Report

There are several mistakes I've made inside my reference monitor.

The biggest flaw is that I forgot the implement the thread lock for my code. This is a point I completely ignored when I do the assignment. The vulnerability it brings is also pretty obvious: if multiple threads are created, there are probability that my program will crush. For example, in my implementation, I have two variables pending_data and pending_offset that is responsible for the undo feature, and they are shared by all functions in the class. In the undo function, what happens is that the pending_data and pending_offset will be set to None. And in writeat function, pending_data and pending_offset will be set to the arguments passed from the function. So if thread lock is not implemented, there would be an attack that utilize the multiple threads, and some threads are doing nothing but undo, some other threads are doing the writeat. Due to the nature of the thread, it might happen that the thread is interrupted in the middle of the execution and switched to another thread. Therefore, if the writeat thread is interrupted just at the position where pending_data is set but pending_offset is not set, then switched to one of the undo threads, and the undo threads cleared the value in pending_data and pending_offset, and then switched back to the writeat thread and let writeat thread finish the rest of its work. Then an inconsistent state will be left, that is, pending_data is set to None but pending_offset has the correct data. In this case, when the pending data is write to the file, the content will not be expected, and there will be errors raising as well since pending_data is None. Fixing this issue is pretty simple, just acquire the lock at the beginning of each function and release the lock and the end of each function.

The second flaw comes from a specification I did not noticed. So there is an specified order of exception when multiple exceptions exist. For example, if writeat at negative offset after file is closed, there would be two errors: FileClosedError and RepyArgumentError. But according to the specification, RepyArgumentError has higher priority than FileClosedError and should be raised first. In my implementation, I completely ignored this specification and let FileClosedError raised first. To fix this, just switch the codes that raise the FileClosedError and RepyArgumentError.