B7 Kernel-Independent Tools for Deep, Live Digital Forensic Investigation

Golden G. Richard III, PhD, University of New Orleans, Department of Computer Science, New Orleans, LA 70148*

After attending this presentation, attendees will understand the importance of developing kernel-independent, deep analysis tools to support live digital forensic investigations. Existing live forensics tools either perform shallow analysis, relying on the target's operating system to retrieve data of evidentiary value, or target specific operating system and kernel versions. This limits their usefulness for investigating machines in the wild. This presentation discusses ongoing research in developing deep analysis tools for live forensics, which automatically adapt to different operating system and kernel versions.

This presentation will impact the forensic science community by discussing methods for designing portable live forensics tools that are applicable to a wide range of forensic targets. This work is important because manually developing support for the large number of operating system and kernel versions now widely deployed is impractical.

A number of factors have contributed to an increasing interest in live forensics, where the machine under investigation continues to run while forensic evidence is collected. These factors include a huge increase in the size of forensic targets, increasing case backlogs as more criminal activity involves the use of computer systems, and the need to turn around cases very rapidly to counter acts of terrorism or other criminal activity where lives or property may be in imminent danger. In addition, a live investigation may reveal a substantial amount of volatile evidence that would be lost if only a traditional "pull the plug" investigation were performed. This volatile evidence includes lists of running processes, network connections, data fragments such as chat or email messages, and keying material for drive encryption. All of this volatile information can be crucial in expediting processing of a case, by providing critical contextual information that supplements traditional analysis, such as processing disk images.

Early live forensics efforts typically involved running a number of statically linked binaries on the forensic target (e.g., *ls, ps, lsmod*, *lsof,* etc. under Linux) and capturing the output of these commands for later analysis. A physical memory dump might also be captured, but analysis of the physical memory dump was often limited to simple string searches. Recently, research into deeper, advanced techniques has substantially increased the capabilities of live investigation. Physical memory dumps can now be analyzed to reconstruct models of the current and historical states of a live system under investigation. This kind of analysis relies on deep understanding of data structures in the running operating system kernel to extract evidence pertinent to an investigation.

A key problem in developing tools for deeply analyzing live systems is that the format of key kernel structures changes across versions of the installed operating system. This is a problem for both Microsoft Windows and for Linux. For Microsoft operating systems, support is required for Windows NT, 2000, XP, Vista, Windows 7, and the various server offerings, with patch levels (e.g., XP with SP2) introducing even more diversity. For Linux, the problem is much more severe, because official kernel versions are released much more frequently and a large installed base of each version may be present. Furthermore, different distributions, such as Ubuntu, Red Hat, SUSE, etc. may introduce distribution-specific modifications to the kernel. Finally, since the Linux kernel is open source, individual users can modify and install custom versions of the kernel. Linux kernel structures that must be analyzed for deep, live forensics therefore change quite frequently and constantly *manually* updating tools to support new kernel variants is essentially an impossible task.

Automatic model key kernel data structures, such as process descriptors and process memory maps, regardless of kernel version will be presented. The techniques rely on on-the-spot disassembly of important kernel routines, with pattern matching applied to the resulting

disassemblies, to discern the structure of kernel data. This is basically an automatic, on-the-spot reverse engineering effort against important kernel components. A description of the techniques and the live forensics tools built using these techniques will be provided in the presentation. The tools use both list-walking approaches as well as data carving techniques to discover both current (e.g., processes now running, currently open network connections) and historical (e.g., terminated processes, closed network connections) volatile evidence.

Digital Forensics, Live Forensics, Linux

 ** Presenting Author*