Assignment 2 Part 3

My experience with this was as predicted by Prof Cappos in his research paper “Teaching the Security Mindset with Reference Monitors”. Not knowing anything about the language and Reference Monitors to now having a grasp about them is a very motivating thing.

As a result of this assignment, I was able to bring the number of vulnerabilities able to produce any output down to under 10. The major flaw present in my code initially was the usage of Locks, unintentionally I created a large flaw that caused my program to get stuck in negative offset conditions. I would like to roughly quote a reading included in our course which states that “security precautions must take into account the usability of application at hand”, in an attempt to protect against a highly unlikely threading attack I attempted (unsuccessfully) to protect against it.

A large majority of the vulnerabilities I had were “logical errors”, firstly I dealt with file size errors. I did this by taking into account that a file that was already created (earlier or otherwise) could be opened and written and undid on. For this I had to have a file size that I did not alter at a later point, and a file size that I could alter at a later point. This second file size I used to check for negative offsets, to know if there was going to be an overlap of existing text and newly written text, this also helped in the undo function where I used it to roll back from writes, something that I was inspired by in the announcement sent before assignment 2. What I mean by this is that I was simplistic in viewing the usage of undo, I trusted that the other aspects such as file size and EOF would be undone once I emptied the buffer and offset variables. Hence I created duplicate variables that I would alter during writes and the actual variables I would alter only on double writes or closes.

A second dummy error I made was to delete files, This I found was uncalled for as many attacks did this, however I believed this to be cleaning after the write operation. Removing this removed a large chunk of the nearly 250+ attacks going through my reference monitor. Another thing I should have done is that places where I was altering the default writeat function I should have seen the avalanche effect that would have happened to the exceptions, which is why I wrote a quite intricate if else ladder to deal with the throwing of exceptions. An error I was still not able to fix was that of the priority of throwing these exceptions when multiple exceptions were thrown.

Finally, one aspect I wanted to discuss as having fixed is that of the file size that I handled through a competitive coding problem I solved using the max function in python, to remain elegant by not using an if else condition(found on line 78). I also would like to report that the KISS principel could not be held more true for the submission I had initially and the transition it had to the final one, ie thinking as an attacker and then to someone defending it completely flipped my perspective to the mistakes that could exist.