Aatish Balaji
ab10047@nyu.edu

Assignment 2.3 – Improvements to Reference Monitor

I implemented a series of changes to the original reference monitor code with the aim of rectifying multiple vulnerabilities and strengthening its resilience against potential attack scenarios. These adjustments were made to bolster the code's overall security, accuracy, and efficiency of the code as mentioned in the assignment.

One of the modifications involved the removal of constraints associated with `MAX_DATA_SIZE', a restriction that previously limited the reference monitor's capacity to handle large amounts of data. By eliminating these constraints, I allowed the `writeat` function to manage a more substantial volume of data, enhancing the code's accuracy and efficiency.

To significantly enhance the reference monitor's ability to function in a multi-threaded environment, I introduced locking mechanism provided by r2py. This mechanism utilized a locking instance that was initiated within the constructor. Its primary function was to safeguard critical sections of the code where file content manipulation and file closure activities were conducted. The integration of 'lock.acquire' and 'lock.release' was pivotal in ensuring the code's capability to manage concurrent operations securely. Additionally, the incorporation of a finally block served as a failsafe measure, ensuring that the lock would be released even in the event of an error, thus mitigating the risk of potential deadlocks.

Another important issue that required addressing in the original code was its vulnerability to exploitation by attack cases attempting to access closed files. To rectify this security concern, I implemented measures to set the `LPfile` attribute to `None` upon file closure, signaling that the file was genuinely closed. Furthermore, I incorporated rigorous checks in all functions to verify whether a file had been closed. If any attempt was made to access a closed file, the code was designed to raise an error, significantly enhancing the reference monitor's security and reliability.

The management of pending data underwent a substantial overhaul in the modified code. In its previous state, the code would commit prior write changes to the file and set pending data to incoming data even before verifying the validity of a write operation. This approach was flawed as it resulted in pending data and offset not accurately reflecting the last valid write operation. To address this, I revised the code to facilitate changes directly within the `writeat` function immediately before closing the file. This strategic adjustment streamlined the code structure and greatly improved its manageability.

Tracking the file size before and after various function calls was made a complex process, when using a single variable to keep track of file length before and after write or undo function was being performed. To simplify this operation, I introduced two distinct variables. The first variable was dedicated to monitoring the file size with pending data, making it a critical component for the detection of 'SeekPastEndOfFileError'. The second variable, named 'prev_size', played a role in executing the 'undo' function, allowing for the seamless restoration of the 'new_size' to its previous value. This ensured the consistent and accurate management of data within the system.

In conclusion, these comprehensive modifications led to the development of a reference monitor that is both more secure, accurate and efficient. This enhanced reference monitor effectively navigated a wide range of attack cases, while adhering to best practices to the ruleset defined by the assignment part 1.