Samuel Vieira Restrepo
Prof. Justin Cappos
Assignment 2 Part 3

For Assignment 2 Part 1 I created a very rudimentary reference monitor. I had a hard time understanding the instructions and very limited knowledge. My focus was on eliminating all the error messages that appeared with the template provided and passing the test cases provided by the professor, and it shows as more than 240 attacks bypassed the security layer –although many of them are invalid. In order to fix the vulnerabilities, I had to fortify many of the functions, add more attributes to the class in order to keep track of size and implement locks.

Because I was trying to put out the fires that came with the original program, there were many cases that I didn't consider in my program. One of the main problems was the *undo* function. I had an attribute that retain the previous state of the file. This was because I misunderstood the functionality of *writeat* and *undo*. I believed I had to keep track of the entire content of the file in order to revert to it in case *undo* was called. What I didn't understand was that a *writeat* didn't write the current data to the file, rather the pending data that was sent by the previous *writeat*. Also, my implementation of the previous state was a *readat* for 10000 bytes, so anything over that would not be saved into the previous state. This was a very poor implementation that had to be fixed by a complete overhaul of the functions. Finally, I did not take into account the fact that multiple threads might be used and did not implement any locks which I had to implement later.

The most important fixes that I made to the attributes of the class was getting rid of the *previous_state* attribute and adding two new attributes: *file_size* and *pending_file_size.* This allowed me to keep track of the offsets and ensure that that they would not write past the end of the current file. The other main differente was the implementation of locks. I nested all the functions in *try-finally* statements to ensure that all the computation inside these functions was performed before releasing the lock. These major improvements have proven to be very effective as very few attacks now pass and most of the ones that do pass are invalid. There were some cases where I would raise errors but this made it difficult to discern whether the error being raised was my implementation or that there was an actual error that the attack case was showing so I removed most of the errors that would be raised, mainly *RepyArgumentError.*