International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE)
ISSN: 0976-1353 Volume 11 Issue 1 –NOVEMBER 2014.

Operating Systems and Security

P Ravindra

Lecturer, Dept of Computer Sci., KBN College,Vijayawada, A.P, India

ponugumatiravindra@gmail.com

***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 flavors of Unix (eg:Solaris), 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.

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 flavors 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

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 direc tory in Microsoft IIS

Gain direct access to protected system resources by misusing privileges. For example, a compromised "sendmail" 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.

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.

What are the operating system's role os internal features: privileged mode, memory protection, files access permissions, etc. Internal features protect the operating system against users this necessary but not sufficient and File permissions protect users (and the OS) against other user again, this is necessary but not sufficient. File permissions are based on user identity,which is based on authentication.

How does an OS authenticate users
something you know
something you have
something you are
something you know :

easy or commom passwords
hashed passwords
challenge and response authentication

something you have :

cryptographic secret
PIN numbers

something you are:

Fingerprint readers
Iris scans

Biometrics generally have a fale positive rate and a false negative rate — be careful how you set your parameters. Biometrics works best if stored locally — over the network, all that's seen is a string of bits.

Attacks and Defences:Attacking techniques:
Trojan horses — "come and get it" attack
Login spoofing
Buggy software — the big one

Trojan horses: Trick someone into executing a program that does nasty things (Many viruses and worms spread that way) Unix-type file permissions don't help — the attack program can change permissions Need mandatory access control (MAC)

A better idea is for the OS to provide sandboxes —" an environment where the program can execute but can't affect the rest of the machine.

Strong isolation is conceptually pretty easy — run the program on a separate machine, or under VMware or the like

Login Spoofing: Is that the real login prompt? A fake one could capture your login and password(We see similar things today with fake ATMs!) A trusted path is a user-initiated sequence that is guaranteed to get you to the real OS.

Example: ctrl+alt+delete on Windows

Well, it was supposed to be one. But you have to train people not to log in unless they've initiated the sequence. Must protect all password prompts that way

III. Viruses and Worms:

Viruses spread by themselves within a machine, but require human intervention to infect other machines. Worms spread between machines, though they may require human assistance (i.e., opening an attachment) to infect another machine

One sometimes hears that "Windows is infested with these things because it has no (effective) file protection" .File protection would prevent OS contamination, but worms can and do spread with user permissions. The IBM Christmas Card "Virus" (1987) relied on a Trojan Horse emailed shell script. The Morris Internet Worm (1988) was multi-exploit, multi-platform and didn't violate any OS protections.

Applications
The real purpose of an operating system is to run certain applications
The issue isn't how secure the OS is, it's how secure the applications are
Again, most worms don't violate OS security

The Challenge
A useful secure OS should make it easier to write secure applications
That means things like useful sandboxes
Need more flexible permission model; DAC is too simple and MAC is too restrictive
In my opinion, no commercial OS satisfies these criteria Of course, most applications are designed for the facilities we do have
We'll always have buggy code; the trick is to build an application and an OS that will mostly resist attack and will protect the important assets