B10 Live Forensic Analysis of Kernel Code for Malware Detection in Cloud Computing Environments

Irfan Ahmed, PhD, and Golden G. Richard, PhD, Univ of New Orleans, Dept of Computer Science, 2000 Lakeshore Dr, New Orleans, LA 70148*

After attending this presentation, attendees will understand key characteristics of cloud computing environments, including typical uses of virtualization and how virtualization can be used to support live forensic analysis of operating system kernel code for malware detection. Existing techniques to assess the health of kernel code require pre-computing values such as cryptographic hashes, which is problematic in the face of dynamic updates and multiple kernel versions. A technique that leverages virtual machine introspection to assess kernel code health at runtime without relying on pre-computed values will be illustrated.

This presentation will impact the forensic science community by discussing a method to perform correlation of kernel code running in multiple virtual machines in a cloud environment without relying on traditional techniques, such as cryptographic hashes. This work is important because developing and maintaining pre-computed values for kernel code components for a large number of operating systems and kernel versions is cumbersome and has a negative impact on performance.

The integrity of operating system kernel code needs to be maintained in order to avoid disruption of service and to prevent malware from either interfering with normal system operation or accessing sensitive information. Kernel code integrity is usually violated by either patching the kernel with malicious code or loading malicious kernel modules/device drivers. Before corrective actions such as snapshot restoration can take place, malicious kernel tampering must first be detected. State-of-the-art solutions typically use a dictionary of hashes of kernel code or malware signatures for detecting malware infection. In the case of cryptographic hashes, the hashes for key “known good” kernel components are pre-computed and these values are periodically compared with the hashes of code currently loaded in the kernel. Any discrepancy indicates a potential infection.

The key problem in maintaining hashes is that code changes across different kernel versions of an operating system. Even kernel patches (e.g., for removing bugs and security vulnerabilities) change the kernel code. Thus, the dictionary needs to be maintained for all kernel versions and all the patches for each kernel version, which is a time-consuming and cumbersome task. Also, the dictionary needs to be upgraded constantly to accommodate new patches and kernel versions in order to ensure effective malware detection. Moreover, if the dictionary needs to be maintained for a cloud server where several different operating systems are run on tens or hundreds of virtual machines at a time, the scale of the problem increases significantly, requiring maintaining the dictionary for all the operating systems in the cloud server.

Recent work to automatically perform live forensic analysis of kernel code at runtime in a cloud server without maintaining a dictionary of hashes will be presented. The work uses a typical scenario in a cloud server where a pool of virtual machines runs an identical kernel version for a guest operating system. Such pools of virtual machines are maintained to simplify the maintenance process so that applying patches and upgrading systems can be automated. The technique in the presented work compares the in-memory kernel code (including kernel modules) of all the virtual machines in a pool and is based on the assumption that if kernel code has not been altered, it should be the same across all the virtual machines and any discrepancy potentially indicates a malware infection. When discrepancies are noted, additional steps such as gathering snapshots for further forensic investigation and restoring the system to a known good state can be undertaken.

A prototype implementation of the technique will be discussed in the presentation. This tool runs on a privileged virtual machine where it can access the physical memory of other virtual machines in the pool through virtual machine introspection. This technique is resistant to tampering by malware because no components run inside the guest virtual machines. This prototype is for Microsoft® Windows® and the technique has been tested against a variety of exploits, including those based on opcode replacement and DLL hooking. This technique successfully detects all of the tested exploits and does not affect the performance of the guest operating systems.

Digital Forensics, Code Integrity, Cloud Computing

 ** Presenting Author*