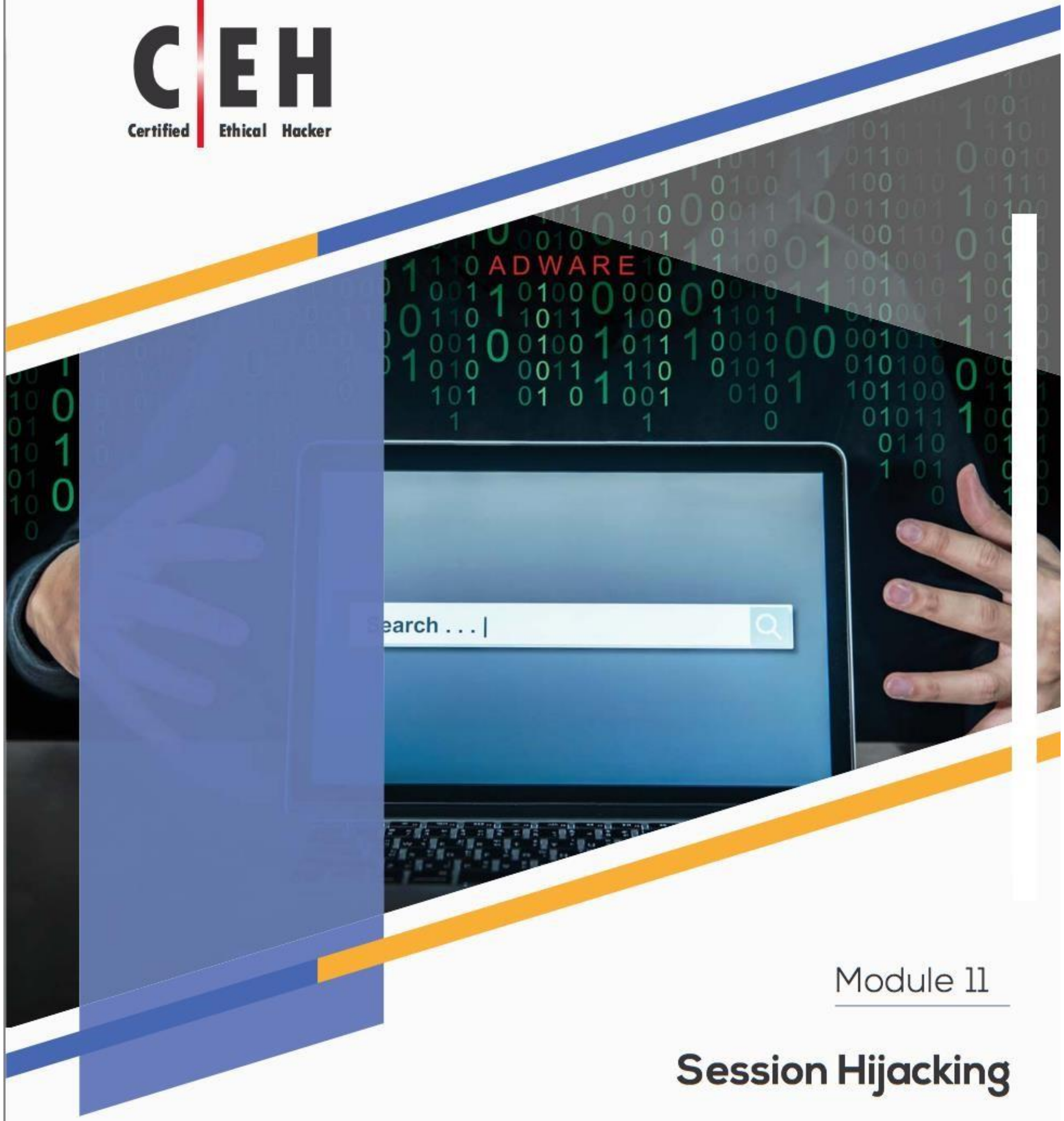
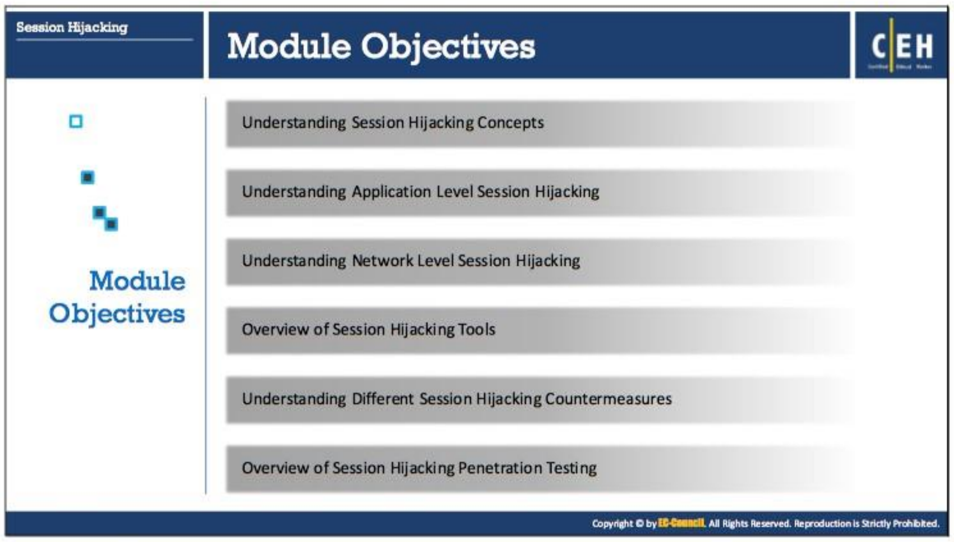
C|EH™
Certified Ethical Hacker



Module 11

Session Hijacking



Session Hijacking

Module Objectives

Module Objectives

- Understanding Session Hijacking Concepts
- Understanding Application Level Session Hijacking
- Understanding Network Level Session Hijacking
- Overview of Session Hijacking Tools
- Understanding Different Session Hijacking Countermeasures
- Overview of Session Hijacking Penetration Testing

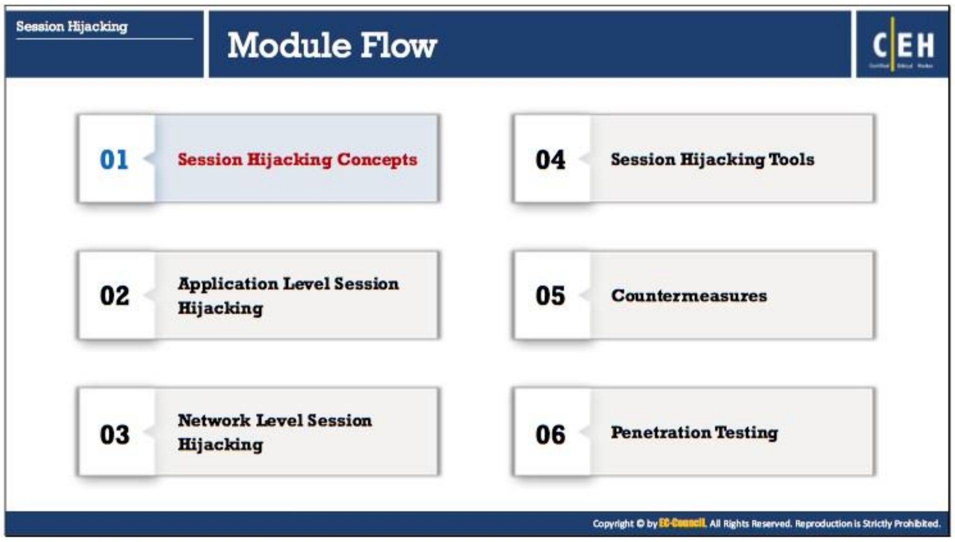
Module Objectives

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

This module aims to provide comprehensive information on session hijacking. It starts with an introduction to session hijacking concepts, and provides an insight into session hijacking at the application and network-levels. Later, the module discusses session hijacking tools and the countermeasures. It concludes with an overview of penetration testing steps an ethical hacker should follow while assessing security assessment.

At the end of this module, you will be able to:

- Describe the session hijacking concepts
- Perform application level session hijacking
- Perform network level session hijacking
- Apply different session hijacking tools
- Apply session hijacking countermeasures
- Perform session hijacking penetration testing



Session Hijacking

Module Flow

01 Session Hijacking Concepts

02 Application Level Session Hijacking

03 Network Level Session Hijacking

04 Session Hijacking Tools

05 Countermeasures

06 Penetration Testing

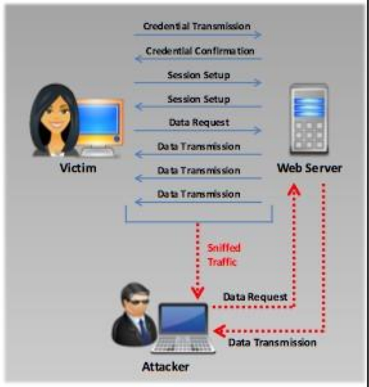
Session Hijacking Concepts

To understand session hijacking, familiarization with basic concepts is important. This section answers: What is session hijacking? 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.

Session Hijacking
Session Hijacking Concepts

What is Session Hijacking?

- Session hijacking refers to an attack where an attacker takes over a **valid TCP communication session** between two computers
- Since most **authentication only occurs at the start of a TCP session**, this allows the attacker to gain access to a machine
- Attackers can sniff all the traffic from the established TCP sessions and perform **identity theft**, **information theft**, **fraud**, etc.
- The attacker steals a valid session ID and uses it to **authenticate himself with the server**



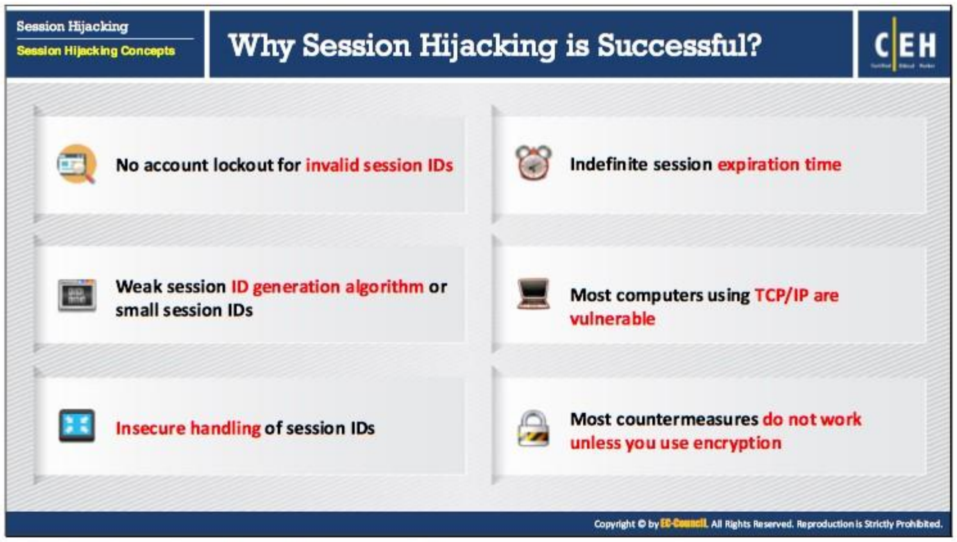
What is Session Hijacking?

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

Session hijacking refers to an attack where an attacker takes over a valid TCP communication session between two computers. Since most authentication only occurs at the start of a TCP session, it allows the attacker to gain access to a machine. Attackers can sniff all the traffic from the established TCP sessions and perform identity theft, information theft, fraud, etc.

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 the 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 Denial-of-Service (DoS) attacks. In MITM attack, the attacker places himself between the client and server. Session hijacking enables attackers to insert themselves in between the authorized client and the web server to ensure information flowing in either direction must pass through them. The client and the server believe they are directly communicating with each other however, the traffic between them passes through the attacker. Attackers can sniff sensitive information and disrupt the sessions to cause a DoS attack.



Session Hijacking
Session Hijacking Concepts

Why Session Hijacking is Successful?

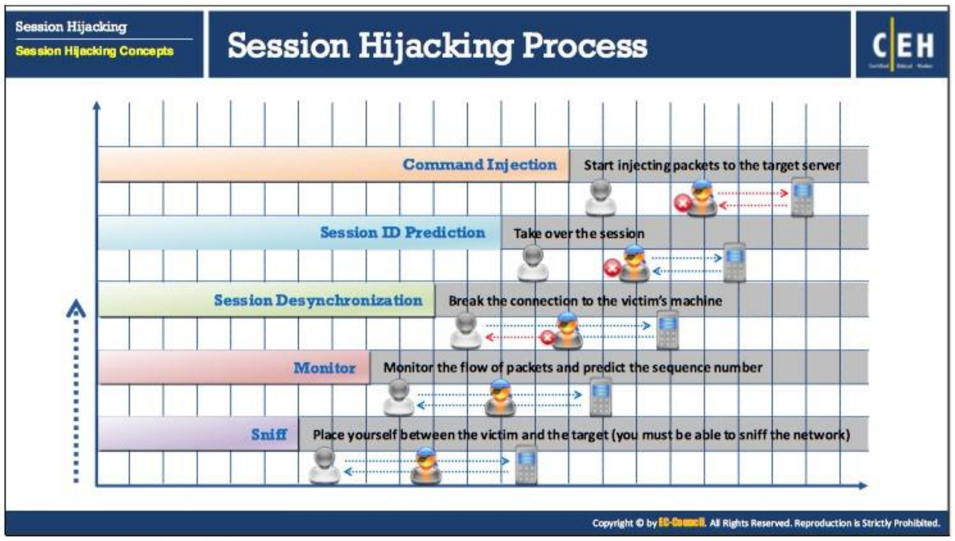
- No account lockout for invalid session IDs
- Indefinite session expiration time
- Weak session ID generation algorithm or small session IDs
- Most computers using TCP/IP are vulnerable
- Insecure handling of session IDs
- Most countermeasures do not work unless you use encryption

Why Session Hijacking is Successful?

Session hijacking is successful because of the following factors:

- **Weak session-ID generation algorithm or small session IDs:** Most websites use linear algorithms to predict variables such as time or IP address for generating session IDs. By studying the sequential pattern and generating multiple requests, an attacker can easily alleviate the search space necessary to forge 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-timeout:** 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 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 without encryption:** It is easy to sniff session ID on a flat network if transport security is not set up properly during transmission of session ID cookies, even if web application uses SSL encryption. An attacker's job becomes even easier if he/she captures the session IDs containing actual logon information.
- **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 exploitation of a bug in the browser.

- **Computers using TCP/IP are vulnerable:** All machines running TCP/IP are vulnerable to session hijacking because of 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 several attempts to connect with varying session IDs embedded in a genuine URL. An attacker can continue until the actual session ID is determined. This is also known as brute force attack. During brute-force attack, the web server does not display a warning message or complaint, thus allowing the attacker to determine the valid session ID.


Session Hijacking
Session Hijacking Concepts
Session Hijacking Process
CEH
Command Injection
Start injecting packets to the target server
Session ID Prediction
Take over the session
Session Desynchronization
Break the connection to the victim's machine
Monitor
Monitor the flow of packets and predict the sequence number
Sniff
Place yourself between the victim and the target (you must be able to sniff the network)
Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Session Hijacking Process

It is easier to sneak into a system as a genuine user than entering 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 traffic destined to the user's IP address goes to the attacker's system. During this period, the attacker can plant backdoors or gain additional access to the system. How does an attacker go about hijacking a session?

Session hijacking can be divided 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 identifying victim, an attacker captures the sequence and acknowledgment numbers of the victim because TCP checks the sequence/ 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 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 the 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 the 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 without affecting anything else. Thus, synchronizing the server and the target.

Another approach is to send a reset flag to the server to bring down the connection on the server side. Ideally, it 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 the packet, the attacker immediately sends an RST packet and a SYN packet with exactly the same 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 and sends back an ACK packet to the server. Now the server is in the established state. The aim is to keep the target conversant, and switch to the established state once it receives the first SYN/ACK packet from the server. Both server and target are now desynchronized, but in a established state.

An attacker can also use a FIN flag, but this will make server respond with an ACK thus giving 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 the 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.

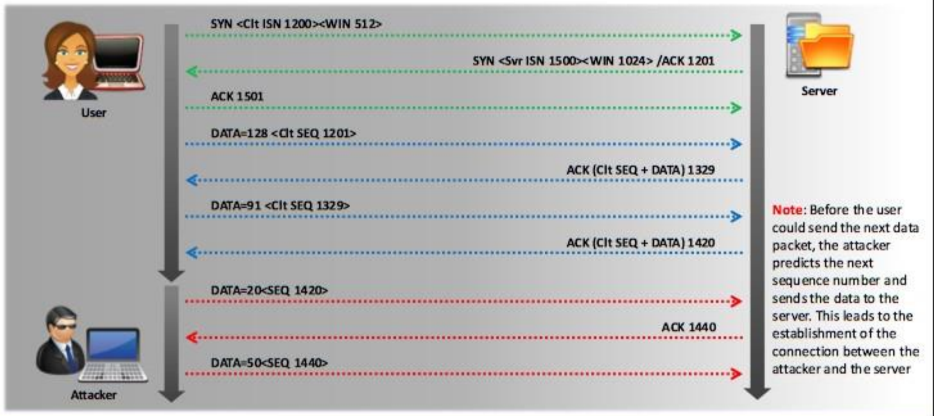
An attacker can add the desynchronizing stage to the hijack sequence to deceive the target host. Without desynchronizing, the attacker injects data into the server while keeping his or her identity by spoofing an IP address. However, the attacker should ensure that the server responds to the target host as well.

- **Injecting the attacker's packet**

  Once the attacker has interrupted the connection between the server and the target, he or she 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.

Session Hijacking
Session Hijacking Concepts

Packet Analysis of a Local Session Hijack


User
SYN <Clt ISN 1200><WIN 512>
SYN <Svr ISN 1500><WIN 1024> /ACK 1201
ACK 1501
DATA=128 <Clt SEQ 1201>
ACK (Clt SEQ + DATA) 1329
DATA=91 <Clt SEQ 1329>
ACK (Clt SEQ + DATA) 1420
DATA=20<SEQ 1420>
ACK 1440
DATA=50<SEQ 1440>
Server
Attacker

Note: Before the user could send the next data packet, the attacker predicts the next sequence number and sends the data to the server. This leads to the establishment of the connection between the attacker and the server

Packet Analysis of a Local Session Hijack

Session hijacking are high-level attack vectors which affects many systems. Many systems that establish LAN or the Internet connections 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:

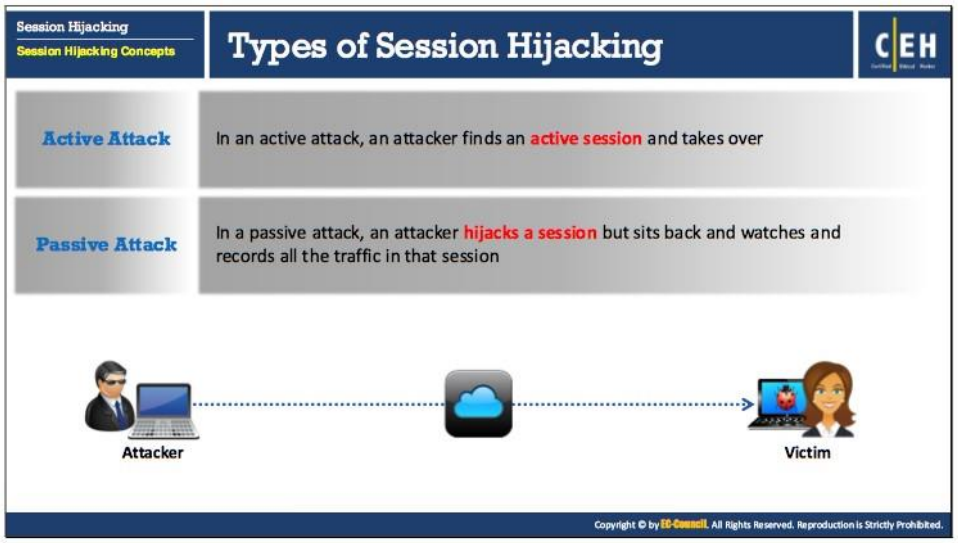
- Track a session
- Desynchronizes session
- Injects attacker's commands in between

By sniffing network traffic, an attacker can monitor or track the session. The next step in session hijacking is to desynchronize. It is easy to accomplish this attack, if the attacker knows the next sequence number used by the client. A session can be hijacked by using the sequence number before the client can use it. There are two possibilities to determine sequence numbers. One is to sniff the traffic, finding the ACK packet and then determining the next sequence number based on the ACK packet. The other is to transmit the data with guessed sequence numbers. The second one is not reliable. If you can access the network and sniff the TCP session, you can easily determine the sequence number. This kind of session hijacking refers to "local session hijacking."

The diagram in the slide shows the packet analysis of a local session hijacking.

According to the diagram, the next expected sequence number should 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 would send the data with the expected sequence number before the user sends it. Now, the server is in synchronization with the attacker. This leads to establishment of a connection between the attacker and the server. Once the connection between the attacker and the server is established, 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, as she/he is not receiving an ACK for her/his TCP packet. However, the server would drop the packet again. Thus, completing an attacker's local session hijacking attack.



Session Hijacking
Session Hijacking Concepts

Types of Session Hijacking

Active Attack	In an active attack, an attacker finds an **active session** and takes over
Passive Attack	In a passive attack, an attacker **hijacks a session** but sits back and watches and records all the traffic in that session

Attacker
Victim

Types of Session Hijacking

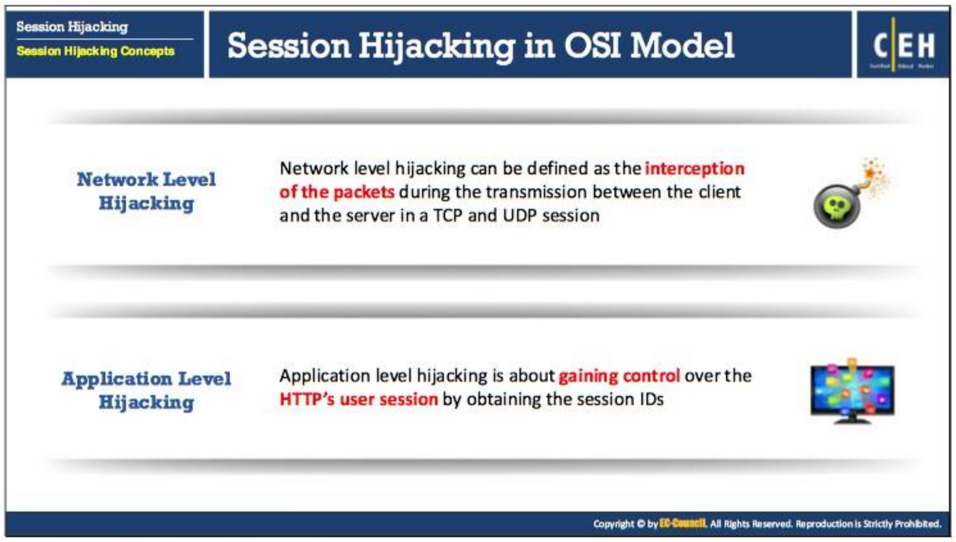
Session hijacking can be either active or passive, depending on the degree of involvement of the attacker. The essential difference between an active and passive hijacking is that while an active attack takes over an existing session, a passive hijack monitors an ongoing session.

- **Passive Session Hijacking**

  With a passive attack, an attacker hijacks a session but sits back and watches and records all the traffic that is being sent forth. 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 enjoy the privileges. Password sniffing is the simplest attack to obtain raw access to a network. Countering this attack involves 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.

- **Active Session Hijacking**

  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. To make this attack to successful, 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 it harder to predict sequential numbers.



Session Hijacking
Session Hijacking Concepts

Session Hijacking in OSI Model

Network Level Hijacking — Network level hijacking can be defined as the **interception of the packets** during the transmission between the client and the server in a TCP and UDP session

Application Level Hijacking — Application level hijacking is about **gaining control** over the **HTTP's user session** by obtaining the session IDs

Session Hijacking in OSI Model

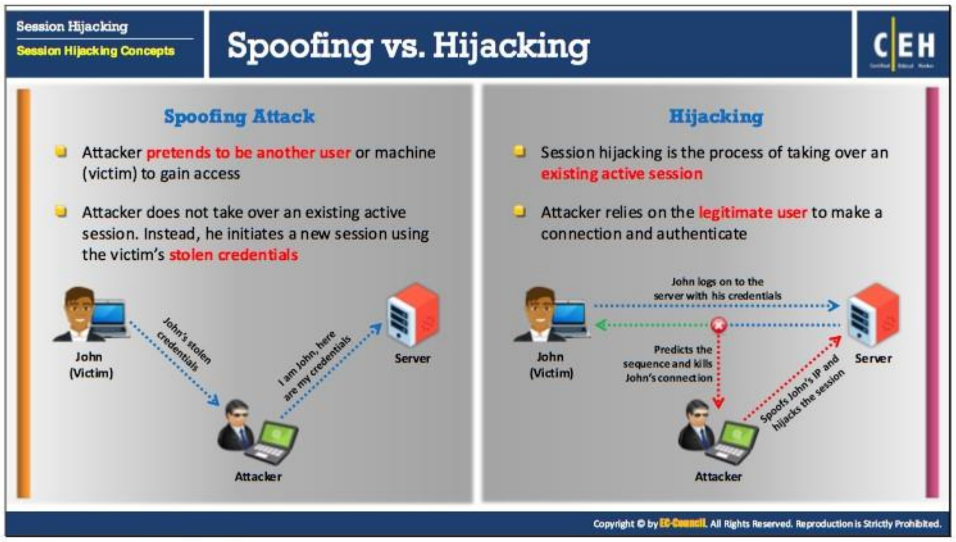
There are two levels of Session hijacking in the OSI model - the network level and the application level.

- **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 will provide the attacker with crucial information, which will be used to attack the application level sessions. Most likely attackers perform network-level hijacking because 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 create new unauthorized sessions by using stolen data. In general, both of them occur together, according to the system being attacked.



Session Hijacking
Session Hijacking Concepts

Spoofing vs. Hijacking

Spoofing Attack

- Attacker **pretends to be another user** or machine (victim) to gain access
- Attacker does not take over an existing active session. Instead, he initiates a new session using the victim's **stolen credentials**

John (Victim)
John's stolen credentials
I am John, here are my credentials
Server
Attacker

Hijacking

- Session hijacking is the process of taking over an **existing active session**
- Attacker relies on the **legitimate user** to make a connection and authenticate

John logs on to the server with his credentials
John (Victim)
Predicts the sequence and kills John's connection
Spoofs John's IP and hijacks the session
Server
Attacker

Spoofing vs. Hijacking

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

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.

In spoofing attack, an attacker pretends to be another user or machine (victim) to gain access. Attacker does not take over an existing active session. Instead he initiates a new session using the victim's stolen credentials. 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.

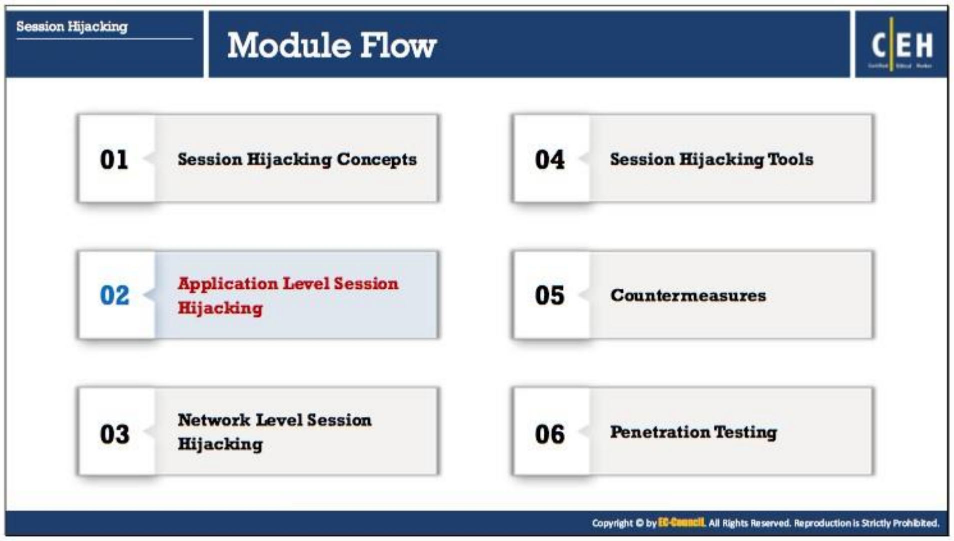
Another method is to use source-routed IP packets. This man-in-the-middle attack allows an attacker to become a part of the target-host conversation by deceptively guiding the IP packets to pass through his or her system.

Session hijacking is the process of taking over an existing active session. An attacker relies on the legitimate user to make a connection and authenticate. 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.

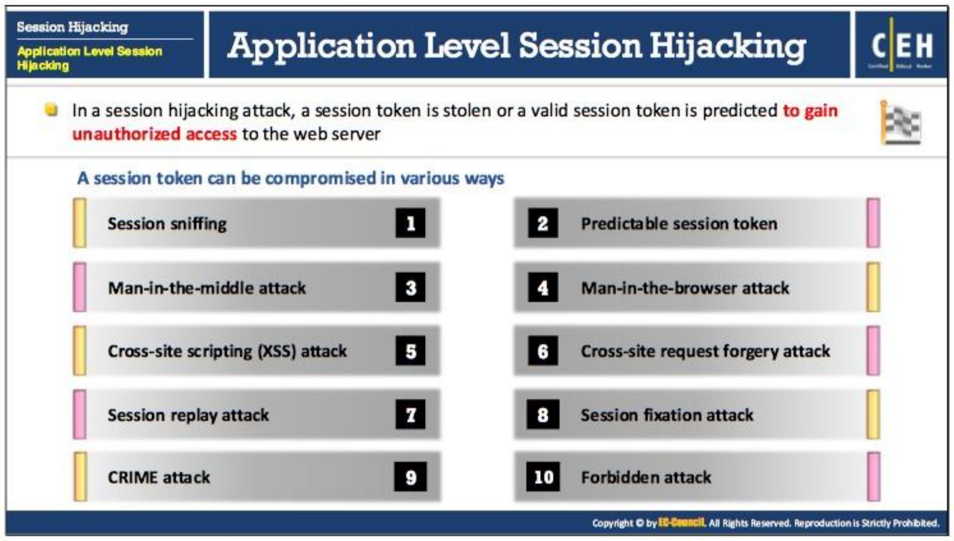


Session Hijacking

Module Flow

01 Session Hijacking Concepts	04 Session Hijacking Tools
02 Application Level Session Hijacking	05 Countermeasures
03 Network Level Session Hijacking	06 Penetration Testing

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.



Session Hijacking
Application Level Session Hijacking

Application Level Session Hijacking

- In a session hijacking attack, a session token is stolen or a valid session token is predicted **to gain unauthorized access** to the web server

A session token can be compromised in various ways

1 Session sniffing	2 Predictable session token
3 Man-in-the-middle attack	4 Man-in-the-browser attack
5 Cross-site scripting (XSS) attack	6 Cross-site request forgery attack
7 Session replay attack	8 Session fixation attack
9 CRIME attack	10 Forbidden attack

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Application Level Session Hijacking

In application-level hijacking, the attacker obtains the session IDs to get control over 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 such as session sniffing, predictable session token, etc.

In an application-level session hijacking, an attacker steals or predicts a valid session to gain unauthorized access to the web server. Usually, network-level and application-level session hijacking occur together this is so because 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.

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

- **Stealing:** Attackers use different techniques to steal session IDs. An attacker can steal the session key through 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 SteelCentral Packet Analyzer to sniff the traffic between the client and server to extract the session IDs from the packets.
- **Guessing:** An attacker tries to guess the session IDs by observing session variables. As far as guessing the session ID in to hijack the session is concerned, 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.

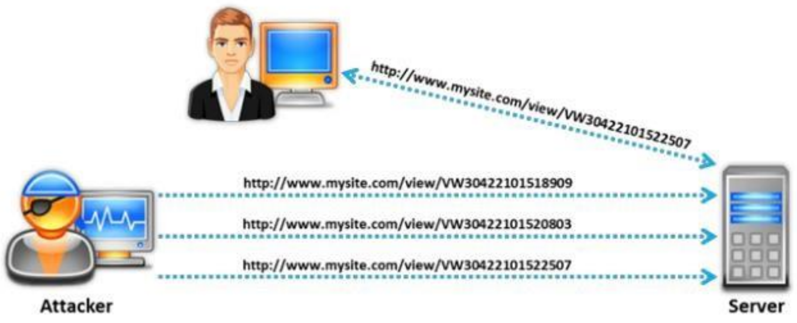

http://www.mysite.com/view/VW30422101522507
http://www.mysite.com/view/VW30422101518909
http://www.mysite.com/view/VW30422101520803
http://www.mysite.com/view/VW30422101522507
Attacker
Server

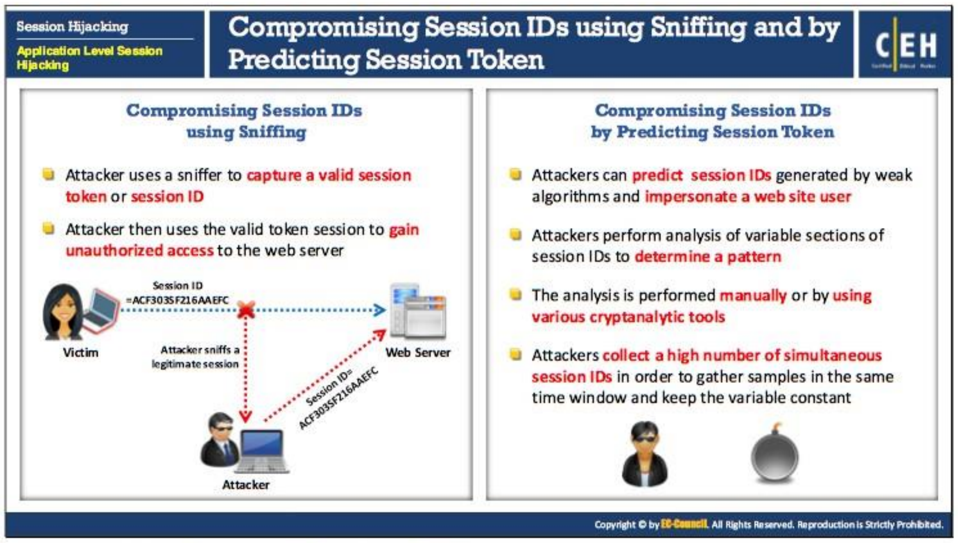
FIGURE 11.1: Attacker brute-forcing session ID of a user

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 hope 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.

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

A session token can be compromised in various ways:

- Session sniffing
- Predictable session token
- Man-in-the-middle attack
- Man-in-the-browser attack
- Cross-site scripting (XSS) attack
- Cross-site request forgery attack
- Session replay attack
- Session fixation attack
- CRIME attack
- Forbidden attack

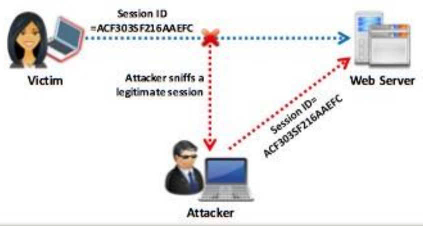


Session Hijacking
Application Level Session Hijacking

Compromising Session IDs using Sniffing and by Predicting Session Token

Compromising Session IDs using Sniffing

- Attacker uses a sniffer to **capture a valid session token** or **session ID**
- Attacker then uses the valid token session to **gain unauthorized access** to the web server


Session ID =ACF303SF216AAEFC
Victim
Attacker sniffs a legitimate session
Web Server
Session ID= ACF303SF216AAEFC
Attacker

Compromising Session IDs by Predicting Session Token

- Attackers can **predict session IDs** generated by weak algorithms and **impersonate a web site user**
- Attackers perform analysis of variable sections of session IDs to **determine a pattern**
- The analysis is performed **manually** or by **using various cryptanalytic tools**
- Attackers **collect a high number of simultaneous session IDs** in order to gather samples in the same time window and keep the variable constant

Compromising Session IDs using Sniffing

Web server identifies the user's connection through unique session ID (also known as session token). The web server sends a session token to the 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.

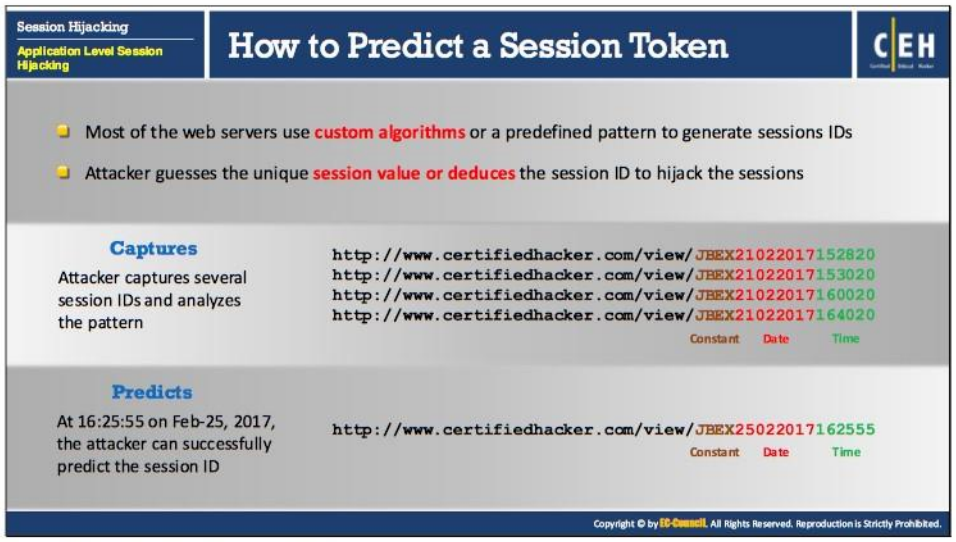
The attacker uses packet sniffing tools such as **Wireshark**, **SteelCentral Packet Analyzer**, among others to intercept the HTTP traffic between the victim and the web server. He/she then analyzes the data in the captured packets to identify valuable information such as session IDs, passwords. Once 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. Attacker uses the valid token session to gain unauthorized access to the web server. This way, an attacker takes control over an existing legitimate session.

Compromising Session IDs by Predicting Session Token

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, attackers can predict session IDs generated by weak algorithms and impersonate a web site user. Attackers perform analysis of variable section of session IDs to determine the existence of a pattern. She/he performs this analysis either manually or by using various cryptanalytic tools.

Attackers collect 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. She/he then studies 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.



Session Hijacking
Application Level Session Hijacking

How to Predict a Session Token

- Most of the web servers use **custom algorithms** or a predefined pattern to generate sessions IDs
- Attacker guesses the unique **session value or deduces** the session ID to hijack the sessions

Captures

Attacker captures several session IDs and analyzes the pattern

```
http://www.certifiedhacker.com/view/JBEX21022017152820
http://www.certifiedhacker.com/view/JBEX21022017153020
http://www.certifiedhacker.com/view/JBEX21022017160020
http://www.certifiedhacker.com/view/JBEX21022017164020
                                    Constant Date Time
```

Predicts

At 16:25:55 on Feb-25, 2017, the attacker can successfully predict the session ID

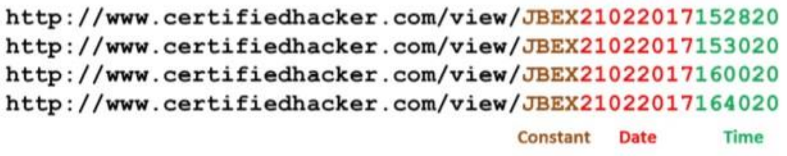
```
http://www.certifiedhacker.com/view/JBEX25022017162555
                                    Constant Date Time
```

How to Predict a Session Token

Most web servers generate Session IDs using custom algorithms or a pre-defined pattern that might simply increase static numbers, whereas others use more complex procedures such as factoring in time and other computer specific variables. Thus, one can identify the session 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.certifiedhacker.com/view/JBEX21022017152820
http://www.certifiedhacker.com/view/JBEX21022017153020
http://www.certifiedhacker.com/view/JBEX21022017160020
http://www.certifiedhacker.com/view/JBEX21022017164020
                                    Constant Date Time
```

FIGURE 11.2: Sample sessions captured by the attacker

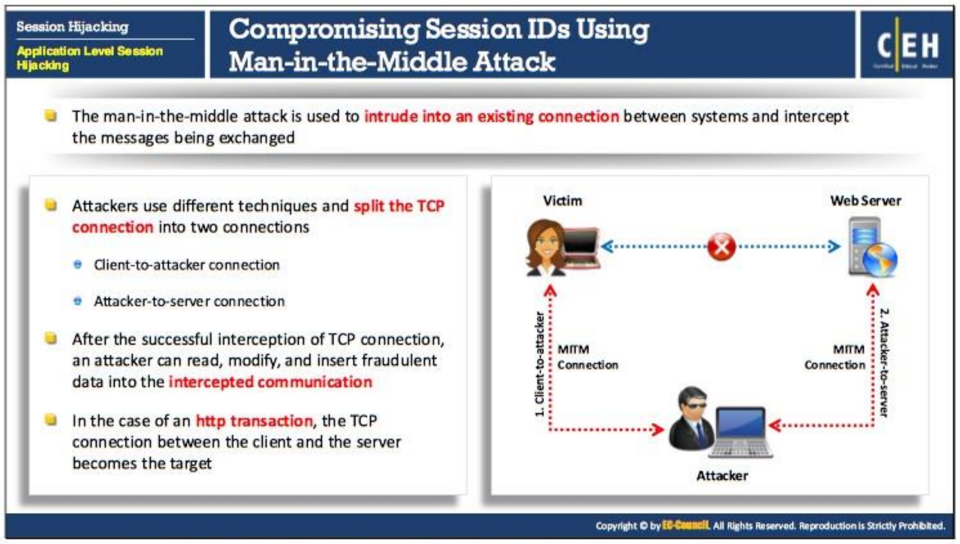
On analyzing the pattern, at 16:25:55 on Feb-25, 2017, the attacker can successfully predict the session ID:

```
http://www.certifiedhacker.com/view/JBEX25022017162555
                                  Constant  Date   Time
```

FIGURE 11.3: A session predicted by the attacker

Now, the attacker can mount an attack as follows:

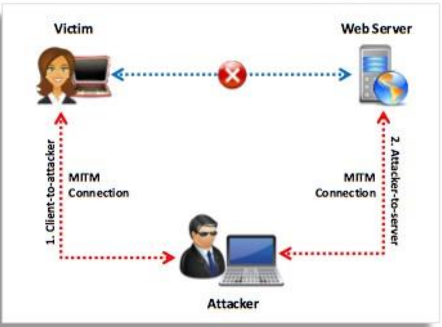
- Acquires the current session ID and connects to the web application
- Implements brute-force technique or calculates the next session ID
- Modifies the current value in the cookie/URL/hidden form-field and assumes the next user's identity



Session Hijacking
Application Level Session Hijacking

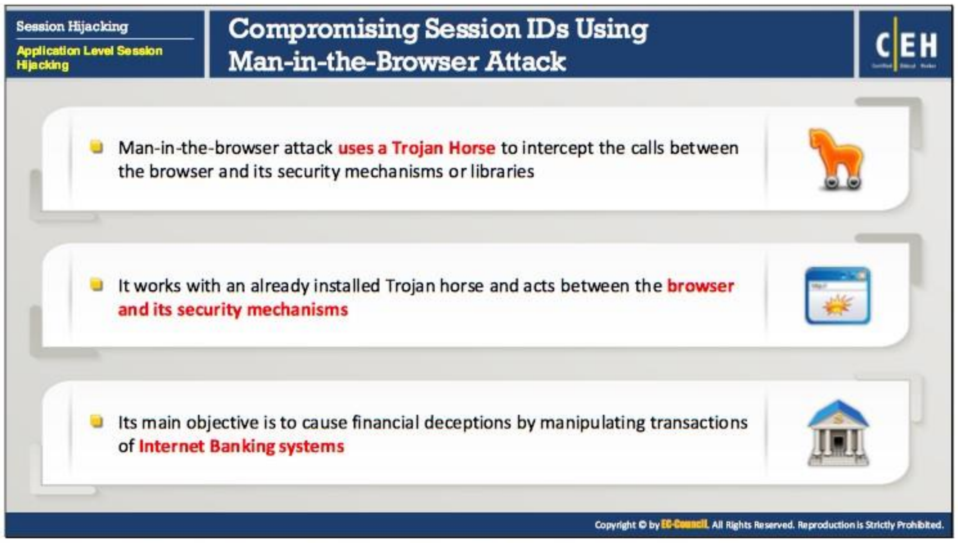
Compromising Session IDs Using Man-in-the-Middle Attack

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



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 and 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.



Session Hijacking
Application Level Session Hijacking

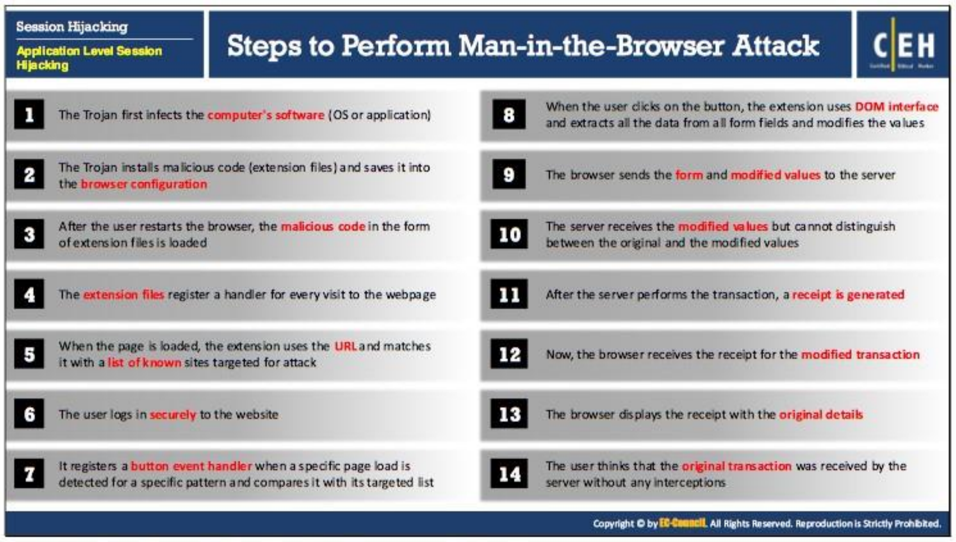
Compromising Session IDs Using Man-in-the-Browser Attack

- Man-in-the-browser attack **uses a Trojan Horse** to intercept the calls between the browser and its security mechanisms or libraries
- It works with an already installed Trojan horse and acts between the **browser and its security mechanisms**
- Its main objective is to cause financial deceptions by manipulating transactions of **Internet Banking systems**

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

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 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.



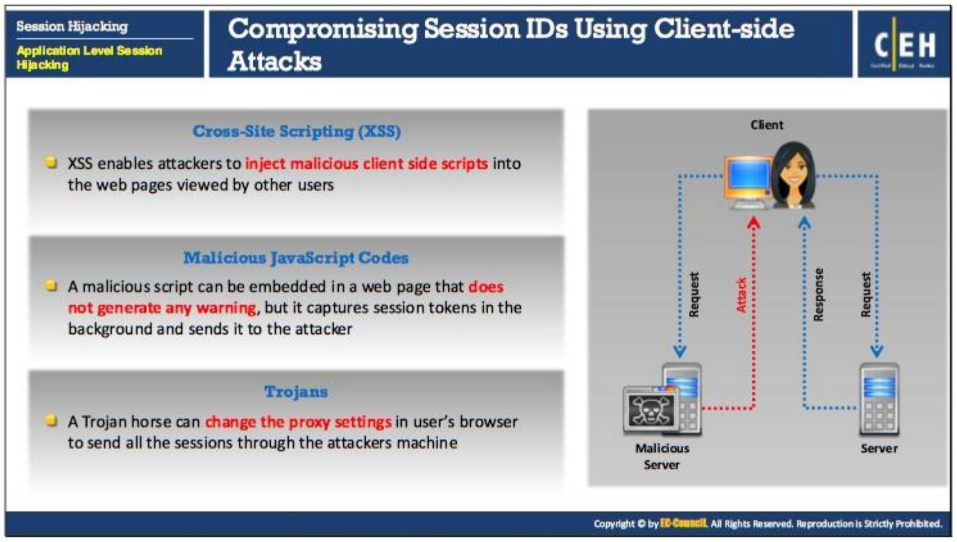
Session Hijacking
Application Level Session Hijacking

Steps to Perform Man-in-the-Browser Attack

1. The Trojan first infects the computer's software (OS or application)
2. The Trojan installs malicious code (extension files) and saves it into the browser configuration
3. After the user restarts the browser, the malicious code in the form of extension files is loaded
4. The extension files register a handler for every visit to the webpage
5. When the page is loaded, the extension uses the URL and matches it with a list of known sites targeted for attack
6. The user logs in securely to the website
7. It registers a button event handler when a specific page load is detected for a specific pattern and compares it with its targeted list
8. 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
9. 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

Steps to Perform Man-in-the-Browser Attack

- The Trojan first infects the computer's software (OS or application)
- The Trojan installs malicious code (extension files) and saves it into the browser configuration
- After the user restarts the browser, the malicious code in the form of extension files is loaded
- The extension files register a handler for every visit to the webpage
- When the page is loaded, the extension uses the URL and matches it with a list of known sites targeted for attack
- The user logs in securely to the website
- It registers a button event handler when a specific page load is detected for a specific pattern and compares it with its targeted list
- 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
- The browser sends the form and modified values to the server
- The server receives the modified values but cannot distinguish between the original and the modified values
- After the server performs the transaction, a receipt is generated
- Now, the browser receives the receipt for the modified transaction
- The browser displays the receipt with the original details
- The user thinks that the original transaction was received by the server without any interceptions



Session Hijacking
Application Level Session Hijacking

Compromising Session IDs Using Client-side Attacks

Cross-Site Scripting (XSS)

- XSS enables attackers to inject malicious client side scripts into the web pages viewed by other users

Malicious JavaScript Codes

- A malicious script can be embedded in a web page that does not generate any warning, but it captures session tokens in the background and sends it to the attacker

Trojans

- A Trojan horse can change the proxy settings in user's browser to send all the sessions through the attackers machine

Compromising Session IDs Using Client-side Attacks

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 otherwise tricks the user into visiting a malicious website. 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.

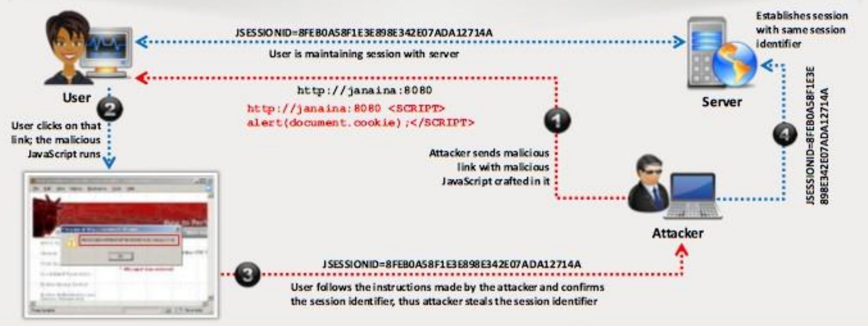
The following client-side attacks can be used to compromise Session IDs:

- **Cross-Site Scripting (XSS):** XSS enables attackers to inject malicious client side scripts into the web pages viewed by other users.
- **Malicious JavaScript Codes:** A malicious script can be embedded in a web page that does not generate any warning but it captures session tokens in the background and send it to the attacker.
- **Trojans:** A Trojan horse can change the proxy settings in user's browser to send all the sessions through the attackers machine.

Session Hijacking
Application Level Session Hijacking

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

- If an attacker sends a **crafted link** to the victim with the **malicious JavaScript**, the JavaScript will run and complete the instructions made by the attacker when the victim clicks on the link


Establishes session with same session identifier
JSESSIONID=8FEB0A58F1E3E898E342E07ADA12714A
User is maintaining session with server
User
Server
http://janaina:8080
http://janaina:8080 <SCRIPT> alert(document.cookie);</SCRIPT>
2
1
4
User clicks on that link; the malicious JavaScript runs
Attacker sends malicious link with malicious JavaScript crafted in it
JSESSIONID=8FEB0A58F1E3E898E342E07ADA12714A
Attacker
3
JSESSIONID=8FEB0A58F1E3E898E342E07ADA12714A
User follows the instructions made by the attacker and confirms the session identifier, thus attacker steals the session identifier

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 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 above diagram, 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 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>
```

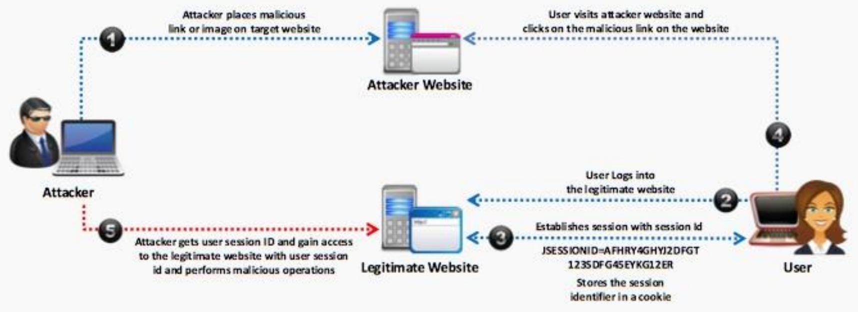
Thereafter, an attacker uses the stolen session ID to establish a valid session with the server.

Session Hijacking
Application Level Session Hijacking

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

CEH

- Cross-Site Request Forgery (CSRF) attack **exploits a victim's active session** with a trusted site in order to perform malicious activities


1
Attacker places malicious link or image on target website
Attacker Website
User visits attacker website and clicks on the malicious link on the website
4
Attacker
User Logs into the legitimate website
2
5
Attacker gets user session ID and gain access to the legitimate website with user session id and performs malicious operations
Legitimate Website
3
Establishes session with session id
JSESSIONID=AFHRY4GHYJ2DFGT 123SDFG45EYKG12ER
Stores the session identifier in a cookie
User

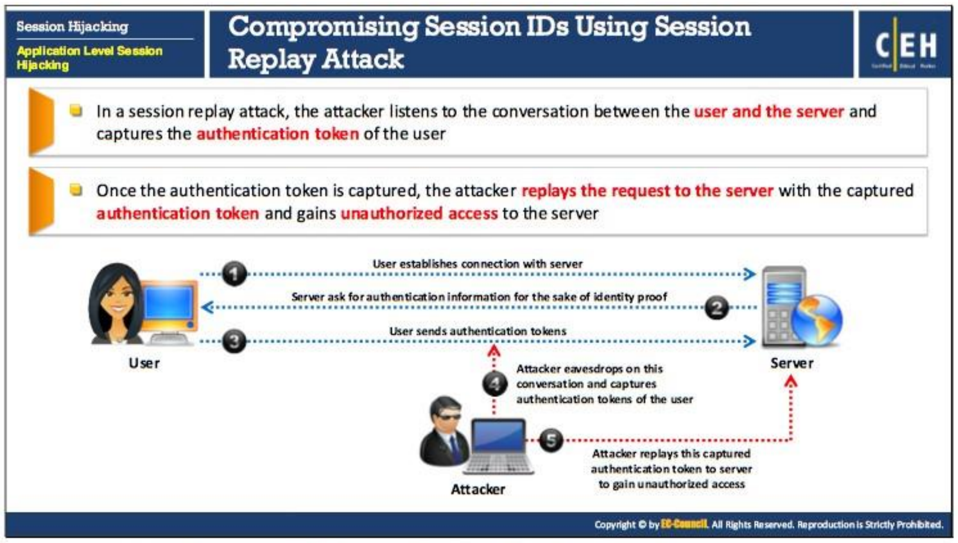
Compromising Session IDs Using Client-side Attacks: Cross-site Request Forgery Attack

Cross-site Request Forgery (CSRF), also known as a one-click attack or session riding, 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.



Session Hijacking
Application Level Session Hijacking

Compromising Session IDs Using Session Replay Attack

- In a session replay attack, the attacker listens to the conversation between the **user and the server** and captures the **authentication token** of the user
- Once the authentication token is captured, the attacker **replays the request to the server** with the captured **authentication token** and gains **unauthorized access** to the server

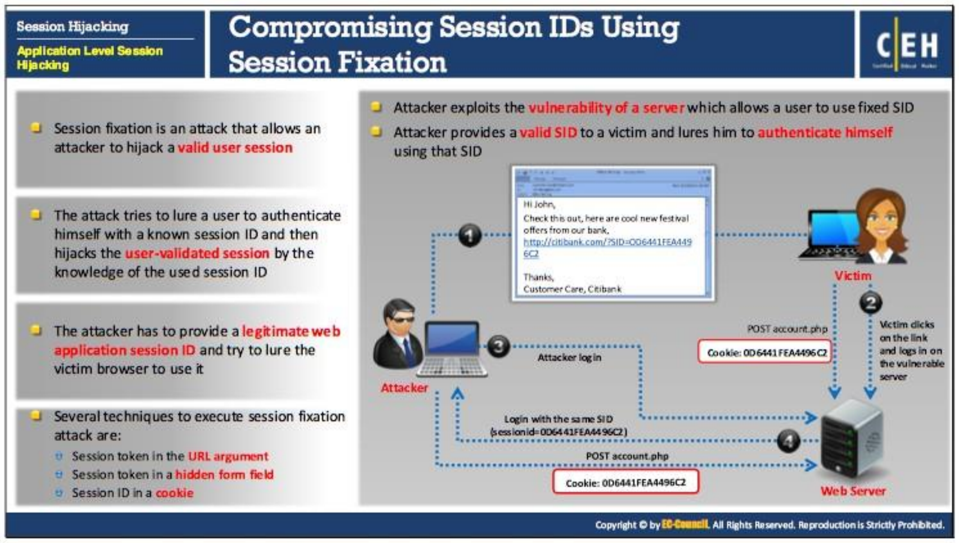
1 User establishes connection with server
2 Server ask for authentication information for the sake of identity proof
3 User sends authentication tokens
User
Server
4 Attacker eavesdrops on this conversation and captures authentication tokens of the user
5 Attacker replays this captured authentication token to server to gain unauthorized access
Attacker

Compromising Session IDs Using Session Replay Attack

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

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.

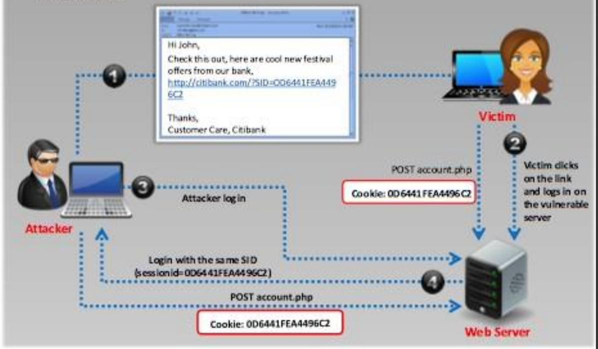


Session Hijacking
Application Level Session Hijacking

Compromising Session IDs Using Session Fixation

- Session fixation is an attack that allows an attacker to hijack a **valid user session**
- The attack tries to lure a user to authenticate himself with a known session ID and then hijacks the **user-validated session** by the knowledge of the used session ID
- The attacker has to provide a **legitimate web application session ID** and try to lure the victim browser to use it
- Several techniques to execute session fixation attack are:
  - Session token in the **URL argument**
  - Session token in a **hidden form field**
  - Session ID in a **cookie**

- Attacker exploits the **vulnerability of a server** which allows a user to use fixed SID
- Attacker provides a **valid SID** to a victim and lures him to **authenticate himself** using that SID



Hi John,
Check this out, here are cool new festival offers from our bank,
http://citibank.com/?SID=0D6441FEA4496C2

Thanks,
Customer Care, Citibank

Compromising Session IDs Using Session Fixation

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 time of login.

The attacker performs the session fixation attack to hijack a valid user session: the attacker takes the advantage of the limitation 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 such as;

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

The attacker has to choose which technique to use based on how the web application deals with session tokens. An attacker exploits the vulnerability of a server which allows a user to use fixed SID. Then the attacker provides a valid SID to a victim and lures him to authenticate himself using that SID.

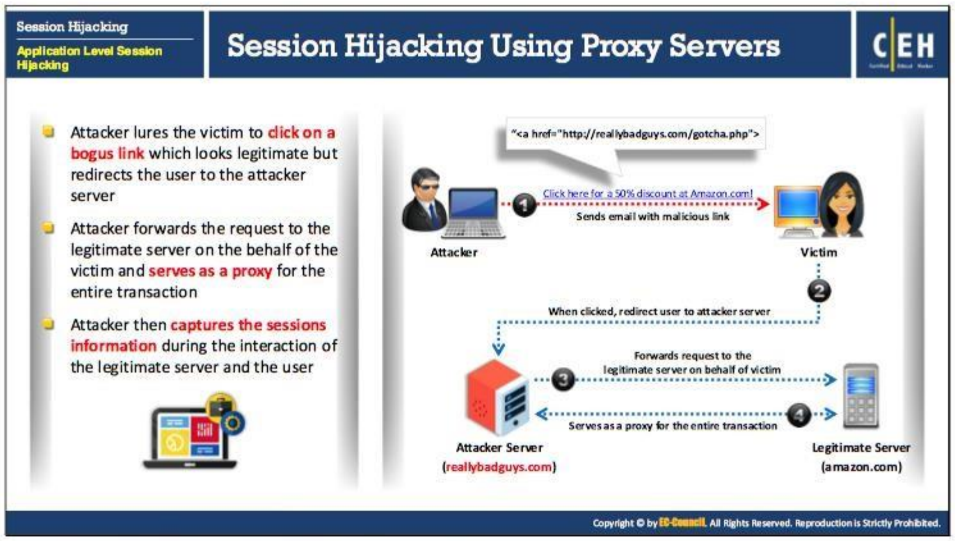
There are three phases to carry out Session fixation attack:

- **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*.

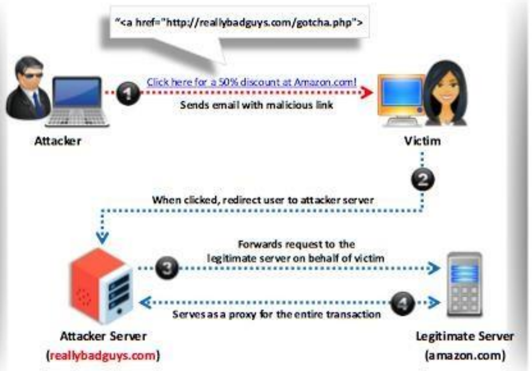
As the session ID is set by the attacker before the user logged in, we can say that user has logged into the attacker's session.



Session Hijacking
Application Level Session Hijacking

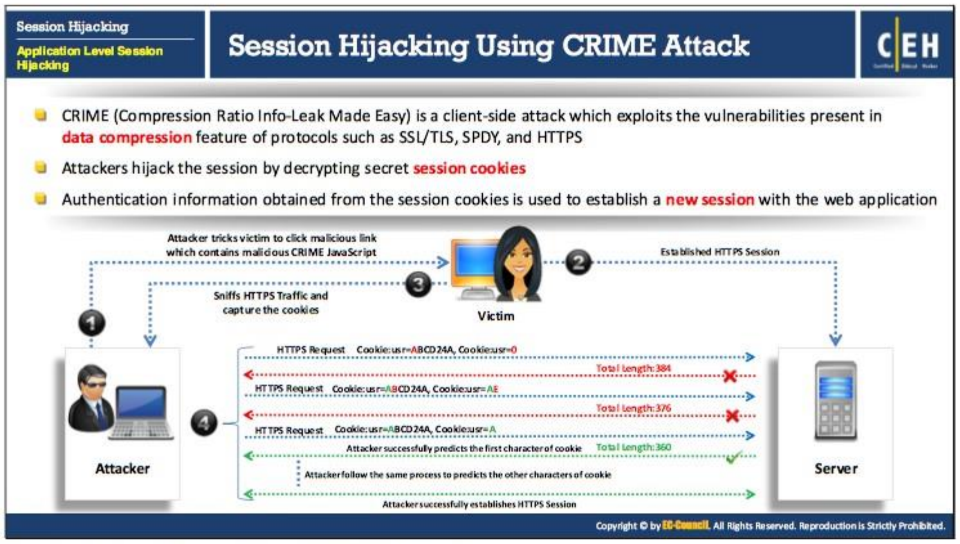
Session Hijacking Using Proxy Servers

- Attacker lures the victim to **click on a bogus link** which looks legitimate but redirects the user to the attacker server
- Attacker forwards the request to the legitimate server on the behalf of the victim and **serves as a proxy** for the entire transaction
- Attacker then **captures the sessions information** during the interaction of the legitimate server and the user


"<a href="http://reallybadguys.com/gotcha.php">
Click here for a 50% discount at Amazon.com!
Sends email with malicious link
Attacker
Victim
When clicked, redirect user to attacker server
Forwards request to the legitimate server on behalf of victim
Serves as a proxy for the entire transaction
Attacker Server (reallybadguys.com)
Legitimate Server (amazon.com)

Session Hijacking Using Proxy Servers

Attackers lure victim to click on bogus link which looks legitimate but redirect user to attacker's server. The attacker then forwards the request to the legitimate server on behalf of victim and serve as a proxy for the entire transaction. Acting as a proxy, the attacker captures the sessions information during interaction of legitimate server and user.



Session Hijacking
Application Level Session Hijacking

Session Hijacking Using CRIME Attack

- CRIME (Compression Ratio Info-Leak Made Easy) is a client-side attack which exploits the vulnerabilities present in **data compression** feature of protocols such as SSL/TLS, SPDY, and HTTPS
- Attackers hijack the session by decrypting secret **session cookies**
- Authentication information obtained from the session cookies is used to establish a **new session** with the web application

Attacker tricks victim to click malicious link which contains malicious CRIME JavaScript
Established HTTPS Session
Sniffs HTTPS Traffic and capture the cookies
Victim
1
2
3
4
HTTPS Request Cookie:usr=ABCD24A, Cookie:usr=0
Total Length:384
HTTPS Request Cookie:usr=ABCD24A, Cookie:usr=AE
Total Length:376
HTTPS Request Cookie:usr=ABCD24A, Cookie:usr=A
Attacker successfully predicts the first character of cookie
Total Length:360
Attacker follow the same process to predicts the other characters of cookie
Attacker successfully establishes HTTPS Session
Attacker
Server

Session Hijacking Using CRIME Attack

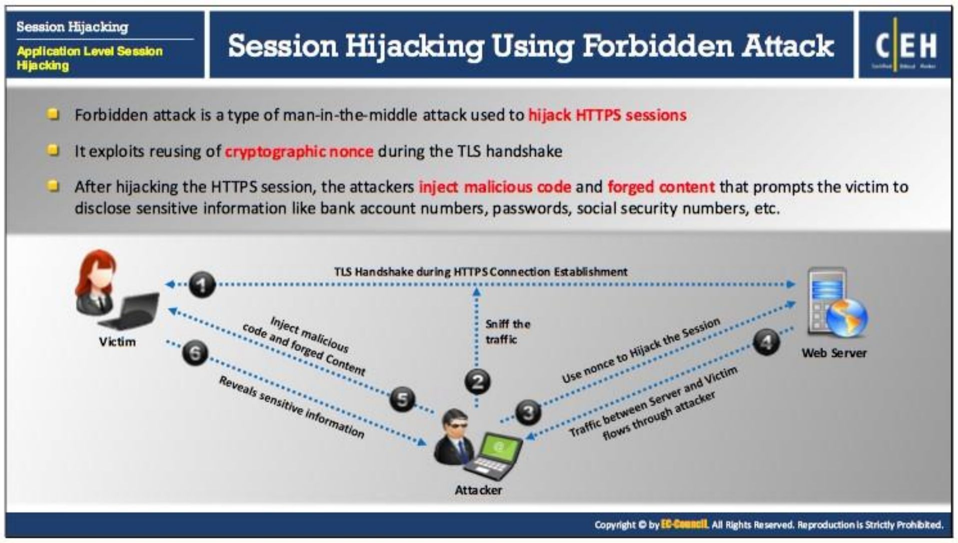
CRIME (Compression Ratio Info-Leak Made Easy) attack is a client-side attack, which exploits the vulnerabilities present in data compression feature of protocols such as SSL/TLS, SPDY, and HTTPS. The possibility of mitigation against HTTPS compression is less which makes this vulnerability even more dangerous than other compression vulnerabilities.

When two hosts on the Internet establish a connection using HTTPS protocol, a TLS session is established and the data is transmitted in encrypted form. Hence, it is difficult for an attacker to read or modify the messages between the two hosts. When a user logs on to a web application, authentication data is stored in a cookie. Whenever the browser sends any HTTPS request to the web application, the stored cookie is used for authentication. In this attack, the attacker tries to access the authentication cookie to hijack the victim's session.

In HTTPS, cookies are compressed using lossless data compression algorithm (DEFLATE) and then encrypted. Hence, it is difficult for an attacker to obtain the value of the cookie with simple sniffing.

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

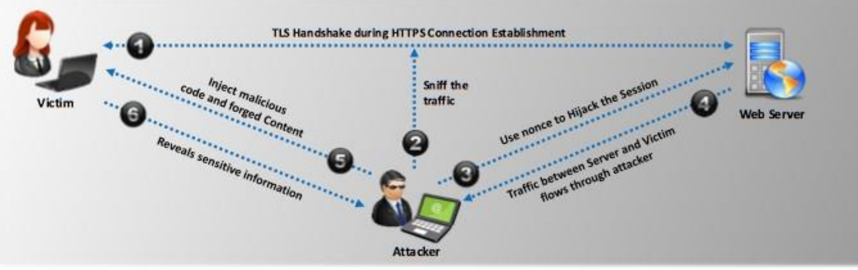
After obtaining the authentication cookie, the attacker impersonates the victim and hijacks the victim's session with the secure web application to steal confidential information like passwords, social security numbers, credit card numbers, etc. Use tools such as CrimeCheck to detect whether a web server has TLS or HTTP Compression Enabled and thus vulnerable to these attacks.



Session Hijacking
Application Level Session Hijacking

Session Hijacking Using Forbidden Attack

- Forbidden attack is a type of man-in-the-middle attack used to **hijack HTTPS sessions**
- It exploits reusing of **cryptographic nonce** during the TLS handshake
- After hijacking the HTTPS session, the attackers **inject malicious code** and **forged content** that prompts the victim to disclose sensitive information like bank account numbers, passwords, social security numbers, etc.

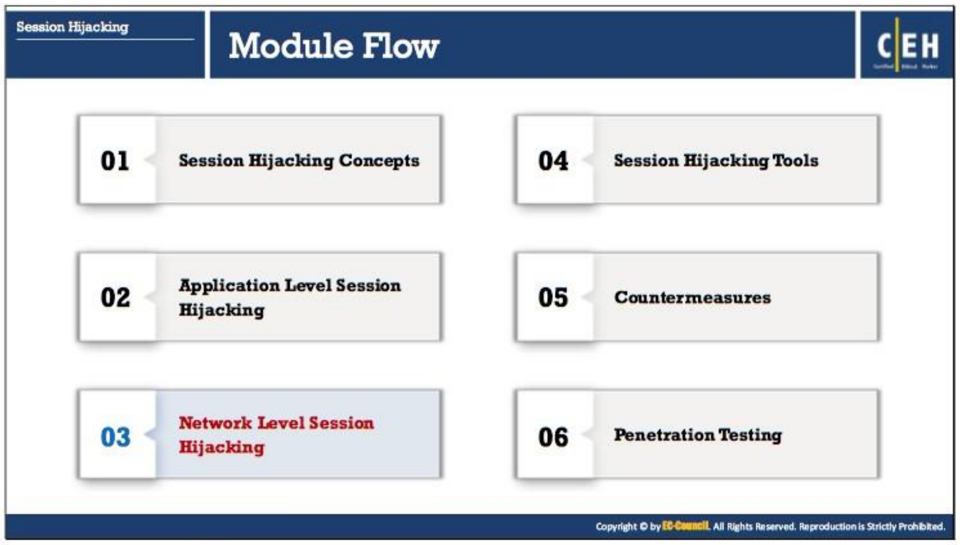

TLS Handshake during HTTPS Connection Establishment
1
Victim
Web Server
Sniff the traffic
2
Inject malicious code and forged Content
Use nonce to Hijack the Session
4
Traffic between Server and Victim flows through attacker
3
5
6
Reveals sensitive information
Attacker

Session Hijacking Using Forbidden Attack

The Forbidden attack is a type of man-in-the-middle attack. This attack is possible when a cryptographic nonce is reused while establishing a HTTPS session with the server. According to TLS specification, these arbitrary pieces of data must be used once. This attack exploits vulnerability through TLS implementation that incorrectly reuses the same nonce when data is encrypted (using AES-GCM) during the TLS handshake. Attackers exploit this vulnerability to perform man-in-the-middle 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 like JavaScript code or web fields that prompt the user to disclose passwords, social security numbers or other confidential information.

Steps involved in the Forbidden attack:

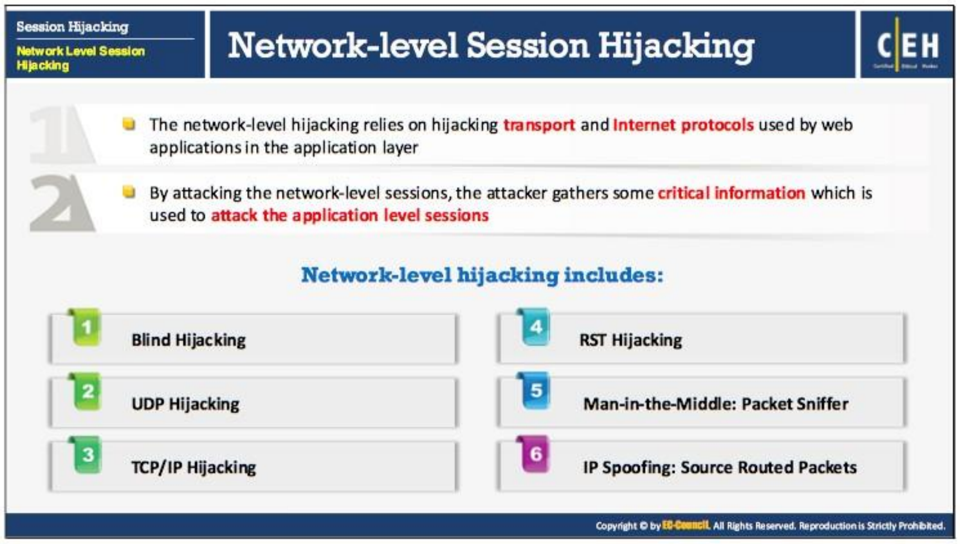
- Attacker monitors the connection between the victim and web server and sniffs the nonce from the TLS handshake messages.
- Attacker generates authentication keys using the nonce and hijacks the connection.
- All the traffic between victim and web server flows through the attacker's machine.
- Now, the attacker injects JavaScript code or web fields into the transmission towards victim.
- Victim reveals sensitive information like bank account no, passwords, social security numbers, etc. to the attacker.



Session Hijacking

Module Flow

01 Session Hijacking Concepts	04 Session Hijacking Tools
02 Application Level Session Hijacking	05 Countermeasures
03 Network Level Session Hijacking	06 Penetration Testing



Session Hijacking
Network Level Session Hijacking

Network-level Session Hijacking

1. The network-level hijacking relies on hijacking **transport** and **Internet protocols** used by web applications in the application layer
2. By attacking the network-level sessions, the attacker gathers some **critical information** which is used to **attack the application level sessions**

Network-level hijacking includes:

1. Blind Hijacking
2. UDP Hijacking
3. TCP/IP Hijacking
4. RST Hijacking
5. Man-in-the-Middle: Packet Sniffer
6. IP Spoofing: Source Routed Packets

Network Level Session Hijacking

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.

The network-level hijacking relies on hijacking transport and Internet protocols used by web applications in the application layer. By attacking the network-level sessions, the attacker gathers some critical information which is used to attack the application level sessions.

Network-level hijacking includes:

- Blind Hijacking
- UDP Hijacking
- TCP/IP Hijacking
- RST Hijacking
- Man-in-the-Middle: Packet Sniffer
- IP Spoofing: Source Routed Packets

The 3-Way Handshake

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 client side connection is in the closed state, and the server side 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 communicate.

On receiving the client's acknowledgement, the server enters the established state and sends the acknowledgment, incrementing the client's sequence number. Connection can be closed either using the FIN or RST flag or time out.

If the RST flag of a packet is set, the receiving host enters the CLOSED state and frees all resources associated with this 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 the IP address and port number is easy; they are available in the IP packets, which do not change throughout the session. After discovering the addresses communicating with the ports, the information exchanged remains the same for the rest 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 is successful in hijacking the session.

Example:

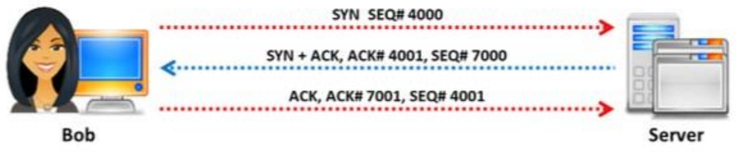


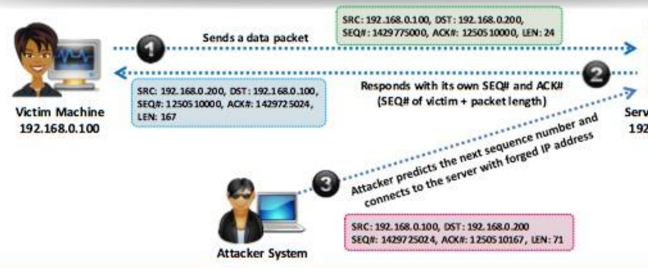
FIGURE 11.4: Three-way handshake

1. Bob initiates a connection with the server and sends a packet to the server with the SYN flag set
2. The server receives this packet and sends a packet with the SYN + ACK flag and an ISN (Initial Sequence Number) for the server
3. Bob sets the ACK flag acknowledging the receipt of the packet and increments the sequence number by 1
4. Now, the two machines successfully established a session

If the attacker can anticipate the next sequence number and ACK number that Bob will send, he/she will spoof Bob's address and start a communication with the server.

TCP/IP Hijacking

- TCP/IP hijacking uses **spoofed packets** to take over a connection between a victim and a target machine
- The victim's connection hangs, and the attacker is then able to **communicate with the host's machine** as if the attacker is the 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


Victim Machine
192.168.0.100
1
Sends a data packet
SRC: 192.168.0.100, DST: 192.168.0.200, SEQ#: 1429775000, ACK#: 1250510000, LEN: 24
Server Machine
192.168.0.200
2
Responds with its own SEQ# and ACK# (SEQ# of victim + packet length)
SRC: 192.168.0.200, DST: 192.168.0.100, SEQ#: 1250510000, ACK#: 1429725024, LEN: 167
3
Attacker predicts the next sequence number and connects to the server with forged IP address
SRC: 192.168.0.100, DST: 192.168.0.200 SEQ#: 1429725024, ACK#: 1250510167, LEN: 71
Attacker System

TCP/IP Hijacking

In TCP/IP hijacking, an attacker intercepts an established connection between two communicating parties using spoofed packets, and then pretends to be one of them. In this approach, the attacker uses spoofed packets to redirect the TCP traffic to his/her 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 victim and attacker must be on the same network. 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.

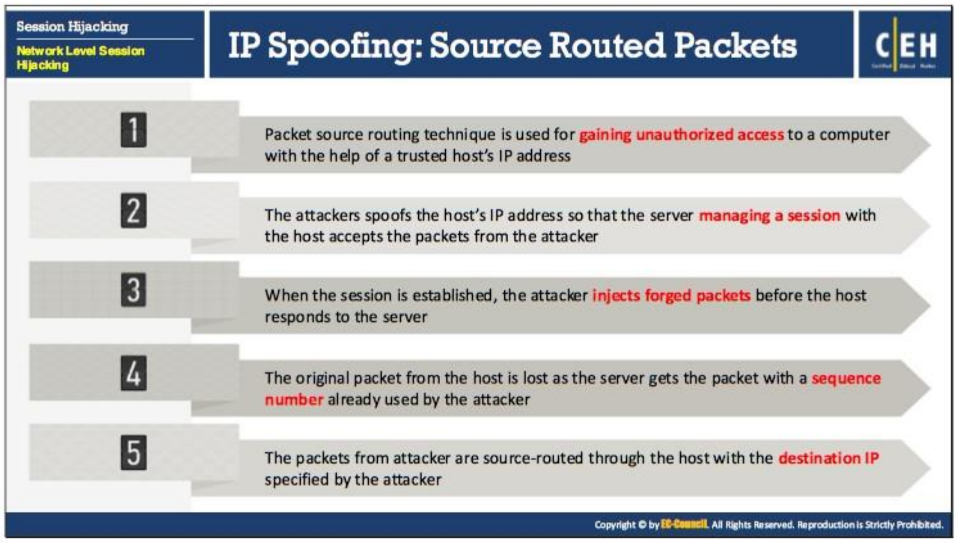
In the diagram above:

- 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.

Steps involved in TCP/IP hijacking:

- The attacker sniffs the victim's connection and uses the victim's IP to send a spoofed packet with the predicted sequence number
- The receiver processes the spoofed packet, increments the sequence number, and sends acknowledgement to the victim's IP

- The victim machine is unaware of the spoofed packet, so it ignores the receiver machine's ACK packet and turns sequence number count off
- Therefore, the receiver receives packets with the incorrect sequence number
- The attacker forces the victim's connection with the receiver machine to a desynchronized state
- The attacker tracks sequence numbers and continuously spoofs packets that comes from the victim's IP
- The attacker continues to communicate with the receiver machine while the victim's connection hangs



Session Hijacking
Network Level Session Hijacking

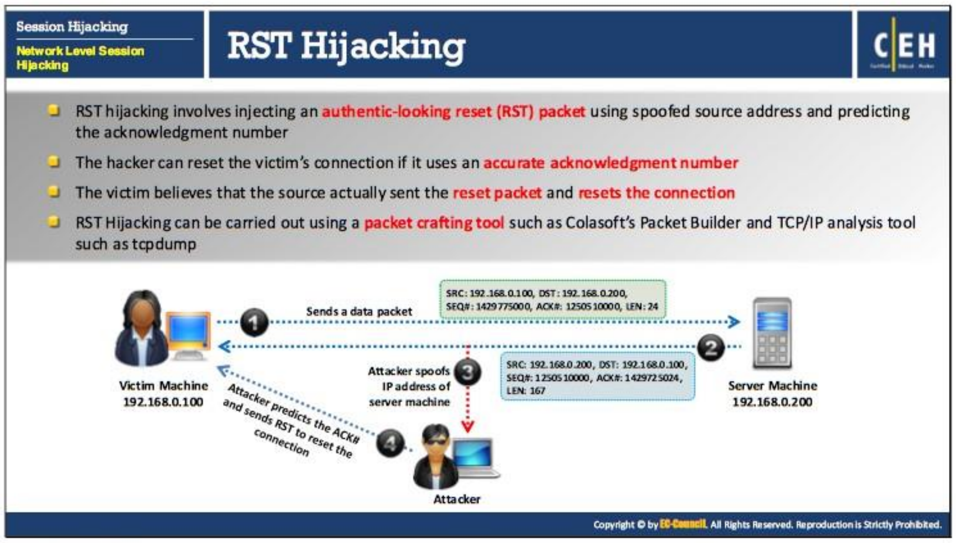
IP Spoofing: Source Routed Packets

1. Packet source routing technique is used for **gaining unauthorized access** to a computer with the help of a trusted host's IP address
2. The attackers spoofs the host's IP address so that the server **managing a session** with the host accepts the packets from the attacker
3. When the session is established, the attacker **injects forged packets** before the host responds to the server
4. The original packet from the host is lost as the server gets the packet with a **sequence number** already used by the attacker
5. The packets from attacker are source-routed through the host with the **destination IP** specified by the attacker

IP Spoofing: Source Routed Packets

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 so that the server managing a session with the host, accepts the packets from the attacker. 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 and the acknowledgment numbers. 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 between 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.



Session Hijacking
Network Level Session Hijacking
CEH

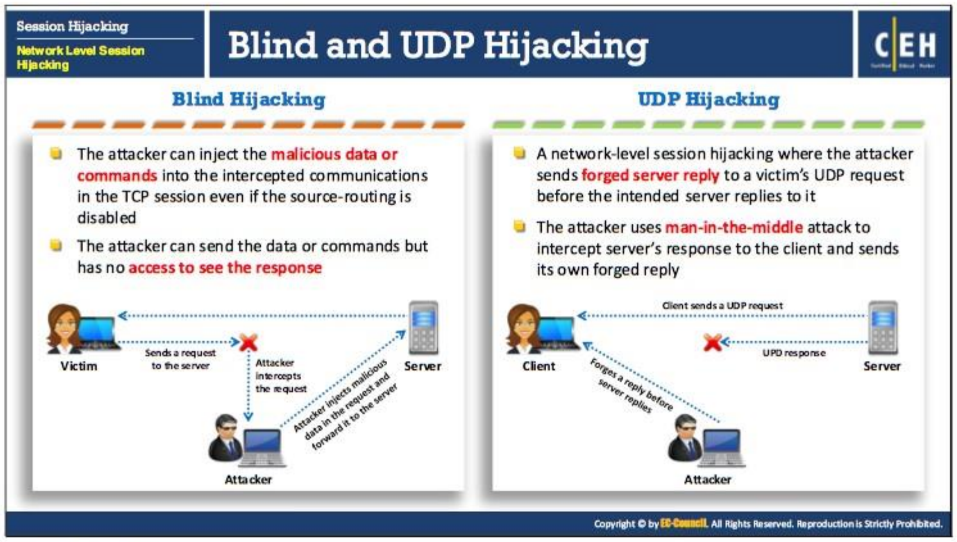
RST Hijacking

- 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

1 Sends a data packet
SRC: 192.168.0.100, DST: 192.168.0.200, SEQ#: 1429775000, ACK#: 1250510000, LEN: 24
2
SRC: 192.168.0.200, DST: 192.168.0.100, SEQ#: 1250510000, ACK#: 1429725024, LEN: 167
3 Attacker spoofs IP address of server machine
4 Attacker predicts the ACK# and sends RST to reset the connection
Victim Machine 192.168.0.100
Server Machine 192.168.0.200
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 has 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.



Session Hijacking
Network Level Session Hijacking

Blind and UDP Hijacking

Blind Hijacking

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

UDP Hijacking

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

Victim
Sends a request to the server
Attacker intercepts the request
Server
Attacker injects malicious data in the request and forward it to the server
Attacker
Client
Client sends a UDP request
UPD response
Server
Forges a reply before server replies
Attacker

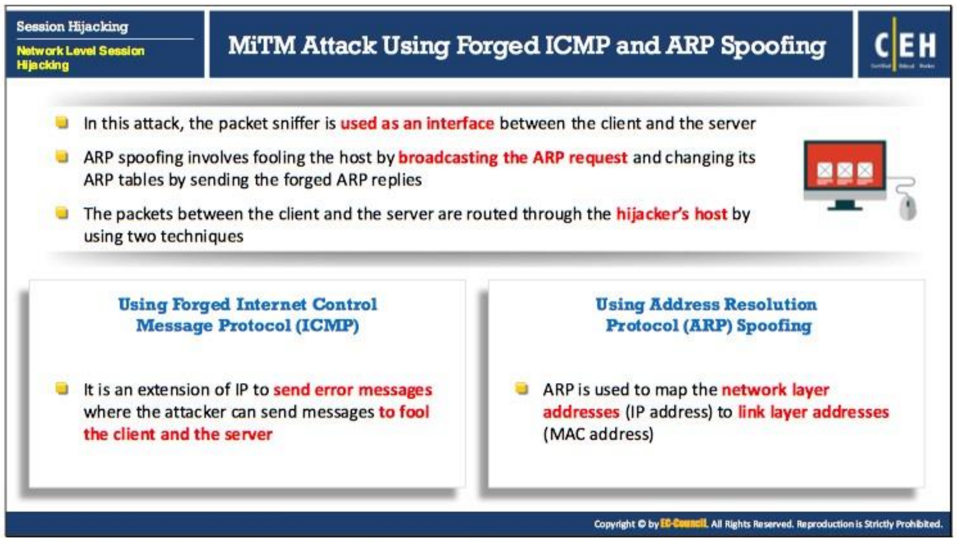
Blind Hijacking

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 password setting to allow access from another location on the network, but the attacker can never see the response. To be able to see the response, a man-in-the-middle attack works much better.

UDP Hijacking

UDP does not use packet sequencing and synchronizing, so an attacker can 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. A network-level session hijacking where 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.



Session Hijacking
Network Level Session Hijacking

MiTM Attack Using Forged ICMP and ARP Spoofing

- In this attack, the packet sniffer is **used as an interface** between the client and the server
- ARP spoofing involves fooling the host by **broadcasting the ARP request** and changing its ARP tables by sending the forged ARP replies
- The packets between the client and the server are routed through the **hijacker's host** by using two techniques

Using Forged Internet Control Message Protocol (ICMP)

- It is an extension of IP to **send error messages** where the attacker can send messages **to fool the client and the server**

Using Address Resolution Protocol (ARP) Spoofing

- ARP is used to map the **network layer addresses** (IP address) to **link layer addresses** (MAC address)

MiTM Attack Using Forged ICMP and ARP Spoofing

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 packets between the client and the server are routed through the hijacker's host by using two techniques.

- **Using Forged Internet Control Message Protocol (ICMP)**

  ICMP (Internet Control Message Protocol) is an extension of IP to send error messages and an attacker can use it send messages to fool the client and the server. 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 indicating problems in processing packets through the original connection. This fools the server and client into routing through hijacker's path instead.

- **Using Address Resolution Protocol (ARP) Spoofing**

  Another technique used is ARP (Address Resolution Protocol) spoofing. Hosts use 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 the forged ARP replies. The attacker sends forged ARP replies that update the ARP tables of 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 existing between the client and server through his/her machine.

Session Hijacking

Module Flow

CEH

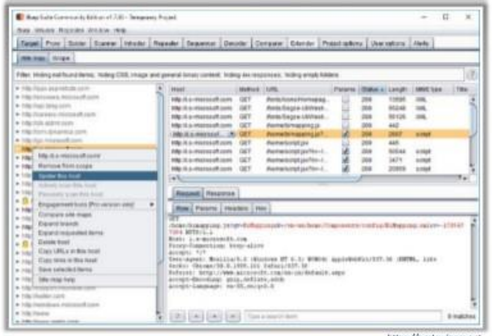
Session Hijacking
Session Hijacking Tools

Session Hijacking Tools

CEH

Burp Suite

Burp suite allows the attacker to inspect and modify traffic between the browser and the target application



https://portswigger.net

- **OWASP ZAP** *https://www.owasp.org*
- **BetterCAP** *https://www.bettercap.org*
- **netool toolkit** *https://sourceforge.net*
- **WebSploit Framework** *https://sourceforge.net*
- **sslstrip** *https://pypi.python.org*

Session Hijacking Tools

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

Burp Suite

Source: *https://portswigger.net*

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 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
- The **CSRF PoC Generator function**, for generating proof-of-concept of cross-site request forgery (CSRF) attack for a given request

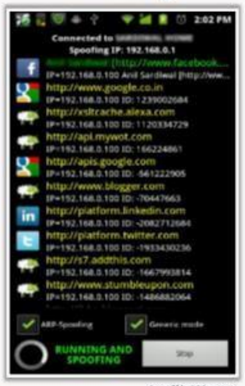
Some of the session hijacking tools are listed below:

- OWASP ZAP (*https://www.owasp.org*)
- BetterCAP (*https://www.bettercap.org*)
- netool toolkit (*https://sourceforge.net*)
- WebSploit Framework (*https://sourceforge.net*)
- sslstrip (*https://pypi.python.org*)
- JHijack (*https://sourceforge.net*)
- Ettercap (*http://ettercap.github.io*)
- Cookie Cadger (*https://www.cookiecadger.com*)
- CookieCatcher (*https://github.com*)
- hamster (*https://github.com*)

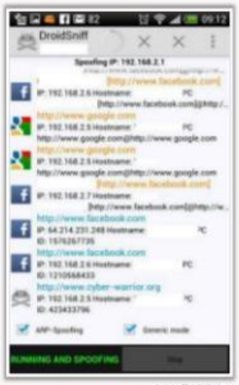
Session Hijacking
Session Hijacking Tools

Session Hijacking Tools for Mobile

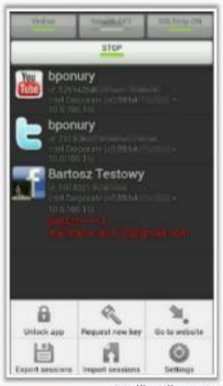
DroidSheep


http://droidsheep.org

DroidSniff


https://github.com

FaceNiff


http://faceniff.ponury.net

Session Hijacking Tools for Mobile

- **DroidSheep**

  Source: *http://droidsheep.org*

  DroidSheep tool is a used for session hijacking on Android devices connected on common wireless network. It gets the session ID of active user on Wi-Fi network and uses it to access the website as an authorized user. The droidsheep user can easily see what the authorized user is doing or seeing on the website. It can also hijack the social account by obtaining the session ID.

- **DroidSniff**

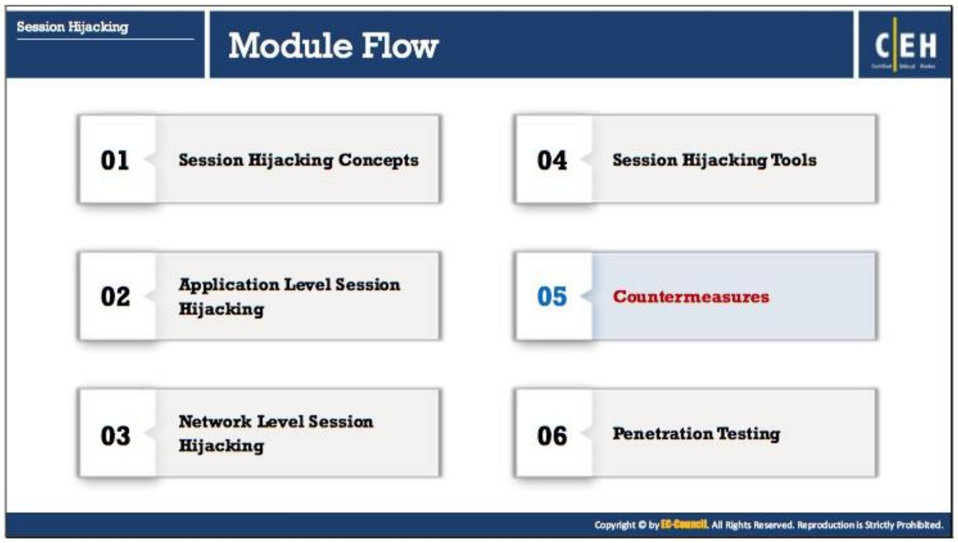
  Source: *https://github.com*

  DroidSniff is an Android app for Security analysis in wireless networks and capturing Facebook, Twitter, LinkedIn and other accounts. This tool is used for testing the security of user accounts. It identifies the poor security properties of network connections without encryption.

- **FaceNiff**

  Source: *http://faceniff.ponury.net*

  FaceNiff is an Android app that allows you to sniff and intercept web session profiles over the WiFi that your mobile is connected to. It is possible to hijack sessions only when WiFi is not using EAP, but it will work on any private network (Open/WEP/WPA-PSK/WPA2-PSK).



Session Hijacking

Module Flow

01 Session Hijacking Concepts

02 Application Level Session Hijacking

03 Network Level Session Hijacking

04 Session Hijacking Tools

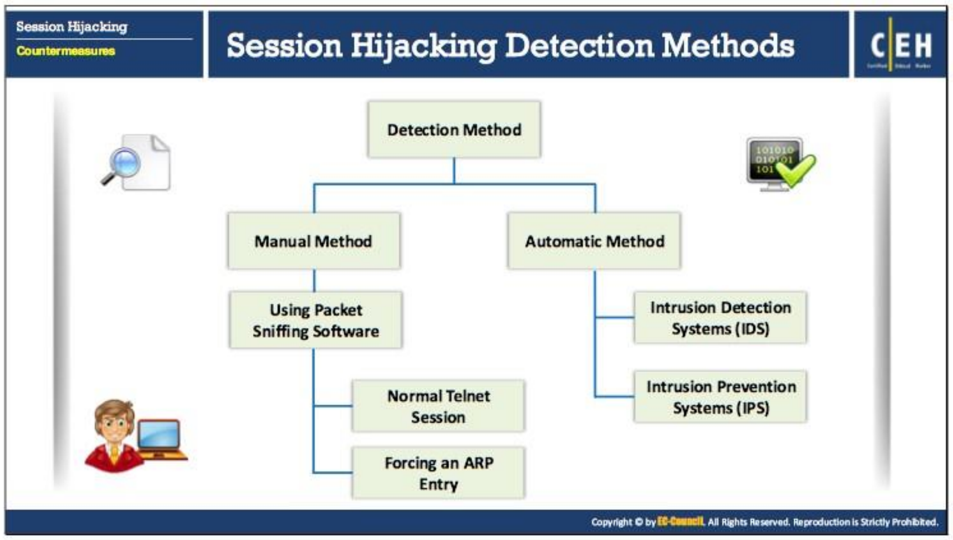
05 Countermeasures

06 Penetration Testing

Countermeasures

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

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


Session Hijacking
Countermeasures
Session Hijacking Detection Methods
CEH
Detection Method
Manual Method
Automatic Method
Using Packet Sniffing Software
Normal Telnet Session
Forcing an ARP Entry
Intrusion Detection Systems (IDS)
Intrusion Prevention Systems (IPS)
Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Session Hijacking Detection Methods

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 Wireshark and SteelCentral Packet Analyzer 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 the session is established, the communication between client and server takes place via Telnet protocol. A "normal" Telnet session means a client has send packets to the server, and the server acknowledges the packet.

- **Forcing an ARP Entry**

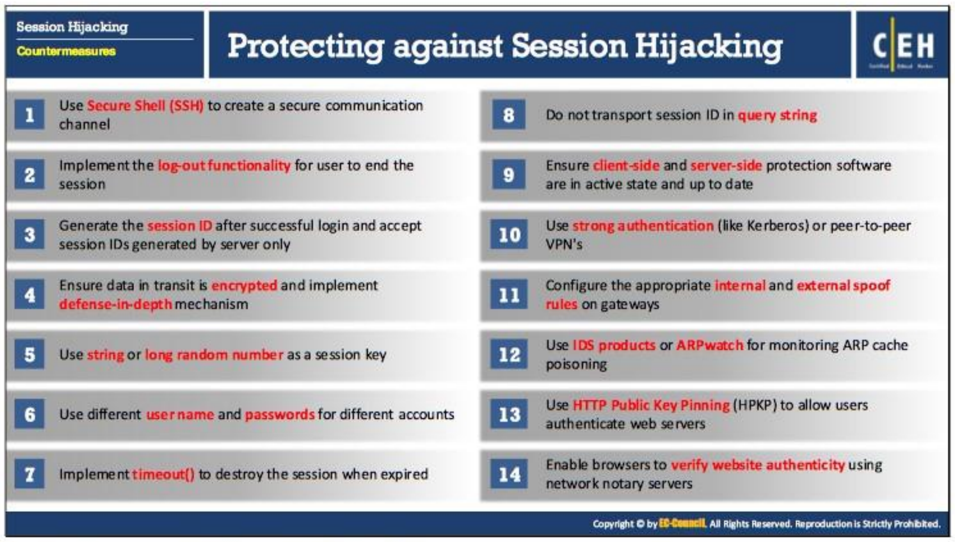
  Forcing ARP entry involves replacing the MAC address of a compromised machine residing in the ARP cache of the server with a different one, the goal being to restrict network traffic to the compromised machine.

Look for:

- 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.



Session Hijacking
Countermeasures

Protecting against Session Hijacking

CEH

1 Use Secure Shell (SSH) to create a secure communication channel

2 Implement the log-out functionality for user to end the session

3 Generate the session ID after successful login and accept session IDs generated by server only

4 Ensure data in transit is encrypted and implement defense-in-depth mechanism

5 Use string or long random number as a session key

6 Use different user name and passwords for different accounts

7 Implement timeout() to destroy the session when expired

8 Do not transport session ID in query string

9 Ensure client-side and server-side protection software are in active state and up to date

10 Use strong authentication (like Kerberos) or peer-to-peer VPN's

11 Configure the appropriate internal and external spoof rules on gateways

12 Use IDS products or ARPwatch for monitoring ARP cache poisoning

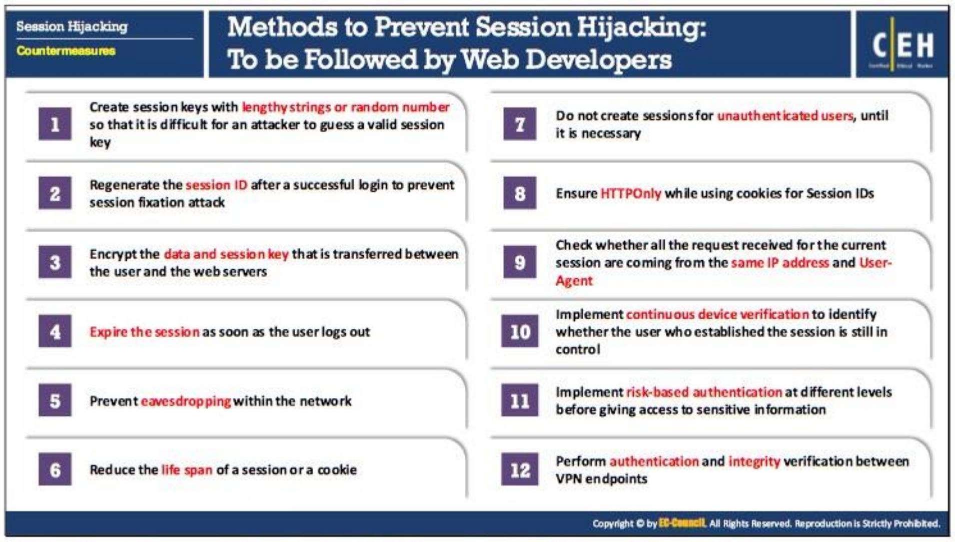
13 Use HTTP Public Key Pinning (HPKP) to allow users authenticate web servers

14 Enable browsers to verify website authenticity using network notary servers

Protecting against Session Hijacking

- Use Secure Shell (SSH) to create a secure communication channel.
- Pass the authentication cookies over HTTPS connection.
- Implement the log-out functionality for user to end the session.
- Generate the session ID after successful login and accept session IDs generated by server only.
- Ensure data in transit is encrypted and implement defense-in-depth mechanism.
- Use string or long random number as a session key.
- Use different user name and passwords for different accounts.
- Educate the employees and minimize remote access.
- Implement timeout () to destroy the session when expired.
- Avoid including the session ID in the URL or query string.
- Use switches rather than hubs and limit incoming connections.
- Ensure client-side and server-side protection software are in active state and up to date.
- Use strong authentication (like Kerberos) or peer-to-peer VPN's.
- Configure the appropriate internal and external spoof rules on gateways.
- Use IDS products or ARPwatch for monitoring ARP cache poisoning.
- Use encrypted protocols that available at OpenSSH suite.
- 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).
- Use IPsec to encrypt session information.
- Use HTTP Public Key Pinning (HPKP) to allow users authenticate web servers.
- Enable browsers to verify website authenticity using network notary servers.
- Implement DNS-based authentication of named entities.
- Disable compression mechanism of HTTP requests.
- Use cipher-chaining block (CBC) ciphers incorporating random padding up to 255 bytes. Thus, making extraction of confidential information difficult for an attacker.
- Restrict the cross site scripts known as Cross Site Request Forgery (CSRF) from the client side.
- Upgrade web browsers to the latest versions.
- Use vulnerability scanners like masscan to detect any insecure configuration of HTTPS session settings on sites.



Session Hijacking
Countermeasures

Methods to Prevent Session Hijacking: To be Followed by Web Developers

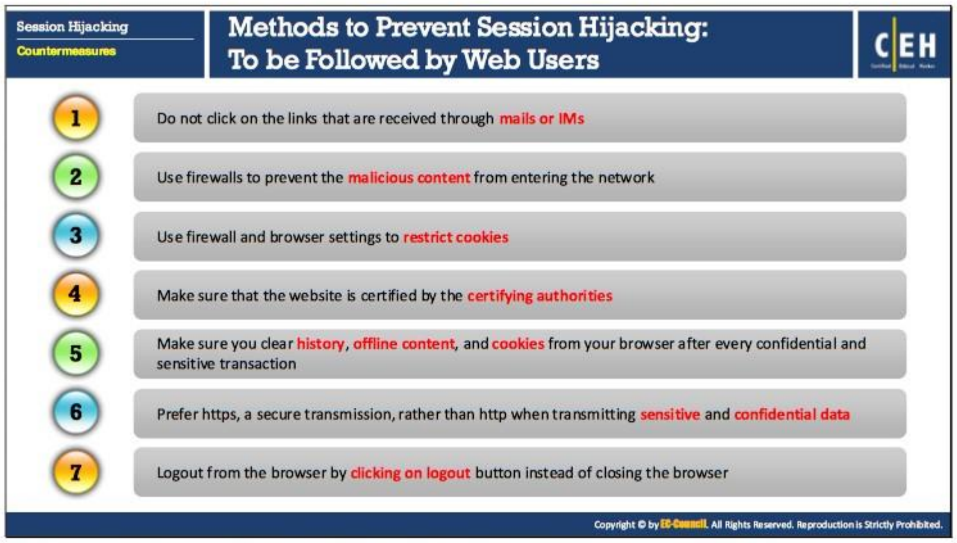
1. Create session keys with lengthy strings or random number so that it is difficult for an attacker to guess a valid session key
2. Regenerate the session ID after a successful login to prevent session fixation attack
3. Encrypt the data and session key that is transferred between the user and the web servers
4. Expire the session as soon as the user logs out
5. Prevent eavesdropping within the network
6. Reduce the life span of a session or a cookie
7. Do not create sessions for unauthenticated users, until it is necessary
8. Ensure HTTPOnly while using cookies for Session IDs
9. Check whether all the request received for the current session are coming from the same IP address and User-Agent
10. Implement continuous device verification to identify whether the user who established the session is still in control
11. Implement risk-based authentication at different levels before giving access to sensitive information
12. Perform authentication and integrity verification between VPN endpoints

Methods to Prevent Session Hijacking: To be Followed by Web Developers

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 should consider the guidelines given below to minimize/eliminate the risk of session hijacking:

- Create session keys with lengthy strings or random number so that it is difficult for an attacker to guess a valid session key.
- Regenerate the session ID after a successful login to prevent session fixation attack.
- Encrypt the data and session key that is transferred between the user and the web servers.
- Implement Secure Sockets Layer (SSL) to encrypt all the information in transit via the network.
- Expire the session as soon as the user logs out.
- Prevent eavesdropping within the network.
- Reduce the life span of a session or a cookie.
- 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.
- Do not create sessions for unauthenticated users, until it is necessary.
- Ensure HTTPOnly, while using cookies for Session IDs.
- Check whether all the request received for the current session are coming from the same IP address and User-Agent.

- Implement continuous device verification to identify whether the user who established the session is still in control.
- Implement risk-based authentication at different levels before giving access to sensitive information.
- Perform authentication and integrity verification between VPN endpoints.



Session Hijacking
Countermeasures

Methods to Prevent Session Hijacking: To be Followed by Web Users

1. Do not click on the links that are received through **mails or IMs**
2. Use firewalls to prevent the **malicious content** from entering the network
3. Use firewall and browser settings to **restrict cookies**
4. Make sure that the website is certified by the **certifying authorities**
5. Make sure you clear **history**, **offline content**, and **cookies** from your browser after every confidential and sensitive transaction
6. Prefer https, a secure transmission, rather than http when transmitting **sensitive** and **confidential data**
7. Logout from the browser by **clicking on logout** button instead of closing the browser

Methods to Prevent Session Hijacking: To be Followed by Web Users

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

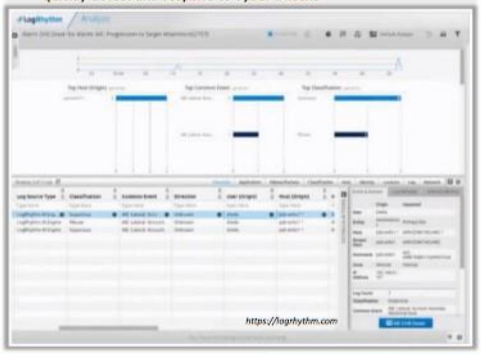
- Do not click on the links received through mails or IMs.
- Use firewalls to prevent the malicious content from entering the network.
- Use firewall and browser settings to restrict cookies.
- Ensure website is certified by the certifying authorities.
- Ensure you clear history, offline content, and cookies from your browser after every confidential and sensitive transaction.
- Prefer https, a secure transmission, rather than http when transmitting sensitive and confidential data.
- Logout from the browser by clicking on logout button instead of closing the browser.
- 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.

Session Hijacking
Countermeasures

Session Hijacking Detection Tools

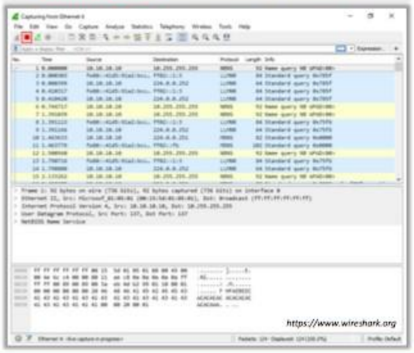
LogRhythm

- LogRhythm's **threat Lifecycle Management Platform** helps us to quickly detect and respond to cyber attacks


https://logrhythm.com

Wireshark

- Wireshark allows you to **capture and interactively browse the traffic** running on a computer network


https://www.wireshark.org

Session Hijacking Detection Tools

Session hijacking attacks are difficult to detect and in most cases attacks go unnoticed causing severe damage to confidential data. Tools such as packet sniffers, IDSs and SIEMs can be used to detect session hijacking attacks.

- **LogRhythm**

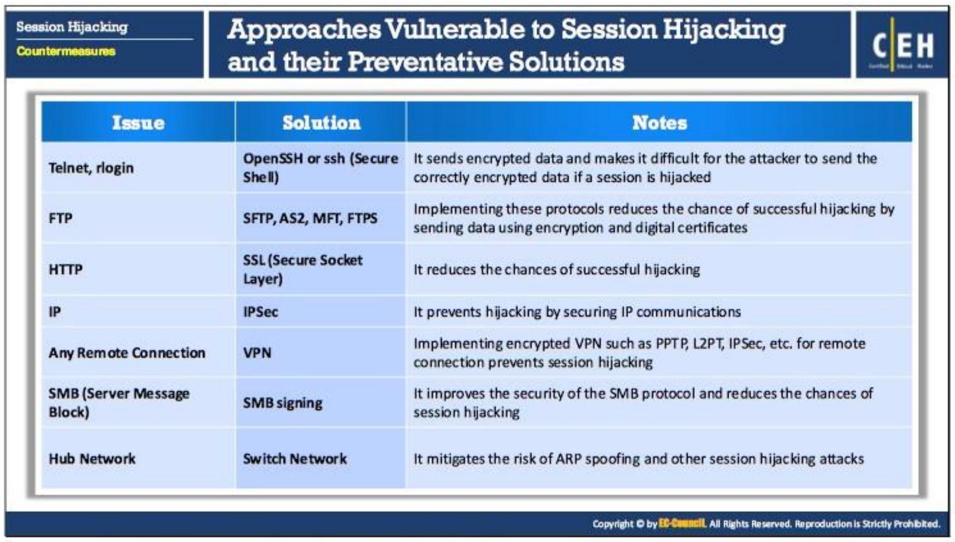
  Source: *https://logrhythm.com*

  LogRhythm's threat Lifecycle Management Platform helps in quickly detecting and responding to cyber attacks. LogRhythm includes SIEM, log management, network and endpoint monitoring, user and entity behavior analytics (UEBA), and security automation and orchestration (SAO). LogRhythm's Advanced Intelligence Engine can be used to detect session hijacking attacks.

Wireshark

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

Wireshark allows to capture and interactively browse the traffic running on a computer network. This tool uses Winpcap to capture packets, so it can only capture the packets on the networks supported by Winpcap. It captures live network traffic from Ethernet, IEEE 802.11, PPP/HDLC, ATM, Bluetooth, USB, Token Ring, Frame Relay, FDDI networks. Wireshark can be used to monitor and detect session hijacking attempts.



Session Hijacking
Countermeasures

Approaches Vulnerable to Session Hijacking and their Preventative Solutions

Issue	Solution	Notes
Telnet, rlogin	OpenSSH or ssh (Secure Shell)	It sends encrypted data and makes it difficult for the attacker to send the correctly encrypted data if a session is hijacked
FTP	SFTP, AS2, MFT, FTPS	Implementing these protocols reduces the chance of successful hijacking by sending data using encryption and digital certificates
HTTP	SSL (Secure Socket Layer)	It reduces the chances of successful hijacking
IP	IPSec	It prevents hijacking by securing IP communications
Any Remote Connection	VPN	Implementing encrypted VPN such as PPTP, L2PT, IPSec, etc. for remote connection prevents session hijacking
SMB (Server Message Block)	SMB signing	It improves the security of the SMB protocol and reduces the chances of session hijacking
Hub Network	Switch Network	It mitigates the risk of ARP spoofing and other session hijacking attacks

Approaches Vulnerable to Session Hijacking and their Preventative Solutions

Implementing encryption and signing protocols prevents attackers from hijacking the session. Shown in the table below are various issues and their respective solutions that on implementation prevent or impede taking over the valid session.

Issue	Solution	Notes
Telnet, rlogin	OpenSSH or ssh (Secure Shell)	It sends encrypted data and makes it difficult for the attacker to send the correctly encrypted data if a session is hijacked
FTP	SFTP, AS2, MFT, FTPS	Implementing these protocols reduces the chance of successful hijacking by sending data using encryption and digital certificates
HTTP	SSL (Secure Socket Layer)	It reduces the chances of successful hijacking
IP	IPSec	It prevents hijacking by securing IP communications
Any Remote Connection	VPN	Implementing encrypted VPN such as PPTP, L2PT, IPSec, etc. for remote connection prevents session hijacking
SMB (Server Message Block)	SMB signing	It improves the security of the SMB protocol and reduces the chances of session hijacking
Hub Network	Switch Network	It mitigates the risk of ARP spoofing and other session hijacking attacks

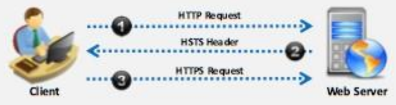
TABLE 11.1: Approaches vulnerable to session hijacking and their preventative solutions

Session Hijacking
Countermeasures

Approaches to Prevent Session Hijacking

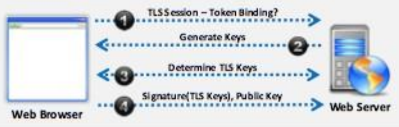
HTTP Strict Transport Security (HSTS)

- HTTP Strict Transport Security (HSTS) is a **web security policy** that protects HTTPS websites against MITM attacks
- It allows web servers to **enforce web browsers** to interact with it using secure HTTPS protocol



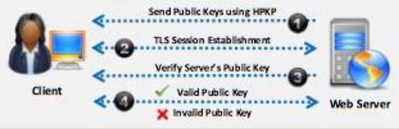
Token Binding

- When a user logs on to a web application, it generates a cookie with a **session identifier**, called **token**
- Token binding **protects client server communication** against session hijacking attacks


Web Browser
1 TLS Session – Token Binding?
2 Generate Keys
3 Determine TLS Keys
4 Signature(TLS Keys), Public Key
Web Server

HTTP Public Key Pinning (HPKP)

- HPKP is **Trust on First Use** (TOFU) technique used in an HTTP header
- HPKP allows a web client to **associate a specific public key certificate** with a particular server to minimize the risk of MITM attacks


Client
1 Send Public Keys using HPKP
2 TLS Session Establishment
3 Verify Server's Public Key
4 ✓ Valid Public Key
✗ Invalid Public Key
Web Server

Approaches to Prevent Session Hijacking

- **HTTP Strict Transport Security (HSTS)**

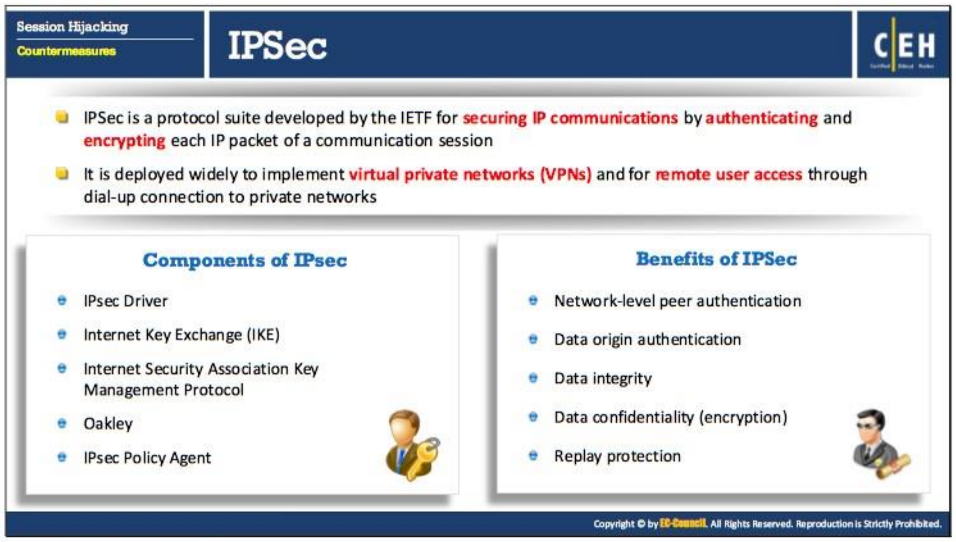
  HTTP Strict Transport Security (HSTS) is a web security policy that protects HTTPS websites against man-in-the-middle attacks. HSTS policy helps web servers to enforce web browsers to interact with it using secure HTTPS protocol. With HSTS policy, all the insecure HTTP connections are automatically converted into HTTPS connections. This policy ensures that all the communication between the web server and web browser is encrypted and all responses that are delivered and received are originated from an authenticated server.

- **Token Binding**

  When a user logs on to a web application, it generates a cookie with a session identifier, called token. The user uses 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 the valid session identifier. 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 the client connects to the server, it generates a signature using 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. This ensures the server that the message was sent by an authentic client because only the client has its private key. If an attacker is able to capture the signature, it is not possible for him to regenerate the signature or reuse it for another connection. For every new connection, a new pair of public and private keys are used.

- **HTTP Public Key Pinning (HPKP)**

  HTTP Public key Pinning (HPKP) is 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 man-in-the-middle attacks with fraudulent certificates. In TLS sessions, to verify the authenticity of a server's public key, the public key is enclosed in a X.509 digital certificate, which is signed by a Certification Authority (CA). Attackers by compromising any CA can perform man-in-the-middle attacks on various TLS sessions. HPKP protects TLS sessions from such type of attacks by delivering to the client, the list of public keys owned by a web server. When the client connects to the web server, it verifies the server's certificate in the certificate chain obtained using HPKP. If the server sends any unidentified public key, the client issues a warning message to the user.



Session Hijacking
Countermeasures

IPSec

- IPSec is a protocol suite developed by the IETF for **securing IP communications** by **authenticating** and **encrypting** each IP packet of a communication session
- It is deployed widely to implement **virtual private networks (VPNs)** and for **remote user access** through dial-up connection to private networks

Components of IPsec

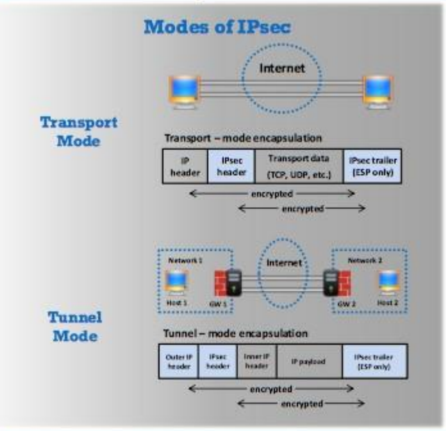
- IPsec Driver
- Internet Key Exchange (IKE)
- Internet Security Association Key Management Protocol
- Oakley
- IPsec Policy Agent

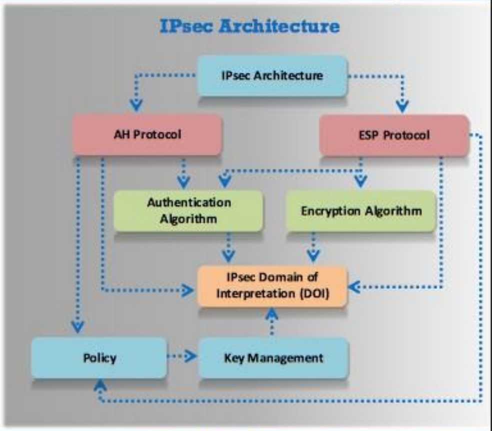
Benefits of IPSec

- Network-level peer authentication
- Data origin authentication
- Data integrity
- Data confidentiality (encryption)
- Replay protection

Session Hijacking
Countermeasures

IPSec (Cont'd)





IPSec

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
- Data integrity
- Limited traffic flow confidentiality
- Network-level peer authentication
- Replay protection

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

Components of IPsec

- **IPsec driver:** A software, that performs protocol-level functions required to encrypt and decrypt the packets.
- **Internet Key Exchange (IKE):** IPsec protocol that produces security keys for IPsec and other protocols.
- **Internet Security Association Key Management Protocol;** Software that allows two computers to communicate by encrypting the data exchanged between them.
- **Oakley:** A protocol, which uses the Diffie-Hellman algorithm to create master key, and a key that is specific to each session in IPsec data transfer.
- **IPsec Policy Agent:** A service of the Windows 2000 collects IPsec policy settings from the active directory and sets the system configuration system at startup.

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 principles initiate a key exchange, establishing an ISAKMP SA (ISAKMP Security Association) and a shared secret key.
- Both principles 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.

Modes of IPsec

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, can be used to provide VPN services for network utilizing NAT.

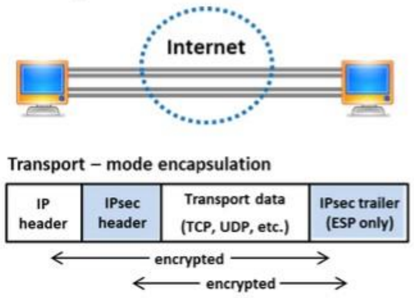


FIGURE 11.5: Transport mode encapsulation

- **Tunnel Mode**

  In tunnel mode (also AH [Authentication Header]), the IPsec encrypts both the payload and header. Hence, there is more security in tunnel mode. After receiving, the IPsec-compliant device decrypts the data. Tunnel model is used to create VPNs over the Internet for network-to-network communication, host-to-network communication and host-to-host communication. It is compatible with NAT and supports NAT traversal.

Tunnel mode is used to create virtual private networks for network-to-network communications (e.g. between routers to link sites), host-to-network communications (e.g. remote user access) and host-to-host communications (e.g. private chat).

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.

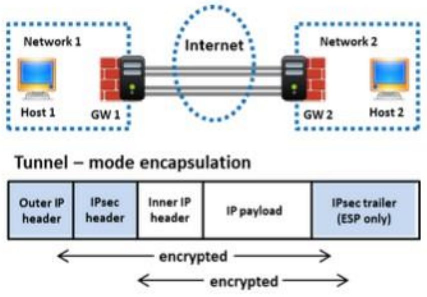


FIGURE 11.6: Tunnel mode encapsulation

IPsec Architecture

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 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

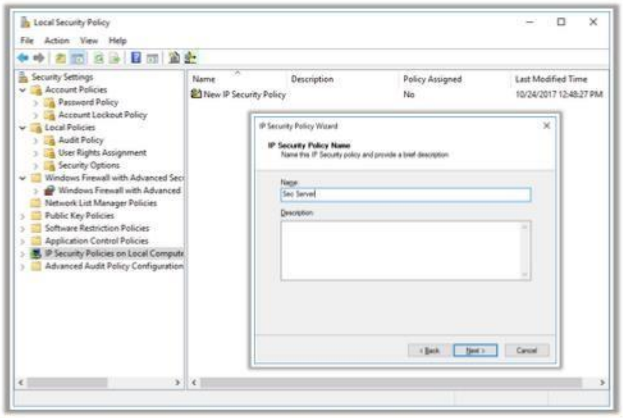
combining the security concepts of authentication, key management, and security associations.

- **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.

Session Hijacking
Countermeasures

IPsec Authentication and Confidentiality

- IPsec uses two different security services for authentication and confidentiality
  - **Authentication Header (AH)**: Provides data authentication of the sender
  - **Encapsulation Security Payload (ESP)**: Provides both data authentication and encryption (confidentiality) of the sender


Local Security Policy
File Action View Help
Security Settings
Account Policies
Password Policy
Account Lockout Policy
Local Policies
Audit Policy
User Rights Assignment
Security Options
Windows Firewall with Advanced Sec
Windows Firewall with Advanced
Network List Manager Policies
Public Key Policies
Software Restriction Policies
Application Control Policies
IP Security Policies on Local Compute
Advanced Audit Policy Configuration
< Back
Next >
Cancel

Name	Description	Policy Assigned	Last Modified Time
New IP Security Policy		No	10/24/2017 12:48:27 PM

IP Security Policy Wizard

IP Security Policy Name
Name this IP Security policy and provide a brief description

Name:
Sec Server

Description:

IPsec Authentication and Confidentiality

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 the service to protect against replays, an optional service on 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.

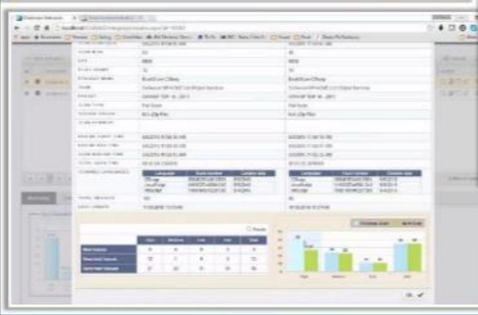
Session Hijacking
Countermeasures

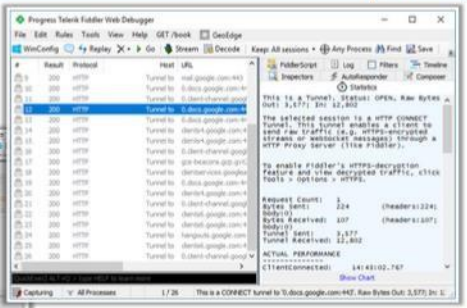
Session Hijacking Prevention Tools

CxSAST

Checkmarx CxSAST is a unique **source code analysis solution** that provides tools for identifying, tracking, and repairing technical and logical flaws in the source code

https://www.checkmarx.com





Fiddler

Fiddler is used for **security testing of web applications** such as decrypting HTTPS traffic, and manipulating requests using a man-in-the-middle decryption technique

https://www.telerik.com

Session Hijacking Prevention Tools

To prevent session hijacking security testing of web applications and analysis of static code to identify vulnerabilities in web applications are required. Identifying vulnerabilities at the early stage helps to implement security measures to protect against session hijacking attacks.

- **CxSAST**

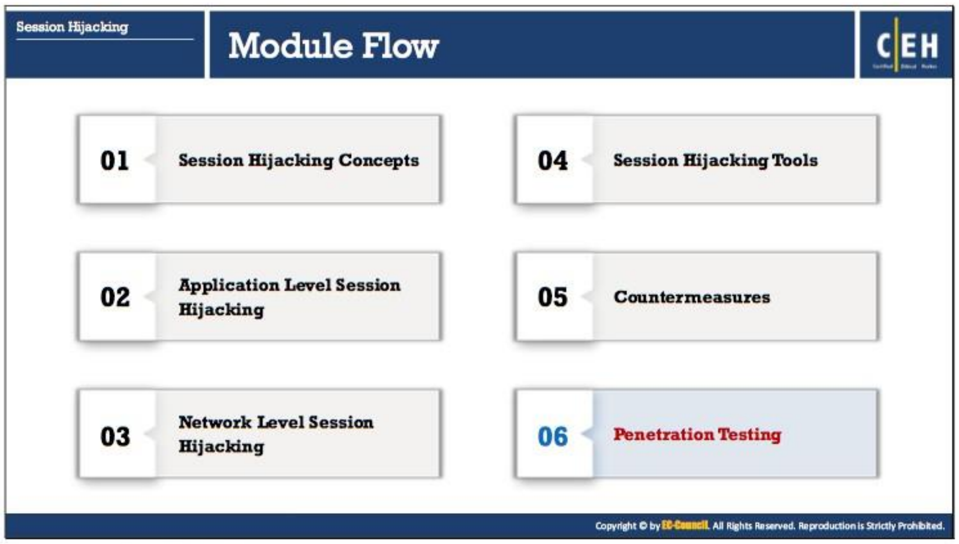
  Source: *https://www.checkmarx.com*

  Checkmarx CxSAST is a unique source code analysis solution that provides tools for identifying, tracking, and repairing technical and logical flaws in the source code, such as security vulnerabilities, compliance issues, and business logic problems. CxSAST supports Open Source Analysis (CxOSA) enabling licensing and compliance management, vulnerabilities alerts, policy enforcement and reporting. This tool supports a wide range of OS platforms, programming languages and frameworks.

- **Fiddler**

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

  Fiddler is used for security testing of web applications such as decrypting HTTPS traffic, and manipulating requests using a man-in-the-middle decryption technique. Fiddler is a web debugging proxy which logs all HTTP(s) traffic between a computer and the Internet. This tool can be used to debug traffic from PC, Mac or Linux systems and mobile devices and to manipulate and edit web sessions.



Session Hijacking

Module Flow

01 Session Hijacking Concepts

02 Application Level Session Hijacking

03 Network Level Session Hijacking

04 Session Hijacking Tools

05 Countermeasures

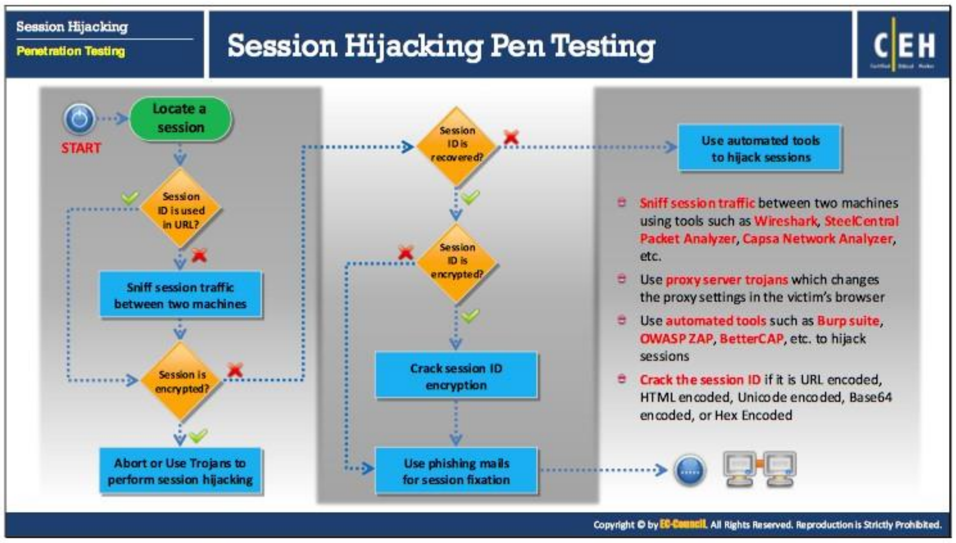
06 Penetration Testing

Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

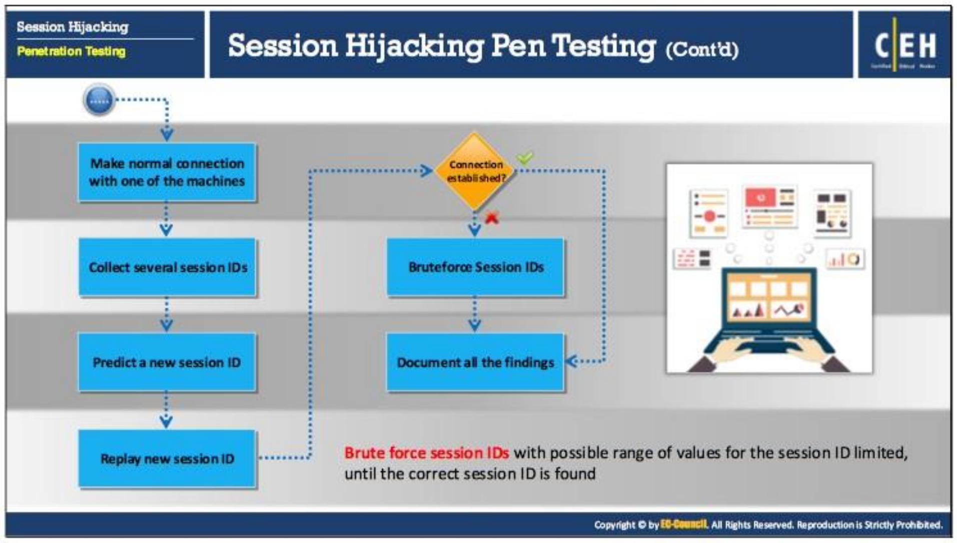
Penetration Testing

Various hijacking methods exist, using which attackers exploit design flaws inherent in the TCP/IP protocol suite to hijack 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
Penetration Testing
Session Hijacking Pen Testing
START
Locate a session
Session ID is used in URL?
Sniff session traffic between two machines
Session is encrypted?
Abort or Use Trojans to perform session hijacking
Session ID is recovered?
Session ID is encrypted?
Crack session ID encryption
Use phishing mails for session fixation
Use automated tools to hijack sessions
Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

- Sniff session traffic between two machines using tools such as Wireshark, SteelCentral Packet Analyzer, Capsa Network Analyzer, etc.
- Use proxy server trojans which changes the proxy settings in the victim's browser
- Use automated tools such as Burp suite, OWASP ZAP, BetterCAP, etc. to hijack sessions
- Crack the session ID if it is URL encoded, HTML encoded, Unicode encoded, Base64 encoded, or Hex Encoded


Session Hijacking
Penetration Testing
Session Hijacking Pen Testing (Cont'd)
Make normal connection with one of the machines
Collect several session IDs
Predict a new session ID
Replay new session ID
Connection established?
Bruteforce Session IDs
Document all the findings
Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Brute force session IDs with possible range of values for the session ID limited, until the correct session ID is found

Session Hijacking Pen Testing

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 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. After locating a session, check whether the URL uses a session ID; if it does, 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, SteelCentral Packet Analyzer, Capsa Network Analyzer, etc. 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 Burp suite, OWASP ZAP, or BetterCAP 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 inject yourself between the victim's machine and the remote machine and use this unauthorized connection for 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 masquerading 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.

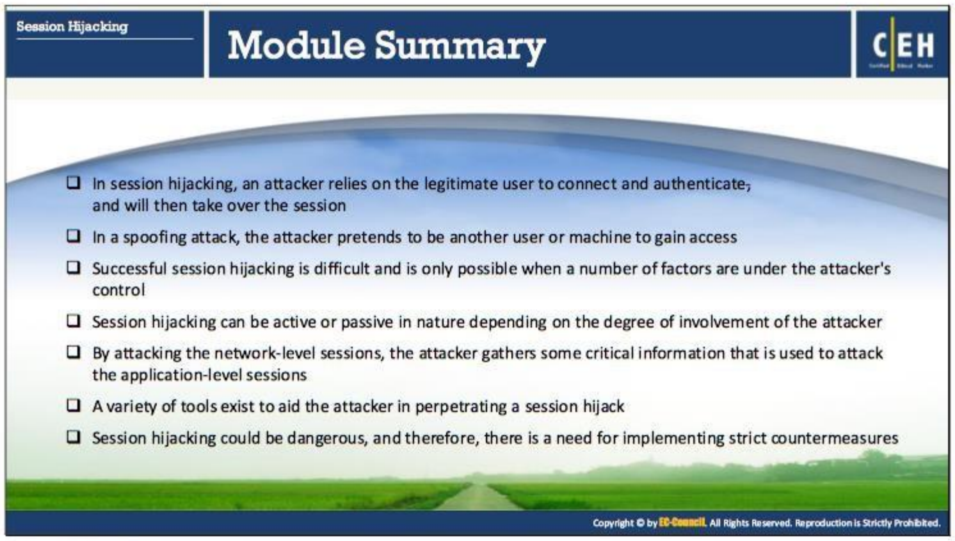
  Now, check for the establishment of connection. If the connection is established, then document all the findings of the penetration testing; otherwise, apply a brute force attack to find the current valid session ID in order to establish the connection.

- **Step 9: Brute force session IDs**

  Brute force session IDs with a possible range of values for the session ID is limited, until a correct session ID is obtained. This involves creating 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.



Session Hijacking

Module Summary

- In session hijacking, an attacker relies on the legitimate user to connect and authenticate, and will then take over the session
- In a spoofing attack, the attacker pretends to be another user or machine to gain access
- Successful session hijacking is difficult and is only possible when a number of factors are under the attacker's control
- Session hijacking can be active or passive in nature depending on the degree of involvement of the attacker
- By attacking the network-level sessions, the attacker gathers some critical information that is used to attack the application-level sessions
- A variety of tools exist to aid the attacker in perpetrating a session hijack
- Session hijacking could be dangerous, and therefore, there is a need for implementing strict countermeasures

Module Summary

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. The next module will discuss about how attackers evade network security components such as IDS, firewalls, and honeypots, as well as countermeasures to prevent it.