Security in Innovative New Operating Systems

Cynthia E. Irvine
Computer Science Department
Naval Postgraduate School
Monterey, California 93943
irvine@cs.nps.navy.mil

1 A Traditional Approach to Trust

The *Reference Monitor Concept* [1] is an abstraction describing properties that must be satisfied by a mechanism enforcing a policy regulating access to information. Those properties are: tamperproof, always invoked, and small enough to be subjected to analysis and testing, the completeness of which can be assured.

It is realized by the creation of a *reference validation mechanism* which uses an authorization database to validate access by users, or their surrogates, viz. processes, to information in the form of data.

Traditional approaches to constructing reference validation mechanisms have combined hardware and software mechanisms to create protection domains [8]. The reference validation mechanism must create a domain for its own execution that satisfies both the requirements for tamperproofness and non-bypassability. Otherwise, the mechanism is not conceptually complete. It is impossible to demonstrate that a mechanism is tamperproof if it can be bypassed and the underlying resources upon which it depends can be manipulated. Similarly, a mechanism cannot be called non-bypassable, if it is subject to tampering which might permit detours around the validation checks.

In traditional operating systems, whether or not they contain a reference validation mechanism, encapsulation of privileged mechanisms is costly. Context switches between the external domain and that containing the operating system or reference validation mechanism are required. The result is a performance penalty.

2 New Operating Systems

Today, a principal criterion by which new operating systems are judged is the level of performance that they provide for applications. To this end, new operating systems have sought novel approaches to performance enhancement. A theme common to many of these initiatives is that of specialization. Instead of an operating system designed to serve all applications (either equally well or equally badly), the operating system is adapted to serve the needs of the application. The intent is not to provide a different static operating system for each application but to allow the operating system to be dynamically modified or "specialized" to best serve each application.

These include:

- Abandonment of resource abstraction by the operating system in exchange for small policy-neutral kernels designed to multiplex hardware-level resources to the application domain where library operating systems eliminate the need for context switching to obtain many traditional operating system services.
- An approach in which costly context switches required for applications to access underlying resources are eliminated by allowing applications to insert extensions into the kernel at runtime while still maintaining the integrity of the operating system and isolation of applications.
- Exploration of the use of type-safe languages as a way to insure policy enforcement by the operating system and/or applications.
- New designs to provide performance optimization by creating just-in-time operating system support: the dynamic generation and composition of operating system code.
- Innovative use of virtual machine concepts to build highly efficient component-based systems that will be flexible with repect to inter-component trust and to the control of resources by various subsystems.
- The use of code components to which are bound proofs of safety obligations which are imposed by the target system and which may be checked just prior to execution.

Not only are we presented with the possibility that the boundary between the application and the security policy enforcement mechanism is blurred, but the dynamic nature of these systems raises serious questions regarding the chain of evidence that can be used to demonstrate that the policy is being correctly enforced. What can be stated about the assurance of correct security policy enforcement in a system resulting from the dynamic creation and composition of modules? In the past, the use of compilers to enforce security was considered inadequate to protect information of different sensitivity levels [6]. What can be said today about approaches that depend upon languages and compilers?

3 The Panel

The five operating system efforts to be presented in this panel are: the Exokernel Project [3], the Fluke Project [4], the Fox Project [7], the Scout Project [5], and the SPIN Project [2]. We hope to give an overview of the innovative techniques being used to enhance performance in these systems and to discuss the effect of those enhancements on our ability to reason about the security properties of systems.

References

[1] J. P. Anderson. Computer Security Technology Planning Study. Technical Report ESD-TR-73-51, Air Force Electronic Systems Division, Hanscom AFB, Bedford, MA, 1972. (Also available as Vol. I,DITCAD-758206. Vol. II, DITCAD-772806).

[2] B. Bershad, S. Savage, P. Pardyak, E. Sirer, M. Fiuczynski, D. Becker, C. Chambers, and S. Eggers. Extensibility, Safety and Performance in the SPIN Operating System. In *Operating Systems Review*, volume 29, pages 267–284, December 1995.

[3] D. Engler, M. Kaashoek, and J. O'Toole. Exokernel: An Operating System Architecture for Application-Level Resource Management. In *Operating Systems Review*, volume 29, pages 251–266, December 1995.

[4] B. Ford, M. Hibler, J. Lepreau, P. Tullmann, G. Back, and S. Clawson. Microkernels Meet Recursive Virtual Machines . In *Proceedings of the Second Symposium on Operatins Systems Design and Implementation*, pages 137–151, Seattle, WA, October 1996. USENIX Assoc.

[5] A. B. Montz, D. Mosberger, S. W. O'Malley, L. L. Petersonand, T. A. Proebsting, and J. H. Hartman. Scout: A Communications-Oriented Operating System. Technical Report 94-20, Department of Computer Science, University of Arizona, June 1994.

[6] National Computer Security Center. *Final Evaluation Report of UNISYS Corporation A Series MCP/AS, Release 3.7*, CSC-EPL-87/003, Fort Meade, MD, 5 August 1987.

[7] G. Necula and P. Lee. Safe Kernel Extensions without Runtime Checking. In *Proceedings Second Symposium on Operating Systems Design and Implementaitons*, pages 229–243, Seattle, WA, October 1996. USENIX Assoc.

[8] J. H. Saltzer and M. D. Schroeder. The Protection of Information in Computer Systems. *Proceedings of the IEEE*, 63(9):1278–1308, 1975.