Step 1: Explore Confidentiality

Objective: Understand how confidentiality is maintained and identify security violations when unauthorized access occurs.

1. Create a File Containing Sensitive Information
   - Open a terminal or command prompt.
   - Create a text file named sensitive_file.txt:
     - Linux: echo "Confidential Data: Usernames and Passwords" > sensitive_file.txt
     - Windows: Open Notepad, type "Confidential Data: Usernames and Passwords", and save the file as sensitive_file.txt.

2. Restrict File Permissions
   - Set file permissions so only the owner can access it:
     - Linux: Run chmod 600 sensitive_file.txt.
       - Explanation: The 600 permission allows the owner to read/write the file but denies access to others.
     - Windows: Right-click the file → Properties → Security → Edit permissions → Deny access for all users except the owner.

3. Simulate Unauthorized Access
   - Switch to another user or simulate unauthorized access:
     - Linux: Use su or create a new user, then try accessing the file: cat sensitive_file.txt.
     - Windows: Switch user accounts or create a new user, then try opening the file.
   - Observe the error message (e.g., "Permission denied").
   - Discussion: Analyze how this protects confidentiality. Discuss real-world examples, such as protecting medical or financial records.

```
student@student-VMware-Virtual-Platform:~$ echo "Confidential Data: Usernames and Passwords" > sensitive_file.txt
student@student-VMware-Virtual-Platform:~$ cat sensitive_file.txt
Confidential Data: Usernames and Passwords
student@student-VMware-Virtual-Platform:~$ chmod 600 sensitive_file.txt.
chmod: cannot access 'sensitive_file.txt.': No such file or directory
student@student-VMware-Virtual-Platform:~$ echo "Confidential Data: Usernames and Passwords" > sensitive_file.txt
student@student-VMware-Virtual-Platform:~$ cat sensitive_file.txt
Confidential Data: Usernames and Passwords
student@student-VMware-Virtual-Platform:~$ chmod 600 sensitive_file.txt
student@student-VMware-Virtual-Platform:~$ ls -l sensitive_file.txt
-rw------- 1 student student 43 Mar 27 16:15 sensitive_file.txt
student@student-VMware-Virtual-Platform:~$ su alice
Password:
alice@student-VMware-Virtual-Platform:/home/student$ cat /home/student/sensitive_file.txt
cat: /home/student/sensitive_file.txt: Permission denied
alice@student-VMware-Virtual-Platform:/home/student$
```

Step 2: Analyze Integrity

Objective: Understand how data integrity can be compromised and verify its integrity using

hashing.

1. Create or Access a Log File
   - Use an existing log file or create a simulated one:
     - Linux: sudo nano /var/log/syslog (requires root access).
     - Windows: Open Event Viewer (eventvwr) or create a text file named logfile.txt.

2. Modify the Log File (Simulate Unauthorized Changes)
   - Add or change log entries to simulate a security violation:
     - Linux: Edit the file: sudo nano /var/log/syslog → Add a fake entry: Jan 1 12:00:00 UnauthorizedAccess: Admin login.
     - Windows: Open logfile.txt in Notepad and add UnauthorizedAccess: Admin login.

3. Verify Integrity with Hashing
   - Calculate the file's hash before and after modification:
     - Linux: Use sha256sum logfile.txt and note the hash.
     - Windows: Use PowerShell: Get-FileHash .\logfile.txt -Algorithm SHA256

   - Observe the hash difference.
   - Discussion: Discuss how unauthorized changes compromise data integrity. Relate this to tampering scenarios in real-world applications, such as altering financial records or audit logs.

```
alice@student-VMware-Virtual-Platform:/home/student$ sudo nano /var/log/syslog
alice@student-VMware-Virtual-Platform:/home/student$ sha256sum logfile.txt
sha256sum: logfile.txt: Permission denied
alice@student-VMware-Virtual-Platform:/home/student$ sudo nano /var/log/syslog
alice@student-VMware-Virtual-Platform:/home/student$ sha256sum logfile.txt
sha256sum: logfile.txt: Permission denied
alice@student-VMware-Virtual-Platform:/home/student$ ls
ls: cannot open directory '.': Permission denied
alice@student-VMware-Virtual-Platform:/home/student$
```

Step 3: Examine Availability

Objective: Explore how system availability is affected during an attack or resource overload.

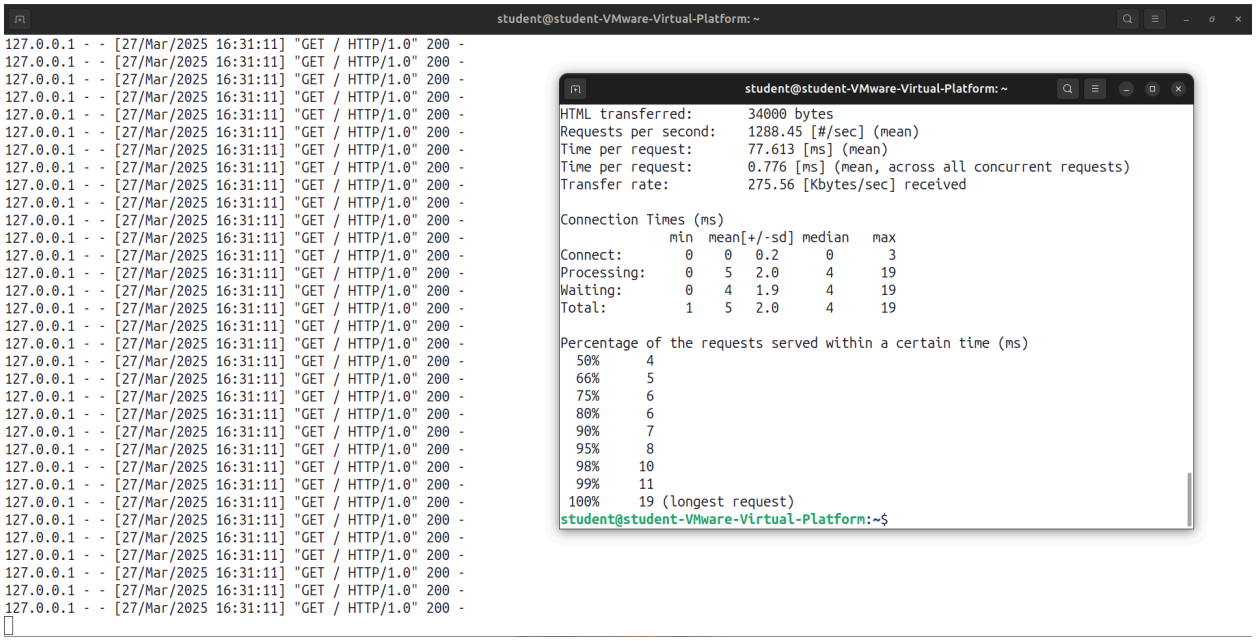
1. Set Up a Simple Web Server (Optional)
   - Linux: Use Python to start a basic web server: python3 -m http.server 8080
   - Windows: Use IIS or WAMP/XAMPP to set up a local server.

2. Simulate Denial-of-Service (DoS) Attack
   - Use a tool to overload the server with requests:
     - o Linux: Install and use ab (Apache Benchmark):
       ab -n 1000 -c 100 http://localhost:8080/
       -n: Total number of requests.
       -c: Number of concurrent requests.
     - Windows: Use a custom PowerShell script or any load-testing tool like JMeter.

3. Monitor Server Behavior
   - Observe the server response time during the attack:
     - Linux: Check server logs or terminal output for delays or errors.
     - Windows: Use Task Manager or Resource Monitor to track CPU and network usage.
   - Note any timeouts or connection refusals.

4. Restore Normal Operations
   - Stop the attack and ensure normal availability:
     - Linux: Terminate the ab command or server process (Ctrl+C).
     - Windows: Stop the server or restart it via IIS Manager.
   - Verify that the server is responsive again.

5. Discussion
   - Relate this to real-world examples of denial-of-service attacks on popular websites or applications. Discuss strategies to mitigate such attacks, such as rate-limiting, firewalls, or load balancers.



Step 4: Document Observations

a. For each step, document:

- Actions performed.
- Observations (e.g., error messages, response times).
- Screenshots of commands and results.

b. Analyze how each violation impacts the overall security of the system.