An Architecture for Holistic Collaborative Operating System Monitoring

Jesus Navarro, 2013, Ruben Martinez, 2015

The traditional model for Operating System (OS) monitoring and protection employs a trustworthy virtual machine (VM) where security solutions are placed overseeing an OS assumed to be unreliable and easily compromised. The OS plays no active role in its own monitoring, which creates a debilitating semantic gap; the difference between the information available to the OS and the information available to the VM is a weakness left exploitable to malware. Our goal with this project is to look at this problem with a different security model, one in which the OS and the VM holistically cooperate in the monitoring and protection process. This approach bridges the semantic gap as the OS is able to easily provide rich information to the VM, without opening the OS to further vulnerabilities.

For our approach, we used Bochs, an Intel x86 emulator as the VM and Ubuntu Linux 10.10 as the OS to implement a proof-of-concept prototype. Our model consists of two parts: the downcall and the downcall manager. The downcall is a “message” that the OS sends to the VM in the form of a software interrupt. When this message is issued, the OS will stop whatever work it was doing in order to serve this message (in other words, it is interrupted from its original work and will continue once the interruption has been dealt with). We have added downcalls throughout the OS to notify our VM of special events, such as the creation of a process, so it is able to keep track of what is happening within the OS better than it would otherwise. The downcall manager is a portion of the VM we have added that will receive a downcall from the OS and handle it accordingly. This is the crux of our security solution. The downcall manager must manage and interpret the information that is being passed by the OS to the VM (see fig 1). Its tasks can range from simply gathering information to tagging pieces of memory as suspicious and untrustworthy. The downcall and downcall manager work together to build a database of information that will aid in the prevention and detection of malicious software within the OS.

Our results show that our security model is promising. To test this, we implemented two rootkits: one that attempts to tamper with the OS itself and one that attempts to issue its own downcall in order to sabotage our downcall manager by feeding it false information. We also installed multiple benign loadable kernel modules, which are pieces of software that add functionality to the OS without the need to rewrite the entire OS. Our solution was able to correctly identify the benign modules as harmless and the malicious rootkits as harmful.

Fig 1: List of processes gained from OS downcalls. The list includes the process name, process id, and a pointer to a location in memory

Faculty Mentor: Daniela Oliveira
Funded by the Freedman Summer Research Fellowship