Question on command “lsof” list open files-

Q1: Find which process is using a specific file
Scenario: A user complains that they cannot delete the file /var/log/app.log.

- **Task:** Identify which process is using the file and terminate it.
- **Expected Command:**

lsof /var/log/app.log
(Followed by kill <PID> if necessary.)

Q2: Find all open files by a specific user
Scenario: The user john is running multiple processes, and you need to check which files are being accessed by them.

- **Task:** List all open files by user john.

Q3: Identify which process is using a specific port
Scenario: A web server fails to start on port 8080, likely because another process is using it.

- **Task:** Find the process ID (PID) of the application occupying port 8080.

Q4: Find all open network connections
Scenario: The system is experiencing high network activity.

- **Task:** Find all active network connections along with their associated processes.

🔹 **Intermediate-Level Questions**
Q5: Find and kill a process holding a deleted file
Scenario: A process is holding a deleted log file, consuming disk space.

- **Task:** Find the process keeping a deleted file open and release the space.

Q6: Check which files are open in /tmp
Scenario: The /tmp directory is filling up quickly.

- **Task:** List all open files in /tmp.

Q7: Identify processes writing to a large file
Scenario: A large file is growing rapidly, and you need to find out which process is writing to it.

- **Task:** Identify the process and its PID.

Q8: Detect unauthorized file access
Scenario: A security team suspects that an unauthorized user is accessing /etc/passwd.

- **Task:** Identify which processes are opening /etc/passwd.

🔹 **Advanced-Level Questions**

Q9: Identify the most file descriptor-intensive processes

Scenario: The system is running out of file descriptors (Too many open files error).

- **Task:** Find processes with a high number of open files.
- **Expected Command:**

lsof | awk '{print $2}' | sort | uniq -c | sort -nr | head

Q10: Find zombie processes holding open files

Scenario: A zombie process is keeping certain files locked.

- **Task:** Identify zombie processes with open files.

Q11: List all open files of a particular process

Scenario: You want to debug a running database server (e.g., MySQL).

- **Task:** Find all files opened by the MySQL process.

Q12: Monitor real-time file access on a directory

Scenario: You suspect that a process is modifying files in /var/www/html/.

- **Task:** Monitor live file accesses.

Q13: Identify processes using shared libraries

Scenario: A shared library (libssl.so) needs an update, but you cannot replace it because it's in use.

- **Task:** Identify which processes are using libssl.so.

Q14: Find files opened by a process within a specific time

Scenario: You want to check if a process accessed a file in the last 10 minutes.

- **Task:** List open files modified recently by PID 1234.

Q15: Detect file locks causing issues

Scenario: A user reports that a file cannot be accessed because it is locked.

- **Task:** Identify the process holding a lock on the file /var/data.db.

Q16: Why does lsof -i :22 **sometimes return no output?**

- **Hint:** Consider which service listens on port 22 and if it's running or not.

Q17: If lsof **does not return any results, but you are sure a file is in use, what could be the reason?**

- **Hint:** Consider **permissions** and **user privileges**.

Questions on - traceroute

1️⃣ Basic Understanding of traceroute

Q1: What is the purpose of traceroute, and how does it work?

- **Expected Answer:** Explain how traceroute uses **TTL (Time to Live)** and **ICMP/UDP** to map the route of packets.

Q2: What is the difference between traceroute and ping?

- **Hint:** Compare functionality, TTL usage, and response types.

Q3: What command would you use to trace the route to google.com?

- **Expected Command:**

traceroute google.com

2️⃣ Analyzing Network Latency & Hops

Q4: You run traceroute and see some hops marked with * * *. What does this mean?

- **Expected Answer: Packet loss**, **firewall blocking ICMP**, or **rate-limiting of ICMP responses**.

Q5: How can you check if a specific router is introducing high latency in a connection?

- **Hint:** Compare response times between consecutive hops in the traceroute output.

Q6: A traceroute shows an IP address that does not resolve to a hostname. What does this indicate?

- **Hint:** Could be a **private/internal IP**, **no reverse DNS setup**, or **firewall restrictions**.

3️⃣ Handling Different Protocols and Ports

Q7: By default, traceroute uses UDP. How can you make it use ICMP instead?

- **Expected Command:**

traceroute -I <destination>

Q8: How do you run traceroute on port 443 (HTTPS)?

- **Expected Command:**

traceroute -T -p 443 <destination>

Q9: You suspect that a firewall is blocking ICMP packets. How can you check this using traceroute?

- **Hint:** Use -T (TCP) or -U (UDP) options instead of default ICMP.

4️⃣ Real-World Network Troubleshooting Scenarios

Q10: A user reports slow internet access. How can traceroute help in diagnosing the issue?

- **Hint:** Look for **high latency** or **packet loss** at specific hops.

Q11: You see a loop in the traceroute output (e.g., Hop 3 -> Hop 4 -> Hop 3). What could be the issue?

- **Hint:** Possible **routing misconfiguration** or **load balancer issues**.

Q12: You run traceroute and notice a sudden increase in response times after a certain hop. What does this indicate?

- **Hint:** Possible **network congestion** or **ISP throttling**.

5️⃣ Advanced Use Cases & Debugging

Q13: How can you limit the number of hops traceroute checks?

- **Expected Command:**

traceroute -m 10 <destination>

Q14: How can you increase the number of packets sent per hop to get more accurate results?

- **Expected Command:**

traceroute -q 5 <destination>

Q15: If traceroute fails completely and shows Destination Unreachable, what steps will you take to diagnose?

- **Hint:**
- Check if the destination is reachable via ping.
- Use traceroute -T (TCP mode) in case ICMP is blocked.
- Verify firewall rules or ISP restrictions.

Bonus Challenge

Q16: You run traceroute to a website, and the last few hops show private IP addresses (192.168.x.x or 10.x.x.x). What does this mean?

- **Hint:** Could be a **CDN**, **NAT setup**, or **internal routing used by ISPs**.

Would you like **practical lab exercises** for engineers to solve, or should I create **scenario-based assignments**? 🚀

Q17: Scenario: You are a system administrator managing a shared directory /data on a Linux server used by multiple departments in your organization. The directory contains subdirectories for different departments, and each department's directory requires specific access control based on roles and hierarchy.

Requirements:

Finance Team (group finance_team):

Should have full access (read, write, execute) to /data/finance.
Files created within this directory should automatically belong to the finance_team group.
HR Team (group hr_team):

Should have read and execute permissions on /data/finance but must not modify or delete any files.
IT Team (group it_team):

Should have full access (read, write, execute) to /data/it.
They should have read-only access to /data/finance and /data/hr.
Department Heads (group dept_heads):

Should have full access to all subdirectories under /data.
User-specific Access:

User bob (Finance Head) should have exclusive access to /data/finance/reports. No other user or group should have access to this subdirectory.
Default ACLs:

Any new files or subdirectories created within /data must inherit the correct permissions based on their respective department.

Tasks:
Set up the directory structure under /data as follows:

/data/finance
/data/hr
/data/it
/data/finance/reports
Configure appropriate permissions for the groups finance_team, hr_team, it_team, and dept_heads as per the requirements.

Configure specific permissions for the user bob to ensure exclusive access to /data/finance/reports.

File System Security Scenarios

Scenario 1: Secure Departmental Backups

You are administering a Linux server for a company that requires secure backups for each department.

1. **Setup Departmental Backup Directories**:
   Create separate backup directories for each department under `/backups/` (e.g., `/backups/sales/`, `/backups/finance/`, `/backups/engineering/`).
2. **Encrypt Backup Data**:
   Encrypt all files in `/backups/sales/` using `gpg` with a passphrase. Document the process of encrypting and decrypting these backups.

3. **Prevent Accidental Deletion of Backups**:
   Use file attributes (`chattr`) to protect the files in `/backups/finance/` from deletion or modification.
4. **Access Control for Shared Backups**:
   The engineering team needs read-only access to the finance backups. Use ACLs to configure this permission.
5. **Audit Backup Directory Access**:
   Configure audit rules using `auditctl` to log all access to the `/backups/engineering/` directory.

Scenario 2: Isolating Sensitive Data

Your organization handles sensitive client data, and certain files must only be accessible to a specific team.

1. **Setup Restricted Directories**:
   Create a directory `/sensitive_data/` and restrict access to only the `client_services` group.
2. **Mask Sensitive Files**:
   Ensure that filenames in `/sensitive_data/` are not visible to unauthorized users. Use `chmod` or `setfacl` to enforce this.
3. **Prevent Unauthorized Copying**:
   Restrict the use of USB devices on the system to prevent unauthorized copying of sensitive files.
4. **Encrypt Sensitive Files**:
   Use `openssl` or `gpg` to encrypt files in `/sensitive_data/` before sharing them with external clients. Automate this process using a script.
5. **Log All Access**:
   Use the `auditd` service to log any read or write operations in the `/sensitive_data/` directory for compliance purposes.

User and Group Management Scenarios

Scenario 1: Configuring Temporary Employees

You have hired temporary employees who require limited access to specific resources.

1. **Create Temporary Groups**:
   Create a new group `temporary_staff`.

2. **Add Temporary Users**:
   Create user accounts for two temporary employees, `temp1` and `temp2`, and add them to the `temporary_staff` group.
3. **Restrict Temporary User Access**:
   Configure `/etc/security/time.conf` to allow `temp1` and `temp2` to log in only during business hours (9 AM to 6 PM).
4. **Set Expiry for Temporary Accounts**:
   Ensure `temp1` and `temp2` accounts expire automatically in 30 days.
5. **Limit Temporary User Disk Usage**:
   Configure quotas to restrict disk usage for the `temporary_staff` group to 1GB per user.

Scenario 2: Managing Role Transitions

A team member, Eve, has moved from the `support` department to the `development` department.

1. **Update Group Memberships**:
   Remove Eve from the `support` group and add her to the `development` group.
2. **Transfer Ownership of Files**:
   All files owned by Eve in `/support_data/` must now be moved to `/development_data/` and have their ownership updated to reflect her new role.
3. **Archive Old Data**:
   Archive Eve's old files in `/support_data/` using `tar` and move them to `/archive/support/`.
4. **Audit Eve's Activities**:
   Use `last` and `auditd` logs to ensure Eve's activities comply with company policies during her transition period.
5. **Update Login Shell**:
   Eve needs a new development shell environment. Change her default shell to `/bin/zsh` and ensure she has appropriate configuration files.

Scenario 3: Managing Contractor Accounts

Contractors have joined your company for a short-term project. They require shared access to resources but no access to sensitive company data.

1. **Create Contractor Accounts**:
   Add accounts for `contractor1` and `contractor2` with home directories in `/contractors/`.
2. **Shared Directory Setup**:
   Create a shared directory `/project_resources/` accessible to both contractors. Ensure that any files created in this directory are automatically owned by the group `contractors`.
3. **Restrict File System Access**:
   Configure `/etc/fstab` to mount `/project_resources/` with `noexec` and `nodev` options for added security.
4. **Disable SSH Access**:
   Block SSH access for contractor accounts using `/etc/ssh/sshd_config`.
5. **Log Contractor Activities**:
   Use `acct` to monitor and log the activities of contractor accounts.

Basic Level

1. **Managing File Permissions**:
   A file named `/shared/data.log` must be read-only for all users except a user named "Admin" who should have full permissions. Configure the appropriate file permissions.
2. **Group File Ownership**:
   A directory `/projects/team_alpha` is owned by the group `team_alpha`. Ensure that all new files created in this directory automatically belong to the `team_alpha` group, irrespective of the file creator's primary group.
3. **Create User with Expiration**:
   A temporary intern, Sam, is joining for a 1-month internship. Create a user account `sam_intern` that expires in 30 days. Ensure the account has a home directory and `/bin/bash` as the default shell.
4. **Setting Directory Quotas**:
   Assign a disk quota of 1GB to a user named `developer` for the `/dev_projects` directory. Ensure the user cannot exceed this limit.
5. **Restricting File Deletion**:
   A shared directory `/data/shared_docs` must allow all users in the group `team_shared` to read and write files, but no one should be able to delete files except the directory owner.

Intermediate Level

6. **Managing SUID/GUID**:
   A script `/usr/local/bin/db_backup.sh` needs to run with the privileges of its owner

(a user named `backup_admin`) regardless of the user executing it. Configure this using SUID or GUID and justify your choice.

7. **Configuring ACLs**:
   A directory `/data/projects` needs to be accessible to the group `project_managers` with read-write permissions. Additionally, the group `qa_team` should have read-only access to this directory.
8. **Custom Shell Access**:
   A user `analytics` should only execute a predefined set of scripts located in `/data/scripts/` and should not have access to an interactive shell. Configure this restriction.
9. **Shared Directory with Default Permissions**:
   Create a directory `/shared/finance_docs` where files created by any member of the `finance_team` group should have default permissions of `rw-rw----`.
10. **Home Directory Protection**:
    Ensure that the home directories of all users are accessible only to their respective owners and not to any other users, even if they belong to the same group.

Advanced Level

11. **User Account Automation**:
    Write a script to automatically create 10 users (`user1` to `user10`), assign them to a group `temp_users`, set default passwords, and expire the passwords after the first login.
12. **Group Migration**:
    A group `developers` is being renamed to `software_team`. Ensure that all existing users and file permissions associated with the `developers` group are seamlessly migrated to the new group.
13. **Securing Logs**:
    A directory `/var/log/custom_logs` must store sensitive application logs. Only the `sysadmin` group should have full access, while others should be denied access. Protect this directory further by applying the `immutable` attribute to its files.
14. **Audit File Access**:
    Monitor and log all read/write access to the file `/etc/secure_data.conf`. Ensure the logs are stored in `/var/log/audit/secure_data.log`.
15. **Backup and Restore User Accounts**:
    Back up all user accounts (along with their home directories) to a tar archive and store it in `/backups/user_data.tar.gz`. Simulate a user deletion and restore the user account along with their files from the backup.

Scenario-Based

16. **Isolated Development Environment**:
    Create an isolated development environment where:
    - Users in the group `devs` can access `/dev_env/`.
    - Processes run within this directory cannot access files outside of `/dev_env/`.
17. **Temporary Privileges**:
    Grant a user `temp_admin` sudo privileges to manage services (e.g., restart Apache or Nginx) for 7 days. Ensure this privilege is automatically revoked after the 7-day period.
18. **Service-Specific User**:
    Create a user `ftp_user` dedicated to accessing the FTP service. Ensure this user has no shell access and cannot log in interactively.
19. **Cross-Department Collaboration**:
    The `marketing` team needs read-write access to `/company/shared/marketing_resources`, while the `sales` team should only have read access. Configure this with ACLs and verify the permissions.
20. **File Integrity Monitoring**:
    Set up a monitoring system to track any changes (create, modify, delete) in the `/secure_data/` directory. Alert the `security_admin` user whenever a change occurs.

Process Management Scenarios

1. **Monitoring CPU Usage**:
   A server is experiencing high CPU usage. Use a command to identify the process consuming the most CPU resources. Kill the process only if it belongs to a non-critical service.
2. **Running a Process in the Background**:
   A script `/home/user/backup.sh` needs to run as a background process, even if the user logs out. Execute the script in such a way that it runs uninterrupted and logs its output to `/var/log/backup.log`.
3. **Process Prioritization**:
   A data analysis task `data_analyze` is running but consuming too many resources. Change its priority to run with a lower CPU priority.
4. **Killing a Stuck Process**:
   A process with the PID `1234` has become unresponsive. Write the steps to:
   - Send a graceful termination signal.
   - Force kill the process if it does not terminate.
5. **Debugging a Hung Application**:
   An application `/usr/bin/myapp` is not responding. Use `strace` or `gdb` to debug the process and identify the reason for the hang.

6. **Limiting Process Resources**:
   A user is running a program that consumes excessive memory. Configure a limit so that their processes cannot exceed 1GB of memory.
7. **Process Ownership Transfer**:
   A process running under the user `temp_user` needs to be transferred to `admin_user` without stopping it. Perform the transfer securely.
8. **Scheduled Task Monitoring**:
   A scheduled `cron` job `/usr/local/bin/cleanup.sh` runs every night but frequently fails. Find and debug the reason for the failure by checking the logs.
9. **Persistent Process Identification**:
   Identify all instances of a service `nginx`, along with their parent-child relationships, and write the steps to stop and restart them in an orderly manner.
10. **Isolating a Process**:
    A resource-intensive process `simulate_weather` needs to run in isolation to prevent it from interfering with other users. Create a cgroup to restrict its CPU and memory usage.

Network Command Scenarios

11. **Port Availability**:
    A web application needs to run on port `8080`, but it fails to start. Use appropriate commands to:
    - Check if the port is already in use.
    - Identify which process is using it.
    - Terminate the process if needed and free the port.
12. **Network Connectivity Testing**:
    A server cannot connect to an external database at `192.168.1.50` on port `3306`. Troubleshoot this by:
    - Checking the server's connectivity to the database using `ping` and `telnet`.
    - Verifying if the port is open using `nc` or `nmap`.
13. **Traffic Analysis**:
    A network admin suspects unusual traffic to and from the server. Use `tcpdump` to capture packets on interface `eth0` and filter traffic on port `22`. Save the output to a file for further analysis.
14. **Bandwidth Monitoring**:
    The IT team wants to monitor real-time network bandwidth usage. Use a command to show network bandwidth statistics per interface.
15. **Blocking an IP**:
    An IP address `203.0.113.5` is making suspicious requests to the server. Use `iptables` or `ufw` to block all incoming traffic from this IP.
16. **Network Service Status**:
    A DNS resolution issue is reported on a server.

    - Verify if the `named` service is running.
    - Restart the service if it is stopped.
    - Check DNS resolution for `example.com`.
17. **Routing Table Issues**:
    A user reports connectivity issues to a specific subnet `10.1.0.0/24`. Use `ip route` to check the server's routing table and add a static route if needed.
18. **Persistent Network Configuration**:
    The server needs a static IP address `192.168.10.100/24` with a gateway `192.168.10.1`. Configure this in a way that persists across reboots.
19. **Firewall Configuration**:
    Allow only HTTP and HTTPS traffic on a server using `ufw`. Deny all other incoming traffic while allowing all outgoing traffic. Verify the rules after configuration.
20. **SSH Troubleshooting**:
    A user cannot connect to the server via SSH. Investigate the issue by:
    - Checking the `sshd` service status.
    - Verifying that port `22` is open.
    - Confirming firewall rules and allowing SSH if necessary.

Advanced Process & Network Scenarios

21. **Load Balancer Simulation**:
    Simulate a load balancer on the server by forwarding traffic received on port `80` to three backend servers on ports `8081`, `8082`, and `8083` using `iptables` or `nftables`.
22. **Secure File Transfer**:
    A file needs to be securely transferred between two servers `A` and `B` over an untrusted network. Use `scp` or `rsync` with encryption to transfer the file.
23. **Persistent SSH Tunnels**:
    Set up an SSH tunnel to forward traffic from a local port `9090` to a remote server's port `3306` for secure database access. Ensure the tunnel persists even if the SSH connection drops.
24. **Network Namespace Isolation**:
    Create a network namespace and configure it to use a virtual network interface `veth0`. Assign an IP `10.0.0.1/24` to the namespace and verify its isolation from the host network.
25. **Process and Network Logging**:
    A suspicious process on the server is making outbound connections to unknown IPs. Use `lsof` and `netstat` to monitor and log all such connections in real-time.