Vulnerability Assessment Penetration Testing (VAPT) Lab

Experiment 1. Implementation of networking commands to retrieve device information and network configuration using Linux.

Aim: To study and implement various networking commands used to retrieve device information and network configuration in both **Linux** and **Windows** operating systems.

Theory:

Networking commands are essential tools to analyze and manage network configurations in an operating system. They help retrieve critical details such as **IP address, MAC address, interface status, routing table, DNS configuration, and connectivity**.

In **Linux**, commands like ip, ss, ethtool, and nmcli provide detailed system and network interface information. In **Windows**, commands like ipconfig, netsh, and PowerShell cmdlets like Get-NetIPConfiguration offer similar insights.

These commands are widely used for **network troubleshooting**, **interface monitoring**, and **diagnostic purposes**.

Commands and Description (Linux):

No.	Task	Command	Purpose
1	Check Network Interfaces	ip addr show	Displays all interfaces with IP (IPv4/IPv6), MAC address, and interface states (UP/DOWN).
2	Check Active Network Connections	ss -tuln *(Alt: netstat -tuln)*	Lists active sockets: TCP (-t), UDP (-u), listening (-l), and numeric output (-n).
3	Display Routing Table	ip route show	Shows routing details including default gateway, connected subnets, and route decisions.
4	Display DNS Configuration	cat /etc/resolv.conf	Shows DNS servers the system is using.
5	Check Interface Link Status	ethtool eth0	Displays link status, speed, duplex, and RX/TX statistics. *(Install: sudo apt install ethtool)*
6	View System-Wide Network Config	nmcli device show	Provides full configuration of each network interface including IP, MAC, DNS, and gateway.
7	Check Hostname and IP	hostname -I	Displays system hostname and assigned IP address.
8	View ARP Table	ip neigh	Displays local MAC and IP address mappings.
9	Test Connectivity	ping -c 4 google.com	Tests network connectivity and DNS resolution.

No.	Task	Command	Purpose
10	Trace Network Path	traceroute google.com or tracepath google.com	Shows route packets take to the destination. *(Install: sudo apt install traceroute)*

Conclusion:
Using these commands, essential **device and network configuration details** can be easily retrieved and analyzed on both Linux and Windows systems. They are invaluable for **system diagnostics, network monitoring, and troubleshooting**.

Experiment 2. Apply network mapping (NMAP) to discover network.

Aim
Perform host identification and enumerate running services using Nmap.

Theory
Host discovery reduces large IP ranges to a list of active hosts before heavier scanning. Nmap's discovery goes beyond simple ICMP ping and includes combinations of TCP SYN/ACK, UDP, SCTP, and ICMP probes to elicit responses that prove an IP is active. Defaults (unless overridden) are equivalent to -PE -PS443 -PA80 -PP. ARP (IPv4) and Neighbor Discovery (IPv6) are used automatically on local Ethernet because they are faster and more reliable.

You can skip discovery (-Pn), only list targets (-sL), or perform a ping-only sweep (-sn). Mix and match probe types (e.g., -PS, -PA, -PU, -PY, -PE, -PP, -PM, -PO) to increase chances of detecting hosts behind filters. Use --disable-arp-ping to turn off implicit ARP/ND on local networks. --discovery-ignore-rst prevents treating spoofed RSTs as proof of life. Post-scan traceroutes can be requested with --traceroute.

Host Discovery Options (short descriptions)

- **-sL — List Scan**
  Lists targets (does not send probes). Performs reverse-DNS lookups by default. Useful as a sanity check.
- **-sn — No port scan (Ping Sweep)**
  Runs host discovery only (ICMP, TCP SYN to 443, TCP ACK to 80, ICMP timestamp by default). Good for counting or lightly reconnoitering hosts.
- **-Pn — No ping**
  Skip host discovery and treat all targets as up (useful when discovery is blocked). ARP/ND still used on local Ethernet unless disabled.
- **-PS<ports> — TCP SYN Ping**
  Sends TCP SYN probes to listed ports (default 80). A SYN/ACK or RST indicates host is up.
- **-PA<ports> — TCP ACK Ping**
  Sends TCP ACK probes (default 80). RST responses reveal live hosts — handy to bypass some filters.

- **-PU<ports> — UDP Ping**
  Sends UDP packets (default port 40125); ICMP port-unreachable replies indicate hosts are up.
- **-PY<ports> — SCTP INIT Ping**
  Sends minimal SCTP INIT chunks; INIT-ACK or ABORT reveal host status.
- **-PE, -PP, -PM — ICMP Ping Types**
  -PE echo request, -PP timestamp request, -PM netmask request. Useful when ICMP echo is allowed or other ICMP types are overlooked.
- **-PO<protocols> — IP Protocol Ping**
  Sends bare IP packets with chosen protocol numbers; responses or protocol-unreachable ICMP indicate host availability.
- **--disable-arp-ping**
  Stops implicit ARP/ND on local Ethernet (useful when proxy ARP gives false positives).
- **--discovery-ignore-rst**
  Ignore RSTs during discovery (useful if firewalls spoof RSTs).
- **--traceroute**
  Perform traceroute post-scan using the port/protocol information learned.

Service Discovery (port & version scanning — concise)

- **-sV** — Service/version detection (probes service banners).
- **-p** — Specify ports or ranges, e.g., -p 1-1000 or -p 22,80,443.
- **-sS** — TCP SYN (stealth) scan — requires root for raw packets.
- **-sT** — TCP connect() scan (unprivileged fallback).
- **-sU** — UDP scan.
- **-O** — OS detection.
- **-sC or --script=default** — Run default NSE scripts.
- **--traceroute** — Add traceroute to discovery.

Step-by-Step Guide (To Be Done In Practical)

1. **Install Nmap (Debian/Ubuntu):**
2. sudo apt update && sudo apt install nmap
3. **List potential targets:**
4. nmap -sL 192.168.1.0/24
5. **Ping sweep (host discovery only):**
6. nmap -sn 192.168.1.0/24
7. **Stealthy TCP SYN ping sweep:**
8. nmap -PS80,443 -sn 192.168.1.0/24
9. **Service & version scan on a live host (ports 1–1000):**
10. sudo nmap -sV -p 1-1000 192.168.1.10
11. **Aggressive scan with OS, scripts, and traceroute:**
12. sudo nmap -sV -O --traceroute -sC 192.168.1.10

Conclusion

Nmap’s flexible host discovery and service enumeration options let you efficiently find live hosts and enumerate services while adapting probes to bypass filters and minimize noise.

Experiment 3. Implement network packet sniffing using Wireshark.

Aim

Implement packet sniffing to capture and analyze network traffic using **Wireshark**.

Theory

Packet sniffing is the process of capturing packets that traverse a network interface to inspect their headers and payloads. **Wireshark** is a GUI-based network protocol analyzer that captures live traffic or reads capture files (PCAP). It uses **capture filters** (applied while recording) to limit which packets are saved, and **display filters** (applied after capture) to isolate packets of interest. Promiscuous mode allows an interface to receive all frames on the network segment; without it, the NIC sees only traffic addressed to it. Packet sniffing supports troubleshooting, protocol analysis, performance measurement, and security investigations — always performed with authorization because it can reveal sensitive data.

Tools / Commands

- Install on Debian/Ubuntu:

  sudo apt update && sudo apt install wireshark

- Install on Windows / macOS: Download installer from Wireshark website.
- Command-line capture (tshark):

  sudo tshark -i eth0 -w capture.pcap

Capture Filters

- Capture only TCP traffic: tcp
- Capture UDP on port 53: udp port 53
- Capture only traffic to/from an IP: host 192.168.1.10
- Capture specific port range: portrange 1000-2000

Display Filters (applied after capture / in GUI)

- Show HTTP packets: http
- Show packets with IP address X: ip.addr == 192.168.1.10
- Show TCP streams on port 443: tcp.port == 443
- Show DNS responses: dns.flags.response == 1

Step-by-Step Procedure

1. **Install Wireshark / tshark.**
2. **Select interface** (Ethernet, Wi-Fi) in Wireshark.
3. **Set a capture filter** if needed (e.g., host 192.168.1.10), enable/disabling **promiscuous mode** as required.
4. **Start capture** and reproduce the network activity you need to observe.
5. **Stop capture** and use display filters to narrow results (e.g., http or ip.addr==x).

6. **Follow TCP stream**: right-click a TCP packet → **Follow** → **TCP Stream** to view session payloads.
7. **Use built-in statistics**: Statistics → Protocol Hierarchy, IO Graphs, Endpoints, Conversations for traffic summaries.
8. **Save capture**: File → Save As (PCAP/PCAPNG).
9. **Automated CLI capture** (example):

   sudo tshark -i eth0 -f "tcp port 80" -a duration:60 -w web_traffic.pcap

Conclusion

Wireshark provides powerful packet sniffing and analysis via capture/display filters, stream reassembly, and rich statistics—making it indispensable for troubleshooting and security analysis when used responsibly.

Experiment 4. Implementation of Kioptrix machine using virtual box.

Aim

To safely set up and configure the **Kioptrix vulnerable virtual machine (VM)** using **VirtualBox** for practicing ethical hacking in an isolated lab environment.

Theory

Kioptrix is a series of intentionally vulnerable virtual machines designed for **ethical hacking practice** and **penetration testing**. These VMs simulate real-world vulnerabilities and are commonly used for **CTF-style learning**. To ensure safety, Kioptrix must only be run in an **isolated lab network** that you own or control.

Virtualization software like **VirtualBox** allows running multiple virtual machines (such as **Kioptrix** and **Kali Linux**) on a single host. Using **Host-only** or **Internal networking**, you can create an **isolated environment** where the attacker (Kali) and target (Kioptrix) communicate securely without exposing the vulnerable VM to the Internet.

Snapshots are essential to **save system states**, allowing you to revert the VM to a clean condition after testing. This controlled setup helps learners safely explore vulnerabilities and improve cybersecurity skills without real-world risk.

Requirements

- **VirtualBox** (latest version) and optional **Extension Pack**
- **Kioptrix VM file** (.OVA or ISO from a trusted source)
- **Attacker VM (Kali Linux)**
- **2–4 GB RAM** on host system
- **Host-only / Internal network** for isolation

Procedure (Step-by-Step – short form)

1. Import or Create the Kioptrix VM

- **If OVA file available:**
  VirtualBox → *File* → *Import Appliance* → Select .ova → *Next* → Review → *Import*.
- **If ISO available:**
  New VM → *Linux* → *Other Linux (32-bit)* → Set RAM (256–512 MB) → Create virtual disk (8–20 GB) → Attach ISO → Install OS.

2. Configure VM Hardware

- CPU: 1 core
- RAM: 256–1024 MB
- Video: Default
- Storage: As provided or custom 8–20 GB

3. Configure Network (Isolation + Connectivity)

- **Kioptrix (Target):**
  Adapter 1 → *Host-only Adapter* → vboxnet0
- **Kali (Attacker):**
  Adapter 1 → *NAT* (for Internet access)
  Adapter 2 → *Host-only Adapter* → same vboxnet0

This ensures both VMs communicate within an isolated network, while only Kali has optional Internet access.

4. Create / Verify Host-only Network

- VirtualBox → *File* → *Host Network Manager* → Create vboxnet0
- DHCP optional (enable or set static IP manually)

5. Start and Verify

- Boot both VMs
- In Kali, check interfaces:
- ip a
- Discover Kioptrix IP:
- nmap -sn 192.168.56.0/24
- Confirm connectivity and perform service scan:
- nmap -sC -sV -p- <target-ip>

6. Take Snapshot (Before Testing)

- VirtualBox → *Snapshots Tab* → *Take Snapshot* → Name it “Clean-Install”.

Conclusion

Kioptrix provides a safe, isolated environment for ethical hacking and penetration testing practice. Using VirtualBox with host-only networking and snapshots ensures secure, controlled experimentation without affecting real networks.

Experiment 5. Implementation of basic pen testing using VMware.

Aim - To implement a basic penetration-testing lab using VMware.

Software / Hardware Required

- **Software:** VMware Workstation Player/Pro or VMware Fusion
- **ISO/VMs:** Kali Linux (attacker), Metasploitable2 or DVWA (vulnerable target)
- **Hardware:** Host with ~8 GB RAM and ~50 GB disk (more if running multiple VMs)

Theory (expanded — minimal commands, more explanation)

Penetration Testing (Ethical Hacking)

Penetration testing is a controlled, authorized process that simulates real-world attacks to identify and validate security weaknesses before malicious actors can exploit them. It combines technical discovery with risk assessment to prioritize remediation.

Purpose of a Lab

A pentesting lab provides a safe, isolated environment to practice offensive techniques, validate defenses, and train analysts without risking production systems. Virtual machines let you reproduce realistic scenarios (outdated services, weak credentials, misconfigurations) while keeping all activity contained.

Roles of the VMs

- **Kali Linux (Attacker):** collection of offensive tools (Nmap, Metasploit, sqlmap, Nikto, Burp, John/Hashcat) used for reconnaissance, enumeration, exploitation, and post-exploit tasks.
- **Metasploitable2 / DVWA (Victim):** intentionally vulnerable targets with known flaws and insecure web apps for safe practice of exploits and post-exploit techniques.

Network Isolation & Safety

Use **Host-Only** or **NAT** networking so attacker and victim communicate only within the host or virtual network. Never bridge vulnerable VMs to the host LAN or the Internet—this prevents accidental exposure and real harm. Take VM **snapshots** before testing to enable fast rollback to clean states.

Pentest Workflow (concepts & goals)

1. **Reconnaissance:** passive and active discovery to map the target (identify live hosts, open ports, and services). The goal is to build an accurate attack surface model.
2. **Vulnerability Scanning / Enumeration:** probe services for versions, misconfigurations, and known issues to prioritize viable attack paths.
3. **Exploitation:** attempt controlled exploits against confirmed vulnerabilities to gain access (shells, web shells, or service takeover). Follow safe practices—avoid destructive payloads in the lab.
4. **Post-Exploitation:** gather system information, check privileges, extract proof-of-concept artifacts (files, credentials), and evaluate lateral movement or persistence options within the authorized scope.
5. **Reporting:** document findings with evidence, impact, and remediation: target details, open ports/services, exploits used, access obtained, and prioritized mitigation steps.

Procedure (high level — no command snippets)

1. Create two VMs (Kali — attacker; Metasploitable2/DVWA — victim).
2. Configure Host-Only networking so VMs can communicate in isolation.
3. Verify connectivity from Kali to the victim.
4. Follow the pentest workflow: reconnaissance → vulnerability scanning → exploitation → post-exploitation → reporting.
5. Snapshot machines at key stages and restore as needed.

Result

A secure, repeatable penetration-testing lab in VMware with Kali (attacker) and Metasploitable2/DVWA (victim) configured on an isolated network; ready for scanning, exploitation practice, and reporting.

Conclusion (one line)
A properly isolated VMware lab enables safe, controlled practice of the full penetration-testing lifecycle—reconnaissance, scanning, exploitation, post-exploit analysis, and reporting—while preserving host and network safety.

Experiment 6. Implementation of Damn Vulnerable Web Applications (DVWA) using VMware.

Aim
Install and configure **DVWA** on an Ubuntu web server so you can safely practise web vulnerabilities in an isolated lab.

Theory

DVWA is an intentionally vulnerable web application for learning and testing web security flaws (SQL injection, XSS, file upload, etc.). It runs on a **LAMP** stack and requires permissive PHP settings and writable application paths for its "lab-only" insecure modes. Deploy DVWA in an isolated network, initialise its database through the web setup, set security level to vary difficulty, and attack only from an attacker VM on the same isolated network. Never expose DVWA to production or the Internet.

Requirements (brief)

- Ubuntu server with Apache, MariaDB/MySQL, PHP and required PHP modules (mysqli, gd, curl, zip, xml, mbstring).
- `git` and access to `/var/www/html`.
- DVWA repository URL.
- DB user credentials for lab use (example: user `dvwa`, password `p@ssw0rd!`).
- Snapshots enabled for rollback.

High-level Setup Steps (no code)

1. **Obtain DVWA**

- Remove any default index page from the webroot and clone the DVWA repository into `/var/www/html/dvwa`.

2. **Configure DVWA**
   - Copy DVWA's sample config file to create `config.inc.php`.
   - Edit the config to set the database user, password and database name (keep the DB server as `127.0.0.1` or `localhost`). Optionally add Google reCAPTCHA keys later.
3. **Prepare the database**
   - Create a `dvwa` database and a `dvwa` database user with the chosen password, and grant that user appropriate privileges on the `dvwa` database.
4. **Adjust PHP for lab-only mode**
   - Enable permissive PHP flags that DVWA's low security expects: `allow_url_include = On`, `allow_url_fopen = On`, and `file_uploads = On`. Restart Apache to apply changes. (If these directives are missing, add them under the PHP section.)
5. **Fix file permissions DVWA needs**
   - Ensure the DVWA directory is owned by the web server user (www-data) and make the config and upload/external directories writable as required by DVWA.
6. **Initialise DVWA via browser**
   - Open the VM's IP address in a browser at `/dvwa/setup.php` and click **Create / Reset Database**. Then visit `/dvwa/login.php` and log in with the default lab credentials (admin / password). Set DVWA security level (low/medium/high/impossible) from the DVWA Security page.
7. **Optional: Attacker VM**
   - Place a Kali attacker VM on the same NAT or Host-Only network. Verify connectivity from Kali to the DVWA VM and use web-testing tools (sqlmap, gobuster, Nikto, Burp Suite) against `http://<DVWA_IP>/dvwa/`.
8. **Snapshots**
   - Take snapshots at key stages (fresh OS, LAMP installed, DVWA initialised) so you can quickly revert to a clean state.

Result / Conclusion (one line)
DVWA will be ready for safe, repeatable web-security exercises once the repository is in the webroot, `config.inc.php` is set, the database is initialised, permissive PHP lab flags are enabled, and file permissions are corrected — all done within an isolated lab and snapshot-protected.

Experiment 7. Implementation of Database Password Auditing.

Aim
To perform a database password audit and discover weak or common passwords on database servers using DBPwAudit and complementary tools.

Theory
DBPwAudit is a Java-based tool for auditing database passwords on Oracle, MySQL, MS-SQL, and IBM DB2 servers. It simplifies adding new DBMS drivers and helps a pentester discover valid user accounts when systems lack a secure password policy. DBPwAudit currently uses

dictionary-based password attacks: it tests username/password pairs from supplied wordlists against the target database. Tools like Hashcat, John the Ripper, Cain & Abel, and other password-auditing utilities complement DBPwAudit by cracking hashes or performing more advanced attacks. Blind SQL techniques (e.g., via BSQL Hacker or manual injection) and automation tools like sqlmap are used to extract data when direct queries do not return results.

Basic Usage (short)
Audit a MySQL database with username/password wordlists (dictionary-based):

- Shell wrapper example:
  `./dbpwaudit.sh -s <server> -d <database> -D <driver> -U users.txt -P passwords.txt`
  - `-s` = server IP/name
  - `-d` = database name
  - `-D` = driver alias (e.g., MySQL)
  - `-U` = file of usernames
  - `-P` = file of passwords
- JAR example (MySQL):
  `java -jar DBPwAudit.jar -s localhost -d testdb -D MySQL -U users.txt -P /usr/share/wordlists/rockyou.txt`

Setup Notes (brief)

- Ensure Java is available and DBPwAudit runs without errors (`java -jar DBPwAudit.jar`).
- Prepare the database (MySQL/MariaDB) and ensure you have network access to the DB server.
- Create small `users.txt` and `passwords.txt` or reuse standard wordlists (e.g., `rockyou.txt` from `/usr/share/wordlists` on Kali). Unzip compressed wordlists first if needed.

Complementary Tools & Roles (short)

- **Hashcat / John the Ripper** — crack password hashes offline (fast, GPU-accelerated).
- **Cain & Abel** — GUI password recovery and cracking suite (Windows).
- **BSQL Hacker** — automated blind SQL injection exploitation.
- **sqlmap** — automates SQL injection (useful when DB access must be obtained via web app).

Use DBPwAudit for direct DB authentication checks and the others for hash cracking or web-based extraction.

Workflow Summary (minimal steps)

1. Identify target DB server and reachable port.
2. Confirm supported driver with `-L`.
3. Build/choose username and password wordlists.
4. Run DBPwAudit against target (`-s`, `-d`, `-D`, `-U`, `-P`).
5. If hashes are obtained, use Hashcat/John to crack them offline.
6. For web-facing backdoors or blind SQL, use sqlmap or BSQL Hacker to extract credentials.
7. Record valid accounts, access levels, and remediation recommendations.

Common Practical Tips (short)

- Start with small, focused wordlists to avoid noise; escalate to larger lists if authorized.
- Watch for account lockout policies — aggressive guessing can lock accounts or trigger alerts.
- Use timing and throttling options when available to avoid detection or disruption.
- Prefer offline cracking (hashcat, john) once you have hashes to avoid noisy online guessing.

Conclusion (one line)
DBPwAudit is a practical dictionary-based tool for discovering weak database credentials; combined with hash-crackers and SQL automation tools, it forms a comprehensive approach to database password auditing when used ethically and within scope.

Experiment 8. Implement SQL Injection to test the security of database.

Aim
Demonstrate SQL injection exploitation and perform a basic password audit using John the Ripper.

Theory — SQL Injection (concise, retaining original wording)
SQL injection is a code-injection technique that inserts malicious SQL into an application's database queries via web page inputs. An attacker crafts payloads that alter the intended SQL statement so the database executes attacker-controlled logic — e.g., returning all rows or leaking data. It is one of the most common web hacking techniques and can fully compromise a database-backed application if inputs are not properly validated or parameterized.

How it works (short): user input is interpolated into a query such as
SELECT * FROM books_table WHERE book_name = '<input>';
If the attacker supplies payload ' OR 1=1-- - the WHERE clause becomes always true (1=1), and the rest is commented out — returning all records. Tools like **sqlmap** automate discovery and exploitation of SQL injection vulnerabilities.

Execution — SQL Injection (minimal steps, little/no code)

1. **Identify injectable parameter:** find a URL/parameter that accepts input (form field, query string).
2. **Test a basic payload:** try ' OR 1=1-- - in the input field to see if results change (indicates vulnerability).
3. **Automate safely with sqlmap:** run a single targeted scan against the vulnerable URL to enumerate DBMS, databases, tables, and dump chosen data. Use crawling or cookie options only if needed.
4. **Use testing sites:** practice on legally safe targets (e.g., intentionally vulnerable test sites or your lab DVWA). Example test target: http://testphp.vulnweb.com/login.php.
5. **Defence summary:** prevent by using parameterized queries (prepared statements), input validation, least privilege DB accounts, and WAF rules.

Theory — Password Auditing
A password audit tests password strength against dictionary and brute-force attacks to find weak credentials. Tools like **John the Ripper** and **Hashcat** are commonly used: John is a

fast, extensible cracker for many hash types; Hashcat adds GPU acceleration for large-scale cracking. Auditing may operate online (guessing against a live service) or offline (cracking password hashes obtained from /etc/shadow or DB dumps). Ethical audits are performed only on systems you own or are authorized to test.

Execution — Password Auditing with John (minimal steps, little/no code)

1. **Collect hashes (authorized only):** combine /etc/passwd and /etc/shadow into a single file for offline cracking.
2. **Run John:** invoke John with that file; it will try wordlists and rules by default. Optionally supply your own wordlist.
3. **Interpret results:** John reports cracked passwords and which accounts are weak. Use findings to enforce stronger policies (longer/passphrase passwords, hashing algorithms, salt, rate-limiting, MFA).

Tools Mentioned (one-line each)

- **sqlmap:** automates SQL injection detection and exploitation.
- **John the Ripper (JTR):** offline password cracker, good defaults and rule sets.
- **Hashcat:** GPU-accelerated password cracking for large-scale/offline attacks.

Conclusion (one line)

SQL injection manipulates backend queries via unsanitized inputs; password auditing finds weak credentials — together they illustrate why input validation, least privilege, secure hashing, and multi-factor authentication are essential defenses.

Experiment 9. To generate the observation report using NESSUS tool.

Aim
Download, install and run **Nessus** to perform vulnerability scanning and generate scan reports.

Theory
Nessus, from Tenable, is a vulnerability-scanning platform that discovers assets and finds security issues across devices, applications, operating systems, cloud services and network resources. Nessus provides high-speed asset discovery, configuration auditing, target profiling, malware detection and sensitive-data discovery. It identifies missing patches, software flaws, default credentials, misconfigurations and other security weaknesses. Scans produce actionable reports that help prioritize remediation.

Requirements (brief)

- System with sufficient resources (Kali/Ubuntu recommended).
- Nessus package for your OS from Tenable (Linux .deb for Debian/Ubuntu).
- Browser to access the web UI (https://localhost:8834).
- Optional: target hosts (lab VMs such as Kioptrix or Metasploitable) for scanning.

Install & Start (minimal commands / prose)

1. Download the Nessus installer for your platform (select Linux → Debian/amd64).
2. Install the package with the system package installer (e.g., dpkg -i <Nessus>.deb).
3. Start the Nessus daemon/service (systemctl start nessusd).
4. Open the web UI at https://<host>:8834/ (accept the browser security warning).
5. Choose product edition (Nessus Essentials for free use), register with name/email to receive an activation code, paste it into the UI, and create an admin account.
6. Allow Nessus to download its plugin feed; once plugins finish, Nessus is ready.

Basic Usage (short)

- **Create a scan:** In the Nessus UI click **New Scan**, choose a scan template (Basic Network Scan, Host Discovery, etc.), set target IP(s) or range, and start the scan.
- **Run host discovery:** Use a host range to find live hosts before vulnerability scans.
- **Select targets:** Scan lab targets (e.g., Kioptrix/Metsploitable) in an isolated environment.
- **View results:** After completion, open the scan in **My Scans** to review vulnerabilities, risk levels and remediation suggestions.

Generate Scan Reports (concise)

- From the scan results page, use **Report → Generate Report**.
- Choose output format: **PDF, HTML, or CSV**.
- Optionally filter by hosts or vulnerability rows before exporting.
- Nessus allows customizing the report title and logo (Customized Reports available in paid editions).

OpenVAS / Greenbone (short note)

- OpenVAS (Greenbone) is an alternative open-source scanner. A convenient installer script can set up the Greenbone Community Edition (GVM). Pre-reqs include curl, docker, and docker-compose. The community script automates deployment and runs the scanner service.

Conclusion (one line)

Nessus efficiently discovers assets and identifies vulnerabilities; its web UI and reporting features make it practical for security assessments when used on authorized targets and in properly isolated labs.

Experiment 10. To generate the report using Belarc Advisor.

Aim

Study Belarc Advisor: what it is, where it's used, and how to interpret a generated report.

Theory (concise — paraphrased, retaining original wording)

Belarc Advisor builds a detailed profile of a Windows system and presents the results in a web browser. The profile includes installed software and hardware, missing security patches,

antivirus status, and relevant NIST/SCAP security configuration data (examples: USGCB, FDCC). Belarc Advisor is intended for **personal use only** (not for commercial or government use).

The tool inventories assets and highlights security gaps so users can quickly see missing updates, software versions, installed applications, and configuration status. Its local HTML report is easy to read and useful for inventorying, troubleshooting, or preparing a remediation plan.

Key Features (short)

- Full software and hardware inventory.
- Missing patch and update reporting.
- Antivirus and security configuration status.
- Displays results locally in a browser as an HTML report.
- NIST/SCAP configuration checks (USGCB/FDCC examples).

Where It's Used

- Personal PCs for inventory and patch awareness.
- Small-scale audits where a quick local snapshot of system state is needed.
- Education/demonstration environments to show inventory and patch gaps.

How to Install & Run (minimal prose)

1. Download the Belarc Advisor installer from Belarc's website and run the EXE.
2. During install choose options as prompted (you may decline automatic advisory updates).
3. Let the scan complete; Belarc writes a local HTML report.
4. Open the generated HTML in a browser to view the full system profile.

Interpreting the Report (concise)

- **Summary:** quick view of critical items (missing patches, antivirus status).
- **Software Inventory:** lists installed applications and versions — look for outdated or unsupported versions.
- **Missing Patches:** prioritized list of missing hotfixes or security updates.
- **Security Settings / SCAP Checks:** shows compliance with selected benchmarks (if available).
- **Hardware & Network:** basic system and NIC information for asset documentation.

Conclusion (one line)

Belarc Advisor provides a quick, detailed local system profile—software, hardware, patch status and selected SCAP checks—that is useful for personal inventory and remediation planning when combined with network vulnerability scanners for a fuller security picture.