© 2014 IJIRT | Volume 1 Issue 5 | ISSN : 2349-6002

Operating Systems Security – A Review

Ipsita Vashista, Harsha Suri, Disha Papneja
Dronacharya College of Engineering

***Abstract*—Operating System is the core piece of software which runs on all information systems, such as network devices (routers, firewalls, etc.), Web servers, customer desktops, PDAs, and so on. And the security of operating systems is one of the fundamental concerns in the security of cyberspace and e-commerce. Issues of operating system security occupy a central role in applied computer science; yet there has been no satisfactory complete solution to the problem of computer security. Many known vulnerabilities discovered so far are rooted from the bugs or deficiency in operating systems.**

This paper is a review on the security and lack of it in the most commercial operating systems like UNIX and Microsoft Windows, and its effect to the overall security of Web based applications and services.

***Index Terms*— System Security, Mandatory Security, Security Perimeter, SE-Linux, RBAC.**

I. INTRODUCTION

Also known as operating system, kernel is the core piece of software every modern computer system from network servers, workstation desktops, to laptops and hand held devices. It is executed on top of a bare machine of hardware that allocates the basic resources of the system (CPU, memory, device driver, communication port, etc.), and supervises the execution of all applications within the system. Microsoft Windows, different flavors of UNIX (BSD, AIX, HP-UX, Solaris, etc.), Mac OS, and Linux are some of the popular commercial and Open Source operating systems.

Operating systems have a critical role in the operation of any computer systems. The security (or the lack of it) of an operating system has fundamental effects on the overall security of a computer system (including the security of all the applications and softwares running on that system). Any compromises in the security of an operating system will put any application running on the system in danger. Lack of proper control and containment of execution of individual applications in an operating system may lead to attack or break-in from one application to other applications [11].

II. SECURITY OF OPERATING SYSTEMS

Concurrent execution of multiple applications in a single physical computing hardware (which may have multiple processing units) is provided by most of the modern computer systems. In these multitasking and time sharing environments, individual applications share the same system resources, e.g. CPU, disk, memory, etc. under the operating system's control. In order to protect the execution of individual application jobs from possible interference and attack of other jobs, most contemporary operating systems implement some abstract property of containment, such as process (or task) and TCB (Task Control Block), virtual memory space, file, port, and IPC (Inter Process Communication), etc. [11].

The access decisions of most of the commercial operating systems (MS Windows, UNIX, etc.) is based on user identity and ownership. The role of the user, trustworthiness of the programs, sensitivity or integrity of data, and other such security relevant criteria are not considered. It is not possible to control data flows or enforce a system wide security policy as long as users or applications have complete discretion over objects. Due to such weakness it is very easy to break in the security of a system once an application has been compromised. Some examples of potential exploits from a compromised application are [5]:

- Use of unprotected system resources illegitimately. For example, a worm program launches attack via emails to all targets in the address book of a user after it gets control in a user account.
- Subversion of application enforced protection through the control of underneath system. For example, to deface a Web site by gaining the control of the Web server of the site, say changing a virtual directory in Microsoft IIS.
- Gain direct access to protected system resources by misusing privileges. For example, a compromised "send mail" program running as root on a standard Unix OS will result in super user privileges for the attacker and uncontrolled accesses to all system resources.
- Furnish of bogus security decision-making information. For example, spoof of a file handle of Sun's NFS may easily give remote attackers gaining access to files on the remote file server.

Protection against malicious code of an application using existing mechanisms of most commercial operating systems is not possible since a program running under the name of a specific user receives all of the privileges associated with that user.

III. MODEL OF SECURITY

In an access control based security model, there are two sets-a set of objects and a set of subjects, which can itself be an object. Each object and subject has a corresponding security attribute or label or clearance, and a defined set of control rule or security policy. This determines which subject is authorized to access which object. For example, in military security model [7], a security label consists of two components: a security level with one of the four ratings: unclassified, confidential, secret, and top secret, where unclassified < confidential < secret < top secret, and "<" means "less sensitive than"; a set of zero or more categories (also known as compartments) that describe kinds of information, for instance, the names CRYPTO, NUCLEAR might mean information about cryptographic algorithms, and nuclear related technology. Given two security labels, (X, S1) and (Y, S2), (X, S1) is defined as being "at least as sensitive as" (Y, S2) iff X • Y and S2 $\subseteq$ S1. For example, (TOP SECRET, {CRYPTO, NUCLEAR}) > (SECRET, {CRYPTO}) where ">" means "more sensitive than". In general, security labels are partially ordered. That is, it is possible for two labels to be incomparable, in the sense that neither is more sensitive than the other. For example, neither of the following is comparable to each other: (TOP SECRET, {CRYPTO}) (SECRET, {NUCLEAR}). A more generalized hierarchy of security classes (or levels) with a mathematical basis was presented by Bell and La Padula in 1973 [8].

The National Computer Security Center (NCSC), later DOD (Department of Defense) published an official standard called "Trusted Computer System Evaluation Criteria" [1], universally known as "the Orange Book" to direct computer security safeguards to defend classified information in remote access, remote sharing computer systems. The Orange Book defines fundamental security requirements for computer systems and specifies a series of criteria for various levels of security ratings of a computer system based on its system design and security feature [11].

IV. Requirements of Secure Operating Systems

In most operating systems, either all of the privileges are granted, or none of the privileges are granted. This is the one shot approach of access control and is due to the lack of built-in mechanisms for the implementation of security policies. The perception that the users and the programs that they work upon are the good guys could be very dangerous. They can no longer be deemed safe with internet connectivity. The information needs to be restricted within a "security perimeter" with strict rules enforced by the system about who is permitted access to specific resources [11]. Also information should not be allowed to move from a more secure environment to a less secure one.

Some of the basic requirements of an operating system are mandatory security, support of diverse security policies and assurance.

- Mandatory security – It is a built-in mechanism or logic within the operating system (often called system security module or system security administrator) that implements and tightly controls the definition and assignment of security attributes and their actions (security policies) for every operation or function provided by the system [11].
- Support of diverse security policies – A traditional mac mechanisms (such as the multi-level security – mls [8]) usually base its security decisions strictly on security clearances for subjects and security labels for object, and are normally too restricted to serve as a general security solution [11].
- Assurance – A process or methodology to verify the design and implementation of the system that should actually behave as it claims to be and meet the security requirements [11].

V. A Case Study of SE-Linux

In this section, NSA's SE-Linux is discussed as a case study of the recent efforts in the development of secure operating systems [6].

Background

National Security Agency (NSA) which is the ultimate gatekeeper of information security and assurance within USA, has been involved in determining security criteria/requirements for information systems. The development of SE-Linux is indeed the results of several previous projects of NSA, especially the dtos and flask [3, 4].

An important attribute of SE-Linux release is that it follows the same Open Source Initiative as that of the Linux. All documentation and source code of SE-Linux are publicly available at NSA Web site [6] under the same terms and conditions of Linux. This is in hope to reach a wide audience and to encourage further efforts and research of secure operating systems.

Architecture

The SE-Linux is an adoption of the Flask security architecture in Linux operating system. The integration of the security architecture with Linux is accomplished in a way that a new kernel module, called the Security Server (SS) that implements the security policy decision logic, is added into a non-security- enhanced Linux (hereafter as ordinary Linux) that is patched with LSM (Linux Security Module) [11-13] for maintaining security attributes in kernel data structures and for the mechanism of security control enforcement. Security contexts are not directly bound to objects in the system. Instead, each object that requires a security label is assigned with a security identifier (SID) that is mapped to a security context. This mapping is maintained by SS at run time.

An identity is given to every subject (process) of the system. This comes from a user when the user logs on to the system (this identity is orthogonal to Linux UID, and will remain unchanged even after a process changes its UID). A set of roles can be defined in security policies for individual users that may be entered by processes with the given user's identity. Each role is specified by a security policy for allowable actions whenever a subject assumes the role (role-based access control RBAC). However, different from the typical RBAC in which permissions are directly granted to roles, type enforcement (TE) is used with roles for fine- grained access controls in SE-Linux.

Security policies are specified in text-based policy configuration files using a simple language developed for SS. The policy configuration for a specific installation of SE-Linux is checked and compiled into binary and loaded at boot time into SS (if allowed by the policy, it may also be reloaded at runtime)[11].

VI. Conclusion

With the ever growing security alerts, a better way to address the root causes of vulnerabilities in the operating systems should be explored. The methods discussed in this article – executing applications from a strongly guarded, secure operating system – can provide a frontier in battling with many of existing cyber-space threats of the real world

Although, not all the dangers of current cyber space may be eradicated and the security of individual applications may still suffer from the vulnerabilities of their own with these techniques, with a secure operating system, the damages and the impacts among various applications can be controlled.

References

[1] DOD 5200.28-STD, "DOD Trusted Computer System Evaluation Criteria" (Orange Book), 26 December 1985, http://www.radium.ncsc.mil/tpep/library/rainbow/5200.28-STD.pdf..

[2] DOD 5200.28-STD, "DOD Trusted Computer System Evaluation Criteria" (Orange Book), 26 December 1985, http://www.radium.ncsc.mil/tpep/library/rainbow/5200.28-STD.pdf..

[3] "Flask: Flux Advanced security Kernel", http://www.cs.utah.edu/flux/fluke/html/flask.html.

[4] DTOS Technical Reports, http://www.securecomputing.com/randt/HTML/technical-docs.html.

[5] Chris Dalton and Tse Huong Choo, "An Operating System Approach to Securing E-Services", *Communications of the ACM*, V. 44, No. 2, p. 58, 2001.

[6] Charlie Kaufman, Radia Perlman, and Mike Speciner, "Network Security: Private Communication in a Public World", *PTR Prentice Hall*, Englewood Cliffs, New Jersey, 1995.

[7] D.E. Bell and L. J. La Padula, "Secure Computer Systems: Mathematical Foundations and Model", *Technical Report M74-244, The MITRE Corporation*, Bedford, MA, May 1973.

[8] Ames, Stanley R., Jr., and J.G. Keeton-Williams, "Demonstrating security for trusted applications on a security kernel base", *IEEE Comp. Soc. Proc. 1980 Symp. Security and Privacy*, April 1980.

[9] Stephen Smalley and Timothy Fraser, "A Security Policy Configuration for the Security-Enhanced Linux", http://www.nsa.gov/selinux/doc/policy.pdf..

[10] H. Chen, P. Belhumeur, and D. Jacobs. In search of illumination invariants. In *IEEE Conf. on Comp. Vision and Patt. Recog.*, pages 254–261, 2000 Linux Security Modules: General Security Hooks for Linux, http://lsm.immunix.org/docs/overview/linuxsecuritymodule.html.

[11] Global Information Assurance Certification Paper, SANS University