Fine-grained OS Behavior Characterization

Lorenzo Cavallaro
Department of Computer Science
Vrije Universiteit, Amsterdam
sullivan@cs.vu.nl

Cristiano Giuffrida*
Department of Computer Science
Vrije Universiteit, Amsterdam
giuffrida@cs.vu.nl

Andrew S. Tanenbaum
Department of Computer Science
Vrije Universiteit, Amsterdam
ast@cs.vu.nl

Monolithic operating systems (OSes) are complex pieces of software that usually offer very little reliability and security guarantees. Faulty user-space applications can generally be restarted without affecting the existing concurrent communications but those involving the faulty processes. On the other hand, in a monolithic OS design, the kernel and all its components share a common address space and any component can potentially invoke any kernel function. In this scenario, it becomes extremely complicated—if not impossible—to isolate and restart faulty kernel components as it is generally hard to define their boundaries and interactions (e.g., what kernel control paths are executed and how information is shared).

Unfortunately, run-time bugs are not the only security threats an OS must deal with. For instance, malicious components may undermine the security of the whole system from its root. Kernel rootkits can be installed on the system to replace or modify the legitimate behavior of arbitrary subsystems of the OS. It is under this perspective that it becomes clear we need techniques that enable us to characterize and describe the behavior of the whole OS, i.e., the kernel and its components. The research community has so far proposed a number of approaches aimed at characterizing the behaviors of user-space processes. Although each one different from the other, such techniques generally rely on the same underlying intuition, i.e., the behavior of a process can be expressed by the sequence of system calls the process invokes [1]. Unfortunately, such an intuition easily drops when monolithic OSes are considered, because well-defined and easy-to-enforce communication interfaces among kernel subsystems are generally missing. While promising, recent attempts to characterize the behavior of kernel OS can be indeed quite complicated and currently confined to malware analysts rather than on-line behavior-based policy enforcement for end-user systems. For instance, the completeness and scalability of the analysis proposed in [3] depend on the particular selection of kernel execution paths and well-defined knowledge on what can be considered sensitive.

It is our belief that a microkernel and multiserver OS design offers better opportunities to characterize and describe the behavior of the whole operating system. Recent studies have shown that program-centric analyses approaches may not be well-suited to describe the behavior of generic user-space processes, as they may not generalize well, especially when the considered applications are exposed to a plethora of previously-unseen and realistic input [2]. In such scenarios a system-centric model seems to be more appropriate and recent results have backed-up such claims especially when it comes to telling benign and malicious behaviors apart. However, a microkernel multiserver OS is made of different components (e.g., core components and device drivers) that perform, by design, specific tasks. We believe these highly specific task-oriented behaviors can indeed be easier to characterize than those of generic user-space processes. In fact, under our design, OS subsystems boundaries are well-defined and interactions among components follow specific rules subjected to well-defined APIs (i.e., IPC calls) responsible for carrying out the communication among all the components.

Our approach is to leverage such a multiserver microkernel-based OS design augmented with an IPC interceptor that constantly monitors the global IPC traffic. We propose a warming-up phase to perform a per-component stress testing to learn the behavior of each component in terms of IPC interactions with other components. Once a model of the component behavior is available, the IPC interceptor can monitor the IPC traffic at runtime and raise an alert when an anomalous behavior is detected with respect to a number of predefined, but customizable, policies. We believe this infrastructure for a fine-grained OS behavior characterization is important for a broad number of appealing dependability applications. For instance, we are planning to use our approach to detect buggy OS behaviors in case of byzantine failures. Likewise, we are interested in evaluating the effectiveness of our approach to detect a malicious behavior as a consequence of a newly installed component (e.g., malware) or security attacks exploiting OS vulnerabilities. Another interesting aspect in characterizing the behavior of OS components is to employ our approach for online patch validation. In particular, this is useful to estimate the functional changes in terms of behaviors between two different versions of a component. For instance, it should be possible to automatically detect a new version of a component that violates a given update specification.

References

[1] FORREST, S., HOFMEYR, S. A., SOMAYAJI, A., AND LONGSTAFF, T. A. A Sense of Self for Unix Processes. In *Proc. of the IEEE Symposium on Security & Privacy* (1996).

[2] LANZI, A., BALZAROTTI, D., KRUEGEL, C., CHRISTODERESCU, M., AND KIRDA, E. AccessMiner: Using System-Centric Models for Malware Protection. In *Proc. of 17th ACM CCS* (Oct 2010).

[3] LANZI, A., SHARIF, M. I., AND LEE, W. K-Tracer: A System for Extracting Kernel Malware Behavior. In *Proc. of the 16th Annual NDSS Symposium* (Feb 2009).

*PhD student at Vrije Universiteit, Amsterdam