Reference Monitor

Trent Jaeger
Systems and Internet Infrastructure Security Lab
Pennsylvania State University

Related Concepts

– Access control
– Access control policy
– Security kernel

Definition

A reference monitor concept defines a set of design requirements on a reference validation mechanism, which enforces an access control policy over subjects' (e.g., processes and users) ability to perform operations (e.g., read and write) on objects (e.g., files and sockets) on a system.

– The reference validation mechanism must always be invoked (*complete mediation*).
– The reference validation mechanism must be tamperproof (*tamperproof*).
– The reference validation mechanism must be small enough to be subject to analysis and tests, the completeness of which can be assured (*verifiable*).

The claim is that a reference validation mechanism that satisfies the reference monitor concept will correctly enforce a system's access control policy, as it must be invoked to mediate all security-sensitive operations, must not be tampered, and has undergone complete analysis and testing to verify correctness.

Background

In 1972, James P. Anderson coordinated the Computer Security Technology Planning Study with a panel of ten industry, government, and academic security experts [2]. The goal of the panel was to determine the requirements for U.S. government computer systems to execute securely in the presence of malicious users. Contemporary systems assumed that all threats originated from external attackers, but malicious users in control of processes running on the system may also violate the system's access control policy. For example, a malicious programmer may steal information accessed by processes running her program or a malicious user may steal other users's data stored on the same system.

To prevent unauthorized access by malicious users, the panel recommended that computing systems be designed in accordance with requirements embodied by the *reference monitor concept*. The reference monitor concept envisions that a system component, called a reference validation mechanism, will be responsible for enforcing the system's access control policy over user process operations. The reference monitor concept defines the requirements for implementing such a mechanism in a manner that ensures that malicious users cannot circumvent policy enforcement.

Theory

The claim is that by implementing a reference validation mechanism in accordance with the reference monitor concept all accesses by user processes will adhere to an access control policy enforced by the mechanism. This claim is based on successful implementation of the three design requirements of the definition above.

First, the *complete mediation requirement* specifies that the reference validation mechanism mediates all security-sensitive operations by user processes. Complete mediation enables the reference validation mechanism to authorize each security-sensitive operation against the access control policy. Since only security-sensitive operations can violate the access control policy, such mediation ensures that user processes can only perform operations authorized by the access control policy.

Second, the *tamperproof requirement* specifies that the reference validation mechanism cannot be modified by user processes. This prevents a malicious user from modifying the behavior of the reference validation, e.g., to approve operations that are not allowed by the access control policy. In practice, the tamperproof requirement also covers the access control policy itself. This prevents a malicious user from gaining unauthorized access by modifying the access control policy enforced by the reference validation mechanism.

Third, the *verifiable requirement* specifies that the reference validation mechanism be small enough to enable practical verification of correctness. A reference validation mechanism is correct if it generates the correct access control query, processes that query correctly against the access control policy, and correctly implements the resultant decision (allow or deny). While not strictly implied by the requirement, it is also desirable to determine whether the access control policy is correctly specified relative to some goal.

Application

The Multics operating system [14] was the first one designed with comprehensive security enforcement, aiming to protect the secrecy of data and the integrity of the operating system and trusted software. However, as the Multics project matured, it became clear that the Multics system was complex, and this hindered the developers ability to determine whether the proposed security guarantees were correctly enforced [13].

The development of the reference monitor concept aimed to remedy this problem by defining the requirements for a correct and unbypassable reference validation mechanism. Subsequently, several projects emerged to demonstrate the efficacy of building reference validation mechanisms based on these design requirements. These types of operating systems were called *security kernels*, as they each contained a small, core component that included a reference validation mechanism designed to satisfy the reference monitor concept [1]. Examples of commercial security kernels were Scomp [4] and GEMSOS [12].

Since that time, the reference monitor concept has been considered a foundation of secure system design. This is reflected in the U.S. Government's criteria for building secure systems, the Trusted Computer System Evaluation Criteria [17] (the "Orange" book), where the reference monitor is used as a motivation for the choice of its security evaluation classes. Gasser's seminal book [5], "Building a Secure Computer System," also based its design methodology on the reference monitor concept. This book motivated a variety of researchers and companies to leverage microkernel and hypervisor systems for security enforcement, as their small size made them more amenable to verification. Finally, Irvine suggested that the reference monitor concept be used as a guiding principle for computer security education [6].

Despite this convergence on the reference monitor concept in the security community [3,9,11], commercial operating system designs were largely unaffected. Trusted Solaris [15] was an exception, as the Solaris system was forked in the late 1980s to deploy an operating system for enforcing multilevel security. However, this version of Solaris remained separate from mainstream Solaris until 2006. It was not until the early 2000s, when the fundamental security problems in commercial systems were finally accepted [10], that a reference validation mechanism aiming for the reference monitor concept was developed for a mainstream, commercial operating system. The Linux Security Modules (LSM) framework was implemented for Linux 2.6 [18], enabling the support of multiple reference validation mechanism. That is, the LSM framework provides complete mediation, whereas the choice of module (and supporting system services) determines how tamperproofing and verification are achieved. A similar design has been applied to TrustedBSD, MAC OS X, the Xen hypervisor, and some user-space programs, most notably X Windows.

Open problems

The main open problems in meeting the requirements of the reference monitor concept involve verifying those requirements on implementations of reference validation mechanisms [7]. Each requirement has its own unique challenges. Complete mediation requires that all security-sensitive operations are identified, but often these operations are not precisely defined. Tools have been built to verify complete mediation for the LSM framework [8,16,19], and several bugs were found (since fixed). For tamperproofing, the problem is that most systems have a trusted computing base that is too large to determine whether tampering is prevented. In many systems, several user-level processes are trusted with the authority to modify the kernel (e.g., install modules), but many of these processes themselves are vulnerable to tampering. However, verification is the most difficult of the requirements to satisfy, as designing a general algorithm to prove that an arbitrary program behaves correctly is tantamount to solving the Halting problem. While current algorithms can prove correctness properties of specific programs, the variety of reference validation code and the complexity of correctness properties preclude verification for all but the smallest, most specialized systems.

Recommended Readings

[1] S. A. Ames, M. Gasser, and R. R. Schell. Security kernel design and implementation: An introduction. *IEEE Computer*, 16(7):14–22, 1983.

[2] J. P. Anderson. Computer security technology planning study. Technical Report ESD-TR-73-51, `http://seclab.cs.ucdavis.edu/projects/history/`, The Mitre Corporation, Air Force Electronic Systems Division, Hanscom AFB, Badford, MA, 1972. Volumes I and II.

[3] M. Branstad, H. Tajalli, F. L. Mayer, and D. Dalva. Access mediation in a message passing kernel. In *Proceedings of the 1989 IEEE Symposium on Security and Privacy*, 1989.

[4] L. J. Fraim. SCOMP: A solution to the multilevel security problem. *IEEE Computer*, 16(7):26–34, 1983.

[5] M. Gasser. *Building a Secure Computer System*. Van Nostrand Reinhold, 1988. `http://cs.unomaha.edu/~stanw/gasserbook.pdf`.

[6] C. Irvine. The reference monitor concept as a unifying principle in computer security education. In *Proceedings of the 1st World Conference on Information Systems Security Education*, June 1999.

[7] T. Jaeger. *Operating System Security*. Morgan & Claypool, 2008.

[8] T. Jaeger, A. Edwards, and X. Zhang. Consistency analysis of authorization hook placement in the Linux security modules framework. *ACM Transactions on Information and System Security (TISSEC)*, 7(2):175–205, May 2004.

[9] P. A. Karger, M. E. Zurko, D. W. Bonin, A. H. Mason, and C. E. Kahn. A retrospective on the VAX VMM security kernel. *IEEE Transactions on Software Engineering*, 17(11):1147–1165, 1991.

[10] P. A. Loscocco, S. D. Smalley, P. A. Muckelbauer, R. C. Taylor, S. J. Turner, and J. F. Farrell. The Inevitability of Failure: The flawed assumption of security in modern computing environments. In *Proceedings of the 21st National Information Systems Security Conference*, pages 303–314, October 1998.

[11] S. E. Minear. Providing policy control over object operations in a Mach-based system. In *Proceedings of the 5th USENIX Security Symposium*, pages 141–156, 1995.

[12] R. Schell, T. Tao, and M. Heckman. Designing the GEMSOS security kernel for security and performance. In *Proceedings of the National Computer Security Conference*, 1985.

[13] M. D. Schroeder. Engineering a security kernel for Multics. In *Proceedings of the Fifth ACM Symposium on Operating Systems Principles*, pages 25–32, 1975.

[14] M. D. Schroeder, D. D. Clark, J. H. Saltzer, and D. Wells. Final report of the MULTICS kernel design project. Technical Report MIT-LCS-TR-196, MIT, March 1978.

[15] Sun Microsystems. Trusted Solaris 8 Operating System. `http://www.sun.com/software/solaris/trustedsolaris/`, February 2006.

[16] Lin Tan, Xiaolan Zhang, Xiao Ma, Weiwei Xiong, and Yuanyuan Zhou. AutoISES: Automatically inferring security specifications and detecting violations. In *Proceedings of the 17th USENIX Security Symposium*, pages 379–394. USENIX Association, 2008.

[17] Trusted Computer System Evaluation Criteria (Orange Book). Technical Report DoD 5200.28-STD, U.S. Department of Defense, December 1985.

[18] C. Wright, C. Cowan, S. Smalley, J. Morris, and G. Kroah-Hartman. Linux Security Modules: General security support for the Linux kernel. In *Proceedings of the 11th USENIX Security Symposium*, pages 17–31, August 2002.

[19] X. Zhang, A. Edwards, and T. Jaeger. Using CQUAL for static analysis of authorization hook placement. In *Proceedings of the 11th USENIX Security Symposium*, pages 33–48, San Francico, CA, USA, August 2002.