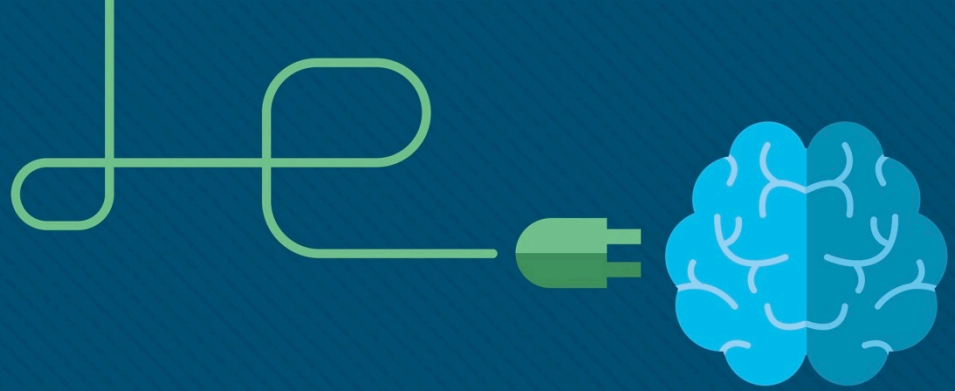


Chapter 2: Linux Operating System

Information Security

Dr. Ayman Aljarbouh

Chapter 2 - Sections & Objectives

- 2.1 Linux Overview
  - Perform basic operations in the Linux shell.
    - Explain why Linux skills are essential for network security monitoring and investigation.
    - Use the Linux shell to manipulate text files.
    - Explain how client-server networks function.
- 2.2 Linux Administration
  - Perform basic Linux administration tasks.
    - Explain how a Linux administrator locates and manipulates security log files..
    - Manage the Linux file system and permissions.
- 2.3 Linux Hosts
  - Perform basic security-related tasks on a Linux host.
    - Explain the basic components of the Linux GUI.
    - Use tools to detect malware on a Linux host.

Chapter 2 - Sections & Objectives

- 2.1 Linux Overview
  - Perform basic operations in the Linux shell.
    - Explain why Linux skills are essential for network security monitoring and investigation.
    - Use the Linux shell to manipulate text files.
    - Explain how client-server networks function.
- 2.2 Linux Administration
  - Perform basic Linux administration tasks.
    - Explain how a Linux administrator locates and manipulates security log files..
    - Manage the Linux file system and permissions.
- 2.3 Linux Hosts
  - Perform basic security-related tasks on a Linux host.
    - Explain the basic components of the Linux GUI.
    - Use tools to detect malware on a Linux host.

Today

2.2 Linux Administration

Module Objectives

Module Title: Linux Administration

Module Objective: Perform basic Linux administration tasks.

Topic Title	Topic Objective
Basic Server Administration	Explain how a Linux administrator locates and manipulates security log files.
The Linux File System	Manage the Linux file system and permissions.

Service Configuration Files

- In Linux, services are managed using configuration files.
- Common options in configuration files 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.
- A server configuration file usually has instructions that begin with a comment like a hash #. Comments are ignored by the software.

```
[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 --enable-ppm --enable-
perfprofiling --enable-zlib --enable-active-response --enable-normalizer --enable-reload --
enable-react --enable-flexresp3
<output omitted>
##################################################
# 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
```

Hardening Devices

- Device hardening involves implementing proven methods of securing the device and protecting its administrative access.
- The following are basic best practices 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 re-using old passwords

Monitoring Service Logs

- Log files are records to keep track of important computer events.
- Linux has the following types of logs:
  - Application Logs
  - Event Logs
  - Service Logs
  - System Logs

Log file analysis allows an administrator to guard against upcoming issues before they occur.

Linux Log File	Description
/var/log/messages	Used to store informational and non-critical system messages
/var/log/auth.log	Stores all authentication-related events
/var/log/secure	Used by RedHat and CentOS and tracks sudo logins, SSH logins, and errors logged by SSSD
/var/log/boot.log	Stores boot related messages during startup
/var/log/dmesg	Contains kernel ring bugger messages
/var/log/kern.log	Contains information logged by the kernel
/var/log/cron	A service used for scheduling automated tasks in Linux
/var/log/mysqld.log or /var/log/mysql.log	Logs all debug, failure and success messages related to the mysql process and mysqld_safe daemon

Monitoring Service Logs (Contd.)

- The command output shows a portion of **/var/log/messages** log file.
- Each line represents a logged event.
- The timestamps at the beginning of the lines mark the moment the event took place.

```
[analyst@secOps ~]$ sudo cat /var/log/messages
Mar 20 15:28:45 secOps kernel: Linux version 4.15.10-1-ARCH (builduser@heftig-18961) (gcc version 7.3.1
20180312 (GCC)) #1 SMP PREEMPT Thu Mar 15 12:24:34 UTC 2018
Mar 20 15:28:45 secOps kernel: Command line: BOOT_IMAGE=/boot/vmlinuz-linux root=UUID=07c6b457-3f39-
4ddf-bfd8-c169e8a877b2 rw quiet
Mar 20 15:28:45 secOps kernel: KERNEL supported cpus:
Mar 20 15:28:45 secOps kernel:   Intel GenuineIntel
Mar 20 15:28:45 secOps kernel:   AMD AuthenticAMD
Mar 20 15:28:45 secOps kernel:   Centaur CentaurHauls
Mar 20 15:28:45 secOps kernel: x86/fpu: Supporting XSAVE feature 0x001: 'x87 floating point registers'
Mar 20 15:28:45 secOps kernel: x86/fpu: Supporting XSAVE feature 0x002: 'SSE registers'
Mar 20 15:28:45 secOps kernel: x86/fpu: Supporting XSAVE feature 0x004: 'AVX registers'
Mar 20 15:28:45 secOps kernel: x86/fpu: xstate_offset[2]:  576, xstate_sizes[2]:  256
Mar 20 15:28:45 secOps kernel: x86/fpu: Enabled xstate features 0x7, context size is 832 bytes, using
'standard' format.
Mar 20 15:28:45 secOps kernel: e820: BIOS-provided physical RAM map:
Mar 20 15:28:45 secOps kernel: BIOS-e820: [mem 0x0000000000000000-0x000000000009fbff] usable
Mar 20 15:28:45 secOps kernel: BIOS-e820: [mem 0x000000000009fc00-0x000000000009ffff] reserved
Mar 20 15:28:45 secOps kernel: BIOS-e820: [mem 0x00000000000f0000-0x00000000000fffff] reserved
Mar 20 15:28:45 secOps kernel: BIOS-e820: [mem 0x0000000000100000-0x00000000003fefffff] usable
Mar 20 15:28:45 secOps kernel: BIOS-e820: [mem 0x00000000003fff0000-0x00000000003fffffff] ACPI data
Mar 20 15:28:45 secOps kernel: BIOS-e820: [mem 0x00000000fec00000-0x00000000fec00fff] reserved
Mar 20 15:28:45 secOps kernel: BIOS-e820: [mem 0x00000000fee00000-0x00000000fee00fff] reserved
Mar 20 15:28:45 secOps kernel: BIOS-e820: [mem 0x00000000fffc0000-0x00000000ffffffff] reserved
Mar 20 15:28:45 secOps kernel: NX (Execute Disable) protection: active
Mar 20 15:28:45 secOps kernel: random: fast init done
Mar 20 15:28:45 secOps kernel: SMBIOS 2.5 present.
Mar 20 15:28:45 secOps kernel: DMI: innotek GmbH VirtualBox/VirtualBox, BIOS VirtualBox 12/01/2006
Mar 20 15:28:45 secOps kernel: Hypervisor detected: KVM
Mar 20 15:28:45 secOps kernel: e820: last_pfn = 0x3fff0 max_arch_pfn = 0x400000000
Mar 20 15:28:45 secOps kernel: MTRR: Disabled
Mar 20 15:28:45 secOps kernel: x86/PAT: MTRRs disabled, skipping PAT initialization too.
Mar 20 15:28:45 secOps kernel: CPU MTRRs all blank - virtualized system.
```

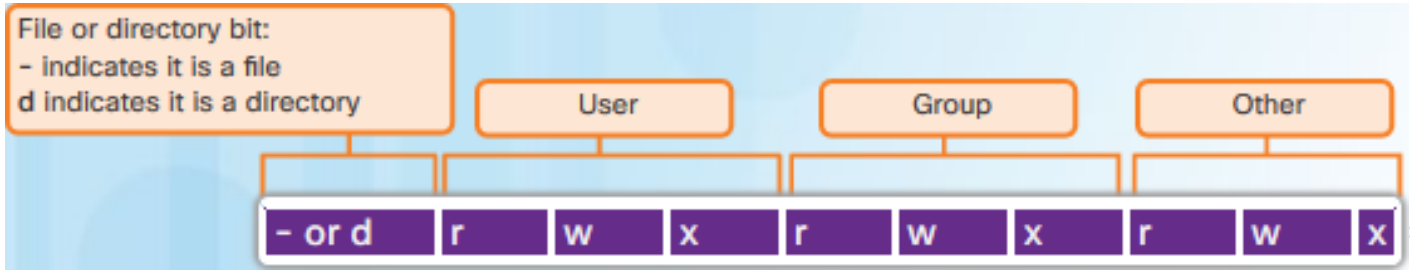
The File System Types in Linux

File System Type	Description
ext2 (second extended file system)	Is the file system of choice for flash-based storage media.
ext3 (third extended file system)	Is an improved successor to ext2 with the additional feature of journaling of all the file system changes.
ext4 (fourth extended file system)	Is designed as a successor to ex3 with increased support file sizes and better performance than ext3.
NFS (Network File System)	Is a network-based file system, allowing file access over the network.
CDFS (Compact Disc File System)	Was created specifically for optical disk media.
Swap File System	Is used when the system runs out of RAM.
HFS+ (Hierarchical File System Plus)	Is the primary file system used by Apple in its Macintosh computers.
APFS (Apple File System)	An updated file system used by Apple devices that provides strong encryption and is optimized for flash and solid state drives.
Master Boot Record (MBR)	Is located in the first sector of a partitioned computer and stores all the information about the way the file system is organized.

Linux Roles and File Permissions

Every file in Linux carries its file permissions, which define the actions that the owner, the group, and others can perform with the file. The possible permission rights are Read, Write, and Execute

- **User** - the file owner's permission
- **Group** - a group's permission to a file
- **Other** – any user, non-owner's permission to a file
- **Read** – the ability to look at a file's contents
- **Write** – the ability to change a files contents
- **Execute** – the ability to run or launch a file (scripts and programs)



Linux Roles and File Permissions (Contd.)

The output of the **ls -l** command provides a lot of information about the file **space.txt**:

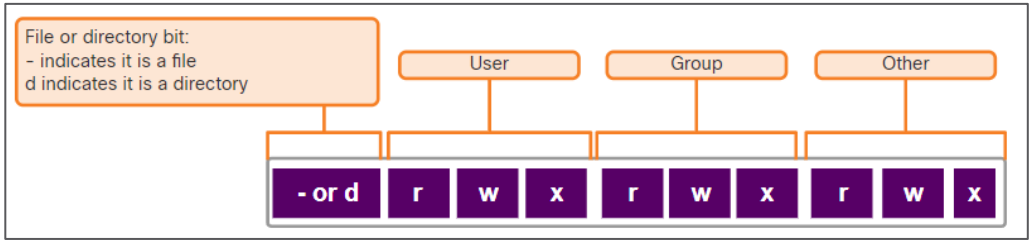
- The first field displays the permissions with **space.txt** (**-rwxrw-r--**).
- The second field defines the number of hard links to the file (number **1** after the permissions).
- The third and fourth field display the user (**analyst**) and group (**staff**) who own the file, respectively.
- The 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 file name.

```
[analyst@secOps ~]$ ls -l space.txt
-rwxrw-r-- 1 analyst staff 253 May 20 12:49 space.txt
 (1)(2)(3)(4)(5)(6)(7)
[analyst@secOps ~]$
```

Linux Roles and File Permissions (Contd.)

The figure here shows a breakdown of file permissions in Linux. The file **space.txt** has the following permissions:

- The dash (-) means that this is a file.
- The first set of characters (**rwx**) is for user permission. The user (**analyst**) who owns the file can **R**ead, **W**rite and e**X**ecute the file.
- The second set of characters is for group permissions (**rw-**). The group (**staff**) who owns the file can **R**ead and **W**rite to the file.
- The third set of characters is for any other user or group permissions (**r--**) who can only **R**ead the file.




```
[analyst@secOps ~]$ ls -l space.txt
-rwxrw-r-- 1 analyst staff 253 May 20 12:49 space.txt
 (1)(2)(3)(4)(5)(6)(7)
[analyst@secOps ~]$
```

Linux Roles and File Permissions (Contd.)

- Octal values are used to define permissions.
- File permissions are a fundamental part of Linux and cannot be broken.
- The only user that can override file permission on a Linux computer is the root user.

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

Hard Links and Symbolic Links

- A hard link is another file that points to the same location as the original file.
- Use the command **ln** to create a hard link.
- The first argument is the existing file and the second argument is the new file.
- As shown in the command output, the file **space.txt** is linked to **space.hard.txt** and the link field now shows 2.
- 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**.

```
[analyst@secOps ~]$ ln space.txt space.hard.txt
[analyst@secOps ~]$
[analyst@secOps ~]$ ls -l space*
-rw-r--r-- 2 analyst analyst 239 May  7 18:18 space.hard.txt
-rw-r--r-- 2 analyst analyst 239 May  7 18:18 space.txt
[analyst@secOps ~]$
[analyst@secOps ~]$ echo "Testing hard link" >> space.txt
[analyst@secOps ~]$
[analyst@secOps ~]$ ls -l space*
-rw-r--r-- 2 analyst analyst 257 May  7 18:19 space.hard.txt
-rw-r--r-- 2 analyst analyst 257 May  7 18:19 space.txt
[analyst@secOps ~]$
[analyst@secOps ~]$ rm space.hard.txt
[analyst@secOps ~]$
[analyst@secOps ~]$ more space.txt
Space... The final frontier…
These are the voyages of the Starship Enterprise. Its continuing mission:
- To explore strange new worlds…
- To seek out new life; new civilizations…
- To boldly go where no one has gone before!
Testing hard link
[analyst@secOps ~]$
```

Hard Links and Symbolic Links (Contd.)

- 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 the command output, 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 **text.txt** file means that **mytext.txt** is now linked to a file that no longer exists.

```
[analyst@secOps ~]$ echo "Hello World!" > test.txt
[analyst@secOps ~]$
[analyst@secOps ~]$ ln -s test.txt mytest.txt
[analyst@secOps ~]$
[analyst@secOps ~]$ echo "It's a lovely day!" >> mytest.txt
[analyst@secOps ~]$
[analyst@secOps ~]$ more test.txt
Hello World!
It's a lovely day!
[analyst@secOps ~]$
[analyst@secOps ~]$ rm test.txt
[analyst@secOps ~]$
[analyst@secOps ~]$ more mytest.txt
more: stat of mytest.txt failed: No such file or directory
[analyst@secOps ~]$
[analyst@secOps ~]$ ls -l mytest.txt
lrwxrwxrwx 1 analyst analyst 8 May  7 20:17 mytest.txt -> test.txt
[analyst@secOps ~]$
```

Hard Links and Symbolic Links (Contd.)

The following table shows several benefits of symbolic links over hard links:

Hard Links	Soft Links
Locating hard links is difficult.	Symbolic links show the location of the original file in the **ls -l** command.
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 as the system itself uses hard links to define the hierarchy of the directory structure.	Symbolic links can link to directories.

New Terms and Commands

• Symlink • Hard link • ext2, ext3, ext4 • NFS (Network File System) • CDFS (Compact Disc File System) • Swap File System • HFS+ (Hierarchical File System Plus)	• APFS (Apple File System) • Master Boot Record (MBR) • Application Logs • Event Logs • Service Logs • System Logs

Lab 6 - Navigating the Linux Filesystem and Permission Settings

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