Linux Network Security Access & Monitoring Service Tools

Mahesh Kumar [1]
[1] Department of Computer Science & Engineering,
Ganga Institute of Technology and Management,
Kablana, Jhajjar, Haryana, INDIA

***Abstract*--It has been recognized that securing applications is only half of the battle: a computer system must also employ security policies at the OS level, and the current model of user vs. administrator that we find in standard Unix is insufficient. For the basic security features, Linux has password authentication, file system discretionary access control, and security auditing. These three fundamental features are necessary to achieve a security evaluation at the C2 level [4]. Most commercial server-level operating systems, including AIX (IBM), Windows NT, and Solaris, have been certified to this C2 level. Generally speaking workstations/servers are used by people that don't really care about the underlying technology, they just want to get their work done and retrieve their email in a timely fashion. There are however many users that will have the ability to modify their workstation, for better or worse (install packet sniffers, warez ftp sites, www servers, irc bots, etc). To add to this most users have physical access to their workstations, meaning you really have to lock them down if you want to do it right.**

***Keywords*:- *Security, Linux OS*, Server, Security, monitoring tool**

I. INTRODUCTION

By expanding the basic standard security features we have:

- User and group separation
- File system security
- Audit trails
- PAM authentication

A. User and Group Separation

User accounts are used to verify the identity of the person using a computer system. By checking the identity of a user through username and password credentials, the system is able to determine if the user is permitted to log into the system and, if so, which resources the user is allowed to access.

Groups are logical constructs that can be used to group user accounts together for a particular purpose. For example, if a company has a group of system administrators, they can all be placed in a system administrator group with permission to access key resources of the OS. In addition, through group creation and assignment of privileges, access to restricted resources can be controlled for those who need them and denied to others.

The ability for a user to access a machine is determined by whether or not that user's account exists. Access to an application or file is granted based on the permission settings for the file. This helps to ensure the integrity of sensitive information and key resources against accidental or purposeful damage by users.

After a normal user account is created, the user can log into the system and access any applications or files they are permitted to access. Linux determines whether or not a user or group can access these resources based on the permissions assigned to them.

There are three permissions for files, directories, and applications. Table 1 lists the symbols used to indicate each of them. Each of the three permissions is assigned to three defined categories of users. The categories are listed in Table 2.

TABLE 1.

Permission character symbols

Symbol	Description
r	Indicates that a given category of user can **read** a file.
w	Indicates that a given category of user can **write** to a file.
x	Indicates that a given category of user can **execute** the file.
-	A fourth symbol indicates that no access is permitted.

TABLE 2.

PERMISSION CATEGORIES

Category	Description
Owner	The owner of the file or application.
Group	The group that owns the file or application.
Everyone	All users with access to the system.

One can easily view the permissions for a file by invoking a long format listing using the command ls -l.

For instance, if the user kambing creates an executable file named foo, the output of the command ls -l foo would look something like this:

-rwxrwxr-x 1 kambing kambing 0 Sep 2 12:25 foo

The permissions for this file are listed at the start of the line, starting with set of rwx.

- This first set of symbols defines owner access.
- The next set of rwx symbols define group access,
- The last set of symbols defining access permitted for all other users.

This listing indicates that the file is readable, writable, and executable by the user who owns the file (user kambing) as well as the group owning the file (which is a group named kambing). The file is also world-readable and world-executable, but not world-writable.

II. FILE SYSTEM SECURITY

A very true statement of a UNIX/Linux system, everything is a file; if something is not a file, it is a process. Most files are just files, called regular files; they contain normal data, for example text files, executable files or programs, input to or output from a program and so on. While it is practically safe to say that everything you encounter on a Linux system is a file, there are some exceptions as listed below:

- Directories: files that are lists of other files.
- Special files: the mechanism used for input and output. Most special files are in /dev for example USB and CD-ROM.
- Links: a system to make a file or directory visible in multiple parts of the system's file tree. It is a shortcut.
- (Domain) sockets: a special file type, similar to TCP/IP sockets, providing inter-process networking protected by the file system's access control.
- Named pipes: act more or less like sockets and form a way for processes to communicate with each other, without using network socket semantics.

e following table gives an overview of the characters ermining the file type:

TABLE 3.

FILE TYPES CHARACTER SYMBOLS

Symbol	Meaning
-	Regular file
d	Directory
l	Link
c	Special file
s	Socket
p	Named pipe
b	Block device

On Linux system, every file is owned by a user and a group user. There is also a third category of users, those that are not the user owner and don't belong to the group owning the file. For each category of users, read, write and execute permissions can be granted or denied.

The long option to list files using the ls -l command, also displays file permissions for these three user categories; they are indicated by the nine characters that follow the first character, which is the file type indicator at the beginning of the file properties line. As seen in the following examples, the first three characters in this series of nine display access rights for the actual user that owns the file.

ls -l Mine

-rw-rw-r-- 1 mike users 5 Jul 15 12:39 Mine

ls -l /bin/ls

-rwxr-xr-x 1 root root 45948 Aug 10 15:01 /bin/ls*

The next three are for the group owner of the file, the last three for other users. The permissions are always in the same order: read, write, execute for the user, the group and the others. The first file is a regular file (first dash). Users with user name mike or users belonging to the group users can read and write (change/move/delete) the file, but they can't execute it (second and third dash). All other users are only allowed to read this file, but they can't write or execute it (fourth and fifth dash).

The second example is an executable file, the difference is everybody can run this program, but you need to be root to change it.

For easy use with commands, both access rights or modes and user groups have a code shown in Table 4 and 5.

TABLE 4.

ACCESS MODE CODES

Code	Meaning
0 or -	The access right that is supposed to be on this place is not granted.
4 or r	read access is granted to the user category defined in this place
2 or w	write permission is granted to the user category defined in this place
1 or x	execute permission is granted to the user category defined in this place

TABLE 5.

USER GROUP CODES

Code	Meaning
u	user permissions
g	group permissions
o	permissions for others

This straight forward scheme is applied very strictly, which allows a high level of security even without network security. Among other functions, the security scheme takes care of user access to programs; it can serve files on a need-to-know basis or least privilege and protect sensitive data such as home directories and system configuration files. We can use the chmod command to modify the file permission, changing of the access mode of a file. The chmod command can be used with alphanumeric or numeric options, whatever you like best. The following shows the examples.

>/hello

bash: ./hello: bad interpreter: Permission denied

>cat hello

#!/bin/bash

echo "Hello, World"

>ls -l hello

-rw-rw-r-- 1 mike mike 32 Jul 1 16:29 hello

>chmod u+x hello

>./hello

Hello, World

>ls -l hello

-rwxrw-r-- 1 mike mike 32 Jul 1 16:29 hello*

The + and - operators are used to grant or deny a given right to a given group. Combinations separated by commas are allowed. The following is another example, which makes the file from the previous example a private file to user mike:

>chmod u+rwx,go-rwx hello

>ls -l hello

-rwx------ 1 mike mike 32 Jan 15 16:29 hello*

If you encounter problems resulting in an error message saying that permission is denied, it is usually a problem with access rights in most cases.

When using chmod with numeric arguments, the values for each granted access right have to be counted together per group. Thus we get a 3-digit number, which is the symbolic value for the settings chmod has to make. The following table lists the most common combinations:

TABLE 5.

FILE PROTECTION WITH CHMOD

Command	Meaning
chmod 400 file	To protect a file against accidental overwriting.
chmod 500 directory	To protect you from accidentally removing, renaming or moving files from this directory.
Chmod 600 file	A private file only changeable by the user who entered this command.
chmod 644 file	A publicly readable file that can only be changed by the issuing user.
chmod 660 file	Users belonging to your group can change this file; others don't have any access to it at all.
chmod 700 file	Protects a file against any access from other users, while the issuing user still has full access.
chmod 755 directory	For files that should be readable and executable by others, but only changeable by the issuing user.
chmod 775 file	Standard file sharing mode for a group.
chmod 777 file	Everybody can do everything to this file.

If you enter a number with less than three digits as an argument to chmod, omitted characters are replaced with zeros starting from the left. There is actually a fourth digit on Linux systems that precedes the first three and sets special access modes.

A. The File Mask

When a new file is saved somewhere, it is first subjected to the standard security procedure. Files without permissions don't exist on Linux. The standard file permission is determined by the mask for new file creation. The value of this mask can be displayed using the umask command:

>umask

0002

Instead of adding the symbolic values to each other, as with chmod, for calculating the permission on a new file they need to be subtracted from the total possible access rights. In the example above, however, we see 4 digits displayed, yet there are only 3 permission categories: user, group and other. The first zero is part of the special file attributes settings. It might just as well be that this first zero is not displayed on your system when entering the umask command and that you only see 3 numbers representing the default file creation mask.

Each UNIX-like system has a system function for creating new files, which is called each time a user uses a program that creates new files, for instance, when downloading a file from the Internet, when saving a new text document. This function creates both new files and new directories. Full read, write and execute permission is granted to everybody when creating a new directory. When creating a new file, this function will grant read and write permissions for everybody, but set execute permissions to none for all user categories. In this case, before the mask is applied, a directory has permissions 777 or rwxrwxrwx, a plain file 666 or rw-rw-rw-.

The umask value is subtracted from these default permissions after the function has created the new file or directory. Thus, a directory will have permissions of 775 by default, a file 664, if the mask value is (0)002. This is demonstrated in the following examples:

>mkdir newdir

>ls -ld newdir

drwxrwxr-x 2 mike mike 2096 Jul 28 13:45 newdir/

>touch newfile

>ls -l newfile

-rw-rw-r-- 1 mike mike 0 Jul 28 13:52 newfile

A directory gets more permission by default, it always has the execute permission. If it wouldn't have that, it would not be accessible.

If you log in to another group using the newgrp command, the mask remains unchanged. Thus, if it is set to 002, files and directories that you create while being in the new group will also be accessible to the other members of that group; you don't have to use chmod. The root user usually has stricter default file creation permissions as shown below:

[root@tenouk root]# umask 022

These defaults are set system-wide in the shell resource configuration files, for instance /etc/bashrc or /etc/profile. You can change them in your own shell configuration file.

III. AUDIT TRAILS

Linux kernel 2.6 comes with auditd daemon. It's responsible for writing audit records to the disk. During startup, the rules in /etc/audit.rules are read by this daemon. You can open /etc/audit.rules file and make changes such as setup audit file log location and other option. The default file is good enough to get started with auditd. In order to use audit facility you need to use following utilities:TABLE 6.

AUDIT UTILITY

Utility	Description
auditctl	A command to assist controlling the kernel's audit system. You can get status, and add or delete rules into kernel audit system
ausearch	A command that can query the audit daemon logs based for events based on different search criteria.
aureport	A tool that produces summary reports of the audit system logs.

IV. Pluggable Authentication Modules authentication (PAM)

PAM [5] was invented by SUN Microsystems. Linux-PAM provides a flexible mechanism for authenticating users. It consists of a set of libraries that handle the authentication tasks of applications on the system. The library provides a stable general interface to which privilege-granting programs (such as login) defer to perform standard authentication tasks.

Historically, authentication of Linux users relied on the input of a password which was checked with the one stored in /etc/passwd. At each improvement (e.g. /etc/shadow, one-time passwords) each program (e.g. login, ftp) had to be rewritten. PAM is a more flexible user authentication mechanism. Programs supporting PAM must dynamically link themselves to the modules in charge of authentication. The administrator is in charge of the configuration and the attachment order of modules. All applications using PAM must have a configuration file in /etc/pam.d. Each file is composed of four columns:

TABLE 7.

PAM'S PAM.D CONTENT

Column	Description
Module type	▪ auth: user authentication ▪ account: user restriction (e.g.: hour restriction) ▪ session: tasks to perform at login and logout e.g.: mounting directories ▪ password: update of the user authentication token
success control	▪ required: a least one of the required modules ▪ requisite: all the requisite modules ▪ sufficient: only one sufficient module ▪ optional: a least one of the required modules is necessary if no other has succeeded
path to the module	Usually /lib/security.
optional arguments	-

Other PAM functionalities are listed in the following Table.

TABLE 8.

OTHER PAM FUNCTIONALITY

Functionality	Description
/etc/pam.d/other file	Provides default configuration for all modules not specified in the configuration file of the application.
pam_cracklib	Uses the cracklib library to check the "strength" of a password and to check it was not built based on the old one.
pam_limits	This module can restrict, depending on the user and/or group, the number of simultaneous processes, CPU time, the number of files simultaneously opened, their size, and the maximum number of simultaneous connections. The configuration file is: /etc/security/limits.conf
pam_rootok	Enables root to access a service without using his password. To be used with chfn or chsh and not with login.
pam_time	Control the access time. The configuration file is: /etc/security/time.conf.
pam_wheel	Allow access to root only to users of the wheel group. For use with su.
pam_cap	This module can force all privileges to a user.

Keep in mind that PAM however does not itself have an authenticated access to the kernel.

LINUX SECURITY EXTENSIONS

The Linux family of products has provided a highly secure environment since its original delivery in early 2002. The features discussed in the following sections have been added to the Linux OS. For example, the Red Hat Enterprise Linux Update 3, shipped in September 2004 contains:

- ExecShield [6] – With the **N**o e**X**ecute (NX) [7], [8], or e**X**ecute **D**isable (XD) and Segmentation features.
- Position Independent Executables (PIE) [9]

Then, in Red Hat Enterprise Linux v.4, shipped in February 2005 contains the following security features:

- SecurityEnhanced Linux (SELinux) [10]
- Compiler and library enhancements [11]
- Advanced glibc memory corruption checker [11]
- Secure version of the printf and other string manipulation functions. [11]

- gcc buffer bound checking [11]

In term of the Linux OS security breaches, most of the problems originated from the buffer overflow issue. The buffer overflow exploits unprotected and or unchecked fixed sized buffers, overwriting the area beyond it. The overwritten area may be filled with the malicious codes, containing code that pointing to the customized return address. There are many buffer locations in the memory area. It is used to temporarily store data.

A. *ExecShield*

The ExecShield supports two technologies that protect application from being compromised by most of the buffer exploit types. The goal of these features is to prevent code that is maliciously written in the data areas of an application from being executed. These NX/XD and Segmentation features use different techniques but to achieve the similar result. PAX [12], [13], [14] is similar, earlier technology that will not be discussed here.

1) *The NX/XD:*

The NX term is used by AMD for its Opteron/Athlon64 processors, while the XD is used by Intel for its Itanium2 and the x86/EM64T processors. These capabilities provides a new memory management feature that that allows individual pages of an application's memory to be marked as non executable. The problem is, previously the only level of control over memory pages was read and write. However, a page that was enabled for read could also be executed.

This meant that data areas such as the stack, heap and I/O buffers, which are typically only used for read/write could also be used to execute codes. It is a common form of exploit that involves writing code in a stack buffer and then executing it. So the ability to disable execution enhances the application and system security. The NX/XD support is available for most new processors including the recent model Intel x86 CPUs.

V. ADMINISTRATIVE MONITORING TOOLS ACCESS

A. *Telnet*

Telnet is by far the oldest and well known remote access tool, virtually ever Unix ships with it, and even systems such as NT support it. Telnet is really only useful if you can administer the system from a command prompt (something NT isn't so great at), which makes it perfect for Unix. Telnet is incredibly insecure, passwords and usernames as well as the session data flies around as plain text and is a favourite target for sniffers. Telnet comes with all Linux distributions. You should never ever use stock telnet to remotely administer a system.

B. *SSL Telnet*

SSL Telnet is telnet with the addition of SSL encryption which makes it much safer and far more secure. Using X.509 certificates (also referred to as personal certificates) you can easily administer remote systems. Unlike systems such as SSH, SSL Telnet is completely GNU and free for all use. You can get SSL Telnet server and client from: ftp://ftp.replay.com/.

C. *SSH*

SSH was originally free but is now under a commercial license, it does however have many features that make it worthwhile. It supports several forms of authentication (password, rhosts based, RSA keys), allows you to redirect ports, and easily configure which users are allowed to login using it. SSH is available from: ftp://ftp.replay.com/. If you are going to use it

commercially, or want the latest version you should head over to: http://www.ssh.fi/.

D. *LSH*

LSH is a free implementation of the SSH protocol, LSH is GNU licensed and is starting to look like the alternative (commercially speaking) to SSH (which is not free anymore). You can download it from: http://www.net.lut.ac.uk/psst/, please note it is under development.

E. *REXEC*

REXEC is one of the older remote UNIX utilities, it allows you to execute commands on a remote system, however it is seriously flawed in that it has no real security model. Security is achieved via the use of "rhosts" files, which specify which hosts/etc may run commands, this however is prone to spoofing and other forms of exploitation. You should never ever use stock REXEC to remotely administer a system.

F. *Slush*

Slush is based on OpenSSL and supports X.509 certificates currently, which for a large organization is a much better (and saner) bet then trying to remember several dozen passwords on various servers. Slush is GPL, but not finished yet (it implements most of the

required functionality to be useful, but has limits). On the other hand it is based completely in open source software making the possibilities of backdoors/etc remote. Ultimately it could replace SSH with something much nicer. You can get it from: http://violet.ibs.com.au/slush/.

G. *NSH*

NSH is a commercial product with all the bells and whistles (and I do mean all). It's got built in support for encryption, so it's relatively safe to use (I cannot really verify this as it isn't open source). Ease of use is high, you cd //computername and that 'logs' you into that

computer, you can then easily copy/modify/etc. files, run ps and get the process listing for that computer, etc. NSH also has a Perl module available, making scripting of commands pretty simple, and is ideal for administering many like systems (such as workstations). In addition to this NSH is available on multiple platforms (Linux, BSD, Irix, etc.). NSH is available from:

http://www.networkshell.com/, and 30 day evaluation versions are easily downloaded.

H. Fsh

Fsh is stands for "Fast remote command execution" and is similar in concept to rsh/rcp. It avoids the expense of constantly creating encrypted sessions by bring up an encrypted tunnel using ssh or lsh, and running all the commands over it. You can get it from:

http://www.lysator.liu.se/fsh/.

I. secsh

secsh (Secure Shell) provides another layer of login security, once you have logged in via ssh or SSL telnet you are prompted for another password, if you get it wrong secsh kills off the login attempt. You can get secsh at: http://www.leenux.com/scripts/.

J. YaST

YaST (Yet Another Setup Tool) is a rather nice command line graphical interface (very similar to scoadmin) that provides an easy interface to most administrative tasks. It does not however have any provisions for giving users limited access, so it is really only useful for cutting down on errors, and allowing new users to administer their systems. Another problem is unlike Linuxconf it is not network aware, meaning you must log into each system you want to manipulate.

K. sudo

Sudo gives a user setuid access to a program(s), and you can specify which host(s) they are allowed to login from (or not) and have sudo access (thus if someone breaks into an account, but you have it locked down damage is minimized). You can specify what user a command will run as, giving you a relatively fine degree of control. If granting users access be sure to

specify the hosts they are allowed to log in from and execute sudo, as well give the full pathnames to binaries, it can save you significant grief in the long run (i.e. if I give a user setuid access to "adduser", there is nothing to stop them editing their path statement, and copying "bash" into /tmp). This tool is very similar to super but with slightly less fine control.

Sudo is available for most distributions as a core package or a contributed package. Sudo is available at: http://www.courtesan.com/sudo/ just in case your distribution doesn't ship with it Sudo allows you to define groups of hosts, groups of commands, and groups of users, making long term administration simpler. Several /etc/sudoers examples:

Give the user 'seifried' full access seifried ALL=(ALL) ALL

Create a group of users, a group of hosts, and allow then to shutdown the server as root Host_Alias WORKSTATIONS=localhost, station1, station2

User_Alias SHUTDOWNUSERS=bob, mary, jane

Cmnd_Alias REBOOT=halt, reboot, sync

Runas_Alias REBOOTUSER=admin

SHUTDOWNUSERS WORKSTATIONS=(REBOOTUSER) REBOOT

L. Super

Super is one of the very few tools that can actually be used to give certain users (and groups) varied levels of access to system administration. In addition to this you can specify times and allow access to scripts, giving setuid access to even ordinary commands could have

unexpected consequences (any editor, any file manipulation tools like chown, chmod, even tools like lp could compromise parts of the system). Debian ships with super, and there are rpm's available in the contrib directory (buildhost is listed as "localhost", you might want to find the source and compile it yourself). This is a very powerful tool (it puts sudo to shame), but requires a significant amount of effort to implement properly, I think it is worth the effort

though. The head end distribution site for super is at: ftp://ftp.ucolick.org/pub/users/will/.

M. Remote

Webmin

Webmin is a (currently) a non commercial web based administrative tool. It's a set of perl scripts with a self contained www server that you access using a www browser, it has modules for most system administration functions, although some are a bit temperamental. One of my favourite features is the fact is that it holds it's own username and passwords for access to webmin, and you can customize what each user gets access to (i.e. user1 can administer users, user2 can reboot the server, and user3 can fiddle with the apache settings). Webmin is available at: http://www.webmin.com/.

N. Linuxconf

Linuxconf is a general purpose Linux administration tool that is usable from the command line, from within X, or via it's built in www server. It is my preferred tool for automated system administration (I primarily use it for doing strange network configurations), as it is relatively light from the command line (it is actually split up into several modules). From within X it provides an overall view of everything that

can be configured (PPP, users, disks, etc.). To use it via a www browser you must first run Linuxconf on the machine and add the host(s) or network(s) you want to allow to connect (Conf > Misc > Linuxconf network access), save changes and quit, then when you connect to the machine (by default Linuxconf runs on port 98) you must enter a username and password, it only accepts root as the account, and Linuxconf doesn't support any encryption, so I would have to recommend very strongly against using this feature across public networks. Linuxconf ships with RedHat Linux and is available at: http://www.solucorp.qc.ca/linuxconf/. Linuxconf also doesn't seem to ship with any man pages/etc, the help is contained internally which is slightly irritating.

O. COAS

The COAS project (Caldera Open Administration System) is a very ambitious project to

provide an open framework for administering systems, from a command line (with semi

graphical interface), from within X (using the qt widget set) to the web. It abstracts the actual

configuration data by providing a middle layer, thus making it suitable for use on disparate Linux platforms. Version 1.0 was just released, so it looks like Caldera is finally pushing ahead with it. The COAS site is at: http://www.coas.org/.

P. Log files and other forms of monitoring

One integral part of any UNIX system are the logging facilities. The majority of logging in Linux is provided by two main programs, sysklogd and klogd, the first providing logging services to programs and applications, the second providing logging capability to the Linux kernel. Klogd actually sends most messages to the syslogd facility but will on occasion pop up messages at the console (i.e. kernel panics). Sysklogd actually handles the task of processing most messages and sending them to the appropriate file or device, this is configured from within /etc/syslog.conf. By default most logging to files takes place in /var/log/, and generally speaking programs that handle their own logging (such as apache) log to /var/log/progname/, this centralizes the log files and makes it easier to place them on a separate partition (some attacks can fill your logs quite quickly, and a full / partition is no fun). Additionally there are programs that handle their own interval logging, one of the more interesting being the bash command shell. By default bash keeps a history file of commands executed in ~username/.bash_history, this file can make for extremely interesting reading, as oftentimes many admins will accidentally type their passwords in at the command line. Apache handles all of it's logging internally, configurable from httpd.conf and extremely

flexible with the release of Apache 1.3.6 (it supports conditional logging). Sendmail handles it's logging requirements via syslogd but also has the option (via the command line -X switch) of logging all SMTP transactions straight to a file. This is highly inadvisable as the file will grow enormous in a short span of time, but is useful for debugging. See the sections in network security on apache and sendmail for more information.

Q. sysklogd / klogd

In a nutshell klogd handles kernel messages, depending on your setup this can range from almost none to a great deal if for example you turn on process accounting. It then passes most messages to syslogd for actual handling, i.e. placement in a logfile. the man pages for sysklogd, klogd and syslog.conf are pretty good with clear examples. One exceedingly powerful and often overlooked ability of syslog is to log messages to a remote host running syslog. Since you can define multiple locations for syslog messages (i.e. send all kern messages to the /var/log/messages file, and to console, and to a remote host or multiple remote hosts) this allows you to centralize logging to a single host and easily check log files for security violations and other strangeness. There are several problems with syslogd and klogd however, the primary ones being the ease of which once an attacker has gained root access to deleting/modifying log files, there is no authentication built into the standard logging facilities.

The standard log files that are usually defined in syslog.conf are:

/var/log/messages

/var/log/secure

/var/log/maillog

/var/log/spooler

The first one (messages) gets the majority of information typically, user login's, TCP_WRAPPERS dumps information here, IP firewall packet logging typically dumps information here and so on. The second typically records entries for events like users

changing their UID/GID (via su, sudo, etc.), failed attempts when passwords are required and so on. The maillog file typically holds entries for every pop/imap connection (user login and 30 logout), and the header of each piece of email that goes in or out of the system (from whom, to where, msgid, status, and so on). The spooler file is not often used anymore as the number

of people running usenet or uucp has plummeted, uucp has been basically replaced with ftp and email, and most usenet servers are typically extremely powerful machines to handle a full, or even partial newsfeed, meaning there aren't many of them (typically one per ISP or more depending on size). Most home users and small/medium sized business will not (and should not in my opinion) run a usenet server, the amount of bandwidth and machine power required is phenomenal, let alone the security risks.

You can also define additional log files, for example you could add:

kern.* /var/log/kernel-log

And/or you can log to a separate log host:

*.emerg @syslog-host

mail.* @mail-log-host

Which would result in all kernel messages being logged to /var/log/kernel-log, this is useful on headless servers since by default kernel messages go to /dev/console (i.e. someone logged in at the machines). In the second case all emergency messages would be logged to the host "syslog-host", and all the mail log files would be sent to the "mail-log-host" server, allowing you to easily maintain centralized log files of various services.

R. *secure-syslog*

The major problem with syslog however is that tampering with log files is trivial. There is however a secure versions of syslogd, available at http://www.core-sdi.com/ssyslog/ (these

guys generally make good tools and have a good reputation, in any case it is open source

software for those of you truly paranoid). This allows you to cyrptographically sign logs and other ensure they haven't been tampered with, ultimately however an attacker can still delete the log files so it is a good idea to send them to another host, especially in the case of a firewall to prevent the hard drive being filled up.

next generation syslog Another alternative is "syslog-ng" (Next Generation Syslog), which seems much more customizable then either syslog or secure syslog, it supports digital signatures to prevent log tampering, and can filter based on content of the message, not just the facility it comes from or priority (something that is very useful for cutting down on volume). Syslog-ng is available at: http://www.balabit.hu/products/syslog-ng.html.

S. *Log monitoring*

1) Logcheck:

logcheck will go through the messages file (and others) on a regular basis (invoked via crontab usually) and email out a report of any suspicious activity. It is easily configurable with several 'classes' of items, active penetration attempts which is screams about immediately, bad activity, and activity to be ignored (for example DNS server statistics or SSH rekeying). Logcheck is available from: http://www.psionic.com/abacus/logcheck/.

2) Colorlogs:

colorlogs will color code log lines allowing you to easily spot bad activity. It is of somewhat questionable value however as I know very few people that stare at log files on an on-going basis. You can get it at: http://www.resentment.org/projects/colorlogs/.

3) WOTS

WOTS collects log files from multiple sources and will generate reports or take action based on what you tell it to do. WOTS looks for regular expressions you define and then executes the commands you list (mail a report, sound an alert, etc.). WOTS requires you have perl installed and is available from: http://www.vcpc.univie.ac.at/~tc/tools/.

4) Swatch:

swatch is very similar to WOTS, and the log files configuration is very similar. You can download swatch from: ftp://ftp.stanford.edu/general/security-tools/swatch/.

T. *Kernel logging*

1) Auditd:

auditd allows you to use the kernel logging facilities (a very powerful tool). You can log mail messages, system events and the normal items that syslog would cover, but in addition to this you can cover events such as specific users opening files, the execution of programs, of setuid programs, and so on. If you need a solid audit trail then this is the tool for you, you can get it at: ftp://ftp.hert.org/pub/linux/auditd/.

U. *Shell logging*

1) bash

I will also cover bash since it is the default shell in most Linux installations, and thus it's logging facilities are generally used. bash has a large number of variables you can configure at or during run time that modify how it behaves, everything from the command prompt style to how many lines to keep in the log file.

2) HISTFILE

name of the history file, by default it is ~username/.bash_history

3) HISTFILESIZE

maximum number of commands to keep in the file, it rotates them as needed.

4) HISTSIZE

the number of commands to remember (i.e. when you use the up arrow key).

The variables are typically set in /etc/profile, which configures bash globally for all users, the values can however be over-ridden by users with the ~username/.bash_profile file, and/or by manually using the export command to set variables such as export EDITOR=emacs. This is one of the reasons user directories should not be world readable, as the bash_history file can contain a lot of valuable information to a hostile party. You can also set the file itself non world

readable, set your .bash_profile not to log, set the file non writeable (thus denying bash the ability to write and log to it) or link it to /dev/null (this is almost always a sure sign of suspicious user activity, or a paranoid user). For the root account I would highly

VI. CONCLUSION

The fundamental Linux securities not change so much however there are many Linux security and monitoring extensions enhancement tools. These extensions and tools seem overlapped in many aspects. There should be an independent body that coordinates Linux security framework or tools development and adoption.

We can appreciate that although without starting from scratch in designing new secure kernel, the approaches to provide a secure OS start from designing compiler and using new safer C/C++ libraries.

In dealing with the current vulnerabilities we need to face many new challenges from time to time.

REFERENCES

[1] The Linux Kernel Archives site, "The primary site for the Linux kernel source", http://kernel.org/
[2] The Linux Distributions information site, http://distrowatch.com/
[3] Buffer overflows tutorial, http://www.tenouk.com/Bufferoverflowc/Bufferoverflow1.html
[4] USDA's C2 LEVEL OF TRUST information, http://www.ocio.usda.gov/directives/doc/DM3535-001.htm
[5] The Linux-PAM Guides, http://www.kernel.org/pub/linux/libs/pam/Linux-PAM-html/
[6] The first patch was released by Ingo Molnar of Red Hat and first released in May 2003, ExecShield information, http://people.redhat.com/mingo/exec-shield/
[7] NX/XD bit information at Wikipedia, http://en.wikipedia.org/wiki/NX_bit
[8] Geek.com, "Desktop NX/XD-enabled Intel processors already available", http://www.geek.com/desktop-nxxd-enabled-intel-processors-already-available/?rfp=dta
[9] linuxfromscratch.org, Position Independent Executables (PIE) information, http://www.linuxfromscratch.org/hlfs/view/unstable/glibc-2.6/chapter02/pie.html
[10] Security-Enhanced Linux homepage at National Security Agency (NSA)/Central Security Service (CSS), http://www.nsa.gov/selinux/
[11] Proceedings of the GCC Developers Summit, Ottawa, Ontario Canada, May 25–27, 2003, gccsummit-2003-proceedings.pdf
[12] Homepage of The PaX Team, http://pax.grsecurity.net/
[13] kerneltrap.org, "Linux: PaX vs. ExecShield, An ExecShield Perspective", January 20, 2005 - 6:40pm, by Jeremy, http://kerneltrap.org/node/4590
[14] kerneltrap.org, "Pax vs. ExecShield: Blowing away the smoke", July 9, 2005 - 5:59am, by bluefoxicy on July 9, 2005 - 5:59am, http://kerneltrap.org/node/5396
[15] Rationale for TR 24731 Extensions to the C Library Part I: Bounds-checking interfaces, www.open-std.org/JTC1/SC22/WG14/www/docs/TR24731-Rationale.pdf
[16] ISO/IEC WDTR 24731-2, Specification for Safer C Library Functions — Part II: Dynamic Allocation Functions, www.open-std.org/jtc1/sc22/wg14/www/docs/n1193.pdf
[17] Specification for Safer, More Secure C Library Functions, ISO/IEC draft Technical Report, www.open-std.org/jtc1/sc22/wg14/www/docs/n1135.pdf
[18] The LOCK project, O. S. Saydjari, J. M. Beckman, and J. R. Leaman. LOCK Trek: Navigating Uncharted Space. In *Proceedings of the 1989 IEEE Symposium on Security and Privacy*, pages 167-175, 1989.
[19] Distributed Trusted Mach (DTMach), T. Fine and S. E. Minear. Assuring Distributed Trusted Mach. In *Proceedings IEEE Computer Society Symposium on Research in Security and Privacy*, pages 206-218, May 1993.
[20] The Distributed Trusted Operating System (DTOS) project, S. E. Minear. Providing Policy Control Over Object Operations in a Mach Based System. In *Proceedings of the Fifth USENIX UNIX Security Symposium*, pages 141-156, June 1995.
[21] The Distributed Trusted Operating System (DTOS) Home Page http://www.cs.utah.edu/flux/fluke/html/dtos/HTML/dtos.html
[22] University of Utah, The Flux Research Group, http://www.cs.utah.edu/flux/
[23] University of Utah, Fluke: Flux µ-kernel Environment, http://www.cs.utah.edu/flux/fluke/html/index.html
[24] Flask: Flux Advanced Security Kernel, http://www.cs.utah.edu/flux/fluke/html/flask.html
[25] Role Set Based Access Control, RSBAC, MAC kernel security enhancement project for Linux. http://www.rsbac.org/why
[26] Multi Level security (MLS), "SELinux and MLS: Putting the Pieces Together", by Chad Hanson, Trusted Computer Solutions, Inc.
[27] Trusted Computing Platform Alliance (TCPA), an initiative led by Intel, http://www.trustedpc.org/
[28] Trusted Computing FAQ, "TC / TCG / LaGrande / NGSCB / Longhorn / Palladium / TCPA", Version 1.1, August 2003, by Ross Anderson, http://www.cl.cam.ac.uk/~rja14/tcpa-faq.html
[29] IBM PCI Cryptographic Coprocessor, http://www-03.ibm.com/security/cryptocards/pcicc/overview.shtml
[30] The Bastille Linux homepage, http://www.bastille-linux.org/