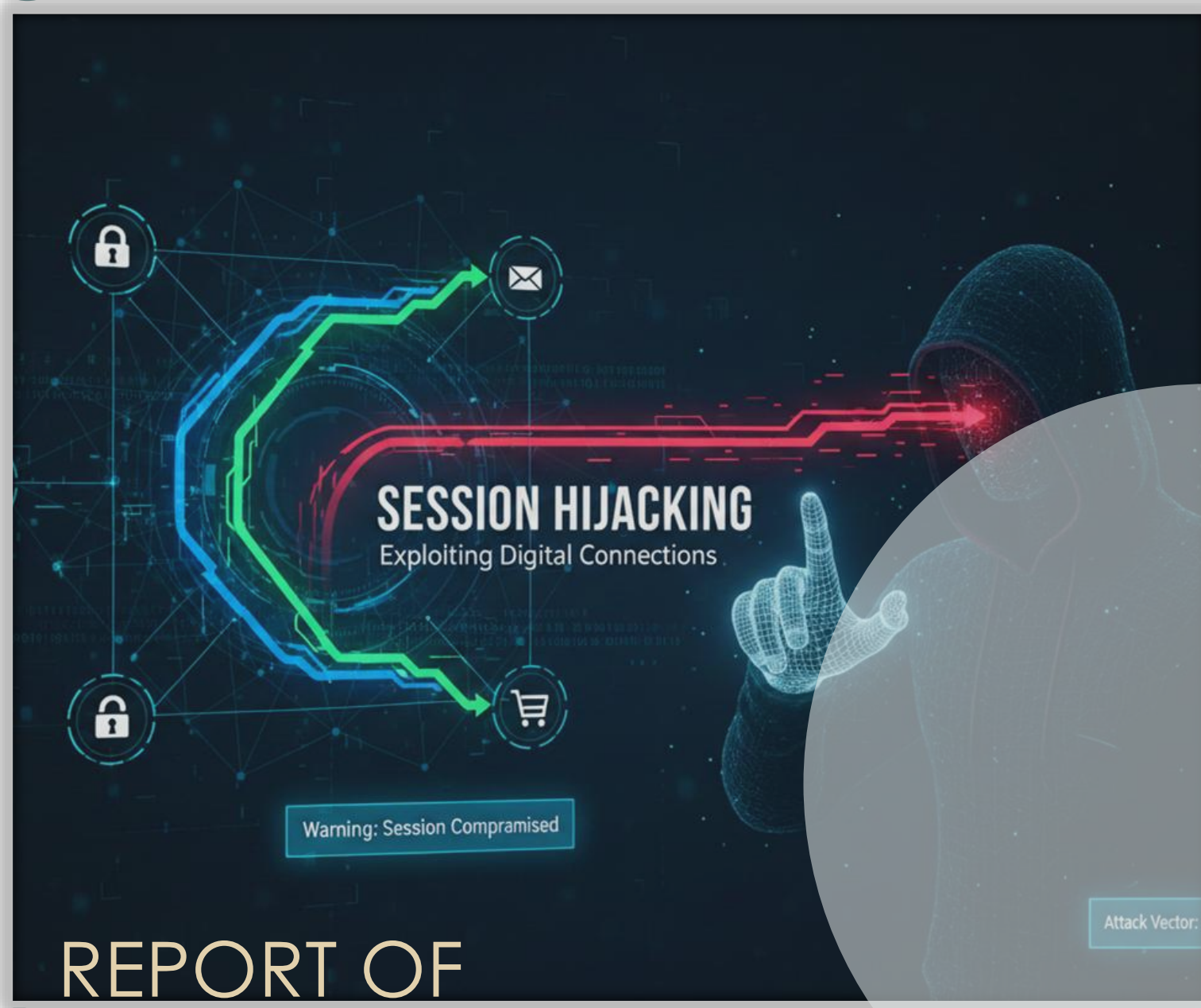


REPORT OF SESSION HIJACKING

BY SACHCHITANAND YADAV

SESSION HIJACKING
MODULE - 11

Learning Objectives -

- Explain Session Hijacking Concepts
- Perform application-level session hijacking
- Perform network-level session hijacking
- Use different session hijacking tools
- Explain Session Hijacking Countermeasures

Table of Contents

5.6 XSS Prevention
5.7 HTTP Strict Transport Security (HSTS)
5.8 Session Binding
5.9 Monitoring and Logging
5.10 Secure Logout Mechanism
5.11 User Awareness

---

6. Module Summary

6.1 Description

Session Hijacking Concepts: -

Session Hijacking

Session hijacking is a **security attack in which an attacker gains unauthorized control over a user's active session** by stealing or guessing the session token. Once compromised, the attacker can access the victim's account and perform actions as if they were the legitimate user—often without raising any alarms.

Core Idea

Web applications rely on sessions to preserve user authentication after login. Each session is identified by a **Session ID**, which acts like a temporary digital identity. If this identifier is exposed or predicted, an attacker can reuse it to impersonate the user and bypass authentication controls.

Types of Session Hijacking

- **Active Hijacking** – The attacker takes over an ongoing session and interacts with the application in real time.
- **Passive Hijacking** – The attacker silently monitors session data without immediately altering communication.
- **Sidejacking** – Session information is intercepted over unsecured networks, commonly public Wi-Fi.
- **XSS-Based Hijacking** – Session cookies are stolen using malicious scripts injected into web pages.
- **Session Fixation** – The attacker forces the victim to use a predefined session ID.
- **Man-in-the-Middle (MITM) Hijacking** – The attacker intercepts communication between the client and server.

Session Hijacking Lifecycle

1. **Target Discovery** – Identifying applications with weak session handling mechanisms.
2. **Session ID Capture** – Obtaining session tokens through methods such as:
    - Packet sniffing
    - Cross-site scripting
    - Predictable session identifiers
    - Session fixation
3. **Session Takeover** – Reusing the stolen session ID to impersonate the victim.
4. **Exploitation** – Accessing confidential data or executing unauthorized operations.

Common Session Hijacking Techniques

- Network sniffing
- Cross-Site Scripting (XSS)
- Session fixation
- Man-in-the-Middle attacks
- Brute-force guessing of session tokens
- Malware or trojan-based attacks

Real-World Case

Firesheep Attack (2010)
Firesheep was a browser extension that exposed how easily sessions could be hijacked over unsecured Wi-Fi networks using sidejacking techniques, bringing public attention to the importance of HTTPS.

Tools Commonly Used

- Ettercap
- Wireshark
- Burp Suite
- BeEF (Browser Exploitation Framework)
- Cookie Cadger
- Cain & Abel

Objective of the Session Hijacking Report

The main aim of this report is to **study and demonstrate session hijacking attacks in a controlled laboratory environment**. The experiment was conducted using virtual machines on a single host system, with **Kali Linux as the attacker platform** and **Windows 11 as the victim system**.

This exercise was designed to:

- Demonstrate how session hijacking can occur when session data such as cookies and authentication tokens are transmitted over unencrypted HTTP connections.
- Emphasize the security risks associated with insecure web communication.

- Illustrate the practical execution of MITM-based session hijacking attacks using tools such as Bettercap, SSLStrip, Cookie Cadger, and Ettercap.
- Educate students and cybersecurity professionals about the importance of implementing strong security measures like HTTPS, secure cookies, and HSTS.

This report serves as an **educational reference and awareness document**, highlighting the real-world impact of session hijacking and reinforcing the necessity of secure web application practices.

Session Hijacking Through Manual Cookie Theft

Session hijacking through manual cookie theft is a **straightforward yet highly effective attack technique** in which an attacker gains control of a user's active session by directly copying the session cookie from the victim's browser. This approach does not depend on automated hacking tools; instead, it involves **manual access to browser-stored session data**.

In this method, the attacker extracts the session ID using browser features such as cookie or through other forms of local or remote access. If a user logs into a web application and leaves the session active—particularly on shared systems or in public environments—an attacker with temporary access can easily retrieve the session cookie.

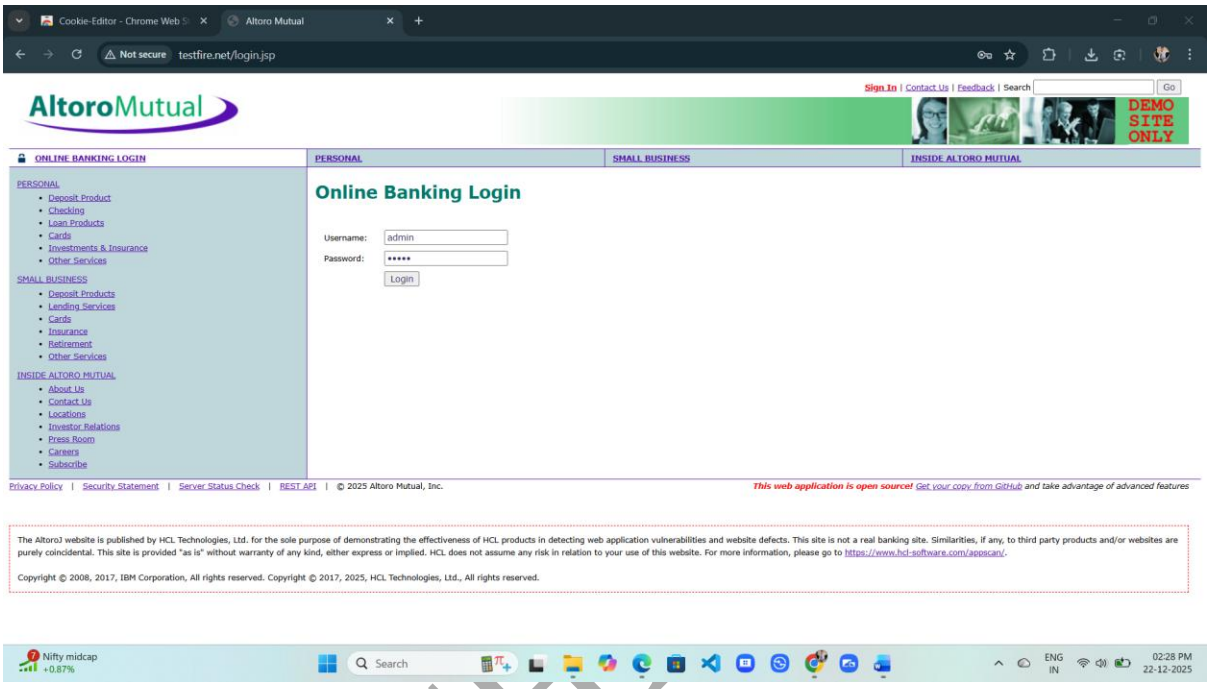
Once the session cookie is copied, it can be reused in another browser to **impersonate the legitimate user**, effectively bypassing authentication without requiring a username or password.

How It Is Performed

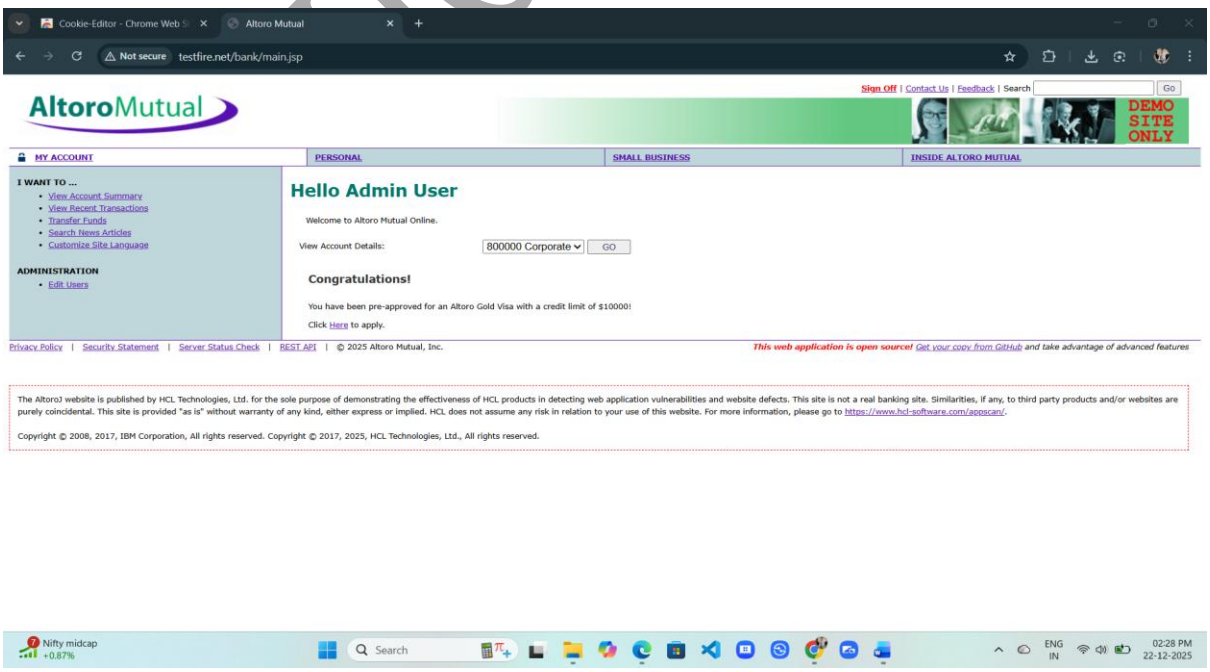
• Identify the target website and ensure the victim is logged in
• Access the victim's browser while the session is active
• Open browser developer tools and locate stored session cookies
• Copy the active session ID associated with the authenticated session
• Inject or replace the cookie in another browser to take over the session

How to do it -:

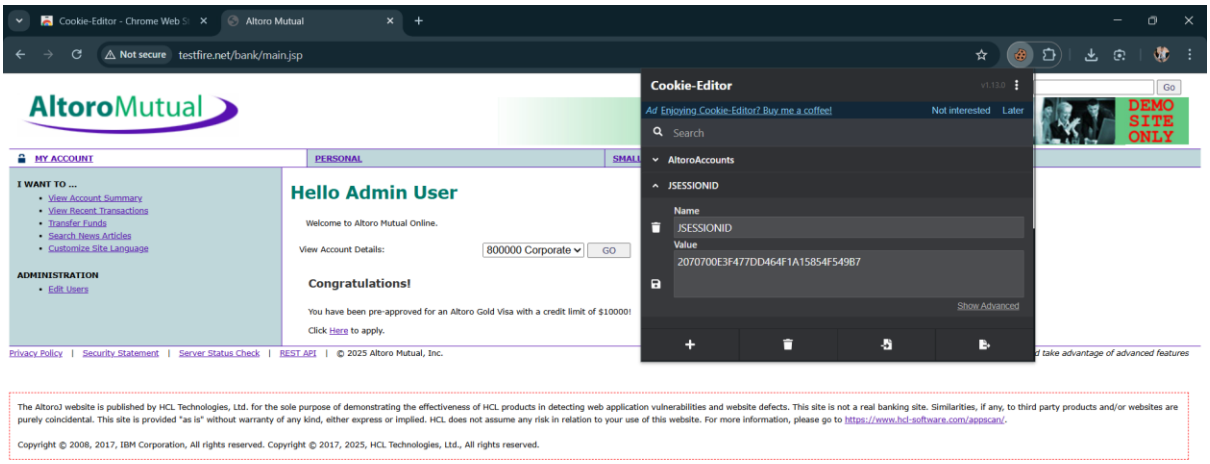
- Target Website
- Enter username and password
- Click on Login



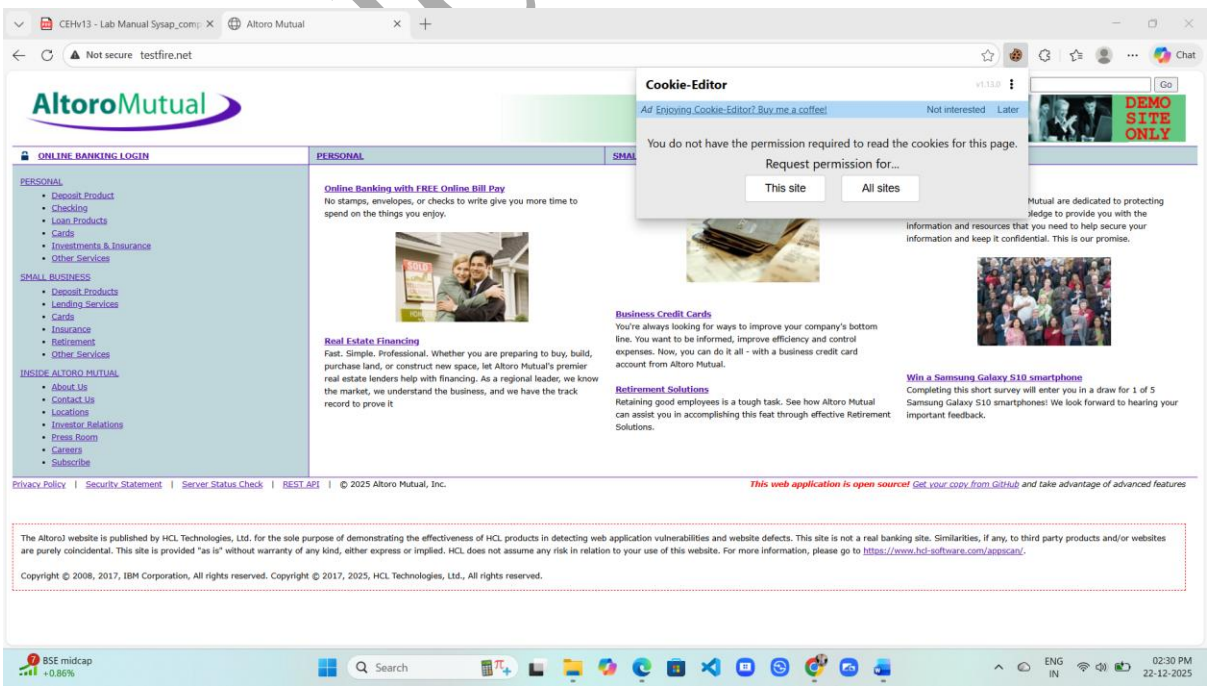
- Login Successful



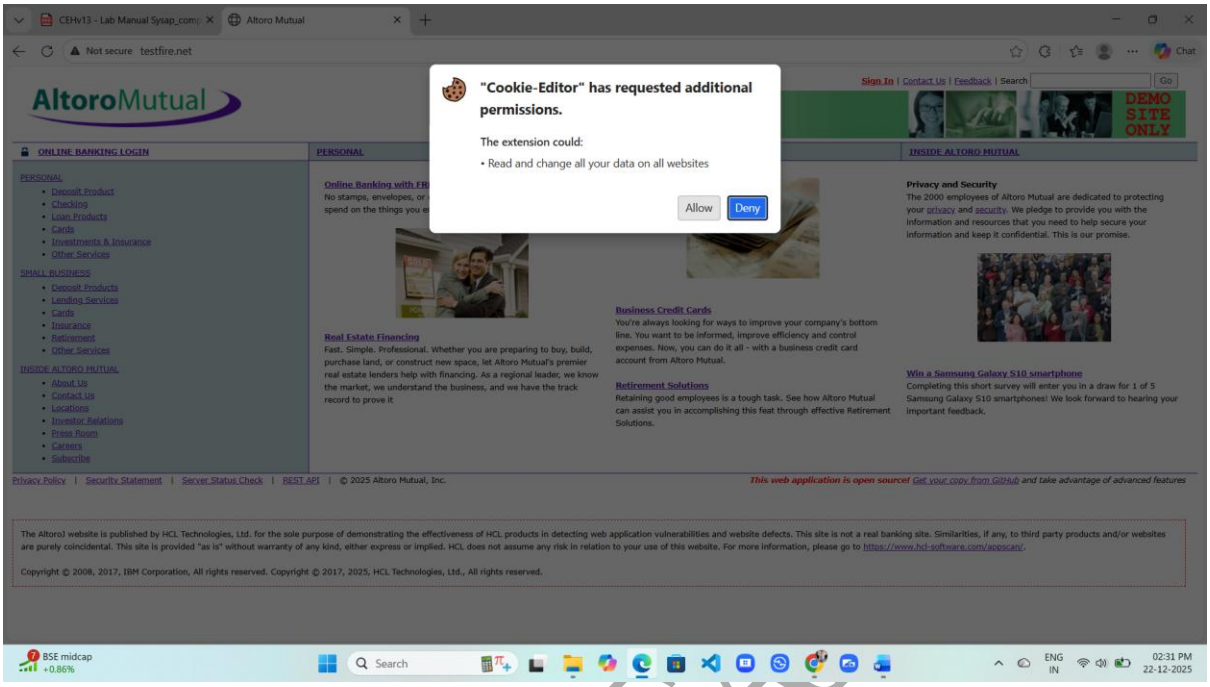
- Now , Click on Cookies Section
- Click on JSESSIONID
- Now, copy this session Id



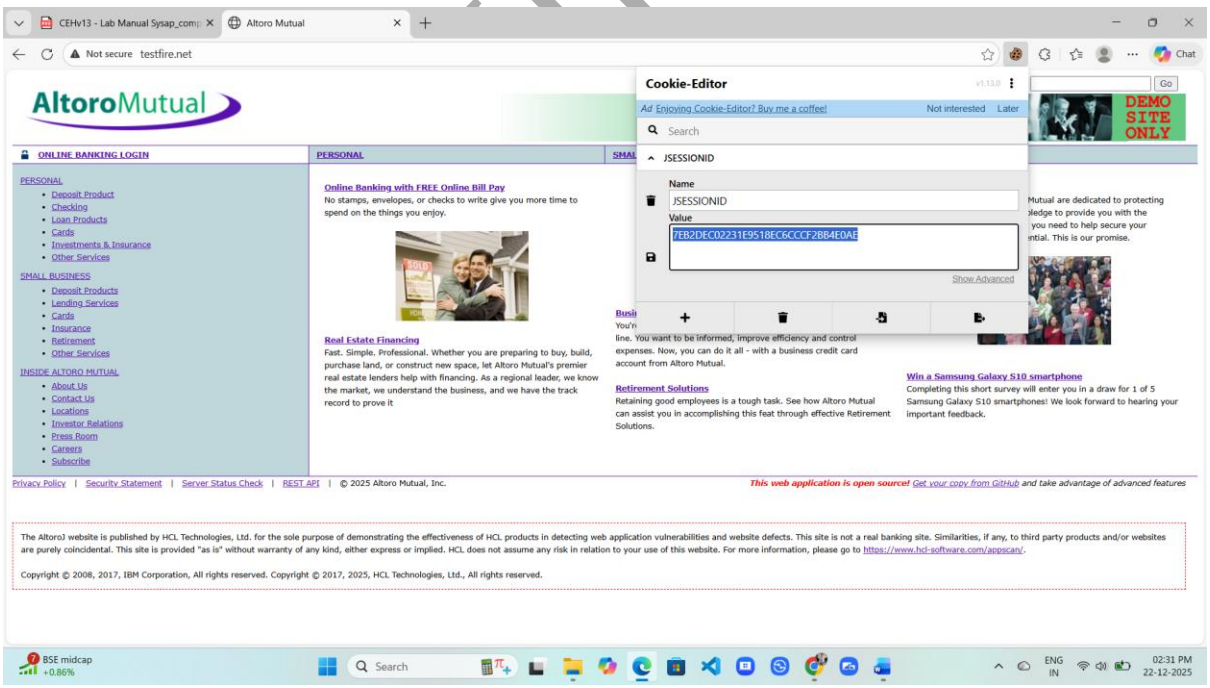
- **Now Switch to another browser and open same website**



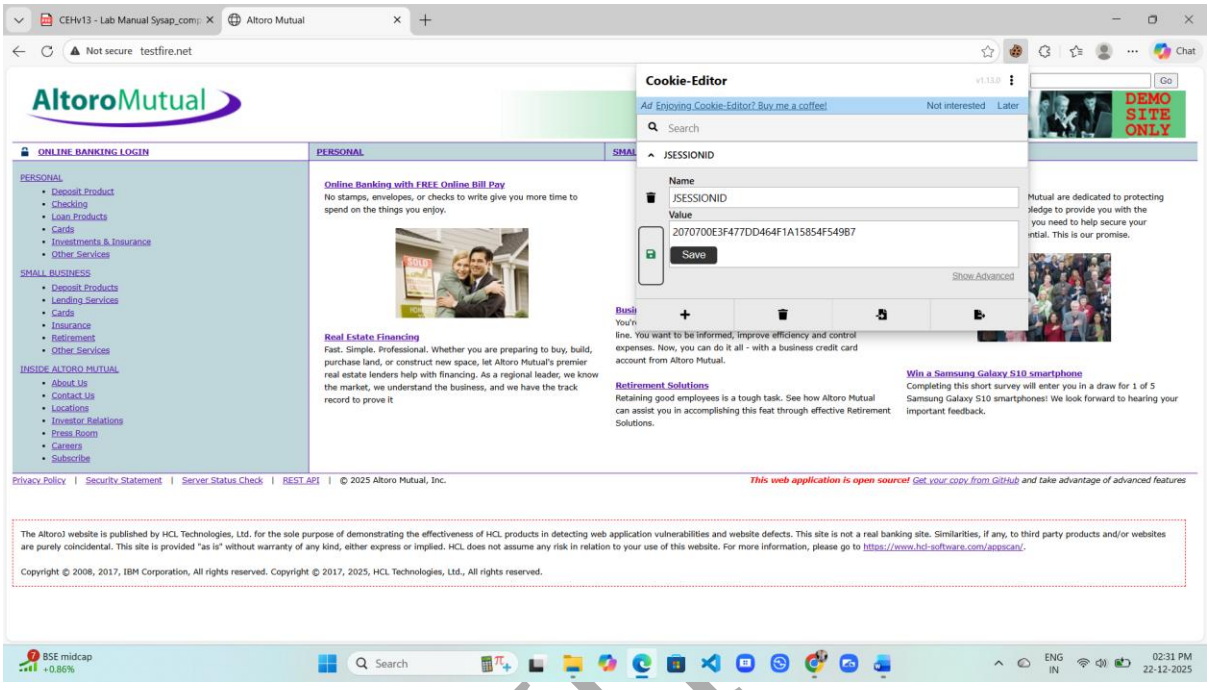
- Open cookies Editor Extension



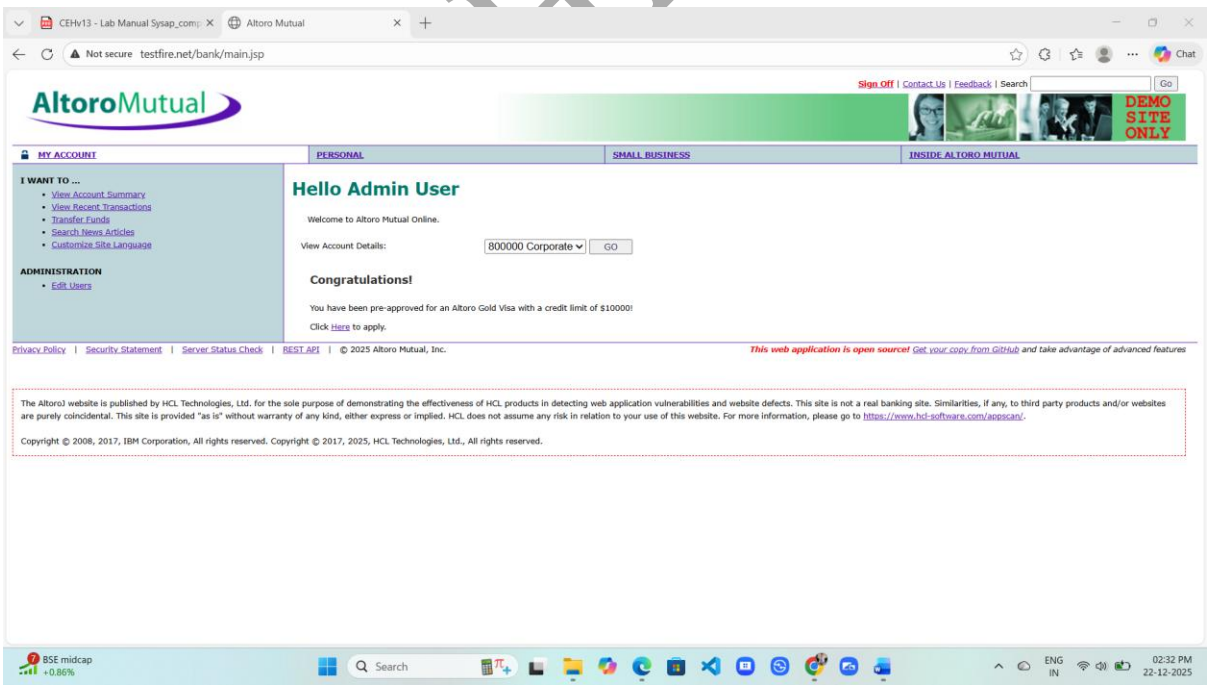
- And replace this session id



- To this and then save it and refresh the Browser



- **Login Successful**



Testing Session ID Predictability with Burp Suite Sequencer

Burp Suite Sequencer is a specialized testing component used to **analyze the quality and randomness of session identifiers** issued by web applications. By gathering a large sample of session tokens, the tool performs statistical tests to evaluate their entropy and determine whether the values follow any predictable patterns.

In this experiment, session cookies generated by the application were captured and examined to verify whether the session IDs were **strong, unique, and sufficiently unpredictable**. Session tokens that lack proper randomness may be vulnerable to guessing or brute-force attacks, potentially resulting in session hijacking and unauthorized access.

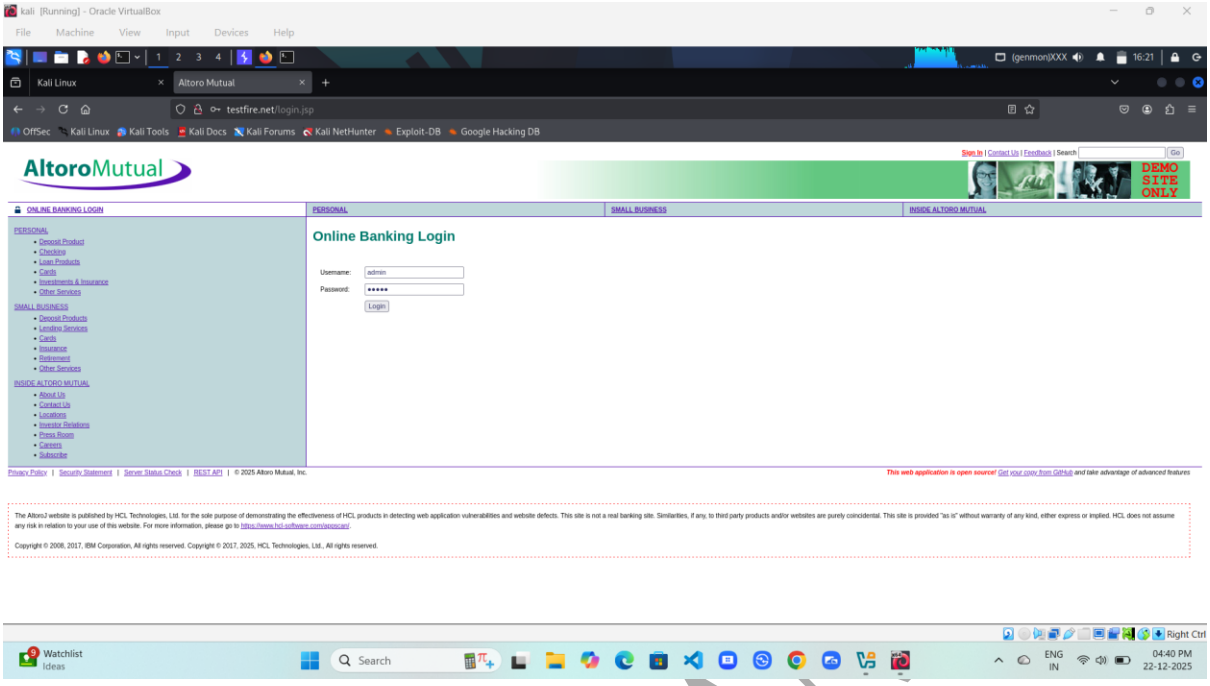
This assessment plays an important role in validating the **security of the application's session handling mechanism** and its ability to withstand session ID prediction attacks.

How to Use Burp Suite Sequencer

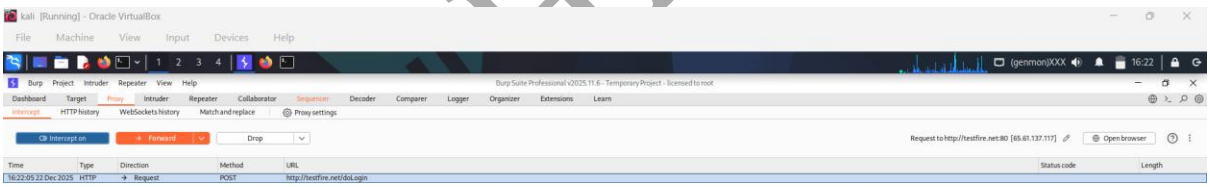
• Set the target website in Burp Suite and enable traffic interception.
• Interact with the application to generate multiple session IDs.
• Forward the captured session tokens to the Sequencer tool.
• Execute the analysis to measure randomness and entropy.
• Interpret the results to identify potential weaknesses in session ID generation.

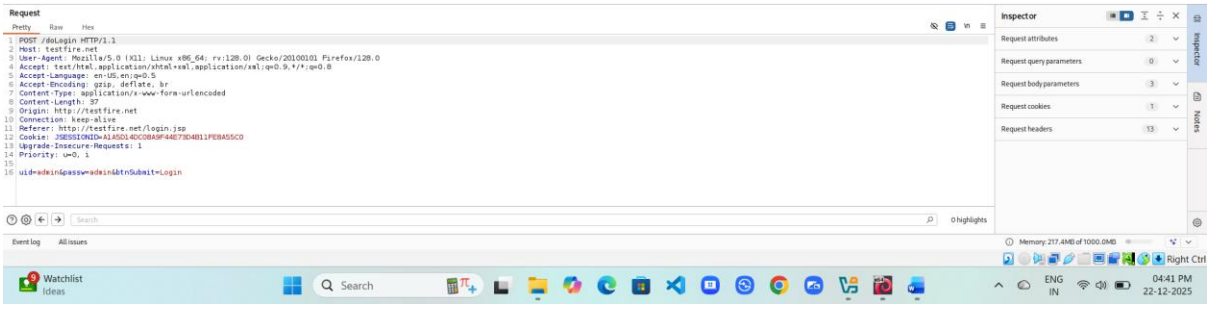
How to use it -:

- Target Website
- Enter username and password and click on login
- Account Login

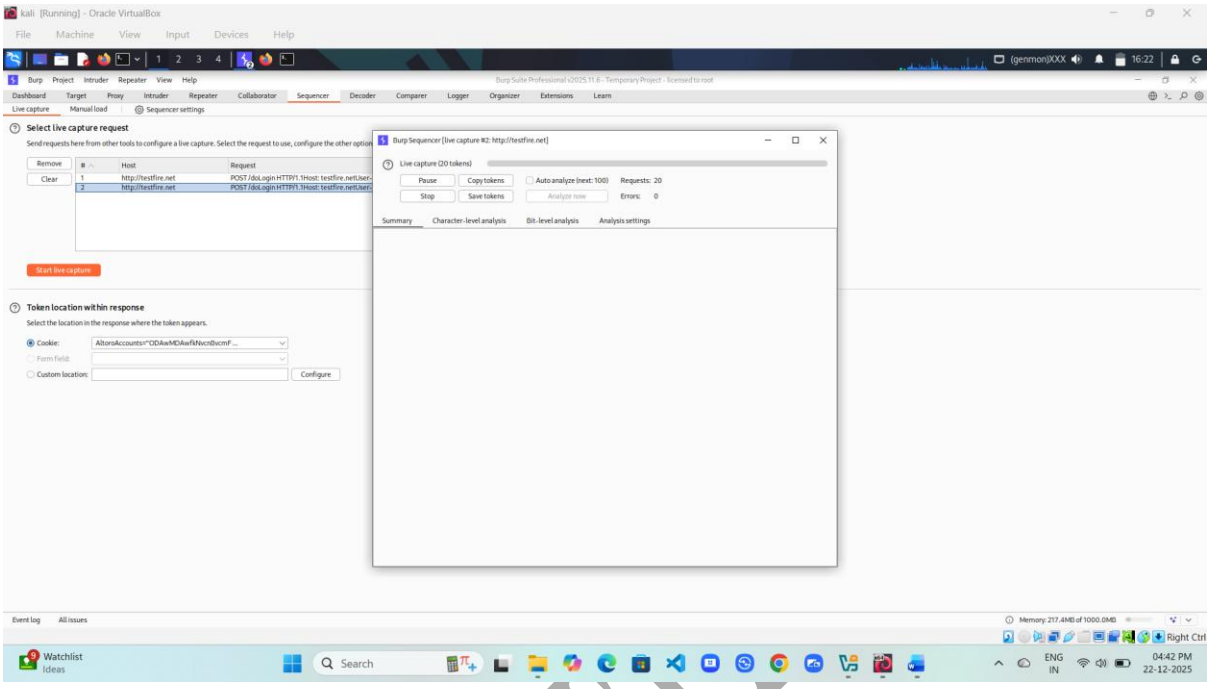


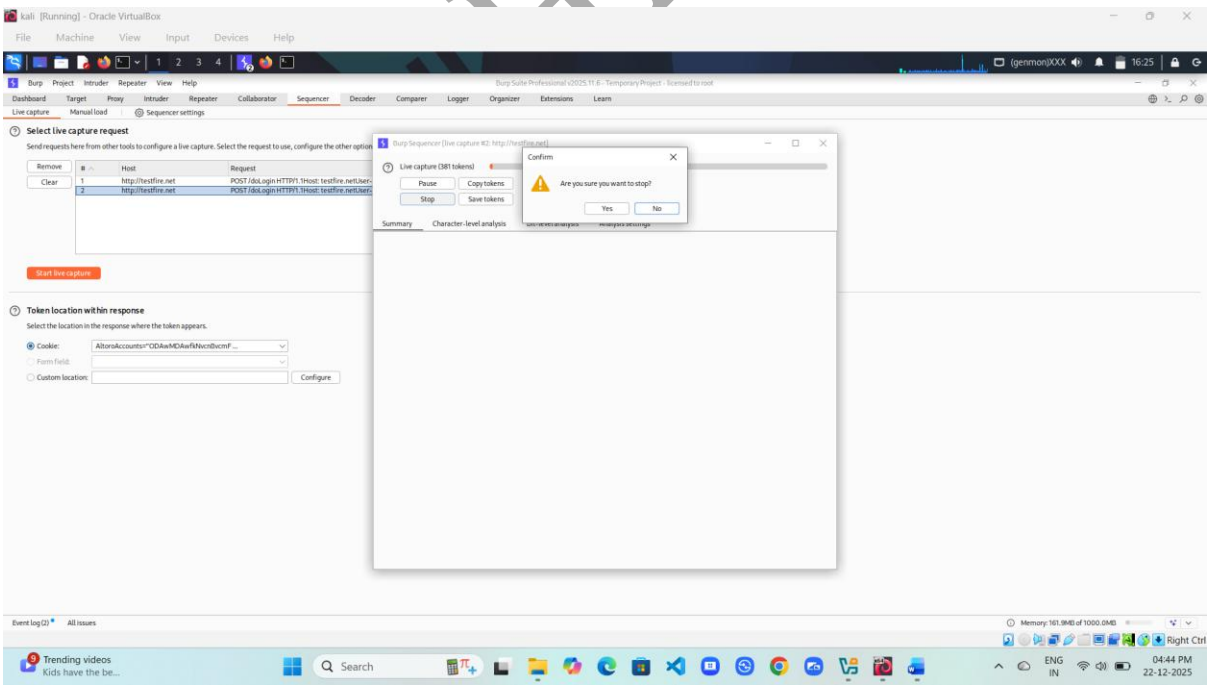
- Open burp suite and click on target option
- Send to Sequencer



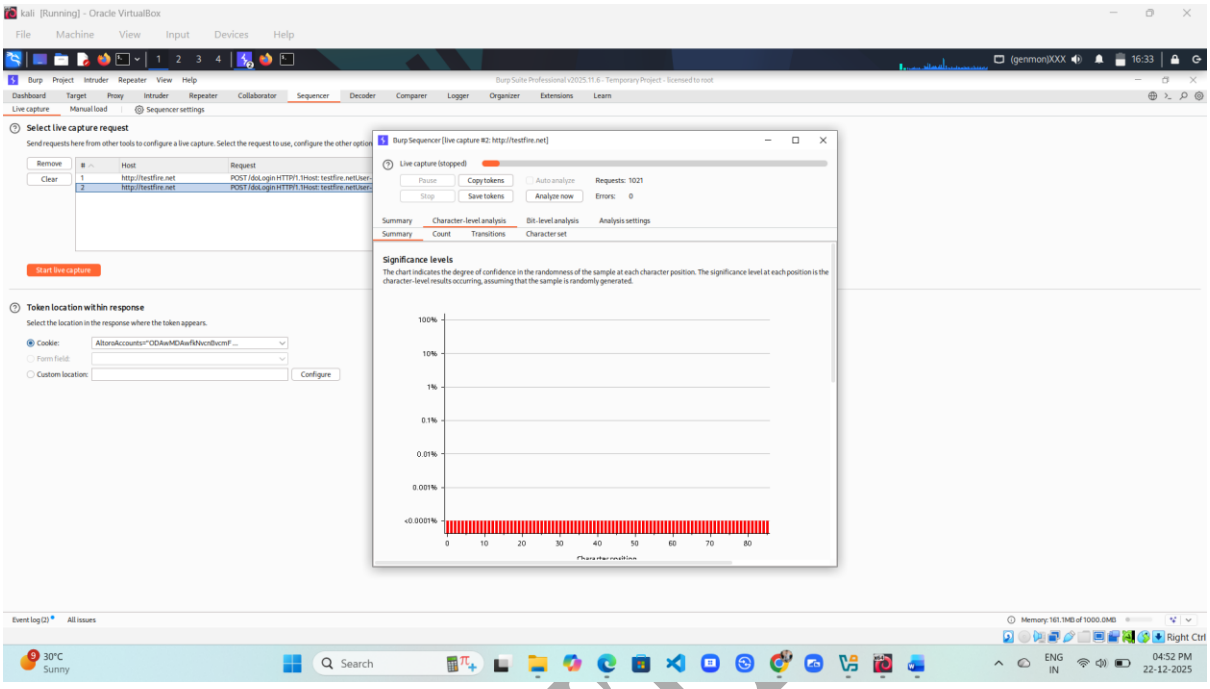


- Click on Start live capture
- Started





- Now click on Analyze now
- Here , result



Session Hijacking Using Bettercap Tool

Bettercap – Overview

Bettercap is a **powerful, modern, and flexible Man-in-the-Middle (MITM) attack framework** used for advanced network security testing. It enables attackers or ethical security professionals to **intercept, analyze, and manipulate network traffic in real time**.

Widely regarded as the **next-generation successor to Ettercap**, Bettercap is designed to work efficiently with modern networks and protocols, offering modularity, automation, and high performance.

Features of Bettercap

ARP Spoofing
Allows the attacker to redirect network traffic by impersonating legitimate devices on the network, effectively positioning the attacker as a MITM.

SSL Stripping
Forces encrypted HTTPS connections to downgrade to HTTP, making it possible to inspect otherwise encrypted traffic.

Session Hijacking
Captures active session cookies or authentication tokens, enabling unauthorized access to user sessions.

Live Traffic Sniffing
Provides real-time packet capture and deep inspection of network traffic.

Modular Architecture
Supports multiple attack modules and scripting capabilities, allowing flexible and customized attack execution.

Role of Bettercap in Session Hijacking

Bettercap plays a critical role in session hijacking attacks by combining MITM positioning with real-time traffic manipulation.

Step 1: Perform ARP spoofing to place the attacker between the victim and the network gateway.

Step 2: Enable live traffic sniffing to monitor network packets.
Step 3: Apply SSL stripping techniques, if required, to downgrade HTTPS traffic.
Step 4: Capture session cookies or authentication tokens transmitted over the network.
Step 5: Reuse the captured cookies to hijack active user sessions.

How to Use Bettercap

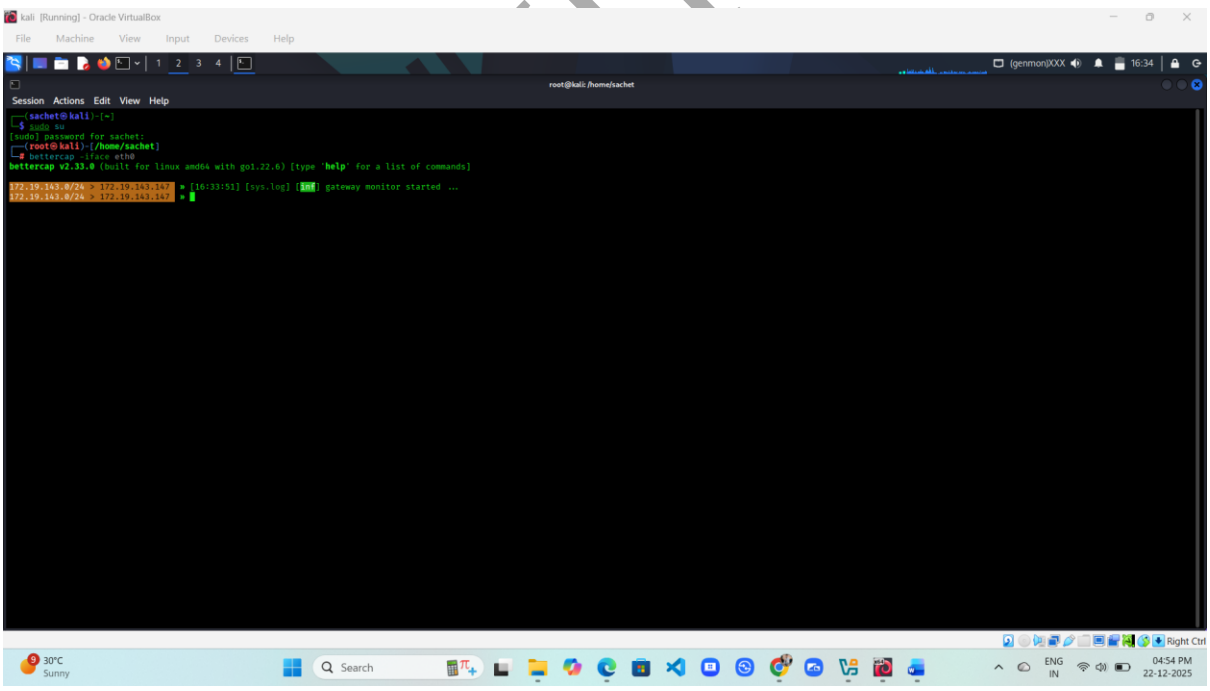
Command to Start Bettercap

```
sudo bettercap -iface eth0
```

Explanation: -

This command launches Bettercap with administrative privileges on the specified network interface (`eth0`). It allows Bettercap to intercept and manipulate network traffic flowing through that interface. The interface name may vary depending on the system configuration.

- Bettercap Started



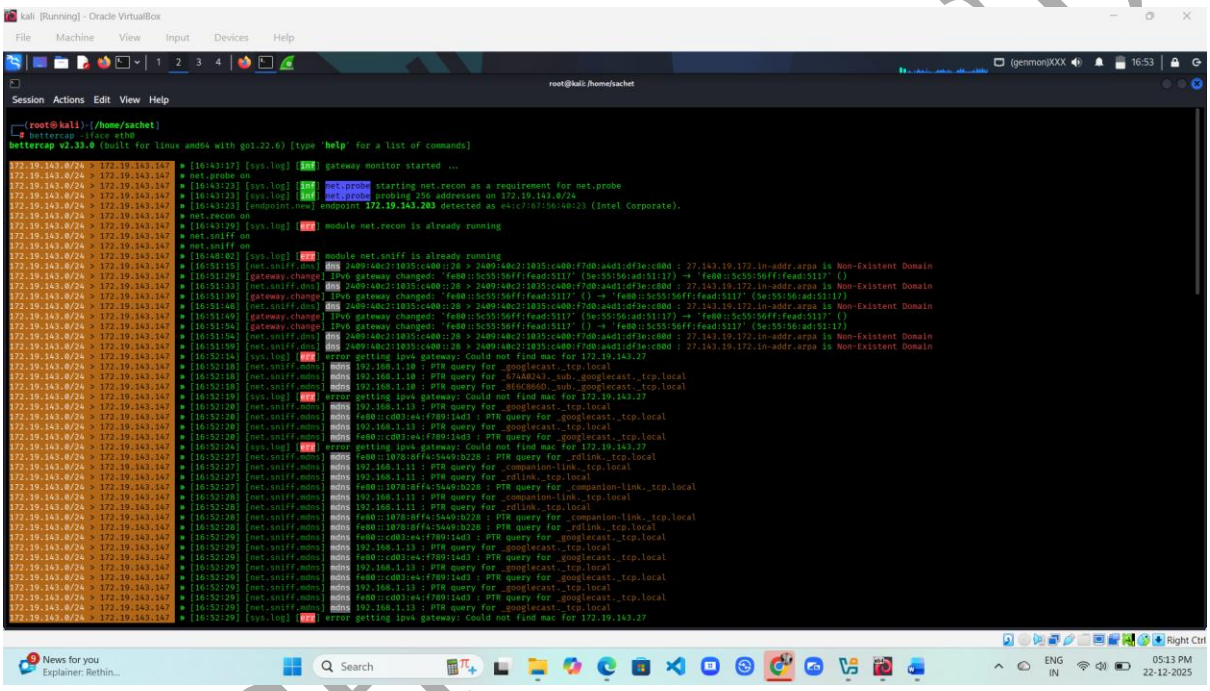
- Use Next Command

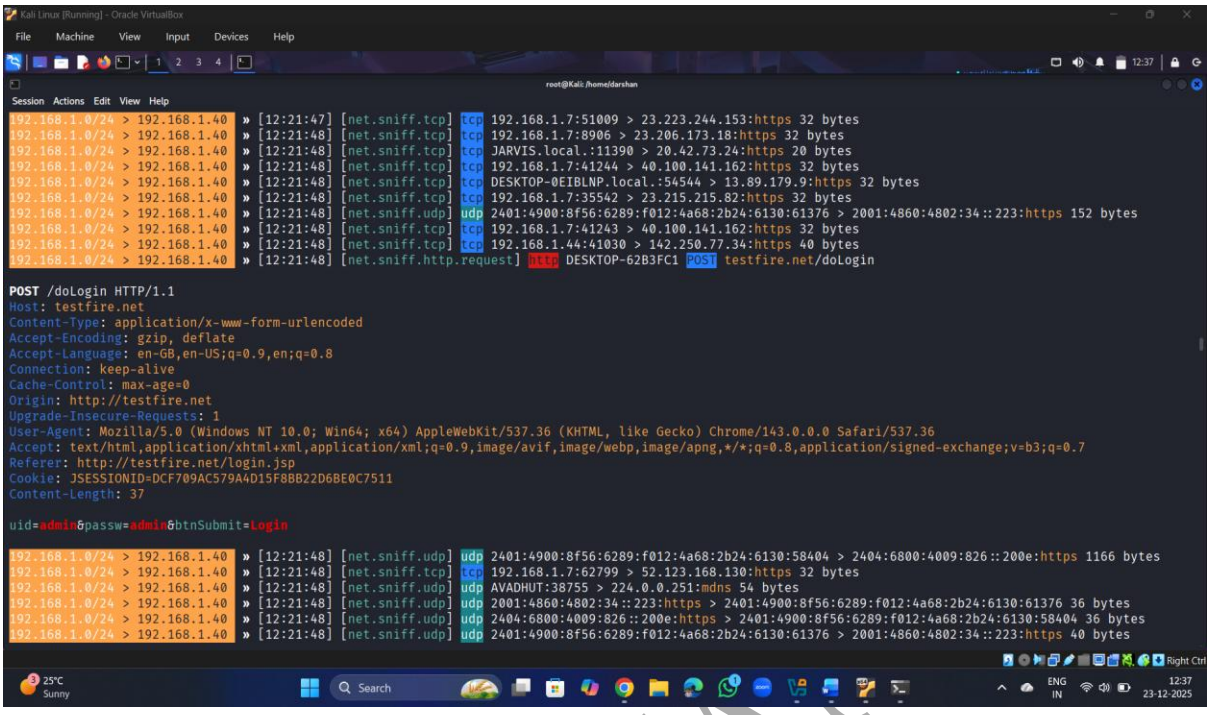
Command: - net.probe on

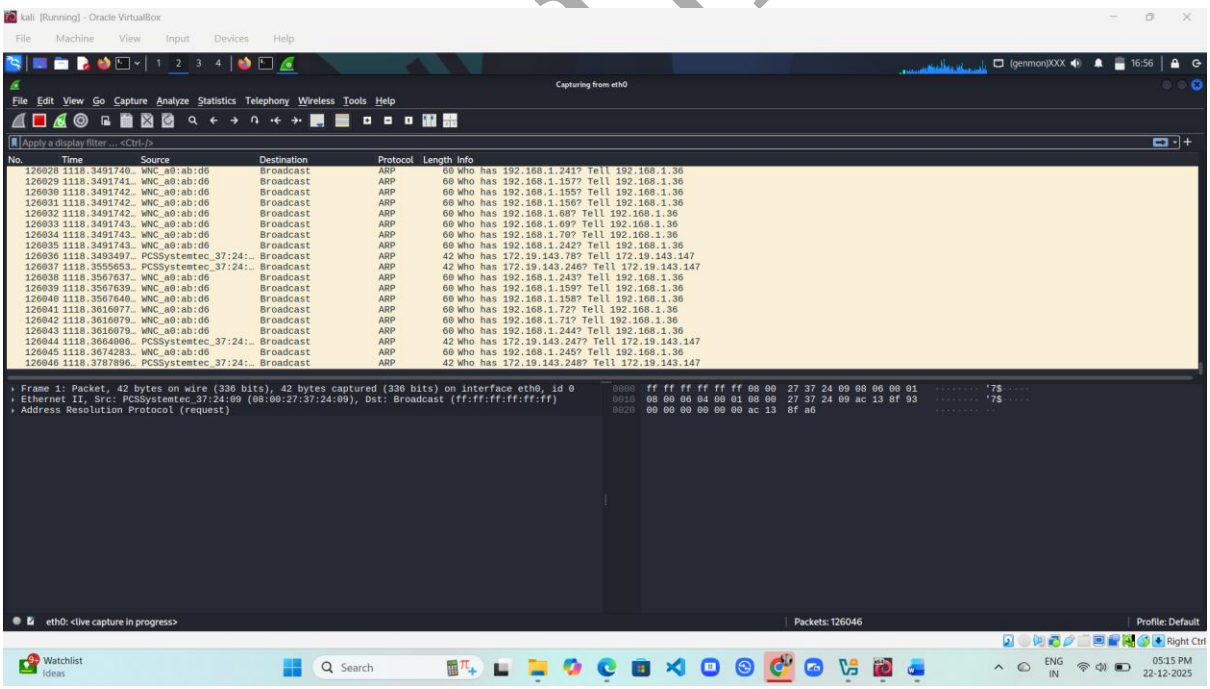
- Next Command

Command: - net.sniff on

Explanation: - Start ARP sniffing




Kali Linux [Running] - Oracle VirtualBox
File Machine View Input Devices Help
root@Kali: /home/darshan
Session Actions Edit View Help
192.168.1.0/24 > 192.168.1.40 » [12:21:47] [net.sniff.tcp] tcp 192.168.1.7:51009 > 23.223.244.153:https 32 bytes
192.168.1.0/24 > 192.168.1.40 » [12:21:48] [net.sniff.tcp] tcp 192.168.1.7:8906 > 23.206.173.18:https 32 bytes
192.168.1.0/24 > 192.168.1.40 » [12:21:48] [net.sniff.tcp] tcp JARVIS.local.:11390 > 20.42.73.24:https 20 bytes
192.168.1.0/24 > 192.168.1.40 » [12:21:48] [net.sniff.tcp] tcp 192.168.1.7:41244 > 40.100.141.162:https 32 bytes
192.168.1.0/24 > 192.168.1.40 » [12:21:48] [net.sniff.tcp] tcp DESKTOP-0EIBLNP.local.:54544 > 13.89.179.9:https 32 bytes
192.168.1.0/24 > 192.168.1.40 » [12:21:48] [net.sniff.tcp] tcp 192.168.1.7:35542 > 23.215.215.82:https 32 bytes
192.168.1.0/24 > 192.168.1.40 » [12:21:48] [net.sniff.udp] udp 2401:4900:8f56:6289:f012:4a68:2b24:6130:61376 > 2001:4860:4802:34::223:https 152 bytes
192.168.1.0/24 > 192.168.1.40 » [12:21:48] [net.sniff.tcp] tcp 192.168.1.7:41243 > 40.100.141.162:https 32 bytes
192.168.1.0/24 > 192.168.1.40 » [12:21:48] [net.sniff.tcp] tcp 192.168.1.44:41030 > 142.250.77.34:https 40 bytes
192.168.1.0/24 > 192.168.1.40 » [12:21:48] [net.sniff.http.request] HTTP DESKTOP-62B3FC1 POST testfire.net/doLogin
POST /doLogin HTTP/1.1
Host: testfire.net
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Connection: keep-alive
Cache-Control: max-age=0
Origin: http://testfire.net
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://testfire.net/login.jsp
Cookie: JSESSIONID=DCF709AC579A4D15F8BB22D6BE0C7511
Content-Length: 37
uid=admin&passw=admin&btnSubmit=Login
192.168.1.0/24 > 192.168.1.40 » [12:21:48] [net.sniff.udp] udp 2401:4900:8f56:6289:f012:4a68:2b24:6130:58404 > 2404:6800:4009:826::200e:https 1166 bytes
192.168.1.0/24 > 192.168.1.40 » [12:21:48] [net.sniff.tcp] tcp 192.168.1.7:62799 > 52.123.168.130:https 32 bytes
192.168.1.0/24 > 192.168.1.40 » [12:21:48] [net.sniff.udp] udp AVADHUT:38755 > 224.0.0.251:mdns 54 bytes
192.168.1.0/24 > 192.168.1.40 » [12:21:48] [net.sniff.udp] udp 2001:4860:4802:34::223:https > 2401:4900:8f56:6289:f012:4a68:2b24:6130:61376 36 bytes
192.168.1.0/24 > 192.168.1.40 » [12:21:48] [net.sniff.udp] udp 2404:6800:4009:826::200e:https > 2401:4900:8f56:6289:f012:4a68:2b24:6130:58404 36 bytes
192.168.1.0/24 > 192.168.1.40 » [12:21:48] [net.sniff.udp] udp 2401:4900:8f56:6289:f012:4a68:2b24:6130:61376 > 2001:4860:4802:34::223:https 40 bytes
25°C Sunny
Search
ENG IN
12:37 23-12-2025


kali [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Capturing from eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
126028 1118.3491740.. WNC_a0:ab:d6 Broadcast ARP 60 Who has 192.168.1.241? Tell 192.168.1.36
126029 1118.3491741.. WNC_a0:ab:d6 Broadcast ARP 60 Who has 192.168.1.157? Tell 192.168.1.36
126030 1118.3491742.. WNC_a0:ab:d6 Broadcast ARP 60 Who has 192.168.1.155? Tell 192.168.1.36
126031 1118.3491742.. WNC_a0:ab:d6 Broadcast ARP 60 Who has 192.168.1.156? Tell 192.168.1.36
126032 1118.3491742.. WNC_a0:ab:d6 Broadcast ARP 60 Who has 192.168.1.68? Tell 192.168.1.36
126033 1118.3491743.. WNC_a0:ab:d6 Broadcast ARP 60 Who has 192.168.1.69? Tell 192.168.1.36
126034 1118.3491743.. WNC_a0:ab:d6 Broadcast ARP 60 Who has 192.168.1.70? Tell 192.168.1.36
126035 1118.3491743.. WNC_a0:ab:d6 Broadcast ARP 60 Who has 192.168.1.242? Tell 192.168.1.36
126036 1118.3493497.. PCSSystemtec_37:24:.. Broadcast ARP 42 Who has 172.19.143.78? Tell 172.19.143.147
126037 1118.3555653.. PCSSystemtec_37:24:.. Broadcast ARP 42 Who has 172.19.143.246? Tell 172.19.143.147
126038 1118.3567637.. WNC_a0:ab:d6 Broadcast ARP 60 Who has 192.168.1.243? Tell 192.168.1.36
126039 1118.3567639.. WNC_a0:ab:d6 Broadcast ARP 60 Who has 192.168.1.159? Tell 192.168.1.36
126040 1118.3567640.. WNC_a0:ab:d6 Broadcast ARP 60 Who has 192.168.1.158? Tell 192.168.1.36
126041 1118.3616077.. WNC_a0:ab:d6 Broadcast ARP 60 Who has 192.168.1.72? Tell 192.168.1.36
126042 1118.3616079.. WNC_a0:ab:d6 Broadcast ARP 60 Who has 192.168.1.71? Tell 192.168.1.36
126043 1118.3616079.. WNC_a0:ab:d6 Broadcast ARP 60 Who has 192.168.1.244? Tell 192.168.1.36
126044 1118.3664006.. PCSSystemtec_37:24:.. Broadcast ARP 42 Who has 172.19.143.247? Tell 172.19.143.147
126045 1118.3674283.. WNC_a0:ab:d6 Broadcast ARP 60 Who has 192.168.1.245? Tell 192.168.1.36
126046 1118.3787896.. PCSSystemtec_37:24:.. Broadcast ARP 42 Who has 172.19.143.248? Tell 172.19.143.147
Frame 1: Packet, 42 bytes on wire (336 bits), 42 bytes captured (336 bits) on interface eth0, id 0
Ethernet II, Src: PCSSystemtec_37:24:09 (08:00:27:37:24:09), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Address Resolution Protocol (request)
0000 ff ff ff ff ff ff 08 00 27 37 24 09 08 06 00 01
0010 08 00 06 04 00 01 08 00 27 37 24 09 ac 13 8f 93
0020 00 00 00 00 00 00 ac 13 8f a6
eth0: <live capture in progress>
Packets: 126046
Profile: Default
Watchlist Ideas
Search
ENG IN
05:15 PM 22-12-2025

Session Hijacking Countermeasures

1. Use Secure Communication (HTTPS / SSL / TLS)

CEH strongly emphasizes encrypting all sensitive communication using **HTTPS**. Encryption ensures that session cookies and authentication tokens cannot be captured through packet sniffing or Man-in-the-Middle (MITM) attacks.

- Prevents sniffing and sidejacking
- Protects session data on public or untrusted networks
- Mandatory for login pages and authenticated areas

👉 CEH truth: HTTP is a liability, not a feature.

---

2. Implement Secure Session Cookies

Session cookies should be configured with proper attributes:

- **Secure flag**: Ensures cookies are sent only over HTTPS
- **HttpOnly flag**: Prevents access via JavaScript, reducing XSS risks
- **SameSite attribute**: Restricts cross-site cookie usage

These flags reduce the chances of cookie theft via scripts or cross-site attacks.

---

3. Use Strong and Random Session IDs

According to CEH, session IDs must be:

- Long enough
- Cryptographically random
- Non-predictable
- Unique per session

Weak session IDs can be guessed or brute-forced, leading directly to session hijacking.

👉 CEH exam favorite: predictable tokens = compromised authentication.

---

4. Regenerate Session IDs Frequently

Session IDs should be regenerated:

- After successful login
- After privilege escalation
- During logout

This prevents **session fixation attacks**, where attackers force victims to use known session IDs.

---

5. Implement Session Timeouts

CEH recommends both:

- **Idle timeout** – Ends session after inactivity
- **Absolute timeout** – Ends session after a fixed duration

Shorter session lifetimes reduce the impact of stolen session IDs.

---

6. Prevent Cross-Site Scripting (XSS)

Since XSS is a major cause of session cookie theft, CEH stresses:

- Input validation
- Output encoding
- Content Security Policy (CSP)

Blocking XSS blocks cookie theft at the source.

---

7. Use HTTP Strict Transport Security (HSTS)

HSTS forces browsers to communicate only via HTTPS, even if users type HTTP manually. This prevents **SSL stripping attacks**, commonly used during MITM session hijacking.

---

8. Bind Session to Client Attributes

Sessions can be bound to:

- IP address
- User-Agent string

- Device fingerprint

If these change unexpectedly, the session can be invalidated. CEH highlights this as an advanced defense layer.

---

9. Monitor and Log Session Activities

CEH recommends monitoring for:

- Sudden IP changes
- Concurrent logins
- Abnormal request patterns

Intrusion Detection Systems (IDS) and Web Application Firewalls (WAFs) can help identify hijacked sessions.

---

10. Proper Logout and Session Destruction

Logout must:

- Destroy the session on the server
- Invalidate session IDs immediately

Client-side logout alone is useless if the session remains active on the server.

---

11. User Awareness and Secure Practices

CEH also emphasizes human factors:

- Avoid public Wi-Fi for sensitive logins
- Always log out from shared systems
- Do not save cookies or sessions on public devices

Module Summary

This module provided a comprehensive understanding of **session hijacking attacks**, focusing on how attackers exploit weak session management mechanisms to gain unauthorized access to user accounts. It explored the fundamental concept of web sessions, the role of session IDs, and why session tokens are treated as temporary authentication credentials in modern web applications.

The module examined various **types of session hijacking**, including active and passive hijacking, sidejacking, XSS-based hijacking, session fixation, and Man-in-the-Middle attacks. Practical attack techniques such as packet sniffing, cookie theft, and token prediction were discussed to demonstrate real-world exploitation scenarios.

Additionally, the module introduced commonly used **tools** like Bettercap, Burp Suite, Wireshark, Ettercap, and Cookie Cadger, highlighting their role in identifying, intercepting, and analyzing session data during ethical hacking and penetration testing activities.

A significant focus was placed on **countermeasures**, emphasizing secure communication using HTTPS, strong and unpredictable session IDs, secure cookie attributes, session regeneration, timeout mechanisms, XSS prevention, and continuous session monitoring. The importance of user awareness and secure web practices was also reinforced.

Overall, this module bridged the gap between **theoretical concepts and practical implementation**, enabling learners to understand both the attacker's mindset and the defensive strategies required to protect web applications against session hijacking attacks.

THANK YOU