Virtualization based physical memory tracing

Subject: IT Security
Thesis advisor: Dr. Endre Bangerter
Expert: Reto Inversini (MELANI)

Memory tracing is a novel technique for capturing memory changes of an unknown running program or a whole system over time. The solution we present here works using hardware virtualization, thus capturing changes in the entire physical memory. The visualization of the captured data (after some processing) allows us to understand some key aspects of an unknown software in little time. Gaining this knowledge quickly is important to efficiently fight malicious software (e.g. viruses, rootkits, bootkits, etc.).

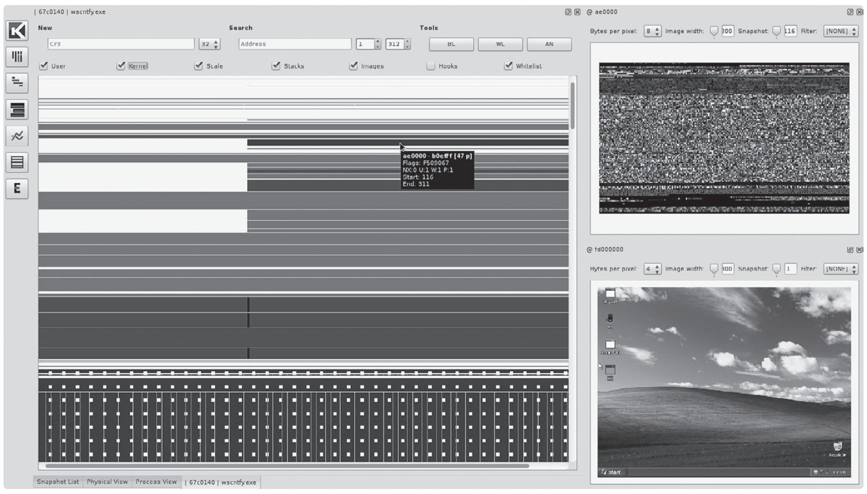
In 2012, we presented a new technology to analyze (malicious) software. Back then, the system collected runtime information of an arbitrary Windows XP process. The collected data consisted of various process memory, their changes over time and a list of executed system functions. To overcome the huge amount of data generated by the system, we developed a few preprocessing scripts to calculate meta data. The final step then was to visualize the recorded data with its meta data resulting in an innovative system to reverse engineer software. Our system proved to be an invaluable tool for analyzing and understanding malicious software. However, there were some drawbacks with this system. Especially portability difficulties and the lack of the system memory. We needed a new, more generic system, that captures all memory changes in the whole physical memory of any modern operating system.

The new solution consists mainly of a recorder and a viewer. The recorder is implemented in C as a Linux kernel module which links itself at runtime to the Kernel Virtual Machine (KVM). That means, whatever KVM is able to run, the recorder is able to record – e.g. a program, a system driver or a whole system boot on any version of Windows, Linux or Mac. The hardware virtualization of KVM also gives us great advantages such as stealthiness, security and a high recording performance. In the current implementation the analyst can specify at what frequency a snapshot should be made, that is, enumerating all memory changes and writing them to the hard disk.

The viewer component is implemented in C++ with help of the frameworks boost and Qt. It visualizes the raw data along with some preprocessed meta information. There is also a basic tool-set included to explore, view, export, blacklist, whitelist or diff memory sections. The additional capturing of the system memory, compared to its predecessor, gives us the ability to get information about the state of the operating system at the time a snapshot was made, e.g the running processes, open files and network connections. We import such information with help of the Volayility memory forensic tool to annotate the visualization further.

Currently, we focus only on the reverse engineering aspect, but there are a lot of other applications memory tracing can offer. Another use case would be memory integrity checks on running systems to prove the correctness of a system.

Dominic Fischer



Visualized memory trace of a banking trojan in Windows XP

VA BU BE BI