KernelSec: An Authorization Model in the Operating System Kernel

Manigandan Radhakrishnan and Jon A. Solworth
University of Illinois at Chicago
{mani, solworth}@rites.uic.edu http://www.rites.uic.edu/kernelSec

An authorization system is at the core of the mechanisms that provide system security. It is responsible for allowing or denying user actions (like request to read or write a file, to connect to a website or kill a process). Despite considerable research in authorization systems, widely deployed authorization systems struggle to cope with today's security needs.

Some of the challenges facing today's computer systems are

1. the programs installed on (or downloaded onto) a system come from a variety of sources, not all are equally trusted,
2. the users of computer systems vary widely in terms of their understanding of the underlying computer security mechanisms, and
3. the lack of sufficient operating system security mechanisms, in the past, has lead to applications providing their own security features. This has several negative consequences, including the fact that a successful exploitation of a bug in the application can render these protections ineffective.

Hence there is a need for sound operating system based authorizations that can secure the system in the face of these challenges.

KernelSec Project: KernelSec is a general-purpose authorization model that provides

strong protections using a mandatory access control model implemented as part of the operating system kernel;

easy configuration using a two-level approach to configuration;

robustness by allowing security changes to the system to be encoded as part of the authorization state; and

better compatibility or usability by making the enforcement transparent to applications.

Enforcement Model: The privileges a process has is a reflection of the set of actions that the process is allowed to perform. Traditionally, the set of privileges primarily depends on the user running the process The disadvantage is that all the processes of a user are executed with the entire set of user privileges. In KernelSec, the enforcement engine bases the authorization not only on the user, but also on the *executable* (or the program) and the *history of accesses.* Keeping track of the history allows KernelSec to change the privileges available (dynamically) at run-time.

Associated with every process is a *domain* which describes its privileges. In KernelSec, the domain may be changed to obtain a needed (but missing) privilege. Such dynamic domain transitions allows KernelSec to track process history and suitably change privileges by switching to a different domain.

Sometimes user actions need to be recorded beyond the lifetime of a process. KernelSec implements *active transitions* to support this. KernelSec also implements a sophisticated group mechanism that among other things can represent relationship among groups.

Implementation: The project at this time is being implemented as part of the GNU/Linux operating system, with the enforcement engine being implemented as part of the Linux kernel (v2.6) using the Linux Security Modules (LSM).

Poster and Demo: We use the particular case of *Sandboxing* to illustrate how the general-purpose mechanisms of KernelSec can be used to dynamically confine an application inside a sandboxed-environment in the event that it is subjected to untrusted data.