In my reference monitor, I had some vulnerabilities that were present that I had taken the time to fix. The vulnerabilities primarily involved attacks with threading, inability to handle valid read commands, inability to prevent invalid write ats and read ats, end of file errors, invalid file opening, and invalid updos and write ats..

A significant vulnerability that surfaced during the evaluation of the reference monitor was its susceptibility to threading attacks. Multiple threads were allowed to interact with the program simultaneously, leading to unexpected behavior and potential program crashes. This was an unintended outcome, as I didn't consider the effect of using multithreading on the reference monitor as a potential security concern that could lead to bypassing the monitor. The primary solution was to implement locks within the reference monitor to prevent concurrent write access by different threads. This ensured that only one thread could modify the program at a time, thus mitigating the threading vulnerabilities and preventing the errors of an attack program trying to write with multiple threads to the same file.

Another vulnerability was the improper handling of valid user commands. The reference monitor occasionally would not execute a valid read and instead return a negative one since I had a condition in the monitor that wouldn't allow the monitor to read if the reading index was less than what was currently in the offset. This was obviously incorrect as a user then couldn't read from zero when the offset was greater than it, thus preventing a valid read. This resulted in an inconsistent user experience and undermined the reference monitor's functionality.

The remedy was to correct the erroneous if condition and remove the code that hindered the execution of valid reads. This enhancement ensured that the reference monitor accurately allowed all reads to go through which was fine since if a write-at was pending it wouldnt reflect in the read as the read would only read what was in the file anyways and not anything else thus allowing all valid reads to go through in the user programs.

Another critical vulnerability was related to the potential for writing past the End-of-File (EOF) marker, a scenario that could have catastrophic consequences. The reference monitor did not adequately check for EOF errors, which posed a significant risk. To address this vulnerability, modifications were made to the reference monitor to implement a mechanism for tracking the EOF. By doing so, the reference monitor could prevent write operations that would extend beyond the EOF, effectively safeguarding against this serious vulnerability.

Finally, to resolve my other errors and concerns i added in try and except blocks to reflect in the event a user responded with erroneous commands to prevent them from being executed. the vulnerabilities in the reference monitor, including threading issues, prevention of valid user commands, and writing past the EOF marker, were identified and addressed, by implementing locking mechanisms for threading, correcting the if condition for valid user commands, and tracking the EOF respectively. These solutions efforts have significantly improved the robustness and security of the reference monitor. The lessons learned from these vulnerabilities underscore the importance of rigorous testing and the proactive identification of potential security risks in software systems.