ISP Assignment 2 Part 3 Report - Avinash Narasimhan (an4098)

This 1-page write-up describes the vulnerabilities I identified in my initial reference monitor and the steps I took to fix them making the monitor more secure. The vulnerabilities and the respective patches introduced to fix them are described as follows:

1) In the writeat() function, I had written code to handle FileClosedError, RepyArgumentError and SeekPastEndOfFileError such that the monitor does not throw an error. This means any attack code calling the writeat() function with negative offset OR offsets greater than file size OR calling the function after the file is closed, the monitor would not throw any error and silently fail. So, I introduced if-condition clauses which would check if the offset is non-negative and if it is lesser than or equal to the file size and throw the respective error instead of silently failing. I also introduced an ***is_closed*** field to check if the file has been closed or not. The improved monitor also checks if the arguments are of the correct type. A locking mechanism has been introduced to ensure multiple threads do not execute concurrently. This was done by introducing a ***self.lock*** variable which would create a lock when the function starts and release when it finishes execution.

2) The readat() function had similar vulnerabilities as well. If an attack code passes a negative offset OR an offset greater than the file size to the function, OR if the function is called after the file is closed, the monitor would silently fail without throwing any error. Fixes similar to the ones done in the writeat() function were done for the readat() function as well. Similar to the writeat() function, a locking mechanism was introduced to ensure multiple threads do not execute code at the same time.

3) The close() function also had a vulnerability wherein if an attack called the close() function on an already closed file, the monitor silently failed. That was fixed in the improved monitor which now would raise a FileClosedError if the close() function is called on a closed file. Similar to the above functions, a locking mechanism was introduced to ensure multiple threads do not execute code at the same time.

4) The undo() function in my initial monitor did not check if the file has already been closed. The improved monitor checks that using the ***is_closed*** field. Additionally, the initial monitor did not adjust the file size when the undo() function was called, leading to an incorrect file size being stored. This was fixed in the improved monitor. Similar to the above functions, a locking mechanism was introduced to ensure multiple threads do not execute code at the same time.

5) In the initial monitor, when the init() function runs, the filesize is not being calculated. It was assumed that the file size was 0. This was being exploited with attacks where a file is opened, written to, closed and opened again. The initial write was not being considered. In the new monitor, the file size is being calculated in the init() function.

To summarise, the main vulnerabilities I identified and patched, involved proper handling of the basic RepyAPI errors such as the FileClosedError, RepyArgumentError and SeekPastEndOfFileError. The second major vulnerability which has been patched in the new monitor is a locking mechanism to prevent concurrent threads from running at the same time. Thirdly, the file size logic has been updated in the undo() and init() functions apart from the writeat() functions.