Kernel Pool Monitoring to Support Malware Forensics in a Cloud Computing Environment

Irfan Ahmed, Ph.D., University of New Orleans, Department of Computer Science, 2000 Lakeshore Drive, New Orleans, LA 70148
Golden G. Richard III, Ph.D., University of New Orleans, Department of Computer Science, 2000 Lakeshore Drive, New Orleans, LA 70148

After attending this presentation, attendees will understand the internals of dynamic memory allocations (a.k.a. kernel pools) in Microsoft (MS) Windows and how virtual machine introspection can be used to monitor the pools in a cloud-computing environment for detecting malware infection. Malware typically modifies function pointers to redirect the control flow of the system to execute malicious code. Tracing and identifying such stealthy hooks has significant importance in malware forensics. In this presentation, we particularly focus on monitoring and logging any suspicious modification of function pointers in the kernel pool to facilitate post-incident malware forensics.

This presentation will impact the forensic community by discussing a method of monitoring and logging suspicious modification of function pointers in kernel pools. The method is unique in that it works on MS Windows – a high priority target for malware developers, and only relies on the artifacts residing in the physical memory of the target system to work. It is also the first system of its kind that allows both 32- and 64-bit versions of the Windows kernel to be monitored for function pointer integrity.

Microsoft has introduced kernel patch protection or PatchGuard to protect kernel code and important data structures such as the interrupt descriptor table (IDT) that are typically targeted by traditional malware. This has made it difficult for malware developers to exploit the code and data structures protected by the PatchGuard. Thus, they look for low profile targets for exploitation in unexplored regions. The kernel pool is one such region where, in particular, function pointers are targeted to execute malicious code. They are numerous function pointers in the kernel pool, which provides an attractive opportunity for an attacker to install stealthy hooks. State-of the art solutions perform static analysis of kernel source code to obtain precise definitions of kernel data structures, which also identify the locations of function pointer fields in the data structures. Moreover, they generate traversal graphs by linking pointers from one data structure to another, which is then used to target the data structures in the physical memory of a target system for integrity checking. Since the solutions require source code for the operating systems kernel, they are generally not applicable to MS Windows. The seminal work of Yin et al. [1] on MS Windows uses taint analysis to learn about the definition of and contextual information for kernel data structures. However, it is dependent on taint analysis for accuracy. The work is also dependent on the relocation table to track function pointers in binary code. However, relocation table does not contain entries to locate function pointers in 64-bit Windows, which limits the applicability of their work to 32-bit versions of Windows only.

Our approach works in a virtualized environment and identifies function pointers that are maliciously modified in a kernel pool. It runs in a privileged virtual machine and uses virtual machine introspection to access the physical memory of a target virtual machine running MS Windows. Since it runs outside the target VM, it is less prone to subversion if a target VM is compromised. It obtains the list of function pointers directly from reliable sources in the physical memory of the target machine, without relying on source code, disassembling the kernel binary or traversing the relocation table. The pointer list is then used to find the instances of function pointers in kernel pool data for integrity checking. It does not require hooking to obtain the kernel pool data; instead, it uses kernel data structures maintained by Windows to track memory allocations to locate appropriate dynamic allocations in kernel pools. The authors have implemented the technique and details will be discussed in the presentation. Our experimental results show that a small region in the kernel pool has a high concentration of function pointers, which is also non-pageable, providing a rich reliable attack surface for exploitation. The implemented tool is able to perform real-time monitoring of the region.

References:

[1] H. Yin, P. Poosankam, S. Hanna, and D. Song, "HookScout: Proactive Binary-Centric Hook Detection", in Proceedings of the 7th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA'10), Bonn, Germany, 2010, pp. 1-20.

Keywords: Kernel Pool, Integrity Checking, Cloud computing