Linux Foundations ivan 2022

1. CLI

Command Syntax

- 1. start with command
  2. provide additional options
  3. profile a target (argument) / targets for the command (if applicable)
  4. repeate 2 and 3 as applicable

The File System

- Everything is file
  Has directories, subdirectories, folders
  Organized as a tree, begins with root

Navigation

- Traversing the tree
  We're always at one particular level of depth
  Mode up, down, check where we are, check what files are at our current level

Commands

- pwd - where am i?
- ls - list segments
  ls - a , will show all files at the current level, including hidden (files whose names begin with a '.')
- cd - change directory
  . - current directory
  .. - parent directory

Creation / Modification / Deletion

touch - create file

- this command expects an argument filename (file.txt)
  creates file in present directory with that filename

mkdir - create new directorry

- expects an argument directory name (more_fish)
  will create directory into present working directory

mv - move file

- expects 2 arguments
  first = file you wish to move
  second = destination to place it
  If you provide the second argument as a file name instead of a directory you can also use it to rename the file

- cp - copy file
  - expects a target source file and target file name

- rm - remove file
  - expects an argument representing the file to delete
    rmdir - to remove folders, folder must be empty first
    -r option - recursive (repeat), use this with rm to say delete all of the files in this particular directory as well as the directory itself

- cat - view contents of a file
  - provide cat with just one argument
    can cat multiple files at once

- Manual
  - If you cant remember commands, use:
    man - manual, provide command you are looking for
    -f - search description of commands
    man -f "list directory"

2. Searching Files

- Finding files
  - find - use to search for files in the tree
    - expects a place to begin its search, can use . to say "start here"
      can provide the attribute in which we want to search as an option, e.g. -name to search on the name of a file
      we provide an argument to say what we are looking to match
      find . -name xxxxx

- Using wildcards
  - find all of the files with ____ in the name
    used to specify generalities in argument
    - * = this can be anything
    - ? = must exist but this one character can be anything
    - [] = character I am specifying must be in the following range ex [a-z][0-9]
      e.g. find . -name *fish.txt

- Searching files
  - search through contents of a particular file
  - grep - use grep to do this ^
    - requires search string as the first argument and the file(s) to search as the remaining
      e.g. grep 'yellow' lfish.txt

3. Linux Filters

- Filter Commands
  - head - displays the first 10 lines (by default) of the contents of a file
    - syntax: head <options> <filename>
      -n specifies how many lines from the top are to be displayed
  - tail - displays the last 10 lines (byb default) of the contents of a file
    - syntax: tail <options> <filename>
      -n: Specifies how many lines from the bottom are to be displayed
  - wc - shows number of lines, word count, byte/character count in the specified file
    - syntax: wc <options> <filename>
      -l - Specifies the number of lines in the given file
      -w - Specifies the number of words in the given file
      -c - Specifies the number of bytes in the given file
  - sort - shows the output of the given file in sorted order
    - syntax: sort <options> <filename>
      -r - Sorts and displays the given file in reverse order
  - uniq - shows the output of t he given file with any duplicated lines omitted
    - syntax: uniq <options> <filename>
      -c - Displays the number of times each line was repeated
- stdin / stdout / stderr
  - Every command (process) launched in linux is given 3 input streams, stdin, stdout, and stderr. Additionally, each stream is given a number as it's file descriptor.
    - stdin
      - The standard input stream
        It accepts text as input
        Given 0 as a file descriptor

- stdout
  - The standard output stream
    Delivers text output to the terminal (shell)
    Given 1 as a file descriptor
- stderr
  - Error messages from the given command are given to stderr
    Given 2 as a file descriptor
    Common to route errors to /dev/null to filter out error messages from a given command:
    2> /dev/null
- *Note
  - Remember, everything in Linux is a file! So you can see each of these file descriptors for each process in the /proc/<PID>/fd (where <PID> is the specific process ID for the running command). (Note that they will disappear when the process is terminated)

- Redirection
  - >
    - Redirects stdout for the command run to the filename given to it
      Often called write / overwrite
      If there is no file currently in existence, then > creates a new file
      If the file exists, and has content in it, > overwrites the old contents of the file with the newly specified content
    - e.g. echo "Hello" > hi.txt --> creates a file called hi.txt that contains the string "Hello"

      echo "Hi There" > hi.txt --> overwrites the content of hi.txt with the string "Hi There"
  - >>
    - Redirects stdout for the command run to the filename given to it
      Often called append
      If there is no file currently in existence, then >> creates a new file
    - e.g. echo "Hello" > hi.txt --> creates a file called hi.txt that contains the string "Hello"

      echo "Hi There" >> hi.txt --> appends the string "Hi There" to the end of hi.txt
  - | (pipe)

- Redirects stdout for the command run on the left side of | to the stdin for the command on the right side of |
- e.g. grep "password" /usr/share/wordlists/unix_passwords.txt | wc -l
  - This will search unix_passwords.txt to find all the passwords that have the string "password" in them
  - Then it will run the wc command and show how many lines there are in that filtered output.
  - Since there is only 1 password per line, we will know how many passwords in unix_passwords.txt have the string "password" in them!

4. Analysis Tools

Tools

- 1. file - Displays the type of the given file
  syntax file <options> <filename>

  2. strings - Displays the human readable strings from a file
  syntax strings <options> <filename>
  -n Specifies the minimum length of the strings to display

  3. tar - Creates/extracts compressed archive files.
  syntax tar <options> <archive_filename> <file_to_be_archived>
  -c Create an Archive
  -x Extract an Archive
  -v Displays verbose information
  -f use the file specified to compress/decompress

  4. gzip/gunzip - Used to compress/decompress a file.
  syntax gzip(gunzip) <options> <filename>
  -d decompresses the file
  note: Adds a .gz extension when compressing, and specifically looks for a .gz extension when decompressing

  5. zip - Used to compress a file in .zip format
  syntax zip <options> <filename>

  6. unzip Used to decompress a .zip file
  syntax unzip <filename>

  7. unrar Used to extract files from rar archives
  Syntax unrar <options> <filename>
  e extract a file (note: do not use the - character with this option)

Encoding

- The process of converting data from one form into another form. Some popular encoding types are ASCII (has 128 characters) and Unicode (has 143,859 characters)
  - Command Line Tool xxd:
    # ENCODE:
    echo "cyber" | xxd -p
    --> 63796265720a
    # DECODE:
    echo "63796265720a" | xxd -r -p
    --> cyber
- Base64
  - Base 64 (64 total characters)
    char range is a-z,A-Z,0-9,+, and /
    a Base64 string will always be a multiple of 4, so = and == are sometimes used to pad out the string to ensure it is a multiple of 4
  - # ENCODE:
    echo "cyber" | base64
    --> Y3liZXIK
    # DECODE:
    echo "Y3liZXIK" | base64 -d
    --> cyber

5. Security Concepts

- 3 Essential Principles
  - Least privilege - software should be given only those privileges that it needs to complete its task
    - Do not run everything as / root
  - Open design - security of software, and of what that software provides, should not depend on the secrecy of its design or implementation
    no security through obscurity
    most important for cryptographic processes
  - Abstraction - hide the internals of each layer, make only the interfaces available

6. User & File Mgmt

- Organization in Linux
  - Users (which belong to one or more groups)
    Root (admin)
    Users
    Service accounts (simply users with no login access)

- Objects
  Files - everything is represented as a file (programs, network connections, processes, etc)
  Directories

- Discretionary Access Control
  - Users have 3 actions based on permissions: read, write, execute
  - each object has 3 sets of permissions: user-owner, group-owner, everyone else
  - DAC is enforced by Linux kernel, Root can modify DAC and load/unload kernel modules
- Transaction Handling
  - Subject (user or process) > Action (R, W, X) > Object (file, directory, or special file)

7. Commands Review

- core commmands
  - echo
    cat
    man
    passwd
    history
- navigation commands
  - pwd
    ls
    cd
    .
    ..
- file commands
  - touch
    mkdir
    mv
    cp
    rm
    rmdir
    find
    locate
    updatedb
    which
    whereis
- filter commands

- head
  tail
  wc
  sort
  uniq
  tr
- output manipulation
  - >
    2>
    >>
    2>>
    |
    STDOUT
    STDERR
    STDIN
    /dev/null
- Analysis Commands
  - file
    strings
    base64
    xxd
    zip
    gzip
    tar
    unrar
- user + group commands
  - adduser / useradd
    addgroup / groupadd
    usermod
    deluser / userdel
    whoami
    getent
    /etc/passwd
    /etc/shadow
    /etc/group
- Permission commands
  - ls -l
    chmod
    chown
    chgrp

- Environment commands
  - /bin/bash
    /bin/sh
    ps
    kill
    crontab -e
    env
    apt
    vim
    nano