

EICAR 18th Annual Conference
11 May to 12 May 2009
with a pre-conference program on
9 May and 10 May 2009

Abstract of Student Workshop

Anthony Desnos

ESIEA – SI&S - France

Live Memory Forensics techniques under Linux

Abstract: Forensic analysis is not a new technical science, and there exist now many ways to make it as easiest as possible. But what happens whenever nothing is on the hard disk?

Write access on hard disks is the worst case for the hacker or the attacker, because traces left on the device are bound to compromise the attack afterwards. Consequently this situation enables investigators to find evidences and thus to frame the attacker. The forensic analysis of memory has become over the years an important thing to recover evidence of a cyber crime scene. However, the development of technical solutions has been developed on widely public systems only. But quite almost nothing is known with respect to on Linux servers, which are however mostly used for store and carry out sensitive operations?

The talk will start with the forensic analysis of memory on a Linux operating system by considering the worst case and we will show the difficult way to realize the forensics operations. For example, no hardware physical access, no lkm supports, and no modified kernel which can format internals structures of the kernel. We will have only the physical and virtual memory devices -- respectively /dev/mem and /dev/kmem, ie the same access as hackers - to our disposal.

We will show how through a simple dump of memory, or a direct access to the virtual device, to find information as the process list at an instant time t, but also how to rebuild the binaries to enable a static or dynamic analysis after memory dump. This analysis will therefore allow us to recover the tools that a hacker was able to run on our machine, even if those have been erased, and perhaps not even written on the hard disk. We will also see that although this method is workable, there are new ways for hackers to leave the minimum amount of information in memory.

We will use the same methods as a hacker to access the memory of the system, to walk through this memory, of course with significant differences however, but in the end we will know that the techniques of his enemy is the best way to fight him.

Gregoire Jacob

ESIEA - France

JavaScript and Visual Basic Script Threats: different languages for different malicious purposes

Abstract: JavaScript (JS) and Visual Basic Script (VBS) come from the same family of languages called scripting languages. Originally, these two languages have been introduced to enrich web pages with dynamic features. But the comparison stops here. In fact these two languages show fundamental differences explaining they are now used for different purposes.

This tutorial will thus begin by a comparison of the JS and VBS languages: do they provide the same facilities? Do they offer the same portability? Do interpreters enforce security mechanisms? Responding to these questions, the presentation will explain why the threats have evolved towards two different directions between web-application malware for JS and stand-alone malware for VBS. For each of these threats, their main principle will be presented in details: a first recall will be given on the well-known malware in VBS (IRC-Worms, Mail-Worms, Drive-Worms), followed by a second survey on web-application malicious scripts in JS (XRSF attacks, XSS attacks). All along, the speech will be illustrated with several code analyses of different malware examples.

From the defence perspective, the tutorial will cover different existing solutions deployed against malicious scripts. Citing script certification, binary scanners or event collectors, all these protection tools do not come without constraints. Conversely, the tutorial will also address the techniques for scripts to avoid detection, and in particular obfuscation techniques. Even if these languages are interpreted, and the source code thus available; advanced obfuscation remain possible and hard to reverse.