**Assignment 4: Network Diagram Creation Using Scanning Data**

**Scenario:** After performing a network scan for a client, you need to visualize the network topology to help them understand device placements, open ports, and potential attack paths.

**Objectives**

* Use scanning data to create a detailed network diagram.
* Highlight critical devices and vulnerabilities.
* Provide a clear visual representation of the network.

**Instructions**

1. **Gather Scan Data**: Use Nmap results (from Assignment 1 or a new scan) to identify devices, IPs, and open ports. Export the data to a structured format (e.g., XML with -oX).
2. **Choose a Diagram Tool**: Select a tool like draw.io, Lucidchart, or even manual drawing to create the diagram. Ensure it includes devices, connections, and labels for open ports/services.
3. **Map the Network**: Place routers, switches, servers, and workstations based on scan data. Use icons and color-coding to differentiate device types and highlight vulnerabilities.
4. **Add Annotations**: Label critical vulnerabilities (e.g., "Outdated Apache on Port 80") and potential attack paths (e.g., "Direct access to database server").

**Visual Network Topology from Nmap Scan**

**Step 1: Perform Nmap Scan & Export Structured Data**

Use Nmap to scan the network and export the results in XML format for structured parsing:

bash

nmap -sS -sV -O -T4 -p- 192.168.0.0/24 -oX network\_scan.xml

* -sS: SYN scan (stealth).
* -sV: Service version detection.
* -O: OS detection.
* -p-: Scan all 65535 ports.
* -oX: Export to XML format.

**Step 2: Parse the Scan Data**

* **Nmap XML Viewer** (e.g., https://nmap.org/book/output-formats-output-to-xhtml.html)
* **Nmap Parser Tools**:
  + xsltproc network\_scan.xml -o scan\_report.html (HTML view)
  + Tools like nmap-vulners or Nmap Parser for Python

**Extract**:

* Hostnames / IPs
* Open ports (with services)
* OS guess
* MAC address (for device type inference)

**Step 3: Choose a Diagramming Tool**

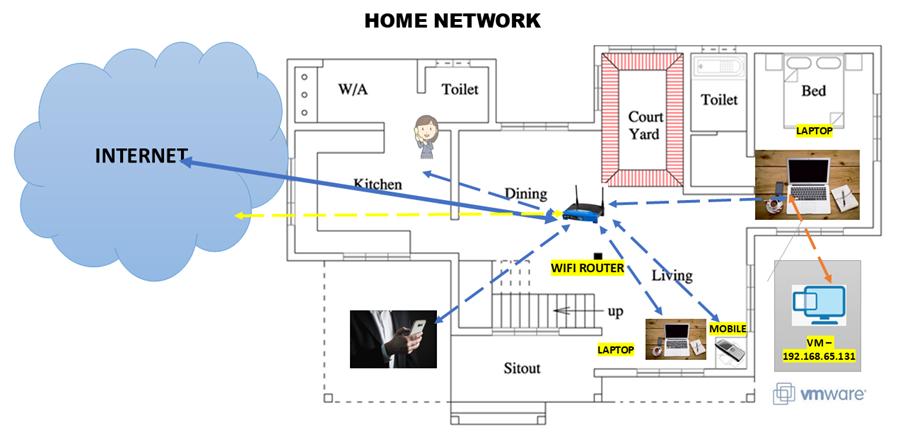
**Use one of the following:**

* **PowerPoint** – for quick visual creation

**Step 4: Design the Network Topology Diagram**

**🖼️ Suggested Visual Structure:**

* **Internet** (cloud icon)  
  ↓
* **Edge Router/Firewall**  
  ↓
* **Switch** (hub icon)**/WIFI**  
  ↙️  ↘️  ↘️
* **Devices** (use color-coded icons):
  + **192.168.0.1** – Router: Ports 80, 7547, 5500 (highlight port 7547 if it's TR-069 enabled – common exploit vector)
  + **192.168.0.170** – Workstation: Connected with several HTTP TIME\_WAIT states
  + **192.168.0.168 / .181** – Workstations/IoT: show MACs, open ports
  + **10.247.161.248 → 8.211.36.117** – Possible outbound traffic (highlight if suspicious)



**🟡 Color Coding Suggestions:**

* 🔵 Blue = Safe service (SSH, HTTPS)
* 🟡 Yellow = Caution (HTTP, FTP)
* 🔴 Red = Known vulnerability or unusual port (e.g., 7547/TR-069)

**Step 5: Annotate the Diagram**

Add labels like:

* **"Port 7547 open – known TR-069 exploit vector"**
* **"Workstation .170 – high HTTP session activity"**
* **"Outbound connection to 8.211.36.117 – needs investigation"**
* **"Router with open port 9000 – check for remote admin exposure"**

Highlight **attack paths**, e.g.:

* Internet → Port 7547 → Router → LAN
* HTTP Exploit → Workstation .170 → Pivot to .168 / .181

**VIRTUAL MACHINE VULNERABILITY REPORT**

| **Port** | **Service** | **Version** | **Banner/Notes** | **CVE/Threat** | **Risk Level** |
| --- | --- | --- | --- | --- | --- |
| 21 | FTP | vsftpd 2.3.4 | Vulnerable version | **CVE-2011-2523** – backdoor | 🔴 High |
| 22 | SSH | OpenSSH 4.7p1 | Outdated | Multiple CVEs (e.g., CVE-2008-4109) | 🟠 Medium |
| 23 | Telnet | Linux telnetd | Unencrypted login | Credential sniffing, brute force | 🔴 High |
| 25 | SMTP | Postfix smtpd | Email relay/open relay risk | CVE-2011-1720 | 🟠 Medium |
| 80 | HTTP | Apache 2.2.8 (Ubuntu) | Outdated, vulnerable to DoS, buffer overflow | **CVE-2007-6388**, CVE-2009-3555 | 🔴 High |
| 111 | RPCBind | v2 | NFS-related RPC service | Used in NFS attacks | 🟠 Medium |
| 139/445 | Samba | smbd 3.x - 4.x | Legacy SMB version | **CVE-2017-7494**, EternalBlue | 🔴 High |
| 512-514 | rsh, login | netkit-rsh, login | Rsh is insecure | Transmits cleartext credentials | 🔴 High |
| 1099 | Java RMI | GNU Classpath | Remote method invocation | RCE via deserialization | 🔴 High |
| 1524 | Bindshell | Metasploitable shell | Backdoor shell | Pre-installed backdoor | 🔴 High |
| 2049 | NFS | v2-4 | NFS share exposure | NFS misconfigurations | 🟠 Medium |
| 2121 | FTP | ProFTPD 1.3.1 | Multiple known vulnerabilities | CVE-2010-4221 | 🔴 High |
| 3306 | MySQL | 5.0.51a | Outdated DB | CVE-2008-2079, weak auth | 🟠 Medium |
| 5432 | PostgreSQL | 8.3.0 | Old version | CVE-2009-3230 | 🟠 Medium |
| 5900 | VNC | Protocol 3.3 | No auth by default | Unauthenticated access | 🔴 High |
| 6667/6697 | IRC | UnrealIRCd | Known trojanized version | **CVE-2010-2075** | 🔴 High |
| 8009 | AJP13 | Apache JServ | May allow ghostcat RCE | **CVE-2020-1938** | 🔴 High |
| 8180 | Tomcat | JSP Engine 1.1 | Likely vulnerable admin console | Weak creds, RCE | 🔴 High |