03 Linux

Notebook:	FGA Cyber		
Created:	7/4/2019 9:10 AM	**Updated:**	7/4/2019 10:04 AM
Author:	clink200032@gmail.com		
URL:	https://itexamanswers.net/ccna-cyber-ops-version-1-1-chapter-3-linux-operating-syste...		

Linux Overview (3.1)

In this section, you will learn Linux basics including the Linux shell and the role of Linux servers and clients.

Linux Basics (3.1.1)

In this topic, you will learn what Linux is, its value, its role in a Security Operations Center (SOC), and some of the more important Linux tools.

What is Linux? (3.1.1.1)

Linux is an operating system created in 1991. Linux is open source, fast, reliable, and small. It requires very little hardware resources to run, and is highly customizable. Unlike other operating systems such as Windows and macOS X, Linux was created, and is currently maintained, by a community of programmers. Linux is part of several platforms and can be found on devices anywhere from "wristwatches to supercomputers."

Another important aspect of Linux is that it is designed to be connected to the network, which makes it much simpler to write and use network-based applications. Because Linux is open source, any person or company can get the kernel's source code, inspect it, modify it, and recompile it at will. They are also allowed to redistribute the program with or without charges.

A Linux distribution is the term used to describe packages created by different organizations. Linux distributions (or distros) include the Linux kernel with customized tools and software packages. While some of these organizations may charge for their Linux distribution support (geared toward Linux-based businesses), the majority of them also offer their distribution for free without support. Debian, Red Hat, Ubuntu, CentOS, and SUSE are just a few examples of Linux distributions.

The Value of Linux (3.1.1.2)

Linux is often the operating system of choice in the SOC. These are some of the reasons to choose Linux:

- Linux is open source: Any person can acquire Linux at no charge and modify it to fit specific needs. This flexibility allows analysts and administrators to tailor-build an operating system specifically for security analysis.
- The Linux CLI is very powerful: While a GUI makes many tasks easier to perform, it adds complexity and requires more computer resources to run. The Linux command line interface (CLI) is extremely powerful and enables analysts to perform tasks not only directly on a terminal, but also remotely because the CLI requires very few resources.
- The user has more control over the OS: The administrator user in Linux, known as the root user, or superuser, has absolute power over the computer. Unlike other operating

systems, the root user can modify any aspect of the computer with a few keystrokes. This ability is especially valuable when working with low-level functions such as the network stack. It allows the root user to have precise control over the way network packets are handled by the operating system.

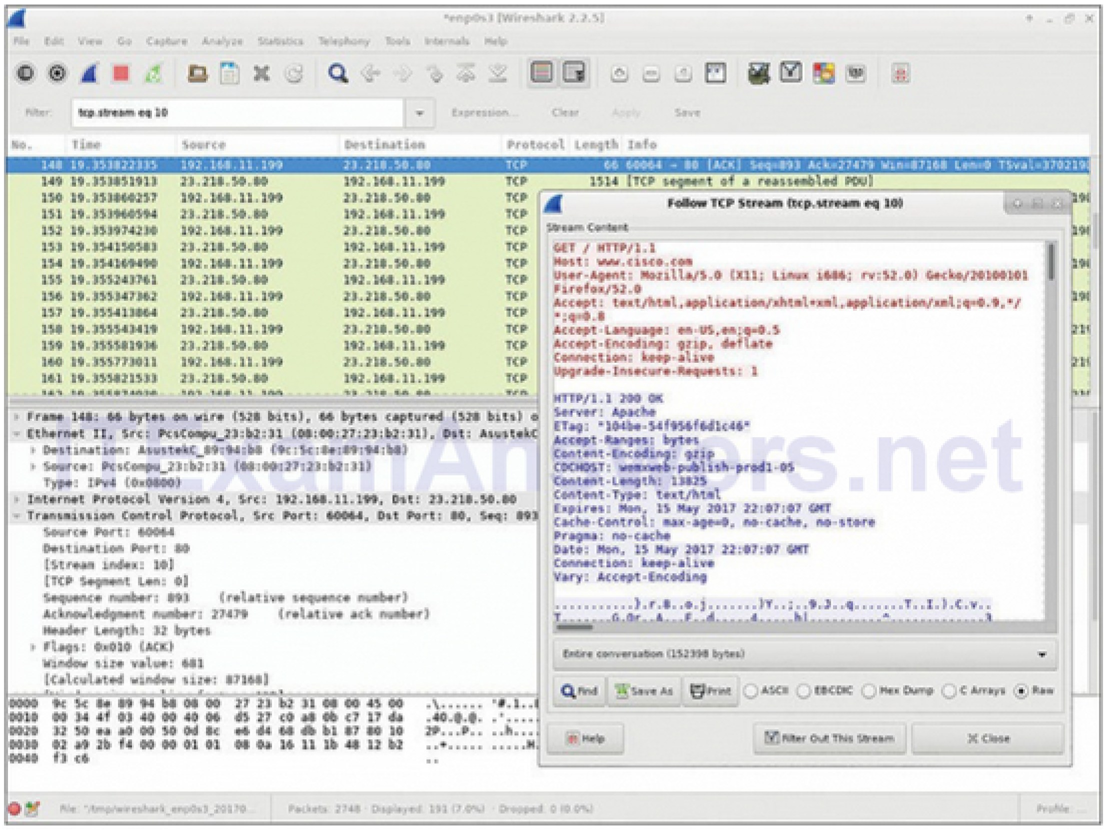
- It allows for better network communication control: Control is an inherent part of Linux. Because the OS can be tweaked and adjusted in practically every aspect, it is a great platform for creating network applications. This is the same reason why many great network-based software tools are available for Linux only.

Linux in the SOC (3.1.1.3)

The flexibility provided by Linux is a great feature for the SOC. The entire operating system can be tailored to become the perfect security analysis platform. For example, administrators can add only the necessary packages to the OS, making it lean and efficient. Specific software tools can be installed and configured to work in conjunction, allowing administrators to build a customized computer that fits perfectly in the workflow of an SOC.

These are a few tools that are often found in an SOC:

- Network packet capture software: This software is used for network packet captures. This is a crucial tool for an SOC analyst as it makes it possible to observe and understand every detail of a network transaction.
  Figure 3-1 shows a screenshot of Wireshark, a popular packet capture tool.


*enp0s3 [Wireshark 2.2.5]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: tcp.stream eq 10
Expression... Clear Apply Save

```
No.   Time          Source           Destination      Protocol Length Info
  148 19.353822335  192.168.11.199   23.218.50.80     TCP        66 60064 → 80 [ACK] Seq=893 Ack=27479 Win=87168 Len=0 TSval=370219
  149 19.353851913  23.218.50.80     192.168.11.199   TCP      1514 [TCP segment of a reassembled PDU]
  150 19.353860257  192.168.11.199   23.218.50.80     TCP
  151 19.353960594  23.218.50.80     192.168.11.199   TCP
  152 19.353974230  192.168.11.199   23.218.50.80     TCP
  153 19.354150583  23.218.50.80     192.168.11.199   TCP
  154 19.354169490  192.168.11.199   23.218.50.80     TCP
  155 19.355243761  23.218.50.80     192.168.11.199   TCP
  156 19.355347362  192.168.11.199   23.218.50.80     TCP
  157 19.355413864  23.218.50.80     192.168.11.199   TCP
  158 19.355543419  192.168.11.199   23.218.50.80     TCP
  159 19.355581936  23.218.50.80     192.168.11.199   TCP
  160 19.355773011  192.168.11.199   23.218.50.80     TCP
  161 19.355821533  23.218.50.80     192.168.11.199   TCP
```

```
▸ Frame 148: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) o
▾ Ethernet II, Src: PcsCompu_23:b2:31 (08:00:27:23:b2:31), Dst: AsustekC
   ▸ Destination: AsustekC_89:94:b8 (9c:5c:8e:89:94:b8)
   ▸ Source: PcsCompu_23:b2:31 (08:00:27:23:b2:31)
     Type: IPv4 (0x0800)
▸ Internet Protocol Version 4, Src: 192.168.11.199, Dst: 23.218.50.80
▾ Transmission Control Protocol, Src Port: 60064, Dst Port: 80, Seq: 893
     Source Port: 60064
     Destination Port: 80
     [Stream index: 10]
     [TCP Segment Len: 0]
     Sequence number: 893    (relative sequence number)
     Acknowledgment number: 27479    (relative ack number)
     Header Length: 32 bytes
   ▸ Flags: 0x010 (ACK)
     Window size value: 681
     [Calculated window size: 87168]

0000  9c 5c 8e 89 94 b8 08 00  27 23 b2 31 08 00 45 00   .\...... '#.1..E
0010  00 34 4f 03 40 00 40 06  d5 27 c0 a8 0b c7 17 da   .4O.@.@. .'.....
0020  32 50 ea a0 00 50 0d 8c  e6 d4 68 db b1 87 80 10   2P...P.. ..h.....
0030  02 a9 2b f4 00 00 01 01  08 0a 16 11 1b 48 12 b2   ..+..... .....H.
0040  f3 c6                                              ..
```

```
GET / HTTP/1.1
Host: www.cisco.com
User-Agent: Mozilla/5.0 (X11; Linux i686; rv:52.0) Gecko/20100101
Firefox/52.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/
*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1

HTTP/1.1 200 OK
Server: Apache
ETag: "104be-54f956f6d1c46"
Accept-Ranges: bytes
Content-Encoding: gzip
CDCHOST: wemxweb-publish-prod1-05
Content-Length: 13825
Content-Type: text/html
Expires: Mon, 15 May 2017 22:07:07 GMT
Cache-Control: max-age=0, no-cache, no-store
Pragma: no-cache
Date: Mon, 15 May 2017 22:07:07 GMT
Connection: keep-alive
Vary: Accept-Encoding

............}.r.8..o.j........}Y..;..9.J..q.......T..I.).C.v..
```

Entire conversation (152398 bytes)
Find
Save As
Print
ASCII
EBCDIC
Hex Dump
C Arrays
[✓ Raw]
Help
Filter Out This Stream
Close
File: "/tmp/wireshark_enp0s3_20170...
Packets: 2748 · Displayed: 191 (7.0%) · Dropped: 0 (0.0%)
Profile: ...

- Malware analysis tools: In the case of new malware detection, these tools allow analysts to safely run and observe malware execution without the riskof compromising the underlying system.
- Intrusion detection systems (IDSs): These tools are used for real-time traffic monitoring and inspection. If any aspect of the currently flowing traffic matches any of the established rules, a predefined action is taken.

- Firewalls: This software is used to specify, based on predefined rules, whether traffic is allowed to enter or leave the network.
- Log managers: Log files are used to record events. Because a large network can generate a very large number of events log entries, log managers are employed to facilitate log monitoring.
- Security information and event management (SIEM): SIEM systems provide real-time analysis of alerts and log entries generated by network appliances such as IDSs and firewalls.
- Ticketing systems: Ticket assignment, editing, and recording is done through a ticket management system.

Linux Tools (3.1.1.4)

In addition to SOC-specific tools, Linux computers used in the SOC often contain penetration testing tools. Also known as pentesting, penetration testing is the process of looking for vulnerabilities in a network or computer by attacking it. Packet generators, port scanners, and proof-of-concept exploits are examples of penetration testing tools.

Kali Linux is a Linux distribution created to group many penetration tools. Kali contains a great selection of penetration testing tools. Figure 3-2 shows a screenshot of Kali Linux. Notice all the major categories of penetration testing tools.

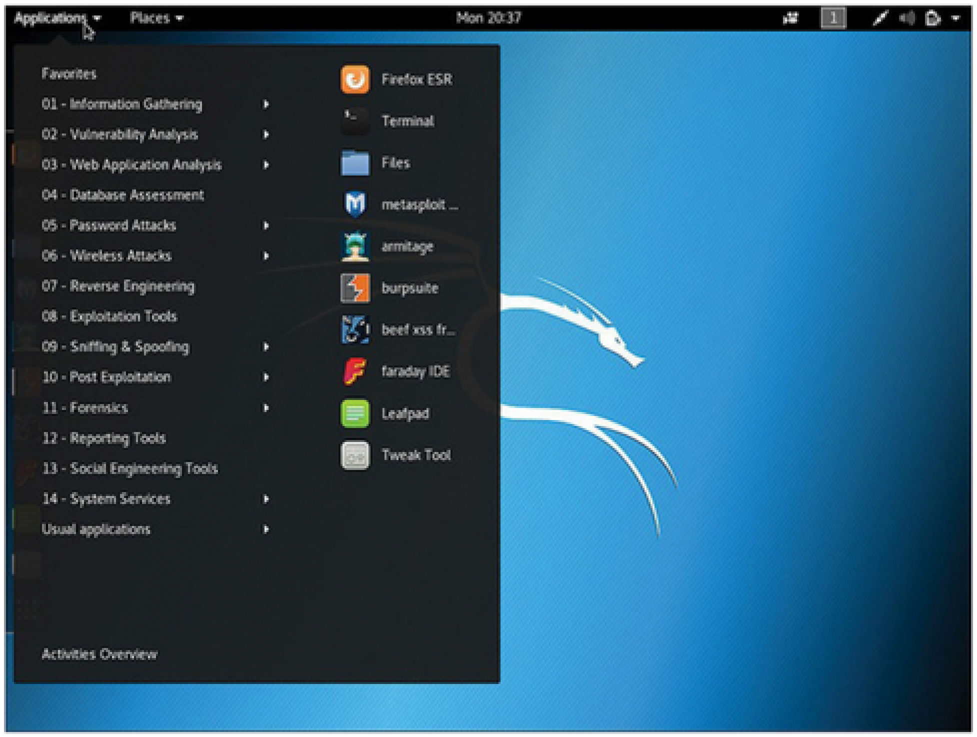


Figure 3-2 Kali Linux Tool Categories

Basic Commands (3.1.2.2)

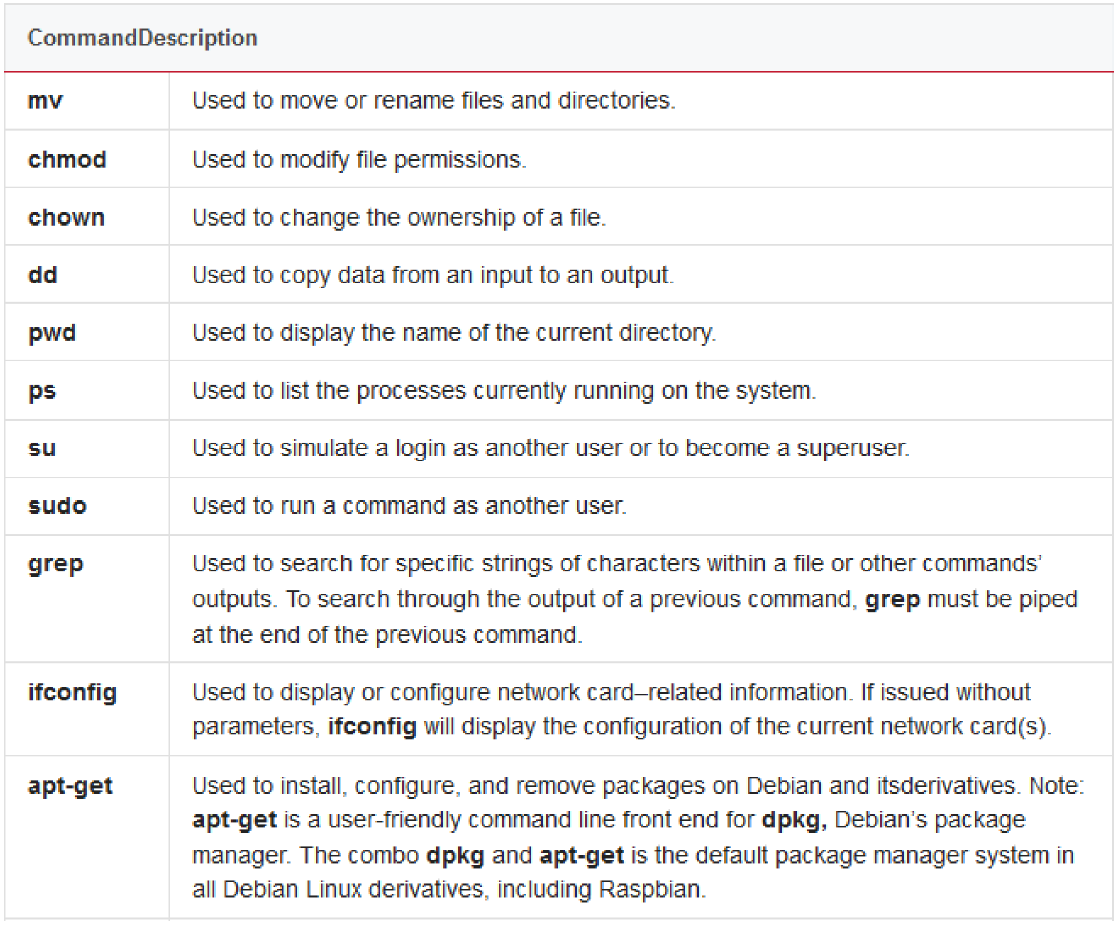
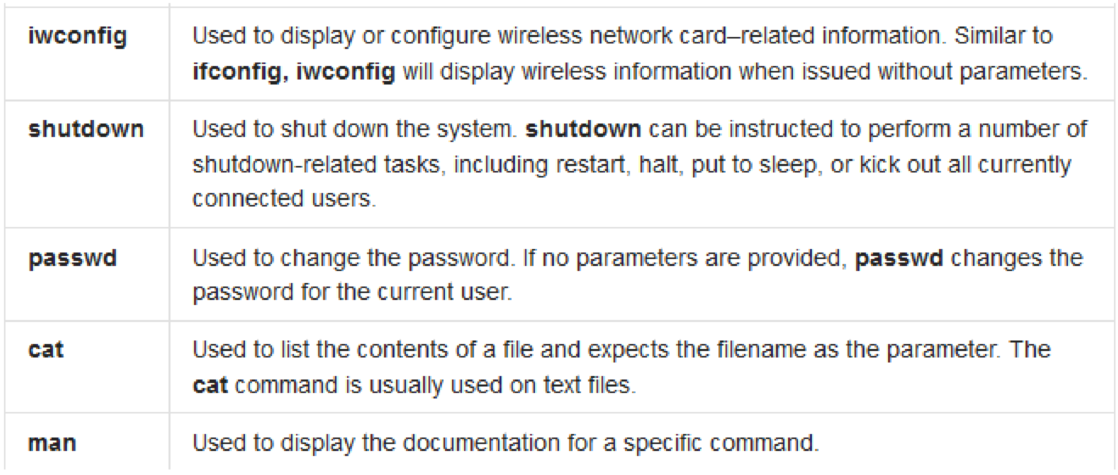
Linux commands are programs created to perform a specific task. Use the **man** command (short for manual) to obtain documentation about commands. As an example, **man ls** provides documentation about the **ls** command from the user manual.

Because commands are programs stored on the disk, when a user types a command, the shell must find it on the disk before it can be executed. The shell will look for user-typed commands in specific directories and attempt to execute them. The list of directories checked by the shell is called the path. The path contains many directories commonly used to store commands. If a command is not in the path, the user must specify its location or the shell will not be able to find it. Users can easily add directories to the path, if necessary.

To invoke a command via the shell, simply type its name. The shell will try to find it in the system path and execute it.

Table 3-1 shows a list of some basic Linux commands and their functions.

Table 3-1 Basic Linux Commands

CommandDescription	
mv	Used to move or rename files and directories.
chmod	Used to modify file permissions.
chown	Used to change the ownership of a file.
dd	Used to copy data from an input to an output.
pwd	Used to display the name of the current directory.
ps	Used to list the processes currently running on the system.
su	Used to simulate a login as another user or to become a superuser.
sudo	Used to run a command as another user.
grep	Used to search for specific strings of characters within a file or other commands' outputs. To search through the output of a previous command, **grep** must be piped at the end of the previous command.
ifconfig	Used to display or configure network card–related information. If issued without parameters, **ifconfig** will display the configuration of the current network card(s).
apt-get	Used to install, configure, and remove packages on Debian and itsderivatives. Note: **apt-get** is a user-friendly command line front end for **dpkg,** Debian's package manager. The combo **dpkg** and **apt-get** is the default package manager system in all Debian Linux derivatives, including Raspbian.
iwconfig	Used to display or configure wireless network card–related information. Similar to **ifconfig, iwconfig** will display wireless information when issued without parameters.
shutdown	Used to shut down the system. **shutdown** can be instructed to perform a number of shutdown-related tasks, including restart, halt, put to sleep, or kick out all currently connected users.
passwd	Used to change the password. If no parameters are provided, **passwd** changes the password for the current user.
cat	Used to list the contents of a file and expects the filename as the parameter. The **cat** command is usually used on text files.
man	Used to display the documentation for a specific command.

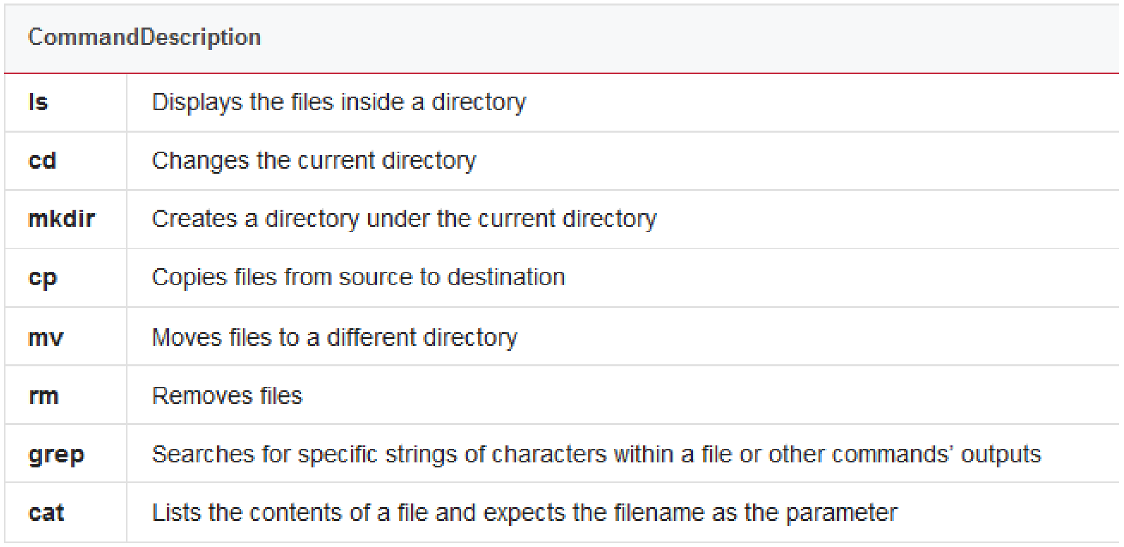
Note

The text here assumes the user has the proper permissions to execute the command. File permissions in Linux are covered later in this chapter.

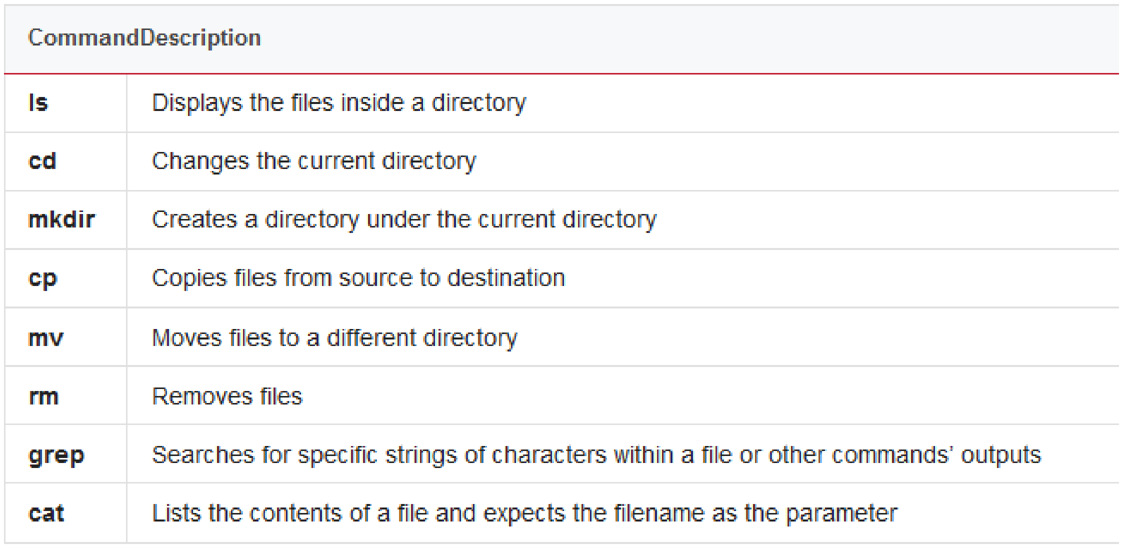
File and Directory Commands (3.1.2.3)

Many command line tools are included in Linux by default. To adjust the command operation, users can pass parameters and switches along with the command. Table 3-2 shows a few of the most common commands related to files and directories.

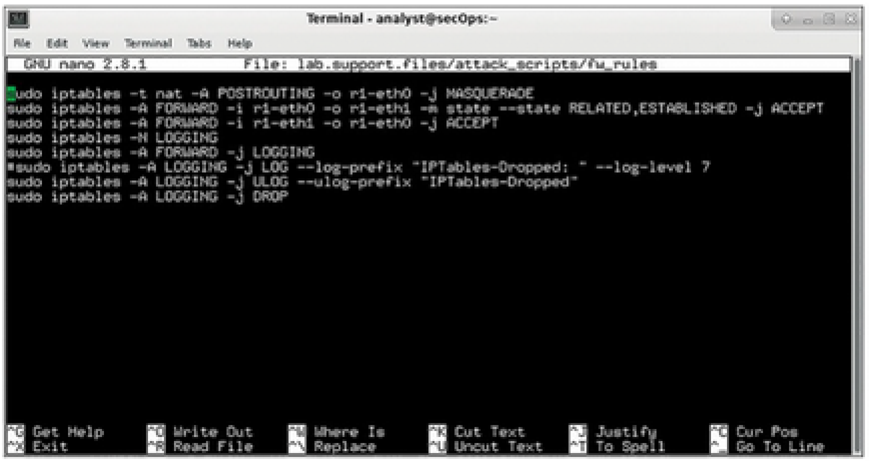
Table 3-2 Common File and Directory Commands



CommandDescription	
ls	Displays the files inside a directory
cd	Changes the current directory
mkdir	Creates a directory under the current directory
cp	Copies files from source to destination
mv	Moves files to a different directory
rm	Removes files
grep	Searches for specific strings of characters within a file or other commands' outputs
cat	Lists the contents of a file and expects the filename as the parameter



CommandDescription	
ls	Displays the files inside a directory
cd	Changes the current directory
mkdir	Creates a directory under the current directory
cp	Copies files from source to destination
mv	Moves files to a different directory
rm	Removes files
grep	Searches for specific strings of characters within a file or other commands' outputs
cat	Lists the contents of a file and expects the filename as the parameter


Terminal - analyst@secOps:~
File Edit View Terminal Tabs Help

```
  GNU nano 2.8.1            File: lab.support.files/attack_scripts/fw_rules

sudo iptables -t nat -A POSTROUTING -o r1-eth0 -j MASQUERADE
sudo iptables -A FORWARD -i r1-eth0 -o r1-eth1 -m state --state RELATED,ESTABLISHED -j ACCEPT
sudo iptables -A FORWARD -i r1-eth1 -o r1-eth0 -j ACCEPT
sudo iptables -N LOGGING
sudo iptables -A FORWARD -j LOGGING
#sudo iptables -A LOGGING -j LOG --log-prefix "IPTables-Dropped: " --log-level 7
sudo iptables -A LOGGING -j ULOG --ulog-prefix "IPTables-Dropped"
sudo iptables -A LOGGING -j DROP

^G Get Help     ^O Write Out    ^W Where Is     ^K Cut Text     ^J Justify      ^C Cur Pos
^X Exit         ^R Read File    ^\ Replace      ^U Uncut Text   ^T To Spell     ^_ Go To Line
```

Figure 3-4 The nano Text Editor

Due to the lack of graphical support, nano can only be controlled with the keyboard. For example, **CTRL-O** saves the current file; **CTRL-W** opens the search menu. GNU nano uses a two-line shortcut bar at the bottom of the screen, where commands for the current context are listed. Press **CTRL-G** for the help screen and a complete list of commands.

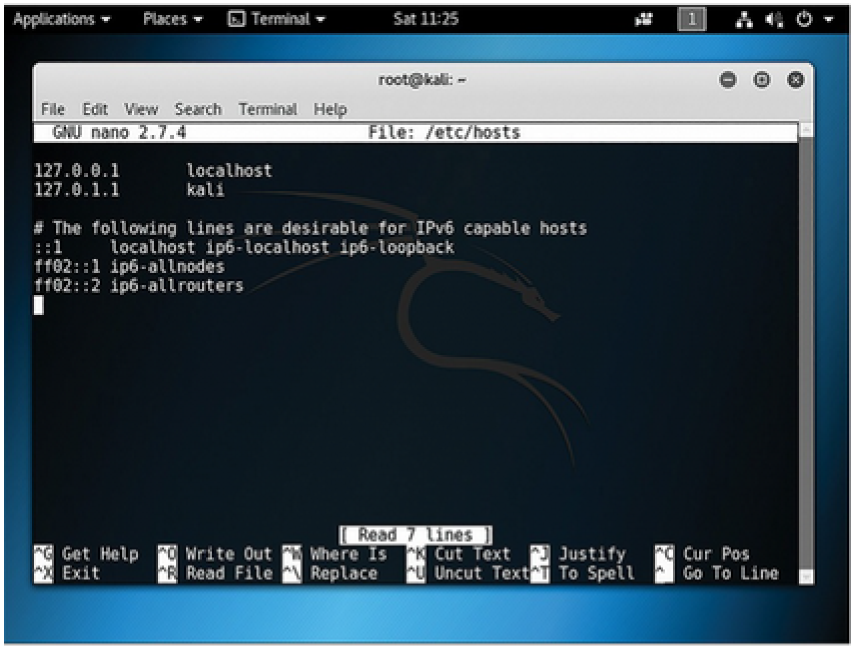
The Importance of Text Files in Linux (3.1.2.5)

In Linux, everything is treated as a file, including the memory, the disks, the monitor, the files, and the directories. For example, from the operating system standpoint, showing information on the display means to write to the file that represents the display device. It should be no surprise that the computer itself is configured through files. Known as configuration files, they are usually text files used to store adjustments and settings for specific applications or services.

Practically everything in Linux relies on configuration files to work. Some services have not one but several configuration files.

Users with proper permission levels can use text editors to change the contents of configuration files. After the changes are made, the file is saved and can be used by the related service or application. Users are able to specify exactly how they want any given application or service to behave. When launched, servicesand applications check the contents of specific configuration files to adjust their behavior accordingly.

In Figure 3-5, the administrator opened the host configuration file in nano for editing. Only the superuser can change the host file.



```
  GNU nano 2.7.4                 File: /etc/hosts

127.0.0.1       localhost
127.0.1.1       kali

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters


                              [ Read 7 lines ]
^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Uncut Text^T To Spell  ^_ Go To Line
```

Figure 3-5 Editing a Text File in nano

Note

The administrator used the command **sudo nano /etc/hosts** to open the file. The command **sudo** (short for "superuser do") invokes the superuser

Lab 3.1.2.6: Working with Text Files in the CLI

In this lab, you will get familiar with Linux command-line text editors and configuration files.

Lab 3.1.2.7: Getting Familiar with the Linux Shell

In this lab, you will use the Linux command line to manage files and folders and perform some basic administrative tasks.

Linux Servers and Clients (3.1.3)

In this topic, you will learn about Linux client-server communications.

An Introduction to Client-Server Communications (3.1.3.1)

Servers are computers with software installed that enables them to provide services to clients. There are many types of services. Some provide resources such as files, email messages, or web pages to clients upon request. Other services run maintenance tasks such as log management, memory management, disk scanning, and more. Each service requires separate server software. For example, the server in Figure 3-6 requires file server software to provide clients with the ability to retrieve and submit files.

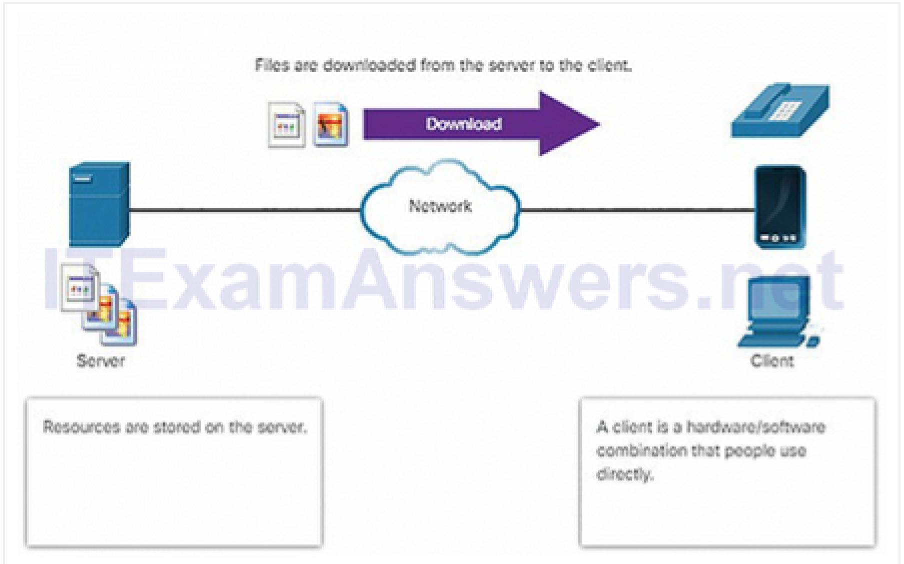

Files are downloaded from the server to the client.
Download
Network
Server
Client
Resources are stored on the server.
A client is a hardware/software combination that people use directly.

Figure 3-6 Server Sending Files to Client

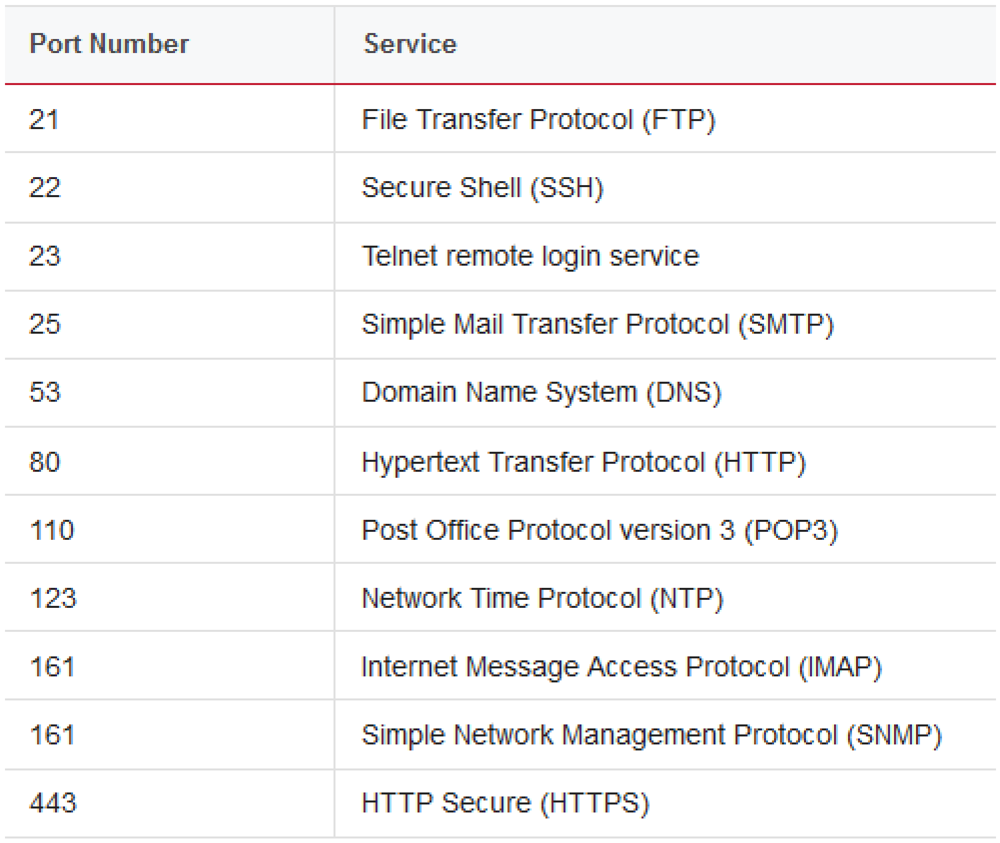
Client-server communications is discussed in more detail later in the course.

Servers, Services, and Their Ports (3.1.3.2)

For a computer to be the server for multiple services, ports are used. A port is a reserved network resource used by a service. A server is said to be "listening" on a port when it has associated itself to that port.

While the administrator can decide which port to use with any given service, many clients are configured to use a specific port by default. To make it easier for the client, it is common practice to leave the service running in its default port. Table 3-3 shows a few commonly used ports and their services. These are also called "well-known ports."

Table 3-3 Common Port Numbers and Services



Port Number	Service
21	File Transfer Protocol (FTP)
22	Secure Shell (SSH)
23	Telnet remote login service
25	Simple Mail Transfer Protocol (SMTP)
53	Domain Name System (DNS)
80	Hypertext Transfer Protocol (HTTP)
110	Post Office Protocol version 3 (POP3)
123	Network Time Protocol (NTP)
161	Internet Message Access Protocol (IMAP)
161	Simple Network Management Protocol (SNMP)
443	HTTP Secure (HTTPS)

Ports and their uses in network communications are discussed in more detail later in the course.

Clients (3.1.3.3)

Clients are programs or applications designed to communicate with a specific server. Also known as client applications, clients use a well-defined protocol to communicate with the server. Web browsers are web clients used to communicate with web servers via the Hypertext Transfer Protocol (HTTP). The File Transfer Protocol (FTP) client is software used to communicate with an FTP server. Figure 3-7 shows a client uploading files to a server.

Lab 3.1.3.4: Linux Servers

In this lab, you will use the Linux command line to identify servers running on a given computer.

Linux Administration (3.2)

In this section, you will learn about basic Linux server administration and the Linux file system.

Basic Server Administration (3.2.1)

In this topic, you will learn about Linux server configuration files, hardening Linux servers, and monitoring Linux services.

Service Configuration Files (3.2.1.1)

In Linux, services are managed using configuration files. Common options are port number, location of the hosted resources, and client authorization details. When the service starts, it looks for its configuration files, loads them into memory, and adjusts itself according to the settings in the files. Configurationfile modifications often require restarting the service before the changes take effect.

Because services often require superuser privileges to run, service configuration files often require superuser privileges for editing.

Example 3-1 shows a portion of the configuration file for Nginx, a lightweight web server for Linux.

Example 3-1 Nginx Web Server Configuration File

```
[analyst@secOps ~]$ cat /etc/nginx/nginx.conf

#user html;
worker_processes 1;

#error_log logs/error.log;
#error_log logs/error.log notice;
#error_log logs/error.log info;

#pid logs/nginx.pid;

events {
worker_connections 1024;
}

http {
include mime.types;
default_type application/octet-stream;

#log_format main '$remote_addr - $remote_user [$time_local]
"$request" '
# '$status $body_bytes_sent "$http_referer" '
# '"$http_user_agent" "$http_x_forwarded_for"';

#access_log logs/access.log main;

sendfile on;#tcp_nopush on;

#keepalive_timeout 0;
keepalive_timeout 65;

#gzip on;

<output omitted>
```

Example 3-2 shows the configuration file for the Network Time Protocol, NTP.

Example 3-2 NTP Configuration File

```
[analyst@secOps ~]$ cat ls /etc/ntp.conf
cat: ls: No such file or directory
# Please consider joining the pool:

#
# http://www.pool.ntp.org/join.html
#
# For additional information see:
# - https://wiki.archlinux.org/index.php/Network_Time_Protocol_daemon
# - http://support.ntp.org/bin/view/Support/GettingStarted
# - the ntp.conf man page

# Associate to Arch's NTP pool
server 0.arch.pool.ntp.org
server 1.arch.pool.ntp.org
server 2.arch.pool.ntp.org
server 3.arch.pool.ntp.org

# By default, the server allows:
# - all queries from the local host
# - only time queries from remote hosts, protected by rate limiting
and kod
restrict default kod limited nomodify nopeer noquery notrap
restrict 127.0.0.1
restrict ::1
# Location of drift file
driftfile /var/lib/ntp/ntp.drift
[analyst@secOps ~]$
```

Example 3-3 shows the configuration file for Snort, a Linux-based intrusion detection system (IDS).

Example 3-3 Snort Configuration File

```
[analyst@secOps ~]$ cat /etc/snort/snort.conf
#--------------------------------------------------
# VRT Rule Packages Snort.conf
#
# For more information visit us at:
# http://www.snort.org Snort Website
# http://vrt-blog.snort.org/ Sourcefire VRT Blog
#
# Mailing list Contact: snort-sigs@lists.sourceforge.net
# False Positive reports: fp@sourcefire.com
# Snort bugs: bugs@snort.org
#
# Compatible with Snort Versions:
# VERSIONS : 2.9.9.0
#
# Snort build options:
# OPTIONS : --enable-gre --enable-mpls --enable-targetbased --enableppm
--enable-perfprofiling --enable-zlib --enable-active-response --
enable-normalizer
--enable-reload --enable-react --enable-flexresp3
#
# Additional information:
# This configuration file enables active response, to run snort in
# test mode -T you are required to supply an interface -i
# or test mode will fail to fully validate the configuration and
# exit with a FATAL error
#--------------------------------------------------
###################################################
# This file contains a sample snort configuration.
# You should take the following steps to create your own custom
configuration:
```

```
#
# 1) Set the network variables.
# 2) Configure the decoder
# 3) Configure the base detection engine
# 4) Configure dynamic loaded libraries
# 5) Configure preprocessors
# 6) Configure output plugins
# 7) Customize your rule set
# 8) Customize preprocessor and decoder rule set
# 9) Customize shared object rule set
###################################################

###################################################
# Step #1: Set the network variables. For more information, see
README.variables
###################################################
# Setup the network addresses you are protecting
###ipvar HOME_NET any
###ipvar HOME_NET [192.168.0.0/24,192.168.1.0/24]
ipvar HOME_NET [209.165.200.224/27]
# Set up the external network addresses. Leave as "any" in most
situations
ipvar EXTERNAL_NET any
# List of DNS servers on your network
###ipvar DNS_SERVERS $HOME_NET
ipvar DNS_SERVERS 209.165.200.236
# List of SMTP servers on your network
###ipvar SMTP_SERVERS $HOME_NET
ipvar SMTP_SERVERS 209.165.200.236
# List of web servers on your network
###ipvar HTTP_SERVERS $HOME_NETipvar HTTP_SERVERS 209.165.200.235
# List of sql servers on your network
###ipvar SQL_SERVERS $HOME_NET
ipvar SQL_SERVERS 209.165.200.235
# List of telnet servers on your network
###ipvar TELNET_SERVERS $HOME_NET
ipvar TELNET_SERVERS 209.165.200.236
# List of ssh servers on your network
###ipvar SSH_SERVERS $HOME_NET
```

```
ipvar SSH_SERVERS 209.165.200.236
<output omitted>
[analyst@secOps ~]$
```

There is no rule for a configuration file format; it is the choice of the service's developer. However, the **option = value** format is often used. In Example 3-3, variable **ipvar** is configured with several options. The first option, HOME_NET, has the value 209.165.200.224/27.

Hardening Devices (3.2.1.2)

Device hardening involves implementing proven methods of securing the device and protecting its administrative access. Some of these methods involve maintaining passwords, configuring enhanced remote login features, and implementing SSH. Defining administrative roles in terms of access is another important aspect of securing infrastructure devices because not all information technology personnel should have the same level of access to the infrastructure devices.

Depending on the Linux distribution, many services are enabled by default. Some of these features are enabled for historical reasons, but are no longer required. Stopping such services and ensuring they do not automatically start at boot time is another device hardening technique.

OS updates are also extremely important to maintaining a hardened device. New vulnerabilities are discovered every day. OS developers create and issue fixes and patches regularly. An up-to-date computer is less likely to be compromised.The following list provides a few basic recommended steps for device hardening:

- Ensure physical security.
- Minimize installed packages.
- Disable unused services.
- Use SSH and disable the root account login over SSH.
- Keep the system updated.
- Disable USB auto-detection.
- Enforce strong passwords.
- Force periodic password changes.
- Keep users from reusing old passwords.
- Review logs regularly.

Many other steps exist and are often service- or application-dependent.

Monitoring Service Logs (3.2.1.3)

Log files are the records that a computer stores to keep track of important events.

Kernel, services, and applications events are all recorded in log files. It is very important for an administrator to periodically review the logs of a computer to keep it healthy. By monitoring Linux log files, an administrator gains a clear picture of the computer's performance, security status, and any underlying issues. Log file analysis allows an administrator to guard against upcoming issues before they occur.

In Linux, log files can be categorized as

- Application logs
- Event logs
- Service logs
- System logs

Some logs contain information about daemons that are running in the Linux system. A daemon is a background process that runs without the need for user interaction. For example, the System Security Services Daemon (SSSD)manages remote access and authentication for single sign-on capabilities. The following are a few popular Linux log files and their functions:

- **/var/log/messages:** This directory contains generic computer activity logs. It is mainly used to store informational and noncritical system messages. In Debian-based computers, the /var/log/syslog directory serves the same purpose.
- **/var/log/auth.log:** This file stores all authentication-related events in Debian and Ubuntu computers. Anything involving the user authorization mechanism can be found in this file.
- **/var/log/secure:** This directory is used by Red Hat and CentOS computers instead of /var/log/auth.log. It also tracks sudo logins, SSH logins, and other errors logged by SSSD.
- **/var/log/boot.log:** This file stores boot-related information and messages logged during the computerstartup process.
- **/var/log/dmesg:** This directory contains kernel ring buffer messages. Information related to hardware devices and their drivers is recorded here. It is very important because, due to their low-level nature, logging systems such as syslog are not running when these events take place and, therefore, are often unavailable to the administrator in real time.
- **/var/log/kern.log:** This file contains information logged by the kernel.
- **/var/log/cron:** Cron is a service used to schedule automated tasks in Linux and this directory stores its events. Whenever a scheduled task (also called a cron job) runs, all its relevant information including execution status and error messages are stored here.
- **/var/log/mysqld.log or /var/log/mysql.log:** This is the MySQL log file. All debug, failure, and success messages related to the mysqld process and mysqld_safe daemon are

logged here. Red Hat, CentOS, and Fedora store MySQL logs under /var/log/mysqld.log, while Debian and Ubuntu maintain the log in the /var/log/mysql.log file.

Example 3-4 shows a portion of the /var/log/syslog log file. Each line represents a logged event. The timestamps at the beginning of the lines mark the moment the event took place.

Example 3-4 Output of /var/log/syslog

```
[analyst@secOps]$ cat /var/log/syslog
Nov 15 09:17:13 secOps kernel: [ 0.000000] Linux version 4.10.10-1-
ARCH
(builduser@tobias) (gcc versi
on 6.3.1 20170306 (GCC) ) #1 SMP PREEMPT Wed Apr 12 19:10:48 CEST
2017
Nov 15 09:17:13 secOps kernel: [ 0.000000] ------------[ cut here ]--
----------
Nov 15 09:17:13 secOps kernel: [ 0.000000] WARNING: CPU: 0 PID: 0 at
arch/x86/
kernel/fpu/xstate.c:595
fpu__init_system_xstate+0x465/0x7b2
Nov 15 09:17:13 secOps kernel: [ 0.000000] XSAVE consistency problem,
dumping
leaves
Nov 15 09:17:13 secOps kernel: [ 0.000000] Modules linked in:
Nov 15 09:17:13 secOps kernel: [ 0.000000] CPU: 0 PID: 0 Comm:
swapper Not
tainted 4.10.10-1-ARCH #1
Nov 15 09:17:13 secOps kernel: [ 0.000000] Call Trace:
Nov 15 09:17:13 secOps kernel: [ 0.000000] dump_stack+0x58/0x74
Nov 15 09:17:13 secOps kernel: [ 0.000000] __warn+0xea/0x110
Nov 15 09:17:13 secOps kernel: [ 0.000000] ?
fpu__init_system_xstate+0x465/0x7b2
Nov 15 09:17:13 secOps kernel: [ 0.000000]
warn_slowpath_fmt+0x46/0x60
Nov 15 09:17:13 secOps kernel: [ 0.000000]
fpu__init_system_xstate+0x465/0x7b2
Nov 15 09:17:13 secOps kernel: [ 0.000000]
fpu__init_system+0x18c/0x1b1
Nov 15 09:17:13 secOps kernel: [ 0.000000] early_cpu_init+0x110/0x113
Nov 15 09:17:13 secOps kernel: [ 0.000000] setup_arch+0xe4/0xbb6
Nov 15 09:17:13 secOps kernel: [ 0.000000] start_kernel+0x8f/0x3ce
Nov 15 09:17:13 secOps kernel: [ 0.000000]
i386_start_kernel+0x91/0x95
Nov 15 09:17:13 secOps kernel: [ 0.000000] startup_32_smp+0x16b/0x16d
Nov 15 09:17:13 secOps kernel: [ 0.000000] ---[ end trace
c61a827435bb526d ]---
<output omitted>
```

Lab 3.2.1.4: Locating Log Files

In this lab, you will get familiar with locating and manipulating Linux log files.

The Linux File System (3.2.2)

In this topic, you will learn about Linux file system types, Linux roles and file permissions, and creating hard and symbolic links.

The File System Types in Linux (3.2.2.1)

There are many different kinds of file systems, varying in properties of speed, flexibility, security, size, structure, logic, and more. It is up to the administrator to decide which file system type best suits the operating system and the files it will store. Below are a few file system types commonly found and supported by Linux:

- **ext2 (second extended file system):** ext2 was the default file system in several major Linux distributions until supplanted by ext3. Almost fully compatible with ext2, ext3 also supports journaling (see below). ext2 is still the file system of choice for flash-based storage media because its lack of a journal increases performance and minimizes the number of writes. Because flash memory devices have a limited number of write operations, minimizing write operations increases the device's lifetime. However, contemporary Linux kernels also support ext4, an even more modern file system, with better performance and which can also operate in a journal-less mode.
- **ext3 (third extended file system):** ext3 is a journaled file system designed to improve the existing ext2 file system. A journal, the main feature added to ext3, is a technique used to minimize the risk of file system corruption in the event of sudden power loss. The file system keeps a log (or journal) of all the file system changes about to be made. If the computer crashes before the change is complete, the journal can be used to restore or correct any eventual issues created by the crash. The maximum file size in ext3 file systems is 32 TB.
- **ext4 (fourth extended file system):** Designed as a successor of ext3, ext4 was created based on a series of extensions to ext3. While the extensions improve the performance of ext3 and increase supported file sizes, Linux kernel developers were concerned about stability issues and were opposed to adding the extensions to the stable ext3. The ext3 project was split in two; one kept as ext3 and its normal development and the other, named ext4, incorporated the mentioned extensions.
- **NFS (Network File System):** NFS is a network-based file system, allowing file access over the network. From the user standpoint, there is no difference between accessing a file stored locally or on another computer on the network. NFS is an open standard which allows anyone to implement it.
- **CDFS (Compact Disc File System):** CDFS was created specifically for optical disk media.
- **Swap file system**: The swap file system is used by Linux when it runs out of RAM. Technically, it is a swap partition that does not have a specific file system but it is relevant to the file system discussion. When this happens, the kernel moves inactive RAM content to the swap partition on the disk.
  While swap partitions (also known as swap space) can be useful to Linux computers with a limited amount of memory, they should not be considered as a primary solution. A swap partition is stored on disk, which has much lower access speeds than RAM.
- HFS Plus or **HFS+ (Hierarchical File System Plus):** Primary file system used by Apple in its Macintosh computers. The Linux kernel includes a module for mounting HFS+ for read-write operations.
- Master boot record (MBR): Located in the first sector of a partitioned computer, the MBR stores all the information about the way in which the file system is organized. The MBR quickly hands over control to a loading function, which loads the OS.

Mounting is the term used for the process of assigning a directory to a partition. After a successful mount operation, the file system contained on the partition is accessible through the specified directory. In this context, the directory is called the mounting point for that file system. Windows users may be familiar with a similar concept: the drive letter.

Example 3-5 shows the output of the **mount** command issued in the Cisco CyberOPS VM.

Example 3-5 The Output of mount in the CyberOPS VM

```
[analyst@secOps ~]$ mount
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
sys on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
dev on /dev type devtmpfs
(rw,nosuid,relatime,size=511056k,nr_inodes=127764,
mode=755)
run on /run type tmpfs (rw,nosuid,nodev,relatime,mode=755)
/dev/sda1 on / type ext4 (rw,relatime,data=ordered)
securityfs on /sys/kernel/security type securityfs
(rw,nosuid,nodev,noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
devpts on /dev/pts type devpts
(rw,nosuid,noexec,relatime,gid=5,mode=620,ptmxmode=000)
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,mode=755)
cgroup on /sys/fs/cgroup/systemd type cgroup
(rw,nosuid,nodev,noexec,relatime,xattr,release_agent=/usr/lib/systemd/
systemd-cgroups-agent,name=systemd)
pstore on /sys/fs/pstore type pstore
(rw,nosuid,nodev,noexec,relatime)
cgroup on /sys/fs/cgroup/perf_event type cgroup
(rw,nosuid,nodev,noexec,relatime,
perf_event)
cgroup on /sys/fs/cgroup/freezer type cgroup
(rw,nosuid,nodev,noexec,relatime,freezer)
cgroup on /sys/fs/cgroup/net_cls type cgroup
(rw,nosuid,nodev,noexec,relatime,net_cls)
cgroup on /sys/fs/cgroup/cpuset type cgroup
(rw,nosuid,nodev,noexec,relatime,cpuset)
cgroup on /sys/fs/cgroup/devices type cgroup
(rw,nosuid,nodev,noexec,relatime,devices)
cgroup on /sys/fs/cgroup/blkio type cgroup
(rw,nosuid,nodev,noexec,relatime,blkio)
cgroup on /sys/fs/cgroup/pids type cgroup
(rw,nosuid,nodev,noexec,relatime,pids)
cgroup on /sys/fs/cgroup/cpu,cpuacct type cgroup
(rw,nosuid,nodev,noexec,relatime,
cpu,cpuacct)cgroup on /sys/fs/cgroup/memory type cgroup
(rw,nosuid,nodev,noexec,relatime,memory)
systemd-1 on /proc/sys/fs/binfmt_misc type autofs
(rw,relatime,fd=28,pgrp=1,
timeout=0,minproto=5,maxproto=5,direct)
hugetlbfs on /dev/hugepages type hugetlbfs (rw,relatime)
tmpfs on /tmp type tmpfs (rw,nosuid,nodev)
mqueue on /dev/mqueue type mqueue (rw,relatime)
```

```
debugfs on /sys/kernel/debug type debugfs (rw,relatime)
configfs on /sys/kernel/config type configfs (rw,relatime)
tmpfs on /run/user/1000 type tmpfs
(rw,nosuid,nodev,relatime,size=102812k,mode=700,
uid=1000,gid=1000)
[analyst@secOps ~]$
```

When issued with no options, **mount** returns the list of file systems currently mounted in a Linux computer. While many files systems shown are out of the scope of this course, notice the root file system (highlighted). The root file system is represented by the "/" symbol and holds all files in the computer by default. It is also shown in Example 3-5 that the root file system was formatted as ext4 and occupies the first partition of the first drive (/dev/sda1).

Linux Roles and File Permissions (3.2.2.2)

In Linux, most system entities are treated as files. In order to organize the system and reinforce boundaries within the computer, Linux uses file permissions. File permissions are built into the file system structure and provide a mechanism to define permissions on every file. Every file in

Linux carries its file permissions, defining the actions that the owner, the group, and others can do with the file.

The possible permission rights are Read, Write, and Execute. The **ls** command with the **-l** parameter lists additional information about the file. Consider the output of the **ls -l** command in Example 3-6.

Example 3-6 Viewing Permissions for a Linux File

```
[analyst@secOps ~]$ ls -l space.txt
-rwxrw-r-- 1 analyst staff 253 May 20 12:49 space.txt
[analyst@secOps ~]$
```

The output provides a lot of information about the file space.txt.

The first field of the output displays the permissions associated to space.txt **(-rwxrw-r–).** File permissions are always displayed in the User, Group, and Other order, so you can interpret the first field as follows:

- The dash (-) means that this is a file. For directories, the first dash would be a "d" instead.
- The first set of characters is for user permissions (rwx). The user, analyst, who owns the file can Read, Write, and eXecute the file.
- The second set of characters is for group permissions (rw-). The group, staff, who owns the file can Read and Write to the file.
- The third set of characters is for any other user or group permissions (r–). Any other user or group on the computer can only Read the file.

The second field defines the number of hard links to the file (the number 1 after the permissions). A hard link creates another file with a different name linked to the same place in the file system (called an inode). This is in contrast to a symbolic link, which is discussed next.

The third and fourth field display the user **(analyst)** and group **(staff)** who own the file, respectively.

The fifth field displays the file size in bytes. The space.txt file has 253 bytes.

The sixth field displays the date and time of the last modification.

The seventh field displays the filename.

Figure 3-8 shows a breakdown of file permissions in Linux.

Figure 3-8 shows a breakdown of file permissions in Linux.

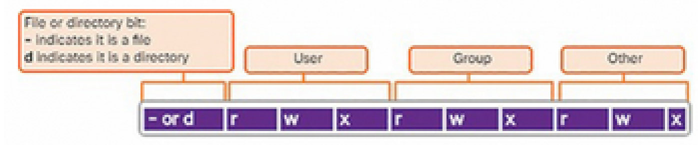


Figure 3-8 File Permissions

Table 3-4 shows how to interpret the octal values for permissions.

Table 3-4 Octal Values for Permissions

Binary	Octal	Permission	Description
000	0	—	No access
001	1	–x	Execute only
010	2	-w-	Write only
011	3	-wx	Write and Execute
100	4	r–	Read only
101	5	r-x	Read and Execute
110	6	rw-	Read and Write
111	7	rwx	Read, Write, and Execute

File permissions are a fundamental part of Linux and cannot be broken. A user has as much rights to a file as the file permissions allow. The only user that can override file permission on a Linux computer is the root user. Because the root user has the power to override file permissions, the root user can write to any file. Because everything is treated as a file, the root user has full control over a Linux computer. Root access is often required before performing maintenance and administrative tasks.

Hard Links and Symbolic Links (3.2.2.3)

A hard link is another file that points to the same location as the original file.

Use the command In to create a hard link. The first argument is the existing file and the second argument is the new file. The file space.txt is linked to space.hard.txt in Example 3-7 and the link field now shows 2.

Example 3-7 Creating Hard Links in Linux

```
[analyst@secOps ~]$ ln space.txt space.hard.txt
[analyst@secOps ~]$ ls -l space*
-rwxrw-r-- 2 analyst staff 253 May 20 14:41 space.hard.txt
-rwxrw-r-- 2 analyst staff 253 May 20 14:41 space.txt
[analyst@secOps ~]$ echo "Testing hard link" >> space.txt
[analyst@secOps ~]$ ls -l space*
-rwxrw-r-- 2 analyst staff 273 May 20 14:41 space.hard.txt
-rwxrw-r-- 2 analyst staff 273 May 20 14:41 space.txt
[analyst@secOps ~]$ rm space.hard.txt
[analyst@secOps ~]$ more space.txt
"Space is big. Really big. You just won't believe how vastly, hugely,
mindbogglingly big it is. I mean, you may think it's a long way down the
road to the
chemist, but that's just peanuts in space."
--Douglas Adams, The Hitchhiker's Guide to the Galaxy


Testing hard link
```

```
[analyst@secOps ~]$
```

Both files point to the same location in the file system. If you change one file, the other is changed, as well. The **echo** command is used to add some text to **space.txt.** Notice that the file size for both **space.txt** and **space.hard.txt** increased to 273 bytes. If you delete the space.hard.txt file with the **rm** command (remove), the **space.txt** file still exists, as verified with the **more space.txt** command.

A symbolic link, also called a symlink or soft link, is similar to a hard link in that applying changes to the symbolic link will also change the original file. As shown in Example 3-8, use the **ln** command option -s to create a symbolic link. Notice that adding a line of text to **test.txt** also adds the line to **mytest.txt.**

However, unlike a hard link, deleting the original **test.txt** file means that **mytest.txt** is now linked to a file that no longer exists, as shown with the **more mytest.txt** and **ls -l mytest.txt** commands.

Example 3-8 Creating Symbolic Links in Linux

```
[analyst@secOps ~]$ echo "Hello World!" > test.txt[analyst@secOps ~]$ ln -s
test.txt mytest.txt
[analyst@secOps ~]$ echo "It's a lovely day!" >> mytest.txt
[analyst@secOps ~]$ more test.txt
Hello World!
It's a lovely day!
[analyst@secOps ~]$ more mytest.txt
Hello World!
It's a lovely day!
[analyst@secOps ~]$ rm test.txt
[analyst@secOps ~]$ more mytest.txt
more: stat of mytest.txt failed: No such file or directory
[analyst@secOps ~]$ ls -l mytest.txt
lrwxrwxrwx 1 analyst staff 8 May 20 15:15 mytest.txt -> test.txt
[analyst@secOps ~]$
```

Although symbolic links have a single point of failure (the underlying file), symbolic links have several benefits over hard links:

- Locating hard links is more difficult. Symbolic links show the location of the original file in the **ls -l** command, as shown in the last line of output in Example 3-8 (**mytest.txt -> test.txt**).
- Hard links are limited to the file system in which they are created. Symbolic links can link to a file in another file system.
- Hard links cannot link to a directory because the system itself uses hard links to define the hierarchy of the directory structure. However, symbolic links can link to directories.

Lab 3.2.2.4: Navigating the Linux Filesystem and Permission Settings

In this lab, you will familiarize yourself with Linux filesystems.

Linux Hosts (3.3)

In this section, you will learn about working with Linux hosts through the GUI and the CLI.

Working with the Linux GUI (3.3.1)

In this topic, you will learn about the Linux GUI.

X Window System (3.3.1.1)

The graphical interface present in most Linux computers is based on the X Window System. Also known as X or X11, X Window is a windowing system designed to provide the basic framework for a GUI. X includes functions for drawing and moving windows on the display device and interacting with a mouse and keyboard.

X works as a server and, as such, allows a remote user to use the network to connect, start a graphical application, and have the graphical window open on the remote terminal. While the application itself runs on the server, the graphical aspect of it is sent by X over the network and displayed on the remote computer.

Notice that X does not specify the user interface, leaving it to other programs such as window managers to define all the graphical components. This abstraction allows for great flexibility and customization as graphical components such as buttons, fonts, icons, window borders, and color scheme are all defined by the user application. Because of this separation, the Linux GUI varies greatly from distribution to distribution. Examples of window managers are Gnome and KDE, as shown in Figures 3-9 and 3-10, respectively. While the look and feel of window managers vary, the main components are still present.

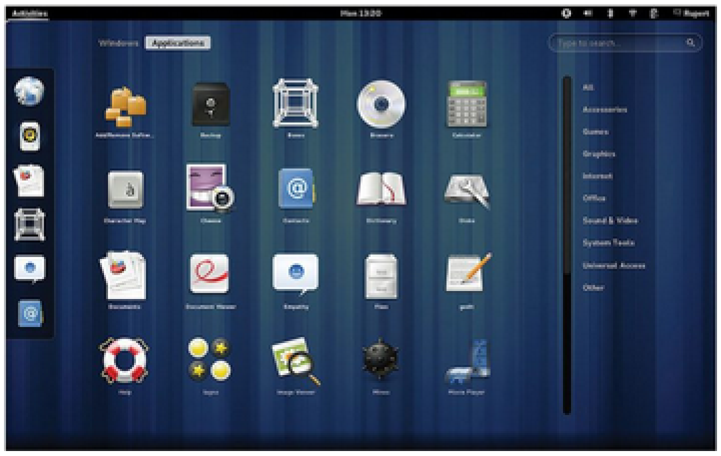


Figure 3-9 Gnome Window Manager

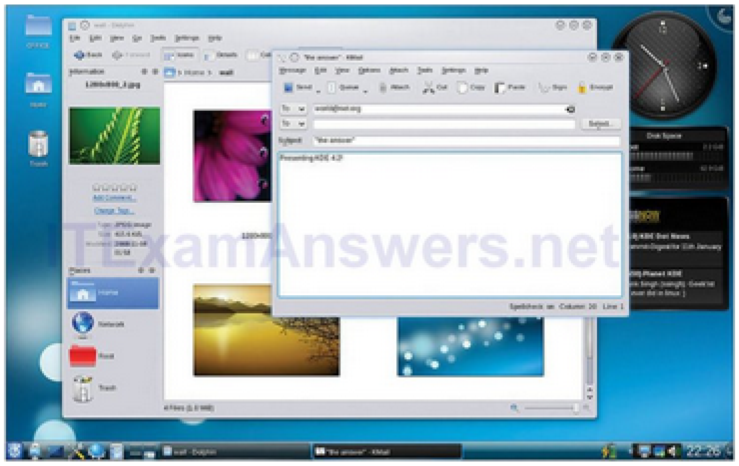


Figure 3-10 KDE Window Manager

For more information on Gnome, visit the following website:

https://www.gnome.org/

For more information on KDE, visit the following website:

https://www.kde.org/

The Linux GUI (3.3.1.2)

Although an operating system does not require a GUI to function, GUIs are considered more user-friendly than the CLI. The Linux GUI as a whole can be easily replaced by the user. As a result of the large number of Linux distributions, this chapter focuses on Ubuntu when covering Linux because it is a very popular and user-friendly distribution.

Ubuntu Linux uses Unity as its default GUI. Unity's goal is to make Ubuntu even more user-friendly. The main UI components of Unity include:

Top Menu Bar: This multipurpose menu bar contains the currently running application. It includes the maximize, minimize, and exit buttons of the application in focus, as well as the system toggles including settings, logout, and shutdown, clock, and other notifications.

Launcher: This is a dock on the left side of the screen that serves as the application launcher and switcher. Click to launch an application and when the application is running, click again to switch between running applications. If more than one instance of an application is running, Launcher will display all instances.

Quicklist: Right-click any application hosted on the Launcher to access a short list of tasks the application can perform.

Dash Search Box: This holds the Search tool and a list of recently used applications. Dash includes Lenses at the bottom of the Dash area which allow the user to fine-tune Dash search results. To access Dash, click the Ubuntu button on the top of the Launcher.

System and Notification Menu: Many important functions are located in the indicator menu, located at the top right corner of the screen. Use theindicator menu to switch users, shut down your computer, control the volume level, or change network settings.

Figure 3-11 shows a breakdown of the Ubuntu Unity Desktop.

Working on a Linux Host (3.3.2)

In this topic, you will learn how to install and run Linux applications, keep your system up to date, and guarding against malware on a Linux host.

Installing and Running Applications on a Linux Host (3.3.2.1)

Many end-user applications are complex programs written in compiled languages. To aid in the installation process, Linux often includes programs called package managers. A package is the term used to refer to a program andall its supported files. By using a package manager to install a package, all the necessary files are placed in the correct file system location.

There are several package managers. For this course, we will use the Advanced Packaging Tool (apt) package manager. Example 3-9 shows the output of a few **apt** commands. The **apt-get update** command is used to fetch the package list from the package repository and update the local package database. The **apt-get upgrade** command is used to update all currently installed packages to their latest versions.

Example 3-9 The Advanced Packaging Tool (APT) Package Manager

```
analyst@cuckoo:~$ sudo apt-get update
[sudo] password for analyst:
Hit:1 http://us.archive.ubuntu.com/ubuntu xenial InRelease
Get:2 http://us.archive.ubuntu.com/ubuntu xenial-updates InRelease
[102 kB]
Get:3 http://security.ubuntu.com/ubuntu xenial-security InRelease
[102 kB]
```

```
Get:4 http://us.archive.ubuntu.com/ubuntu xenial-backports InRelease
[102 kB]
Get:5 http://us.archive.ubuntu.com/ubuntu xenial-updates/main amd64
Packages [534 kB]
<output omitted>
Fetched 4,613 kB in 4s (1,003 kB/s)
Reading package lists... Done
analyst@cuckoo:~$
analyst@cuckoo:~$ sudo apt-get upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
The following packages have been kept back:
linux-generic-hwe-16.04 linux-headers-generic-hwe-16.04
linux-image-generic-hwe-16.04
The following packages will be upgraded:
firefox firefox-locale-en gir1.2-javascriptcoregtk-4.0 gir1.2-
webkit2-4.0
libjavascriptcoregtk-4.0-18libwebkit2gtk-4.0-37 libwebkit2gtk-4.0-37-gtk2 libxen-
4.6
libxenstore3.0 linuxlibc-dev logrotate openssh-client
qemu-block-extra qemu-kvm qemu-system-common qemu-system-x86 qemuutils snapd
ubuntu-core-launcher zlib1g
zlib1g-dev
21 upgraded, 0 newly installed, 0 to remove and 3 not upgraded.
Need to get 85.7 MB of archives.
After this operation, 1,576 kB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Keeping the System Up to Date (3.3.2.2)

Also known as patches, OS updates are released periodically by OS companies to address any known vulnerabilities in their operating system. While companies have update schedules, the release of unscheduled OS updates can happen when a major vulnerability is found in the OS code. Modern operating systems will alert the user when updates are available for download and installation but the user can check for updates at any time.

To update the local package metadata database using the CLI, use the **apt-get update** command.

To upgrade all the currently installed packages using the CLI, use the **apt-get upgrade** command.

To manually check and install updates on Linux using the GUI, click **Dash Search Box**, type **software updater**, and click the **Software Updater** icon, as shown in Figure 3-12.

Processes and Forks (3.3.2.3)

A process is a running instance of a computer program. Multitasking operating systems can execute many processes at the same time.

Forking is a method that the kernel uses to allow a process to create a copy of itself. Processes need a way to create new processes in multitasking operating systems. The fork operation is the only way of doing so in Linux.

Forking is important for many reasons. One of them relates to process scalability. Apache, a popular web server, is a good example. By forking itself, Apache is able to serve a large number of requests with fewer system resources than a single-process-based server.

When a process calls fork, the caller process becomes the parent process, with the newly created process referred to as its child. After the fork, the processes are, to some extent, independent processes; they have different process IDs but run the same program code.

The following are a few commands used to manage processes:

- **ps:** This command is used to list the processes running on the computer atthe time it is invoked. **ps** can be instructed to display running processes that belong to the current user or other users. While listing processes does not require root privileges, killing or modifying other users' processes does.
- **top:** This command is also used to list running processes, but unlike **ps, top** keeps displaying running processes dynamically. Press **q** to exit **top.**
- **kill:** This command is used to modify the behavior of a specific process. Depending on the parameters, **kill** will remove, restart, or pause a process. In many cases, the user will run **ps** or **top** before running **kill.** This is done so the user can learn the PID of a process before running **kill.**

Example 3-10 shows the output of the **top** command on a Linux computer.

Example 3-10 Output of the **top** Command

```
top - 12:37:51 up 28 min, 1 user, load average: 0.07, 0.02, 0.02
Tasks: 99 total, 1 running, 98 sleeping, 0 stopped, 0 zombie
%Cpu0 : 2.8/0.7 3[||| ]
GiB Mem : 94.6/0.981 [ ]
GiB Swap: 0.0/0.000 [ ]


PID USER PR NI VIRT RES %CPU %MEM TIME+ S COMMAND
1 root 20 0 8.9m 3.8m 0.0 0.4 0:00.70 S systemd
173 root 20 0 70.6m 2.4m 0.0 0.2 0:00.06 S `- systemd-journal
205 root 20 0 15.0m 1.8m 0.0 0.2 0:00.09 S `- systemd-udevd
270 root 20 0 5.5m 0.3m 0.0 0.0 0:00.09 S `- ovsdb-server
272 root 20 0 5.7m 0.9m 0.0 0.1 0:00.00 S `- start_pox.sh
281 root 20 0 42.0m 8.2m 0.7 0.8 0:03.47 S `- python2.7
274 root 20 0 23.2m 1.6m 0.0 0.2 0:00.00 S `- rsyslogd
276 root 20 0 7.0m 1.3m 0.0 0.1 0:00.00 S `- systemd-logind
277 dbus 20 0 6.4m 2.0m 0.0 0.2 0:00.18 S `- dbus-daemon
283 systemd+ 20 0 16.6m 0.5m 0.0 0.1 0:00.00 S `- systemd-network
284 root 20 0 7.5m 1.2m 0.0 0.1 0:00.00 S `- ovs-vswitchd
297 root 20 0 29.3m 1.5m 0.0 0.2 0:00.19 S `- VBoxService
314 root 20 0 5.2m 0.7m 0.0 0.1 0:00.00 S `- vsftpd
317 root 20 0 7.6m 0.9m 0.0 0.1 0:00.00 S `- sshd
320 root 20 0 35.3m 6.7m 0.0 0.7 0:00.04 S `- lightdm
332 root 20 0 164.3m 61.5m 2.6 6.1 0:05.76 S `- Xorg
385 root 20 0 31.2m 2.9m 0.0 0.3 0:00.01 S `- lightdm
396 analyst 20 0 5.5m 1.0m 0.0 0.1 0:00.00 S `- sh
416 analyst 20 0 75.7m 26.8m 0.0 2.7 0:00.07 S `- xfce4-session
426 analyst 20 0 60.0m 28.9m 0.0 2.9 0:00.41 S `- xfwm4
427 analyst 20 0 57.6m 25.6m 0.0 2.6 0:00.06 S `- Thunar
428 analyst 20 0 70.3m 31.9m 0.0 3.2 0:00.28 S `- xfce4-panel
459 analyst 20 0 56.7m 26.0m 0.0 2.6 0:00.08 S `- panel-6-systray
462 analyst 20 0 57.9m 25.5m 0.0 2.5 0:00.09 S `- panel-2-actions
432 analyst 20 0 90.2m 33.6m 0.0 3.3 0:00.57 S `- xfdesktop
444 analyst 20 0 78.5m 25.9m 0.0 2.6 0:00.06 S `- polkit-gnome-au
329 root 20 0 7.5m 0.5m 0.0 0.1 0:00.00 S `- nginx
330 http 20 0 8.8m 1.3m 0.0 0.1 0:00.00 S `- nginx
333 root 20 0 38.0m 2.8m 0.0 0.3 0:00.03 S `- accounts-daemon
340 polkitd 20 0 71.2m 10.3m 0.0 1.0 0:00.07 S `- polkitd
391 analyst 20 0 8.9m 1.8m 0.0 0.2 0:00.00 S `- systemd
392 analyst 20 0 12.2m 1.1m 0.0 0.1 0:00.00 S `- (sd-pam)
408 analyst 20 0 6.4m 1.8m 0.0 0.2 0:00.02 S `- dbus-daemon
420 analyst 20 0 10.2m 2.4m 0.0 0.2 0:00.01 S `- xfconfd
671 analyst 20 0 42.9m 6.4m 0.0 0.6 0:00.01 S `- at-spi-bus-laun
```

```
423 analyst 20 0 4.7m 0.2m 0.0 0.0 0:00.00 S `- ssh-agent
425 analyst 20 0 23.3m 0.2m 0.0 0.0 0:00.02 S `- gpg-agent
430 analyst 20 0 67.9m 26.3m 0.0 2.6 0:00.03 S `- xfsettingsd
440 analyst 20 0 80.0m 26.6m 0.0 2.6 0:00.08 S `- xfce4-power-man
448 analyst 20 0 79.8m 26.5m 0.0 2.6 0:00.02 S `- xfce4-power-man
463 root 20 0 52.6m 2.5m 0.0 0.2 0:00.02 S `- upowerd
478 analyst 20 0 15.2m 0.3m 0.0 0.0 0:00.00 S `- VBoxClient
487 analyst 20 0 17.4m 0.4m 0.7 0.0 0:01.78 S `- VBoxClient
479 analyst 20 0 15.2m 0.3m 0.0 0.0 0:00.00 S `- VBoxClient
484 analyst 20 0 16.9m 0.4m 0.0 0.0 0:00.01 S `- VBoxClient
```

Malware on a Linux Host (3.3.2.4)

Linux malware includes viruses, Trojan horses, worms, and other types of malware that can affect the operating system. Due to a number of design components such as file system structure, file permissions, and user account restrictions, Linux operating systems are generally regarded as better protected against malware.

While arguably better protected, Linux is not immune to malware. Many vulnerabilities have been found and exploited in Linux. These range from server software to kernel vulnerabilities. Attackers are able to exploit thesevulnerabilities and compromise the target. Due to the open source nature of Linux, fixes and patches are often made available within hours of the discovery of such problems.

If a malicious program is executed, it will cause damage, regardless of the platform. A common Linux attack vector is its services and processes. Vulnerabilities are frequently found in server and process code running on computers connected to the network. An outdated version of the Apache web server could contain an unpatched vulnerability which can be exploited by an attacker, for example. Attackers often probe open ports to assess the version and nature of the server running on that port. With that knowledge, attackers can research if there are any known issues with that particular version of that particular server to support the attack. As with most vulnerabilities, keeping the computer updated and closing any unused services and ports is a good way to reduce the opportunities for attack in a Linux computer.

Example 3-11 shows an attacker using the **telnet** command to probe the nature and version of a web server. The attacker has learned that the server in question is running nginx version 1.12.0. The next step would be to research known vulnerabilities in the nginx 1.12.0 code.

Note

You will learn more about this attack later in the course.

Example 3-11 Using **telnet** to Probe a Web Server

```
[analyst@secOps ~]$ telnet 209.165.200.224 80
Trying 209.165.200.224...
Connected to 209.165.200.224.
Escape character is '^]'.
type anything to force an HTTP error response
HTTP/1.1 400 Bad Request
Server: nginx/1.12.0
Date: Wed, 17 May 2017 14:27:30 GMT
Content-Type: text/html
Content-Length: 173
Connection: close
<html>
<head><title>400 Bad Request</title></head>
<body bgcolor="white">
<center><h1>400 Bad Request</h1></center>
<hr><center>nginx/1.12.0</center>
</body>
</html>
Connection closed by foreign host.
[analyst@secOps ~]$
```

Rootkit Check (3.3.2.5)

A rootkit is a set of software tools designed to increase a user's privileges, or grant access to portions of the software that should not normally be allowed. Rootkits are also often used to secure a backdoor to a compromised computer.

The installation of a rootkit can be automated (done as part of an infection) or an attacker can manually install it after compromising a computer. A rootkit is destructive because it changes kernel code and its modules, changing the most fundamental operations of the OS itself. With such a deep level of compromise, rootkits can hide the intrusion, remove any installation tracks, and even tamper with troubleshooting and diagnostics tools so that their output now hides the presence of the rootkit. While a few Linux vulnerabilities through history have allowed rootkit installation via regular user accounts, the vast majority of rootkit compromises require root or administrator access.

Because the very nature of the computer is compromised, rootkit detection can be very difficult. Typical detection methods often include booting the computer from trusted media such as a diagnostics operating system live CD. The compromised drive is mounted and, from the trusted system toolset, trusted diagnostic tools can be launched to inspect the compromised file system. Inspection methods include behavioral-based methods, signature scanning, difference scanning, and memory dump analysis.

Rootkit removal can be complicated and often impossible, especially in cases where the rootkit resides in the kernel; reinstallation of the operating system is usually the only real solution to the problem. Firmware rootkits usually require hardware replacement.

chkrootkit is a popular Linux-based program designed to check the computer for known rootkits. It is a shell script that uses common Linux tools such asstrings and **grep** to compare the signatures of core programs. It also looks for discrepancies as it traverses the /proc file system comparing the signatures found there with the output of the **ps** command For more information about **chkrootkit,** visit the following website:

http://www.chkrootkit.org/

While helpful, keep in mind that programs to check for rootkits are not 100% reliable.

Example 3-12 shows the output of **chkrootkit** on Ubuntu Linux.

Example 3-12 Output of the **chkrootkit** Command

```
analyst@cuckoo:~$ sudo ./chkrootkit
[sudo] password for analyst:
ROOTDIR is `/’
Checking `amd’... not found
Checking `basename’... not infected
Checking `biff’... not found
Checking `chfn’... not infected
Checking `chsh’... not infected
Checking `cron’... not infected
Checking `crontab’... not infected
Checking `date’... not infected
Checking `du’... not infected
Checking `dirname’... not infected
Checking `echo’... not infected
Checking `egrep’... not infected
Checking `env’... not infected
Checking `find’... not infected
Checking `fingerd’... not found
Checking `gpm’... not found
Checking `grep’... not infected
Checking `hdparm’... not infected
Checking `su’... not infected
Checking `ifconfig’... not infected
Checking `inetd’... not tested
Checking `inetdconf’... not found
Checking `identd’... not found
Checking `init’... not infected
Checking `killall’... not infected
Checking `ldsopreload’... not infected
Checking `login’... not infected
Checking `ls’... not infected
Checking `lsof’... not infected
Checking `mail’... not found
Checking `mingetty’... not found
Checking `netstat’... not infected
Checking `named’... not found
Checking `passwd’... not infected
Checking `pidof’... not infected
Checking `pop2’... not found
Checking `pop3’... not found
Checking `ps’... not infected
Checking `pstree’... not infected
Checking `rpcinfo’... not found
Checking `rlogind’... not found
Checking `rshd’... not found
Checking `slogin’... not infected
Checking `sendmail’... not found
Checking `sshd’... not infected
Checking `syslogd’... not tested
Checking `tar’... not infected
Checking `tcpd’... not infected
Checking `tcpdump’... not infected
Checking `top’... not infected
Checking `telnetd’... not found
Checking `timed’... not found
Checking `traceroute’... not found
Checking `vdir’... not infected
Checking `w’... not infected
Checking `write’... not infected
Checking `aliens’... no suspect files
Searching for sniffer’s logs, it may take a while... nothing found
Searching for HiDrootkit’s default dir... nothing found
Searching for t0rn’s default files and dirs... nothing found
Searching for t0rn’s v8 defaults... nothing found
Searching for Lion Worm default files and dirs... nothing found
Searching for RSHA’s default files and dir... nothing found
Searching for RH-Sharpe’s default files... nothing found
Searching for Ambient’s rootkit (ark) default files and dirs...
nothing found
Searching for suspicious files and dirs, it may take a while...
/usr/lib/debug/.build-id /lib/modules/4.8.0-36-generic/vdso/.build-id
/lib/
modules/4.8.0-52-generic/vdso/.build-id /lib/modules/4.8.0-49-
generic/vdso/.build-id
/usr/lib/debug/.build-id /lib/modules/4.8.0-36-generic/vdso/.build-id
/lib/
modules/4.8.0-52-generic/vdso/.build-id /lib/modules/4.8.0-49-
generic/vdso/.build-id
Searching for LPD Worm files and dirs... nothing found
Searching for Ramen Worm files and dirs... nothing found
Searching for Maniac files and dirs... nothing found
Searching for RK17 files and dirs... nothing found
Searching for Ducoci rootkit... nothing found
Searching for Adore Worm... nothing found
Searching for ShitC Worm... nothing found
Searching for Omega Worm... nothing found
Searching for Sadmind/IIS Worm... nothing found
Searching for MonKit... nothing found
Searching for Showtee... nothing found
Searching for OpticKit... nothing found
Searching for T.R.K... nothing found
Searching for Mithra... nothing found
Searching for LOC rootkit... nothing found
Searching for Romanian rootkit... nothing found
Searching for Suckit rootkit... nothing found
Searching for Volc rootkit... nothing found
Searching for Gold2 rootkit... nothing found
Searching for TC2 Worm default files and dirs... nothing found
Searching for Anonoying rootkit default files and dirs... nothing
found
Searching for ZK rootkit default files and dirs... nothing found
Searching for ShKit rootkit default files and dirs... nothing found
Searching for AjaKit rootkit default files and dirs... nothing found
Searching for zaRwT rootkit default files and dirs... nothing found
Searching for Madalin rootkit default files... nothing found
Searching for Fu rootkit default files... nothing found
Searching for ESRK rootkit default files... nothing found
Searching for rootedoor... nothing found
Searching for ENYELKM rootkit default files... nothing found
Searching for common ssh-scanners default files... nothing found
Searching for Linux/Ebury - Operation Windigo ssh... not tested
Searching for 64-bit Linux Rootkit ... nothing found
Searching for 64-bit Linux Rootkit modules... nothing found
Searching for Mumblehard Linux ... nothing found
Searching for Backdoor.Linux.Mokes.a ... nothing found
Searching for Malicious TinyDNS ... nothing found
Searching for Linux.Xor.DDoS ... nothing found
Searching for Linux.Proxy.1.0 ... nothing found
Searching for suspect PHP files... nothing found
Searching for anomalies in shell history files... nothing found
Checking `asp’... not infected
Checking `bindshell’... not infected
Checking `lkm’... chkproc: nothing detected
chkdirs: nothing detected
Checking `rexedcs’... not found
Checking `sniffer’... enp0s3: PF_PACKET(/sbin/dhclient)
virbr0: not promisc and no PF_PACKET sockets
Checking `w55808’... not infected
Checking `wted’... chkwtmp: nothing deleted
Checking `scalper’... not infected
Checking `slapper’... not infected
Checking `z2’... user analyst deleted or never logged from lastlog!
Checking `chkutmp’... The tty of the following user process(es) were
not found
in /var/run/utmp !
! RUID PID TTY CMD
! analyst 2597 pts/5 bash
! root 3733 pts/5 sudo ./chkrootkit
! root 3734 pts/5 /bin/sh ./chkrootkit
! root 4748 pts/5 ./chkutmp
! root 4749 pts/5 sh -c ps ax -o "tty,pid,ruser,args"
! root 4750 pts/5 ps ax -o tty,pid,ruser,args
chkutmp: nothing deleted
Checking `OSX_RSPLUG'... not tested
analyst@cuckoo:~$
```

Piping Commands (3.3.2.6)

Although command line tools are usually designed to perform a specific, well-defined task, many commands can be combined to perform more complex tasks by a technique known as piping. Named after its defining character, the pipe (|), piping consists of chaining commands together, feeding the output of one command into the input of another.

For example, the **ls** command is used to display all the files and directories of a given directory. The **grep** command compares searches through a file or text looking for the specified string. If found, **grep** displays the entire contents of the folder where the string was found. The two commands, **ls** and **grep,** can be piped together to filter out the output of **ls,** as shown in Example 3-13 with the **ls -l | grep nimda** command.

Example 3-13 Output of the **grep** Command

```
[analyst@secOps ~]$ ls -l lab.support.files
total 584
-rw-r--r-- 1 analyst analyst 649 Jun 28 2017 apache_in_epoch.log
-rw-r--r-- 1 analyst analyst 126 Jun 28 2017
applicationX_in_epoch.log
drwxr-xr-x 4 analyst analyst 4096 Aug 24 12:36 attack_scripts
-rw-r--r-- 1 analyst analyst 102 Jul 20 09:37 confidential.txt
-rw-r--r-- 1 analyst analyst 2871 Dec 15 2016 cyops.mn
-rw-r--r-- 1 analyst analyst 75 May 24 2017 elk_services
-rw-r--r-- 1 analyst analyst 373 Feb 16 2017 h2_dropbear.banner
-rw-r--r-- 1 analyst analyst 147 Mar 21 2017 index.html
drwxr-xr-x 2 analyst analyst 4096 Aug 24 12:36 instructor
-rw-r--r-- 1 analyst analyst 255 May 2 2017 letter_to_grandma.txt
-rw-r--r-- 1 analyst analyst 24464 Feb 7 2017 logstash-tutorial.log
drwxr-xr-x 2 analyst analyst 4096 May 25 2017 malware
-rwxr-xr-x 1 analyst analyst 172 Jul 25 16:27 mininet_services
drwxr-xr-x 2 analyst analyst 4096 Feb 14 2017 openssl_lab
drwxr-xr-x 2 analyst analyst 4096 Aug 24 12:35 pcaps
drwxr-xr-x 7 analyst analyst 4096 Sep 20 2016 pox
-rw-r--r-- 1 analyst analyst 473363 Feb 16 2017 sample.img
-rw-r--r-- 1 analyst analyst 65 Feb 16 2017 sample.img_SHA256.sig
drwxr-xr-x 3 analyst analyst 4096 Aug 24 10:47 scripts
-rw-r--r-- 1 analyst analyst 25553 Feb 13 2017 SQL_Lab.pcap
[analyst@secOps ~]$ ls -l lab.support.files | grep ap
-rw-r--r-- 1 analyst analyst 649 Jun 28 2017 apache_in_epoch.log
-rw-r--r-- 1 analyst analyst 126 Jun 28 2017
applicationX_in_epoch.log
drwxr-xr-x 2 analyst analyst 4096 Aug 24 12:35 pcaps
-rw-r--r-- 1 analyst analyst 25553 Feb 13 2017 SQL_Lab.pcap
[analyst@secOps ~]$
```

Summary (3.4)

In this chapter, you learned how the Linux operation system is used in a SOC environment, including:

- Linux tools that are used for security monitoring and investigation
- How to use the Linux shell to work with directory and files and how to create, modify, copy, and move text files
- The difference between server and client applications

In this chapter, you also learned how to perform basic Linux administration tasks, including:

- How to view service configuration files
- What features need to be hardened on Linux devices
- The types and location of services logs used for monitoring purposes

You also learned about the various Linux file system types, including:

- ext2, ext3, and ext4
- NFS
- CDFS
- Swap file system
- HFS+
- Master boot record

You learned how roles and file permissions dictate which users or groups can access which files and whether those users or groups have Read, Write, or Execute permissions. You also learned how the root user or owner of a file can change permissions. These files can have hard links or symbolic links. A hard link is another file that points to the same location as the original file. A symbolic link, sometimes called a symlink or soft link, is similar to a hard link in that applying changes to the symbolic link will also change the original file.

Finally, in this chapter you learned how to perform basic security-related tasks on a Linux host, including:

- Installing and running applications from the command line
- Keeping the system up to date with **apt-get update** and **apt-get upgrade**
- Viewing the current processes and forks running in memory
- Using **chkrootkit** to check the computer for known rootkits
- Using piping to chain commands together, feeding one command output into the input of another command

As a cybersecurity analyst, you need a basic understanding of the features and characteristics of the Linux operating system and how Linux is used in a SOC environment.

Practice

The following activities provide practice with the topics introduced in this chapter. The Labs are available in the companion CCNA Cybersecurity Operations Lab Manual (ISBN 9781587134388).

Lab 3.1.2.6: Working with Text Files in the CLI
Lab 3.1.2.7: Getting Familiar with the Linux Shell
Lab 3.1.3.4: Linux Servers
Lab 3.2.1.4: Locating Log Files

Lab 3.2.2.4: Navigating the Linux Filesystem and Permission Settings