iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 2 **Issue:** X **Month of publication:** October 2014

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

Securing Operating System

Sahil Jindal[1], Puneet Gulati[2], Praveen Rohilla[3]
[1,2,3] CSE, Dronacharya College of Engineering, Gurgaon, India

Abstract: One of the fundamental concerns in the security of cyberspace and e-commerce is the security of operating systems that are the core piece of software running in all information systems, such as network devices (routers, firewalls, etc), Web servers, customer desktops, PDAs, and so on. Many of known vulnerabilities discovered so far are rooted from the bugs or deficiency of underneath operating systems.

This paper discusses the security (or lack of security) of most commercial operating systems like Unix and Microsoft Windows, and its effect to the overall security of Web based applications and services. Based on DOD's trusted computer system model, the current effort toward development of secure operating systems is presented, and as a case study, the publicly available security enhanced Linux, SE-Linux, is also analysed.

I. INTRODUCTION

Every modern computer system, from network servers, workstation desktops, to laptops and hand-held devices, has a core piece of software, called kernel or operating system, executed on the top of a bare machine of hardware that allocates the basic resources of the system (e.g., CPU, memory, device driver, communication port, etc), and supervises the execution of all applications within the system. Some popular commercial and Open Source operating systems are Microsoft Windows, different flavours of Unix (BSD, AIX, HP-UX, Solaris, etc), Mac OS, and Linux.Because of the crucial role of the operating system in the operation of any computer systems, the security (or lack of security) of an operation system will have fundamental impacts to the overall security of a computer system, including the security of all applications running within the system. A compromise of the underneath operating system will certainly expose danger to any application running in the system. 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.

With the ever-growing connectivity and E-commerce through the Internet, application security is an ultimate goal for millions of merchants and consumers who turn their business and service electronic and to the public world of cyberspace. On the other hand, efforts to achieve total security of such systems continue to be based on the flawed promise that adequate security can be achieved in applications with the current security mechanisms of mainstream operating system. The reality is that secure applications demand secure operating systems, and tackling application compromises at the OS level by kernel-enforced controls should probably be considered as an attractive and effective approach.

In order to raise the security level of operating systems to next class – B class, the requirement of Mandatory Access Control (MAC) is a necessity. A typical MAC architecture needs the ability to enforce an administratively set security policy over all subjects and objects (users, processes, memory, files, devices, ports, etc) in the system, basing decisions on labels containing a variety of security-relevant information. MAC provides strong separation (or containment) of applications that permits the safe execution of untrustworthy applications, and enables critical processing pipelines (trusted path) to be established and guaranteed. Therefore, it offers critical support for application security by protecting against the tampering with, and bypassing of, secured applications. The benefits derived from MAC would never be possible with the existing DAC Operating systems.

II. SECURITY OF OPERATING SYSTEMS

Most modern information computer systems provide concurrent execution of multiple applications in a single physical computing hardware (which may contain multiple processing units). Within such a multitasking, time-sharing environment, individual application jobs share the same resources of the system, e.g., CPU, memory, disk, and I/O devices, under the control of the operating system. 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. An application is controlled that only given resources (e.g., file, process, I/O, IPC) it can access, and given operations (e.g., execution or read-only) it can perform.

However, the limited containment supported by most commercial operating systems (MS Windows, various flavours of Unix, etc) bases access decisions only on user identity and ownership without considering additional security-relevant criteria

such as the operation and trustworthiness of programs, the role of the user, and the sensitivity or integrity of the data. As long as users or applications have complete discretion over objects, it will not be possible to control data flows or enforce a system-wide security policy. Because of such weakness of current operating systems, it is rather easy to breach the security of an entire system once an application has been compromised, e.g., by a buffer overflow attack. Some examples of potential exploits from a compromised application are:

A. 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.

B. 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.

C. 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.

D. 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.

It is not possible to protect against malicious code of an application using existing mechanisms of most commercial operating systems because a program running under the name of a user receives all of the privileges associated with that user. Moreover, the access controls supported by the operating systems are so coarse only two categories of users: either completely trusted super users (root) or completely un-trusted ordinary users. As the result, most system services and privileged applications in such systems have to run under root privileges that far exceed what they really needed. A compromise in any of these programs would be exploited to obtain complete system control.

III. REQUIREMENTS OF SECURE OPERATION SYSTEMS

Most current operating systems provide discretionary access control, that is, someone who owns a resource can make a decision as to who is allowed to use (access) the resource. Moreover, because the lack of built-in mechanisms for the enforcement of security policies in such systems, the access control is normally a one-shot approach: either all or none privileges are granted, rarely supporting the "principle of least privilege" (without limiting the privileges a program can inherit based on the trustworthiness).

The basic philosophy of discretionary controls assumes that the users and the programs they run are the good guys, and it is up to the operating system to trust them and protect each user from outsiders and other users. Such perception could be extremely difficult to hold true and no longer be considered as secure enough for computer systems of "information era" with broad connectivity through the Internet and heavily commercialization of e-commerce services. Systems with stronger security and protection will require evolving from the approach of discretionary control towards the concept of mandatory (non-discretionary) control where information is confined within a "security perimeter" with strict rules enforced by the system about who is allowed access to certain resources, and not allow any information to move from a more secure environment to a less secure environment.

Some of basic criteria or requirements of a secure operating system are discussed below.

A. Mandatory security –

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. Generally, a mandatory security will require:

1. A policy independent security labelling and decision makes logics. The operating system implements the mechanism, whereas the users or applications are able to define security policies.
2. Enforcement of access control for all operations. All system operations must have permission checks based on security labelling of the source and target objects. Such enforcement requires controlling the propagation of access rights, enforcing fine-grained access rights and supporting the revocation of previously granted access rights, etc.
3. The main security controls include permission or access authorization, authentication usage, cryptographic usage, and subsystem specific usage, etc.

B. Trusted path

A mechanism by which a trustworthiness relationship is established among users and application software so that:

1. A user or application may directly interact with trusted software, which can only be activated by either user or trusted software
2. Mutually authenticated channel is needed to prevent impersonation of either party.
3. The mechanism must be extensible to support subsequent addition of trusted applications.

C. Support of diverse security policies

Traditional MAC mechanisms (such as the multi-level security – MLS) are usually based its security decisions strictly on security clearances for subjects and security labels for objects, and are normally too restricted to serve as a general security solution. A secure architecture requires flexibility for support of a wide variety of security policies:

1. Separation of security policy logic from the mechanism of policy enforcement, so that a system can support diverse security policies.
2. Support for policy definition and policy changes with well-defined policy interfaces and formats.
3. Provider of default security behaviour of the system so that to maintain tight system security without requiring detailed system configuration.

D. 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:

1. The process generally involves two elements, (i) statement of the security properties a system is claimed to satisfy; and (ii) some kind of argument or evidence that the system does satisfy those properties.
2. The structure of such systems normally requires a small security kernel or module so that the system behavior would relatively easy be verified.
3. One of the concerns for a secure operating system is the so-called covert channels, which are the means to circumvent the security barrier enforced by the system in prevention of passing information from one security domain to a less secure domain. For example, one possible covert channel is a “timing channel”, where a Trojan horse program alternately loops and waits, in cycles of, say one minute per bit, and a program outside the perimeter that constantly tests the loading of the system may sense the information the Trojan horse intended to send. There is no general way to prevent all covert channels. It is more practical to introduce enough noise or reduce the bandwidth of such channels in the system so that they won’t be useful to an intruder.

The efforts for the development of secure operating systems can be dated to the earlier days of operating system development. With the rapid growth of Internet connectivity and e-commerce, recent development of secure operating systems spreads from traditional focus of defence or military related systems to more general commercial systems. As a case study, next section presents detailed discussions of a publicly available
Secure system from National Security Agency (NSA).

IV. CONCLUSIONS

With ever growing security alerts and CERT Advisory for systems like Microsoft Windows and the ordinary Linux, people must be wandering how such games of cat-mouse-catching would ever be ended, and if there could be any better way to address the root causes of many of general vulnerabilities of information systems. The approach covered in this article – executing applications from a strongly guarded, secure operating system – certainly opens an alternative
frontier in battling with many of existing cyber-space threats of the real world.

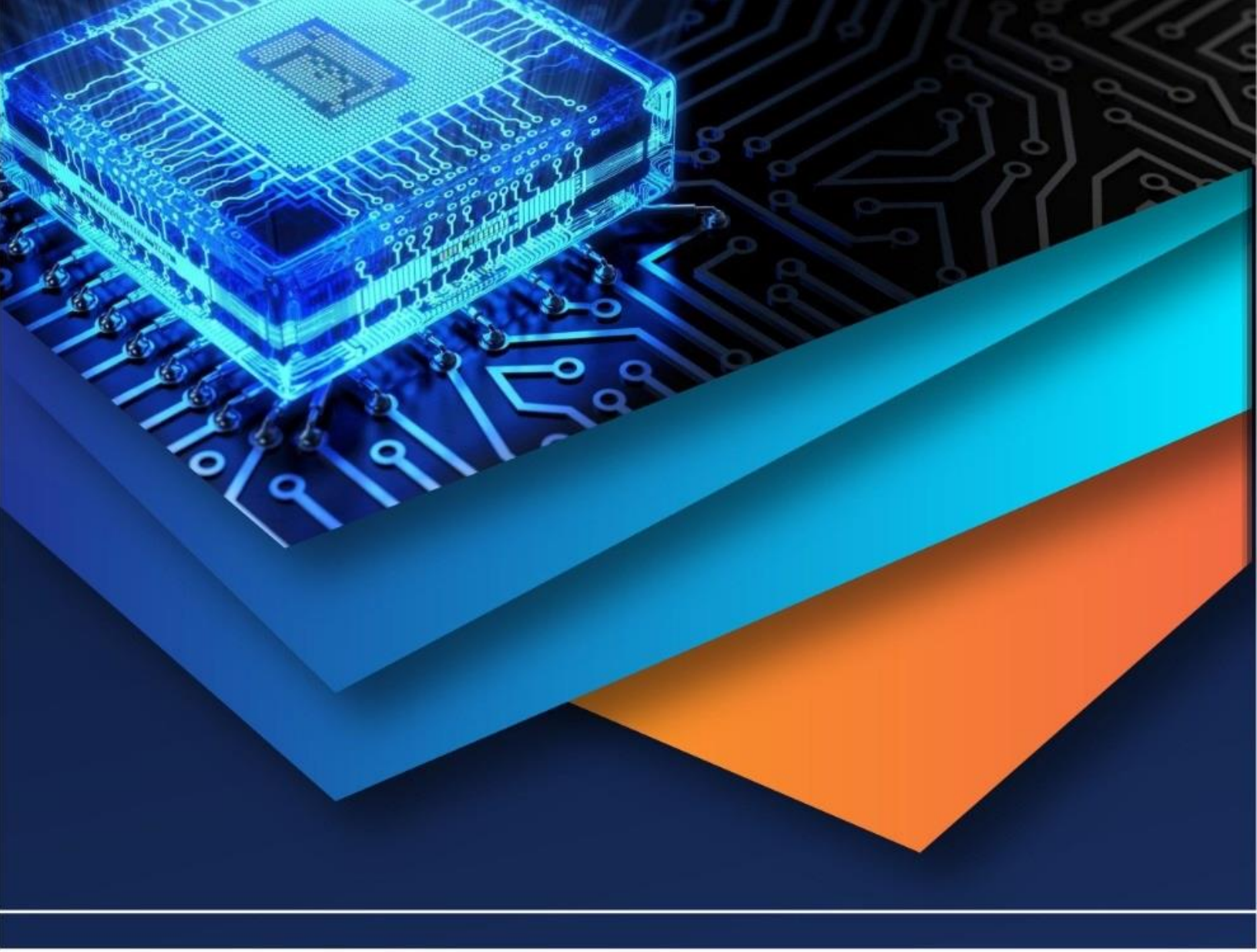
Although, the approach of using secure operating systems will not be a panacea for all the dangers of current cyber space, and the security of individual applications may still suffer from the vulnerabilities of their own, with the strong containment of a secure operation system, the damages caused from a compromise within one application would be much localized, and the impacts among various applications could be much well controlled.

As a demonstration of how mandatory access control can be integrated into a popular, main-stream operating system, the release of SE-Linux to general public assures that the usage of secure operating systems is not necessarily an expensive endeavour limited only to academic and defence related institutions, and encourages further efforts in research and development

of secure operating systems. Not much testing result has been reported regarding the performance impacts and effectiveness of MAC of SE-Linux. It would be interesting to see some experimental deployment and test results using SE-Linux with real-world applications, such as Web servers for e-commerce services.

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. 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", *Proceedings of the 21st National Information Systems Security Conference*, pages 303- 314, Oct. 1998, http://www.nsa.gov/selinux/doc/inevitability.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. Security Enhanced Linux, http://www.nsa.gov/selinux/index.html.
7. Charlie Kaufman, Radia Perlman, and Mike Speciner, "Network Security: Private Communication in a Public World", *PTR Prentice Hall*, Englewood Cliffs, New Jersey, 1995.
8. 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.
9. 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.
10.Wulf, W. A., Levin, R., and Harbison, S. P., "HYDRA/C.mmp: An Experimental Computer System", *McGraw-Hill*, New York, 1981.
11.Peter A. Loscocco and Stephen D. Smalley, "Meeting Critical Security Objectives with Security-Enhanced Linux", http://www.nsa.gov/selinux/doc/ottawa01.pdf.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY