Dominique Eberhard de814@nyu.edu

Assignment 2.3 Writeup

[Note: At the time of writing (November 1, 2023 at 3:31 p.m.), my original reference monitor was broken by ~75 attack cases that were valid, and my revised reference monitor was not broken by any attack cases that were not deemed invalid or under review. I should also mention that when running the attack cases as a batch, there are two valid (at the time of writing) reference cases, "rsh9689_attackcase3.r2py" and "crt7919_attackcase2", that produce errors. However, when we delete all of the variations of "testfile.txt", like "test.txt", and run each of those two attack cases against my revised reference monitor individually, no output is produced.]

The Vulnerabilities of My Original Reference Monitor:

My original reference monitor was "broken" by about 10% of the valid attack cases. The majority of this 10% of attack cases that got through my original reference monitor exploited the fact that my reference monitor's writeat() method did not properly throw errors like the sandbox's standard writeat(). Specifically, my original reference monitor did not properly throw the RepyArgumentError, FileClosedError, or SeekPastEndOfFileError. Regarding the RepyArgumentError, my reference monitor was circumvented by attack cases that called writeat() with a negative offset as an argument. Concerning the FileClosedError, my original reference monitor was beaten by test cases where a file was closed but an error was not thrown if there was a subsequent valid writeat() call. Regarding the SeekPastEndOfFileError, my original reference monitor did not properly throw the error when an attack case tried writing data to an offset that was beyond the actual end of the file.

In addition, my original reference monitor failed to properly deal with certain attack cases that used threading to create race conditions. This is because I forgot to even consider using locks.

Thus, although my original reference monitor was very simple (I only added four lines of code to the original example reference monitor) and was able to keep out about 90% of the attack cases, it did not properly handle all of the edge cases like negative offsets and doing file operations on an already-closed file.

How I Fixed My Updated Reference Monitor:

To fix my original reference monitor, I first addressed my monitor's lack of accuracy in throwing the correct errors like the sandbox's standard writeat(). To properly handle cases of FileClosedError, I added an attribute to the LPFile() class called "is_closed" initially set to False upon the creation of an LPFile() object and set to True when the file is closed. This attribute is checked if an attacker tries to call undo() on a file that is closed. To properly handle instances of SeekPastEndOfFileError, I added two attributes to the LPFile() class called "filesize" and "pending_filesize", respectively, with "pending_filesize" initially set to the file size upon the instantiation of an LPFile object. Essentially this "pending_filesize" makes sure that the file size is what it is supposed to be when the "pending_data" is committed to the "pending_offset" after a call to close() or writeat(). Also, the "pending_file" attribute is updated to the stored "filesize" attribute upon a call to undo(). Finally, to deal with instances of RepyArgumentError, my updated reference monitor's writeat() has a condition that checks if the offset is negative or if the type of the data argument is not a string, in which case there should be a RepyArgumentError. The condition in my reference monitor's writeat() that checks is offset is negative or if the type of data is not a string also checks if the file is closed and if the file size is less than the offset argument. In any of those cases, the sandbox's standard writeat() is called with the inputted data and offset as arguments, which results in the standard errors that are thrown by the sandbox's writeat(), depending on the type of error.

To defend against threaded attacks, I introduced locks to the methods dealing with actually writing to the file, namely writeat() and close(). I also added a lock to the undo() method because it updates the "pending_filesize" attribute. However, my updated reference monitor was initially timing out against certain attack cases. Thus, to remedy this, I introduced try-finally blocks into the writeat(), close(), and undo() methods such that when either of these methods is called, a lock is acquired and then the method tries its operations, but if there is a risk of a timeout, the try block is not executed and the lock is released without there being any illegal writing to the file.

Thus, just a few additions made my reference monitor a lot more robust against the attack programs. Overall this was a very good learning experience and helped me develop my understanding of how to think about security and especially to never overlook any potential edge cases.