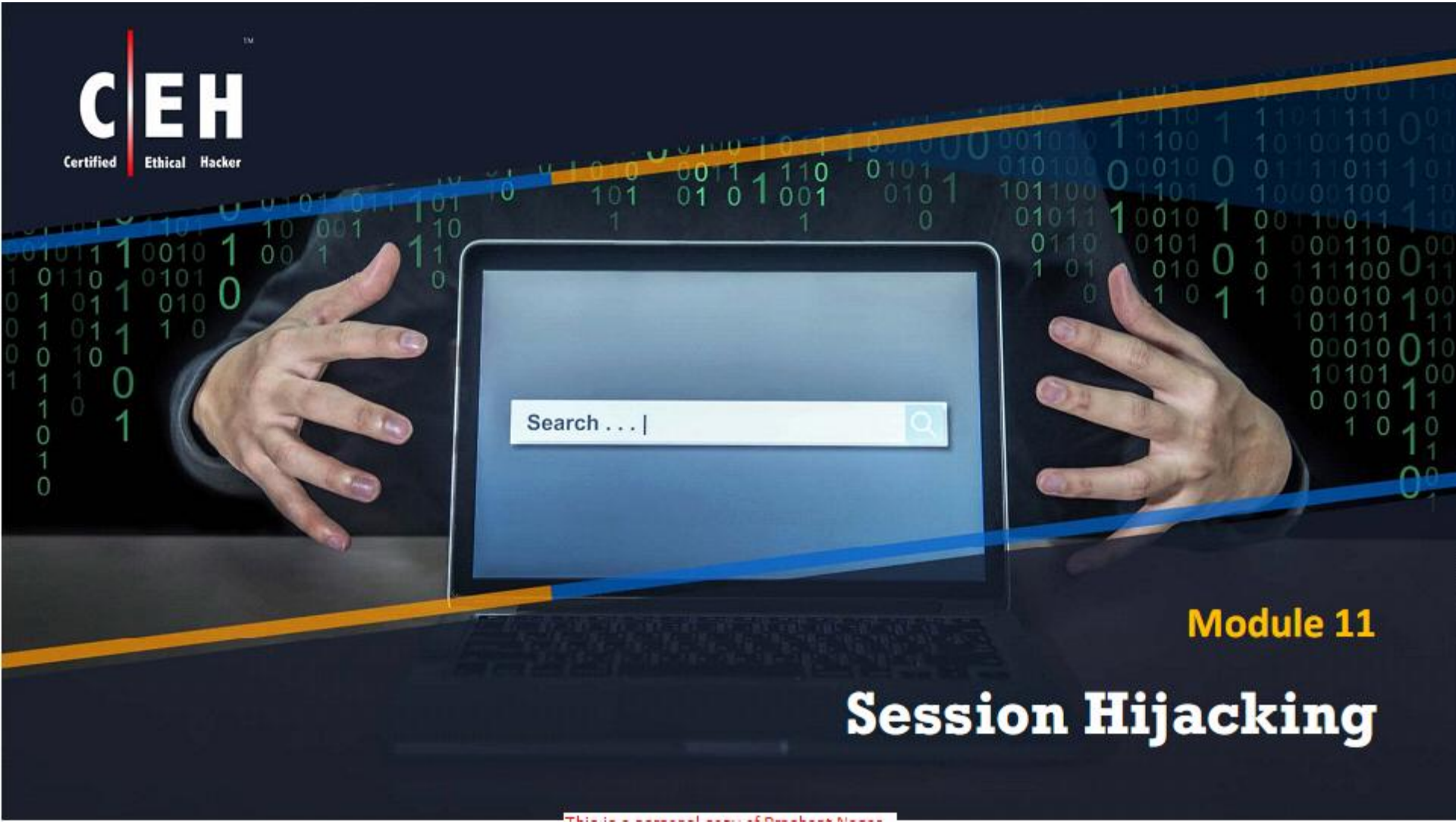


Module 11

Session Hijacking

Module Objectives

Module Objectives

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

Module Flow

01 Session Hijacking Concepts

02 Application Level Session Hijacking

03 Network Level Session Hijacking

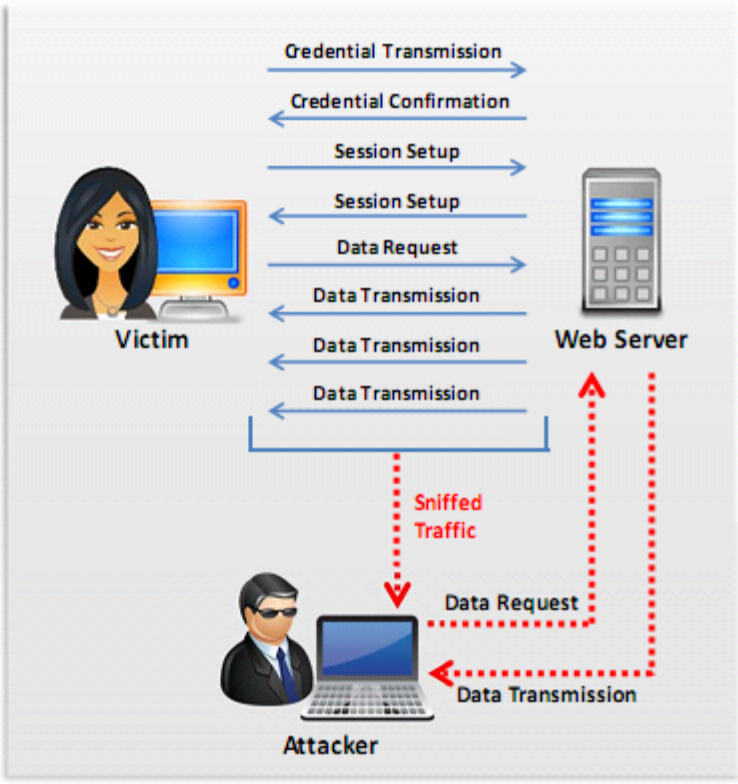
04 Session Hijacking Tools

05 Countermeasures

06 Penetration Testing

What is Session Hijacking?

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



Why Session Hijacking is Successful?

No account lockout for invalid session IDs

Indefinite session expiration time

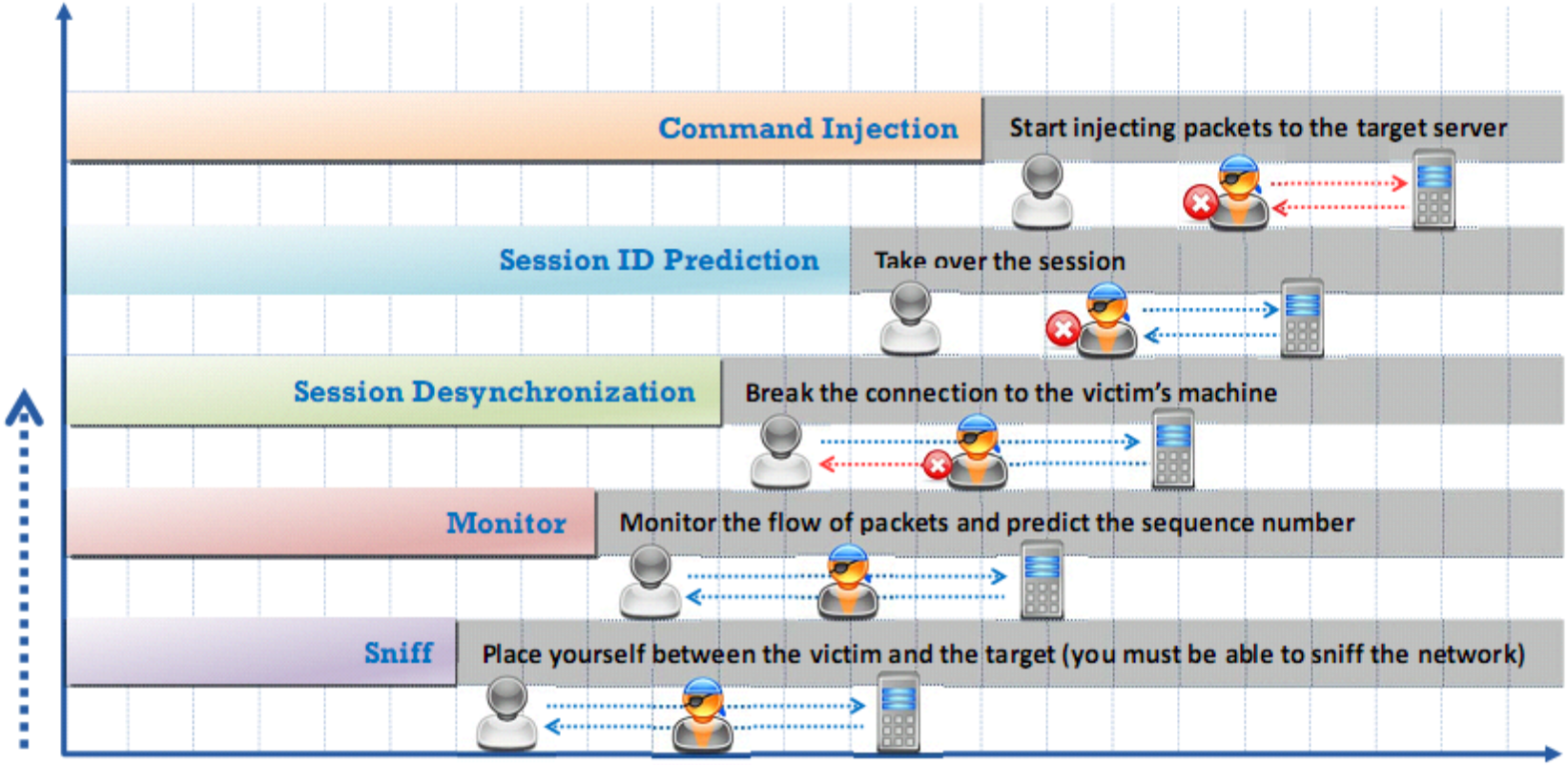
Weak session ID generation algorithm or small session IDs

Most computers using TCP/IP are vulnerable

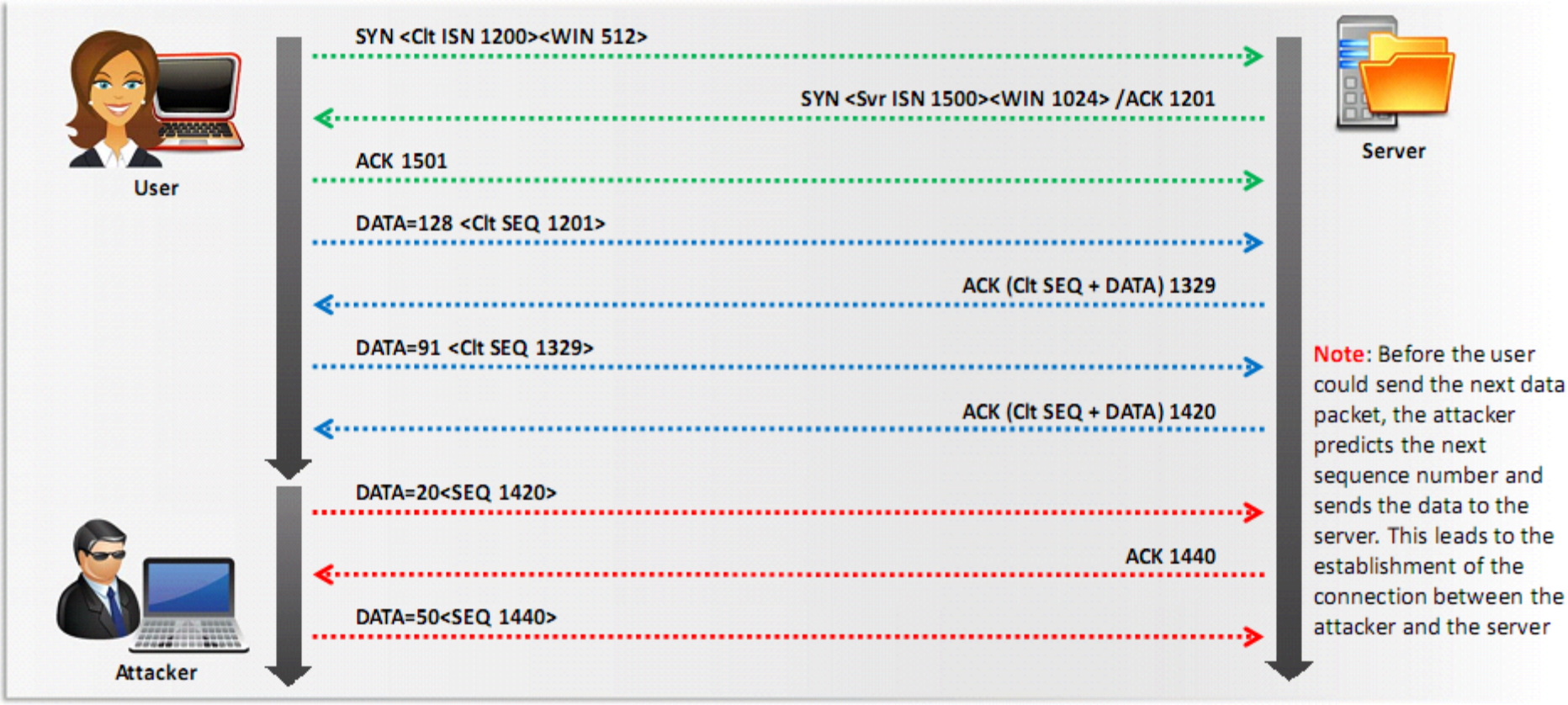
Insecure handling of session IDs

Most countermeasures do not work unless you use encryption

Session Hijacking Process



Packet Analysis of a Local Session Hijack



Types of Session Hijacking

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

Session Hijacking in OSI Model

Network Level Hijacking

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

Application Level Hijacking

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

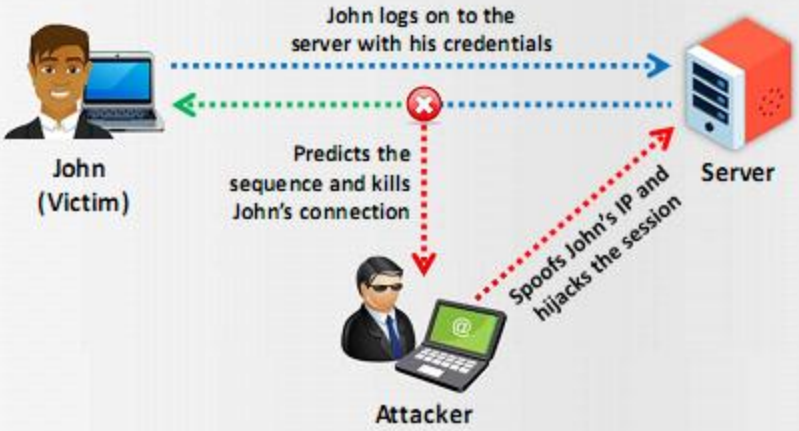
Spoofing vs. Hijacking

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



Module Flow

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

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

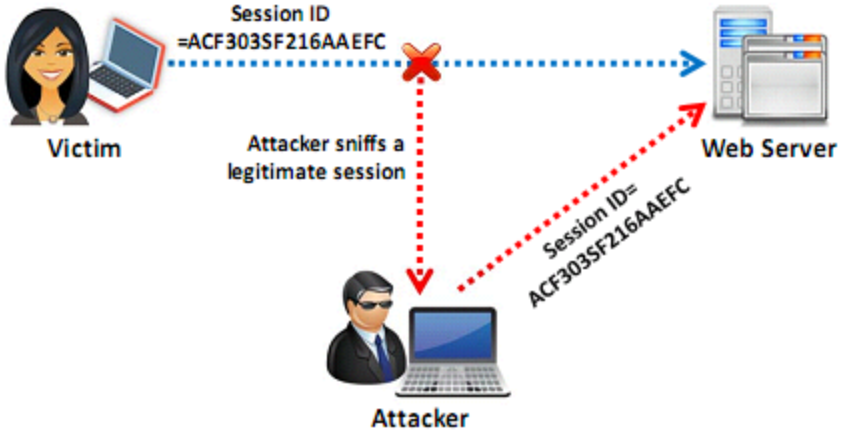
A session token can be compromised in various ways

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

Compromising Session IDs using Sniffing and by Predicting Session Token

Compromising Session IDs using Sniffing

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



Compromising Session IDs by Predicting Session Token

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

How to Predict a Session Token

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

Captures

Attacker captures several session IDs and analyzes the pattern

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

Predicts

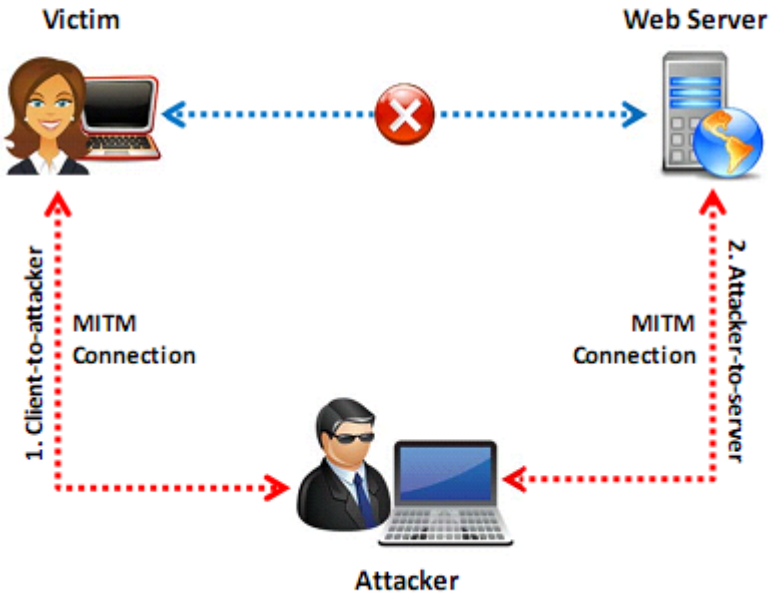
At 16:25:55 on Feb-25, 2017, the attacker can successfully predict the session ID

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

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

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

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



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

Steps to Perform Man-in-the-Browser Attack

1. The Trojan first infects the computer's software (OS or application)
2. The Trojan installs malicious code (extension files) and saves it into the browser configuration
3. After the user restarts the browser, the malicious code in the form of extension files is loaded
4. The extension files register a handler for every visit to the webpage
5. When the page is loaded, the extension uses the URL and matches it with a list of known sites targeted for attack
6. The user logs in securely to the website
7. It registers a button event handler when a specific page load is detected for a specific pattern and compares it with its targeted list
8. When the user clicks on the button, the extension uses DOM interface and extracts all the data from all form fields and modifies the values
9. The browser sends the form and modified values to the server
10. The server receives the modified values but cannot distinguish between the original and the modified values
11. After the server performs the transaction, a receipt is generated
12. Now, the browser receives the receipt for the modified transaction
13. The browser displays the receipt with the original details
14. The user thinks that the original transaction was received by the server without any interceptions

Compromising Session IDs Using Client-side Attacks

Cross-Site Scripting (XSS)

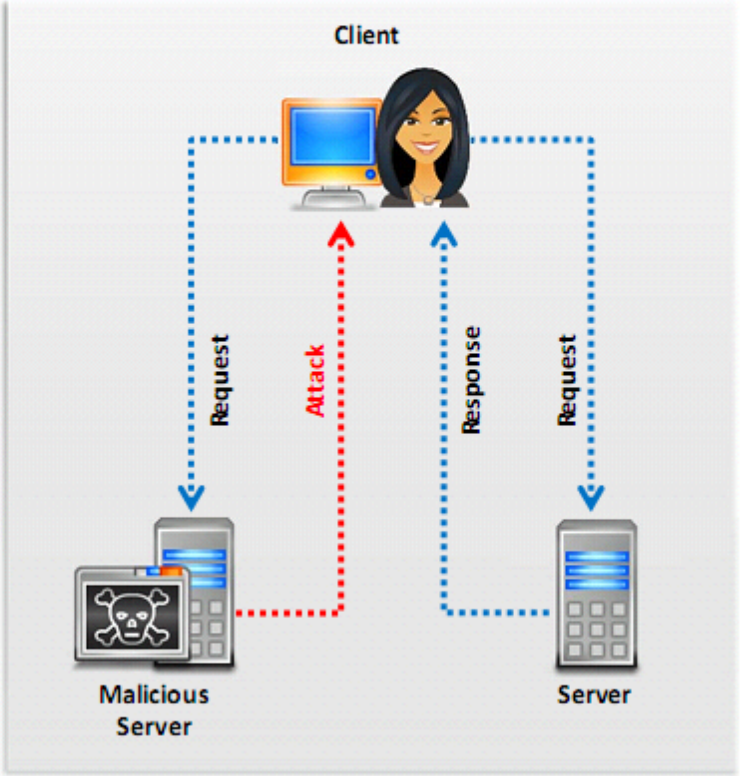
- XSS enables attackers to **inject malicious client side scripts** into the web pages viewed by other users

Malicious JavaScript Codes

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

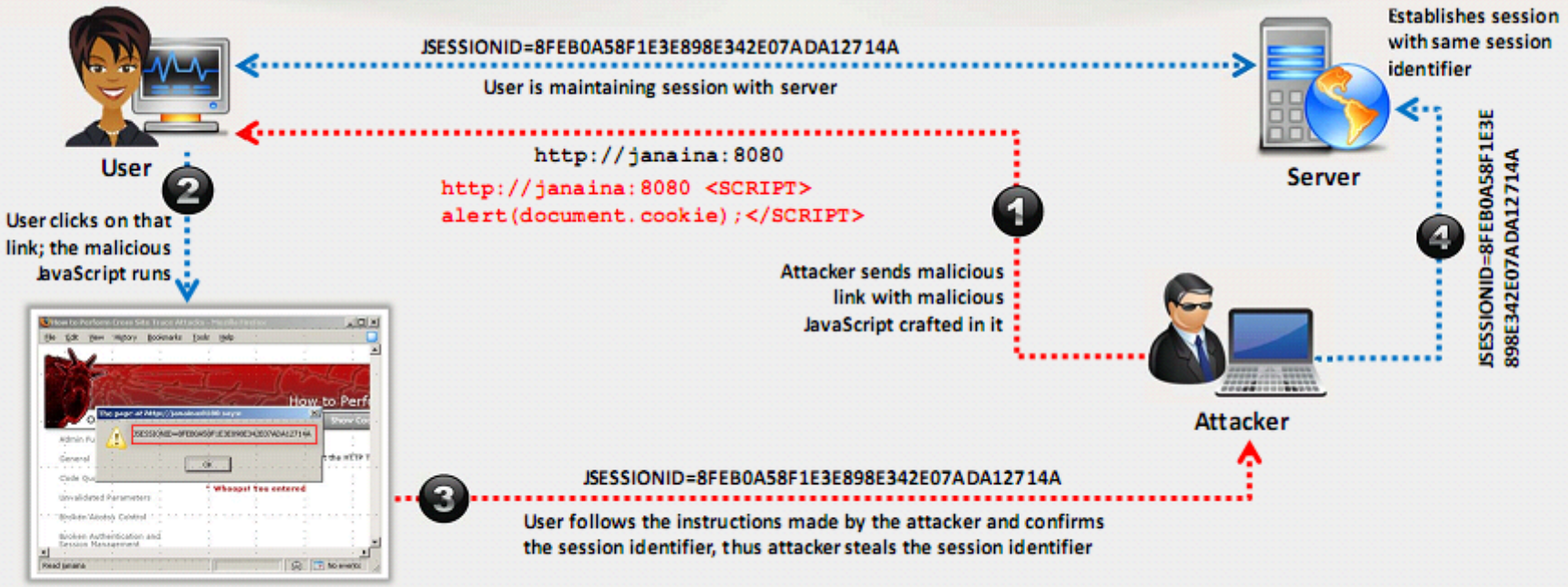
Trojans

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



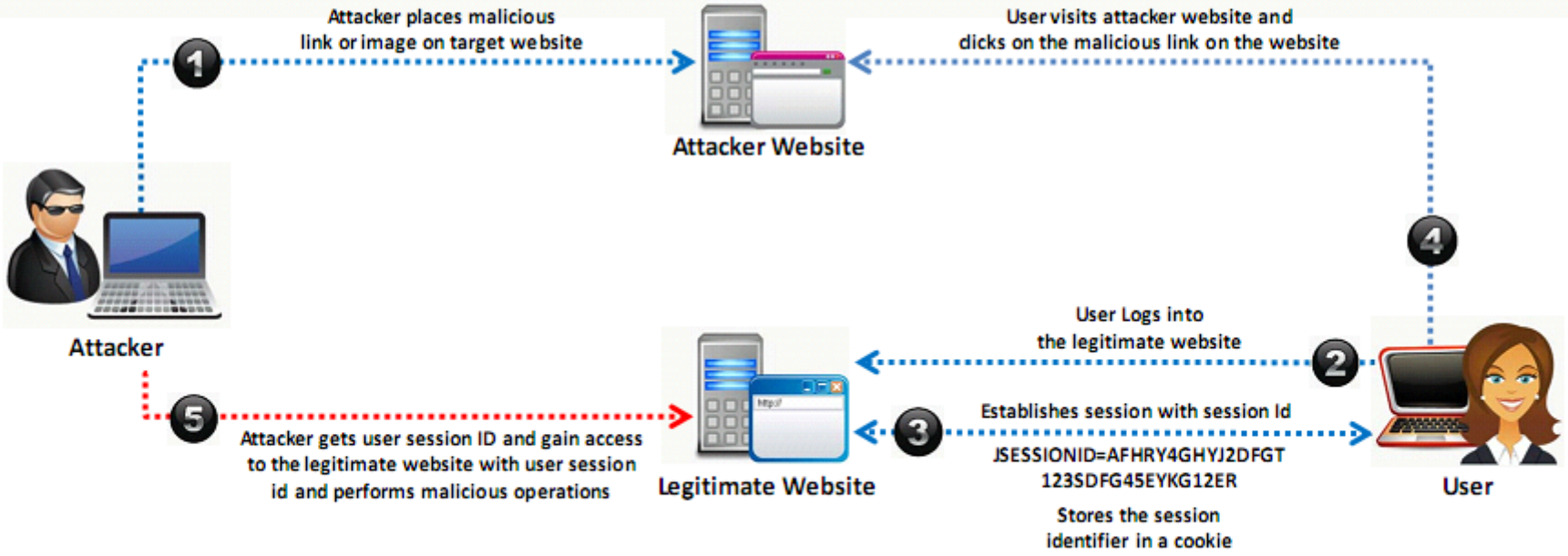
Compromising Session IDs Using Client-side Attacks: Cross-site Script Attack

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



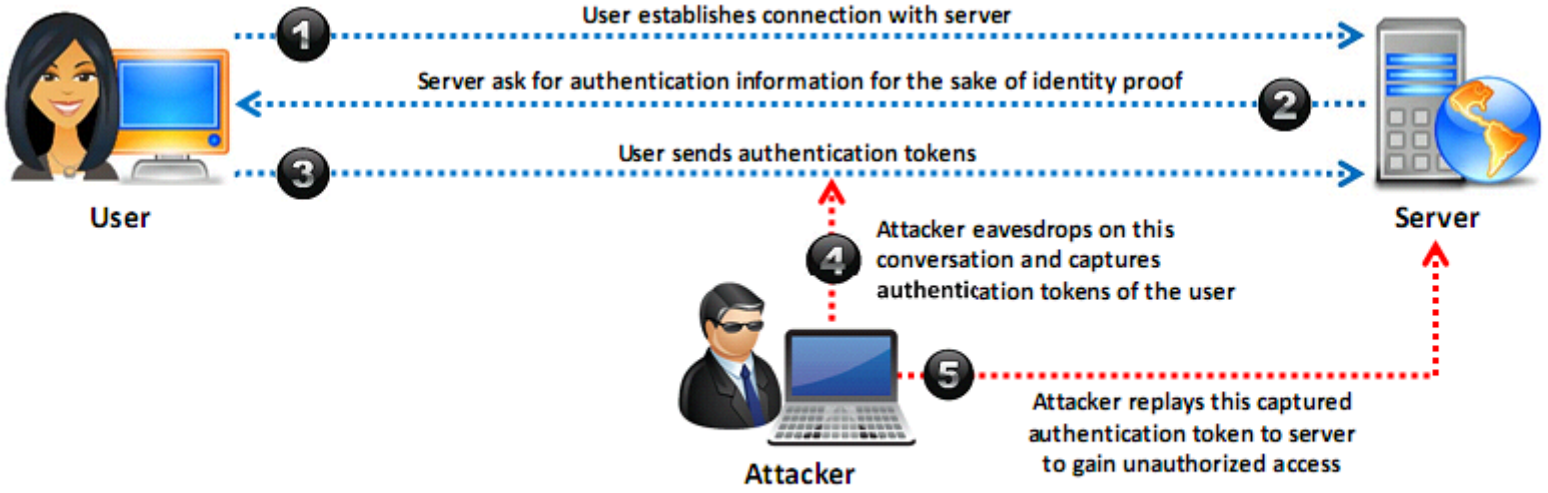
Compromising Session IDs Using Client-side Attacks: Cross-site Request Forgery Attack

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



Compromising Session IDs Using Session Replay Attack

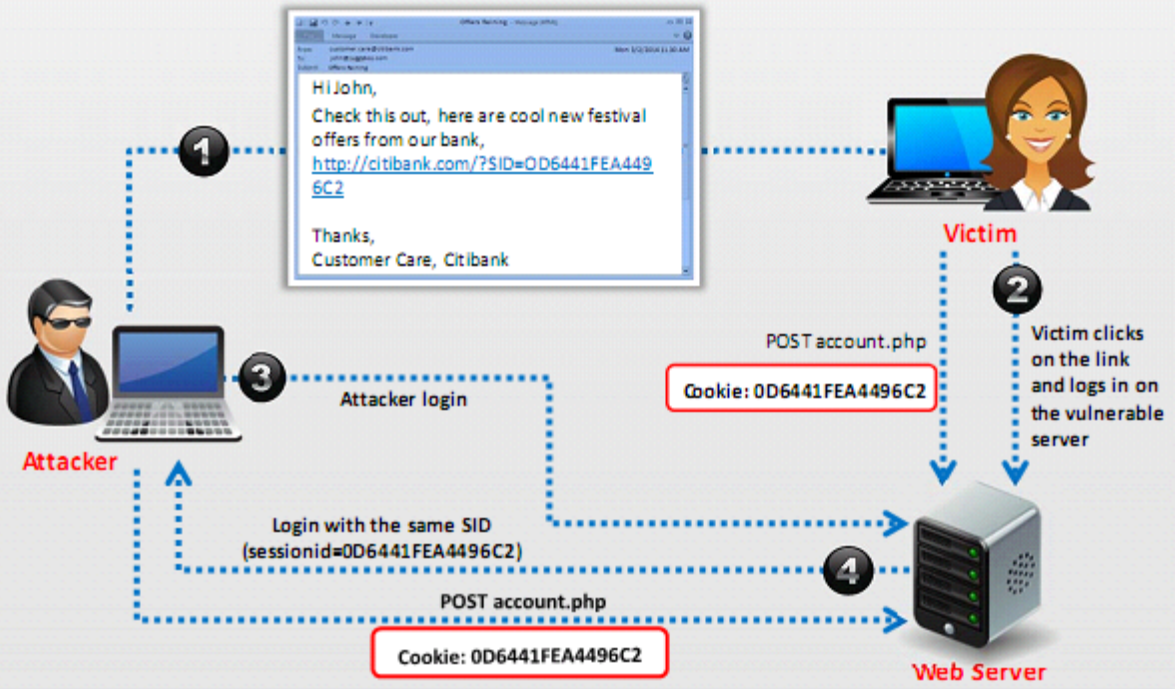
- In a session replay attack, the attacker listens to the conversation between the **user and the server** and captures the **authentication token** of the user
- Once the authentication token is captured, the attacker **replays the request to the server** with the captured **authentication token** and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



Compromising Session IDs Using Session Fixation

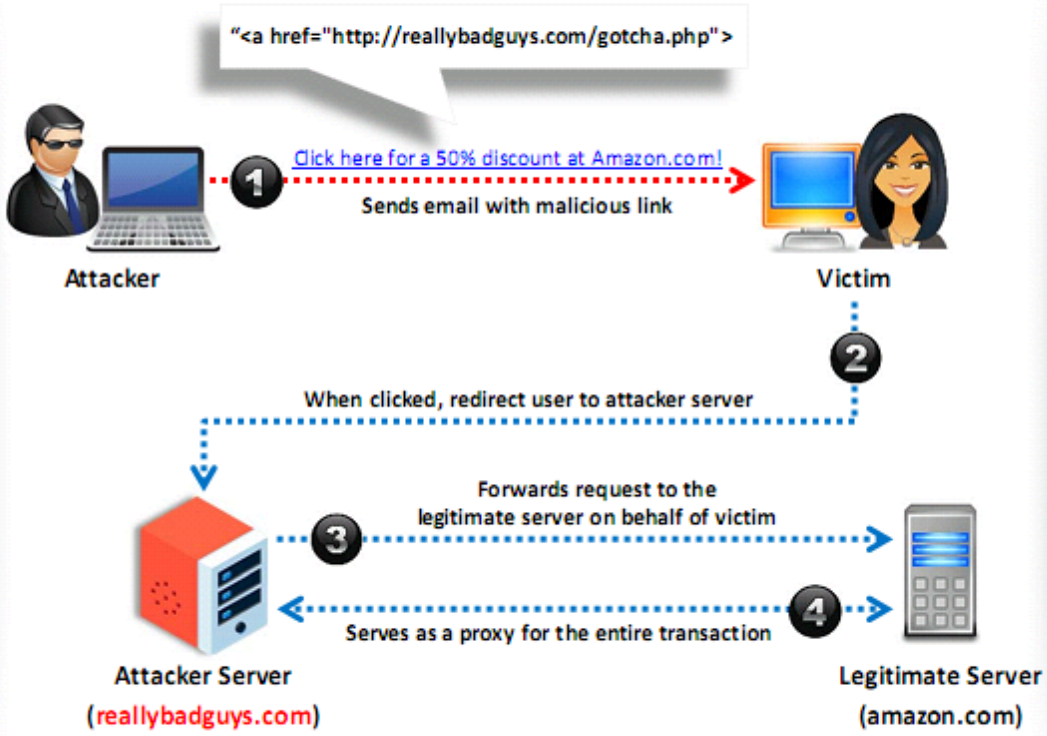
- Session fixation is an attack that allows an attacker to hijack a **valid user session**
- The attack tries to lure a user to authenticate himself with a known session ID and then hijacks the **user-validated session** by the knowledge of the used session ID
- The attacker has to provide a **legitimate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

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



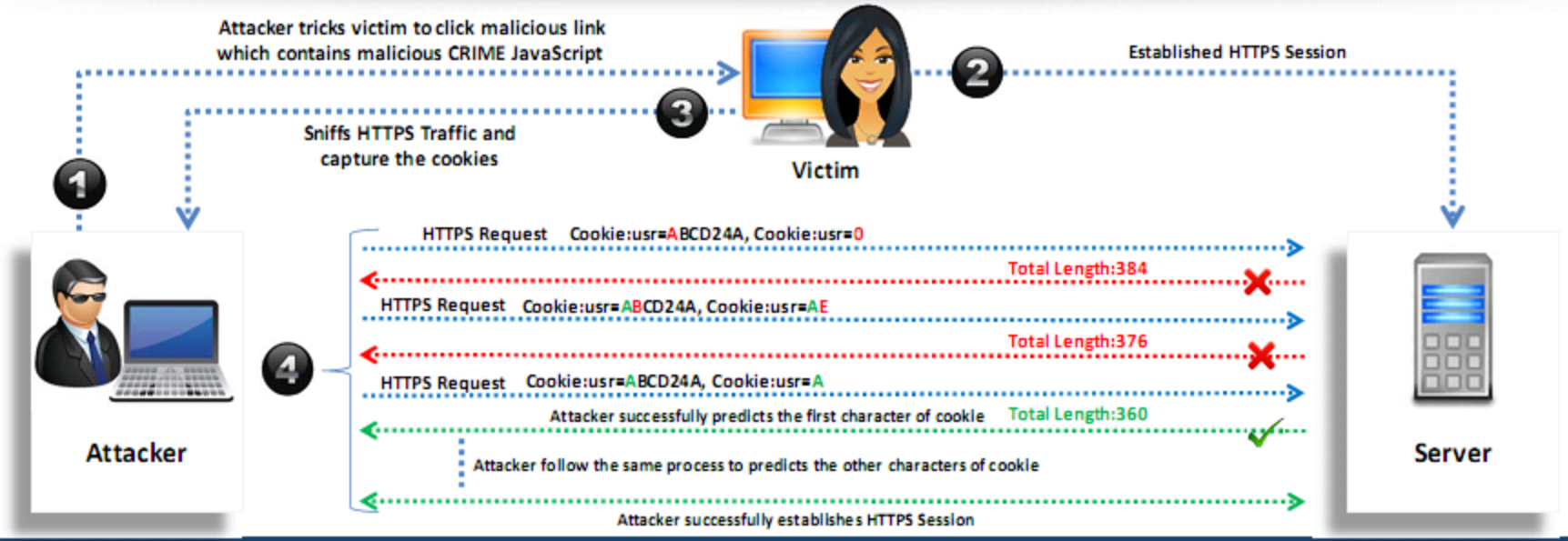
Session Hijacking Using Proxy Servers

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



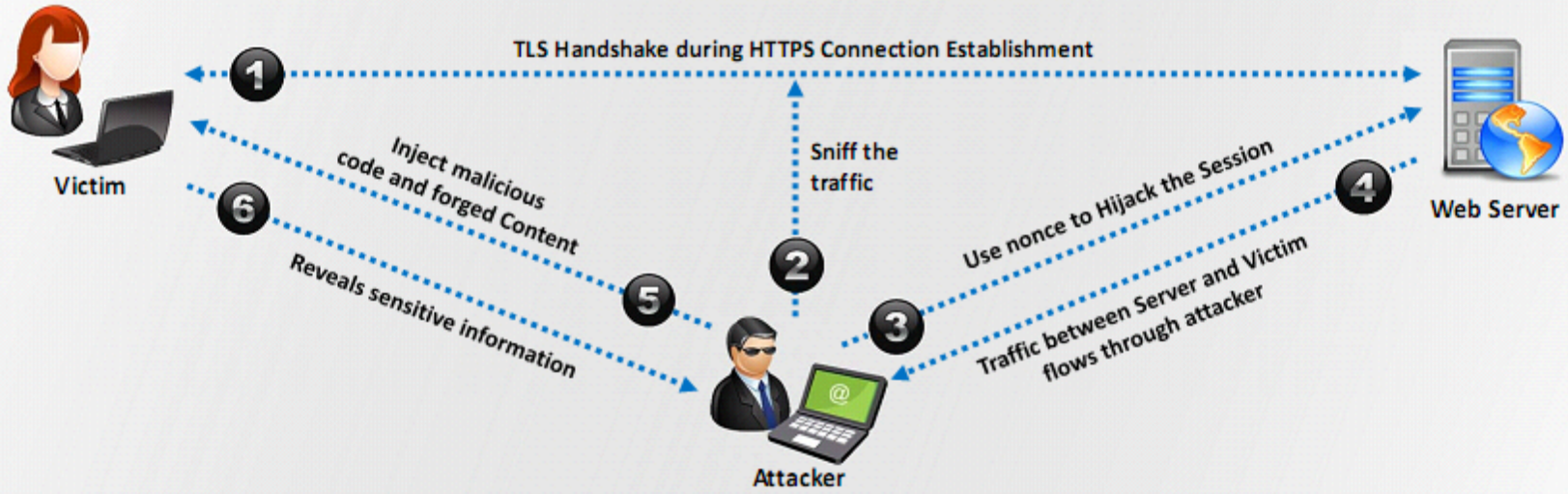
Session Hijacking Using CRIME Attack

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



Session Hijacking Using Forbidden Attack

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



Module Flow

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

Network-level Session Hijacking

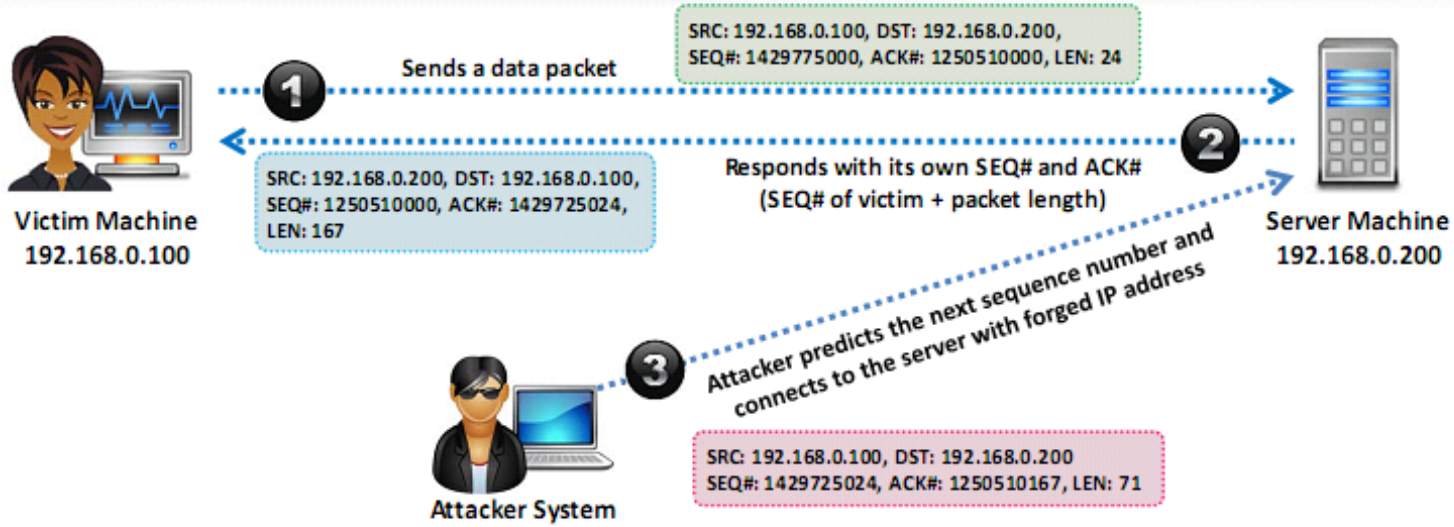
1. The network-level hijacking relies on hijacking **transport** and **Internet protocols** used by web applications in the application layer
2. By attacking the network-level sessions, the attacker gathers some **critical information** which is used to **attack the application level sessions**

Network-level hijacking includes:

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

TCP/IP Hijacking

- TCP/IP hijacking uses **spoofed packets** to take over a connection between a victim and a target machine
- The victim's connection hangs, and the attacker is then able to **communicate with the host's machine** as if the attacker is the victim
- To launch a TCP/IP hijacking attack, the **attacker must be on the same network as the victim**
- The target and the victim machines can be located anywhere

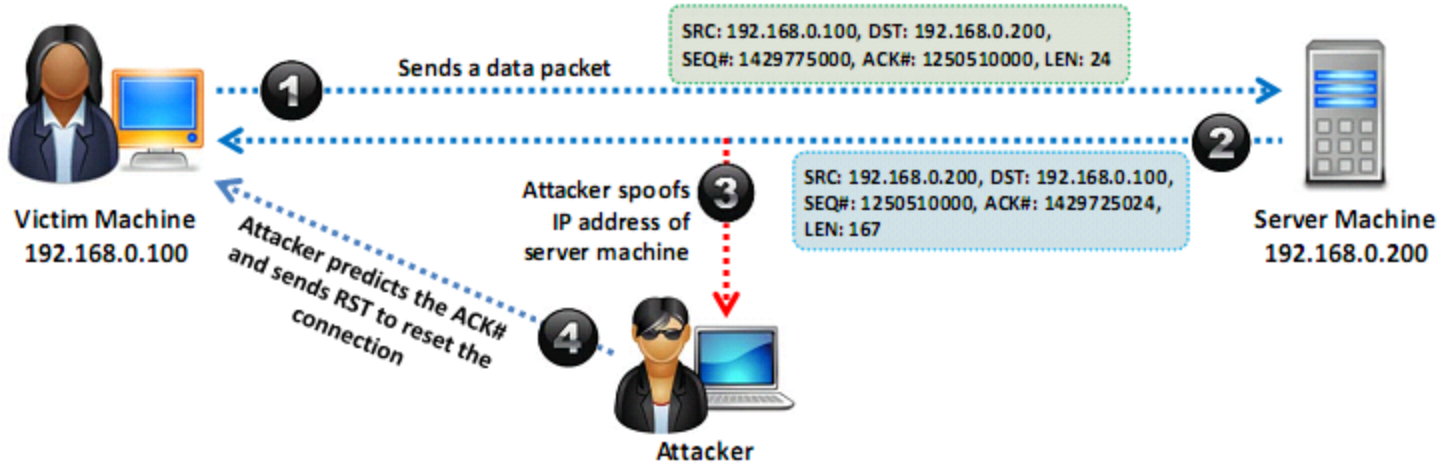


IP Spoofing: Source Routed Packets

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

RST Hijacking

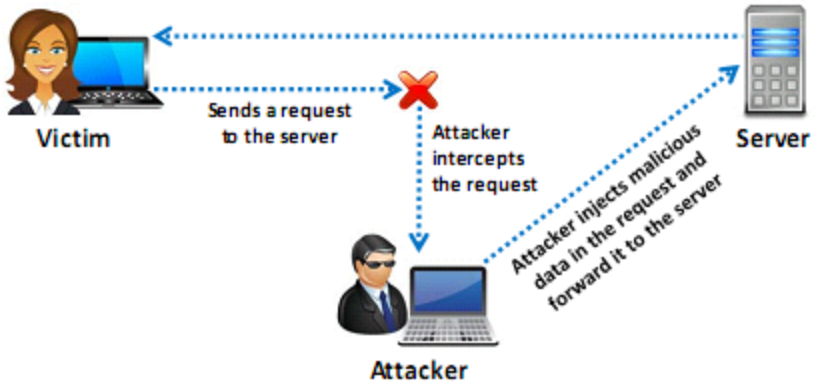
- RST hijacking involves injecting an **authentic-looking reset (RST) packet** using spoofed source address and predicting the acknowledgment number
- The hacker can reset the victim's connection if it uses an **accurate acknowledgment number**
- The victim believes that the source actually sent the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



Blind and UDP Hijacking

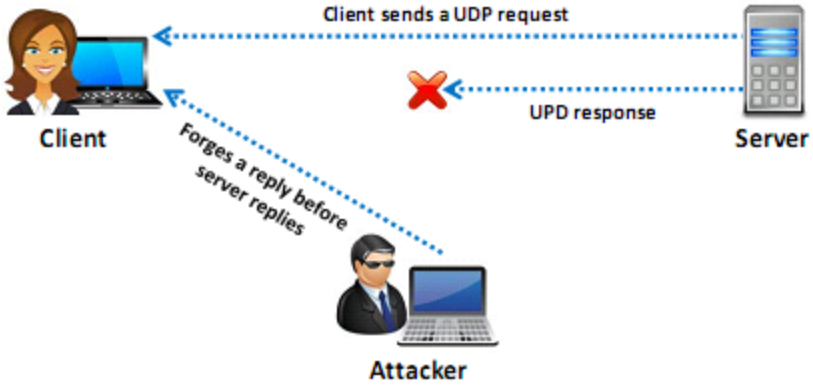
Blind Hijacking

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



UDP Hijacking

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



MiTM Attack Using Forged ICMP and ARP Spoofing

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

Using Forged Internet Control Message Protocol (ICMP)

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

Using Address Resolution Protocol (ARP) Spoofing

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

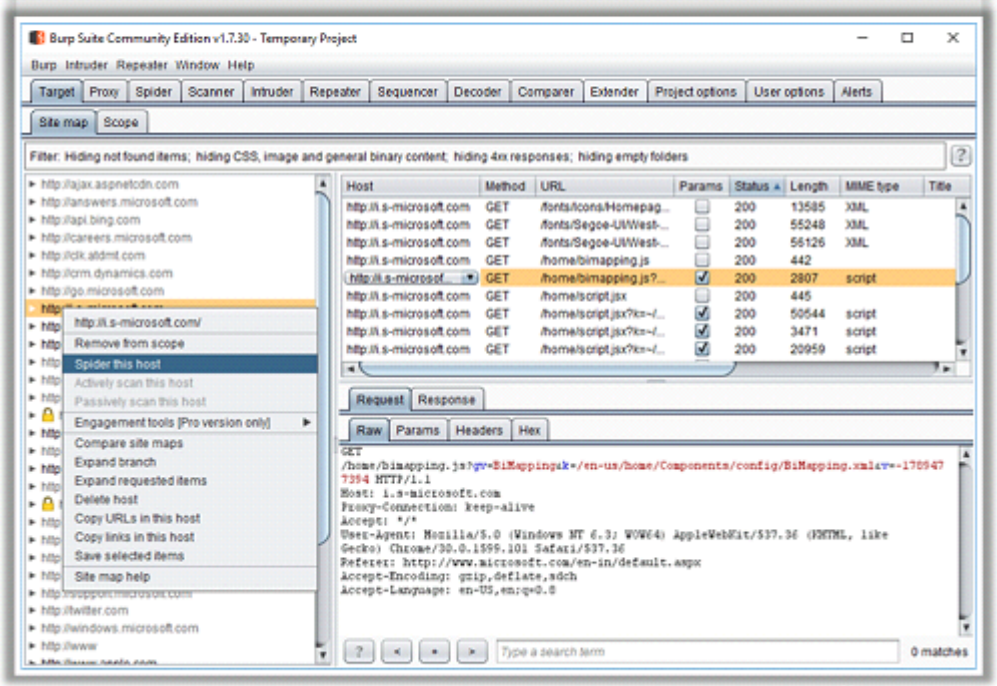
Module Flow

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

Session Hijacking Tools

Burp Suite

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

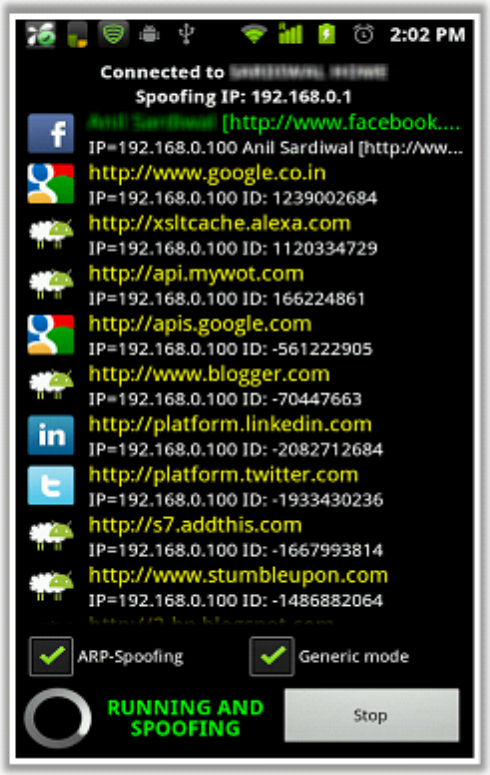


https://portswigger.net

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

Session Hijacking Tools for Mobile

DroidSheep

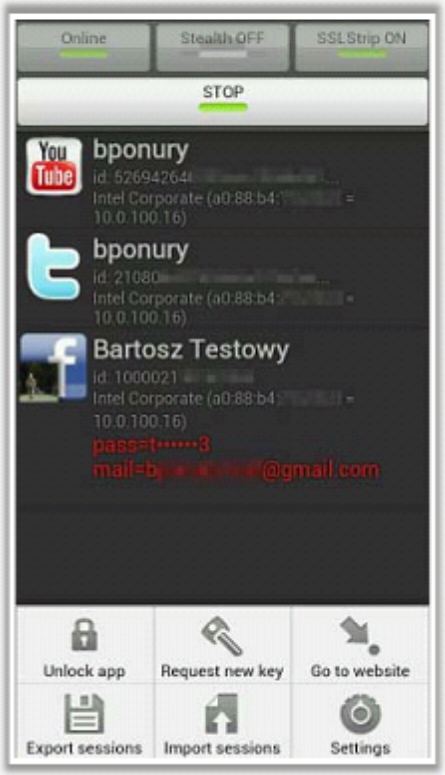


http://droidsheep.org

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

06 Penetration Testing

Session Hijacking Detection Methods

- Detection Method
  - Manual Method
    - Using Packet Sniffing Software
      - Normal Telnet Session
      - Forcing an ARP Entry
  - Automatic Method
    - Intrusion Detection Systems (IDS)
    - Intrusion Prevention Systems (IPS)

Protecting against Session Hijacking

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

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

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

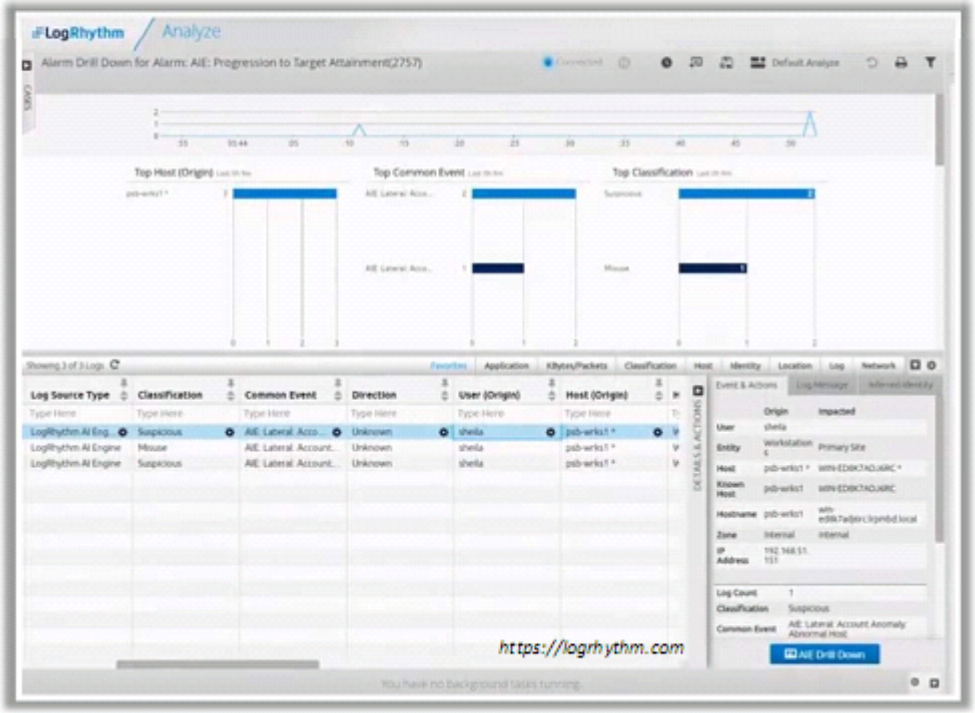
Methods to Prevent Session Hijacking: To be Followed by Web Users

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

Session Hijacking Detection Tools

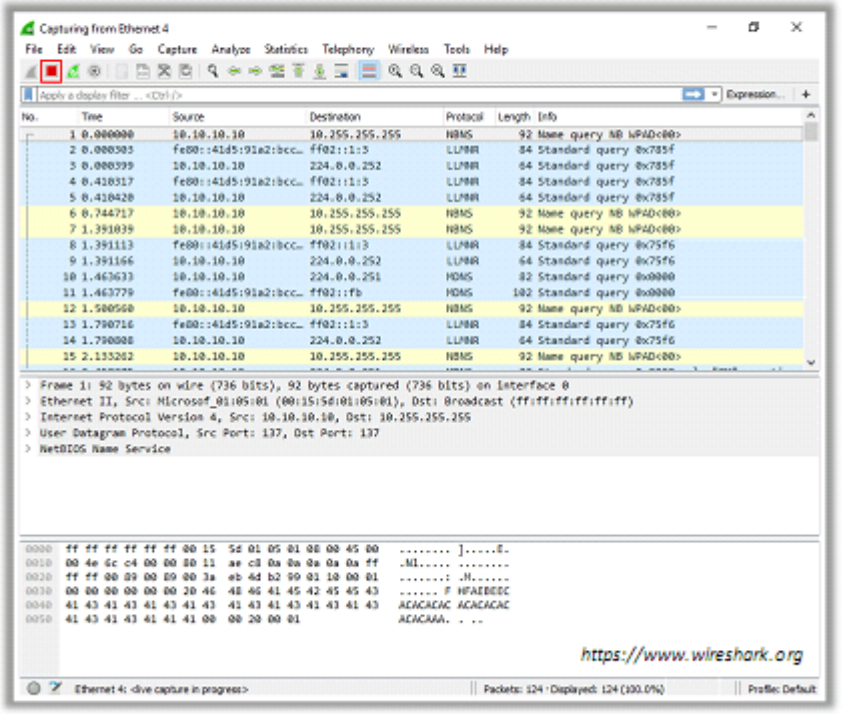
LogRhythm

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



Wireshark

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



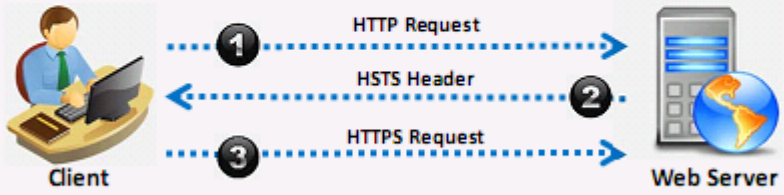
Approaches Vulnerable to Session Hijacking and their Preventative Solutions

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

Approaches to Prevent Session Hijacking

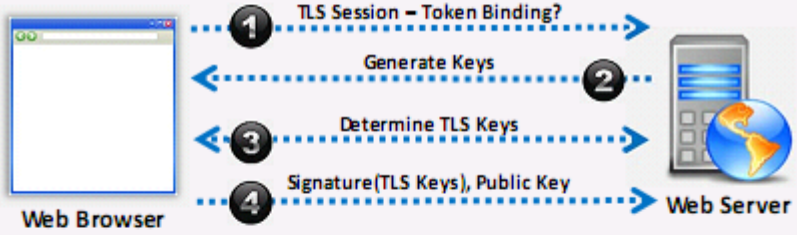
HTTP Strict Transport Security (HSTS)

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



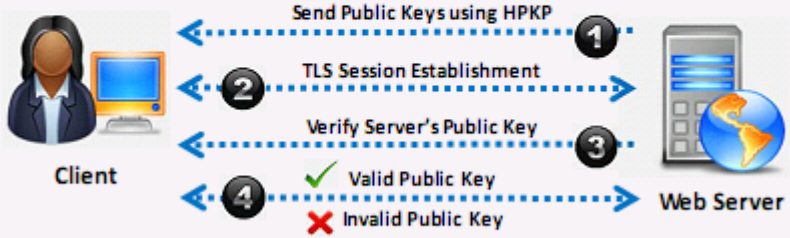
Token Binding

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



HTTP Public Key Pinning (HPKP)

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



IPSec

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

Components of IPsec

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

Benefits of IPSec

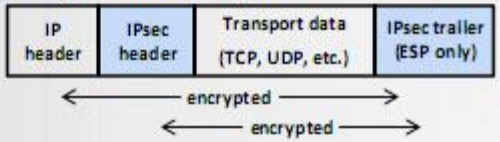
- Network-level peer authentication
- Data origin authentication
- Data integrity
- Data confidentiality (encryption)
- Replay protection

IPSec (Cont'd)

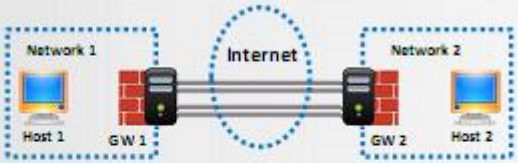
Modes of IPsec

Transport Mode

Transport – mode encapsulation

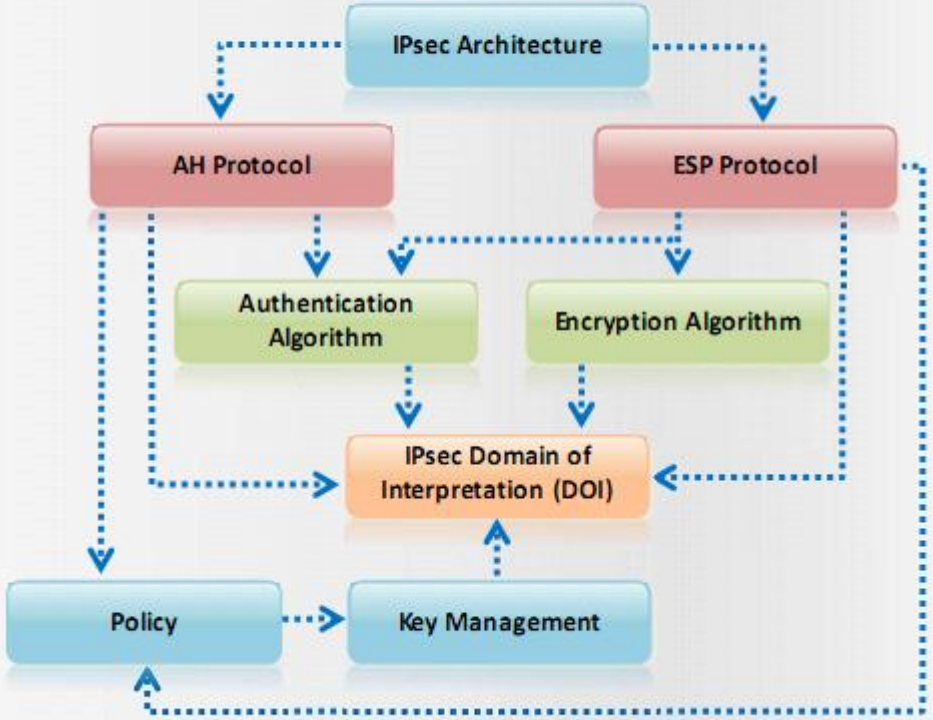


Tunnel Mode



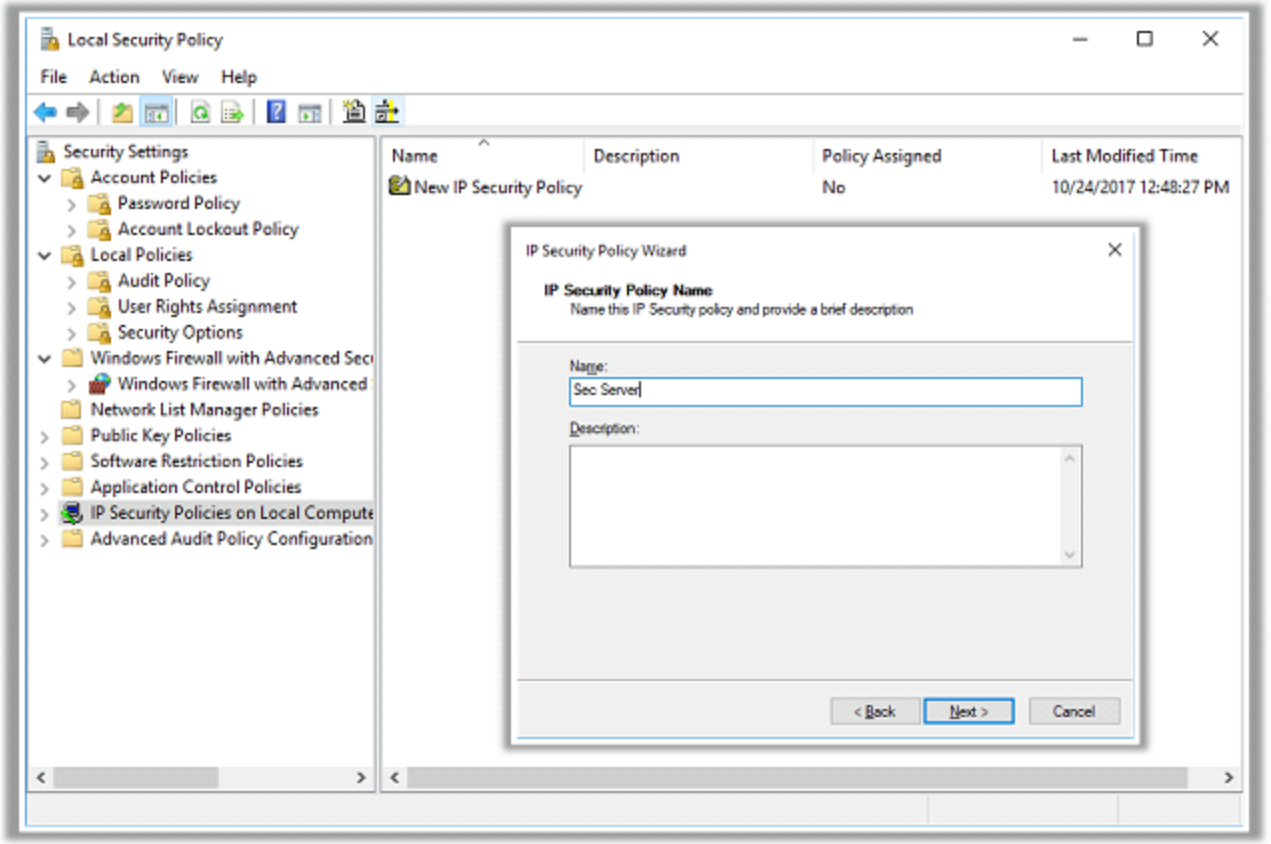
Tunnel – mode encapsulation

IPsec Architecture



IPsec Authentication and Confidentiality

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

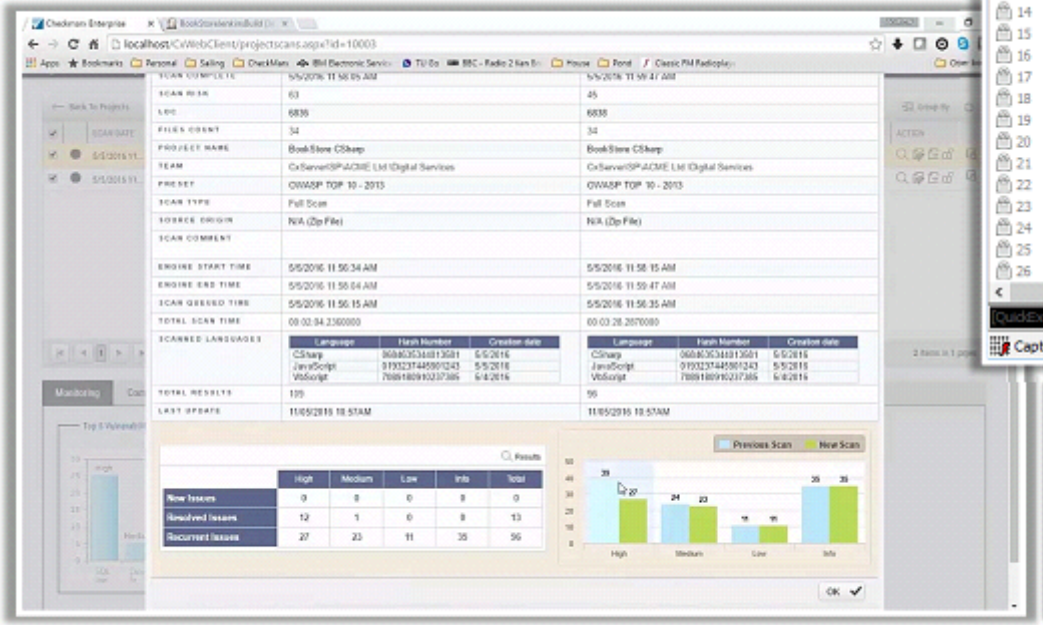


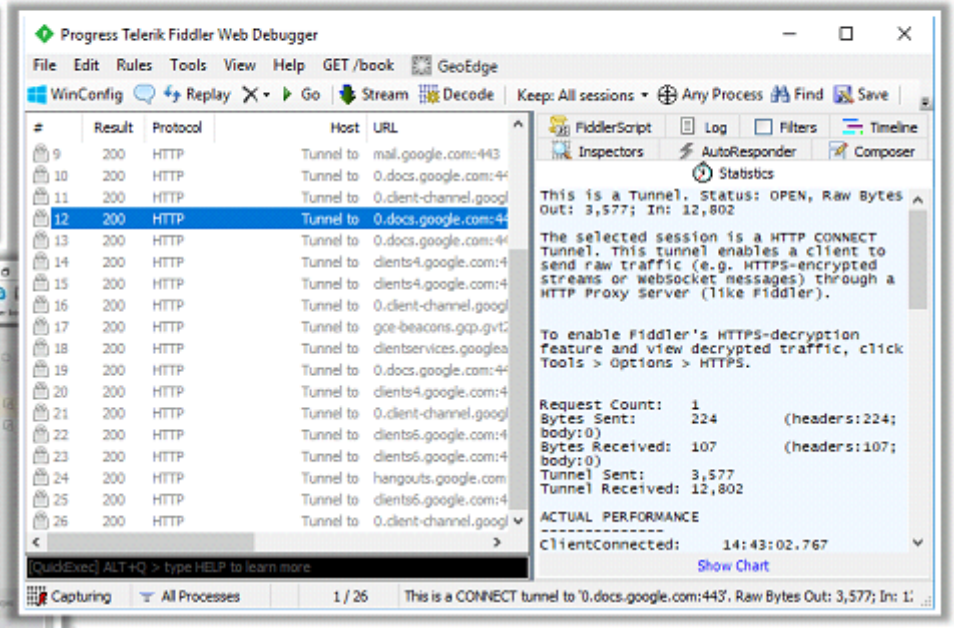
Session Hijacking Prevention Tools

CxSAST

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

https://www.checkmarx.com





Fiddler

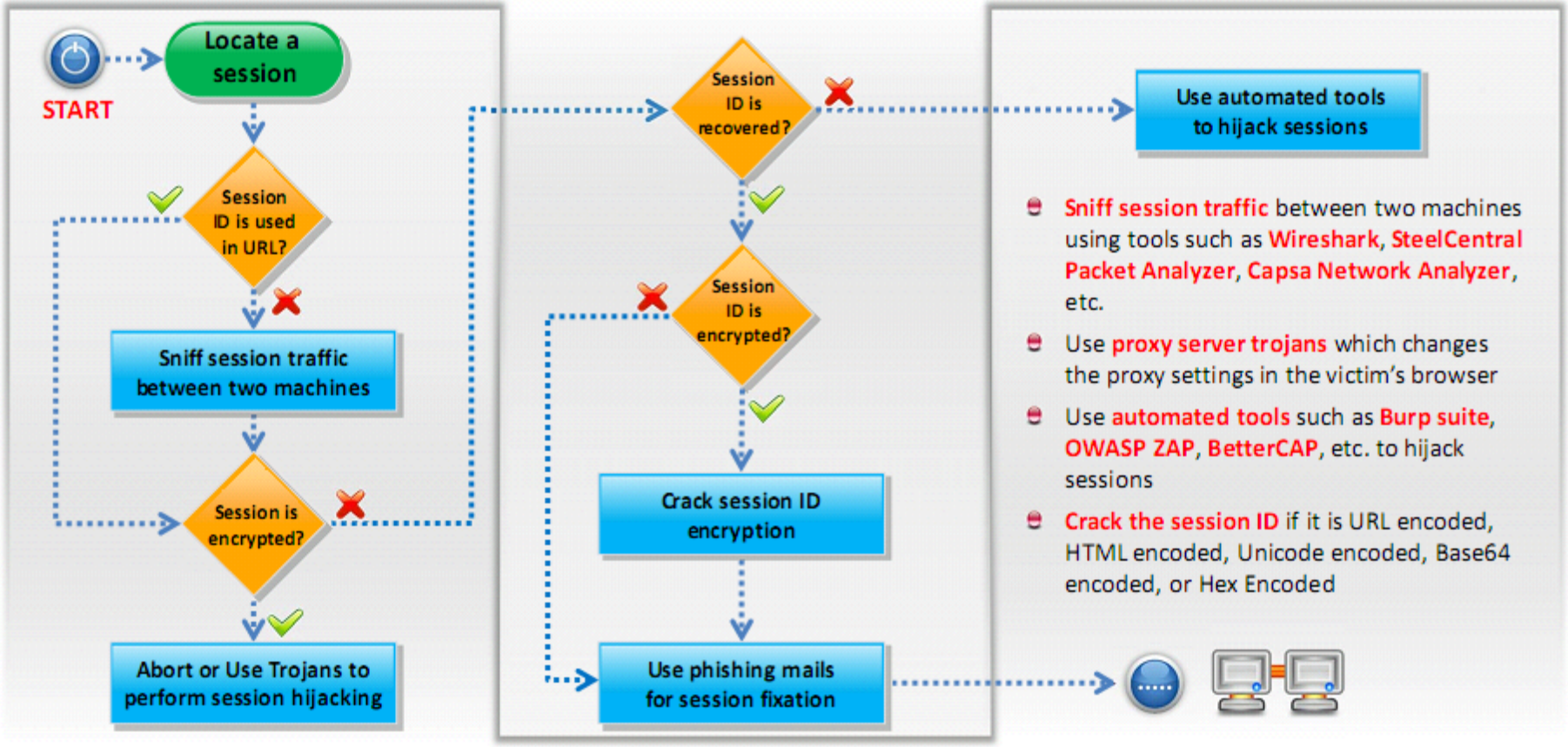
Fiddler is used for **security testing of web applications** such as decrypting HTTPS traffic, and manipulating requests using a man-in-the-middle decryption technique

https://www.telerik.com

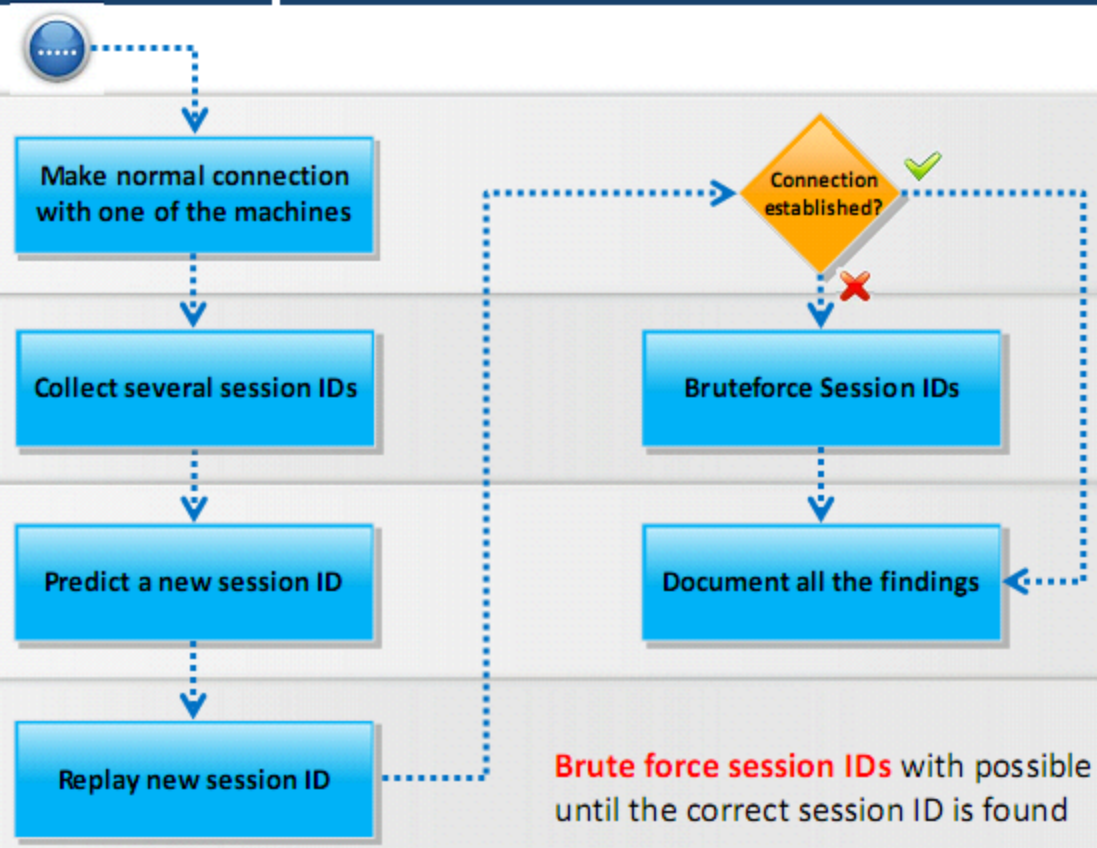
Module Flow

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

Session Hijacking Pen Testing



Session Hijacking Pen Testing (Cont'd)



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

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