The Basics of Linux Security

What every Linux user
should know about security

Presented at the
2007 Nebraska CERT Conference

Adam Haeder
Vice President of Information Technology
AIM Institute

Basic Linux Concepts

- Free (as in speech), Unix-like operating system
- Licensed under the GNU General Public License
- Runs on everything from a wristwatch to a mainframe
- Supported by many companies and software developers from around the world

Basic Linux Concepts

- Shares the same basic concepts with Unix:
  - EVERYTHING IS A FILE
  - Small, simple commands are chained together to perform complex operations
  - Plain ASCII text configurations files for systems and application configuration

Basic Linux Concepts

- The proverbial LEGO operating system
  - Made up of thousands of different pieces, most of them following their own rules
  - Advantage: you can make it do whatever you want; you have complete control
  - Disadvantage: you can make it do whatever you want; you have complete control
- Open source nature theoretically makes all bugs shallow. But they're still there!

Isn't Linux already secure?

- Why do I need to worry about this? I thought Linux was already secure? Haven't all those eyeballs in the bazaar squashed every possible security bug there was to squash?

Short answer: No

The system is only as secure as the person managing it.

Some general guidelines

- Be paranoid! Just because you're not paranoid, doesn't mean they're not out to get you
- Don't think you're not a target
- Trust no one but yourself, and still audit yourself
- Assume the worst will happen, and be prepared when it does

Security in Linux

- Security = Knowledge + Implementation
- Or to put it another way,
  ”Know what's going on, know what needs to be done, and then do it!”
- Step 1 of security: knowledge

Knowledge gathering tools

- ps – what processes are running
- last – who logged in last and from where
- w – who is on the system right now
- netstat – what ports are open
- nmap – port scanning tool
- lsof – List Open Files
- Log files: /var/log/messages, /var/log/secure, /var/log/boot.log

ps – Process List

- Part of the 'procps' package
- Many, many options. ps -aux is usually sufficient for most needs
- while 'ps' gives you a one time snapshot, 'top' will give you a continually updated snapshot

ps -aux output

- User – who owns the process
- PID – Process ID
- %CPU – How much CPU the process is using
- %MEM – How much memory the process is using
- VSZ – Virtual memory usage
- RSS – Real memory usage
- TTY – terminal the process is on
- STAT – Process status

ps -aux output (Cont.)

- Start – Date the process started
- Time – Time the process started
- Command – command line of the process
- Status options:
  - R – Running
  - S – Sleeping
  - I – Idle
  - T – Stopped
  - Z - Zombie
  - D – Waiting on Disk
  - W – Swapped out
  - N – Niced down
  - < - Niced up

Other information tools

- last – login and system boot record
- w – quick snapshot of who is logged in
- netstat – network statistics
- lsof – List Open Files

Examples!

Syslog

- The standard unix syslogd (and klogd) service logs to files in /var/log
- Configured via /etc/syslogd.conf
- Pros: standard software, been around a long time, many tools exist to parse and monitor
- Cons: security, overhead, plain text data
- Other options: syslog-ng, minirsyslogd, modular syslog, multilog, snare

Whom do you trust?

- Where do all these tools get their data? How can I trust their output?
- I want to be more paranoid, can you help me?

Yes I can!

/proc filesystem

- Source of all informational data on a linux system
- Not a 'real' filesystem, but kernel memory and system settings represented as files
- Why? Because.... wait for it.... EVERYTHING IN UNIX IS A FILE
- Mostly used for read-only data, but has some read-write portions as well

/proc filesystem

- /proc/meminfo
- /proc/cpuinfo
- /proc/interrupts
- /proc/ioports
- /proc/kcore
- /proc/[0-9]+ directories

Examples!

What does 'secure' mean?

- To at least start down the security road (because security is a journey, not a destination), we must know, or be able to find out:
  - All login attempts, for all applications that support logins
  - What processes are running and why
  - What files change, when, and why
  - What ports are open, what states are they in

Logins

- Common applications that allow logins:
  - /bin/login
  - sshd
  - ftpd
  - telnetd (just kidding)
  - gdm
  - samba
  - sudo

Logins

Aug 13 01:20:31 amos sshd[26223]: Invalid user webmaster from 201.17.246.26
Aug 13 01:20:31 amos sshd[26224]: input_userauth_request: invalid user webmaster
Aug 13 01:20:31 amos sshd[26223]: pam_unix(sshd:auth): check pass; user unknown
Aug 13 01:20:31 amos sshd[26223]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=c911f61a.bhz.virtua.com.br

Aug 13 01:20:31 amos sshd[26223]: pam_succeed_if(sshd:auth): error retrieving information about user webmaster
Aug 13 01:20:34 amos sshd[26223]: Failed password for invalid user webmaster from 201.17.246.26 port 40048 ssh2
Aug 13 01:20:34 amos sshd[26224]: Received disconnect from 201.17.246.26: 11: Bye Bye
Aug 13 01:20:36 amos sshd[26227]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=c911f61a.bhz.virtua.com.br
  user=mysql
Aug 13 01:20:38 amos sshd[26227]: Failed password for mysql from 201.17.246.26 port 40187 ssh2
Aug 13 01:20:38 amos sshd[26228]: Received disconnect from 201.17.246.26: 11: Bye Bye

Logins

[root@amos log]# cat secure | grep "Failed password for invalid user" | awk -F" " '{print $13}' | sort | uniq -c | sort -rn
    59 201.17.246.26
[root@amos log]# host 201.17.246.26
26.246.17.201.in-addr.arpa domain name pointer c911f61a.bhz.virtua.com.br.
[root@amos log]#

Aug 13 12:35:35 fileserv smbd[11556]: [2007/08/13 12:35:35, 0] lib/access.c:check_access(327)
Aug 13 12:35:35 fileserv smbd[11556]: Denied connection from (89.120.76.82)
Aug 13 12:35:35 fileserv smbd[11557]: [2007/08/13 12:35:35, 0] lib/access.c:check_access(327)
Aug 13 12:35:35 fileserv smbd[11557]: Denied connection from (89.120.76.82)
Aug 13 12:35:36 fileserv smbd[11558]: [2007/08/13 12:35:36, 0] lib/access.c:check_access(327)
Aug 13 12:35:36 fileserv smbd[11558]: Denied connection from (89.120.76.82)

root@adamh-laptop:~# more /var/log/vsftpd.log
Mon Aug 13 15:30:26 2007 [pid 15738] CONNECT: Client "127.0.0.1"
Mon Aug 13 15:31:14 2007 [pid 15802] CONNECT: Client "127.0.0.1"
Mon Aug 13 15:31:16 2007 [pid 15801] [adamh] OK LOGIN: Client "127.0.0.1"
Mon Aug 13 15:31:25 2007 [pid 15811] CONNECT: Client "127.0.0.1"
Mon Aug 13 15:31:30 2007 [pid 15810] [adamh] FAIL LOGIN: Client "127.0.0.1"

Running Processes

- 2 things to do: make sure the processes you want to stay running are, in fact, running, and make sure no new rogue processes start
- Simple scripts can be employed to handle both of these situations.
- From a security standpoint, looking for new processes is probably the more important task
- Process Change Detection System (PCDS) – script for watching for new processes

Changes to open ports

- Use lsof or netstat to see current state of your tcp/udp connections
- **netstat -anp** or **lsof -i**
- Verify you know what each process does and why it needs to listen on a port
- Use a script like nmapparser to watch for changes to port status

Filesystem changes

- What files change, when and why
- What are the changes?
  - File creation/deletion
  - File modification or timestamp change
  - Permission changes
- Probably the single most important thing to monitor. Why?
  Because everything in Unix is a file!

Filesystem changes

- Filesystem monitors can range from the simple to the complex
- See script handout for simple version
- More complex versions: AIDE, Tripwire
- All share the same concept: Create a 'snapshot' of a known good system, then monitor the system periodically for changes against that snapshot

Filesystem changes

- Package management tools have basic filesystem verification built in
- RPM (RedHat Package Manager)
  - Used by RedHat, Fedora, SuSE, CentOS and others
  - common uses:
    - rpm -qa <- List all currently installed packages
    - rpm -q package -l <- list all files provided by package
    - rpm -q –whatprovides /bin/ps <- what package gives us the file /bin/ps
    - rpm -Va <- validate all packages on the system

Filesystem changes

- Can you trust your version of /bin/ps?

```
[root@fileserv ~]# rpm -q --whatprovides `which ps`
procps-3.2.6-3.5
[root@fileserv ~]# rpm -V `rpm -q --whatprovides /bin/ps`
[root@fileserv ~]# mv /bin/ps /tmp/
[root@fileserv ~]# touch /bin/ps
[root@fileserv ~]# rpm -V `rpm -q --whatprovides /bin/ps`
SM5....T    /bin/ps
[root@fileserv ~]#
```

- S = Size is different
- M = Mode is different
- 5 = md5sum is different
- T = creation time is different

Filesystem changes

- Debian-based distributions use dpkg and apt-get to manage packages
- The **debsums** program will maintain md5sums of all files in a package
- Not as flexible as **rpm -Va**, and it's not built in, so you have to install it and maintain it
- Once it's installed, it integrates with apt-get so all new packages that are installed get md5sums created for them

Filesystem changes

- On existing Debian-based systems (Debian and Ubuntu being the most popular), run these commands to get **debsums** up and running:

```
# apt-get install debsums
# cd /var/cache/apt/archives
# debsums --generate=all,keep
# apt-get -d install `debsums -l`
  --reinstall
# debsums --generate=all,keep
```

Application Level Security

- Most apps have at least basic ”what IP addresses are allowed” security options
- Some handle it themselves, others go through xinetd
- Not the be-all, end-all in security, but it can help you to not be the low hanging fruit

Application Level Security

- openssh-server: server config file is /etc/ssh/sshd_config
- Common options:
  - Port 12421
  - Protocol 2
  - PermitRootLogin no
  - IgnoreRhosts yes
  - HostbasedAuthentication no
  - PermitEmptyPasswords no

Application Level Security

- openssh-server uses /etc/hosts.allow and /etc/hosts.deny to control access by IP
- Common configuration: only allow ssh connections from IP addresses in the 10.0.0.0/24 range:

```
/etc/hosts.deny:

sshd: ALL

/etc/hosts.allow:

sshd: 10.0.0.
```

Application Level Security

- Other common apps that use /etc/hosts.allow and /etc/hosts.deny for IP based access restriction:
  - portmap
  - most ftp servers
  - telnetd
  - finger server
  - nfs
- Some notable ones that do NOT: apache, mysql

Securing Apache

- Whole books have been written about this
- Usually it's the cgi programs, not apache, that you have to worry about
- A few basics:
  - Only listen on IP addresses you need to
  - Don't enable modules you won't use
  - Run as a non-privileged user
  - Consider a chroot jail
    - Good step-by-step howto on configuring apache from source in a chroot jail: http://www.securityfocus.com/infocus/1694
  - Keep up to date!

Securing MySQL

- Again, a pretty big topic
- Some basics:
  - Only listen on the IP addresses you need to. Start mysqld with the --**bind-address=IP** option
  - Run on a nonstandard port, and/or filter incoming connections to that port with your firewall
  - Create restricted user accounts for each database you use, giving only what permissions are required
  - Watch those cgi programs! Way too easy to code poorly and be open to SQL injection attacks

File Permissions

- setuid bit – the program will always execute with the permissions of the owner, no matter who runs it
- Find all files on your system that are setuid:

```
# find / -perm -4000
```

- Make sure you know why they are setuid. Some don't need to be: ping, mount, umount
- Some do: passwd, sudo

Intrusion Detection

- In addition to monitoring logins, file systems and processes, you can also log what's going on over your network
- Intrusion Detection systems work by comparing network traffic against known attack signatures
- If a match is found, it's assumed that an attack is in place

Intrusion Detection

- Pros: A little more proactive than the reactive monitoring of processes, files, etc
- Cons:
  - Must keep the attack database up to date
  - Too many false positives cause the real attacks to be buried in the noise
- Most popular option on linux is Snort: http://www.snort.org

Firewalling with iptables

- Do we need a firewall? Why?
- **iptables** is the program used to manipulate the packet mangling options in the linux kernel
- Some possibilities:
  - Block traffic from certain IP addresses
  - Block traffic to certain ports
  - Slow down (throttle) traffic to certain ports
  - Enable port-forwarding on a multi-homed system
  - Enable network address translation on a network gateway
  - Log everything

Frontends

- Bastille Linux - http://www.bastille-linux.org/
  - Menu driven interface to harden a system
  - Includes options like setuid permissions, basic firewalling, restricted limits for users, and many others
- Most distrbutions come with some sort of iptables frontend. But there are many others:
  - shorewall, guidedog, guarddog, ipkungfu, kmyfirewall, knetfilter, lokkit, pyroman, fireflier

In conclusion

- Security is a journey, not a destination
- Get into the security state of mind
- There is knowledge in knowing,
  but wisdom in doing
- The price of security is constant vigilence

The Basics of Linux Security

Presented at the

2007 Nebraska CERT Conference

Adam Haeder

adamh@aiminstitute.org

GnuPG Key:
http://careerlink.com/adamhaederpgp.html

Vice President of Information Technology

AIM Institute