Name: Kaartikeya Panjwani
NetID: kp3291

Errors and vulnerabilities fixed

1. Record of File Size: In my original reference monitor, I was not keeping any records of the current size of the text file, and checking file size in functions, as it was being altered by the attacks. This was a major vulnerability in my reference monitor as it was not be able to accurately assess the impact of file operations on the system's security. This was leading to unauthorized access and even tampering with files that should be protected by the attacks. I fixed this vulnerability by including self.filesize = len(self.LPfile.readat(None,0)) and self.pending_file_size = self.filesize in innit and updating them in writeat() and undo() functions.

2. File Closed Check: In my original reference monitor, I was not checking in any of the functions whether the file was already closed or not. Performing operations on an already closed file can result in unpredictable and inconsistent behavior, which was allowing attacks to exploit my security layer. I fixed this vulnerability by including self.file_closed = False in innit and updating it/checking it in file functions of my security layer, enhancing data integrity and security.

3. Proper File Closure Handling: In my original reference monitor, I was allowing the text file to be closed even if there was any pending data. This was allowing the attacks to correctly assert that my file is not updated after the close operation. I fixed this vulnerability by including:

   ```
   if self.pending_data is not None and self.pending_offset is not None:
           self.LPfile.writeat(self.pending_data, self.pending_offset)
   ```
   in my close() function.

4. Pending Data Management: In my original reference monitor, some attacks were able to exploit its writeat() function by specifying self.pending_offset as None and writing to the file. The writeat method now properly handles pending data and offsets, ensuring that the correct data is written to the file and that the file size is updated accordingly.

5. Proper lock handling: In my original reference monitor, I was creating and releasing locks in every function. This was resulting in the error “un-locked locked released” by some attacks using multithreading to exploit the security layer. I fixed this error by creating a lock in the innit and only acquiring and releasing it in the functions instead of creating one in every function.