Introduction to Linux Administration

Feb 2007

Alok M.

Overview

- Unix Filesystem Layout
- Gentoo Installation
- Some Basic Commands
- Regular Expressions
- More Advanced Commands
- The Log Architecture
- Kernel Audit Subsystem
- Firewall / Packet Filtering
- Securing communications with IPSec
- Keeping Gentoo up-to-date

Filesystem Hierarchy Standard (1)

“This standard consists of a set of requirements and guidelines for file and directory placement under UNIX-like operating systems. The guidelines are intended to support interoperability of applications, system administration tools, development tools, and scripts as well as greater uniformity of documentation for these systems.”

Filesystem Hierarchy Standard (2)

/bin	Essential binaries
/sbin	Essential root binaries
/lib	Essential libraries for /bin and /sbin
/dev	Devices
/etc	Configuration files
/home & /root	Users' & root's home directory
/mnt & /media	Temporary / removable filesystems mount points
/opt	Add-on applications

Filesystem Hierarchy Standard (3)

/proc	Virtual filesystem which provides information about kernel, processes, network, uptime, etc.
/tmp, /usr/tmp & /var/tmp	Temporary files
/usr/bin	Binaries
/usr/sbin	Root binaries
/usr/lib	Libraries for /usr/bin and /usr/sbin

Filesystem Hierarchy Standard (4)

/var/lock	Lock files
/var/log	Log files
/var/run	Information about currently logged-in users and running daemons

- For further information, see:

  http://en.wikipedia.org/wiki/FHS

  http://www.pathname.com/fhs/

  http://www.openaddict.com/documents/Linux-Filesystem-Hierarchy/

Filesystem Hierarchy Standard (5)

- You can have multiple kernels on the same filesystem (i.e. each kernel is located in /boot and the bootloader decides which kernel to boot)
- These kernels will share the same tools
- Note: some applications are kernel version dependent. You might run into trouble if you have different major versions (i.e. mixing 2.4.x and 2.6.x kernels).
- You can install different distributions on different partitions. You can share the /home /boot and swap

Gentoo Installation

- Setup overview
  - Download stage3.tar.bz2
  - Download portage-latest.tar.bz2
  - Unpack these files in the destination drive
  - Install the sources and compile the kernel
  - Install a bootloader
  - Create a user and emerge sudo
  - Boot new kernel
- Documentation

  http://www.gentoo.org/doc/en/handbook/handbook-x86.xml?full=1

Basic Commands (1)

- Switching consoles / shells
  - ctrl-alt-F1, ctrl-alt-F2, etc...
  - ctrl-alt-F7 is the X window (if running)
  - shift-left/shift-right in Konsole
- Getting help
  - man *command*
  - man *config_file*
- Running command with last arguments
  - !*command*

Basic Commands (2)

- Running a command as root

  sudo *command* — The file /etc/sudoers controls who can use sudo, and who can run which commands.

Basic Commands (3)

- Navigating directories and displaying content

  ls — List files in current directory

  ls -l — Column mode

  cd *dir* — Change directory

- Find files, starting from here (.), recursively

  find . -iname "**something**"

- Find files, using database

  updatedb — Update database, you should do this once in a while

  locate *file*

Basic Commands (4)

- Displaying and editing file content

vi *file*	Simple text editor
cat *file*	Display content of file
less *file*	Display content of file in interactive mode (you can search forward and backwards)
head -n *N file*	Display first N lines
tail -n *N file*	Display last N lines
tail -f *file*	Display last lines and keep updating display as new content is added

Basic Commands (5)

- vi survival guide:

i, a	Enter insertion / append mode
/, ?	Text search forward / backward
n	Repeat last search
h, j, k, l	Move cursor (left, down, up, right)
0, $	Move cursor to beginning / end of line
dd	Delete line
dh, dl	Delete character left / right
:w	Save file
:q, :q!	Quit / quit without saving
N cmd	Execute command *N* times

Basic Commands (6)

- Displaying file & disk space usage

  du -h *dir*

  df -h /dev/*disk*

- Running command every N seconds

  watch -n *N* " *cmd* "

- Clear console

  clear

- Break

  ctrl-C

Basic Commands (7)

- ssh lets you access remote computers (remote shell) and also transfer files (scp). You must either log in with a username/password, or use keys.
- To enable ssh daemon:

  /etc/init.d/sshd start (on remote machine)

- Remote shell:

  ssh *remote_computer*

- File transfer:

  (to remote computer) scp *file user@remote_computer:~/*

  (from remote computer) scp *user@remote_computer:path .*

Basic Commands (8)

- To use ssh with keys, you must first generate a pair of keys:

  ssh-keygen

- You must then copy the public key to the server:

  scp ~/.ssh/id_rsa.pub *user@remote_computer:~/new_key*

  ssh *user@remote_computer*

  cat new_key >> ~/.ssh/authorized_keys

  rm new_key

- You can now log in directly on remote server:

  ssh *remote_computer*

- Note: as long as you keep the ssh keys in the standard location (~/.ssh/id_rsa), everything is pretty straight forward and easy.

Basic Commands (9)

- Processes and memory:

  top — Continuously updates the display

- I/O information:

  dstat — Replaces vmstat, iostat, ifstat

- Open file handles:

  lsof

- Processes:

  ps

  ps aux — All processes

  pstree & ps axjf — Tree view

Basic Commands (10)

- Network

tcpdump	Traffic dump. You can provide filters: e.g. tcpdump 'tcp' or tcpdump 'port 80'. Use tcpdump -s 0 for full capture
ntop	Traffic statistics
iftop	Bandwidth usage
pbnj	Monitor network change
iptables	Linux firewall (for command examples, see page 39)
arpstar	Detect arp spoofing

Basic Commands (11)

- Linux has a powerful pipes and redirections:

ls > *file*	Redirect standard output to file
ls >> *file*	Send standard output to end of file
ls -l | less	Pipe output of ls to less
ls 2> /dev/null	Redirect standard error to /dev/null (discards error messages)

- Most commands are designed to use the standard input if no file is provided on the command line:

  grep *regexp file*

  cat *file* | *grep regexp*

Regular Expressions (1)

- Regular expressions are a powerful way for searching, filtering or modifying text
- A lot of tools support some form of regular expressions (perl, awk, sed, grep, etc.)
- Each tool uses it own flavor of regular expressions, but the perl regular expressions are the most widespread
- C programmers can use PCRE to provide perl compatible regular expressions to the user

Regular Expressions (2)

- Metacharacters

^ , $	Beginning / End of string
.	Any character except newline
*	Match 0 or more times
+	Match 1 or more times
?	Match 0 or 1 times; or: shortest match
|	Alternative
( )	Grouping; "storing"
[ ]	Set of characters
{ }	Repetition modifier
\	Quote or special

Regular Expressions (3)

- Repetition

a*	Zero or more a's
a+	One or more a's
a?	Zero or one a's (i.e., optional a)
a{m}	Exactly m a's
a{m,}	At least m a's
a{m,n}	At least m but at most n a's
repetition?	Same as repetition but the shortest match is taken

Regular Expressions (4)

- Special notation with \

\t, \n, \r	Tab, newline, return (CR)
\xNN	Character with hex code NN
\b	"word" boundary
\B	Not a "word" boundary
\w	Matches any single character classified as a "word" character.
\W	Matches any non-"word" character
\s	Matches any whitespace character (space, tab, newline)
\S	Matches any non-whitespace character
\d	Matches any digit character, equiv. to [0-9]
\D	Matches any non-digit character

Regular Expressions (5)

- Character classes

[characters]	Matches any of the characters in the sequence
[x-y]	Matches any of the characters from x to y (inclusively)
[^something]	Matches any character except those that [something] denotes

Regular Expressions (6)

- Examples

/abc/	abc (that exact character sequence, but anywhere in the string)
/^abc/	abc at the beginning of the string
/abc$/	abc at the end of the string
/a|b/	Either of a and b
/^abc|abc$/	The string abc at the beginning or at the end of the string
/ab{2,4}c/	a followed by two, three or four b's followed by a c
/ab{2,}c/	a followed by at least two b's followed by a c
/ab*c/	a followed by any number (zero or more) of b's followed by a c
/ab+c/	a followed by one or more b's followed by a c
/ab?c/	a followed by an optional b followed by a c; that is, either abc or ac

Regular Expressions (7)

/a.c/	a followed by any single character (not newline) followed by a c
/a\.c/	a.c exactly
/[abc]/	Any one of a, b and c
/[Aa]bc/	Either of Abc and abc
/[abc]+/	Any (nonempty) string of a's, b's and c's (such as a, abba, acbabcacaa)
/[^abc]+/	Any (nonempty) string which does not contain any of a, b and c (such as defg)
/\d\d/	Any two decimal digits, such as 42; same as \d{2}
/\w+/	A "word": a nonempty sequence of alphanumeric characters and low lines (underscores), such as foo and 12bar8 and foo_1
/100\s*mk/	the strings 100 and mk optionally separated by any amount of white space (spaces, tabs, newlines)

Regular Expressions (8)

- More information:

  http://www.perl.com/doc/manual/html/pod/perlre.html

- Regular expressions are powerful for text processing. They are however cumbersome for XML files
- XPath & XSLT are languages designed to search/manipulate XML files

Advanced Commands (1)

grep regexp *file*	Display each line in file that match regexp. The -B and -A
grep -B 10 regexp *file*	switches let you control how many lines before and after the
grep -A 10 regexp *file*	match should be displayed.
sed -e *regexp file*	Filter text
awk '{print $N}'	Display Nth column
perl -e 'perl code'	Run perl code from command line
cut	Select parts of each line
expand	Convert tags to spaces

Advanced Commands (2)

sort	Sort lines on stdin
uniq -c	Merge consecutive identical lines. With -c, display count

- To convert UNIX timestamps into human readable date:

```
awk '{print strftime("%c", $1) $0}'
```

Periodic / Scheduled Jobs

- Cron lets you run scripts / program at regular intervals.
- It lets you keep your system up-to-date, rotate logs, check integrity, etc.
- There are multiple cron daemons (e.g. vixie-cron).
- Usually, the file /etc/crontab will control the cron daemon. Some distributions prefer to create /etc/cron.daily/, /etc/cron.hourly/, etc. helper folders.
- The crontab specifies the scheduled time with 5 fields: min, hour, day of month, month, day of week. A * means any value for that field. A /N means every N for that field:

30 18 * * * *cmd*	Run *cmd* every day at 6.30pm
0,30 8-17 * * 1-5 *cmd*	Run *cmd* every half-hour, from 8am to 5.30pm, Mon-Fri
/10 * * * * *cmd*	Run *cmd* every 10 minutes

Linux Log Architecture (1)

- Logs are managed by a syslogd daemon

  socklog, metalog, syslog-ng, newsyslog, etc.

- Each log belongs to an activity (kern, user, mail, lpr, auth, daemon, news, uucp, local0-local7)
- Each log has a level (emerg, alert, crt, err, warning, notice, info, debug, none)
- Coloring syslog files (regex-markup)
- Logs can be sent/received over network (port 514/udp)
- Some daemons let you log to a database

Linux Log Architecture (2)

- Sample configuration for Syslog-ng. File: /etc/syslog-ng/syslog-ng.conf
  - Define sources

```
source src { unix-stream("/dev/log"); internal(); pipe("/proc/kmsg"); };
```

  - Define destinations

```
destination messages { file("/var/log/messages"); };
destination console_all { file("/dev/tty12"); };
```

  - Define log rules (which links one or more sources with one or more destinations)

```
log { source(src); destination(messages); };
log { source(src); destination(console_all); };
```

Linux Log Architecture (3)

- SNMP is an important aspect of the overall log architecture. Installation:

  emerge net-snmp

- To generate snmp information:

  configure /etc/snmp/snmpd.conf

  /etc/init.d/snmpd start

- You can configure snmp to generate traps, and use snmptrapd to log these traps (and post process them)
- Traps can generate handlers (e.g. send email if something bad happens)
- Debugging:

  snmpwalk

  snmpget

Linux Log Architecture (4)

- Tools to manage SNMP:

snmpmon	A very simple snmp monitor
nagios	Lets you monitor your entire network
mrtg	Monitor network bandwidth
snmptt	Snmp trap processing tool. E.g. lets you log snmp traps into a syslog or a database

Linux Log Architecture (5)

- Security Issues with SNMP:
  - SNMP traffic is by default unencrypted. Sensitive information can be sniffed. WRITE data / setting changes can be injected.
  - SNMP v2c provides "authentication" using community strings. But the community strings can still be sniffed !
  - SNMP v3 provides encryption.

Linux Audit Subsystem

- Needs to be activated in the kernel
  - enable inotify
  - enable kernel audit
- emerge auditd for userland daemon
- auditctl to control what gets logged
- Lets you monitor syscalls and also files (through inodes)
- Logs can/should be sent to a remote syslog server
- Important: if administrators are using sudo bash, "who did what" can be harder to trace.

Firewall / Packet Filtering (1)

- Standard Linux firewall is called iptables
- The old firewall was called ipchains
- iptables needs to be enabled in the kernel:
  - CONFIG_NETFILTER=y
  - CONFIG_IP_NF_...
- emerge iptables will install the userland tools
- Multiple graphical and non-graphical interfaces are available, but it's best to learn how to use the standard command line tool.

Firewall / Packet Filtering (2)

- IPTables has 3 main tables (filter, nat, mangle). Each table has multiple chains, and each chain contains rules. The rules in a given chain are processed from top to bottom.
- You can either load the iptables with a bash script or use the binary load/save format built into iptables.
- You can create custom chains to simply the rules logic.
- Displaying tables:

  iptables -v -n -L

  iptables -n --line-numbers -t nat

- Setting the default policy for a chain:

  iptables -P INPUT DROP

Firewall / Packet Filtering (3)

- Adding rules to accept or deny traffic:

  iptables -A OUTPUT -p tcp --dport 80 -j DROP

  iptables -A INPUT -p icmp -j ACCEPT

  iptables -A INPUT -p tcp --dport 22 -s 192.168.1.4 -j ACCEPT

- Adding a rule to accept related traffic (iptables is a stateful firewall):

  iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT

- Logging packets in iptables:

  iptables -A INPUT -j LOG --log-prefix "in input: "

- Deleting a single rule / all rules:

  iptables -D INPUT *rule_number*

  iptables -F INPUT

Firewall / Packet Filtering (4)

- Saving iptables state (in Gentoo):

  /etc/init.d/iptables save

- Loading iptables (in Gentoo):

  /etc/init.d/iptables reload

- Typical iptables script (other distributions):

```
#/bin/bash
iptables -F INPUT
iptables -F OUTPUT
iptables -P INPUT DROP
...
```

Firewall / Packet Filtering (5)

- Documentation:

  http://www.netfilter.org/documentation/index.html

- Note: different distributions have different firewall softwares:
  - Linux: IPTables, (IPChains)
  - Solaris: IPFilter, SunScreen
  - BSD: PF (mostly for simple cases), IPFilter, IPFW (can support more complex rules)
  - IPFilter has been ported to Linux. Other firewalls might also have been cross ported.

IPSec (1)

- Kernel options:
  - CONFIG_NET_KEY=y
  - CONFIG_INET_AH=y
  - CONFIG_INET_ESP=y
  - CONFIG_INET_IPCOMP=y
  - CONFIG_INET_XFRM_TUNNEL=y
  - CONFIG_INET_TUNNEL=y
- Userland tools:
  - emerge strongswan

IPSec (2)

- Create a CA and copy it to each host:

  openssl req -x509 -newkey rsa:2048 -keyout cakey.pem -out cacert.pem

  scp cacert.pem *host*:/etc/ipsec/ipsec.d/cacerts/

- Create a key & cert for each host:

  openssl req -newkey rsa:1024 -keyout hostNkey.pem -out hostNreq.pem

  cp hostNkey.pem /etc/ipsec/ipsec.d/private/

- Sign the host cert with the CA:

  scp hostNreq.pem *cahost*

  openssl ca -cert cacert.pem -keyfile cakey.pem -in hostNreq.pem -out hostNcert.pem

  scp hostNcert.pem *host*:/etc/ipsec/ipsec.d/certs/

IPSec (3)

- There are multiple ways to setup a CRL:
  - Deploy revocation lists using http
  - Deploy revocation lists using ldap
  - Use OCSP, which is more complex but provides more flexibility (live revocation checking, etc.)

IPSec (4)

- Sample /etc/ipsec/ipsec.conf

```
conn host-host
    left=192.168.1.4
    leftcert=host1cert.pem
    right=192.168.1.12
    rightid="C=CH,ST=Host2,O=Host2,..."
    auto=start
    dpdaction=hold
```

- More information:

  http://www.strongswan.org/docs/readme.htm

IPSec (5)

- Starting the daemon:

  /etc/init.d/ipsec start

- Watching connections:

  watch -n 1 "ipsec statusall | grep tunnel"

- Syslog messages
- SNMP (ipsec MIB)
- Sample Tcpdump:

  IP 192.168.1.4 > 192.168.1.12: ESP(spi=0x02002f84,seq=0x3), length 116

  IP 192.168.1.12 > 192.168.1.4: ESP(spi=0x6e37ba1b,seq=0x3), length 116

  IP 192.168.1.12 > 192.168.1.4: ICMP echo reply, id 61795, seq 1, length 64

Gentoo Package Management (1)

- Gentoo package management is called portage. You will use 3 tools:

  esearch

  emerge

  equery

- To update your local repository of packages, you should either run esync, or run emerge-webrsync && eupdatedb if you are behind a firewall.
- glsa-check lets you see what security patches need to be applied right away.
- You can search you local repository by using the esearch tool:

esearch *package*	Searches database for package called *package*
esearch -S *"string"*	Searches description for *string*

Gentoo Package Management (2)

- Updating your entire system:

  emerge -Dupv world — The -p flags means "pretend", doesn't actually do anything, just lets you see what is going to happen

  emerge -Du world — Actually does the update

- Installing a new package:

  emerge -pvt *pkg* — To see what is going to happen

  emerge *pkg* — To perform the installation

- Packages have USE flags, which control their installation options. You can edit USE flags in /etc/make.conf or /etc/portage/package.use. If the package has already been installed, you can reinstall with the new flags with: emerge -N *pkg*

Gentoo Package Management (3)

- Some packages are marked unstable, and need to be unmasked before you can install them. Edit /etc/portage/package.keywords before using emerge
- equery lets you see what files belong where, and what package depends on what:

equery belongs *file*	See which package installed a given file
equery files *pkg*	See what files a given package installed
equery depends *pkg*	See what packages depend on *pkg*
equery depgraph *pkg*	See packages that *pkg* depends on

- Some large applications are split into meta-packages. This way you can install parts of the larger application. A typical example is KDE. In such a case it is a good habit to install the meta package instead of individual programs (e.g. equery depends kcalc will show you that you can install kdeutils-meta instead of kcalc).

Gentoo Package Management (4)

- Usually, when portage updates a given package, it will remove the older version. The exception to this rule is SLOTs, which can coexist. This is useful for libraries (e.g. freetype v1 and v2).
- Note: portage will check for package dependencies and install all required packages automatically. It however will fail if a required package needs to be compiled with specific options. For example, if you don't compile gd with -jpeg and -png support, some other packages might fail to compile. In such cases, you need to reemerge gd with the right flags.
- For more information:

  http://www.gentoo.org/doc/en/handbook/handbook-x86.xml?part=2

Upgrading & Compiling Kernel (1)

- Get the new sources

  emerge gentoo-sources

- Fix the symlink

  cd /usr/src

  rm linux

  ln -s linux-version linux

- Compile the new kernel. The old .config file can be used to save time.

  cd linux

  cp ../linux-previousversion/.config .

  make oldconfig or make menuconfig

  make && make modules_install

Upgrading & Compiling Kernel (2)

- Copy the kernel to the /boot partition. You can override the old kernel or give it a new name.

  cp arch/i386/boot/bzImage /boot/*kernel_name*

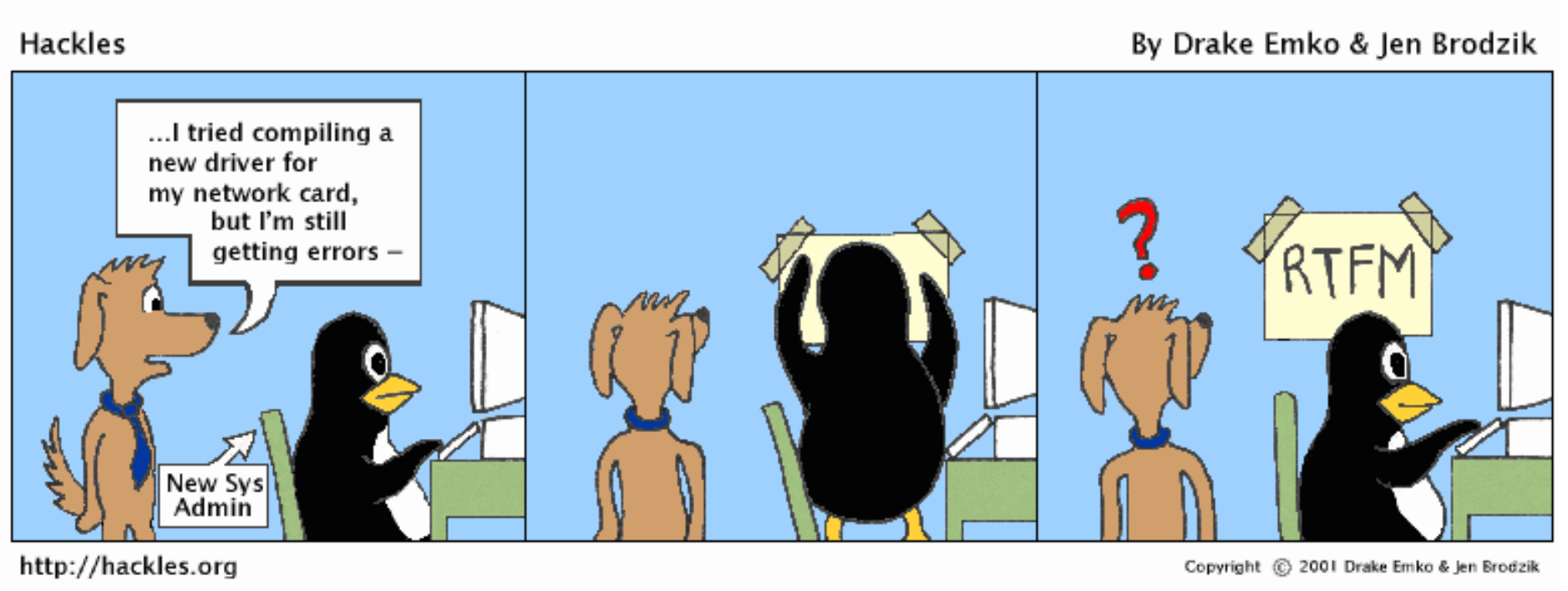
- If required, add a new entry in /etc/lilo.conf

  lilo

  reboot

- If you have any integrity check software, make sure to update the database
- If you have packages that generate modules, they need to be reemerged

What's Next ?


Hackles
By Drake Emko & Jen Brodzik
...I tried compiling a new driver for my network card, but I'm still getting errors –
New Sys Admin
?
RTFM
http://hackles.org
Copyright © 2001 Drake Emko & Jen Brodzik