Reference Monitor Security Report

By
Ym3144@nyu.edu

Vulnerabilities found in the initial reference monitor in assignment 2.1:

- No FileClosedError thrown when tried to writeat after file close.
  - As the user should not be allowed to writeat after the file is closed, this is an issue.
  - this was caused because i assumed the self.LPfile.writeat function call of repy would take of this exception but since we are doing delayed writes, the latest write will not be committed with another write happening, so my monitor wouldn't throw the exception until another valid write happens or close is called.
- No RepyArgument Exception thrown when negative offset was used.
  - negative offset isn't considered valid by repy writeat, and should not be allowed.
- No SeekPastEndOfFileError Exception thrown when tried to writeat with offset > filesize
  - when the user is trying to read past the EOF, this is a security issue and one should not be allowed to do this.

Additional issues discovered:

- Improper release handling of locks which could cause to fail other subsequent function calls.

```
self.lock.acquire(True)
self.pending_data = None
self.pending_offset = None
self.lock.release()
```

Enhancements or fixies performed in assignment 2.3:

1) Added RepyArgument Exception handling
   - Dealing with negative offset
   - Dealing with data not being of string type
2) Added FileClosedError Exception handling
   - Dealing with case of writeat called after file closed
   - Used a closed Boolean flag for storing the current state of the file.
3) Added SeekPastEndOfFileError Exception handling
   - Dealing with case of writeat called with offset > filesize
   - Used fileSize variable to store the initial size of the file, and then keep it updated across the operations read, write and undo.
4) Added a Readat call in the LPfile constructor to get the current file size.
5) maintaining two fileSize variables for previous and commit lengths to perform undo effectively.
6) Updated the undo function such that it keeps the fileSize up to date and clears the pending data as before. Does all of this only if the file is open, if not does nothing. As it's not expected to raise any exceptions.
7) Updated how locks are being acquired and released: we are enclosing the critical section inside a try and except and releasing the lock in finally, this ensures that even if some exception is raised in the critical section, the lock will still be released, thus allowing new calls to acquire the lock.

```
            self.lock.acquire(True)
            try:
                self.tempFileSize = self.fileSize
                self.pending_data = None
                self.pending_offset = None
            finally:
                self.lock.release()
```