Labs

Lab resources are available at: https://github.com/janustechacademy/EL7

Lab 1: Initial Configuration

1. Log into your lab machine.
2. Using redirection (**>**, **>>**, **|**) create a file named **IP-info** which contains the output of the following commands: **ip a**, **ip route**, **ip neigh**.
3. From the contents of **IP-info** create a new file named **192-info**. It should only have lines containing the string **192**.
4. Examine the man pages for: **cp**, **mv**, **ln**, and **history**.
5. Create a file called **TZ-man** which contains a list of all the man pages relating to timezone. Did it work? If not, fix it.
6. Clear your history list by deleting all entries.
7. Copy **IP-info** to **IP-old**; rename **192-info** to **192-old**. Do this on one line.
8. Insert your history at the bottom of **192-old**.
9. Make a directory (**mkdir**) called **192-zzz**
    a. **cat 192-*** – this will return the contents of **192-old**, and an error message because cat can't read a directory.
        i. Create a file **err-192** that contains only the error from this command.
    b. Run the command again, displaying only the error.

Lab 2: Files and Directories

Finding Things

For the following exercises, unless otherwise specified, you may use any commands you choose to determine the answers. These might be helpful: **locate**, **whereis**, **which**, and **find**.

1. Find the location of the **cp** command.
2. Find the location of the man pages for the **mv** command.
3. Find all instances of a file called **findme.script**.
    a. How many did you find?
    b. How many instances of this script are in root's **$PATH**?
    c. Which instance of **findme.script** runs when root calls **findme.script** without a path?
4. Create a file in **/lab/02-Files/** called **mxyzptlk**. Use **locate** to locate it. Did it work? If not, fix it.

Finding Things with find

In this section, use the **find** command to answer the questions. Putting **2>/dev/null** after a query will suppress errors from the **/proc** file system.

5. The **/etc** directory contains directories and files that begin with **rc** and end with **.d**. Find only the directories.
6. Find any unowned files on your system.
7. Find any files with the SUID bit set which ARE ALSO writable by others.
8. Find any files which haven't been accessed in more than 20 years (they do exist!).
9. Use a single line command to locate and **optionally** remove the **/lab/02-Files/killme** directory and all its contents. Do be careful.
10. Find any file in **/boot** greater than 20 megabytes in size.

Manipulating Files and Directories

In this section, we will create, manipulate, and destroy files and directories. You may use any commands you like to achieve these goals.

11. Create a directory in **/lab/02-Files/** called **new**.
12. Change directories to **new**.
13. Create a file named **file1**. Put the phrase "this is file 1" inside of it.
14. Copy **file1** and call the copy **file3**.
15. Rename **file3** to **file2**.
16. Create an empty file called **file3**
17. Create a directory in **/lab/02-Files/new** called **subfiles**
18. Use a single command to copy all files in the **/lab/02-Files/new** directory into **subfiles**.
19. Move **file1** up one level using dot notation.
20. Without using **vi**, read **file1** and append its contents to **file3**.

Lab 3: vi

vi /lab/03-vi/edit-me, follow the directions in the file.

When complete you can check your work:

```
# diff edit-me edit-me.finished | grep "<"
```

Lab 4: Users and Groups

1. Configure your computer such that users are created with:
    a. default password expiration of one month from now
    b. immediate inactivation on password expiration
    c. password aging fields and logon failure delay meets standards
    d. Do not set minimum password length or quality requirements at this point.
2. Enable **`wheel`** in **`sudoers`**.
3. Configure **`/etc/pam.d/su`** to restrict use of **`su -`** to members of **`wheel`**.
4. Create the following users as specified.
   Unless otherwise stated, all should have a primary group of their own.
   All users should have a password set.
   Usernames should be **all lower case**.
    a. Adam and Brenda – regular users
    b. Don and Emma – can use **`sudo`**, members of group **`dev`**.
    c. Frank – cannot login interactively, with a comment noting that fact.
    d. Grace – **`UID = 3001`**, member of **`dev`**, **`helpdesk`**, and **`wheel`**
    e. Harry – **`UID = 3002`**, member of **`recruiting`**,
       home directory **`/lab/05-Permissions/recruiting`**
    f. Jane and Mary – members of **`recruiting`** and **`restricted-users`**
    g. Nick – member of **`helpdesk`**.
5. Try to log in as Frank.
6. Change Brenda's shell to **`vi`**. Change to her environment.
    a. Attempt to read the date into a file – what caused the failure?
7. Check your work by viewing the relevant files in **`/etc/`**.

Lab 5: Ownership, Permissions, and Access

Preparation:

Ensure that **grace** is a member of **helpdesk**, but NOT **recruiting**.
Ensure that **harry** is a member of **recruiting**, but NOT **helpdesk**.

Setting Ownership and Basic Permissions

1. As root, create the following directories: **/lab/05-Permissions/recruiting**, **/lab/05-Permissions/helpdesk**
2. Assign ownership of **recruiting** to user **harry** and group **recruiting**.
3. Assign ownership of **helpdesk** to user **grace** and group **helpdesk**.
4. As **harry**, set the permissions of **recruiting** to **770**.
5. As **harry**, set the access mode of **recruiting** so that new files in that directory will automatically be owned by the **recruiting** group.
6. As **grace**, set the permissions of **helpdesk** to **rwxrwx---**.
7. As **grace**, set the access mode of **helpdesk** so that new files in that directory will automatically be owned to the **helpdesk** group.
8. As **grace**, attempt to access the **recruiting** directory. This should fail.
9. As **harry**, attempt to access the **helpdesk** directory. This should fail.

Extended ACLs – The order in which you do these tasks matters, as some ACL commands may overwrite previous entries. Review the exercise first, and plan your ACLs before applying them.

10. Create a file in **recruiting** called **rfile**. It should contain the line, "this is rfile."
11. Create a file in **helpdesk** called **hfile**. It should contain the line, "this is hfile."
12. Set an extended ACL on recruiting that will allow members of the **helpdesk** group to list the contents of the directory and read the files inside of it.
13. Grant full control of any files created in this directory to **grace**.
14. Others should have no access.
15. Set an extended ACL on helpdesk that will allow members of the **recruiting** group to list the contents of the directory and read the files inside of it.
16. Grant full control of any files created in this directory to **harry**.
17. Others should have no access.
18. Ensure that these ACLs are applied recursively to the existing files in the directories.

Testing SUID

19. Verify that all executables in **/lab/05-Permissions** are **SUID/SGID root:root**
20. **# su - adam**
21. **# cd /lab/05-Permissions**.

22. Run **`who`**, **`id`**, and **`whoami`**. Note the results.
23. Run those three commands once again, this time with a **`./`** preceding each.
24. Add **`.`** to the beginning of Adam's path. Run the commands one last time, this time without the leading **`./`**

Lab 6: Regular Expressions

You will find the files for this exercise in **/lab/06-regex**.

1. Copy **something.com.zone** to **else.com.zone**.
2. Copy **something.named.conf** to **else.named.conf**.
3. Take a moment to read over these two files and familiarize yourself with their contents.
4. Display any lines in **else.com.zone** which contain IP addresses.
5. Place the result in **else.ip**.
6. Display any lines in **else.com.zone** which contain hostnames.
7. Place the result in **else.host**.
8. Repeat these tasks for the **else.named.conf** file.
9. Append the results to **else.ip** and **else.host**.
10. Your network has been renumbered from 192.168.10.0/24 to 10.10.10.0/24.
    Use **sed** to make the appropriate changes IP addresses in the **else.com.zone** and **else.named.conf** files.
11. You have a new domain. Your old domain **something.com** is being replaced with **else.com**.
    Use **awk** to make the appropriate changes in both files.
    a. This includes both hostnames and filename references.
    b. DO NOT change any references to domains other than something.com.
    c. Do not alter any portion of a hostname OTHER THAN the **something.com** domain.

Take a moment to review your work. Compare the new files to the originals. Did you miss anything? Did you accidentally change anything you shouldn't have? If you did, resist the urge to hand correct this with **vi**. Try to use **sed** or **awk** to fix mistakes.

If you finished early, and need something to do, try this:

12. Copy **/etc/passwd** to **/lab/06-regex/passfile**
13. Use **sed** to remove all lines in **passfile** that start with **a** and save the file in place.
14. Use **diff** to compare **passfile** and **/etc/password**.
15. Display **passfile**, sorted alphabetically, saving a copy to **alphapass** in one line.
16. Display **alphapass** to your screen, but replace all the colons with linebreaks.
17. Repeat the last step, but eliminate duplicate lines.
18. Repeat the last step, but only show lines that start with a slash.
    a.

Lab 7: Booting

1. Make a copy of **`/etc/default/grub`**
    a. Alter the copy to:
    b. Have a timeout of 15
    c. Not hide the menu
    d. Have a new boot entry called Other which
        i. Displays all boot messages
        ii. Has SELinux disabled
2. When done have an instructor verify your changes.
3. Commit your changes to **`grub.cfg`**

Lab 8: Processes and Services

Preparation:

Open at least three connections to your server. You may even want more.

Run **`top`** in one session…you will want to keep **`top`** open throughout this lab.

All scripts are in **`/lab/08-Processes`**; alter permissions as needed.

Read through the instructions before executing any given step.

Verify who you are before running **`memmy`**. If run as **`root`**, it will break your box.

1. Pause top; open **`vi`**, and **`man top`** – in the same window.
2. Toggle between them using **`jobs`** and **`fg`**.

3. Run **`chew`**, this will execute **`bc`** (a calculator) – using quite a bit of cpu. Note its PID.
4. Using the signal passing function of **`top`**, pause and resume the **`bc`** process.
5. Background the **`bc`** process from the command line that originated it.
6. **`disown`** it and log out. Did it continue running?

7. Start **`spread`**, this will run the **`bc`** backgrounded at varying nicenesses.
8. Create a file with the PIDs of the **`bc`** process in it
9. Pause all of them except the nicest.
    a. If you get really stuck on this there are hints in **`/lab/08-Process/killer`**.
10. Resume the 5 least nice.
11. Clean up all the **`bc`** processes before proceeding.

12. Start a **watch** for processes named sleep. In a separate window watch for processes belonging to Jane.
13. Run **nappy** as root, Jane, and Mary. This starts multiple **sleeps**.
14. Run **spread** as Jane.
15. What is different when Jane runs **spread**?
16. Kill all of Jane's **sleep** processes, without affecting any others.
17. Kill all of Mary's processes.
18. Clean up the **bc** and **sleep** processes before proceeding.

19. Using **limits.conf**, give Mary a hard limit of 1 minute cpu. Set hard limits on Nick for nproc = 5000, nice and priority =15.
20. As Mary, run **chew**. Observe what occurs when processor time hits 60 seconds.
21. As Nick, run **spread**. Notice the priorities.
22. As Nick run **forkbomb**.
    a. Make sure you don't do this as root!
23. Clean up all of Nick and Mary's processes.

24. Start a **vmstat -a -S M 5**. This will display memory information in MiB.
25. Run **swapoff -a**, watch the swap file drain. Confirm that swap was disabled using **# swapon --summary**.
26. In a new window prepare to issue **pkill grep**, do not press enter yet.
27. As Mary, run **memmy**.
28. Observe the change in memory usage.
29. As soon as Out of memory errors begin **pkill grep** .
30. Near the end of **/var/log/messages**, search for **oom-killer**.
    a. Read down from there.
31. Re-enable swap.
32. Run **memmy** again as Mary. Observe the differences.
33. Create a .slice for Grace, limiting memory usage to .5 G and CPU to 50%.
34. Run **memmy** and **chewy** as Grace. Observe the differences.
35. To ensure that no rogue processes are left over, **# reboot**.

Lab 9: File Systems

Throughout this lab, you will be making extensive changes to disk structures. Please remember to check your work and ensure that the kernel is aware of your changes. If it is not, provoke rescans of your SCSI bus using the methods supplied in the course manual.

Partitioning with **`fdisk`** or **`parted`**

1. You should have an empty hard drive attached to your system. Identify it.
2. On your free disk, create 3 partitions of 1 GB each.
3. Ensure that the partition table is updated to reflect your work.
4. File System creation and mounting
5. Put an **`xfs`** filesystem on your first empty partition.
6. Create a mountpoint called **`/mount1`**.
7. Mount the filesystem to **`/mount1`**.
8. Check it.
9. Unmount the filesystem
10. Configure **`/etc/fstab`** to automatically mount your new filesystem at boot or by mountpoint. Use default options.
11. Mount your new filesystem with: **`# mount /mount1`** .
12. Check your work.

LVM

13. Create a physical volume from your second partition.
14. Create a volume group containing only that physical volume.
15. Create a logical volume that uses 100 percent of the free space in that volume group.
16. Create an xfs filesystem on that logical volume.
17. Create a mount point **`/mount2`** and mount the filesystem to it.
18. Unmount it.
19. Create a **`.mount`** unit for this filesystem.
20. Mount it. Unmount it.
21. Create an **`.automount`** unit for this filesystem and enable it.
22. Enter the **`/mount2`** directory and check the status of your mount unit.
23. Pretend the filesystem is full. Use the 3rd free partition to extend the volume group, logical volume, and filesystem.

Hard Links

1. Create a file in **`/lab/07-Filesystems/`** named **`original`**, with content: "This is the original file."
2. Create a hard link to the file in the same directory. Call the link **`copy`**.
3. Get a list of inodes for all files in **`/lab/07-Filesystems`**. What are the inode numbers for original and copy? Record the inode of original.

4. Read the contents of **copy**. What does it say?
5. Add a line to copy that says, "edited from copy."
6. Read the contents of **original** to the screen. What does it say?
7. What is the link count for **original**? What is it for **copy**?
8. Delete **original**.
9. What is the link count for **copy**?

Symbolic Links

10. Create a symbolic link to **copy** in the same directory. Call the link **original**.
11. What are the inodes of the files? What does a long listing tell you about **original**?
12. Move **original** to **/mount1/original**.
13. Add a line to **original** that reads, "edited from symbolic original".
14. Read **copy**. What does it say?
15. What is the link count for **copy**?
16. Delete **copy**. Attempt to read **/mount1/original**. What happened?
17. Create a file in **07-Filesystems** called **copy**.
18. Add a line to it that reads, "this is not the same file."
19. Attempt to read **/mount1/original**. What happened?
20. Delete **/mount1/original**.
21. Read **copy**. Did deleting **/mount1/original** have an effect on **/lab/copy**?
22. Make a hard link from **/lab/07-Filesystems/copy** to **/lab/07-Filesystems/original**.
23. Delete **/lab/07-Filesystems/copy**. What are the contents of **original**? What is the inode and link count? Does the inode for **original** match the one recorded earlier in this lab?

Lab 10 : Scheduling Events

All tasks for this lab are in **/lab/09-Events**

1. Schedule the following using **at**:
    a. **at.task1** five minutes from now
    b. **at.task2** a few minutes after midnight tonight
    c. **at.task3** an hour from now and Friday afternoon
    d. **at.task4** next Tuesday at 11 am
2. View the jobs.
3. Remove the job which would run **at.task3** Friday afternoon.
4. Schedule the following with **cron**:
    a. **cron.job1** weekly beginning ten minutes from now
    b. **cron.job2** at 2130 on odd numbered dates during the week
5. Schedule these with systemd timers
    a. **cron.job3** at 15 minutes after boot, and every twelve hours after that
    b. **cron.job4** at 1130 on weekdays
6. Use **anacron** and a symbolic link to run **cron.job5** weekly
7. Lower the **batch** threshold to 0.01
8. Run **chewy** from the previous lab

9. **`batch`** a job that will wall "Done with this lab!"
10. End the **`bc`** processes started by **`chewy`**

Lab 11: Networking

Configure and test static networking

1. Confirm that the NetworkManager service is uninstalled.
2. Configure your hostnames and IP addresses to match your student sheet.
3. Ping the another student's machines by FQDN and by single name.

Use network tools

4. Find the name server (NS) and mail exchanger (MX) names and addresses for redhat.com, and doe.gov.
5. Using **`ss`**, start a **`watch`** on all tcp ports.
    a. ssh to localhost, then ping yourself. Notice the difference in results.
6. Run **`ss`** by itself (without `watch`), modifying the results to exclude listening ports and connections from **`::1`**, and include process information.

Configure the firewall

7. Create an at job to disable the firewall 20 minutes from now. If you haven't locked yourself out, remove and renew the job every now and then. If you do lock yourself out, it will let you back in when it runs.
   Note: the most common method of locking yourself out is by creating a default DROP with no other rules configured.
8. Configure your default zone to have a policy of DROP.
9. Allow **`https`** on TCP port 443 inbound from anywhere, for NEW, ESTABLISHED, and RELATED connections.
10. Block **`http`** on TCP port 80 inbound from anywhere.
11. Restrict **`ssh`** access to an appropriate network.
12. Test these rules using **`nmap`** from your other machine.

Lab 12: Remote Access

1. Secure **`sshd`**, provide a login banner.
2. Work with one of your neighbors to **`ssh`** without password as Grace and as root from one machine to the other.
3. Use **`scp`** to copy the a file you generated earlier to /labs/16-Remote on your neighbors machine.
4. Restrict Adam to **`sftp`** only.
5. Create a chrooted sftp home directory for him.
6. Move a copy of your local **`/etc/hosts`** to Adam's home directory on your neighbor's machine.
7. Create a file showing all nonroot logins which have occurred on your computer called **`nonroot.txt`**. Make it available via **`http`**.
8. Get a copy of your classmate's **`nonroot.txt`**, without using a graphical browser.

Lab 13: SELinux

We'll be working with Apache and SELinux in this lab. The httpd daemon should already be installed, and the configuration files on your machines should already be altered to allow the labs to function as written. You should only need to make changes of an SELinux nature to accomplish the stated goals.

1. Ensure **`httpd`** is not running, and that the firewall is off.
2. Change to the **`/etc/httpd/conf`** directory and copy **`httpd.back`** to **`httpd.conf`**.
3. Create a directory, **`/web`** with permissions of **`755`**
4. In **`/web`** create **`index.html`**, with permissions of **`744`** and content "Successfully viewed the file in /web."
5. Start **`httpd`**.
6. Attempt to access this address: **`http://`***`server-name`***`/`**
7. What happened? Did you get the results you were expecting?
8. Place SELinux into Permissive Mode. Try again. Did the issue resolve?
9. Examine log files related to this issue.
10. Correct the issue
11. Turn your firewall back on, shut down **`httpd`**, and check that SELinux is enforcing.

There are no labs for Modules 14 & 15.

Lab 16: Kernel Modules and Parameters

1. In your terminal, display a list of currently loaded kernel modules.
2. Locate **`bnx2fc`**.
3. Display more detailed information about **`bnx2fc`**. What does the description tell you about it? Does it have any associated parameters?
4. What is the default value of **`bnx2fc`**'s debug_logging parameter? Is that the currently loaded value? Verify this.
5. Attempt to alter the **`debug_logging`** parameter of **`bnx2fc`** to **`0x01`** without unloading the module first. Did this succeed? Why or why not?
6. Are there any other modules that depend on **`bnx2fc`**?

Loading and Unloading Kernel Modules

7. Unload **`bnx2fc`**. Did any other modules unload with it? Why or why not?
8. Attempt to load **`bnx2fc`** with the **`debug_logging`** set to **`0x01`**. Did this succeed?
9. Unload the module again.
10. Configure **`bnx2fc`** to load with a **`debug_logging`** value of **`0x02`** when next loaded.
11. Load it. Did the changes take? Why or why not?

Blacklisting Modules

1. Unload **`bnx2fc`**.
2. Blacklist the module. Ensure that this module will not be loaded either manually or automatically.
3. Test it.

Lab 17: Backups

1. Create compressed archives of **`/lab`** called **`lab.tar`** and **`/etc`** called **`etc.tar`**, preserving all attributes
2. Restore the **`lab.tar`** to a new directory **`/new-labs/`**
3. Create an archive of all files changed in **`/etc`** and **`/lab`** in the last two days. Place it in **`/lab/18-disaster/`**.
4. Delete **`/lab/03-vi/editme.finished`**
5. Add a file **`/lab/03-vi/editme.added`**
6. Compare the **`lab.tar`** with the current filesystem using **`tar`**.
7. **`rsync`** all **`.conf`** files from **`/etc/`** to **`/new-labs/saved-confs`** on your neighbor's machine.
8. Delete **`/etc/sane.d/`** on your machine.
9. Add a comment to **`/etc/asound.conf`**
10. Restore **`sane.d`** without overwriting your changes to **`asound.conf`**

Lab 18: Security

1. Install and configure **`aide`**.
2. Write a script to be run daily which archives the results of aide on a neighbor's machine.
3. Audit your installed software and services. Write a series of removal recommendations in **`/lab/19-Security/software-cleanup`**.
    a. Don't forget to check dependencies, don't remove things that would break functionality.
    b. Outline further steps you would take to secure this machine.
    c. Be prepared to justify your results

Lab 20 – Final

You have just inherited a new computer. It has been severely misconfigured, and may have been compromised.

Your new box is currently only reachable by telnet, and is drawing a DHCP address. Your instructor will provide you its current address, as well as the static addresses and hostname that it should be given.

Current logins are **`student`** and **`root`**, both with password **`ujm<KI*(98`**

The **`student`** account can use **`sudo`**, **`root`** is currently locked out.

You will have the use of one of the computers you used during the class, access to the Internet, the course materials, and any notes you have made during the course.

You must secure the computer. You will be given a subset of the STIG as the standard for security.

Some features and functionalities must also be added.

All security and functionality should work without intervention after a reboot.

It is recommended that you review all tasks and the Security Standards before beginning.

1. Configure your IP addresses, DNS, and hostname.
2. Establish secure communication with your box, and remove telnet.
    a. Secure ssh appropriately.
    b. Restrict ssh to your local network.
    c. Create a login banner saying:
    "I've read & consent to terms in IS user agreement."
3. Enable the web management interface.

4. The boot sequence should be protected from unauthorized meddling.
    a. set the boot password to **`BootMe`**
    b. The kernel command line should ensure that fips is on, auditing begins appropriately, and SELinux is enforcing.

5. The firewall and SELinux must run automatically.
    a. The firewall should silently drop packets without responding.
    b. SELinux must be enforcing.

6. Create two partitions of approximately 400 MB each on the second drive.
    a. One of these partitions should be used as the base directory for web services.
    b. This file system should
        i. be mounted at **`/new-web`**
        ii. be labeled **`web`**
        iii. be mounted by label rather than device name.
    c. Create an index.html with the output of hostnamectl and your ip address information.
        i. index.html should only be viewable from your subnet.

7. Create these users:
    a. Kate – member of group **`staff`** and **`developers`**
    b. Mike – member of group **`developers`**, account expires at the end of the year
    c. Nick – member of **`developers`** and **`helpdesk`**, UID = 4001
    d. Paul – UID = 4002, shell = **`vim`**
8. Dave has been terminated. His account should be dealt with.

9. In the second new partition create a shared directory **`/code`** for the developers.
    a. Allow all members of developers, except Nick, read/write access to code.
    b. Nick should have read access only.
    c. Users should not be able to delete files owned by another user.
    d. Any new files or directories created in code should
        i. be owned by the creator and by the developer group
    e. have appropriate permissions

10. Allow only Kate and Mike the ability to use sudo.
11. Configure bidirectional key-based login between your class computer to your new box for Kate.

12. Create a new 200 MB swap file on the second hard drive using a logical volume.
    Add it permanently to the current swap space.

13. Configure **`aide`** to run every day at 11 PM.
    Limit it to no more than one-half of a core of CPU and 250 Megabytes of RAM.

14. Configure **`yum`** to remove all metadata and caches, and create a new cache weekly.
    a. If the computer is powered off, the task must run when it is next powered on.

15. The system must be no more than 30 days out of date.
    a. If a new kernel is available, it should be installed as the default
    b. Two old kernels must remain available and bootable

16. Set the timezone and configure **`ntp`** to use an NIST server or another server as directed.

17. Configure your new machine to send log files to your class machine.

18. Configure auditing to STIG standards.
19. Disable ping to your machine.

20. Configure the kernel parameters appropriate to secure networking.

21. Verify that only root has user id **`0`**

22. Disable CTL-ALT-DEL

23. Configure the machine to default to multi-user, not graphical mode.

24. Find and remove any unowned files.

25. Remove the **`abrt`** package.

26. Create a daily task which will archive all local configuration files to **`/backup`** on your class machine. It should run as a non-root user, with no interaction.