SHERLOCK

Security Review For Mach Finance

Public Best Efforts Contest Prepared For: **Mach Finance**
Lead Security Expert: **eeyore**
Date Audited: **December 17 - December 21, 2024**

Introduction

Mach Finance is an upcoming borrow / lending platform native to Sonic, built on top of Compound v2.

Scope

Repository: Mach-Finance/contracts

Branch: main

Audited Commit: 60e06b7bc8d59055cf399d5c8b09f77482550a85

Final Commit: 521f456cfab033f87742da94a20cff87bfc08f9b

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

High	Medium
0	1

Issues Not Fixed or Acknowledged

High	Medium
0	0

Security experts who found valid issues

0xpetern
0xRiO
miaowu
fuzzysquirrel
c3phas
bbl4de
d4y1ight
silver_eth
zxriptor
vahdrak1
0xc0ffEE
zanderbyte
eeyore

Issue M-1: Missing staleness check in PythOracle can lead to forced liquidations and theft of funds from borrowers.

Source: https://github.com/sherlock-audit/2024-12-mach-finance-judging/issues/41

Found by

0xRiO, 0xc0ffEE, 0xpetern, bbl4de, c3phas, d4y1ight, eeyore, fuzzysquirrel, miaowu, silver_eth, vahdrak1, zanderbyte, zxriptor

Summary

The Pyth integration contract uses the `getPriceUnsafe()` function to retrieve the latest reported price from a specified price feed.

However, Pyth does not have sponsored price feeds on the Sonic chain, that will guaranty the proper data freshness. As such, this function depends on users or the Mach protocol to update the price data with fresh values.

As a result, the prices retrieved using `getPriceUnsafe()` can be significantly outdated, even with the protocol best efforts to maintain price freshness due to its keeper/bot outages.

The documentation for `getPriceUnsafe()` incorrectly suggests that it will revert with a `StalePrice` error when the price is outdated. In reality, this function returns a price with its associated timestamp, regardless of how old the timestamp is, as long as the price was updated at some point in the past.

This issue becomes critical if the prices used by the protocol are not updated for extended periods, during which significant price changes could occur for assets being borrowed or used as collateral in the project.

Root Cause

The Pyth integration contract does not perform a staleness check, and did not switch to fallback oracle in such situation. It only verifies that the price is > 0 (here), which is insufficient.

Although Compound V2 does not enforce staleness checks in its Chainlink oracle integration, the situation differs because Chainlink prices cannot be updated directly by anyone. Even stale prices from Chainlink are deemed acceptable under these circumstances, especially to avoid blocking liquidations.

In contrast, Pyth allows anyone to update the price feed (via `updatePriceFeeds()` function), in situation where the new price adheres to the rule that it must be newer than the previous price and is a valid Pyth price component for given price feed.

This introduces a vulnerability where price updates can be manipulated to extract value from users.

Internal pre-conditions

None.

External pre-conditions

- Pyth prices have not been updated for an extended period.

Attack Path

An attacker can construct a transaction that updates the price feed with a desired value reported at a timestamp between the last update and the current time. This enables the attacker to inflate the price of borrowed assets and deflate the price of collateral assets, leading to forced liquidations.

The reported prices can originate from any valid price component with timestamp within the allowed timeframe. As observed in real-world scenarios, asset prices can rise by 10% in one day and fall by 20% on another day (e.g., FTM).

Consider a three-day timeframe during which Pyth prices for two supported assets are not updated:

1. **Day 1**: Asset A and Asset B prices are accurately reported.
2. **Day 2**: Both assets experience a 10% price increase, but the prices are not updated in Pyth. Although exploitation is possible at this stage, the attacker may wait another day.
3. **Day 3**: Both assets experience a 20% price drop.

The attacker can then update:

- The borrow asset price to the inflated value from Day 2 (+10%).
- The collateral asset price to the deflated value from Day 3 (-20%, or ~-10% from Day 1).

This manipulation allows the attacker to extract maximum liquidable value from users, profiting from the actual prices of the assets.

This attack would not be possible with fresh price updates, as the user shortfall would not fall below 1 in the given scenario of volatile prices.

Impact

- Direct loss of funds from forced liquidation of users, in situations where accurate price updates would prevent shortfalls from falling below 1.

Mitigation

- Enforce a staleness check when using `getPriceUnsafe()`. Utilize a fallback API3 oracle if freshness criteria are not met.
- Develop and deploy an off-chain bot (with a backup mechanism) to push fresh Pyth data to the Sonic network. The bot should use a short heartbeat and low price deviation triggers.

Discussion

sherlock-admin2

The protocol team fixed this issue in the following PRs/commits:
https://github.com/Mach-Finance/contracts/pull/8

Disclaimers

Sherlock does not provide guarantees nor warranties relating to the security of the project.

Usage of all smart contract software is at the respective users' sole risk and is the users' responsibility.