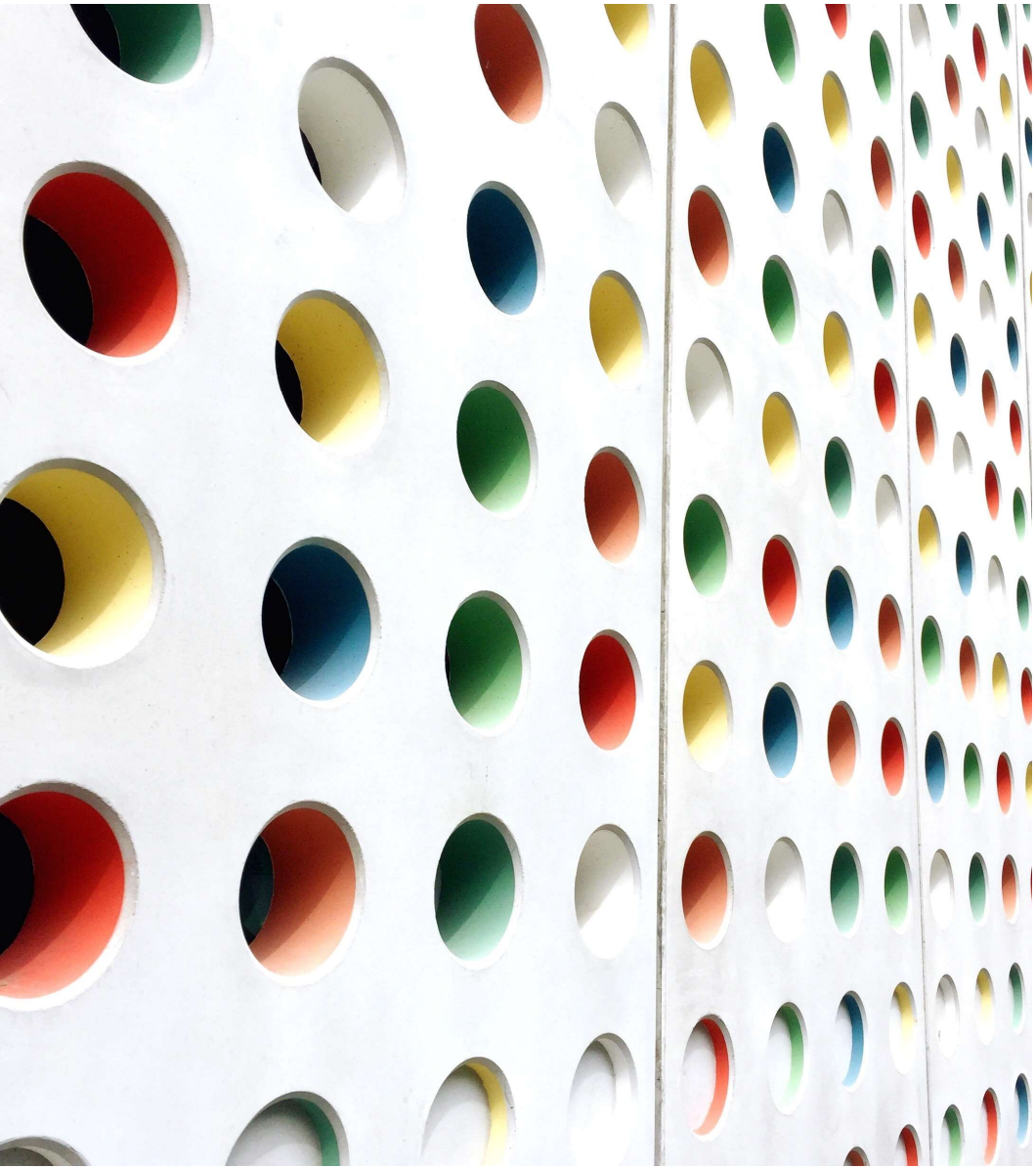


Session Hijacking

Module Objectives

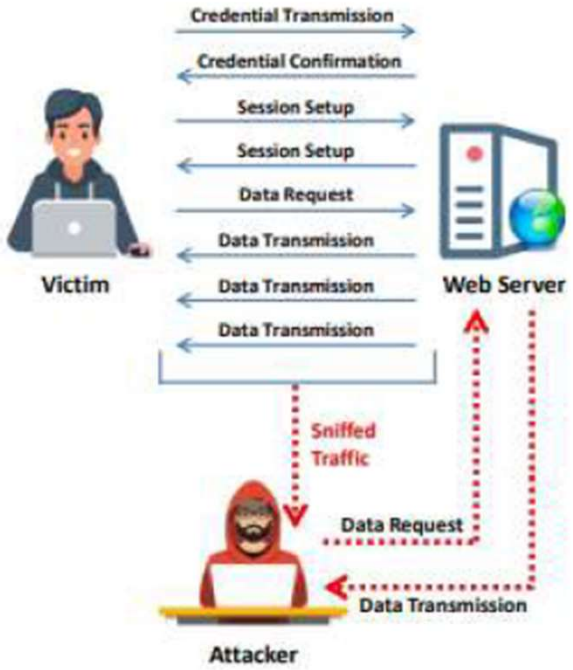
- Understanding Session Hijacking Concepts
- Understanding Application Level Session Hijacking
- Understanding Network Level Session Hijacking
- Overview of Session Hijacking Tools
- Understanding Different Session Hijacking Countermeasures

Module Flow

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

What is Session Hijacking?

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

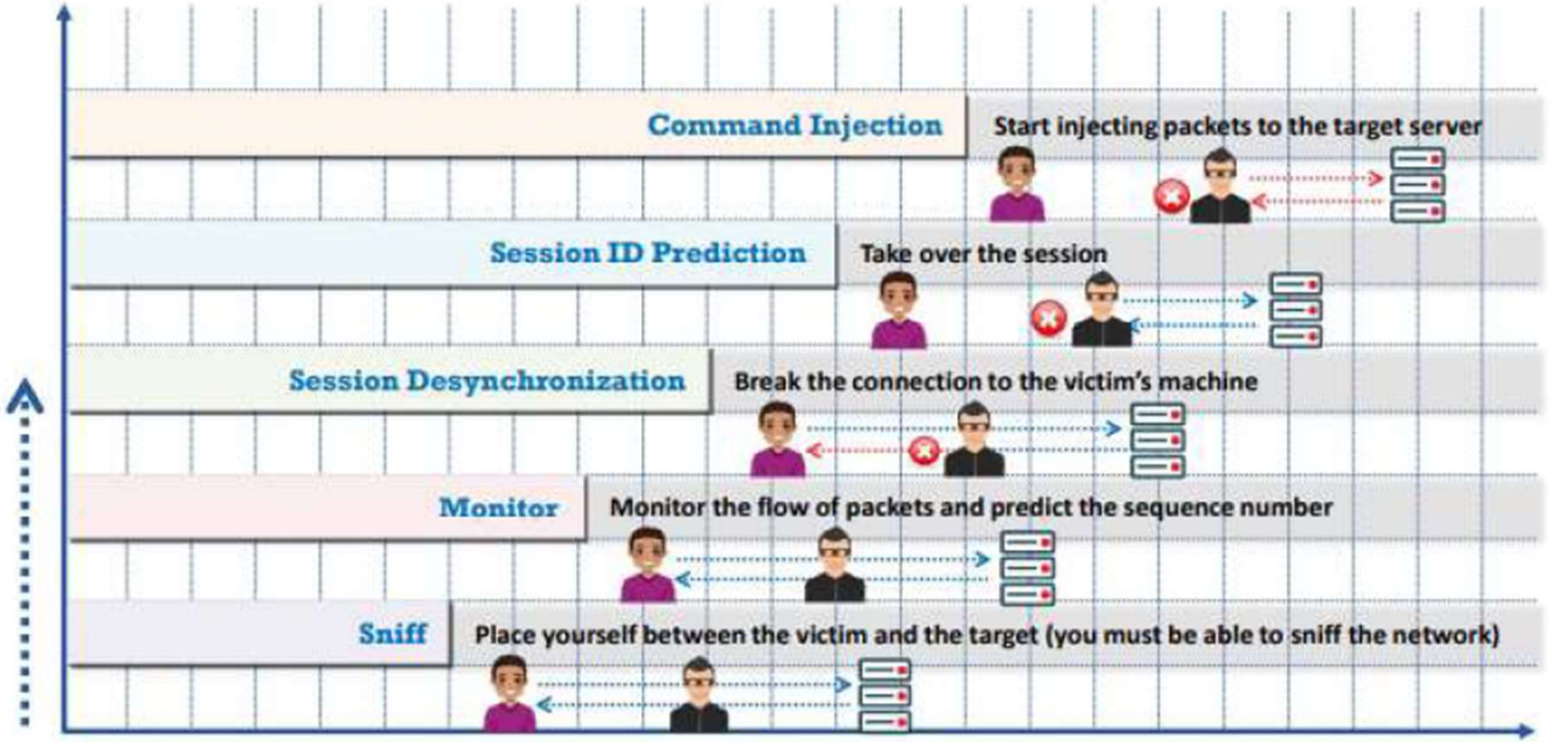


Why is Session Hijacking Successful?

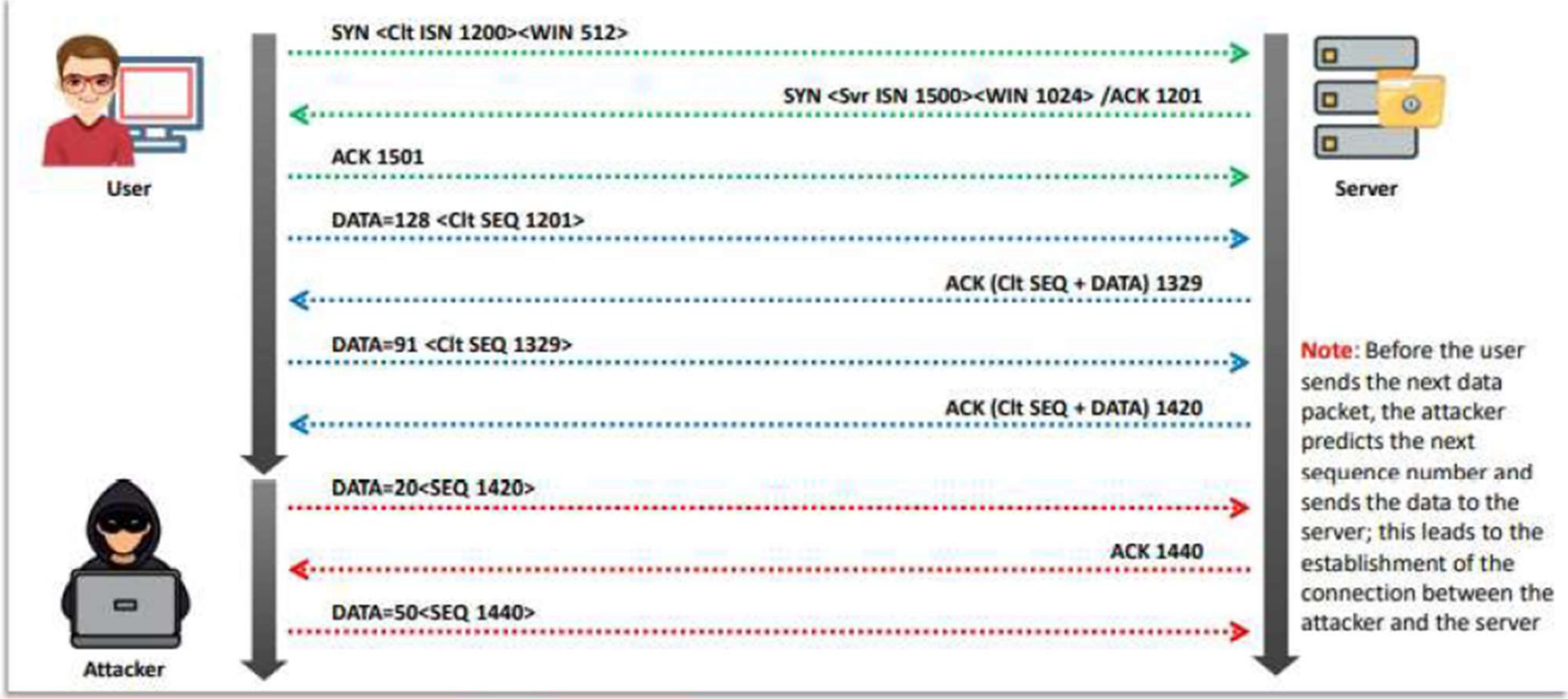
C|EH

- Absence of account lockout for invalid session IDs
- Indefinite session timeout
- Weak session-ID generation algorithm or small session IDs
- Most computers using TCP/IP are vulnerable
- Insecure handling of session IDs
- Most countermeasures do not work without encryption

Session Hijacking Process



Packet Analysis of a Local Session Hijack



Types of Session Hijacking

Passive	In a passive attack, an attacker **hijacks a session** but sits back, watches, and records all the traffic in that session
Active	In an active attack, an attacker finds an **active session** and seizes control of it

Session Hijacking in OSI Model

Network Level Hijacking

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

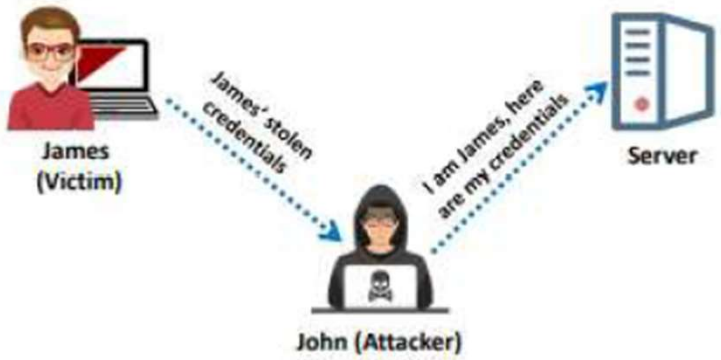
Application Level Hijacking

- Application level hijacking refers to **gaining control** over the **HTTP's user session** by obtaining the session IDs

Spoofing vs. Hijacking

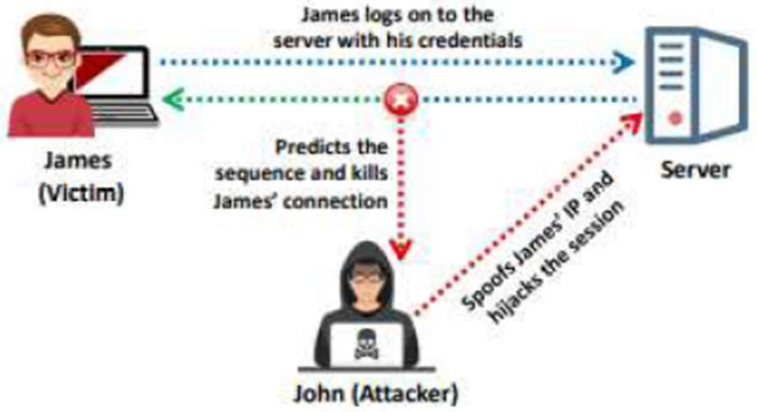
Spoofing Attack

- An attacker **pretends to be another user** or machine (victim) to gain access
- The attacker does not seize control of an existing active session; instead, he or she initiates a new session using the victim's **stolen credentials**



Hijacking

- Session hijacking is the process of seizing control of an **existing active session**
- The attacker relies on the **legitimate user** to create a connection and authenticate



Module Flow

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

Application Level Session Hijacking

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

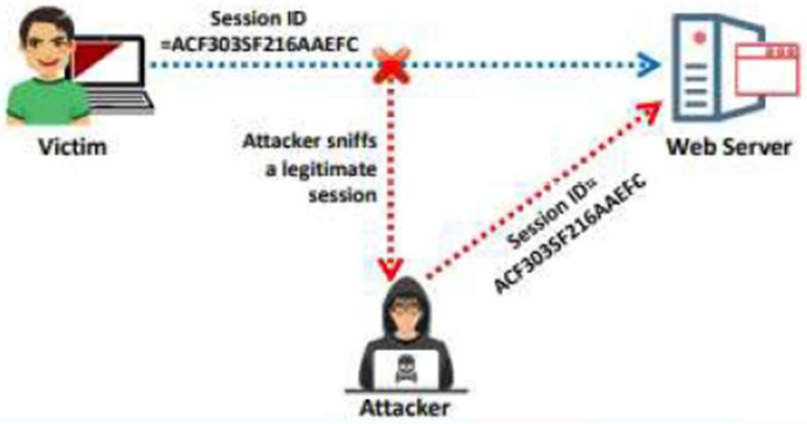
A session token can be compromised in various ways

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

Compromising Session IDs using Sniffing and by Predicting Session Token

Compromising Session IDs using Sniffing

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



Compromising Session IDs by Predicting Session Token

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

How to Predict a Session Token

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

Captures

An attacker captures several session IDs and analyzes the pattern

```
http://www.certifiedhacker.com/view/JBEX18082019152820
http://www.certifiedhacker.com/view/JBEX18082019153020
http://www.certifiedhacker.com/view/JBEX18082019160020
http://www.certifiedhacker.com/view/JBEX18082019164020
                                   Constant  Date  Time
```

Predicts

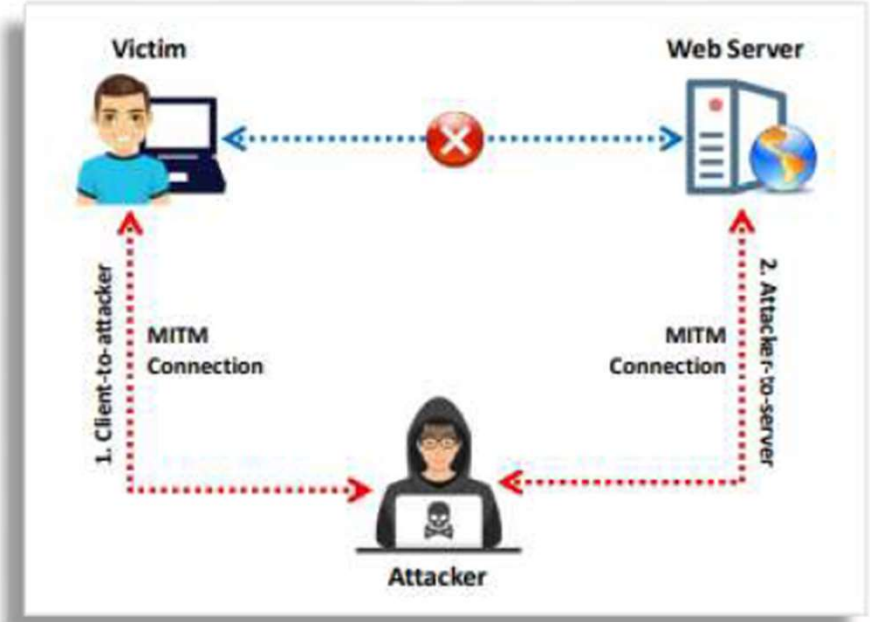
At 16:25:55 on August 23, 2019, the attacker can successfully predict the session ID

```
http://www.certifiedhacker.com/view/JBEX23082019162555
                                   Constant  Date  Time
```

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

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

- Attackers use different techniques and **split the TCP connection** into two connections:
  - Client-to-attacker connection
  - Attacker-to-server connection
- After the interception of the TCP connection, an attacker can read, modify, and insert fraudulent data into the **intercepted communication**
- In the case of an **http transaction**, the TCP connection between the client and the server becomes the target



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

- The 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**

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. The Trojan registers a **button event handler** when a specific page load is detected for a specific pattern and compares it with its targeted list
8. When the user clicks on the button, the extension uses **DOM interface** and extracts all the data from all form fields and modifies the values
9. The browser sends the **form** and **modified values** to the server
10. The server receives the **modified values** but cannot distinguish between the original and the modified values
11. After the server performs the transaction, a **receipt is generated**
12. Now, the browser receives the receipt for the **modified transaction**
13. The browser displays the receipt with the **original details**
14. The user thinks that the **original transaction** was received by the server without any interceptions

Compromising Session IDs Using Client-side Attacks

Cross-Site Scripting (XSS)

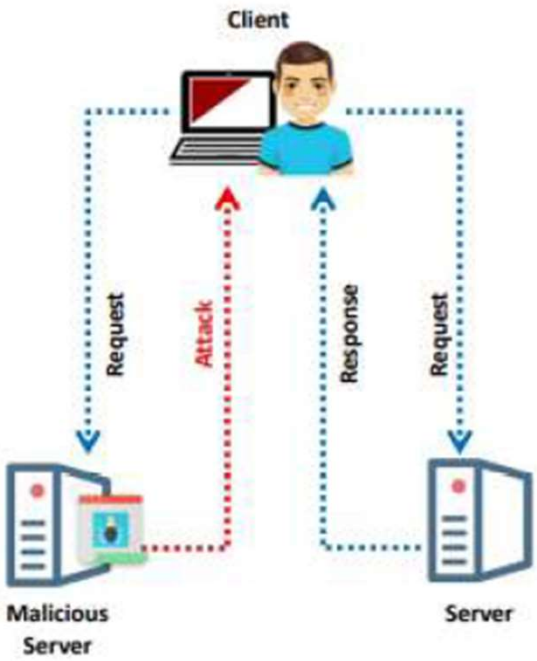
- XSS enables attackers to **inject malicious client-side scripts** into the web pages viewed by other users

Malicious JavaScript Codes

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

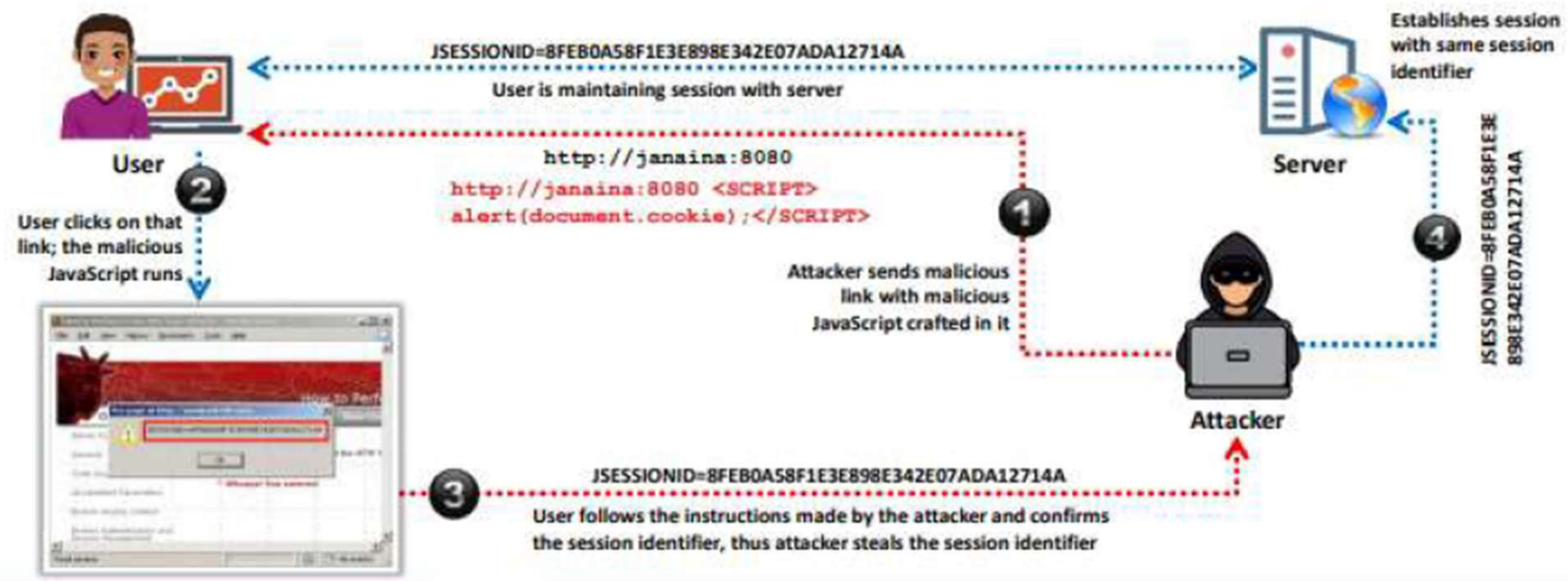
Trojans

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



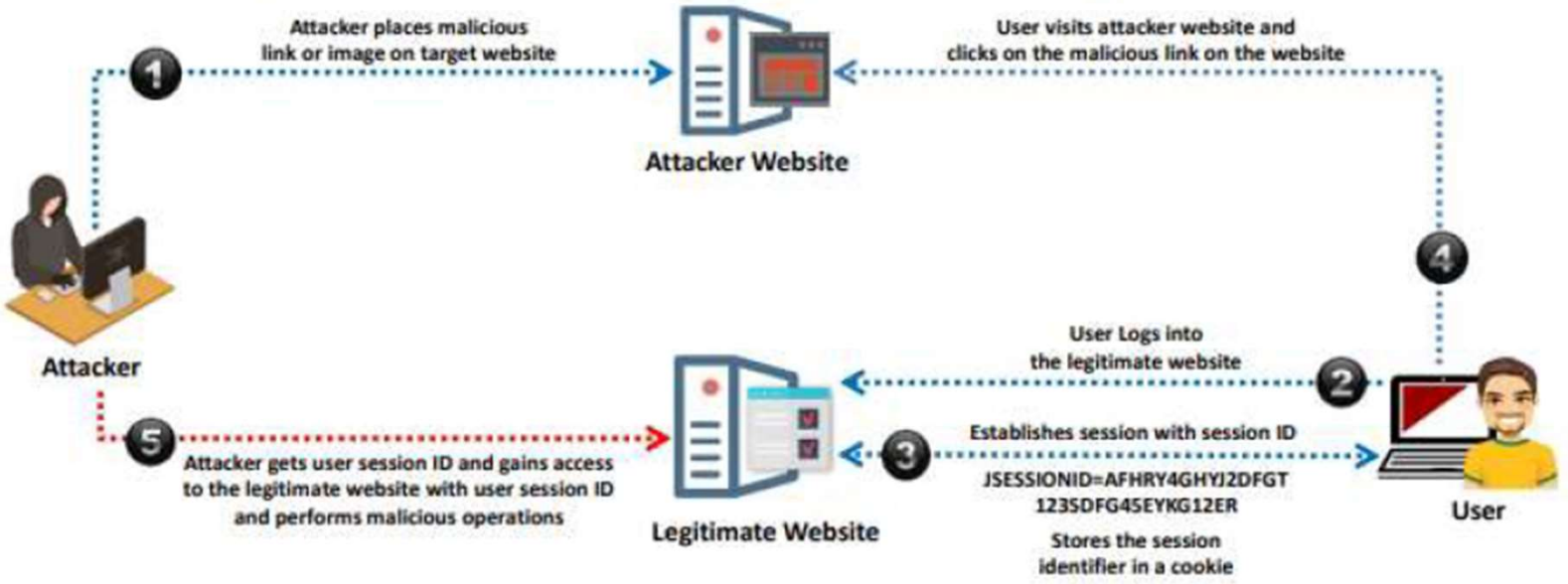
Compromising Session IDs Using Client-side Attacks: Cross-site Script Attack

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



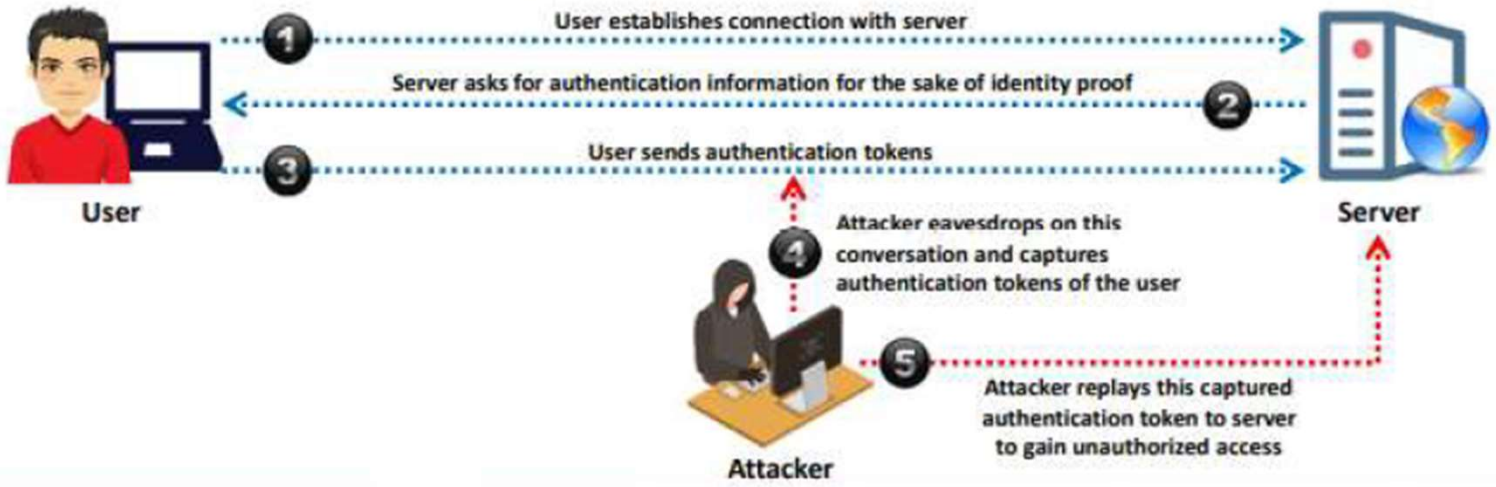
Compromising Session IDs Using Client-side Attacks: Cross-site Request Forgery Attack

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



Compromising Session IDs Using Session Replay Attacks

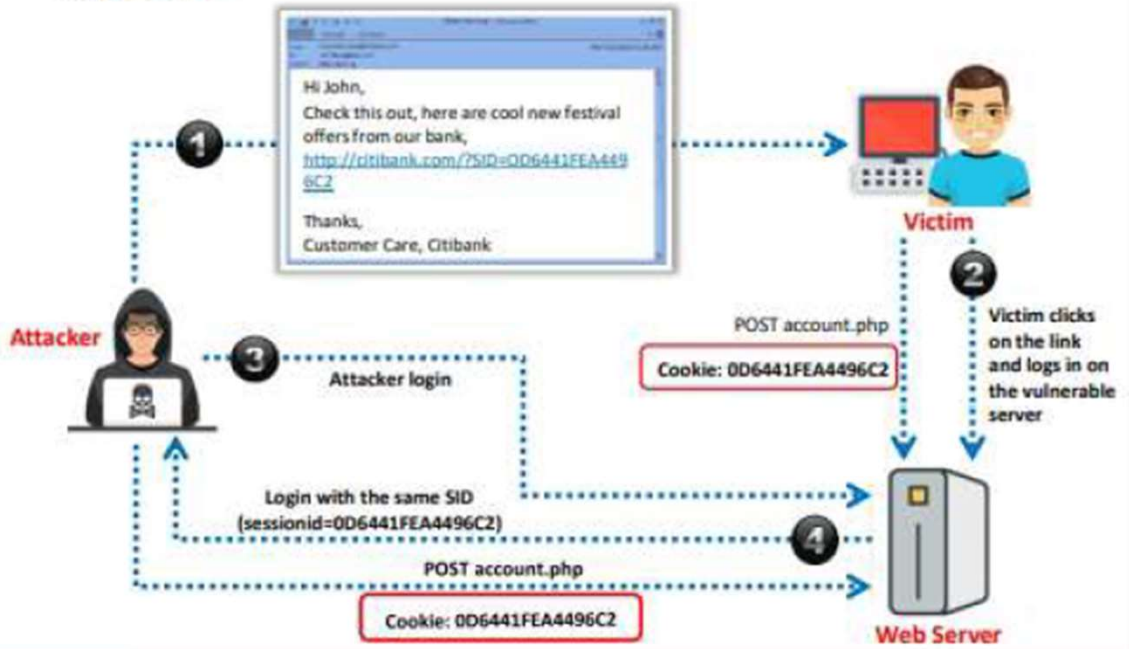
- In a session replay attack, the attacker listens to the conversation between the **user and the server** and captures the **authentication token** of the user
- Once the authentication token is captured, the attacker **replays the request to the server** with the captured **authentication token** and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



Compromising Session IDs Using Session Fixation

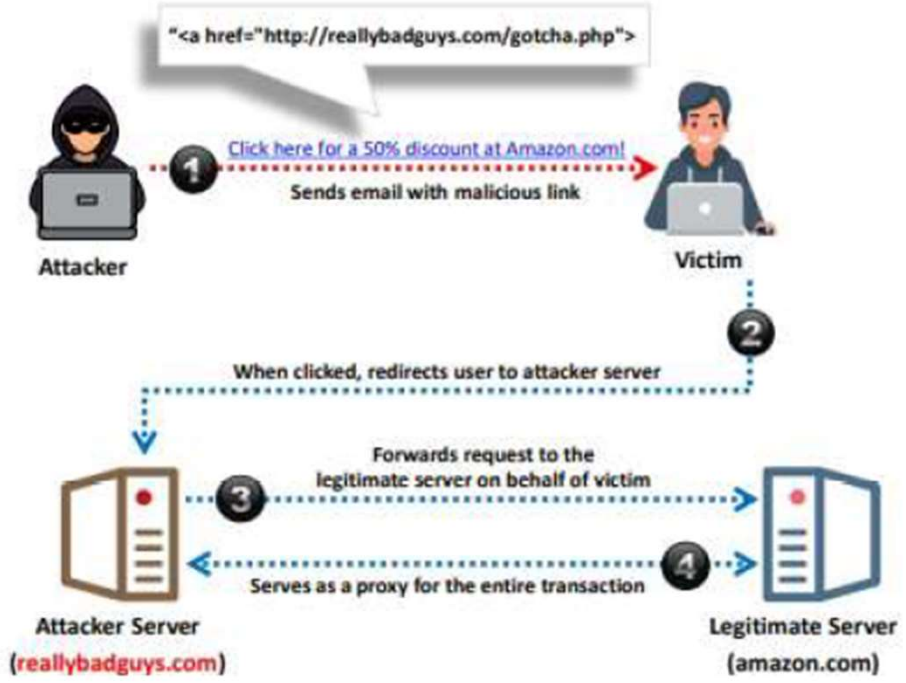
- Session fixation is an attack that allows an attacker to hijack a **valid user session**
- An attacker attempts to lure a user to authenticate himself or herself with a known session ID and then hijacks the **user-validated session** with the knowledge of the used session ID
- The attacker has to provide a **legitimate web application session ID** and attempt to lure the victim's browser to use it
- Some techniques for executing session fixation attacks are as follows:
  - Session token in the **URL argument**
  - Session token in a **hidden form field**
  - Session ID in a **cookie**

- The attacker exploits the **vulnerability of a server** that allows a user to use a fixed SID
- The attacker provides a **valid SID** to a victim and lures him or her to **authenticate** using that SID



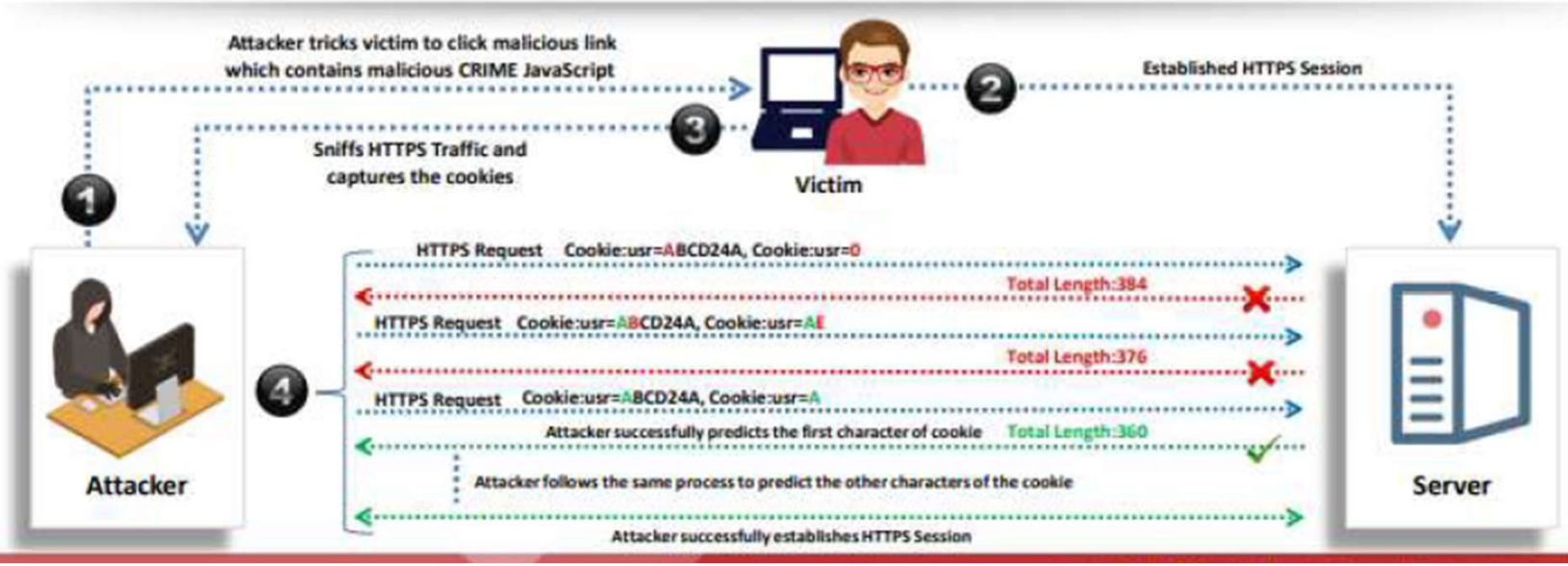
Session Hijacking Using Proxy Servers

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



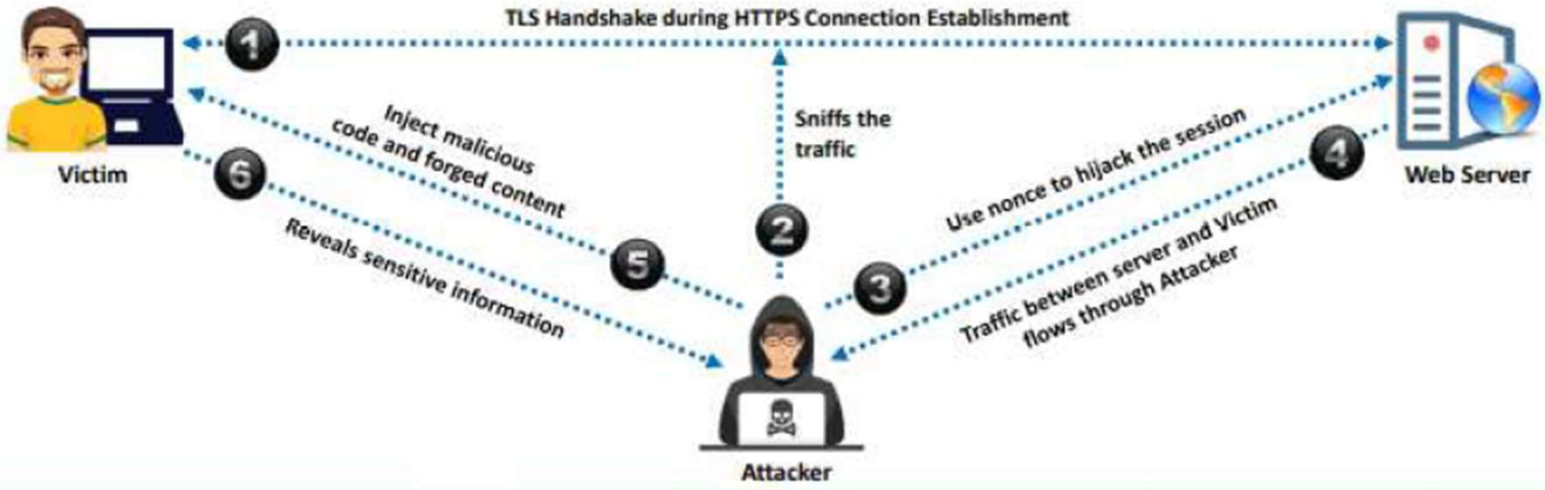
Session Hijacking Using CRIME Attack

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



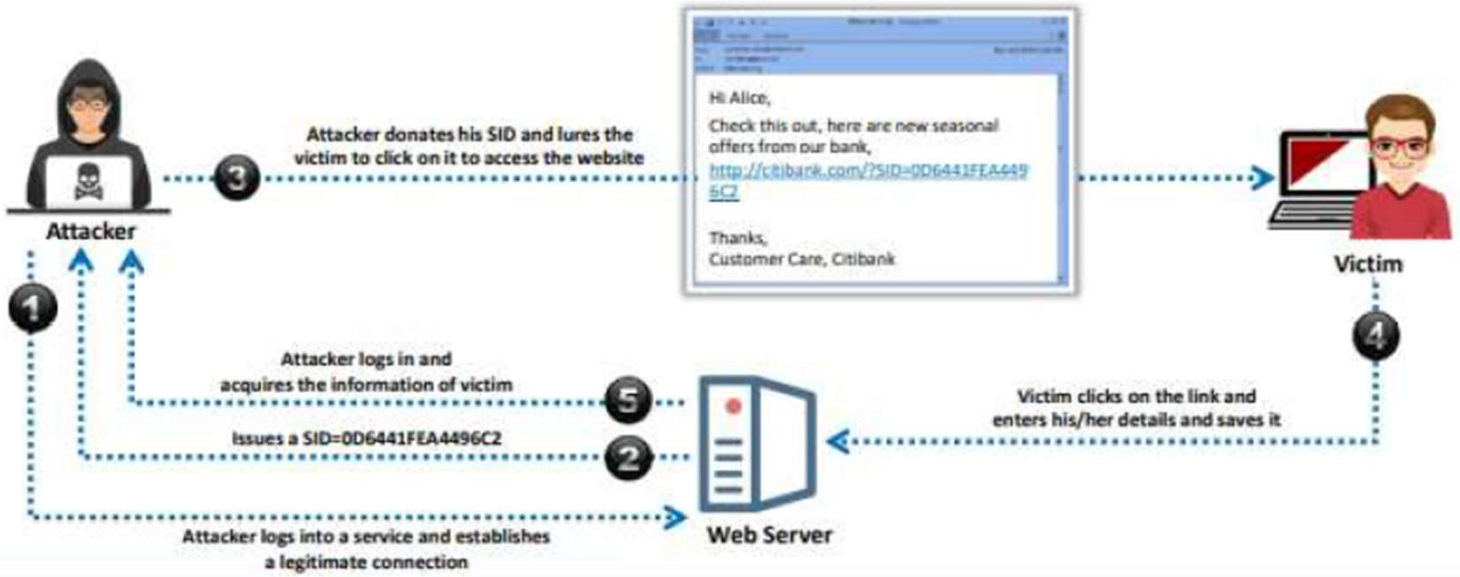
Session Hijacking Using Forbidden Attack

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



Session Hijacking Using Session Donation Attack

- In a session donation attack, an attacker **donates his/her own session identifier (SID)** to the target user
- The attacker first **obtains a valid SID** by logging into a service and later feeds the same SID to the target user
- This SID **links a target user back to the attacker's account page** without any information to the victim



Module Flow

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

Network Level Session Hijacking

1. The network level hijacking relies on hijacking **transport** and **Internet protocols** used by web applications in the application layer

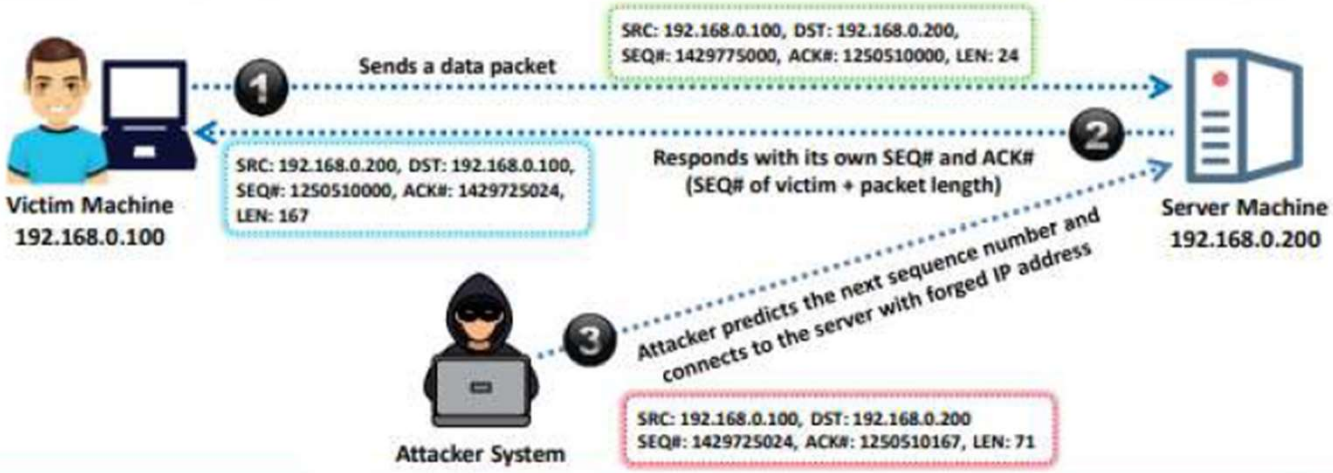
2. By attacking the network level sessions, the attacker gathers some **critical information**, which are used to **attack the application level sessions**

Network level hijacking includes:

1. Blind hijacking
2. UDP hijacking
3. TCP/IP hijacking
6. RST hijacking
7. Man-in-the-middle: Packet sniffer
8. IP spoofing: Source routed packets

TCP/IP Hijacking

- TCP/IP hijacking involves using **spoofed packets** to seize control of a connection between a victim and target machine
- A victim's connection hangs, and an 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 server and the victim machines can be located anywhere

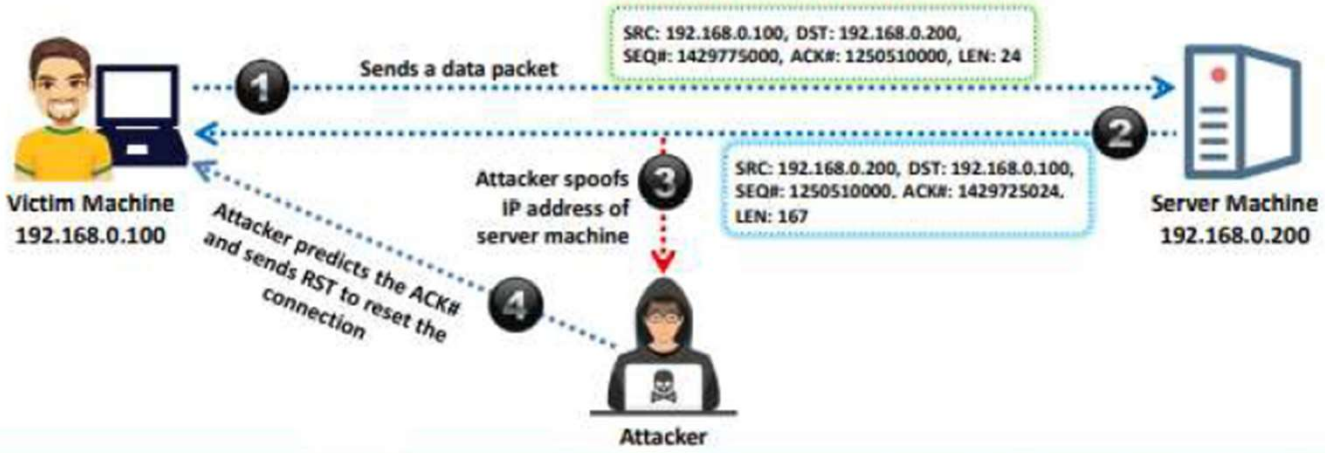


IP Spoofing: Source Routed Packets

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

RST Hijacking

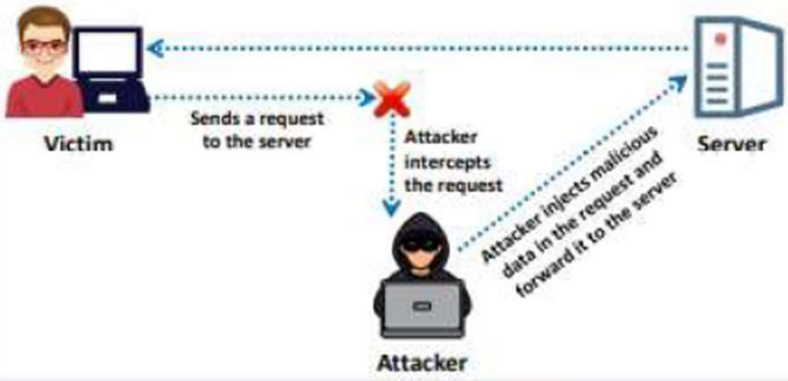
- RST hijacking involves injecting an **authentic-looking reset (RST) packet** using a spoofed source address and predicting the acknowledgment number
- A hacker can reset a victim's connection if it uses an **accurate acknowledgment number**
- The victim would believe that the source sent the **reset packet**, and **reset the connection**
- RST Hijacking can be performed using a **packet crafting tool**, such as Colasoft Packet Builder, and TCP/IP analysis tools, such as tcpdump



Blind and UDP Hijacking

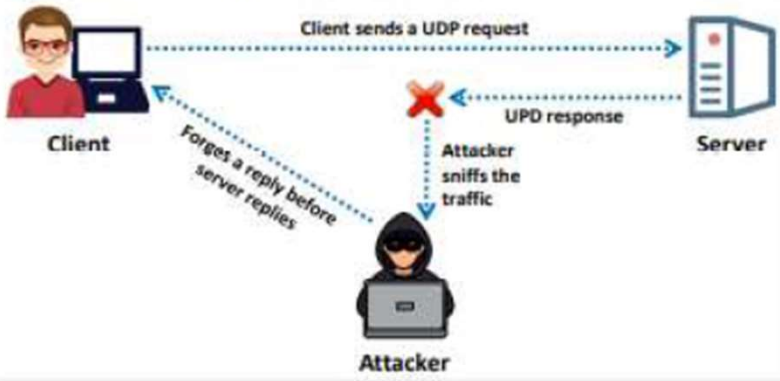
Blind Hijacking

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



UDP Hijacking

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



MiTM Attack Using Forged ICMP and ARP Spoofing

- In this attack, the packet sniffer is **used as an interface** between the client and server
- An attacker changes the **default gateway** of the client's machine and attempts to reroute packets
- The packets between the client and server are routed through the **hijacker's host** using two techniques, as shown below:

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 server**

Address Resolution Protocol (ARP) Spoofing

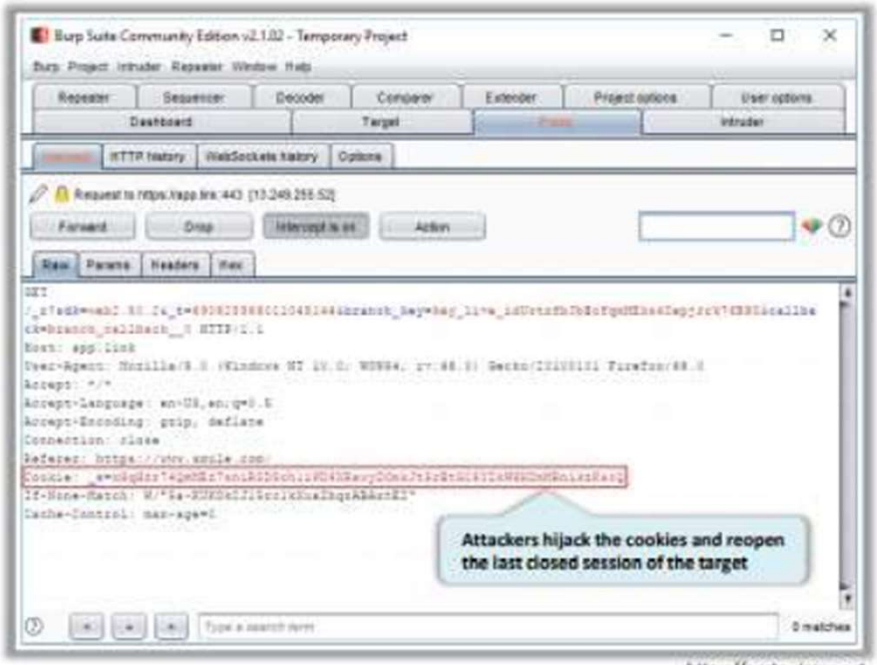
- ARP is used to map the **network layer addresses** (IP address) to **link layer addresses** (MAC address)

Module Flow

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

Session Hijacking Tools

Burp Suite | Burp Suite allows an attacker to **inspect and modify traffic** between the browser and target application



https://portswigger.net

OWASP ZAP
https://www.owasp.org

bettercap
https://www.bettercap.org

netool toolkit
https://sourceforge.net

WebSploit Framework
https://sourceforge.net

sslstrip
https://pypi.python.org

Session Hijacking Tools for Mobile Phones

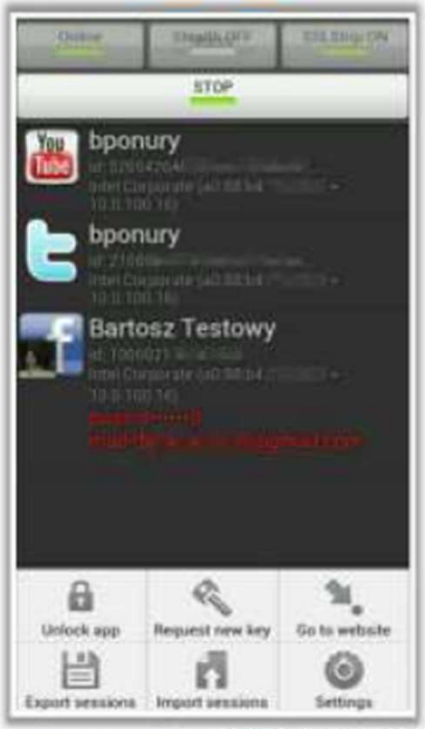
DroidSheep

https://droidsheep.info

DroidSniff

https://github.com

FaceNiff



http://faceniff.ponury.net

Module Flow

01 Session Hijacking Concepts

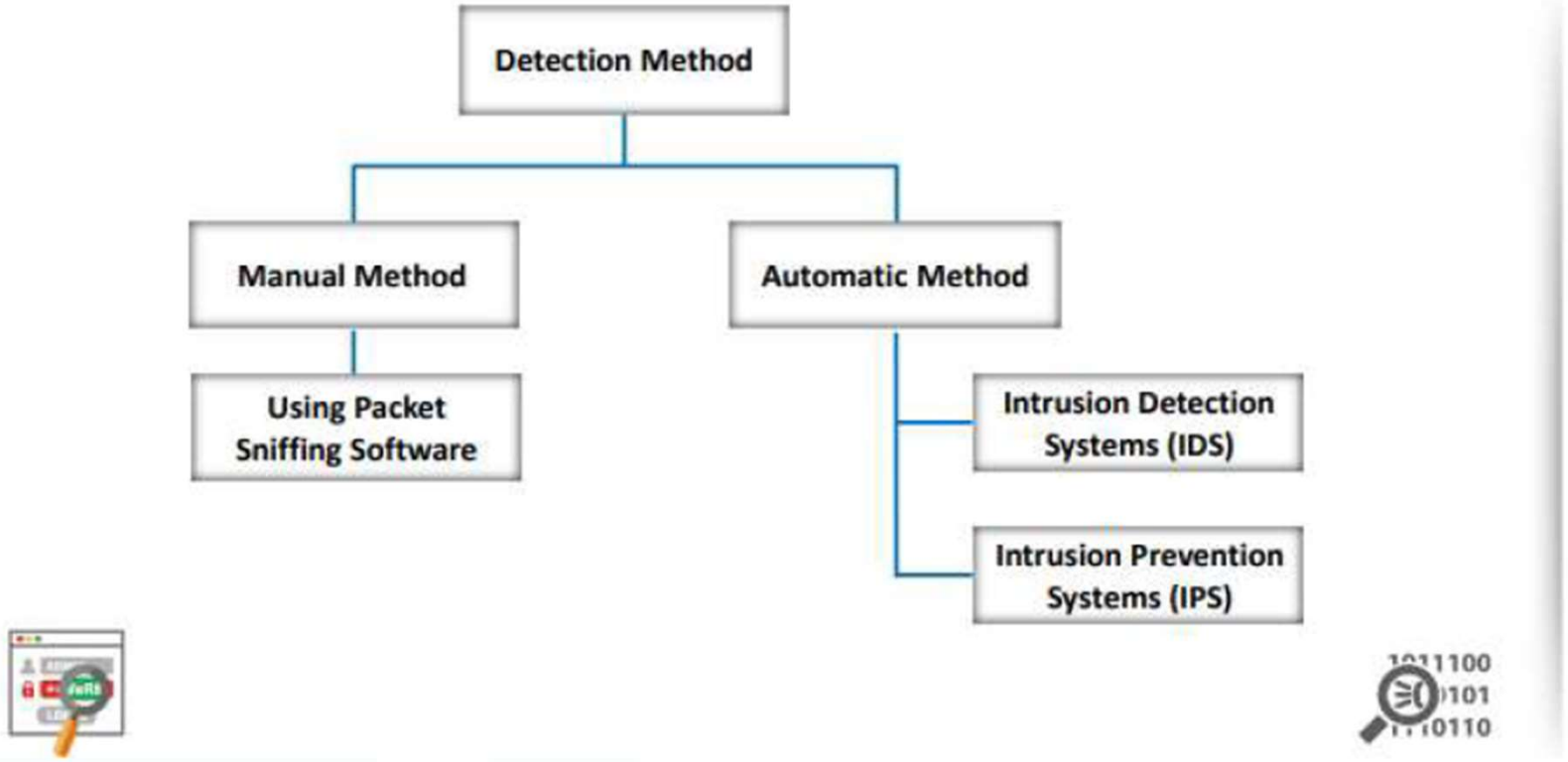
02 Application Level Session Hijacking

03 Network Level Session Hijacking

04 Session Hijacking Tools

05 Countermeasures

Session Hijacking Detection Methods



Protecting against Session Hijacking

1. Use **Secure Shell (SSH)** to create a secure communication channel
2. Implement the **log-out functionality** for the user to end the session
3. Generate the **session ID** after a successful login and accept only session IDs generated by the server only
4. Ensure that data in transit is **encrypted** and implement the **defense-in-depth** mechanism
5. Use **string** or a **long random number** as a session key
6. Use different **usernames** and **passwords** for different accounts
7. Implement **timeout()** to destroy the session when expired
8. Do not transport session ID in **query string**
9. Ensure that **client-side** and **server-side** protection software are active and up-to-date
10. Use **strong authentication** (e.g., Kerberos) or peer-to-peer virtual private networks (VPNs)
11. Configure the appropriate **internal** and **external spoof rules** on gateways
12. Use **IDS products** or **ARPwatch** for monitoring ARP cache poisoning
13. Use **HTTP Public Key Pinning** (HPKP) to allow users authenticate web servers
14. Enable browsers to **verify website authenticity** using network notary servers

Web Development Guidelines to Prevent Session Hijacking

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

Web User Guidelines to Prevent Session Hijacking

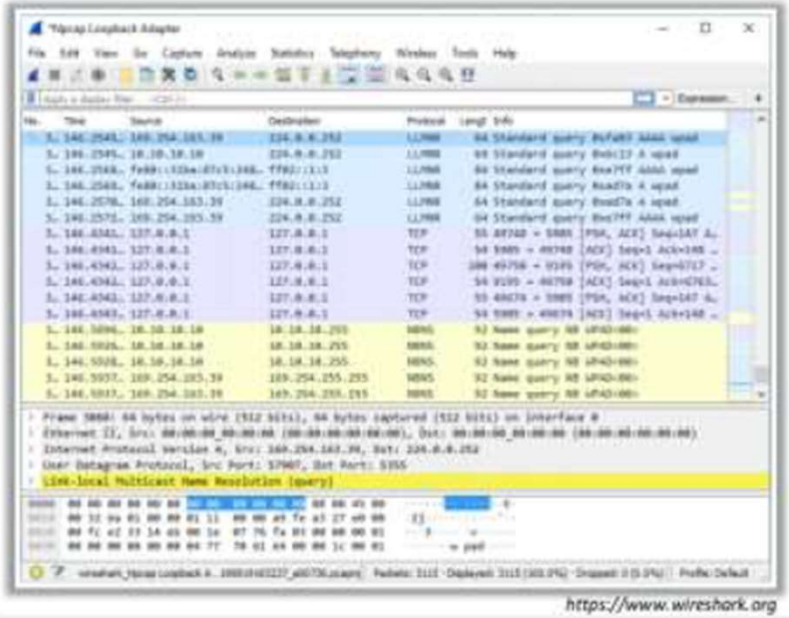
1. Do not click on links received through **emails or IMs**
2. Use firewalls to prevent **malicious content** from entering the network
3. Use firewalls and browser settings to **restrict cookies**
4. Ensure that the website is certified by the **certifying authorities**
5. Ensure that you clear **history**,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

Session Hijacking Detection Tools

AlienVault USM | AlienVault USM delivers **threat detection**, incident response, and compliance management across cloud, on-premises, etc.

https://www.alienvault.com

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



https://www.wireshark.org

Session Hijacking Detection Tools:

- Check Point IPS Software Blade (*https://www.checkpoint.com*)
- LogRhythm (*https://logrhythm.com*)

Approaches Causing Vulnerability to Session Hijacking and their Preventative Solutions

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

Approaches to Prevent Session Hijacking

HTTP Strict Transport Security (HSTS)

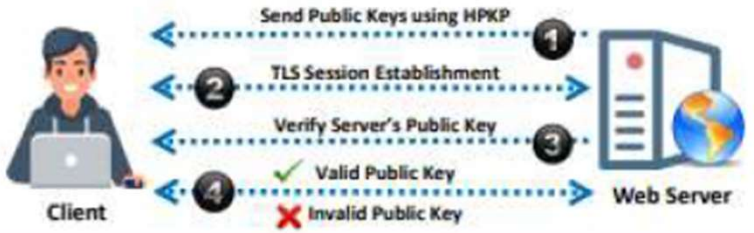
- HTTP Strict Transport Security (HSTS) is a **web security policy** that protects HTTPS websites against MITM attacks
- It allows web servers to **enforce web browsers** to interact with it using secure HTTPS protocol

Token Binding

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

HTTP Public Key Pinning (HPKP)

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



Approaches to Prevent MITM Attacks

WEP/WPA Encryption

- WEP and WPA are the different wireless protocols that are intended to **protect the traffic** that is sent and received by users over a wireless network
- The implementation of these protocols can thwart unwanted users connecting to the network and **prevent MITM attacks**

VPN

- A VPN creates a safe and **encrypted tunnel** over a public network to securely send and receive sensitive information
- The implementation of VPN in the network prevents attackers from **decrypting the data** flowing between the endpoints

Two-Factor Authentication

- A two-factor authentication provides an **extra layer of protection** as it provides another vector of authentication in addition to a user's password
- The implementation of two-factor authentication can prevent attackers from performing **session hijacking** and brute-forcing their way into a user's account

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 **VPNs** and for **remote user access** through dial-up connection to private networks

Components of IPsec

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

Benefits of IPSec

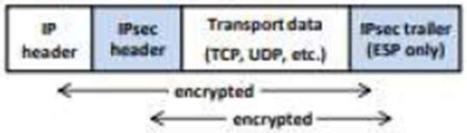
- Network-level peer authentication
- Data origin authentication
- Data integrity
- Data confidentiality (encryption)
- Replay protection

IPSec (Cont'd)

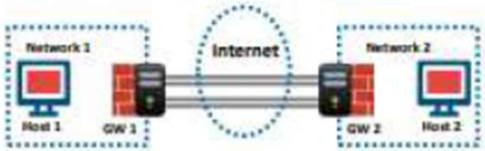
Modes of IPsec

Transport Mode

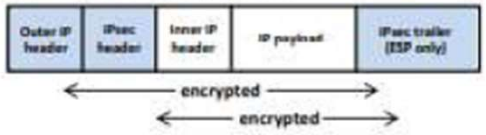
Transport – mode encapsulation



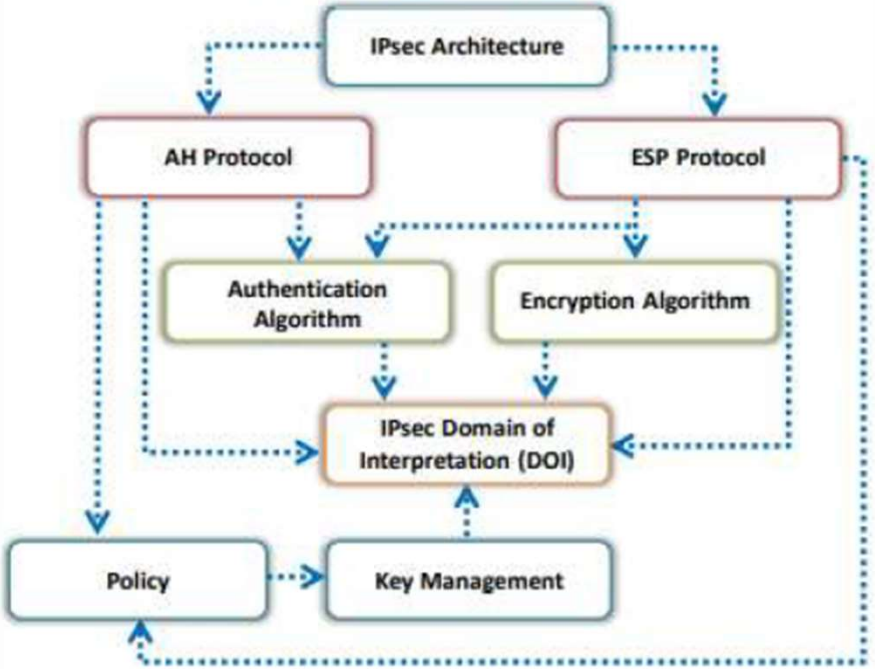
Tunnel Mode



Tunnel – mode encapsulation

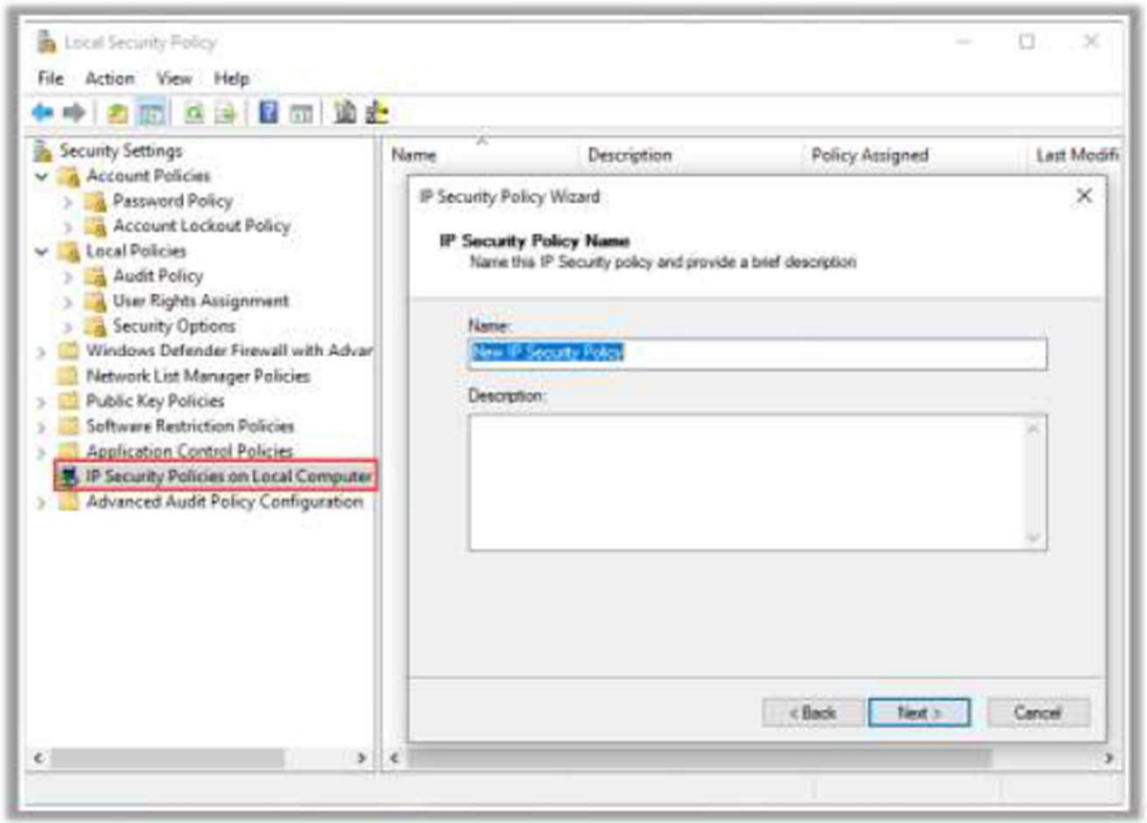


IPsec Architecture



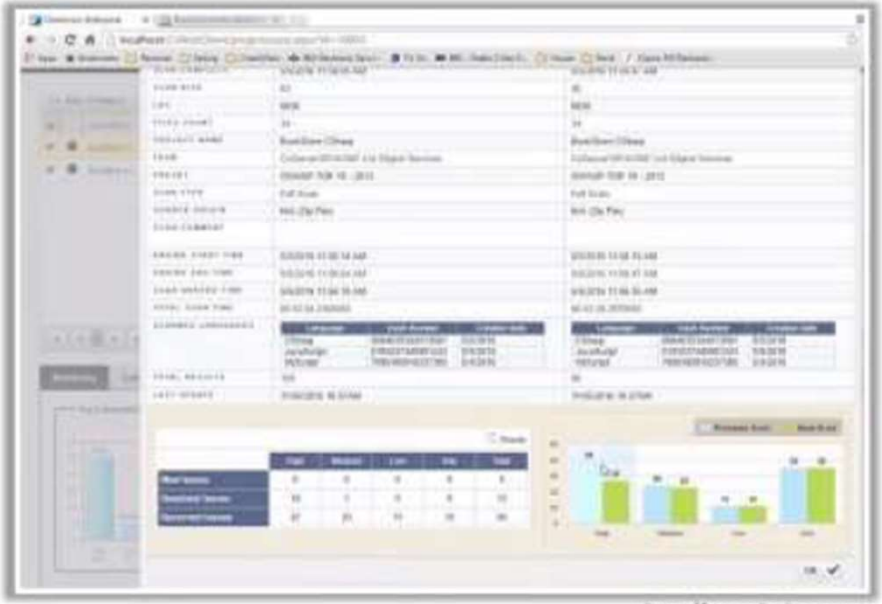
IPsec Authentication and Confidentiality

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



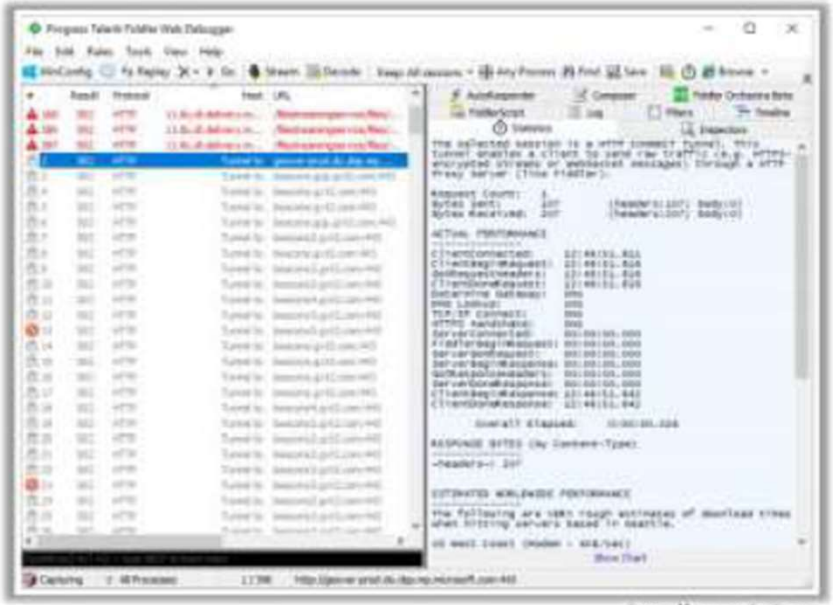
Session Hijacking Prevention Tools

CxSAST | CxSAST is a unique **source code analysis solution** that provides tools to identify, track, and repair technical and logical flaws in the source code



https://www.checkmarx.com

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



https://www.telerik.com

Session Hijacking Prevention Tools:

- Nessus (*https://www.tenable.com*)
- Netsparker (*https://www.netsparker.com*)

Module Summary

- In this module, we have discussed the following:
  - Session hijacking concepts and different types of session hijacking
  - Application level and network level session hijacking attacks
  - Various session hijacking tools
  - How to detect, protect, and defend against session hijacking attacks, as well as various session hijacking detection and prevention tools
  - We concluded with a detailed discussion on various countermeasures to be employed to prevent session hijacking attempts by threat actors
- In the next module, we will discuss in detail how attackers, as well as ethical hackers and pen-testers, evade network security components, such as IDS and firewalls to compromise the infrastructure