The first issue I noticed with my security layer was that I was not raising the FileClosedError when a write, read, or undo call was made on a closed file. This was a fairly simple fix. I just added a bool attribute to the LPFile class that was initialized as True when the class is created and is set to False when the file closes. Then, in each of the methods of the class it checks that the bool is set to True otherwise, it will raise an exception.

While fixing this issue I went through the repy V2 library reference again, and looked at every call that my security layer was replicating and ensured that I was raising the correct error when an issue occurred. Previously when someone made an incorrect call like having a negative offset my security layer did not raise an exception, just pass. I added a few raise exceptions so that now when there is an argument error or seek past end of file error I am raising the correct exceptions that repy V2 expects.

Additionally in class we discussed attacks that use threading and race conditions to break the reference monitor. While I was not fully aware of what this meant I took some time to look into threads and how they function in python and repy V2 and learned that when executing a method I would need to lock my object to stop other functions from accessing shared resources. I implemented this by adding an acquire() before any code I was to execute in each method and put the rest in a try block. I would add a finally at the end to release the lock to ensure that regardless of if the functions were to fail or not the release would always occur. I added this to my write, read, undo and close methods. Hopefully this would fix any race conditions like a check then act condition that may occur within my reference monitor and ensure that variables and arguments are not being changed halfway through a method call.