**Session Hijacking**

Session hijacking is an attack in which an attacker takes over a valid Transmission Control Protocol (TCP) communication session between two computers. Because most types of authentications are performed only at the start of a TCP session, an attacker can gain access to a machine while a session is in progress. A session hijacking attack exploits a session-token generation mechanism or token security controls so that the attacker can establish an unauthorized connection with a target server. The attacker can guess or steal a valid session ID, which identifies authenticated users, and use it to establish a session with the server.

Attackers can use session hijacking to launch various kinds of attacks, such as man-in-the-middle (MITM) and denial-of-service (DoS) attacks. In an MITM attack, an attacker places themselves between an authorized client and a server by performing session hijacking to ensure that information flowing in either direction passes through them.

**Session Hijacking Process**

* **Tracking the connection:** The attacker uses a network sniffer to track a victim and host or uses a tool such as Nmap to scan the network for a target with a TCP sequence that is easy to predict. After identifying a victim, the attacker captures the sequence and acknowledgment numbers of the victim because TCP checks these numbers. The attacker then uses these numbers to construct packets.
* **Desynchronizing the connection:** A desynchronized state occurs when a connection between a target and host is established, or stable with no data transmission or the server’s sequence number is not equal to the client’s acknowledgment number, or vice versa. To desynchronize the connection between the target and host, the attacker must change the sequence number or acknowledgment number (SEQ/ACK) of the server. For this purpose, the attacker sends null data to the server; consequently, the server’s SEQ/ACK numbers advance, while the target machine does not register the increment.

Another approach is to send a reset flag to the server to break the connection on the server side. Ideally, this occurs in the early setup stage of the connection. The attacker’s goal is to break the connection on the server side and create a new connection with a different sequence number. The attacker waits for a SYN/ACK packet from the server to the host. On detecting a packet, the attacker immediately sends an RST packet and a SYN packet with identical parameters, such as a port number with a different sequence number, to the server. The server, on receiving the RST packet, closes the connection with the target and initiates another one based on the SYN packet but with a different sequence number on the same port. After opening a new connection, the server sends a SYN/ACK packet to the target for acknowledgement. The attacker detects (but does not intercept) this packet and sends an ACK packet to the server. Now, the server is in the established state.

* **Injecting the attacker's packet:** Once the attacker has interrupted the connection between the server and target, they can either inject data into the network or actively participate as the man in the middle, passing data from the target to the server and vice-versa while reading and injecting data at will.

**Packet Analysis of a Local Session Hijack**

Session hijacking involves high-level attack vectors, which affect many systems. TCP is used for transmitting data by many systems that establish LAN or Internet connections. For establishing a connection between two systems and for the successful transmission of data, the two systems should perform a three-way handshake. Session hijacking involves the exploitation of this three-way handshake method to take control over the session.

There are two possibilities to determine sequence numbers: one is to sniff the traffic, find an ACK packet, and then determine the NSN based on the ACK packet. The other is to transmit data with guessed sequence numbers, which is not a reliable method. If the attacker can access the network and sniff the TCP session, they can easily determine the sequence number. This type of session hijacking is called "local session hijacking.”**Local Session Hijacking:** The attacker sent the data with the expected sequence number before the user could, the server would be synchronized with the attacker. This leads to the establishment of a connection between the attacker and server. Then, the server would drop the data sent by the user with the correct sequence number, believing it to be a resent packet. The user is unaware of the attacker’s action and may resend the data packet because the user does not receive an ACK for their TCP packet. However, the server would drop all the packets resent by the user.

**Types of Session Hijacking**

* **Passive Session Hijacking:** In a passive attack, after hijacking a session, an attacker only observes and records all the traffic during the session. A passive attack uses sniffers on the network, allowing attackers to obtain information such as user IDs and passwords. The attacker can later use this information to log in as a valid user and enjoy the user’s privileges. Password sniffing is the simplest attack to obtain raw access to a network. Countering this attack involves methods that range from identification schemes (for example, one-time password systems such as S/KEY) to ticketing identification (for example, Kerberos).
* **Active Session Hijacking:** In an active attack, an attacker takes over an existing session either by breaking the connection on one side of the conversation or by actively participating. An example of an active attack is a man-in-the-middle (MITM) attack. To perform a successful MITM attack, attacker must guess the sequence number before the target responds to server.

**Why is Session Hijacking Successful?**

* **Absence of account lockout for invalid session IDs:** The attacker can continue making attempts until the actual session ID is determined. This attack is also known as a brute-force attack. During a brute-force attack, the web server does not display a warning message or complaint, allowing the attacker to determine the valid session ID.
* **Weak Session-ID Generation Algorithm or Small Session IDs:** By studying the sequential pattern and generating multiple requests, an attacker can easily narrow the search space necessary to forge a valid session ID. Even if a strong session-ID generation algorithm is used, an active session ID can be easily determined if the string is short.
* **Insecure Handling of Session IDs:** An attacker can retrieve stored session-ID information by misleading the user’s browser into visiting another site. Before the session expires, the attacker can exploit the information in many ways, such as Domain Name System (DNS) poisoning, cross-site scripting exploitation, and the exploitation of a bug in the browser.
* **Indefinite Session timeout:** Session IDs with an indefinite expiration time provides anattacker with unlimited time to guess a valid session ID. An example of this is the “remember me” option in many websites.
* **Most computers using TCP/Internet Protocol (IP) are vulnerable:** Design flaws
* **Most countermeasures do not work without encryption:** It is easy to sniff session IDs in a flat network if transport security is not set up properly during the transmission of session ID cookies, even if a web application uses Secure Sockets Layer (SSL) encryption.

**🡺 Session Hijacking in OSI Model**

**🡪 Network level hijacking** is the interception of packets during the transmission between a client and server in a TCP/User Datagram Protocol (UDP) session. A successful attack provides the attacker with crucial information, which can be further used to attack application-level sessions. This attack focuses on the data flow of the protocol shared across all web applications.

**Three-Way Handshake**

A three-way handshake starts the connection and exchanges all the parameters needed for the two parties to communicate. TCP uses a three-way handshake to establish a new connection. The Three-Way Handshake Process:

* Client initiates a connection with server and sends a packet to the server with the SYN flag set.
* The server receives this packet and sends a packet with the SYN + ACK flag and an initial sequence number (ISN) for the server.
* Client sets the ACK flag acknowledging the receipt of the packet and increments the sequence number by 1.
* The two machines have successfully established a session.

If the RST flag of a packet is set, the receiving host enters the CLOSED state and frees all resources associated with this connection.

If the packet is sent with the FIN flag turned on, the receiving host closes the connection because it enters the CLOSE-WAIT state.

For the three parties to communicate, the following information is required:

* IP address
* Port numbers
* Sequence numbers

It is easy for an attacker to determine the IP address and port number; these are available in the IP packets, which do not change throughout the session. However, the sequence numbers change.

**TCP/IP Hijacking**

In TCP/IP hijacking, an attacker intercepts an established connection between two communicating parties by using spoofed packets and then pretends to be one of those parties. In this approach, the attacker uses spoofed packets to redirect the TCP traffic to their own machine. Once this is successful, the victim's connection hangs, and the attacker is able to communicate with the host’s machine on behalf of the victim. To launch a TCP/IP hijacking attack, both the victim and attacker must be on the same network. The target server and the victim machines can be located anywhere. By using this technique, an attacker can easily attack systems that use one-time passwords.

**IP Spoofing: Source Routed Packets**

Source routed packets are useful in gaining unauthorized access to a computer with the help of a trusted host’s IP address. First, an attacker spoofs a trusted host’s IP address so that the server managing a session with the host accepts the packets from the attacker. The packets are source routed; therefore, sender specifies the path for packets from the source to the destination IP. By using source-routing technique, attackers fool the server into believing that it is communicating with user.

After spoofing IP address successfully, hijacker alters the sequence and acknowledgment numbers. Once these numbers are changed, the attacker injects forged packets into the TCP session before the client can respond. This leads to a desynchronized state because there the sequence and ACK numbers are not synchronized. The original packets are lost, and server receives a packet with new ISN.

**RST Hijacking**

RST hijacking involves injecting an authentic-looking reset (RST) packet by using a spoofed source IP address and predicting the acknowledgment number. The hacker can reset the victim’s connection if it uses an accurate acknowledgment number. The victim believes that the source has sent the reset packet and resets the connection.

**Blind Hijacking**In blind hijacking, an attacker can inject malicious data or commands into intercepted communications in a TCP session, even if the victim disables source routing. For this purpose, the attacker must correctly guess the next ISN of a computer attempting to establish a connection. Although the attacker can send malicious data or a command, such as a password setting to allow access from another location on the network, the attacker cannot view the response. To be able to view the response, an MITM attack is a much better option.

**UDP Hijacking**The User Datagram Protocol (UDP) does not use packet sequencing or synchronizing. Because UDP is connectionless, it is easy to modify data without the victim noticing. In network level session hijack, the hijacker forges a server reply to a client UDP request before the server can respond. Thus, the attacker takes control of the session. No packets are exchanged between the server and client, because the server’s sequence number fails to match the client’s acknowledgement number. The server’s reply can be easily restricted if sniffing is used. An MITM attack in UDP hijacking can minimize the task of the attacker because it can stop the server’s reply from reaching the client in the first place.

**MITM Attack using Forged Internet Control Message Protocol (ICMP)**

The Internet Control Message Protocol (ICMP) is an extension of IP used to send error messages. An attacker can use ICMP to send messages to fool the client and server. In this technique, ICMP packets are forged to redirect traffic between the client and host through the hijacker’s host. The hacker’s packets send error messages indicating problems in processing packets through the original connection. This fools the server and client into routing through the hijacker’s path instead.

**MITM Attack using Address Resolution Protocol (ARP) Spoofing**

Hosts use Address Resolution Protocol (ARP) tables to map local network layer addresses (IP addresses) to hardware addresses or MAC addresses. This technique involves fooling the host by broadcasting the ARP request and changing its ARP tables by sending forged ARP replies. The attacker sends forged ARP replies that update the ARP tables of the host that is broadcasting ARP requests. This routes the traffic to the attacker’s host instead of the legitimate IP address.

**🡪 Application-level hijacking**

It involves gaining control over the Hypertext Transfer Protocol (HTTP) user session by obtaining the session IDs. At the application level, the attacker gains control of an existing session and can create new unauthorized sessions by using stolen data. An attacker steals or predicts a valid session to gain unauthorized access to a web server.

A session ID brute-forcing attack is known as a session prediction attack if the predicted range of values for a session ID is very small.

A session ID is tagged as proof of an authenticated session established between a user and web server.

**Compromising Session IDs Using Sniffing**

An attacker uses packet sniffing tools such as Wireshark and SteelCentral Packet Analyzer to intercept the HTTP traffic between a victim and web server. The attacker then analyzes the data in the captured packets to identify valuable information such as session IDs and passwords. Once the session ID is determined, the attacker masquerades as the victim and sends the session ID to the web server before the victim does. Attacker uses the valid token session to gain unauthorized access to the web server.

**Compromising Session IDs by Predicting Token**

Session prediction enables an attacker to bypass the authentication schema of an application. An attacker collects a high number of simultaneous session IDs to gather samples in the same time window and keep the variable constant. First, the attacker collects some valid session IDs that are useful in identifying authenticated users. The attacker then studies the session ID structure, the information used to generate it, and the algorithm used by the web application to secure it. From these findings, the attacker can predict the session ID.

Attackers can also guess session IDs by using a brute-force technique, in which they generate and test different session ID values until they succeed in gaining access to the application.

attackers can identify session IDs generated in the following ways:

* Embedding in the URL, which is received by a GET request in the application when the links embedded within a page are clicked by clients.
* Embedding in a form as a hidden field, which is submitted to the HTTP’s POST command.
* Embedding in cookies on the client’s local machine

**Compromising Session IDs Using Man-in-the-Middle Attack**

This attack is used to intrude into an existing connection between systems and to intercept messages being transmitted. In this attack, attackers use different techniques and split a TCP connection into two: a client-to-attacker connection and an attacker-to-server connection.

**Compromising Session IDs using Man-in-the-browser Attack**

A man-in-the-browser attack is similar to an MITM attack. The difference between the two is that a man-in-the-browser attack uses a Trojan horse to intercept and manipulate calls between a browser and its security mechanisms or libraries. An attacker positions a previously installed Trojan between the browser and its security mechanism, and Trojan can modify web pages and transaction content or insert additional transactions. All Trojan’s activities are invisible to both the user and web application.

A man-in-the-browser attack can succeed even in the presence of security mechanisms such as SSL, public key infrastructure (PKI), and two-factor authentication.

**Compromising Session IDs using Client-side Attack**

Client-side attacks target vulnerabilities in client applications that interact with a malicious server or process malicious data. Client-side attacks occur when clients establish connections with malicious servers and process potentially harmful data from them. If no interaction occurs between the client and server, then there is no scope for a client-side attack.

The following client-side attacks can be used to compromise session IDs.

* Malicious JavaScript codes: An attacker can embed in a web page a malicious script that does not generate any warning but captures session tokens in the background and sends them to the attacker.
* Trojans: A Trojan horse can change the proxy settings in the user’s browser to send all sessions through an attacker’s machine.

**Compromising Session IDs Using Client-side Attacks: Cross-site Script Attack**

A cross-site script attack is a client-side attack in which the attacker compromises a session token by using malicious code or programs.

Web sites that create dynamic pages do not have control over how the clients read their output. Thus, attackers can insert a malicious JavaScript, VBScript, ActiveX, Hypertext Markup Language (HTML), or Flash applet into a vulnerable dynamic page. That page then executes the script on the user’s machine and collects personal information of the user, steals cookies, redirects users to unexpected web pages, or executes any malicious code on the user’s systemJavaScript code that fetches the user’s session ID: <SCRIPT>alert(document.cookie);</SCRIPT>

**Compromising Session IDs Using Client-side Attacks: Cross-site Request Forgery Attack**

Cross-site request forgery (CSRF), also known as a one-click attack or session riding, is an attack in which the attacker exploits the victim’s active session with a trusted site to perform malicious activities such as item purchases and the modification or retrieval of account information. In CSRF web attacks, the attacker creates a host form, containing malicious information, and sends it to the authorized user. The user fills in the form and sends it to the web server. Because the data originates from a trusted user, the web server accepts the data.

**Compromising Session IDs Using Session Replay Attacks**

In a session replay attack, the attacker captures the authentication token of a user by listening to a conversation between user and server. Once the authentication token is captured, attacker replays the authentication request to the server with the captured authentication token to dodge the server.

**Session Hijacking Using Proxy Servers**

An attacker lures the victim to click on a fake link, which appears legitimate but redirects the user to the attacker’s server. The attacker then forwards the request to the legitimate server on behalf of the victim and serves as a proxy for the entire transaction.

**Compromising Session IDs Using Session Fixation**

This approach ignores the possibility of the attacker issuing a session ID to the user’s browser, forcing it to use the chosen session ID. This type of attack is called a session fixation attack because an attacker fixes the user's session ID in advance, instead of generating it randomly at the time of login. The attacker performs a session fixation attack to hijack a valid user session.

A session fixation attack is a kind of session hijack. However, instead of stealing the session established between a user and web server after the user logs in, a session fixation attack fixes an established session on the user’s browser; thus, the attack is initiated before the user logs in.

An attacker uses various techniques to perform a session fixation attack:

* Session token in the URL argument
* Session token in a hidden form field
* Session ID in a cookie

**Session Hijacking Using CRIME Attack**

Compression Ratio Info-Leak Made Easy (CRIME) is a client-side attack that exploits vulnerabilities in the data-compression feature of protocols such as SSL/Transport Layer Security (TLS), SPDY, and HTTP Secure (HTTPS).

To perform a CRIME attack, an attacker must use social engineering techniques to trick the victim into clicking on a malicious link. When the victim clicks on the malicious link, it either injects malicious code into the victim’s system or redirects the victim to a malicious website. If the victim has already established an HTTPS connection with a secured web application, the attacker sniffs the victim’s HTTPS traffic using techniques such as ARP spoofing Through sniffing, the attacker captures the cookie value from the HTTPS messages and sends multiple HTTPS requests to the web application with that cookie prepended with a few random characters. Subsequently, the attacker monitors the traffic between the victim and web application to obtain the compressed and encrypted value of the cookie.

**Session Hijacking Using Forbidden Attack**

A forbidden attack is a type of MITM attack that can be executed when a cryptographic nonce is reused while establishing an HTTPS session with a server. According to the TLS specification, these arbitrary pieces of data must be used once. This attack exploits the vulnerability that the TLS implementation incorrectly reuses the same nonce when data are encrypted using the Advanced Encryption Standard–Galois/Counter Mode (AES-GCM) during the TLS handshake. Attackers exploit this vulnerability to perform an MITM attack by generating cryptographic keys used for authentication. Repeating the same nonce during the TLS handshake allows an attacker to monitor and hijack the connection. After hijacking the HTTPS session and bypassing the protection, attackers inject malicious code and forged content into the transmission.

**Session Hijacking Using Session Donation Attack**

In a session donation attack, the attacker donates their own session ID to the target user. In this attack, the attacker first obtains a valid session ID by logging into a service and later feeds the same session ID to the target user. This session ID links a target user to the attacker’s account page without disclosing any information to the victim. When the target user clicks on the link and enters the details in a form, the entered details are linked to the attacker’s account. To initiate this attack, attacker can send their session ID using techniques such as cross-site cooking, an MITM attack, and session fixation.

**🡺 Session Hijacking Detection Methods**

The following are some symptoms of a session hijacking attack:

* A burst of network activity for some time, which decreases the system performance
* Busy servers resulting from requests sent by both the client and hijacker

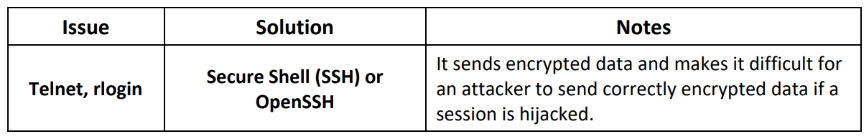
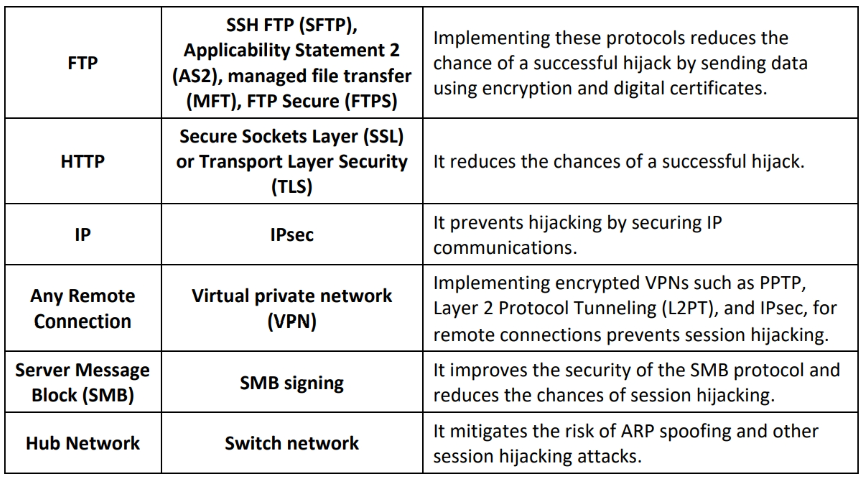
**Manual method** involves the use of packet sniffing software such as Wireshark and SteelCentral Packet Analyzer to monitor session hijacking attacks. The packet sniffer captures packets in transit across the network, which is then analyzed using various filtering tools.

**Forced ARP** entry involves replacing the MAC address of a compromised machine in the ARP cache of the server with a different one in order to restrict network traffic to the compromised machine.

**Automatic method** involves the use of an IDS & IPS to monitor incoming network traffic. If the packet matches any of the attack signatures in the internal database, the IDS generates an alert, whereas the IPS blocks the traffic from entering the database.

Use restrictive cache directives for all the web traffic through HTTP and HTTPS to prevent session hijacking, such as the “Cache-Control: no-cache, no-store” and “Pragma: no-cache” HTTP headers and/or equivalent META tags on all or (at least) sensitive web pages.

To prevent session hijacking use cipher-chaining block (CBC) ciphers incorporating random padding up to 255 bytes, thereby making the extraction of confidential information difficult for an attacker.

**🡺 Approaches Causing Vulnerability to Session Hijacking and their Preventative Solutions**

**Approaches to Prevent Session Hijacking**

* **HTTP Strict Transport Security (HSTS)** is a web security policy that protects HTTPS websites against MITM attacks. The HSTS policy helps web servers force web browsers to interact with them using HTTPS. With the HSTS policy, all insecure HTTP connections are automatically converted into HTTPS connections.
* **Token Binding:** When a user logs into a web application, a cookie with a session ID, called a token, is generated. The user utilizes this random token to send requests to the server and access resources. An attacker can impersonate the user and hijack the connection by capturing and reusing a valid session ID. Token binding protects client–server communication against session hijacking attacks. The client creates a public–private key pair for every connection to a remote server. When a client connects to the server, it generates a signature using a private key and sends this signature along with its public key to the server. The server verifies the signature using the client’s public key.
* **HTTP Public Key Pinning (HPKP)** is a trust on first use (TOFU) technique used in an HTTP header that allows a web client to associate a specific public key certificate with a particular server to minimize the risk of MITM attacks based on fraudulent certificates. In TLS sessions, to verify the authenticity of a server’s public key, the public key is enclosed in an X.509 digital certificate, which is signed by a certification authority (CA). By compromising any CA, attackers can perform MITM attacks on various TLS sessions. HPKP protects TLS sessions from such attacks by delivering to the client the list of public keys owned by a web server.

**Approaches to Prevent MITM Attacks**

* **Wired Equivalent Privacy (WEP) and Wireless Protected Access (WPA)** are wireless protocols that are intended to protect the traffic that is sent and received by users over a wireless network. The implementation of these protocols can thwart the attempts of unwanted users to connect to the network. A weak encryption mechanism enables attackers to brute force credentials and enter the target network to perform an MITM attack.
* **VPN** creates a safe and encrypted tunnel over a public network to securely send and receive sensitive information. It creates a subnet by using key-based encryption for secure communication between endpoints.
* **Two-factor authentication** provides an extra layer of protection because it serves as a vector of authentication in addition to a user’s password.

**🡺 IPSec**

Internet Protocol Security (IPsec) is a set of protocols that the Internet Engineering Task Force (IETF) developed to support the secure exchange of packets at the IP layer. It ensures interoperable cryptographically based security for IPv4 and IPv6, and it supports network level peer authentication, data origin authentication, data integrity, data confidentiality (encryption), and replay protection.

**IPsec Architecture**

IPsec offers security services at the network layer. To provide the requested services, the corresponding cryptographic keys can be employed, if required. Security services offered by IPsec include access control, data origin authentication, connectionless integrity, anti-replay, and confidentiality. To meet these objectives, IPsec uses two traffic security protocols, AH and ESP, as well as cryptographic key management protocols and procedures.

The protocol structure of the IPsec architecture is as follows.

* **Authentication Header (AH):** It is useful in providing connectionless integrity and data origin authentication for IP datagrams and anti-replay protection for the data payload and some portions of the IP header of each packet. However, it does not support data confidentiality (no encryption). A receiver can select the service to protect against replays, which is an optional service on establishing a security association (SA).
* **Encapsulating Security Payload (ESP):** ESP protocol offers confidentiality. Unlike AH, ESP does not provide integrity and authentication for the entire IP packet in transport mode. ESP can be applied alone, in conjunction with AH, or in a nested manner. It protects only the IP data payload in the default setting. In tunnel mode, it protects both the payload and IP header.
* **IPsec Domain of Interpretation (DOI):** It defines the payload formats, types of exchange, and naming conventions for security information such as cryptographic algorithms or security policies. IPsec DOI instantiates ISAKMP for use with IP when IP uses ISAKMP to negotiate security associations.
* **Internet Security Association and Key Management Protocol (ISAKMP):** It is a key protocol in the IPsec architecture that establishes the required security for various communications over the Internet, such as government, private, and commercial communications, by combining the security concepts of authentication, key management, and security associations.
* **Policy:** IPsec policies are useful in providing network security. They define when and how to secure data, as well as security methods to use at different levels in the network.

**Components of IPsec**

* **IPsec driver:** Software that performs protocol-level functions required to encrypt and decrypt packets.
* **Internet Key Exchange (IKE):** A protocol that produces security keys for IPsec and other protocols.
* **Internet Security Association and Key Management Protocol (ISAKMP):** Software that allows two computers to communicate by encrypting the data exchanged between them.
* **Oakley:** A protocol that uses the Diffie–Hellman algorithm to create a master key and a key that is specific to each session in IPsec data transfer.
* **IPsec Policy Agent:** A service included in Windows OS that enforces IPsec policies for all the network communications initiated from that system.

**Modes of IPSec**

The modes are associated with the functions of two core protocols: the Encapsulation Security Payload (ESP) and Authentication Header (AH).

* **Transport mode:** IPsec encrypts only the payload of the IP packet, leaving the header untouched. It authenticates two connected computers and provides the option of encrypting data transfer. It is compatible with network address translation (NAT), therefore, it can be used to provide VPN services for networks utilizing NAT.
* **Tunnel mode:** IPsec encrypts both the payload and header. Hence, in the tunnel mode has higher security than the transport mode. After receiving the data, the IPsec-compliant device performs decryption. The tunnel model is used to create VPNs over the Internet for network-to-network communication (e.g., between routers and link sites), host-to-network communication (e.g., remote user access), and host-to-host communication (e.g., private chat). It is compatible with NAT and supports NAT traversal. In the tunnel mode, the system encrypts entire IP packets (payload and IP header) and encapsulates the encrypted packets into a new IP packet with a new header.