CS5002 | Assignment #6 – Self Assessment

Bartosz Kawalkowski | M13852544 | kawalkba@mail.uc.edu

My primary contribution to the Windows Process Protector project centered on developing the core protection DLL module that provides multiple layers of security for Windows processes. I designed and implemented key security features including PE header encryption, memory scan detection, anti-tampering mechanisms, and thread monitoring functionalities. These systems work together to detect and prevent common attack vectors such as memory scanning, code injection, and unauthorized thread creation. I applied my knowledge of Windows internals and expanded my understanding of low-level security concepts, particularly around process memory protection and runtime integrity verification. My C++ skills grew substantially as I worked with Windows APIs, handled cross-process memory operations, and implemented thread-safe monitoring systems.

The most challenging aspect of my work was designing effective detection mechanisms that balanced security with performance. For example, the memory scan detection feature required careful consideration to avoid false positives while still reliably identifying unauthorized memory access. I successfully overcame obstacles related to Windows' security model limitations by implementing creative solutions like CRC32-based code integrity verification and strategic memory canaries. Through research and experimentation, I developed approaches that function effectively even against sophisticated attacks. This experience has significantly enhanced my system programming skills, deepened my understanding of application security, and improved my ability to consider security implications during software design.

As a team, Harshil and I successfully created an end-to-end process protection solution that combines a powerful protection DLL with an intuitive web-based monitoring interface. The integration between these components was seamless, allowing users to enable various protection mechanisms and monitor their status in real-time through a responsive web UI. We adopted a modular architecture that separated concerns effectively: the protection DLL handled the low-level security features while the web interface provided monitoring and control capabilities. Our collaborative approach to defining the API between these components was particularly successful, as it allowed us to work independently on our respective areas while maintaining compatibility throughout the development cycle.

The most effective aspect of our teamwork was our complementary skill sets. While I focused on the core protection mechanisms and Windows internals, Harshil excelled at developing the web UI and the communication layer between components. We maintained consistent communication through regular check-ins and collaborated effectively on overcoming integration challenges. One area where we could have improved was in establishing more formal testing protocols earlier in the development process, as we occasionally had to resolve compatibility issues that could have been prevented with better initial test coverage. Overall, our efforts were well-balanced, with both team members contributing equally to the project's success.