Process-Based Security™

First Level Descriptive Document

SAGE, Incorporated
112 West 8th Avenue, Suite 402
Amarillo, TX 79101
(800) 580-0025
www.sage-inc.com

Table of Contents

Table of Figures

Executive Overview

Operating systems employing user based access methods are by far the most common in use today. The concept of binding a user's credentials (typically, user name plus password) to a program has roots that go back to the 1960's, a period when all computer systems operated in a trusted environment. Malicious hacking was essentially non-existent. As computers and systems have grown more powerful and sophisticated, operating systems have also grown in size and sophistication, but have not adapted to the new, intruder-rich environment of today's Internet.

Process-based operating systems are designed from the ground up to stop today's sophisticated attackers. They use an entirely different method of establishing program permissions for execution. In a Process-Based Security (PBS) system individual users have no system level access rights, only application level rights. Only the programs (processes) themselves have system level rights and these rights are pre-determined when the system is initially built. System privileges cannot be changed in the field, thus preventing the execution of malicious programs.

Introduction

This paper will discuss the operating system differences between user-based program access and process-based access. We will discuss the principal design elements of a process-based system and how they work in a web server application, applying a rigorously enforced security policy to create a secure web server. Mandatory access controls and rules of least privilege are strictly enforced from the factory through the use of a hardened O/S along with specially tailored applications for http, POP3, SMTP, and FTP services. By hard-coding security policy into the overall system, system administrator errors are eliminated, thus preventing malware execution, or exploits allowed by human error. A side benefit is the *complete elimination* of continual security updating and patching of the operating system.

User-based Security vs. Process-Based Security

A web server must make it possible for millions of unknown people to connect to it and access data. For each user, it performs essentially one task: to serve up web pages. In this environment, the task of the system security is to ensure that the user does not find a way to force the web server to run harmful processes.

The most popular security model for web servers is the user-based security model, also known as discretionary access control (DAC), which functions on user authentication. Processes are assigned to a specific user identity and thus their actions are restricted based on the user's access rights. However, this user-based security model has proven unreliable in recent years.

The main reason for its failure is that it relies heavily on the assumption that a valid user will not engage in any malicious activity. It is processes, however, and not users that compromise or damage computers. Users don't delete files, processes do. If a rogue process can masquerade itself as having legitimate rights of access, then the rogue process could execute at a privileged level and cause damage within the system.

Examples of flaws caused by user-based security:

- Web Server flaws
- CGI / ASP / SSI exploits
- Memory leaks (Buffer overflows)

What is PBS ?

SAGE Inc.'s Process-Based Security™ (PBS) model is a hardened, Linux-based security paradigm. The PBS operating system is not concerned with who initiated the requesting program, only what the program wants. The program is permitted to utilize only authorized resources. It is a default-deny model to operating systems providing resources only to the pre-authorized applications. In other words, it is a default-deny allowing only what the owner of the system wants to security model.

For a more formal definition one could say PBS is a security model replacing user-based access (DAC) with process-based access (mandatory access controls, or MAC), invoking rules of least privilege and separation of duties.

When the system administrator in a DAC-based system loads a new program onto the system, the needs of the program are determined and the administrator sets the authorized (user-based) access profile. In effect, the system administrator is mapping the locally established access rights policy to the system. The program is now associated with a user and that user has specific rights that ultimately determine the program's level of access. If the user has been spoofed, or if a program's access privileges have been hijacked, the rogue program can now roam the system at will.

In a PBS system all system level policies are fixed and cannot be changed in the field. As a program operates and requests access to system resources, the program's authorized process profile checks to determine resource accessibility. This is accomplished by using a fine-grained access control table (ACL file, or access rights table) protected within the system. It maintains an authorized process profile for all known processes requiring access to system resources. This table has no entries for unauthorized or unknown processes.

The problem posed by viruses is an example of PBS capabilities, which are twofold. First, a virus unknown to the system would not have an authorized process profile. The virus is not allowed access to any files or I/O ports, keeping the virus from running or infecting the computer or other computers. Secondly, if a friendly program being loaded

by the administrator had been somehow previously infected, the extent of damage is minimized based on assigned process profiles.

Design Principles

How does PBS work ?

When a process runs in a DAC-based operating system, it invokes several system calls. These system calls can retrieve as well as alter user data and kernel data. Hence they are the main points of risk for system security. If the process is restricted in terms of its access of the system calls, damage can be prevented. In the PBS operating system, there exists a "PBS-specific access right" related to every system call in the kernel. These access rights are in addition to the standard rights defined in Linux operating system. Thus, a process can access the system call only if it has the particular PBS-specific access right.

Since the security is implemented at the kernel level, it is impossible for a user, even a trusted user, to compromise the system. Even if a user attains primary administrator access rights by social engineering or some other means, any system damage will be limited to that one particular process and then only from an application level, not a system level.

In PBS, the only point of risk that remains to be considered is how the access rights are assigned to the process. PBS has a set number of processes which can run in the system, each of which has a predefined set of access rights listed in a file containing the access control list (ACL file). When installing a program onto the system a Master Administrator (defined as the person who builds the original system image), determines the PBS-specific access rights needed for the execution of the program. An entry is accordingly made in the ACL file. This file contains the list of programs that can be safely executed in the system and the PBS-specific access rights required for each program. Thus, when a program starts running in the PBS system, a check is made to verify if it has the access rights specified in the ACL file. Additionally, user-level authentication is implemented for another layer of security.

Due to this restrictive nature of the PBS operating system, the number of processes running at any time is limited to only those that are absolutely needed for the web server or mail server or any other service running in the PBS environment. Although this reduces the flexibility, it increases the security by limiting the potential points of failure.

Run Time Loading of a Process into the PBS System

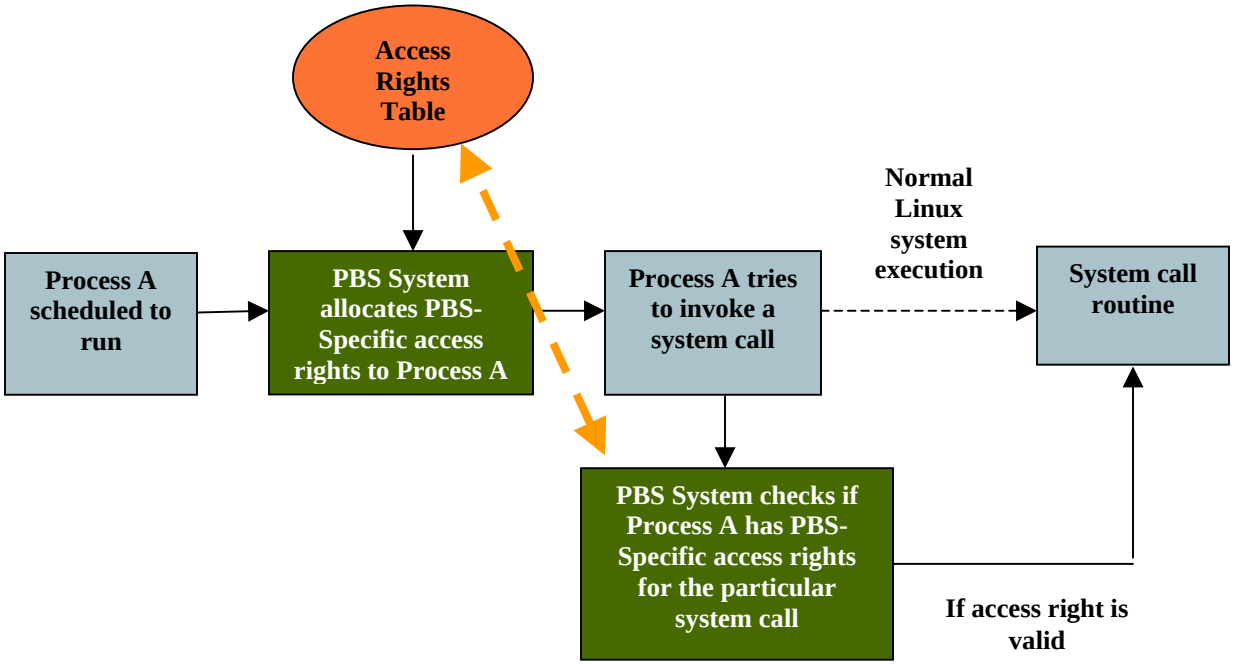

Fig 1: Run Time Loading of a Process into the PBS System

System call Access Verification Process

When a process in the PBS system tries to invoke a system call, an access verification routine is followed before granting access for the process to the system call.

There are certain system calls that require a process to perform an action (e.g. Reboot, Set Time). In such cases, PBS checks if the calling process has appropriate PBS-specific access rights to perform that action.

Other system calls require the process to perform an operation on a file (e.g., Create, Delete, Write files or Mount). In these cases PBS performs several tests. First, it checks the validity of the file name and path name, and whether it already exists. It also checks whether the process has the appropriate access rights for that particular file. For example, if a process issues the system call **mount(fd0, "/home/user"),** PBS first checks if fd0 is a valid device and if "/home/user" is a valid path. Second, it checks if the process has appropriate mount access rights **PBS_MOUNT_DEV** for the device "fd0" and **PBS_MOUNT_PATH** for the path "/home/user." If any of these tests fail, the system call is disallowed. Thus, the access to any system resource is highly restricted and

guarded. Because of the heavily layered security built into PBS it is essentially impossible for an attacker to cause damage.

Hardening the System Call Interface

File Operations

All file operations have a standard PBS check for the validity of the file. If a file with that name does not exist, the system call is disallowed.

File Mode

With conventional security, the mode of any file can be changed using the **chmod** command by root or the file owner. A malicious user can do harm if a harmful file is given execute permission and executed in the system. PBS does not allow changing the mode of a file to *execute* unless the process possesses the PBS-specific change mode access right (**PBS_CHMOD**).

File Deletion

In normal Linux systems the deletion of a file is quite a simple process. The only checks made by the operating system are for the permission bits and owner of the file and its directory. If the ancestor directory permits its children to be changed by any user or group, then any of the directory's descendents can be deleted by that user or group. In contrast, the PBS system deletion of a file has far more rigid requirements. A process can delete a file only if it is required to do so. We might also say the process has a PBS-specific delete access right for that particular file (**PBS_DELETE_FILE**).

System V IPC

System V Interprocess Communication (IPC) objects can be of three kinds: System V message queues, semaphore sets, and shared memory segments. When accessing such objects, the rules of the normal security system are as follows:

- if a process has root privileges, access is always granted (DANGEROUS!)
- if the process' EUID is the owner or creator UID of the object, then the creator permission bit is checked to see if access can be granted.
- if the process' EGID is the owner or creator GID of the object, or one of the process' groups is the owning or creating GID of the object, then the creator group permission bit is checked to see if access can be granted.

However, in the PBS system even more restrictions are enforced on the IPC message system.

- If a process is trying to create a new message queue using the msgget() system call, it is denied access.

- If a process is trying to send a message using the msgsnd() system call, it requires PBS-specific write file access (**PBS_WRITE_FILE**).
- If a process is trying to receive message using the msgrcv() system call, it requires PBS-specific read file access (**PBS_READ_FILE**).
- If a process invoked the message control system call msgctl(), in order to access the IPC_SET option, it should possess PBS-specific access right for queue resizing (**PBS_QUEUE_RESIZE**) and for IPC_RMID option, it should possess PBS-specific access right for file delete (**PBS_DELETE_FILE**).

Signals

A process can set a signal on another process (by using the **kill** system call). The other process would receive and handle the signal asynchronously. In order for a process to send a signal to any arbitrary process, it should either have root privileges, or the effective (or real) user ID of the sending process must equal the real or set user ID of the receiving process. PBS places another check in the kill system call. Further, PBS does not allow sending signals to the init process (process with PID 1), or the current process.

Process Trace

The **ptrace** system call can be used for tracing the progress of a process. PBS does not allow ptrace-ing the current process. Also, for any process to ptrace another process, it should possess (**PBS_TRACE**) access right.

Summary

Clearly, PBS has specific security advantages over user-based systems in today's Internet environment. Malicious programs like worms and Trojan horses are the scourge of today's Internet. By limiting system access to known (trusted) processes, and by limiting the scope of that access to only what is necessary for the process to function, PBS can stop the execution of malicious programs. And since the PBS security model has all program access pre-determined, the need for constant security patching is eliminated.

Trojan horses, viruses and worms cannot function in a PBS environment. Further, spammers cannot take control of an SMTP mail server, nor can the system be used as a "zombie" for distributed denial of service attacks.

SAGE has proven the PBS operating system principles in a secure web appliance, BRICKServer®2. It includes a web server, SMTP and POP3 mail, plus ftp services. The system has withstood the rigors of penetration testing by Sandia National Laboratories, the U.S. Army, various system integrators, and countless hackers since it's availability.

The PBS operating system is available for any application or device with an IP address.