific order in order for the issue to occur
3. The lost ETH can be rescued by the owner of the protocol (we do not take into consideration there is a vulnerability the eth can be stolen by adversary as the Watson has both not mentioned it, nor reported it separately)
4. The biggest loss that can occur is getting liquidated. This would be the case if the user has no more assets to still supply their account. And even if liquidation is to occur, in worst case scenario, they'd lose up to just 20% of their portfolio in IronBank (max liquidation bonus = 125%, `(125-100)/125 = 20%`. With all these external factors, considering in many cases there would be no loss of funds whatsoever and just in a tiny bit of them there would be a loss of 20%, I believe this to be of Medium severity.

**sherlock-admin**

> Escalate for 10 USDC I believe this issue to be of Medium Severity. Here's why:
>
> 1. It is valid for one asset only (ETH)
> 2. It requires to call two specific operations and in a specific order in order for the issue to occur
> 3. The lost ETH can be rescued by the owner of the protocol (we do not take into consideration there is a vulnerability the eth can be stolen by adversary as the Watson has both not mentioned it, nor reported it separately)
> 4. The biggest loss that can occur is getting liquidated. This would be the case if the user has no more assets to still supply their account. And even if liquidation is to occur, in worst case scenario, they'd lose up to just 20% of their portfolio in IronBank (max liquidation bonus = 125%, `(125-100)/125 = 20%`. With all these external factors, considering in many cases there would be no loss of funds whatsoever and just in a tiny bit of them there would be a loss of 20%, I believe this to be of Medium severity.

You've created a valid escalation for 10 USDC!

To remove the escalation from consideration: Delete your comment.

You may delete or edit your escalation comment anytime before the 48-hour escalation window closes. After that, the escalation becomes final.

**ibsunhub**

> The situation mentioned is same with #192. The solution sounds viable and better. Will work on a fix according to the recommendation.

Sorry, I mean #198 related to this issue, not #192 .

**C-Mierez**

My two cents on iamjakethehuman 's escalation.

> 2. It requires to call two specific operations and in a specific order in order for the issue to occur

I would argue that this set of actions is not "too" specific. Deferring liquidity is done to avoid wasting gas by executing `IronBank#_getAccountLiquidity()` multiple times, so you always want to call this at the beginning before performing all other actions (This behaviour/order can even be seen in TestTxBuilderExtension_integration.t.sol as well). Thus having `ACTION_DEFER_LIQUIDITY_CHECK` before any of the problematic `ACTION_X_NATIVE_TOKEN` actions is not far fetched at all, imo.

> 3. The lost ETH can be rescued by the owner of the protocol (we do not take into consideration there is a vulnerability the eth can be stolen by adversary as the Watson has both not mentioned it, nor reported it separately)

I explored this possibility in my own report (#368 ), and I don't think we should just ignore the fact that the ETH stuck in the contract can be stolen. If the ETH were safe then this would just be an inconvenience until the owner comes in, but both facts together create a vector in which the user ***can*** lose their funds without virtually any cost or risk on the malicious actor's side, all due to this implementation flaw on `TxBuilderExtender`

**ib-tycho**

https://github.com/ibdotxyz/ib-v2/pull/53

**0xffff11**

Thanks for both comments. Imo this issue should be high. I agree, that it is quite likely for it to happen.

''' Deferring liquidity is done to avoid wasting gas by executing IronBank#_getAccountLiquidity() multiple times, so you always want to call this at the beginning before performing all other actions (This behaviour/order can even be seen in TestTxBuilderExtension_integration.t.sol as well). ''' Sponsor has also confirmed. This being said, to fully tie together the eth reports, the issue https://github.com/sherlock-audit/2023-05-ironbank-judging/issues/198 should now be valid imo as a medium. Because eth can in fact get stuck in the contract. So

I would say, for this issue, keep it as a high and upgrade https://github.com/sherlock-audit/2023-05-ironbank-judging/issues/198 to medium

**ibsunhub**

fix: https://github.com/ibdotxyz/ib-v2/pull/53

**hrishibhat**

Result: High Has duplicates Affecting only one token can still be a valid high, the given order of operations is not an unlikely scenario. And since this break the assumption of these contracts should not hold eth + it can be stolen as shown in #368, Maintaining the severity as is.

**sherlock-admin**

Escalations have been resolved successfully!

Escalation status:

- iamjakethehuman: rejected

**IAm0x52**

Fix looks good. Msg.value is now cached allowing it to be preserved across contexts

## Issue M-1: PriceOracle.getPrice doesn't check for stale price

Source: https://github.com/sherlock-audit/2023-05-ironbank-judging/issues/9

### Found by

0x3b, 0x52, 0x8chars, 0xHati, 0xStalin, 0xpinky, 3agle, Angry_Mustache_Man, Arabadzhiev, Aymen0909, Bauchibred, BenRai, Bozho, Breeje, Brenzee, BugBusters, CMierez, Delvir0, Diana, Hama, HexHackers, IceBear, Ignite, Jaraxxus, Kodyvim, Kose, Madalad, MohammedRizwan, Norah, Ocean_Sky, Pheonix, Proxy, R-Nemes, Ruhum, Schpiel, SovaSlava, anthony, berlin-101, bin2chen, bitsurfer, branch_indigo, devScrooge, evilakela, gkrastenov, harisnabeel, holyhansss, jayphbee, josephdara, kn0t, kutugu, lil.eth, martin, n33k, ni8mare, plainshift-2, qbs, qpzm, rvierdiiev, saidam017, santipu_, sashik_eth, shaka, shealtielanz, simon135, sl1, thekmj, toshii, tsvetanovv, vagrant

### Summary

PriceOracle.getPrice doesn't check for stale price. As result protocol can make decisions based on not up to date prices, which can cause loses.

### Vulnerability Detail

`PriceOracle.getPrice` function is going to provide asset price using chain link price feeds. https://github.com/sherlock-audit/2023-05-ironbank/blob/main/ib-v2/src/protocol/oracle/PriceOracle.sol#L66-L72

This function doesn't check that prices are up to date. Because of that it's possible that price is not outdated which can cause financial loses for protocol.

### Impact

Protocol can face bad debt.

### Code Snippet

Provided above

### Tool used

Manual Review

## Recommendation

You need to check that price is not outdated by checking round timestamp.

## Discussion

**ibsunhub**

Same with the answer to #25. It's not practical to setup different heartbeat for individual markets. And we have a backend to monitor the price deviation.

**0xffff11**

Due to the off-chain system by Iron, issue can be a low. (Does not really affect the current state of the contracts) @ibsunhub Is it the right resolution, or thinking more about invalid?

**ib-tycho**

We still think this is invalid, thanks

**0xffff11**

Because Iron's off-chain safeguard, invalid

**bzpassersby**

Escalate for 10 USDC I think this is wrongly classified as invalid. (1) It's impossible for Watsons to know that the protocol has off-chain safeguards because the protocol explicitly said there are no off-chain mechanisms in the contest info. It's unfair for Watsons who might be misled by this answer.

```
Q: Are there any off-chain mechanisms or off-chain procedures for the protocol
↪  (keeper bots, input validation expectations, etc)?
nope
```

(2)It's right and should be encouraged for Watsons to point-out insufficient on-chain checks. The current code ignores any data freshness-related variables when consuming chainlink data, which is clearly not the best practice.

And it's understandable that the protocol chose to implement such checks off-chain. But since Watsons wouldn't have known about this and that the code itself clearly has flaws. This should be at least low/informational. It's unfair for Watsons to be punished because of this.

**sherlock-admin**

> Escalate for 10 USDC I think this is wrongly classified as invalid. (1) It's impossible for Watsons to know that the protocol has off-chain safeguards because the protocol explicitly said there are no off-chain

> mechanisms in the contest info. It's unfair for Watsons who might be misled by this answer.
>
> ```
> Q: Are there any off-chain mechanisms or off-chain procedures for the
> ↪  protocol (keeper bots, input validation expectations, etc)?
> nope
> ```
>
> (2)It's right and should be encouraged for Watsons to point-out insufficient on-chain checks. The current code ignores any data freshness-related variables when consuming chainlink data, which is clearly not the best practice.
>
> And it's understandable that the protocol chose to implement such checks off-chain. But since Watsons wouldn't have known about this and that the code itself clearly has flaws. This should be at least low/informational. It's unfair for Watsons to be punished because of this.

You've created a valid escalation for 10 USDC!

To remove the escalation from consideration: Delete your comment.

You may delete or edit your escalation comment anytime before the 48-hour escalation window closes. After that, the escalation becomes final.

**hrishibhat**

Result: Medium Has Duplicates Considering this a valid medium

**sherlock-admin**

Escalations have been resolved successfully!

Escalation status:

- bzpassersby: accepted

**Josephdara**

Hi @hrishibhat @sherlock-admin I believe my issue has been omitted https://github.com/sherlock-audit/2023-05-ironbank-judging/issues/471#issue-1751647942

**jacksanford1**

Issue was labeled "Won't Fix" by protocol team. Categorizing as an acknowledged issue.

## Issue M-2: PriceOracle will use the wrong price if the Chainlink registry returns price outside min/max range

Source: https://github.com/sherlock-audit/2023-05-ironbank-judging/issues/25

### Found by

0x52, 0x8chars, Angry_Mustache_Man, Bauchibred, BenRai, BugBusters, Jaraxxus, Madalad, R-Nemes, bitsurfer, branch_indigo, deadrxsezzz, shaka, thekmj, tsvetanovv

### Summary

Chainlink aggregators have a built in circuit breaker if the price of an asset goes outside of a predetermined price band. The result is that if an asset experiences a huge drop in value (i.e. LUNA crash) the price of the oracle will continue to return the minPrice instead of the actual price of the asset. This would allow user to continue borrowing with the asset but at the wrong price. This is exactly what happened to Venus on BSC when LUNA imploded.

### Vulnerability Detail

Note there is only a check for `price` to be non-negative, and not within an acceptable range.

```
function getPriceFromChainlink(address base, address quote) internal view
↪   returns (uint256) {
    (, int256 price,,,) = registry.latestRoundData(base, quote);
    require(price > 0, "invalid price");

    // Extend the decimals to 1e18.
    return uint256(price) * 10 ** (18 - uint256(registry.decimals(base, quote)));
}
```

https://github.com/sherlock-audit/2023-05-ironbank/blob/main/ib-v2/src/protocol/oracle/PriceOracle.sol#L66-L72

A similar issue is seen here.

### Impact

The wrong price may be returned in the event of a market crash. An adversary will then be able to borrow against the wrong price and incur bad debt to the protocol.

## Code Snippet

https://github.com/sherlock-audit/2023-05-ironbank/blob/main/ib-v2/src/protocol/oracle/PriceOracle.sol#L66-L72

## Tool used

Manual Review

## Recommendation

Implement the proper check for each asset. **It must revert in the case of bad price.**

```
function getPriceFromChainlink(address base, address quote) internal view
 ↪  returns (uint256) {
    (, int256 price,,,) = registry.latestRoundData(base, quote);
    require(price >= minPrice && price <= maxPrice, "invalid price"); // @audit
 ↪  use the proper minPrice and maxPrice for each asset

    // Extend the decimals to 1e18.
    return uint256(price) * 10 ** (18 - uint256(registry.decimals(base, quote)));
}
```

## Discussion

**ibsunhub**

It's not practical to setup the min price and max price for individual asset. It's hard to define a reasonable range for each asset and it will make oracle configuration more complex. It's much easier to make human error.

Also, we had an off-chain backend system to monitor the price from ChainLink. If the price is off, we would intervene to pause the oracle.

**0xffff11**

@ibsunhub If the oracle is paused, wouldn't functions that require of that oracle response also be paused during that time?

**ibsunhub**

Yes, functions that need to retrieve the price will revert. They are `borrow`, `redeem`, `transferIBToken`, and `liquidate`.

**0xffff11**

So, I see what the watson points out. I see that you have an off-chain safeguard for this. Therefore, I would mark the issue as invalid. Though I don't think the solution should be to revert. Liquidations can be key while oracle is paused. I think the fix should be the one from https://github.com/sherlock-audit/2023-05-ironbank-judging/issues/433 (secondary oracle and a try catch)

**0xffff11**

Invalid, Iron has an off-chain safeguard for price deviation that would prevent this

**iamjakethehuman**

Escalate for 10 USDC The off-chain safeguard is never mentioned. Watsons are not supposed to know it exists. Also, the supposed solution imposes an even larger risk as any user would be able to enter tbe market of which the oracle reverts and avoid liquidations. Issue should be marked as valid and another solution should be proposed.

**sherlock-admin**

> Escalate for 10 USDC The off-chain safeguard is never mentioned. Watsons are not supposed to know it exists. Also, the supposed solution imposes an even larger risk as any user would be able to enter tbe market of which the oracle reverts and avoid liquidations. Issue should be marked as valid and another solution should be proposed.

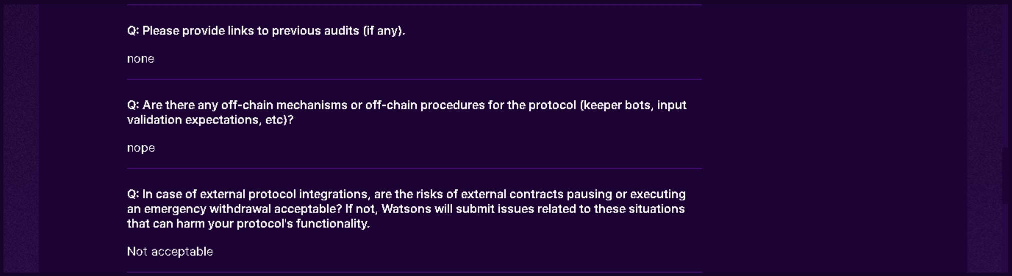
You've created a valid escalation for 10 USDC!

To remove the escalation from consideration: Delete your comment.

You may delete or edit your escalation comment anytime before the 48-hour escalation window closes. After that, the escalation becomes final.

**ADK0010**

Also the contest page doesn't talk about any off-chain safeguards.



Q: Please provide links to previous audits (if any).

none

Q: Are there any off-chain mechanisms or off-chain procedures for the protocol (keeper bots, input validation expectations, etc)?

nope

Q: In case of external protocol integrations, are the risks of external contracts pausing or executing an emergency withdrawal acceptable? If not, Watsons will submit issues related to these situations that can harm your protocol's functionality.

Not acceptable

**hrishibhat**

Result: Medium Has duplicates This is a valid medium

**sherlock-admin**

Escalations have been resolved successfully!

Escalation status:

- iamjakethehuman: accepted

**ib-tycho**

How do you establish a reasonable minimum and maximum price range for each asset? The incident related to Venus that you mentioned was caused by the inherent risk of the LUNA token itself. Evaluating the risk associated with an asset should always be taken into account when listing it. I disagree with relying solely on manual human input for setting the price range, as it does not address the underlying issue faced by Venus. Therefore, we will not make changes to address this matter.

**jacksanford1**

Issue was labeled "Won't Fix" by protocol team. Categorizing as an acknowledged issue.

## Issue M-3: Price Oracle contract does not work in Arbitrum and Optimism

Source: https://github.com/sherlock-audit/2023-05-ironbank-judging/issues/191

### Found by

ast3ros, n1punp, sashik_eth, sufi, tnquanghuy0512

### Summary

The PriceOracle contract relies on the Chainlink Oracle FeedRegistry to get the price of tokens. However, the FeedRegistry is not available in L2 networks such as Arbitrum and Optimism. This means that the PriceOracle contract will fail to return the price of tokens in these networks.

### Vulnerability Detail

The PriceOracle contract uses the Chainlink Oracle FeedRegistry to get the latest round data for a pair of tokens.

```
(, int256 price,,,) = registry.latestRoundData(base, quote);
```

https://github.com/sherlock-audit/2023-05-ironbank/blob/main/ib-v2/src/protocol/oracle/PriceOracle.sol#L67

The Iron Bank is deployed in mainnet, Arbitrum and Optimism. However, according to the Chainlink documentation, the FeedRegistry is only available in mainnet and not in Arbitrum and Optimism.

https://docs.chain.link/data-feeds/feed-registry#contract-addresses

This means that the PriceOracle contract will not be able to get the price of tokens in these L2 networks. This will affect the functionalities of the protocol that depend on the token price, such as liquidation.

https://github.com/sherlock-audit/2023-05-ironbank/blob/main/ib-v2/src/protocol/pool/IronBank.sol#L827-L828

### Impact

The PriceOracle contract will not function properly in L2 networks. This will break the protocol functions that rely on the token price.

## Code Snippet

https://github.com/sherlock-audit/2023-05-ironbank/blob/main/ib-v2/src/protocol/oracle/PriceOracle.sol#L67 https://github.com/sherlock-audit/2023-05-ironbank/blob/main/ib-v2/src/protocol/pool/IronBank.sol#L827-L828

## Tool used

Manual Review

## Recommendation

Reimplement the PriceOracle contract by reading the price feed from AggregatorV3Interface instead of FeedRegistry. Example: https://docs.chain.link/data-feeds/l2-sequencer-feeds#example-code

## Discussion

**thangtranth**

Escalate for 10 USDC

Please help me to review. This is not a duplication of #440.

- #440 is about checking If Arbitrum sequencer is down.
- This issue is about the current PriceOracle contract is wrongly implemented because the FeedRegistry is not existed in Arbitrum and Optimism.

They are two different issues with two different root causes.

**sherlock-admin**

> Escalate for 10 USDC
>
> Please help me to review. This is not a duplication of #440.
>
> - #440 is about checking If Arbitrum sequencer is down.
> - This issue is about the current PriceOracle contract is wrongly implemented because the FeedRegistry is not existed in Arbitrum and Optimism.
>
> They are two different issues with two different root causes.

You've created a valid escalation for 10 USDC!

To remove the escalation from consideration: Delete your comment.

You may delete or edit your escalation comment anytime before the 48-hour escalation window closes. After that, the escalation becomes final.

**ib-tycho**

Regarding the mistake in the contest details mentioned in the README, we apologize for any confusion caused. When we stated that we would deploy on Arbitrum and Optimism, we meant that we would make the necessary modifications before deployment. This is our standard practice of maintaining contracts with different branches, same as what we did in v1: https://github.com/ibdotxyz/compound-protocol/branches

We are aware of the absence of a registry on OP and Arb, as pointed out by some individuals. We would like to inquire if it is possible to offer the minimum reward for an oracle issue on L2. Thank you.

**0xffff11**

Even though the issue is valid, there will be no deployment on L2s as stated by sponsor. Unfortunately, it is invalid due to a miss-understanding from the docs. As watson said, it is not a duplicate of #440 so it should be de-duplicated and closed.

**thangtranth**

Hi everyone,

I totally understand the reasoning and I don’t mean to push this issue. However, I’m concerned that it will set a precedent when Watsons do not know what the scope is and where the source of truth is for each contest.

There are some duplication of this issue such as #18 and #239 which I believe those Watsons also pay attention to every line of codes and check Oracle registries in each L2 to ensure that the code is safely deployed in stated L2 in Readme.

In my opinion, if this issue is invalid, then all other issues related to L2 such as Sequencer should also be marked as invalid since the code is not used in L2.

**hrishibhat**

Result: Medium Has duplicates Although there was an error in the Readme, this would have been easily handled during deployment. However, this is still a valid issue from the contest POV.

**sherlock-admin2**

Escalations have been resolved successfully!

Escalation status:

- thangtranth: accepted

**jacksanford1**

Issue was labeled "Won't Fix" by protocol team. Categorizing as an acknowledged issue.

## Issue M-4: Wrong Price will be Returned When Asset is PToken for WstETH

Source: https://github.com/sherlock-audit/2023-05-ironbank-judging/issues/220

### Found by

branch_indigo

### Summary

Iron Bank allows a PToken market to be created for an underlying asset in addition to a lending market. PTokens can be counted as user collaterals and their price is fetched based on their underlying tokens. However, wrong price will return when PToken's underlying asset is WstETH.

### Vulnerability Detail

Retrieving price for WstETH is a 2 step process. WstETH needs to be converted to stETH first, then converted to ETH/USD. This is properly implemented when the market is WstETH through checking `if (asset==wsteth)`. But when PToken market is created for WstETH, this check will by bypassed because PToken contract address will be different from wsteth address.

PToken market price is set through `_setAggregators()` in PriceOracle.sol where base and quote token address are set and tested before writing into `aggregators` array. And note that quote token address can either be ETH or USD. When asset price is accessed through `getPrice()`, if the input asset is not `wsteth` address, `aggregators` is directly pulled to get chainlink price denominated in ETH or USD.

```
//PriceOracle.sol
//_setAggregators()
                require(
                    aggrs[i].quote == Denominations.ETH ||
                        aggrs[i].quote == Denominations.USD,
                    "unsupported quote"
                );
```

```
//PriceOracle.sol
    function getPrice(address asset) external view returns (uint256) {
        if (asset == wsteth) {
            uint256 stEthPrice = getPriceFromChainlink(
                steth,
                Denominations.USD
            );
            uint256 stEthPerToken = WstEthInterface(wsteth).stEthPerToken();
            uint256 wstEthPrice = (stEthPrice * stEthPerToken) / 1e18;
            return getNormalizedPrice(wstEthPrice, asset);
        }
        AggregatorInfo memory aggregatorInfo = aggregators[asset];
        uint256 price = getPriceFromChainlink(
            aggregatorInfo.base,
            aggregatorInfo.quote
        );
        ...
```

This creates a problem for PToken for WstETH, because `if (asset==wsteth)` will be bypassed and chainlink aggregator price will be returned. And chainlink doesn't have a direct price quote of WstETH/ETH or WstETH/USD, only WstETH/stETH or stETH/USD. This means most likely aggregator price for stETH/USD will be returned as price for WstETH.

Since stETH is a rebasing token, and WstETH:stETH is not 1 to 1, this will create a wrong valuation for users holding PToken for WstETH as collaterals.

## Impact

Since users holding PToken for WstETH will have wrong valuation, this potentially creates opportunities for malicious over-borrowing or unfair liquidations, putting the protocol at risk.

## Code Snippet

https://github.com/sherlock-audit/2023-05-ironbank/blob/main/ib-v2/src/protocol/oracle/PriceOracle.sol#L43

## Tool used

Manual Review

## Recommendation

In `getPrice()`, consider adding another check whether the asset is PToken and its underlying asset is WstETH. If true, use the same bypass for pricing.

## Discussion

**bzpassersby**

Escalate for 10 USDC I think this is wrongly excluded. The report describes a clear vulnerability that the current Oracle implementation doesn't take into account

PToken for WstETH. stETH is a rebasing token ,which is correctly factored in oracle implementation when the market asset is for WstETH. However, when a PToken is created for WstETH, `if(asset==wsteth)` will be bypassed. Since chainlink doesn't have a direct feed for WstETH/ETH or WstETH/USD. Wrong price will be returned.

**sherlock-admin**

> Escalate for 10 USDC I think this is wrongly excluded. The report describes a clear vulnerability that the current Oracle implementation doesn't take into account PToken for WstETH. stETH is a rebasing token ,which is correctly factored in oracle implementation when the market asset is for WstETH. However, when a PToken is created for WstETH, `if(asset==wsteth)` will be bypassed. Since chainlink doesn't have a direct feed for WstETH/ETH or WstETH/USD. Wrong price will be returned.

You've created a valid escalation for 10 USDC!

To remove the escalation from consideration: Delete your comment.

You may delete or edit your escalation comment anytime before the 48-hour escalation window closes. After that, the escalation becomes final.

**0xffff11**

Issue seems valid to me. I would judge it as a med because seems that it is for a only a specific pair, quite an edgecases. Thoughts @ibsunhub ?

**ibsunhub**

We're ok with med.

Although we don't decide that if we will support wstETH's PToken, it's our oversight to handle of the price of wstETH's PToken in the price oracle.

**ibsunhub**

fix: https://github.com/ibdotxyz/ib-v2/pull/57

**hrishibhat**

Result: Medium Unique Considering this issue a valid medium based on the above comments.

**sherlock-admin**

Escalations have been resolved successfully!

Escalation status:

- bzpassersby: accepted

**IAm0x52**

Fix looks good. PTokens now always use their underlying token when determining their price

## Issue M-5: getPriceFromChainlink() doesn't check If Arbitrum sequencer is down in Chainlink feeds

Source: https://github.com/sherlock-audit/2023-05-ironbank-judging/issues/440

### Found by

0x52, 0xMAKEOUTHILL, Angry_Mustache_Man, Arabadzhiev, Arz, Aymen0909, BenRai, Breeje, Brenzee, BugBusters, Delvir0, HexHackers, Ignite, Jaraxxus, Kodyvim, Madalad, MohammedRizwan, Ocean_Sky, Proxy, R-Nemes, SovaSlava, berlin-101, bin2chen, bitsurfer, branch_indigo, deadrxsezzz, devScrooge, josephdara, kutugu, n1punp, n33k, ni8mare, p0wd3r, plainshift-2, rvierdiiev, santipu_, sashik_eth, shaka, simon135, sl1, toshii, tsvetanovv, turvec, vagrant

### Summary

When utilizing Chainlink in L2 chains like Arbitrum, it's important to ensure that the prices provided are not falsely perceived as fresh, even when the sequencer is down. This vulnerability could potentially be exploited by malicious actors to gain an unfair advantage.

### Vulnerability Detail

There is no check: getPriceFromChainlink

```
    function getPriceFromChainlink(address base, address quote) internal view
 ↪  returns (uint256) {
@>      (, int256 price,,,) = registry.latestRoundData(base, quote);
        require(price > 0, "invalid price");

        // Extend the decimals to 1e18.
        return uint256(price) * 10 ** (18 - uint256(registry.decimals(base,
 ↪  quote)));
    }
```

### Impact

could potentially be exploited by malicious actors to gain an unfair advantage.

### Code Snippet

https://github.com/sherlock-audit/2023-05-ironbank/blob/main/ib-v2/src/protocol/oracle/PriceOracle.sol#L66-L72

## Tool used

Manual Review

## Recommendation

code example of Chainlink:
https://docs.chain.link/data-feeds/l2-sequencer-feeds#example-code

## Discussion

**0xffff11**

Valid medium

**ib-tycho**

Regarding the mistake in the contest details mentioned in the `README`, we apologize for any confusion caused. When we stated that we would deploy on Arbitrum and Optimism, we meant that we would make the necessary modifications before deployment. This is our standard practice of maintaining contracts with different branches, same as what we did in v1:
https://github.com/ibdotxyz/compound-protocol/branches

We are aware of the absence of a registry on OP and Arb, as pointed out by some individuals. We would like to inquire if it is possible to offer the minimum reward for an oracle issue on L2. Thank you.

**ib-tycho**

We'll fix this when deploying on L2, but we disagree with Severity. I would consider this as Low

**0xffff11**

According to past reports and sponsor confirmed that they will fix the issue. The issue will remain as a medium.

**MLON33**

Assuming this issue is acknowledged by the protocol team and won’t be fixed.