**Session Hijacking**

**What is Session Hijacking?**

Session hijacking is a cyberattack where an attacker steals a user's session token or cookie to gain unauthorized access to a web application. This allows the attacker to impersonate the legitimate user and perform actions on their behalf.

**Why is Session hijacking successful?**

1. **Absence of account lockout for invalid session IDs:**

This means that an attacker can repeatedly attempt to log in with incorrect session IDs without facing any restrictions. This can lead to brute-force attacks where an attacker can try different combinations of session IDs until they find the correct one.

1. **Indefinite session timeout:**

This means that sessions can remain active for an indefinite period, even after the user has stopped using the system. This can increase the risk of unauthorized access, as an attacker could potentially take over a session that has not been properly timed out.

1. **Weak session-ID generation algorithm or small session IDs:**

This means that the session IDs are easy to guess or predict. This can make it easier for attackers to hijack sessions or impersonate users.

1. **Most computers using TCP/IP are vulnerable:**

This means that most computers connected to the internet are susceptible to attacks that exploit vulnerabilities in session management. This is because TCP/IP is the protocol used for most internet communication.

1. **Insecure handling of session IDs:**

This means that session IDs are not properly protected, which can make them vulnerable to interception or tampering. This can allow attackers to steal session IDs and use them to access the system as the legitimate user.

1. **Most countermeasures do not work without encryption:**

This means that many of the countermeasures used to protect session IDs are ineffective unless they are used in conjunction with encryption. Encryption helps to protect session IDs from being intercepted and tampered with.

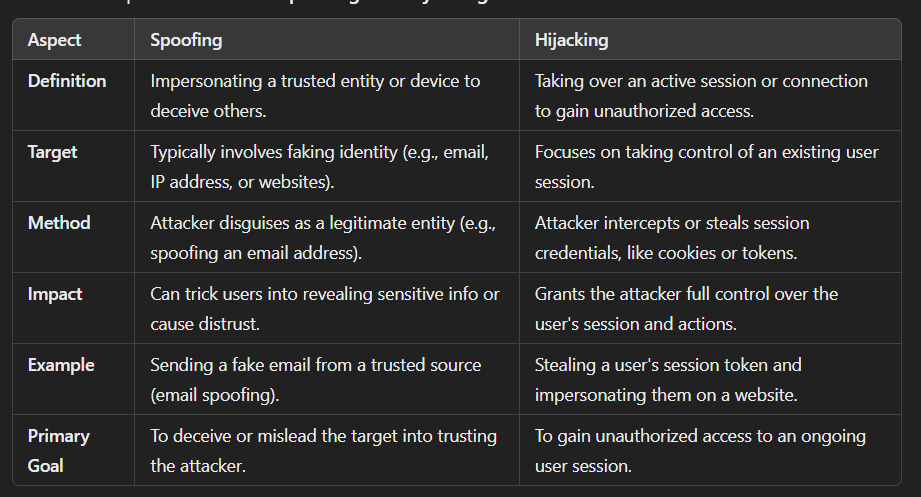
**Types of Session hijacking:**

1. Passive: Here an attacker hijacks a session but sits back, watches, and records all the traffic in that session.
2. Active: Here an attacker finds an active session and seizes control of it.

**Session Hijacking in OSI model:**

1. Network-level hijacking: it’s basically interception of packets during the transmission between a client and the server in a TCP or UDP session.
2. Application-level hijacking: it refers to gaining control over the HTTP’s user session by obtaining the session IDs.

**Spoofing vs Hijacking:**



**How to predict a session token?**

**Session tokens** are used to maintain user sessions on web applications. They are typically stored in **cookies**, **URL parameters**, or **HTTP headers**. If session tokens are predictable or poorly implemented, attackers can hijack or impersonate user sessions. Here's how attackers might attempt to predict or exploit session tokens, and best practices to prevent it:

**Methods of Predicting Session Tokens:**

1. **Weak Randomness / Predictable Generation**: If session tokens are generated using weak algorithms (e.g., simple counters or poor randomization), attackers can guess or predict the next token.
2. **Session Token Reuse**: Reusing the same session token for different users or sessions makes it easy for attackers to hijack valid sessions.
3. **Brute Force Attacks**: If tokens are short or simple (e.g., 6-digit numbers), attackers can try all possible combinations in a brute force attack.
4. **Session Token in URLs**: Tokens passed in URLs can be intercepted or guessed if not properly protected.
5. **Cross-Site Scripting (XSS)**: If a site is vulnerable to XSS, attackers can inject malicious scripts to steal session tokens stored in cookies.
6. **Session Fixation**: In a session fixation attack, an attacker sets the session token before the victim logs in, allowing them to control the session.

**Preventing Session Token Prediction:**

1. **Use Strong Cryptography**: Generate session tokens using secure random number generators (e.g., crypto.randomBytes()), and ensure tokens are long and complex.
2. **Regenerate Tokens After Login**: Always regenerate session IDs after a successful login to prevent session fixation.
3. **Use Secure Cookies**: Store session tokens in cookies with HttpOnly, Secure, and SameSite flags to protect them from being accessed by client-side scripts or stolen over non-HTTPS connections.
4. **Limit Session Lifespan**: Set an expiration time for session tokens to reduce the window of opportunity for attackers.
5. **Implement Multi-Factor Authentication (MFA)**: Adding MFA ensures that even if an attacker guesses the session token, they cannot access the account without the second factor.

**Ethical Reminder:**

Only attempt to predict or test session token vulnerabilities in **authorized environments** (e.g., Capture the Flag challenges, penetration testing with permission). Unauthorized testing is illegal and unethical.

**Session hijacking using proxy servers:**

Session hijacking using proxy servers involves an attacker intercepting and relaying traffic between a client and server through a proxy. By doing so, the attacker can capture session tokens or credentials, impersonate the victim, and gain unauthorized access to the user's session, exploiting weak session management or insecure connections.

**Network-level session hijacking:**

**Network-level session hijacking** occurs when an attacker intercepts and takes control of a user's session by exploiting insecure network connections. Using techniques like **packet sniffing** on unencrypted traffic (e.g., HTTP), the attacker captures session tokens or credentials.

In **Man-in-the-Middle (MitM)** attacks, the attacker positions themselves between the client and server to intercept or modify data. They can also use **session fixation** to set a known session ID before login, allowing them to hijack the session. To prevent such attacks, use **HTTPS**, implement **session expiration**, secure Wi-Fi, and enable **secure cookie flags** (HttpOnly, Secure).

It includes:

1. Blind Hijacking
2. RST hijacking
3. UDP hijacking
4. TCP/IP hijacking
5. IP spoofing

**TCP/IP hijacking:**

**TCP/IP hijacking** occurs when an attacker intercepts and takes control of an active TCP session between two devices. By predicting or stealing sequence numbers, the attacker can inject malicious packets into the communication, gaining unauthorized access to the session.

**IP spoofing:**

**IP spoofing** is a technique where an attacker falsifies the source IP address of a packet to appear as though it’s coming from a trusted or legitimate source. This is often used in attacks like **Denial of Service (DoS)** or **Man-in-the-Middle** to bypass security measures and gain unauthorized access.

**RST hijacking:**

**RST hijacking** involves an attacker sending a **TCP Reset (RST)** packet to one or both parties in an active connection. By spoofing the sequence numbers, the attacker disrupts the connection, causing the session to terminate. This technique can be used to intercept, block, or manipulate communication between two devices.

**Blind Hijacking:**

* An attacker can inject malicious data or commands into the intercepted communications in the TCP session even if the source-routing is disabled.
* The attacker can send the data or commands but has no access to see the response.

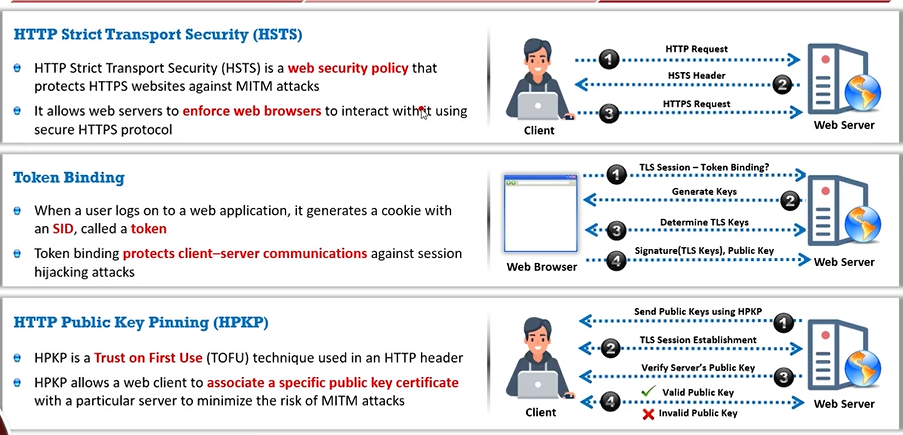
**UDP Hijacking:**

* A network-level session hijacking where the attacker sends a forged server reply to a victim's UDP request before the intended server replies to it.
* The attacker uses a man-in-the-middle attack to intercept the server's response to the client and sends a forged reply.

**Session hijacking tools:**

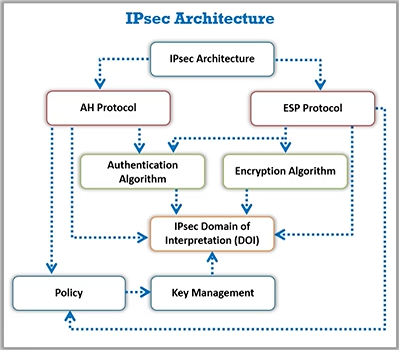
1. OWASP ZAP
2. Burp suite
3. Netool toolkit
4. Sslstrip
5. Wireshark

**Approaches to prevent session hijacking:**

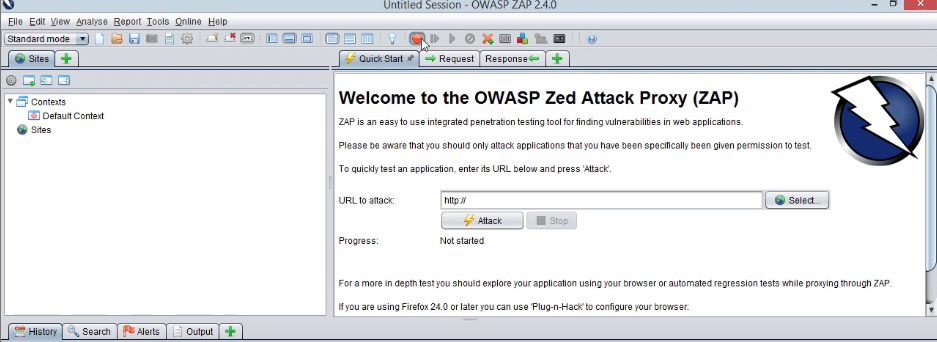


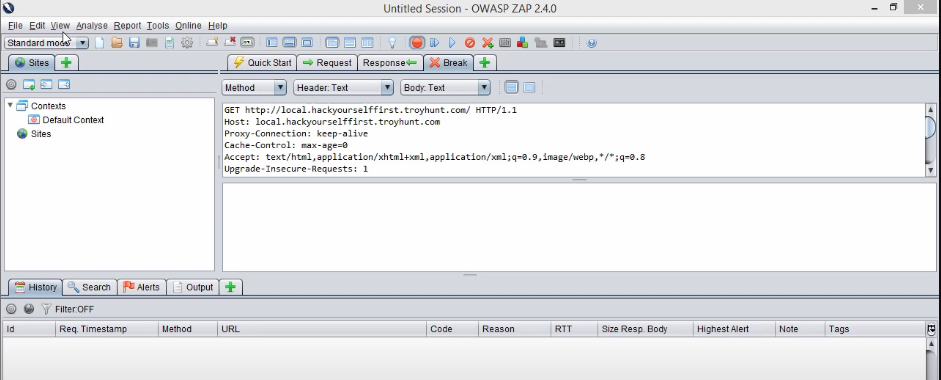
**IPsec:**

IPsec is a protocol suite developed by the IETF for securing IP communications by authenticating and encrypting each IP packet of a communication session.

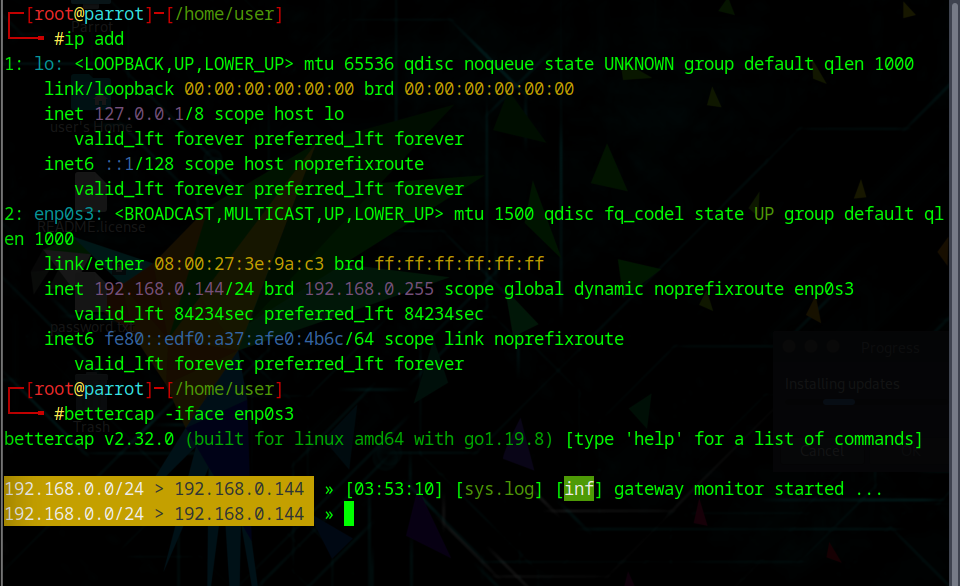


**Hijacking a session using ZAP:**

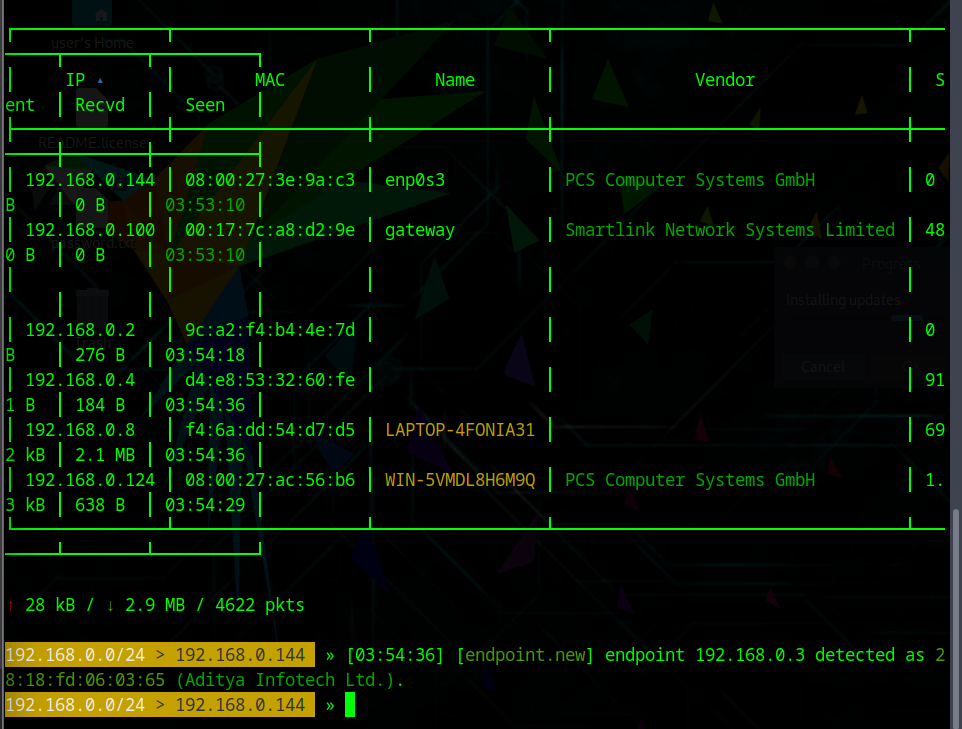


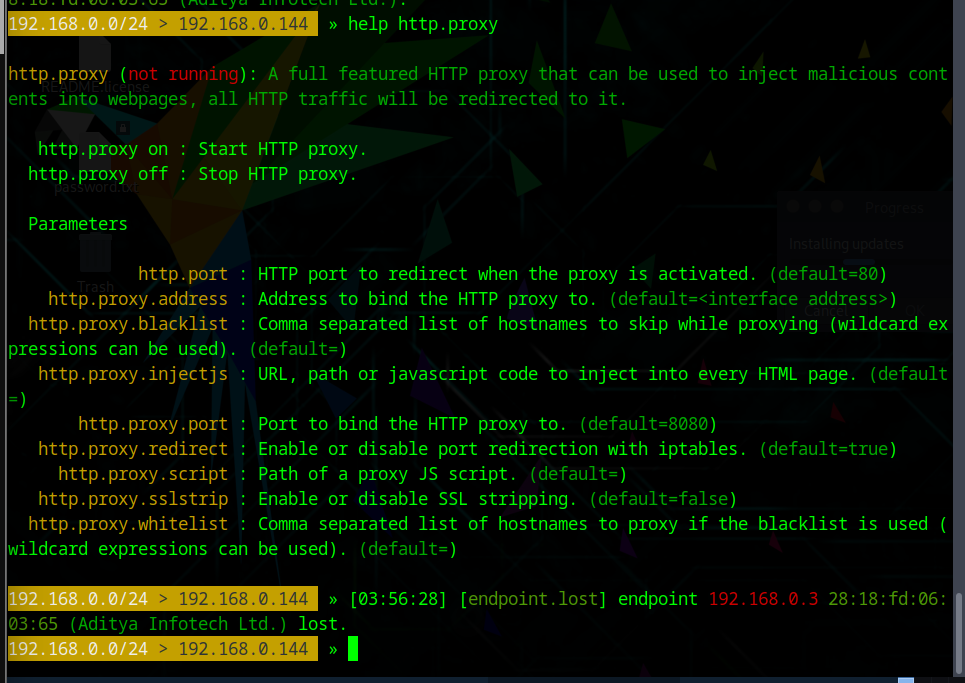


**Intercept HTTP traffic using bettercap:**

****

****

****

****

**Detect session hijacking using wireshark:**

