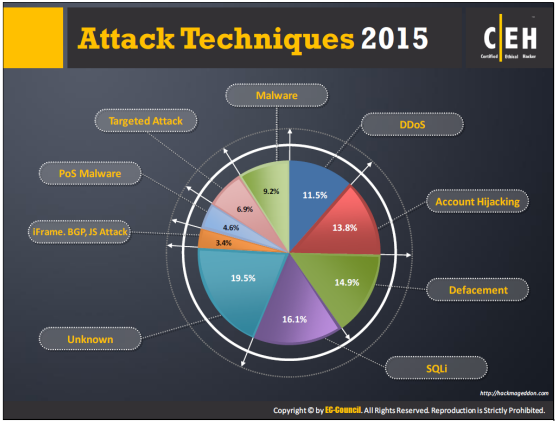
Session Hijacking

Module 10

Ethical Hacking and Countermeasures v9

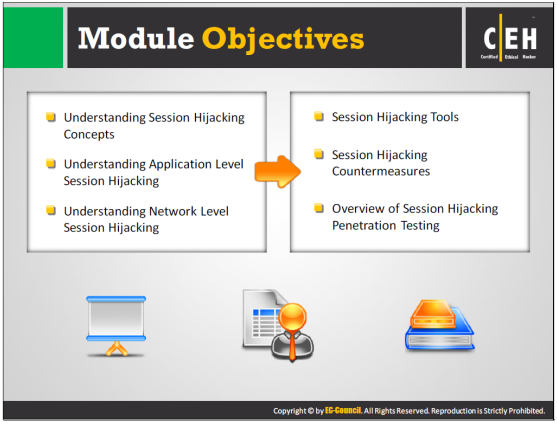
Module 10: Session Hijacking

Exam 312-50



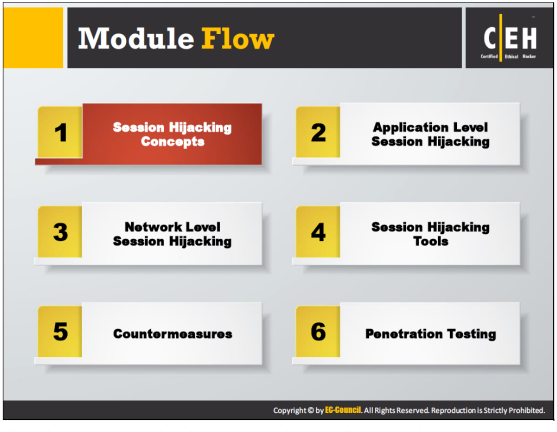
Account hijacking is a type of session hijacking attack wherein the attacker steals a user's email, computer or other type of account information. The attacker might use the stolen information for malicious activities. In March 2015, account hijacking acquired fourth place in the list of most frequently used attack techniques, with an occurrence rate of 13.8%.

Source: *http://hackmageddon.com*

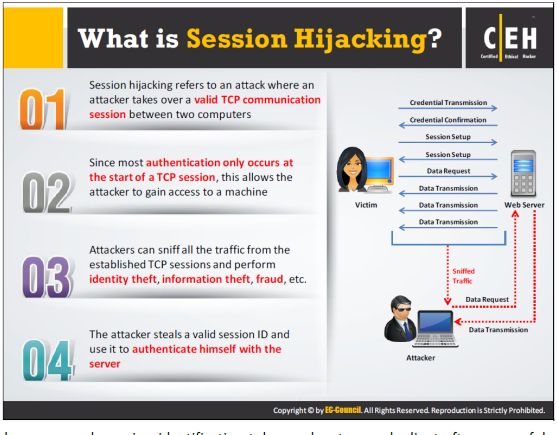


Session hijacking allows attackers to take over an active session by bypassing the authentication process. Thereafter, they can perform any action they desire on that system.

This module aims to provide you with a thorough knowledge on session hijacking. It starts with an introduction to session hijacking concepts, and provides an insight into application and network-level session hijacking. Later, the module discusses session hijacking tools and the countermeasures used to prevent it. It ends with an overview of pen-testing steps an ethical hacker should follow in performing a security assessment of session-hijacking targets.



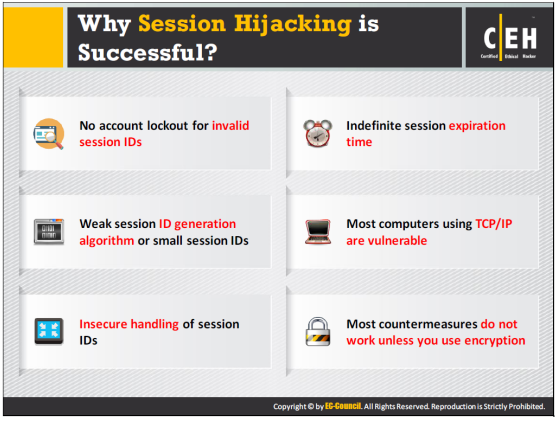
To fully understand session hijacking, one needs to be familiar with its basic concepts. This section answers the questions: What is session hijacking? and Why is it successful? It also discusses session hijacking process, packet analysis of a local session hijack, the types of session hijacking, session hijacking in an OSI model, and spoofing versus hijacking.



A web server sends session identification token or key to a web client after successful client authentication. These session tokens differentiate multiple sessions that the server establishes with various clients. Web servers use various mechanisms to generate random tokens and various controls to secure them during transmission to the clients.

A session hijacking attack refers to the exploitation of 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 uses it to establish a session with server. The web server responds to the attacker's requests as though it were communicating with an authenticated user.

Attackers can use session hijacking to launch various kinds of attacks, such as man-in-the-middle (MITM) and DoS attacks. An MITM attack is one in which the attacker places himself between the client and server. Session hijacking enables attackers to place themselves between the authorized client and the web server, so that all information—traveling in either direction—must pass through them. The client browser believes it is directly communicating with the server, and the server believes it is directly communicating with the authorized client; however, all traffic between them passes through the attacker. Attackers can sniff all the sensitive information from the session and disrupt the sessions to cause a denial-of-service attack.

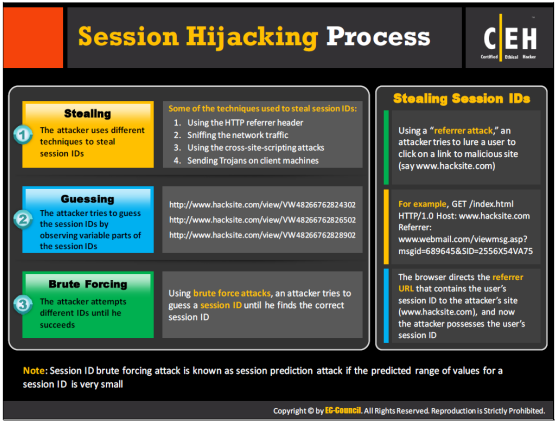


Session hijacking is successful because of the following factors:

- **Weak session-ID generation algorithm or small session IDs:** Most websites use linear algorithms to easily predict variables such as time or IP address for generating session IDs. By studying the sequential pattern of this process and generating many requests, an attacker can easily alleviate the search space necessary to produce a valid session ID. Even though a strong session-ID generation algorithm is used, an active session ID can be easily determined if the length of the string is small.
- **Indefinite session-expiration time:** Session IDs with an indefinite expiration time allows an attacker with unlimited time to guess a valid session ID. An example of this is the "**remember me**" option on many websites. The attacker can use static-session IDs to gain access to the user's web account after capturing the user's cookie file. The attacker can also session-hijack if he/she is able to break into a proxy server, which potentially logs or caches the session IDs.
- **Most countermeasures do not work unless users employ encryption:** It is easy to sniff session ID across a flat network, if the transport security is not set up properly during transmission of session ID cookies from and to the browser, regardless of whether the web application uses SSL. An attacker's job becomes even easier if he/she captures the session IDs that contain the actual logon information in the string
- **Insecure handling of session IDs:** An attacker can retrieve the stored session-ID information by misleading the user's browser into visiting another site. Before the

session expires, an attacker can exploit the information in many ways, such as DNS poisoning, cross-site scripting exploitation, and exploiting a bug in the browser.

- **Most computers using TCP/IP are vulnerable**: All machines running TCP/IP are vulnerable to session hijacking, as the root cause of the attack lies in the design flaws inherent in the TCP/IP protocol.
- **No account lockout for invalid session IDs:** If the website does not implement account lockout, the attacker can make any number of connection attempts with varying session IDs embedded in a genuine URL. An attacker can continue his/her attempts until the actual session ID is determined, otherwise known as "**brute forcing**" the session IDs. During the session-ID brute-force attack, the web server will not display a warning message or complaint, thus allowing the attacker to determine the original session ID.



An attacker implements various techniques such as stealing, guessing, and brute forcing to get a valid session ID, which is useful in taking hold of a valid user's session while that session is still in progress.

Stealing

An attacker can steal the session key by means of physical access; for example, by acquiring the files containing session IDs or memory contents of either the user's system or the server. The attacker can also use sniffing tools such as Wireshark or Cain and Able to sniff the traffic between the client and server and extract the session IDs from the packets.

Guessing

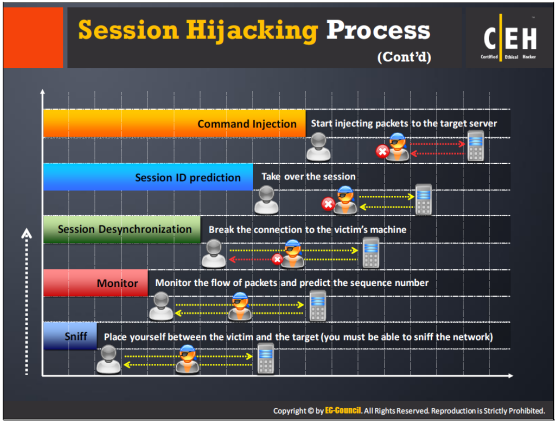
As far as guessing the session ID in an attempt to hijack the session, the possible range of values for the session ID is limited. Thus, guessing techniques are effective only when servers use weak or flawed session-ID generation mechanisms.

Brute forcing

Using the brute-force technique, an attacker can guess session IDs by trying multiple possibilities of patterns until finding one that works. An attacker using a DSL line can generate up to 1,000 session IDs per second. This technique is most useful when the algorithm that produces session IDs is non-random.

FIGURE 10.1: Attacker brute-forcing session ID of a user

In the diagram above, a legitimate user connects to the server with session ID VW30422101522507. Employing various combinations such as VW30422101518909, VW30422101520803, and so on, the attacker tries to brute force the session ID in the hopes of eventually arriving at the correct one. Once the attacker gets the correct session ID, he/she gains complete access to the user's data and can perform operations in place of the legitimate user.



It is easier to sneak in to a system as a genuine user than to attempt to enter a system directly. An attacker can hijack a genuine user's session by finding an established session and taking it over after user authentication. After hijacking the session, the attacker can stay connected for hours without arousing suspicion. All routed traffic destined for the user's IP address comes to the attacker's system. During this time, the attacker can plant backdoors or even gain additional access to the system. How does an attacker go about hijacking a session?

Session hijacking can be broken down into three broad phases:

- **Tracking the connection**

  The attacker uses a network sniffer to track a victim and host or uses a tool like **Nmap** to scan the network for a target with a TCP sequence that is easy to predict. After the identification of victim, an attacker captures the sequence and acknowledgment numbers from the victim. Because TCP checks the packets through sequence and/or acknowledgment numbers, the attacker uses these numbers to construct packets.

- **Desynchronizing the connection**

  A desynchronized state occurs when a connection between the target and host is in the established state, or in a stable state with no data transmission; or, the server's sequence number is not equal to the client's acknowledgment number, or the client's sequence number is not equal to the server's acknowledgment number.

To desynchronize the connection between the target and host, the attacker must change the sequence number or acknowledgment number (SEQ/ACK) of the server. To do this, the attacker sends null data to the server so that the server's SEQ/ACK numbers will advance, while the target machine will not register such an increment. For example, before desynchronization, the attacker monitors the session without any kind of interference, then sends a large amount of null data to the server. These data change the ACK number on the server but do not affect anything else. Thus, the server and target are now desynchronized.

Another approach is to send a reset flag to the server to bring down 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.

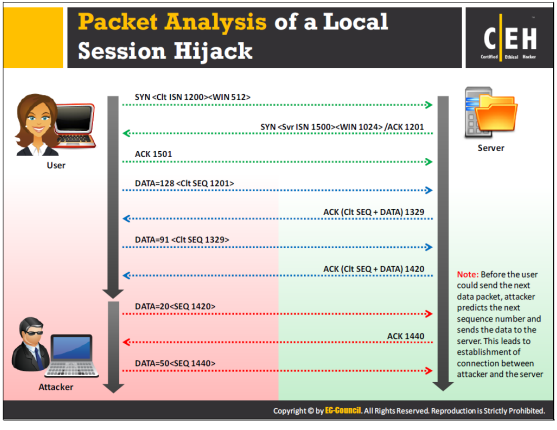
The attacker listens for a SYN/ACK packet from the server to the host. On detecting the packet, the attacker immediately sends an RST packet to the server and a SYN packet with exactly the same parameters, such as a port number, but with a different sequence number. The server, on receiving the RST packet, 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 and sends back an ACK packet to the server. Now the server is in the established state. The main aim is to keep the target conversant, and switch to the established state when it receives the first SYN/ACK packet from the server. Both server and target are now in a desynchronized, but established, state.

An attacker can also use a FIN flag, but this will cause the server to respond with an ACK and give away the attack through an ACK storm. This occurs because of a flaw in this method of hijacking a TCP connection. While receiving an unacceptable packet, the host acknowledges it by sending the expected sequence number. This unacceptable packet generates an acknowledgment packet, thereby creating an endless loop for every data packet. The mismatch in SEQ/ACK numbers results in excess network traffic with both the server and target trying to verify the right sequence. Since these packets do not carry data, retransmission does not occur if the packet is lost. However, since TCP uses IP, the loss of a single packet puts an end to the unwanted conversation between the server and the target.

An attacker can add the desynchronizing stage to the hijack sequence so that the target host is ignorant about the attack. Without desynchronizing, the attacker is able to inject data to the server and even keep his or her identity by spoofing an IP address. However, the attacker should see to it that the server relays response to the target host as well.

- **Injecting the attacker's packet**

  Once the attacker has interrupted the connection between the server and target, he or she can choose to 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, reading and injecting data at will.



Session hijacking attacks are high-level attack vectors which affects many systems. Many systems that establish the connection in a LAN or on the Internet use TCP communication protocol for transmitting data. For connection establishment between two systems and for successful transmission of data, the two systems should establish a three-way handshake. Session hijacking involves exploiting this three-way handshake method to take control over the session.

To conduct a session hijacking attack, the attacker performs three activities:

- Tracks a session
- Desynchronizes the session
- Injects attacker's commands in between

By sniffing the traffic, an attacker can monitor or track the session. The next task in session hijacking is to desynchronize. It is easy to accomplish this attack, if the attacker knows the next sequence number, which a client uses. Once you know the sequence number, you can hijack the session by using the sequence number before the client can use it. There are two possibilities to determine sequence numbers. One way is to sniff the traffic, finding the ACK packet and then determining the next sequence number based on the ACK packet. the other way is to transmit the data with guessed sequence numbers. The second way is not very reliable. If you can access the network and can sniff the TCP session, then you can determine the sequence number easily. This kind of session hijacking refers to "local session hijacking." The diagram that follows shows the packet analysis of a local session hijacking:

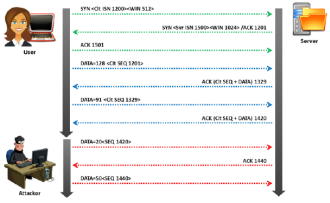


FIGURE 10.2: Screenshot showing the packet analysis of a local session hijack

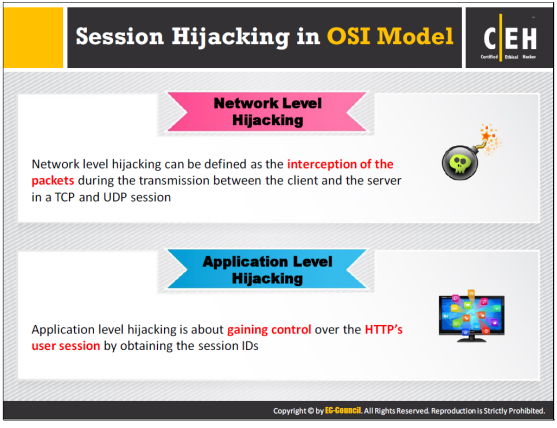
According to the diagram, the next expected sequence number would be 1420. If you can transmit that packet sequence number before the user does, you can desynchronize the connection between the user and the server.

The attacker sends the data with the expected sequence number before the user sends it. Now, the server will be in synchronization with the attacker. This leads to establishment of a connection between the attacker and the server. After establishing the connection between the attacker and the server, though the user sends the data with the correct sequence number, the server drops the data considering it as a resent packet. The user is unaware of the attacker's action and may resend the data packet, as she/he is not receiving an ACK for her/his TCP packet. However, the server drops the packet again. Thus, an attacker performs a local session hijacking attack.

Session hijacking can be either active or passive, depending on the degree of involvement of the attacker. The essential difference between an active and passive hijacking is that while an active attack takes over an existing session, a passive hijack monitors an ongoing 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 on as a valid user and take over privileges. Password sniffing is the simplest attack to obtain raw access to a network. Countering this attack are methods that range from identification schemes (such as a one-time password like S/KEY) to ticketing identification (such as Kerberos). These techniques help in protecting data from sniffing attacks, but they cannot protect against active attacks if there is no encryption, or if it does not carry a digital signature.

In an active attack, the attacker takes over an existing session either by tearing down 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. For this attack to succeed, the attacker must guess the sequence number before the target responds to the server. On most current networks, sequence number prediction does not work because operating-system vendors use random values for the initial sequence number, which makes sequential numbers harder to predict.



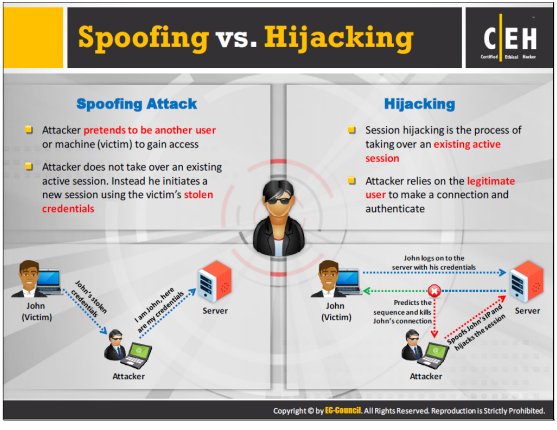
There are two levels to conduct Session hijacking in the OSI model, the network level and the application level.

Network Level Hijacking

Network-level hijacking is the interception of packets during the transmission between client and server in a TCP/UDP session. Successful attack on network level sessions will provide the attacker with crucial information, which will then be used to attack the application level sessions. Attackers most likely perform network-level hijacking as they do not require to modify the attack on a per web application basis. This attack focuses on the data flow of the protocol, shared across all web applications.

Application-Level Hijacking

Application-level hijacking is about gaining control over the HTTP user session by obtaining the session IDs. In the application level, the attacker gets control of an existing session and can try to create new unauthorized sessions using stolen data. In general, both of them occur together, according to the system being attacked.



In 1988, the **Morris worm**, a quickly replicating worm that could hijack sessions, affected nearly 6,000 computers on ARPANET, the predecessor of the global Internet. **Robert T. Morris** exploited the predictable nature of the sequence number that formed the security of a TCP/IP connection. His program spread through the connected computers and performed an action in an infinite loop, copying itself onto every computer within its reach. His program involved both blind spoofing and blind hijacking. In blind hijacking, an attacker predicts the sequence numbers that a victimized host sends in order to create a connection that appears to originate from the host, or a blind spoof.

To understand blind hijacking, it is important to understand sequence number prediction. TCP sequence numbers, unique per byte in a TCP session, provide flow control and data integrity. TCP segments give the initial sequence number (ISN) as a part of each segment header. ISNs do not start at zero for each session; part of the handshake process is for each participant to state the ISN, and it numbers the bytes sequentially from that point.

Remember that blind session hijacking relies on the attacker's ability to predict or guess sequence numbers. An attacker cannot spoof a trusted host on a different network and see the reply packets because there is no route for the packets to go back to his or her IP address. But neither can the attacker resort to ARP cache poisoning, because routers do not route ARP broadcasts across the Internet. As the attacker is unable to see the replies, this forces him or her to anticipate the responses from the victim and prevent the host from sending a TCP/RST packet to the victim. The attacker predicts sequence numbers the remote host is expecting

from the victim, and then hops into the communication. This method is useful to exploit the trust relationships between users and remote machines.

Simple IP spoofing is easy to do and is useful in various attack methods. To create new raw packets, the attacker must have root access on the machine. However, to establish a spoofed connection using this session hijacking technique, an attacker must know the sequence numbers a target machine uses. IP spoofing forces the attacker to forecast the next sequence number. It does not view the response when an attacker uses blind hijacking to send a command.

In the case of IP spoofing not involving a session hijack, guessing the sequence number is not required because there is no session currently open with that IP address. In a session hijack, the traffic would get back to the attacker only if using source routing. Source routing is a process that allows the sender to specify a specific route for an IP packet to take to the destination. The attacker performs source routing and then sniffs the traffic as it passes by the attacker. In session spoofing, captured authentication credentials are useful in establishing a session. Here, active hijacking eclipses a preexisting session. As a result of this attack, the legitimate user may lose access or the normal functionality of her/his established telnet, because an attacker hijacks the session and is now acting with the user's privileges. Because most authentications only happen at the initiation of a session, this allows the attacker to gain access to a target machine.

Another method is to use source-routed IP packets. This man-in-the-middle attack allows an attacker to become a part of the target-host conversation by deceptively guiding the IP packets to pass through his or her system.

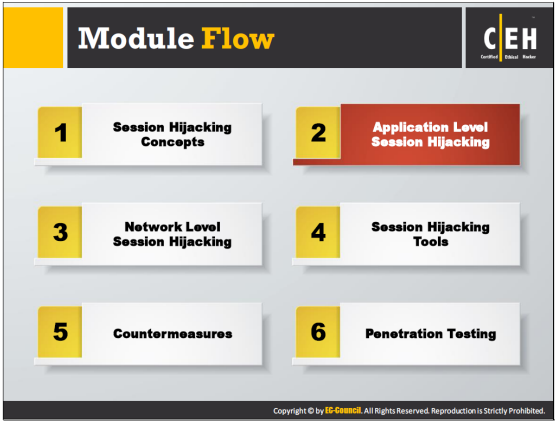
Session hijacking is more difficult than IP address spoofing. In session hijacking, John (an intruder) would seek to insert himself into a session that Jane (a legitimate user) already had set up with \\Mail. John would wait until she establishes a session, then knock her off the air by some means, such as a denial of service, and then pick up the session as though he were she. Then John would send a scripted set of packets to \\Mail and would be able to see the responses. To do this, he would need to know the sequence number in use when he hijacked the session. To calculate the sequence number, he must know the ISN and the number of packets involved in exchanging process.

Successful session hijacking is difficult without the use of known tools and only possible when a number of factors are under the attacker's control. Knowledge of the ISN would be the least of John's challenges. For instance, he would need a way to knock Jane off the air when he wanted to, and need a way to know the exact status of Jane's session at the moment he mounted his attack. Both of these require that John have far more knowledge and control over the session than would normally be possible.

However, IP address spoofing attacks can only be successful if an attacker uses IP addresses for authentication. He or she cannot perform IP address spoofing or session hijacking if there is an execution of checking per-packet integrity in the same way, IP address spoofing and session hijacking is not possible if the session uses encryptions such as SSL or PPTP. Consequently, the attacker cannot participate in the key exchange.

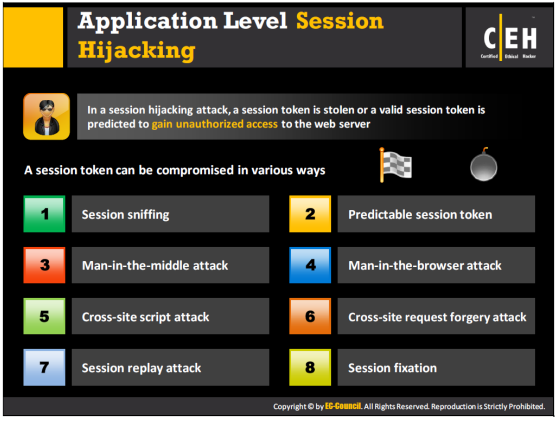
In summary, the hijacking of non-encrypted TCP communications requires the presence of non-encrypted session-oriented traffic, the ability to recognize TCP sequence numbers that predict the next sequence number (NSN), and the ability to spoof a host's MAC or IP address to receive communications that are not destined for the attacker's host. If the attacker is on the local segment, he/she can sniff and predict the ISN + 1 number and route the traffic back to him/her by poisoning the ARP caches on the two legitimate hosts participating in a session.

Source: *http://www.microsoft.com*

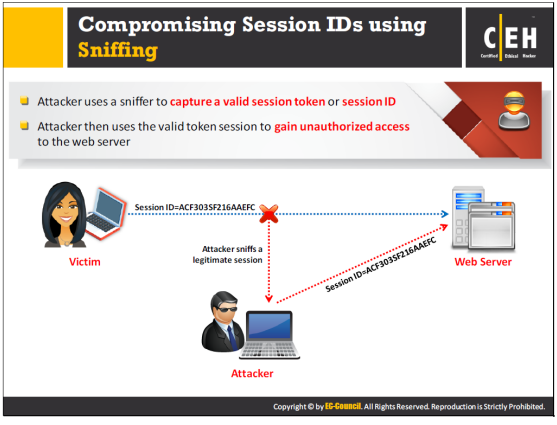


In application-level hijacking, the attacker obtains the session IDs to get control of an existing session or to create a new unauthorized session.

This section deals with application-level session hijacking and various ways to compromise the session token—session sniffing, predictable session token, Man-in-the-Middle attack, Man-in-the-Browser attack, cross-site script attack, cross-site request forgery attack, session replay attack, and session fixation attack.

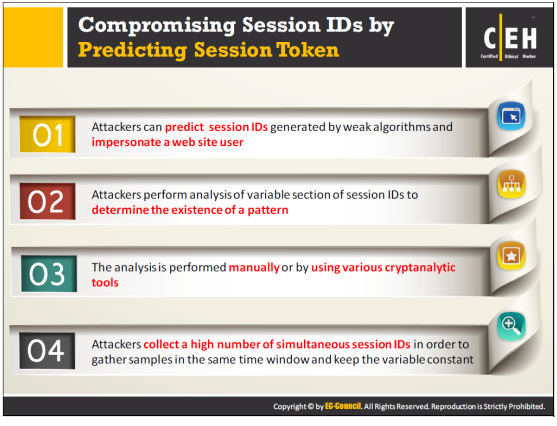


In an application-level session hijacking, an attacker steals or predicts a valid session to gain unauthorized access to the web server. Most probably, network-level and application-level session hijacking occur together, as a successful network-level session hijacking provides an attacker with ample information to perform the application-level session hijacking. Application-level session hijacking relies on HTTP sessions.



Web server identifies the user's connection by means of a unique session ID (also known as session token). The web server sends a session token to a client browser after the successful authentication of client logon. Usually, session token comprises a string of variable width that is useful in various ways, such as in the header of the HTTP requisition (cookie), in the URL, or in the body of the HTTP requisition.

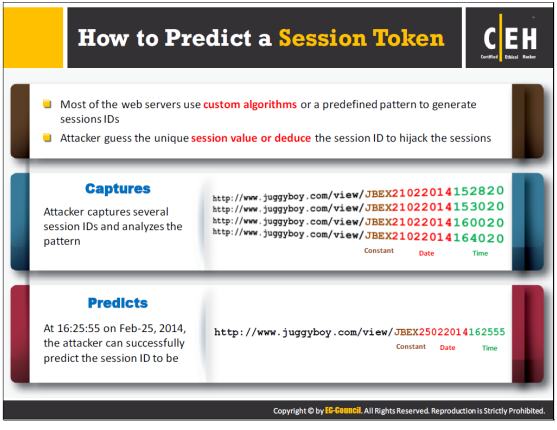
The attacker uses packet sniffing tools such as **Wireshark**, **SmartSniff**, among others to intercept the HTTP traffic between the victim and the web server. He/she then analyzes the data present in those captured packets to identify valuable information such as session IDs, passwords. Once the session ID is determined, the attacker masquerades himself/herself as the victim and sends the session ID to the web server before the victim does it. This way, an attacker takes control over an existing legitimate session.



A session ID is tagged as a proof of the authenticated session established between a user and a Web server. Thus, if an attacker is able to guess or predict the session ID of the user, fraudulent activity is possible. Session prediction enables an attacker to bypass the authentication schema of an application. Usually, an attacker can predict session IDs generated by weak algorithms, by analyzing and understanding the session ID generation process. She/he performs this analysis either manually or by using various cryptanalytic tools.

First, the attacker collects some valid session IDs that are useful in identifying authenticated users. She/he then understands the session ID structure, the information used to generate it, and the algorithm used by the web application to secure it. Using those findings, the attacker can predict the session ID.

An attacker can also guess the Session IDs by using brute force technique, where she/he can generate and test different values of session IDs until she/he succeeds in gaining access to the application.



Most web servers generate Session IDs using custom algorithms or a pre-defined pattern that might simply increment static numbers, whereas others use more complex procedures such as factoring in time and other computer specific variables. One can identify the session thus generated by:

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

The attacker guesses the unique session value or deduces the session ID to hijack the sessions. Here, first an attacker captures several session IDs and analyzes the pattern.

```
http://www.juggyboy.com/view/JBEX21022014152820
http://www.juggyboy.com/view/JBEX21022014153020
http://www.juggyboy.com/view/JBEX21022014160020
http://www.juggyboy.com/view/JBEX21022014164020
                            Constant  Date   Time
```

FIGURE 10.3: Sample sessions captured by the attacker

On analyzing the pattern, at 16:25:55 on Feb-25, 2014, the attacker can successfully predict the session ID to be:

```
http://www.juggyboy.com/view/JBEX25022014162555
                            Constant  Date   Time
```

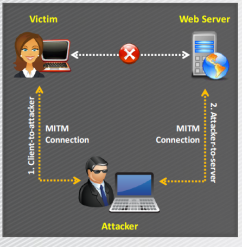
FIGURE 10.4: A session predicted by the attacker

Now, the attacker can mount an attack as follows:

- Acquires the current session ID and connects to the web application
- Implements brute-force technique or calculates the next session ID
- Modifies the current value in the cookie/URL/hidden form-field and assumes the next user's identity

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

The man-in-the-middle attack is used to **intrude into an existing connection** between systems and to intercept messages being exchanged

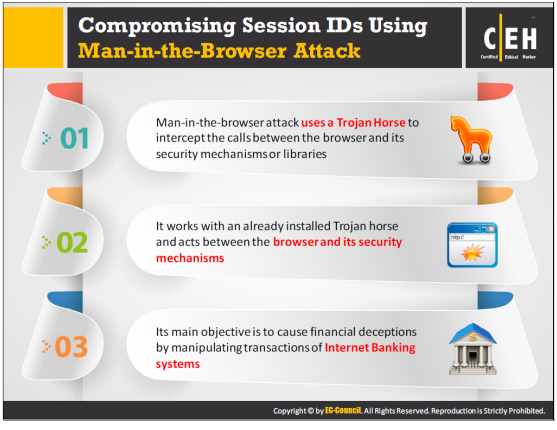


Attackers use different techniques and **split the TCP connection** into two connections

- Client-to-attacker connection
- Attacker-to-server connection

After the successful interception of TCP connection, an attacker can read, modify, and insert fraudulent data into the **intercepted communication**

In the case of an **http transaction**, the TCP connection between the client and the server becomes the target



A man-in-the-browser attack is similar to that of a man-in-the-middle attack. The difference between the two techniques is that the man-in-the-browser attack uses a Trojan horse to intercept and manipulate calls between the browser and its security mechanisms or libraries. An attacker uses previously installed Trojan to act between the browser and its security mechanism, capable of modifying web pages, and modifying transaction content or inserting additional transactions, everything secretly invisible to both the user and web application.

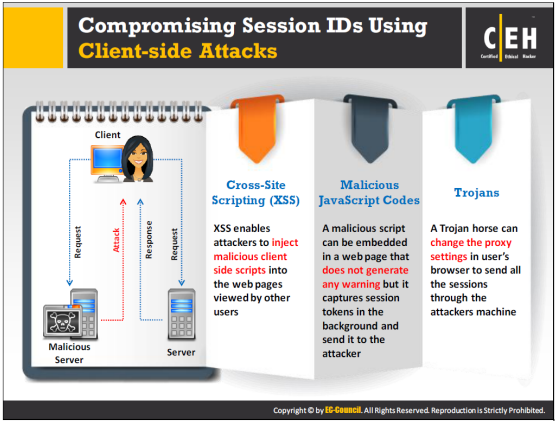
The main objective of this attack is financial theft by manipulating the transactions of Internet banking systems. The man-in-the-browser attack will be successful irrespective of security mechanisms such as SSL, PKI, or two-factor authentication in place, as all the expected controls and security mechanisms would seem to work normally.

Steps to Perform Man-in-the-Browser Attack

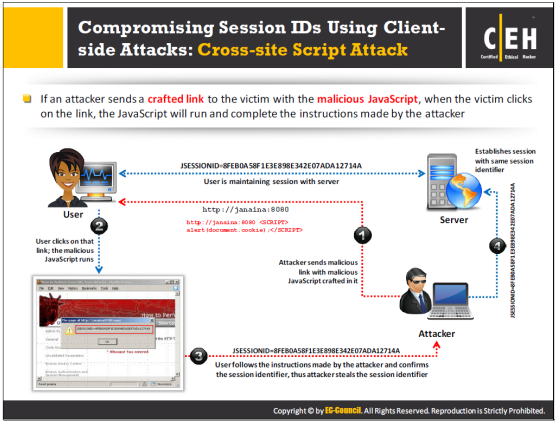
01	The Trojan first infects the computer's software (OS or application)	**02**	The Trojan installs malicious code (extension files) and saves it into the browser configuration
03	After the user restarts the browser, the malicious code in the form of extension files is loaded	**04**	The extension files register a handler for every visit to the webpage
05	When the page is loaded, the extension uses the URL and matches it with a list of known sites targeted for attack	**06**	The user logs in securely to the website
07	It registers a button event handler when a specific page load is detected for a specific pattern and compares it with its targeted list	**08**	When the user clicks on the button, the extension uses DOM interface and extracts all the data from all form fields and modifies the values

Steps to Perform Man-in-the-Browser Attack (Cont'd)

09	The browser sends the form and modified values to the server
10	The server receives the modified values but cannot distinguish between the original and the modified values
11	After the server performs the transaction, a receipt is generated
12	Now, the browser receives the receipt for the modified transaction
13	The browser displays the receipt with the original details
14	The user thinks that the original transaction was received by the server without any interceptions



Client-side attacks target vulnerabilities in client applications that interact with a malicious server or processes malicious data. Depending on the nature of vulnerabilities, an attacker can exploit an application by sending an email with a malicious link or tricks otherwise the user into visiting a malicious web site. Some client-side vulnerable applications include Adobe Acrobat, Java Runtime Environment, and browsers; of these, browsers are the major target. Client-side attacks occur when clients establish connections with malicious servers, as clients happen to process potentially harmful data from them. If no interaction takes place between the client and server, then there is no scope for the client-side attack. One such example is running an FTP client without establishing a connection to an FTP server. In the case of instant messaging, the application is configured in such a way that it leads the clients to log into a remote server, thus making it susceptible to client-side attacks.



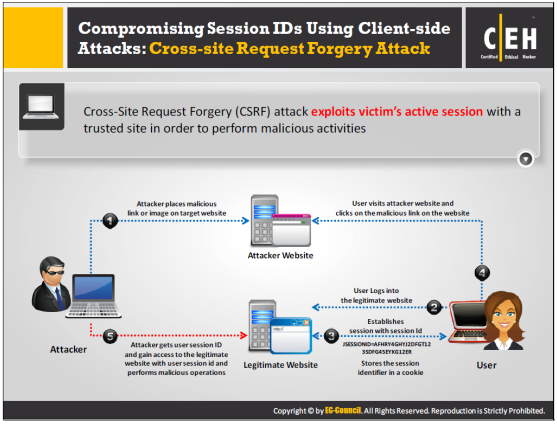
A cross-site script attack is a client-side attack in which the attacker compromises the session token by making use of malicious code or programs. This type of attack occurs when a dynamic Web page gets malicious data from the attacker and executes it on the user's system.

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, HTML, or Flash applet into a vulnerable dynamic page. That page will then execute the script on the user's machine and collect personal information of the user, steal cookies, redirect users to unexpected Web pages, or execute any malicious code on the user's system.

In the diagram show on the slide, the user establishes a valid session with the server. An attacker sends a crafted link to the victim with the malicious JavaScript. When the user clicks on the link, the JavaScript runs automatically and performs the instructions set by the attacker. The result of the attack displays current session ID of the user. Using the same technique, an attacker can create a specific JavaScript code that fetches him/her the user's session ID.

```
<SCRIPT>alert(document.cookie);</SCRIPT>
```

Thereafter, an attacker uses the stolen session ID to establish a valid session with the server.

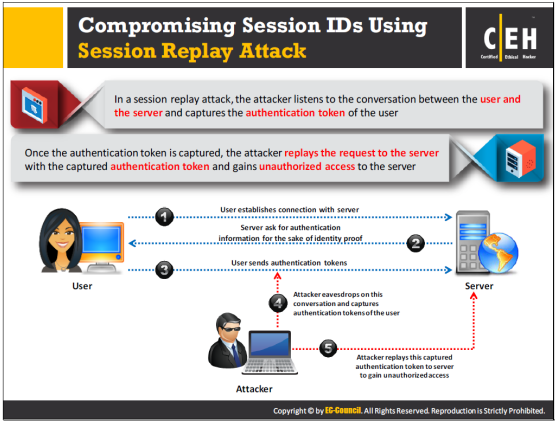


Cross-site Request Forgery (CSRF), also known as a one-click attack or session riding exploits victim's active session with a trusted site to perform malicious activities such as purchase an item, modify, or retrieve account information. In CSRF web attacks, an attacker forces the victim to submit the attacker's form data to the victim's Web server. The attacker creates the host form, containing malicious information, and sends it to the authorized user. The user fills in the form and sends it to the server. Because the data is coming from a trusted user, the Web server accepts the data. Unlike XSS attack, which exploits the trust a user has for a particular website, CSRF exploits the trust that a website has in a user's browser.

Steps involved in a CSRF attack:

- The attacker hosts a Web page with a form that looks legitimate. This page already contains the attacker's request.
- A user, believing the form to be the original, enters a login and password.
- Once the user completes the form, that page gets submitted to the real site.
- The real site's server accepts the form, assuming that it was sent by the user based on the authentication credentials.

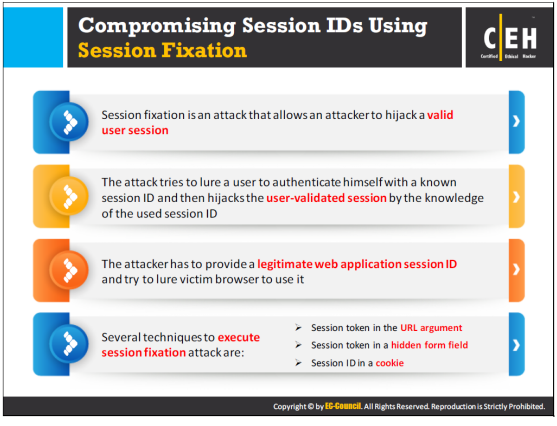
In this way, the server accepts the attacker's request.



In a ***session replay*** attack, an attacker replays an authentication token to fool the server into granting access.

Steps involved in the Session Replay attack:

- User establishes a connection with the web server.
- Server asks the user for authentication information for the sake of identity proof.
- User sends authentication tokens to the server. At this step, an attacker eavesdrops on the conversation between the user and the server and captures the authentication token of the user.
- Once the authentication token is captured, the attacker then replays the request to the server with the captured authentication token and gains unauthorized access to the server.



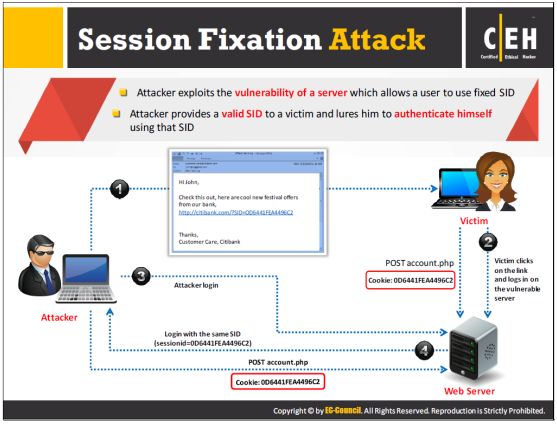
Web session security prevents the attacker from obtaining the session ID by intercepting, brute-forcing, or predicting the session ID issued by the web server to the user's browser as proof of the authenticated session.

However, 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 attack refers to session fixation attack because an attacker fixes the user's session ID in advance, instead of generating it randomly at the login time.

The attacker performs the session fixation attack to hijack a valid user session: the attacker takes the advantage of the limitation present in web application session ID management. The web application allows the user to authenticate him or herself using an existing session ID rather than generating a new session ID. In this attack, the attacker provides a legitimate web application session ID and lures the victim to use it. If the victim's browser uses that session ID, then the attacker can hijack the user-validated session, as the attacker is already aware of the session ID that a victim uses.

The session fixation attack is a kind of session hijacking, which steals the session established between the user and the web server after the user logs in; instead, session fixation attack fixes an established session on the user's browser, thus initiating an attack before the user logs in.

An attacker uses various techniques to perform the session fixation attack. The attacker has to choose which technique to use based on how the web application deals with session tokens.



There are three phases to carry out Session fixation attack:

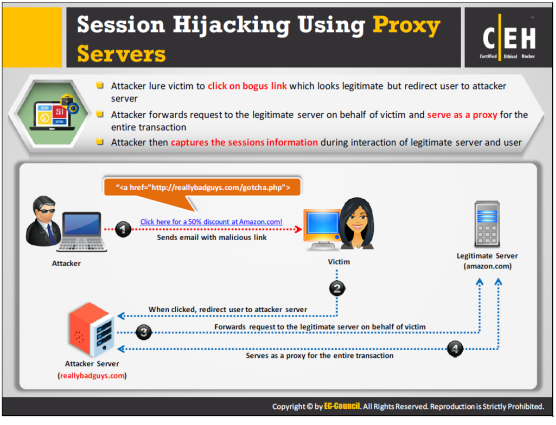
- **Session set-up phase**: In this phase, the attacker first obtains a legitimate session ID by establishing a connection with the target web server. Few web servers support the idle session time-out feature. In such cases, the attacker needs to send requests repeatedly in order to keep the established trap session ID alive.
- **Fixation phase**: In this phase, the attacker introduces the session ID to the victim's browser, thus fixing the session.
- **Entrance phase**: In this phase, the attacker waits for the victim to log in into the target web server using the trap session ID and then enter the victim's session.

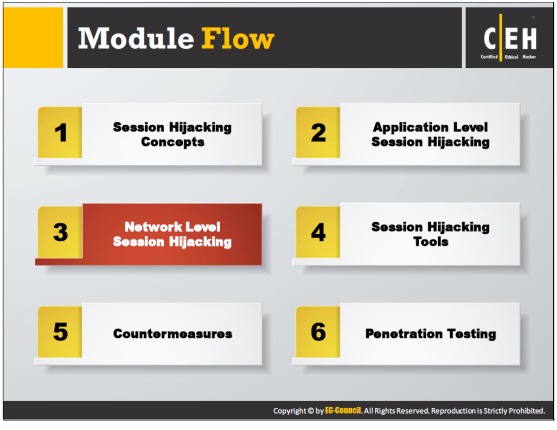
Steps to perform session fixation attack:

- First, the attacker establishes a legitimate connection with the target web server.
- The target web server (http://citibank.com/) issues a session ID, say 0D6441FEA4496C2, to the attacker.
- The attacker then sends a link with the established session ID, say http://citibank.com/?SID=0D6441FEA4496C2, to the victim and lures him/her to click on it to access the website.
- The victim clicks on the link treating it as a legitimate link sent by the bank, this opens the server's login page in the victim's browser for SID=0D6441FEA4496C2.

- The web server checks that the session ID 0D6441FEA4496C2 is already established and is in active state and hence there is no need to create the new session.
- Finally, the victim enters his/her login credentials to the login script, and the server grants him/her access to the bank account.
- At this point, knowing the session ID, the attacker can also access the victim's bank account via http://citibank.com/? SID=0D6441FEA4496C2.

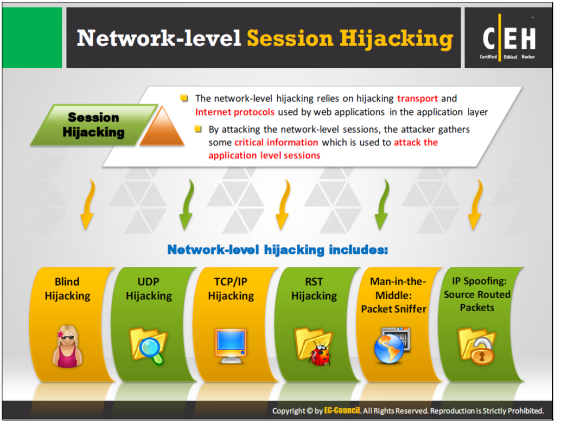
As the session ID is set by the attacker before the user logged in, we can tell that user has logged into the attacker's session.

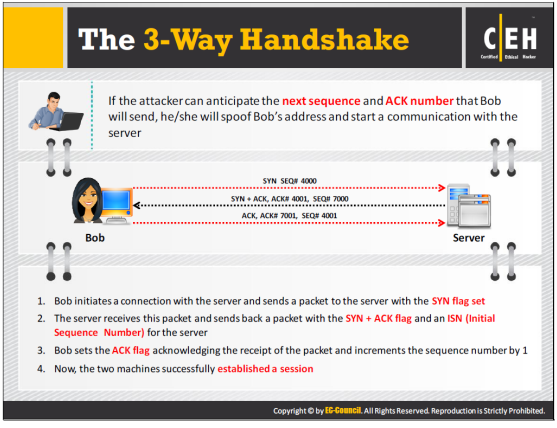




Attackers especially focus on network-level session hijacking, as it does not require host access, as would host-level session hijacking, and they need not tailor their attacks on a per-application basis, as they would at the application level.

This section deals with network-level session hijacking, concepts to be familiar with related to network communications, and various techniques used to perform network-level session hijacking.





When two parties establish a connection using TCP, they perform a 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.

Initially, the connection on the client side is in the closed state, and the one on the server side is in the listening state. The client initiates the connection by sending the Initial Sequence Number (ISN) and setting the SYN flag. The client is now in the SYN-SENT state.

When the server receives this packet, it acknowledges the client sequence number and sends its own ISN with the SYN flag set. The server's state is now SYN-RECEIVED. On receipt of this packet, the client acknowledges the server sequence number by incrementing it and setting the ACK flag. The client is now in the established state. At this point, the two machines have established a session and can begin communication.

On receiving the client's acknowledgement, the server enters the established state and sends back the acknowledgment, incrementing the client's sequence number. Close the connection either using the FIN or RST flag or by timing out.

If the RST flag of a packet is set, the receiving host enters the CLOSED state and frees all resources associated with this instance of the connection. This leads to the connection drop of any additional incoming packets.

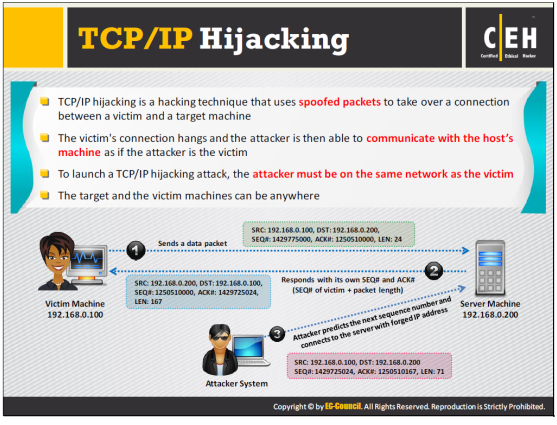
If the packet is sent with the FIN Flag turned on, the receiving host closes the connection as it enters the CLOSE-WAIT mode. The packets sent by the client are accepted in an established connection if the sequence number is within the range and follows its predecessor.

If the sequence number is beyond the range of the acceptable sequence numbers, it drops the packet and sends an ACK packet using the expected sequence number.

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

- IP address
- Port numbers
- Sequence numbers

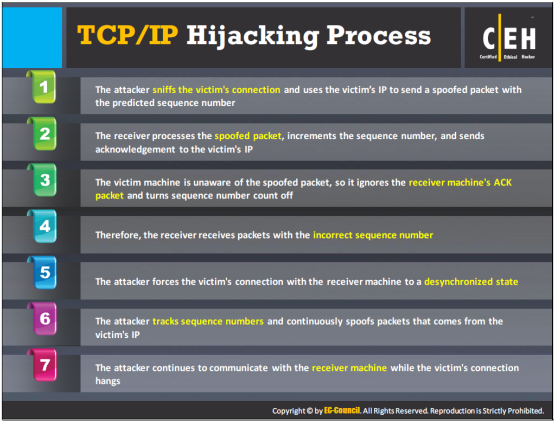
Finding out the IP address and the port number is easy; these are available in the IP packets, which do not change throughout the session. After discovering the addresses communicating with the ports, the information exchanged stays the same for the remainder of the session. However, the sequence numbers change. Therefore, the attacker must successfully guess the sequence numbers for a blind hijack. If the attacker can fool the server into receiving his/her spoofed packets and executing them, the attacker has successfully hijacked the session.

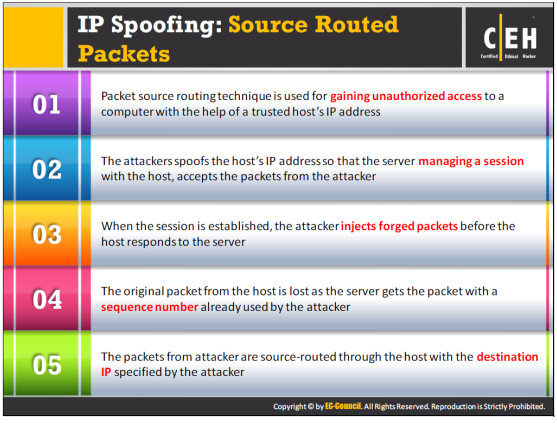


In TCP/IP hijacking, an attacker intercepts an already established connection between any two communicating parties, and then pretends to be one of them. In this approach, attacker makes use of spoofed packets to redirect the TCP traffic to his/her own machine. When this is successful, the victim's connection hangs and the attacker is able to communicate with the host's machine on behalf of the authentic victim. To launch a TCP/IP hijacking attack, the attacker must be on the same network as the victim. The target and the victim machines can be located anywhere. Using this technique, an attacker can easily attack the system, which uses one-time passwords.

In the diagram shown in the slide:

- The hacker sniffs the communication between the victim and the host in order to obtain the victim's ISN (Initial Sequence Number).
- Using the ISN, the attacker sends a spoofed packet from the victim's IP address to the host system.
- The host machine responds to the victim, assuming that the packet has arrived from it. This increments the sequence number.



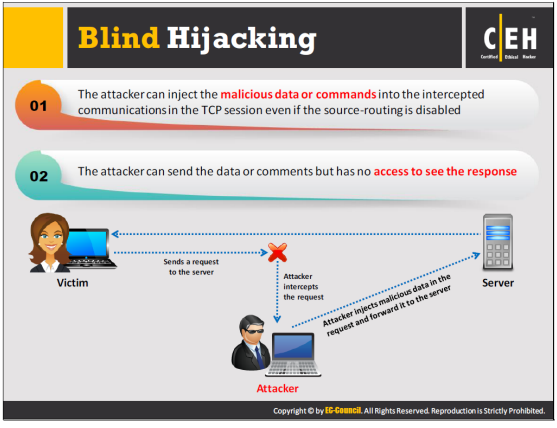


The source-routed packets technique is useful in gaining unauthorized access to a computer with the help of a trusted host's IP address. This type of hijacking allows attackers to create their own acceptable packets to insert into the TCP session. First, the attacker spoofs the trusted host's IP address. The packets are source-routed, so the sender specifies the path for packets from the source to the destination IP. Using this source-routing technique, attackers can fool the server into thinking that it is communicating with the user.

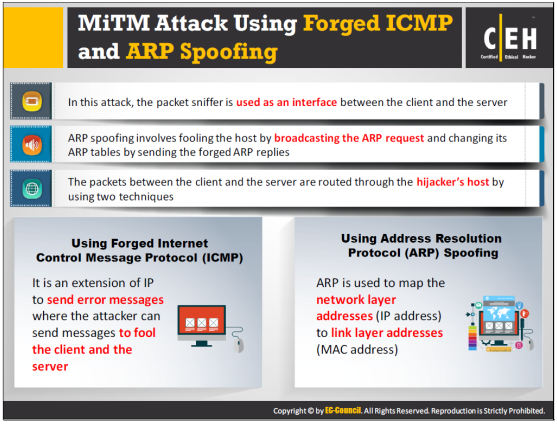
After spoofing the IP address successfully, the hijacker alters the sequence number and the acknowledgment number the server expects. Once this number is changed, the attacker must inject forged packets into the TCP session before the client can respond. This leads to the desynchronized state because there is no synchronization for the sequence and ACK numbers. The original packets are lost, and the server receives a packet with the new ISN (initial sequence number). These packets are source-routed to a patched destination IP specified by the attacker.

RST Hijacking

- RST hijacking involves injecting an **authentic-looking reset (RST) packet** using spoofed source 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 actually sent the **reset packet** and **resets the connection**
- RST Hijacking can be carried out using a **packet crafting tool** such as Colasoft's Packet Builder and TCP/IP analysis tool such as tcpdump



In blind hijacking, a hacker can inject malicious data or commands into the intercepted communications in a TCP session, even if the victim disables source routing. Here, an attacker correctly guesses the next ISN of a computer attempting to establish a connection; the attacker sends malicious data or a command, such as setting a password to allow access from another location on the network, but the attacker can never see the response. To get to the response, a man-in-the-middle attack works much better.

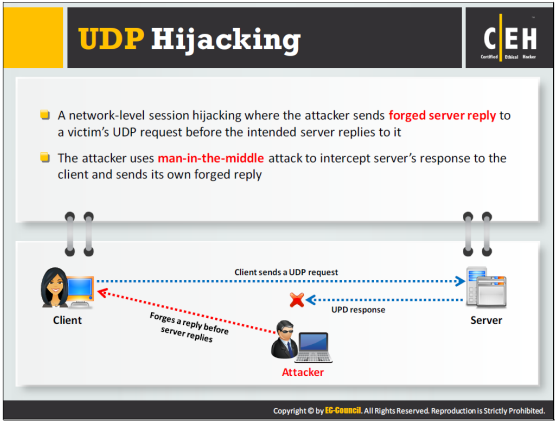


A man-in-the-middle attack uses a packet sniffer to intercept communication between the client and server. The attacker changes the default gateway of the client's machine and attempts to reroute packets.

The technique used is to forge ICMP (Internet Control Message Protocol) packets to redirect traffic between the client and the host through the hijacker's host. The hacker's packets send error messages that indicate problems in processing packets through the original connection. This fools the server and client into routing through its path instead.

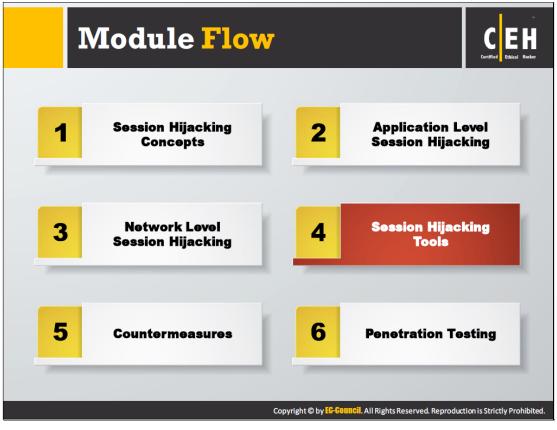
Another technique used is ARP (Address Resolution Protocol) spoofing. Hosts use ARP tables to map local IP addresses to hardware addresses or MAC addresses. The attacker sends forged ARP replies that update the ARP tables at the host that is broadcasting ARP requests. This delivers the traffic to the host instead of delivering it to the legitimate IP.

In both these techniques, an attacker routes the packets that exist between the client and server through her/his machine.



UDP does not use packet sequencing and synchronizing, so an attacker can more easily attack a UDP session than a TCP session. Since UDP is connectionless, it is extremely easy to modify data without the victim's notice. The hijacker forges a server reply to the client UDP request before the server can respond. Thus, the attacker takes the control of managing the session. There will be no exchange of packets between the server and client, as the server's sequence number does not match with that of the client's acknowledgement number and vice-versa.

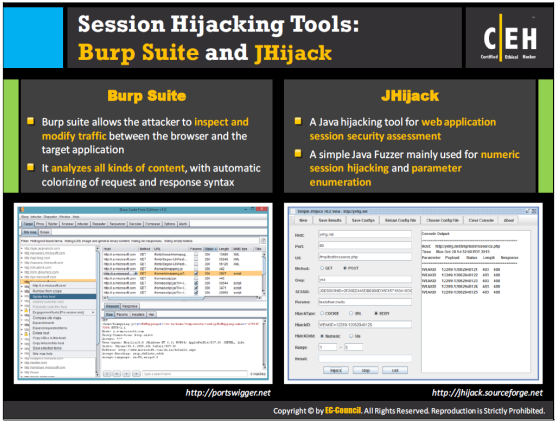
The server's reply can be easily restricted if sniffing is used. A man-in-the-middle attack in UDP hijacking can minimize the task of the attacker, as it can stop the server's reply from reaching the client in the first place.



Attackers can make use of tools such as Zaproxy, Burp Suite, and JHijack to hijack the session between client and server. This section deals with various tools that are helpful in performing session hijacking.

The Zed Attack Proxy (ZAP) is a penetration testing tool for finding vulnerabilities in web applications. This tool helps people with a wide range of security experience and as such is ideal for developers and functional testers new to penetration testing. It provides automated scanners as well as a set of tools that allow you to find security vulnerabilities manually.

Source: *https://www.owasp.org*



Listed below are the tools that help in performing session hijacking:

Burp Suite

Burp Suite is an integrated platform for performing security testing of web applications. Its various tools work together to support the entire testing process, from initial mapping and analysis of an application's attack surface through to finding and exploiting security vulnerabilities.

Burp Suite contains the following key components:

- An **intercepting Proxy**, which lets you inspect and modify traffic between your browser and the target application
- An **application-aware Spider**, for crawling content and functionality
- An **advanced web application Scanner**, for automating the detection of numerous types of vulnerability
- An **Intruder tool**, for performing powerful customized attacks to find and exploit unusual vulnerabilities
- A **Repeater tool**, for manipulating and resending individual requests
- A **Sequencer tool**, for testing the randomness of session tokens

Source: http://portswigger.net

Listed below are the tools that help an attacker to hijack a session:

Surf Jack

Source: *https://code.google.com*

Surf Jack tool allows one to hijack HTTP connections to steal cookies, even if the victim is on HTTPS sites. This tool works on both Wi-Fi (monitor mode) and Ethernet.

Ettercap

Source: *http://ettercap.github.io*

Ettercap is a comprehensive suite for man-in-the-middle attacks. It features sniffing of live connections, content filtering, and many other tricks. It supports active and passive dissection of many protocols, and includes many features for network and host analysis.

TamperIE

Source: *http://www.bayden.com*

TamperIE is a simple Internet Explorer Browser Helper Object that allows lightweight tampering of HTTP requests from Internet Explorer 5 and above. It is useful for security testing web applications, to ensure you do not misinterpret data sent by client browsers. TamperIE works inside IE itself, before placing data on the “wire”; this means that it works well even against HTTPS-secured sites.

PerJack

Source: *http://packetstormsecurity.org*

PerJack is a TCP session hijack tool written in Perl. It performs a man-in-the-middle attack, displays all active sessions, and takes over the selected TCP session.

WhatsUp Gold Engineer's Toolkit

Source: *http://www.whatsupgold.com*

WhatsUp Gold Engineer's Toolkit will help you monitor, diagnose, and troubleshoot a vast majority of network issues. It performs various tasks such as network design and planning, verifying device connectivity and DNS behavior, managing flow traffic settings and passwords, and taking remote control over servers.

Cookie Cadger

Source: *https://www.cookiecadger.com*

Cookie Cadger is an auditing tool for Wi-Fi or wired Ethernet connections. It helps identify information leakage from applications that utilize insecure HTTP GET requests. It is an open-source pen-testing tool for intercepting and replaying specific, insecure HTTP GET requests into a browser.

Firesheep

Source: *http://codebutler.github.io*

A Firefox extension that demonstrates HTTP session hijacking attacks.

CookieCatcher

Source: *https://github.com*

CookieCatcher is an open-source application that assists in the exploitation of XSS (cross-site scripting) vulnerabilities within web applications to steal user session IDs (i.e., session hijack).

Features:

- Prebuilt payloads to steal cookie data
- Will send email notification when new cookies are stolen
- Will attempt to refresh cookies every 3 minutes to avoid inactivity timeouts
- Provides HTTP requests to hijack sessions through a proxy (Burp, etc.)
- Will attempt to load a preview when viewing the cookie data
- PAYLOADS
- Basic AJAX Attack
- HTTPONLY evasion for Apache CVE-20120053

T-sight

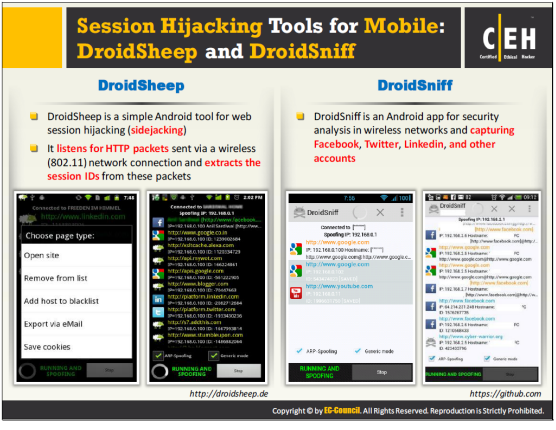
Source: *http://www.engarde.com*

T-sight is an advanced intrusion investigation and response tool for Windows that can assist when an attempt at a break-in or compromise occurs. With T-sight you can monitor all network connections in real-time and observe the composition of any suspicious activity that takes place.

sslstrip

Source: *https://pypi.python.org*

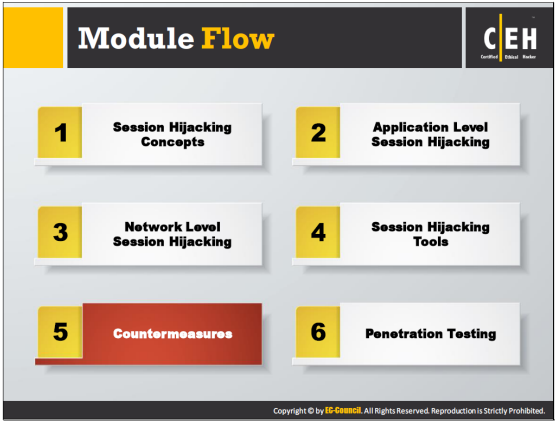
Sslstrip is a MITM tool that implements Moxie Marlinspike's HTTPS stripping attacks. It will transparently hijack HTTP traffic on a network, watch for HTTPS links and redirects, and then map those links into either look-alike HTTP links or homograph-similar HTTPS links. It also supports modes for supplying a favicon, which looks like a lock icon, selective logging, and session denial.



DroidSheep

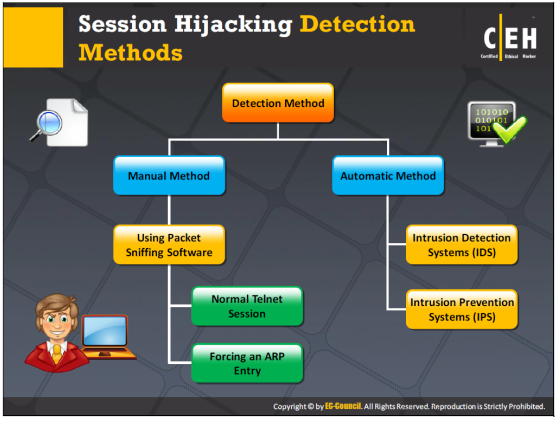
Source: http://droidsheep.de

DroidSheep can capture sessions using the libpcap library and supports: OPEN Networks, WEP encrypted networks, and WPA/WPA2 encrypted networks (PSK only). This software uses libpcap and arpspoof.



In general, hijacking is dangerous. The victim is at risk of identity theft, fraud, and loss of sensitive information. All networks that use TCP/IP are vulnerable to the types of session hijacking discussed earlier. However, following best practices might defend against session hijacking attacks.

This section deals with session hijacking detection methods, various countermeasures to combat session hijacking attacks, approaches vulnerable to session hijacking, and their preventative solutions (e.g., IPsec).



Session hijacking attacks are very hard to detect and users often overlook it, unless there is severe damage caused by the attacker.

Some of the session hijacking attack symptoms includes:

- Burst of network activity for a while, which slows down the system performance
- Busy servers resulting from requests sent by both client and hijacker

Methods to detect session hijacking:

Manual Method

The manual method involves using packet sniffing software such as Ettercap and Wireshark to monitor session hijacking attacks. The packet sniffer captures packets in transfer across the network, which is then analyzed using various filtering tools.

Normal Telnet Session

Once session is established, communication between client and server takes place via Telnet protocol. A "normal" Telnet session means the client sends packets to the server, and the server acknowledges the packet.

Forcing an ARP Entry

Forcing ARP entry involves replacing the MAC address of a compromised machine residing in the ARP cache of the server with a different one, the goal being to restrict network traffic to the compromised machine.

Look for this:

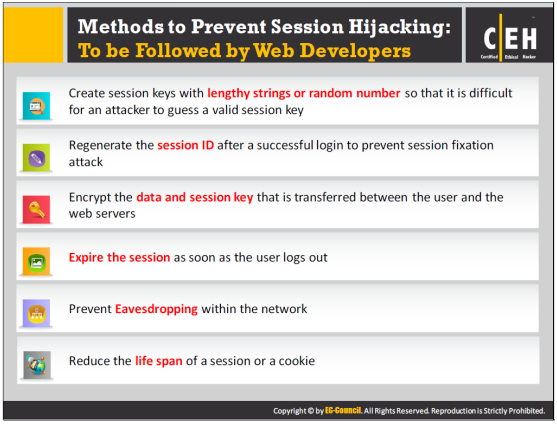
- ARP updates (repeated)
- Frames sent between client and server with different MAC addresses
- ACK storms

Automatic Method

The automatic method involves using Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (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.

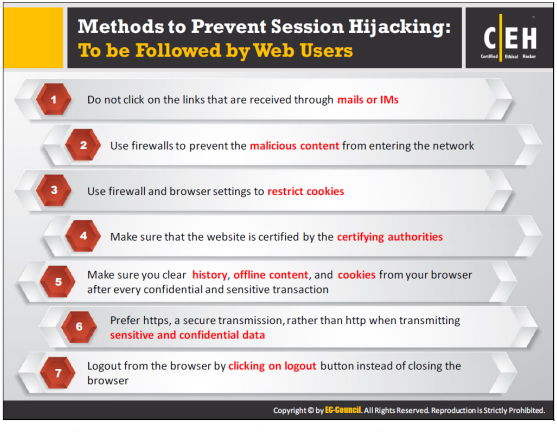
List below are various countermeasures for defending against session hijacking attacks:

- Use firewall and browser settings to confine cookies.
- Protect authentication cookies with SSL.
- Regularly update platform patches to fix TCP/IP vulnerabilities (ex: predictable packet sequences).
- Avoid including the session ID in the URL.
- Use IPsec to encrypt session information.



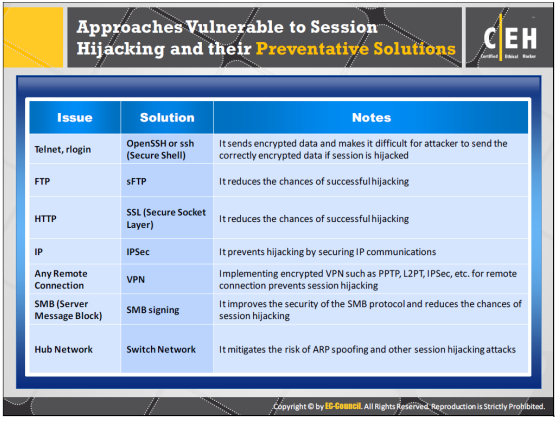
An attacker usually hijacks a session by exploiting the vulnerabilities in mechanisms used for session establishment. Web developers often ignore security. During the development process, they have to consider the guidelines given below to minimize/eliminate the risk of session hijacking:

- Implement Secure Sockets Layer (SSL) to encrypt all the information in transit via the network.
- Use restrictive cache directives for all the web traffic exchanged through HTTP and HTTPS, 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.

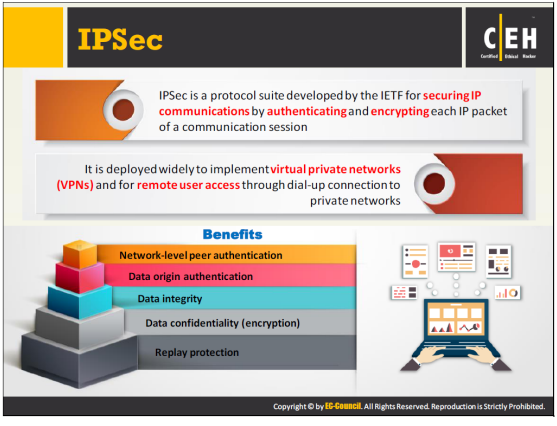


Below is a list of countermeasures intended for web users to defend against session hijacking:

- Verify and disable add-ons from the untrusted site. Enable add-on only if necessary.
- Practice using a one-time password for any critical data transactions (e.g., credit card).
- Frequently update anti-virus signatures to prevent the automatic installation of malware trying to steal cookies.



Implementing encryption and signing protocols makes difficult the attacker's task of hijacking the session. Shown in the slide are various issues and their respective solutions that, on implementation, prevent or impede taking over the valid session.



Internet Protocol Security (IPsec) is a set of protocols that the IETF (Internet Engineering Task Force) developed to support the secure exchange of packets at the IP layer. It ensures interoperable cryptographically based security for IP protocols (IPv4 and IPv6), and supports network-level peer authentication, data origin authentication, data integrity, data confidentiality (encryption), and replay protection. It is widely used to implement virtual private networks (VPNs) and for remote user access through dial-up connection to private networks. It supports transport and tunnel encryption modes, though sending and receiving devices must share a public key.

You can assign IPsec policies through Group Policy configuration of Active Directory domains, organizational units, and IPsec deployment policies at the domain, site, or organizational-unit level.

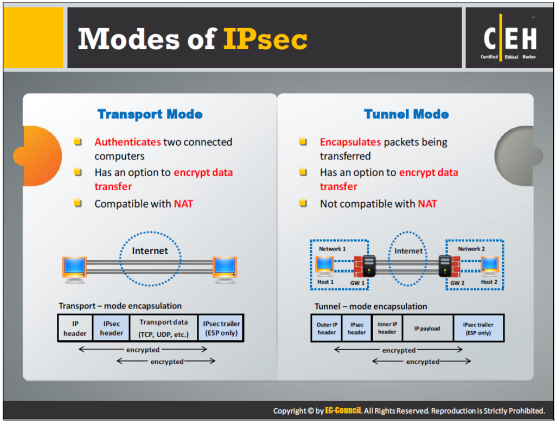
Security services offered by the IPsec include:

- Rejection of replayed packets (a form of partial sequence integrity)
- Data confidentiality (encryption)
- Access control
- Connectionless integrity
- Data origin authentication
- Limited traffic flow confidentiality
- Network-level peer authentication

At the IP layer, IPsec provides all the above-mentioned services offering protection for IP and/or upper layer protocols such as TCP, UDP, ICMP, and BGP.

Listed below are the steps involved in the IPsec process:

- A consumer sends a message to a service provider.
- The consumer's IPsec driver attempts to match the outgoing packet's address or the packet type against the IP filter.
- The IPsec driver notifies ISAKMP (Internet Security Association and Key Management Protocol) to initiate security negotiations with the service provider.
- The service provider's ISKAMP receives the security negotiations request.
- Both principals initiate a key exchange, establishing an ISAKMP SA (ISAKMP Security Association) and a shared secret key.
- Both principals discuss the security level for the information exchange, establishing both IPsec SAs and keys.
- Consumer's IPsec driver transfers packets to the appropriate connection type for transmission to the service provider.
- The provider receives the packets and transfers them to the IPsec driver.
- Provider's IPsec uses the inbound SA and key to check the digital signature and begin decryption.
- Provider's IPsec driver transfers decrypted packets to the OSI Transport layer for further processing.



The configuration of IPsec involves two different modes: ***tunnel* mode** and ***transport* mode**. These modes are associated with the function of two core protocols, the **Encapsulation Security Payload** (ESP) and **Authentication Header** (AH). Selection of mode depends on the requirements and implementation of IPsec.

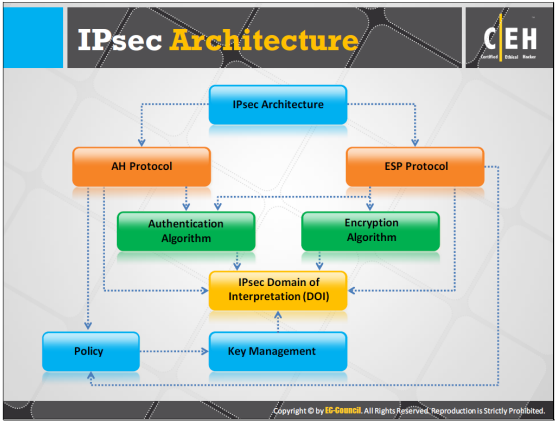
- **Transport Mode**

  In transport mode (also ESP [**Encapsulating Security Payload**]), 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 NAT; therefore, it is useful to provide VPN services for network utilizing NAT.

- **Tunnel Mode**

  In tunnel mode (also AH [**Authentication Header**]), the IPsec encrypts both the payload and the header. Hence, there is more security in tunnel mode. After receiving, the IPsec-compliant device decrypts the data. NAT is unable to rewrite the encrypted IP header and as the tunnel mode encrypts the header of the IP packet, it is not capable of providing VPN services.

  In tunnel mode, the system encrypts the entire IP packet (payload and IP header) and encapsulates the encrypted packets into a new IP packet with a new header. In this mode, ESP encrypts and optionally authenticates the entire inner IP packet, whereas AH authenticates the entire inner IP packet and selected fields of the outer IP header. Tunnel mode is usually useful between two gateways or between a host and a gateway.



IPsec offers security services at the network layer. This provides the freedom to select the required security protocols and determine the algorithms used for services. To provide the requested services, employ the corresponding cryptographic keys, if required. Security services offered by the IPsec include access control, data origin authentication, connectionless integrity, anti-replay, and confidentiality. To meet these objectives, IPsec uses two traffic security protocols AH (Authentication Header) and ESP (Encapsulating Security Payload) and cryptographic key management protocols and procedures.

Protocol structure of the IPsec architecture:

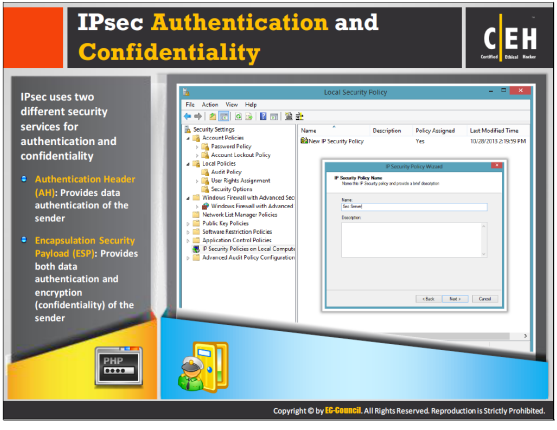
Authentication Header (AH): It offers integrity and data origin authentication, with optional anti-replay features.

Encapsulating Security Payload (ESP): It offers all the services offered by Authentication Header (AH) and in addition, it offers confidentiality.

IPsec Domain of Interpretation (DOI): It defines the payload formats, types of exchange, and naming conventions for security information such as cryptographic algorithm or security policies. IPsec DOI instantiates ISAKMP for use with IP when IP uses ISAKMP to negotiate security associations.

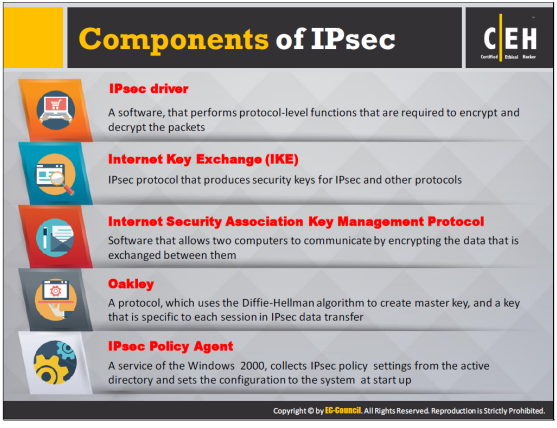
ISAKMP (Internet Security Association and Key Management Protocol): It is a key protocol in the IPsec architecture. It establishes the required security for various communications on the Internet such as government, private, and commercial, by combining the security concepts of authentication, key management, and security associations.

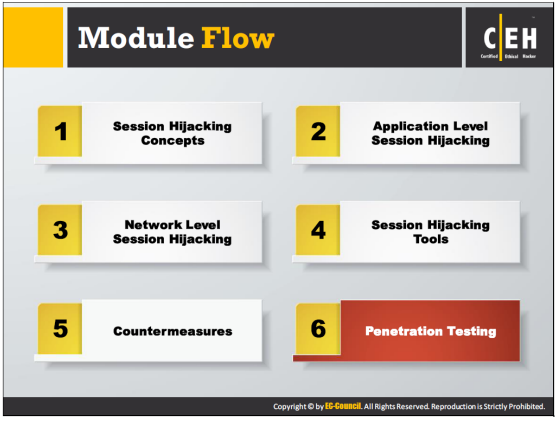
Policy: IPsec policies are useful in providing network security. They define when and how to secure data, and security methods to use at different levels in the network. One can configure IPsec policies to meet security requirements of a system, domain, site, organizational unit, and so on.



IPsec uses two different security services for authentication and confidentiality:

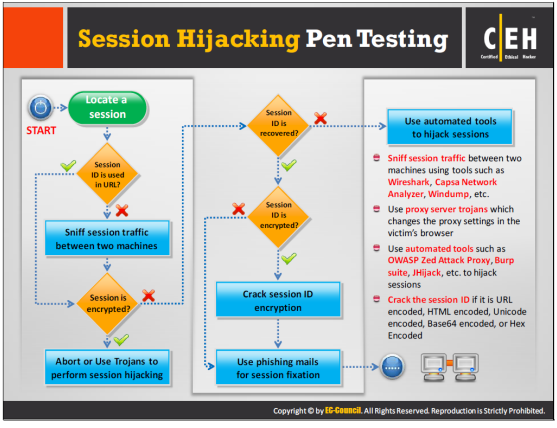
- **Authentication Header (AH)**: It is useful in providing connectionless integrity and data origin authentication for IP datagrams and anti-replay protection for the data payload and some portions of IP header of each packet. It does not support data confidentiality (no encryption). A receiver can select a service to protect against replays, which is an optional service upon establishing a **Security Association** (SA).
- **Encapsulation Security Payload (ESP)**: In addition to the services (data origin authentication, connectionless integrity, and anti-replay service) provided by the Authentication Header (AH), the Encapsulating Security Payload (ESP) protocol offers confidentiality. Unlike AH, ESP does not provide integrity and authentication for the entire IP packet in transport mode. You can apply ESP alone, or in conjunction with AH, or in a nested fashion. It protects only the IP data payload on default setting. In tunnel mode, it protects both the payload and the IP header.





Various hijacking methods exist, using which attackers exploit design flaws inherent in the TCP/IP protocol suite to take over a valid session. Therefore, it is a good practice to pen-test regularly for session hijacking attacks.

This section deals with a pen-testing method that helps in identifying session hijacking attacks at both the application-level and the network-level.



Session hijacking pen testing involves the same process as that of the session hijacking attack. For this, first the pen tester should locate a session. Then he or she should check for various possibilities to hijack a session. This may vary, depending on the network and mechanisms used for communication. Given below is the standard procedure for session hijacking pen testing:

Step 1: Locate a session

As already mentioned, the first step is to locate a target active session through packet sniffing to take control of it.

After locating a session, check whether the URL uses a session ID; if so, then check whether the session is encrypted. If a session ID is not used, then proceed to step 2.

Step 2: Sniff session traffic between two machines

Sniff session traffic between two machines using tools such as Wireshark, Windump, and Capsa Network Analyzer. Once done, check the session for encryption.

After encrypting the session, abort it, or use Trojans to hijack it. If there is no session encryption, then recover the session ID.

If you are not able to recover session IDs from the unencrypted session, use automated tools such as OWASP Zed attack proxy, Burp Suite, or JHijack to hijack sessions.

After recovering the session ID, check whether it is encrypted.

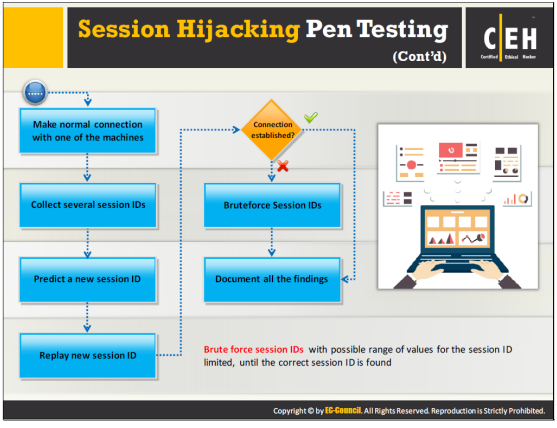
Step 3: Crack Session ID encryption

If the session ID is URL encoded, HTML encoded, Unicode encoded, Base64 encoded, or Hex Encoded, then crack the session ID.

Usually, session IDs are responsible for user authentication. If you are able to recover the session IDs of an authentic user, then you can inject yourself between the victim's machine and the remote machine and use this unauthorized connection for your own malicious purposes.

Step 4: Send Phishing email for Session Fixation

If you succeed in cracking the session ID encryption, or if the session ID is not encrypted, then send phishing mails to the victim to employ session fixation.



Step 5: Make a normal connection with one machine

After employing session fixation, you can make a normal connection with the victim's machine and access the machine remotely by masking yourself as an authorized user of the network.

Step 6: Collect several session IDs

Once you connect to one of the machines in the network, you can collect several session IDs. There are two different techniques available for retrieving session: from a cookie in the response headers, and by matching a regular expression against the response body.

Step 7: Predict a new session ID

Now, analyze the collected session IDs to predict or guess the new session ID. You should predict a new session ID to find the current session ID and engage in a replay attack.

Step 8: Replay new session ID

A replay attack occurs when you copy a stream of messages (session IDs) between two parties and replay the stream to one or more of the parties. Unless mitigated, the computers subject to the attack process the stream as legitimate session IDs, resulting in a range of unwanted consequences, such as redundant orders of an item.

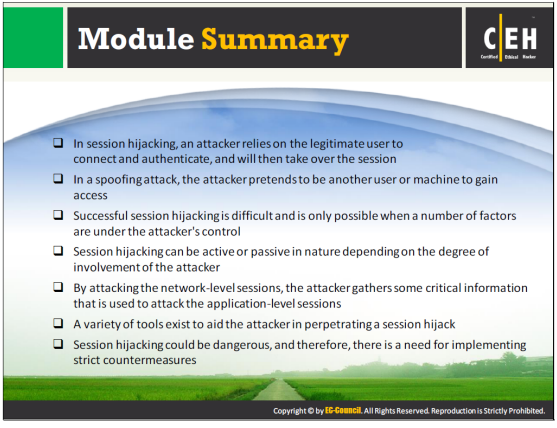
Now, check for the establishment of connection. If the connection is established, then you should document all the findings of the penetration testing; otherwise, you should apply a brute force attack to find the current valid session ID in order to establish the connection.

Step 9: Brute force session IDs

Brute force session IDs with a possible range of values for the session ID limited, until you obtain a correct session ID. This involves making thousands of requests using all the randomly generated session IDs. This technique is comprehensive but a time consuming process.

Step 10: Document all the findings

Finally, document all the findings at each step of the pen testing methodology for analysis and future reference.



This module ends with an overview of session hijacking concepts, application and network-level session hijacking, countermeasures against it, tools for its use, and pen testing. In the next module, we will see how attackers as well as ethical hackers and pen testers perform web server hacking to get valuable information such as credit card numbers and passwords.