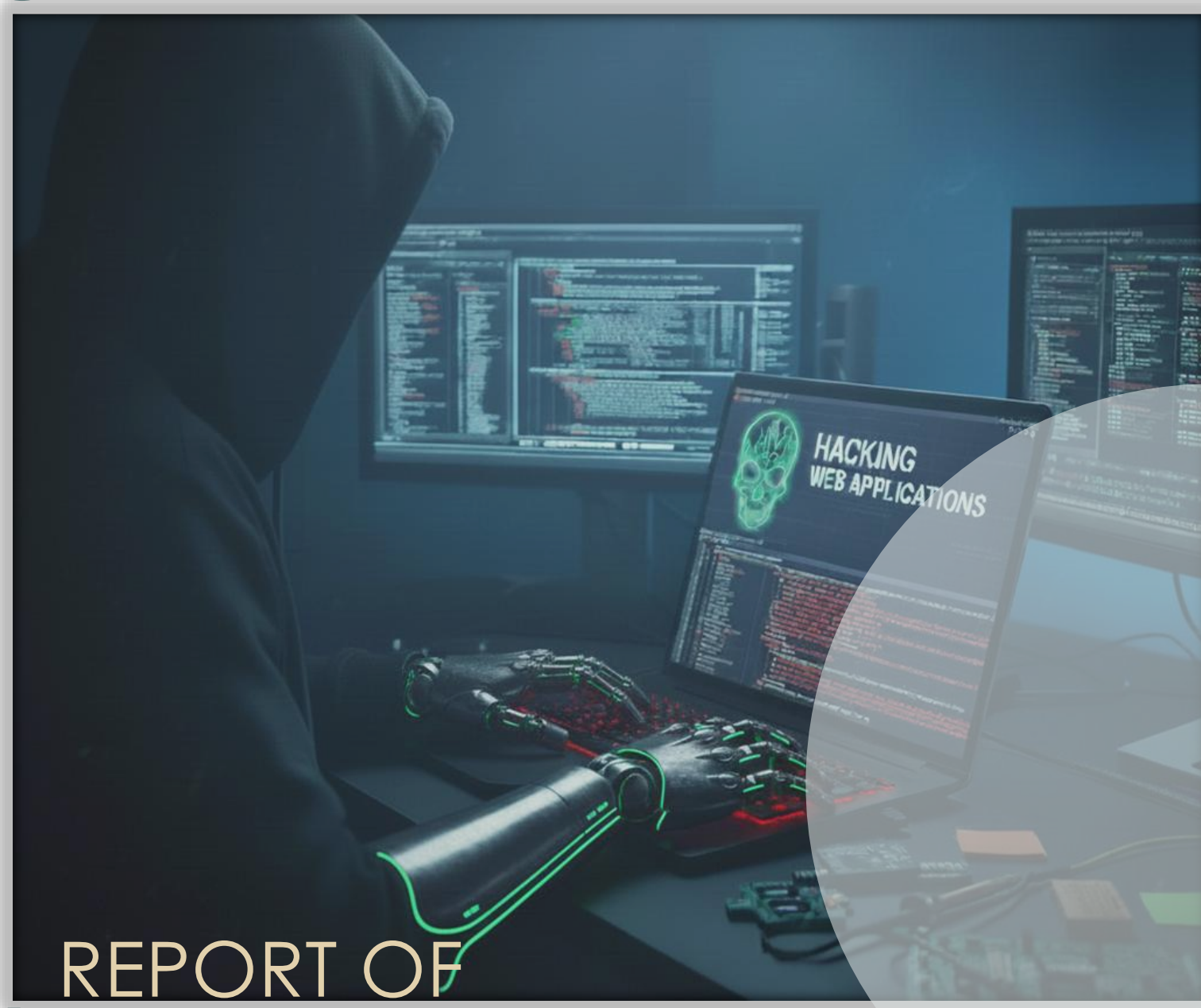


REPORT OF HACKING WEB APPLICATIONS

BY SACHCHITANAND YADAV

HACKING WEB APPLICATIONS

MODULE - 14

Learning Objectives -

- Explain Hacking Web Applications Concepts
- Footprint the Web Infrastructure
- Perform Web Application Attacks
- Detect Web Application Vulnerabilities using Various Web Application Security Tools
- Perform Web Application Hacking using AI
- Explain Hacking Web Applications Countermeasures

Table of Contents

5.1 Intruder
5.2 Sniper Attack
5.3 Battering Ram Attack
5.4 Pitchfork Attack
5.5 Cluster Bomb Attack
5.6 Repeater
5.7 Sequencer

---

6. Detecting Web Application Vulnerabilities

6.1 Role of Security Testing Tools
6.2 Vulnerability Scanning using Wapiti
6.3 Analysis of Scan Reports

---

7. Web Application Hacking using Artificial Intelligence

7.1 Introduction to AI-Based Web Application Hacking
7.2 Advantages of AI in Penetration Testing
7.3 Lab Objectives
7.4 Using Gemini-CLI for Web Application Security Testing

---

8. Countermeasures Against Web Application Hacking

8.1 Strong Input Validation
8.2 Proper Output Encoding
8.3 Secure Authentication and Session Management
8.4 Proper Access Control Enforcement
8.5 Protection Against SQL Injection
8.6 Secure File Upload Handling
8.7 HTTPS Enforcement
8.8 Security Configuration Hardening
8.9 Patch and Update Management
8.10 Security Logging and Monitoring
8.11 Web Application Firewalls (WAF)
8.12 Regular Security Testing

---

9. Web Application Hacking – Module Summary

Hacking Web Applications Concepts: -

Web Application Hacking

Web Application Hacking is the disciplined process of **finding, validating, and reporting security weaknesses** in web-based applications. It's not chaos—it's controlled curiosity. The purpose is simple and serious:

- Identify flaws attackers could abuse
- Understand real-world attack behavior
- Help developers fix issues before headlines do

How Web Applications Work

A web application follows the classic **client–server model**—unchanged, undefeated.

- **Client**: The web browser that sends requests
- **Server**: Processes requests and returns responses
- **Protocol**: Communication happens over **HTTP or HTTPS**

Example HTTP Request

```
GET /index.html HTTP/1.1
Host: example.com
```

The browser asks.
The server answers.
Security decides whether this exchange stays polite or turns hostile.

Web Application Hacking Process

This is the **playbook**—learn it once, use it forever.

1. Information Gathering (Footprinting)

Collect publicly available data about the target.

- Subdomains
- Server details
- Directory structure

Knowledge first. Noise later.

---

2. Scanning

Identify exposed services and technologies.

- Open ports
- Web directories
- Server stack (Apache, PHP, etc.)

You're mapping the battlefield.

---

3. Vulnerability Detection

Find security weaknesses manually or using tools.

- SQL Injection
- XSS
- File upload flaws

Detection is proof that assumptions failed.

---

4. Exploitation

Confirm vulnerabilities by controlled attacks.

- Dumping database data
- Bypassing authentication

This is validation, not vandalism.

---

5. Post-Exploitation

Assess impact after compromise.

- Sensitive data access
- Web shell uploads

6. Reporting

Document everything.

- Vulnerability details
- Proof of concept
- Remediation steps

How Vulnerabilities Work

SQL Injection

Input:

```
' OR '1'='1
```

If input isn't validated, authentication collapses like a bad alibi.

Cross-Site Scripting (XSS)

Input:

```
<script>alert('XSS')</script>
```

Unsanitized input runs attacker code in another user's browser.
Trust broken. Session stolen. Game over.

Command Injection

Input:

```
test; ls -la
```

If passed directly to the OS shell, attackers execute system commands.
That's not a bug—that's negligence.

File Upload Vulnerability

Malicious File:

```
shell.php.jpg
```

Weak validation lets attackers upload executable files disguised as images.

HTTP Methods

- **GET** – Request data (visible in URL)
- **POST** – Send data (hidden from URL)
- **PUT** – Update a resource
- **DELETE** – Remove a resource
- **OPTIONS** – Discover allowed methods

OWASP Top 10

- **A01 – Broken Access Control**
  Users access others' data by changing URLs.
- **A02 – Cryptographic Failures**
  Sensitive data sent without encryption.
- **A03 – Injection**
  Malicious input alters backend commands.
- **A04 – Insecure Design**
  Security ignored during architecture planning.
- **A05 – Security Misconfiguration**
  Default credentials, verbose errors, weak settings.
- **A06 – Vulnerable Components**
  Outdated libraries with known exploits.
- **A07 – Identification & Authentication Failures**
  Weak passwords, no brute-force protection.
- **A08 – Software & Data Integrity Failures**
  Untrusted updates or tampered files accepted.
- **A09 – Logging & Monitoring Failures**
  Attacks occur silently, no alerts triggered.
- **A10 – SSRF**
  Server tricked into accessing internal resources.

Key Web Hacking Concepts

- **Cookies & Sessions** – Track logged-in users
- **CSRF Tokens** – Prevent forged requests
- **Input Validation** – Accept only safe data
- **Encoding** – Neutralize malicious input
- **Authentication Mechanisms** – Passwords, tokens, MFA

Footprint the Web Infrastructure

Web infrastructure footprinting is the process of gathering complete information about the target web application, its related components, and how they work.

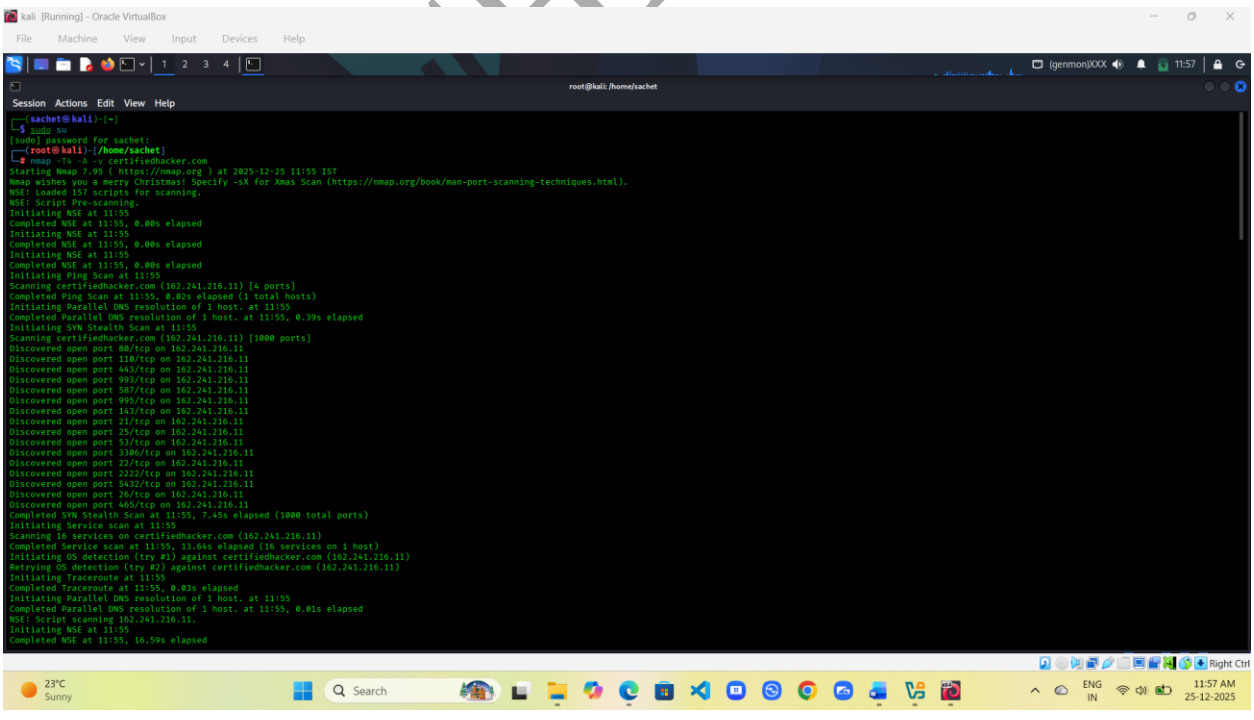
Lab Scenario

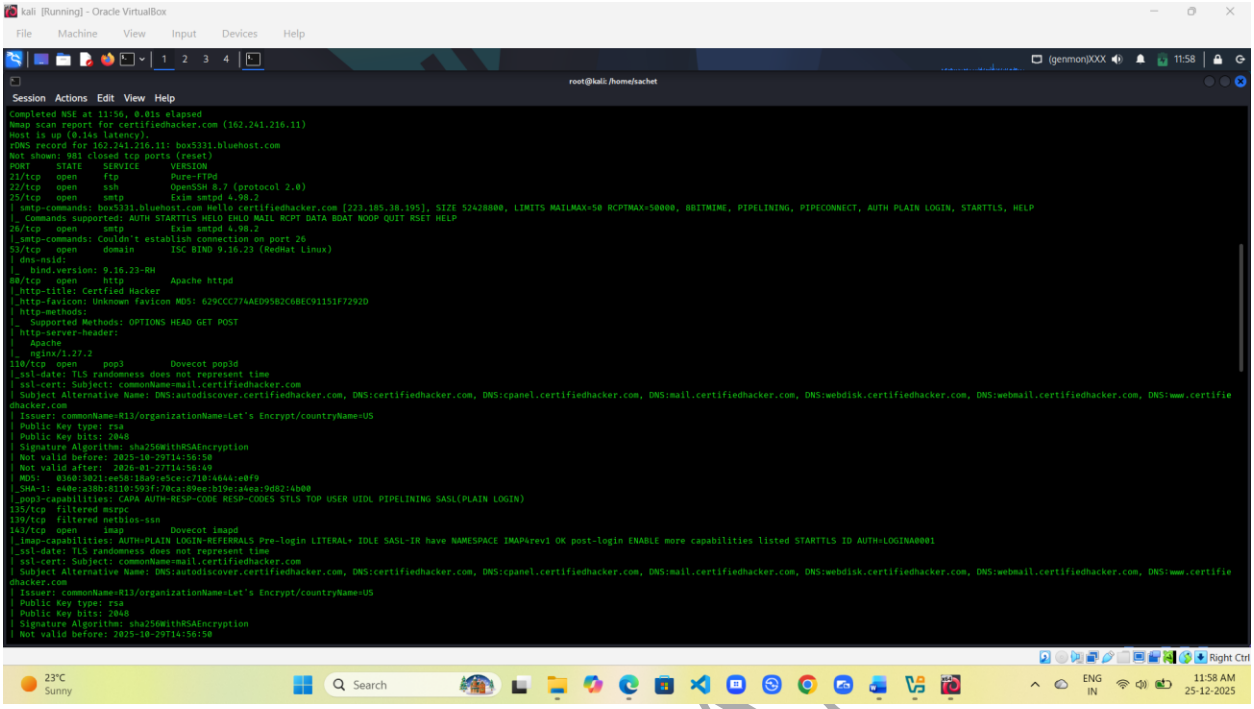
The first step in web application hacking for an ethical hacker or pen tester is to gather the maximum available information about the target organization website by performing web application footprinting using various techniques and tools. In this step, you will use techniques such as web spidering and vulnerability scanning to gather complete information about the target web application.

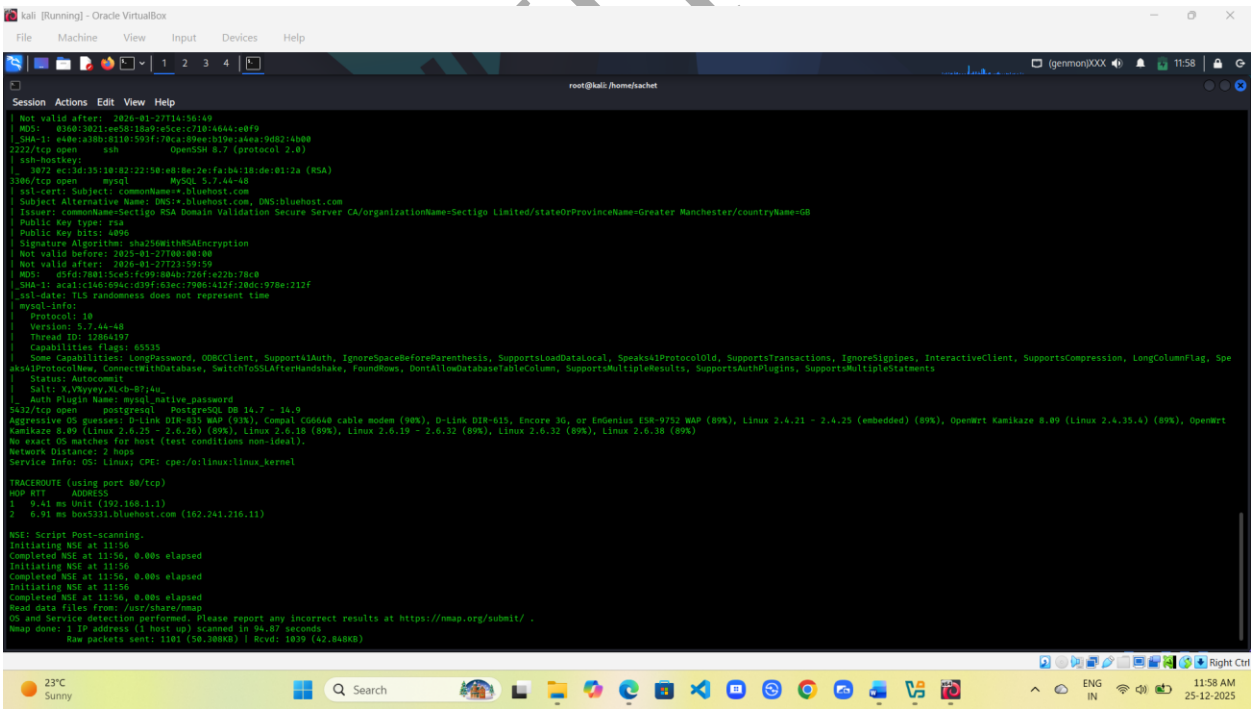
Lab Tasks

Task 1: Perform Web Application Reconnaissance using Nmap and Telnet

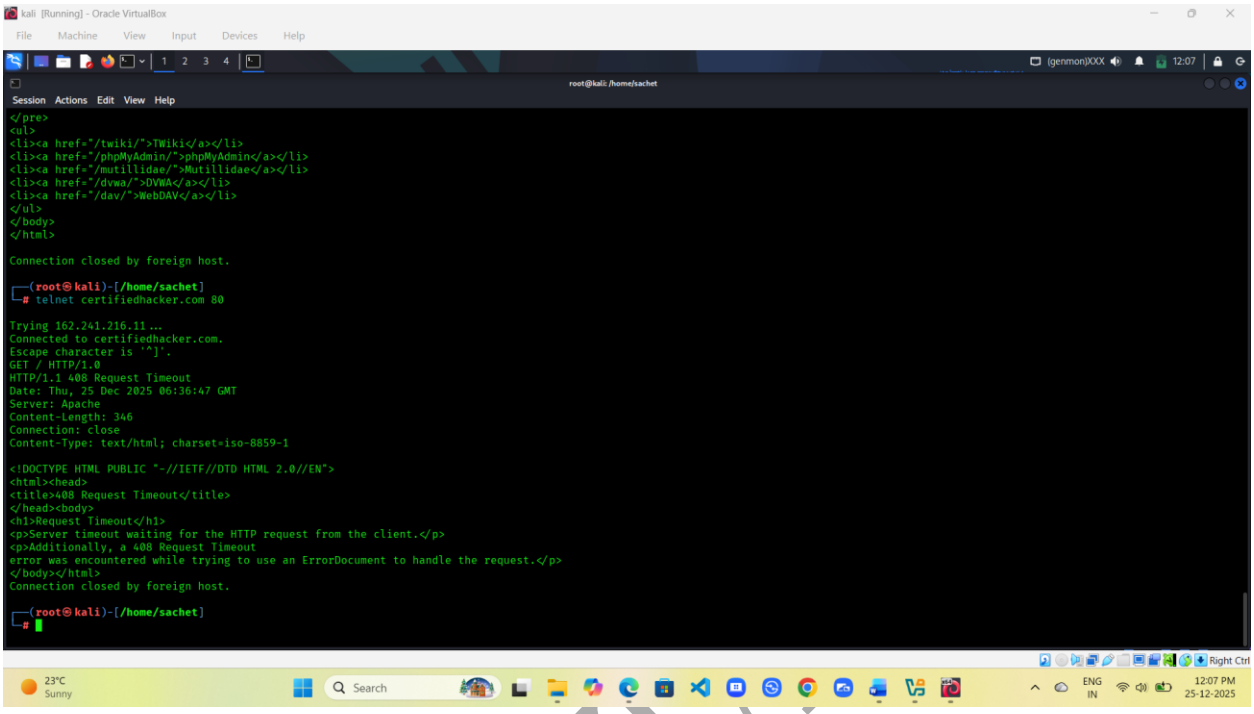
nmap -T4 -A -v certifiedhacker.com




kali [Running] - Oracle VirtualBox
File Machine View Input Devices Help
root@kali: /home/sachet
Session Actions Edit View Help
Completed NSE at 11:56, 0.01s elapsed
Nmap scan report for certifiedhacker.com (162.241.216.11)
Host is up (0.14s latency).
rDNS record for 162.241.216.11: box5331.bluehost.com
Not shown: 981 closed tcp ports (reset)
PORT STATE SERVICE VERSION
21/tcp open ftp Pure-FTPd
22/tcp open ssh OpenSSH 8.7 (protocol 2.0)
25/tcp open smtp Exim smtpd 4.98.2
| smtp-commands: box5331.bluehost.com Hello certifiedhacker.com [223.185.38.195], SIZE 52428800, LIMITS MAILMAX=50 RCPTMAX=50000, 8BITMIME, PIPELINING, PIPECONNECT, AUTH PLAIN LOGIN, STARTTLS, HELP
|_ Commands supported: AUTH STARTTLS HELO EHLO MAIL RCPT DATA BDAT NOOP QUIT RSET HELP
26/tcp open smtp Exim smtpd 4.98.2
|_smtp-commands: Couldn't establish connection on port 26
53/tcp open domain ISC BIND 9.16.23 (RedHat Linux)
| dns-nsid:
|_ bind.version: 9.16.23-RH
80/tcp open http Apache httpd
|_http-title: Certfied Hacker
|_http-favicon: Unknown favicon MD5: 629CCC774AED95B2C6BEC91151F7292D
| http-methods:
|_ Supported Methods: OPTIONS HEAD GET POST
| http-server-header:
| Apache
|_ nginx/1.27.2
110/tcp open pop3 Dovecot pop3d
|_ssl-date: TLS randomness does not represent time
| ssl-cert: Subject: commonName=mail.certifiedhacker.com
| Subject Alternative Name: DNS:autodiscover.certifiedhacker.com, DNS:certifiedhacker.com, DNS:cpanel.certifiedhacker.com, DNS:mail.certifiedhacker.com, DNS:webdisk.certifiedhacker.com, DNS:webmail.certifiedhacker.com, DNS:www.certifiedhacker.com
| Issuer: commonName=R13/organizationName=Let's Encrypt/countryName=US
| Public Key type: rsa
| Public Key bits: 2048
| Signature Algorithm: sha256WithRSAEncryption
| Not valid before: 2025-10-29T14:56:50
| Not valid after: 2026-01-27T14:56:49
| MD5: 0360:3821:ee58:18a9:e5ce:c710:4644:e0f9
|_SHA-1: e40e:a38b:8110:593f:70ca:89ee:b19e:a4ea:9d82:4b00
|_pop3-capabilities: CAPA AUTH-RESP-CODE RESP-CODES STLS TOP USER UIDL PIPELINING SASL(PLAIN LOGIN)
135/tcp filtered msrpc
139/tcp filtered netbios-ssn
143/tcp open imap Dovecot imapd
|_imap-capabilities: AUTH=PLAIN LOGIN-REFERRALS Pre-login LITERAL+ IDLE SASL-IR have NAMESPACE IMAP4rev1 OK post-login ENABLE more capabilities listed STARTTLS ID AUTH=LOGINA0001
|_ssl-date: TLS randomness does not represent time
| ssl-cert: Subject: commonName=mail.certifiedhacker.com
| Subject Alternative Name: DNS:autodiscover.certifiedhacker.com, DNS:certifiedhacker.com, DNS:cpanel.certifiedhacker.com, DNS:mail.certifiedhacker.com, DNS:webdisk.certifiedhacker.com, DNS:webmail.certifiedhacker.com, DNS:www.certifiedhacker.com
| Issuer: commonName=R13/organizationName=Let's Encrypt/countryName=US
| Public Key type: rsa
| Public Key bits: 2048
| Signature Algorithm: sha256WithRSAEncryption
| Not valid before: 2025-10-29T14:56:50
Right Ctrl
23°C Sunny
Search
ENG IN
11:58 AM 25-12-2025

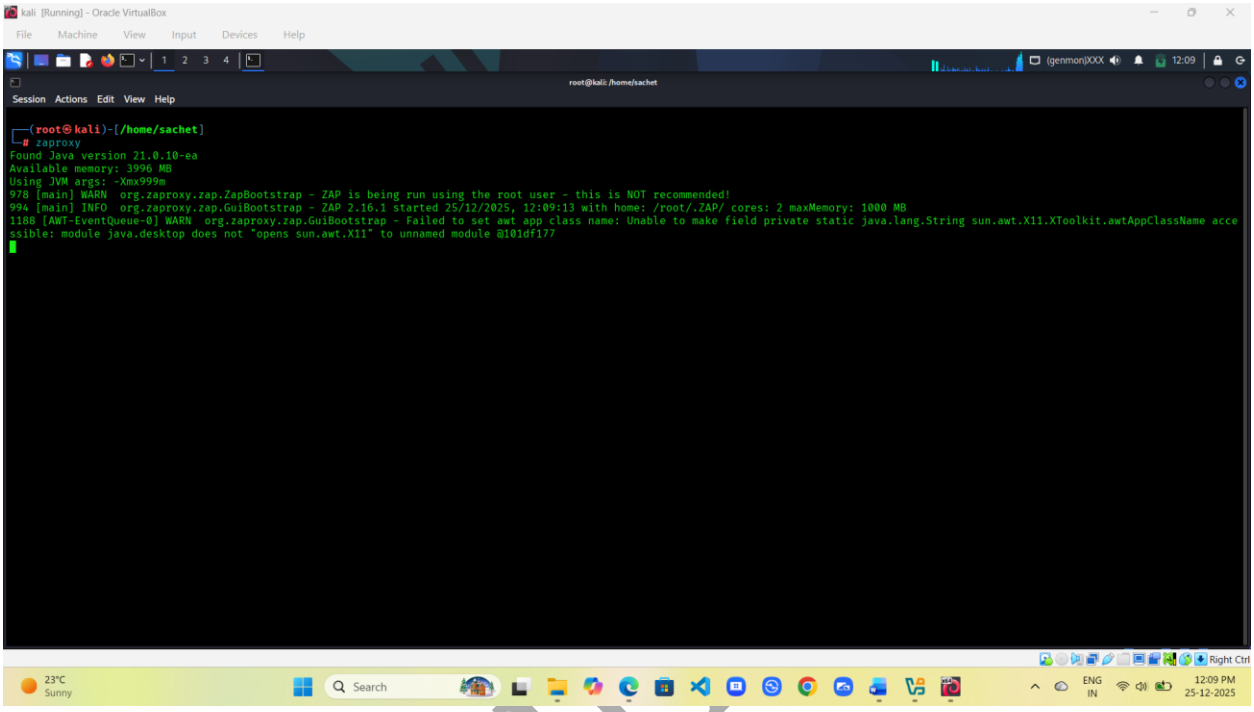

kali [Running] - Oracle VirtualBox
File Machine View Input Devices Help
root@kali: /home/sachet
Session Actions Edit View Help
| Not valid after: 2026-01-27T14:56:49
| MD5: 0360:3821:ee58:18a9:e5ce:c710:4644:e0f9
|_SHA-1: e40e:a38b:8110:593f:70ca:89ee:b19e:a4ea:9d82:4b00
2222/tcp open ssh OpenSSH 8.7 (protocol 2.0)
| ssh-hostkey:
|_ 3072 ec:3d:35:10:82:22:50:e8:8e:2e:fa:b4:18:de:01:2a (RSA)
3306/tcp open mysql MySQL 5.7.44-48
| ssl-cert: Subject: commonName=*.bluehost.com
| Subject Alternative Name: DNS:*.bluehost.com, DNS:bluehost.com
| Issuer: commonName=Sectigo RSA Domain Validation Secure Server CA/organizationName=Sectigo Limited/stateOrProvinceName=Greater Manchester/countryName=GB
| Public Key type: rsa
| Public Key bits: 4096
| Signature Algorithm: sha256WithRSAEncryption
| Not valid before: 2025-01-27T00:00:00
| Not valid after: 2026-01-27T23:59:59
| MD5: d5fd:7801:5ce5:fc99:804b:726f:e22b:78c0
|_SHA-1: aca1:c146:694c:d39f:63ec:7906:412f:20dc:978e:212f
|_ssl-date: TLS randomness does not represent time
| mysql-info:
| Protocol: 10
| Version: 5.7.44-48
| Thread ID: 12864197
| Capabilities flags: 65535
| Some Capabilities: LongPassword, ODBCClient, Support41Auth, IgnoreSpaceBeforeParenthesis, SupportsLoadDataLocal, Speaks41ProtocolOld, SupportsTransactions, IgnoreSigpipes, InteractiveClient, SupportsCompression, LongColumnFlag, Speaks41ProtocolNew, ConnectWithDatabase, SwitchToSSLAfterHandshake, FoundRows, DontAllowDatabaseTableColumn, SupportsMultipleResults, SupportsAuthPlugins, SupportsMultipleStatments
| Status: Autocommit
| Salt: X,V%yxy,XL<b~B7;4u_
|_ Auth Plugin Name: mysql_native_password
5432/tcp open postgresql PostgreSQL DB 14.7 - 14.9
Aggressive OS guesses: D-Link DIR-835 WAP (93%), Compal CG6640 cable modem (90%), D-Link DIR-615, Encore 3G, or EnGenius ESR-9752 WAP (89%), Linux 2.4.21 - 2.4.25 (embedded) (89%), OpenWrt Kamikaze 8.09 (Linux 2.4.35.4) (89%), OpenWrt Kamikaze 8.09 (Linux 2.6.25 - 2.6.29) (89%), Linux 2.6.18 (89%), Linux 2.6.19 - 2.6.32 (89%), Linux 2.6.32 (89%), Linux 2.6.38 (89%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 2 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
TRACEROUTE (using port 80/tcp)
HOP RTT ADDRESS
1 9.41 ms Unit (192.168.1.1)
2 6.91 ms box5331.bluehost.com (162.241.216.11)
NSE: Script Post-scanning.
Initiating NSE at 11:56
Completed NSE at 11:56, 0.00s elapsed
Initiating NSE at 11:56
Completed NSE at 11:56, 0.00s elapsed
Initiating NSE at 11:56
Completed NSE at 11:56, 0.00s elapsed
Read data files from: /usr/share/nmap
OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 94.47 seconds
Raw packets sent: 1101 (50.308KB) | Rcvd: 1039 (42.846KB)
Right Ctrl
23°C Sunny
Search
ENG IN
11:58 AM 25-12-2025

telnet certifiedhacker.com 80

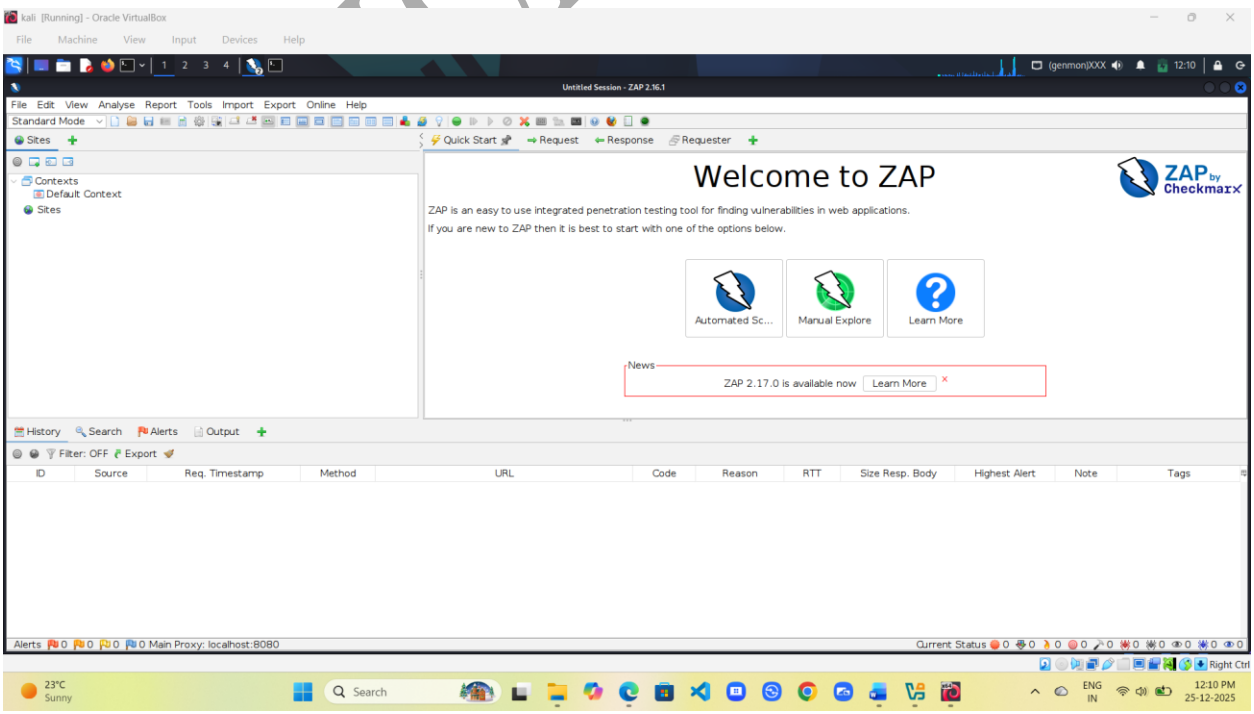


Task 2: Perform Web Spidering using OWASP ZAP

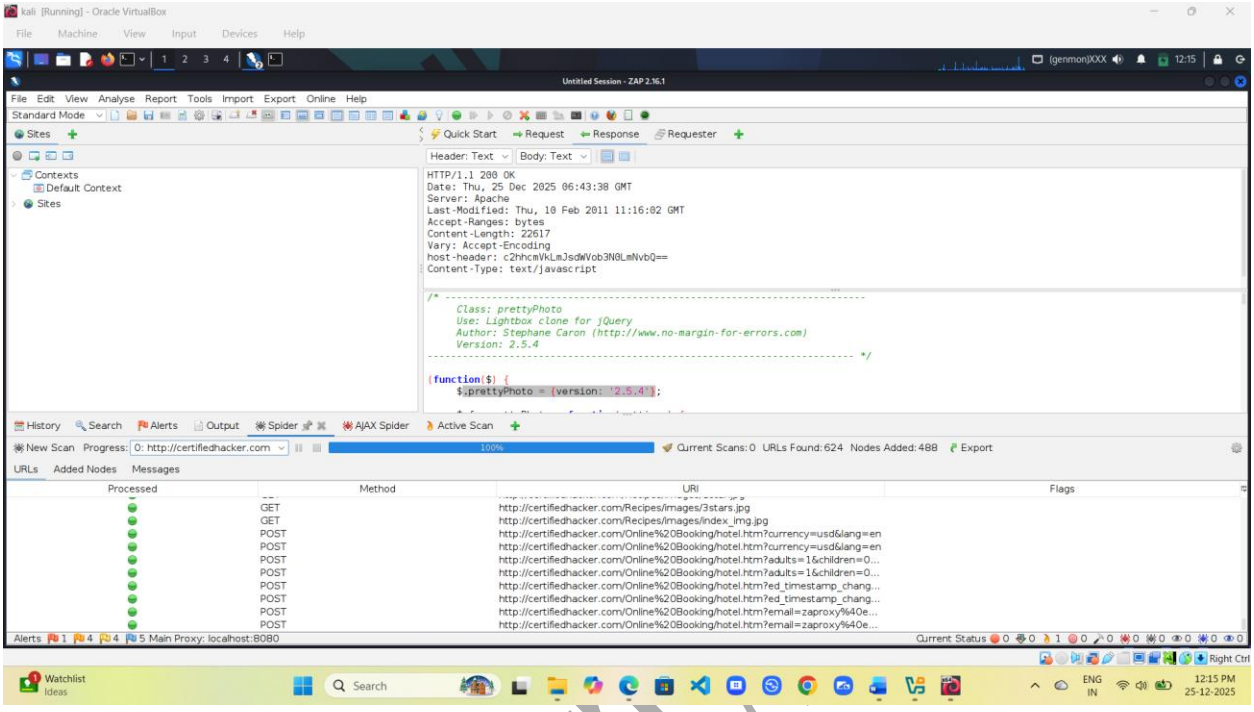
Type - zaproxy



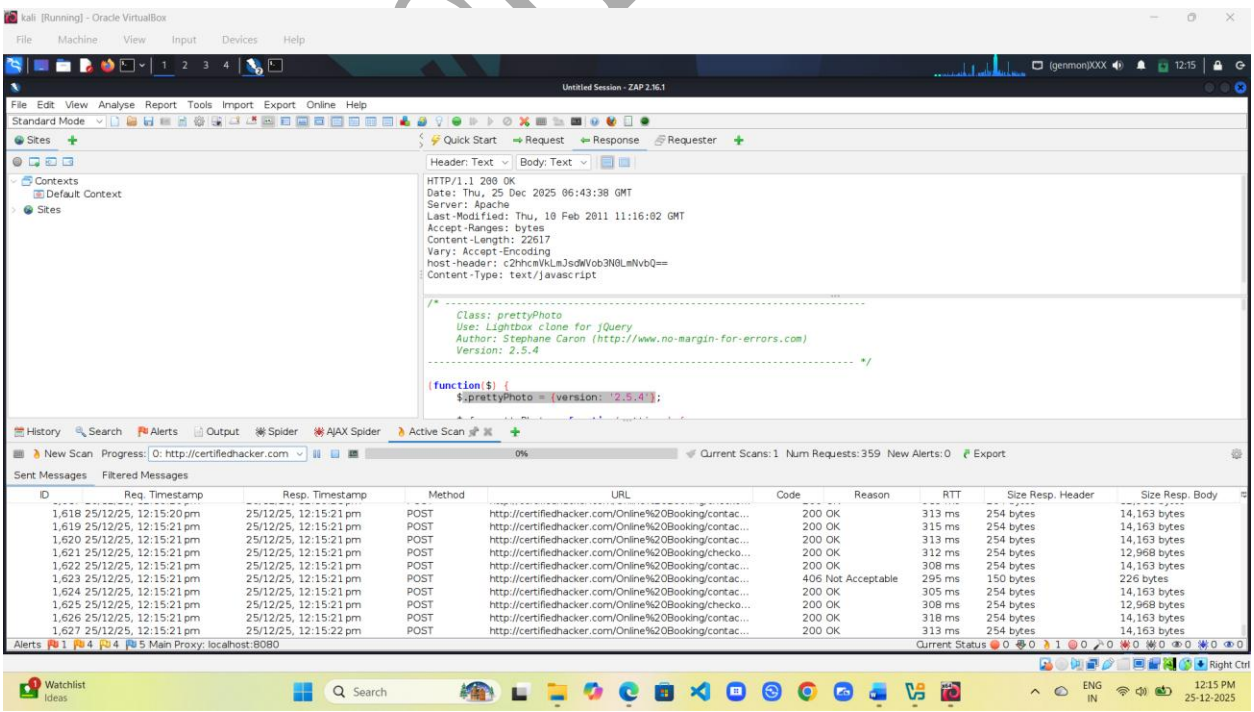
The OWASP ZAP main window appears. Under the Quick Start tab, click the Automated Scan option under Welcome to OWASP ZAP.



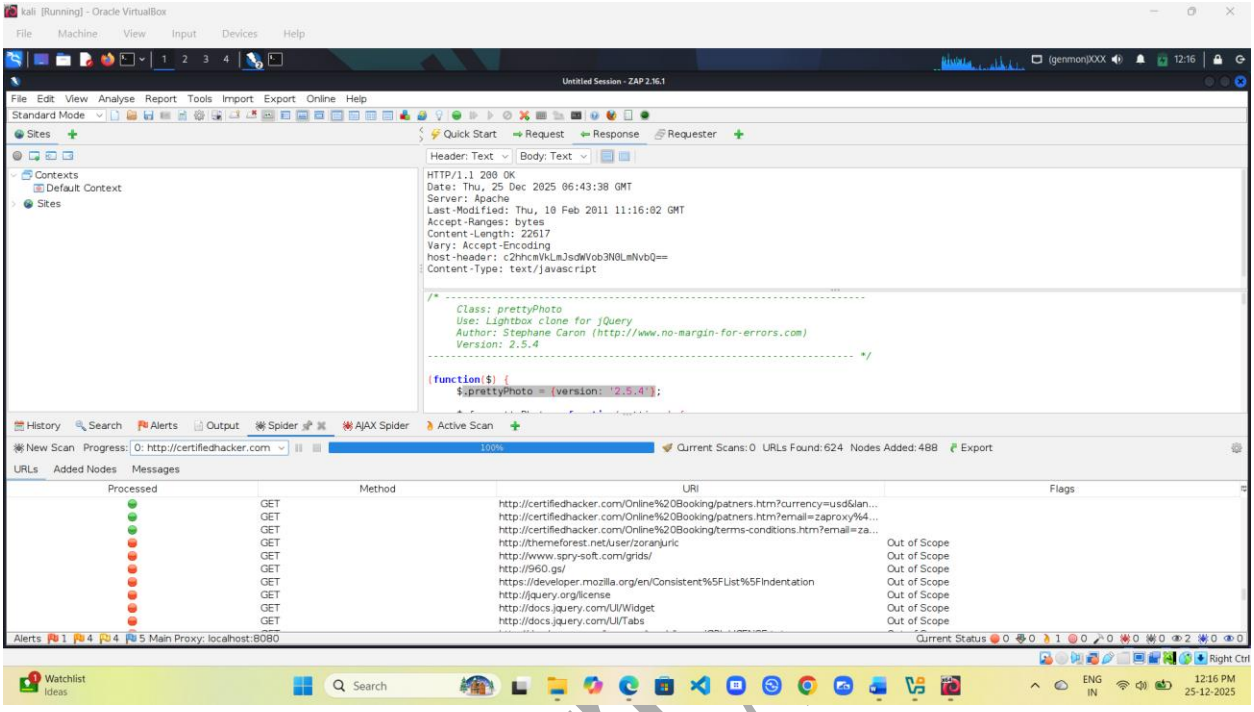
Now Start Scanning



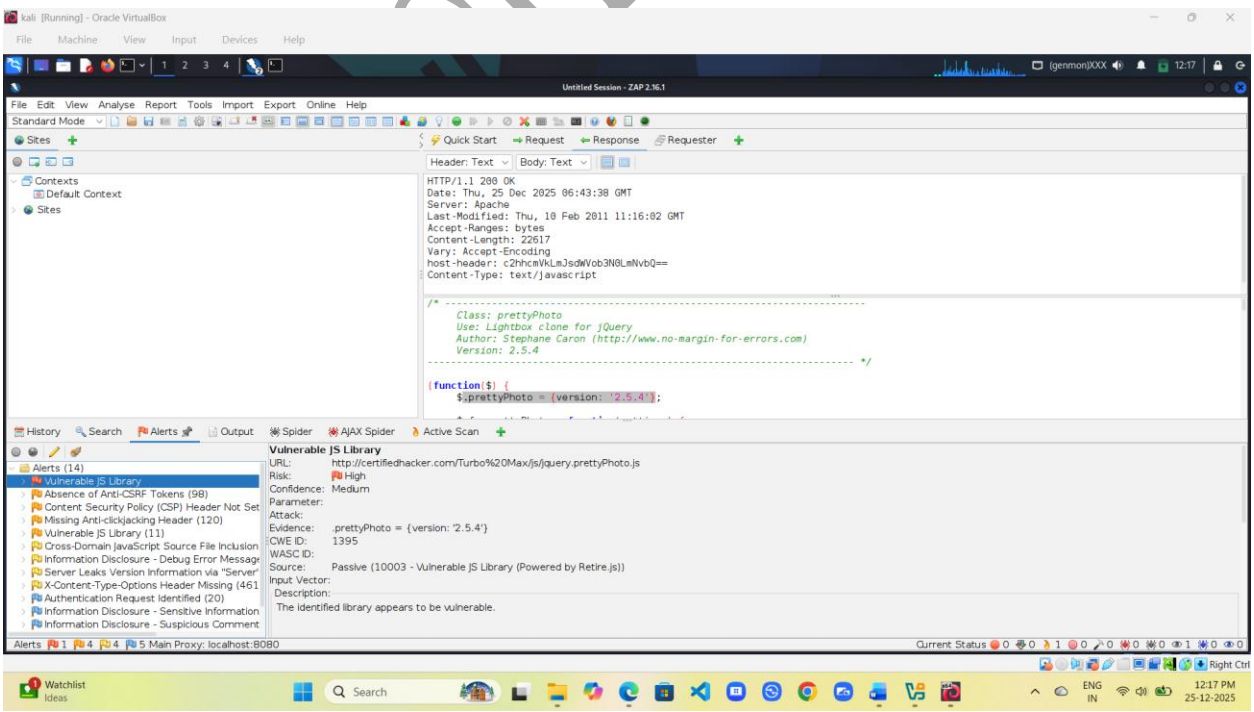
Active Scan



Spider

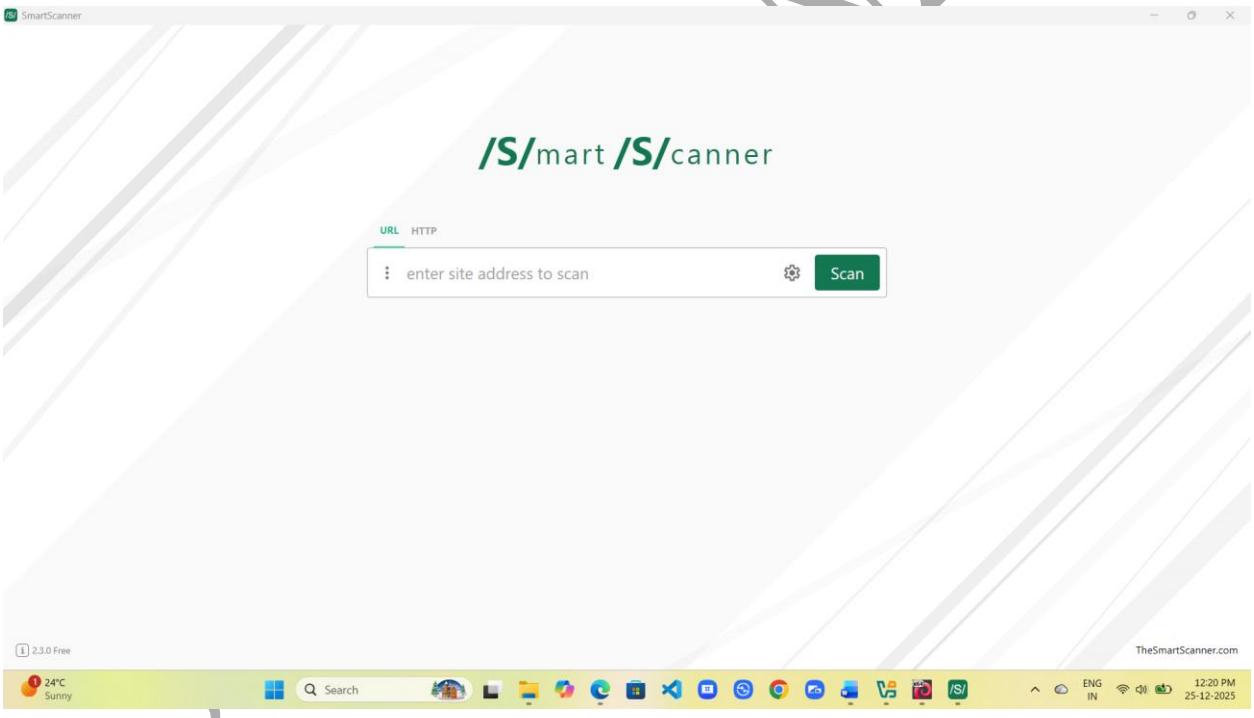


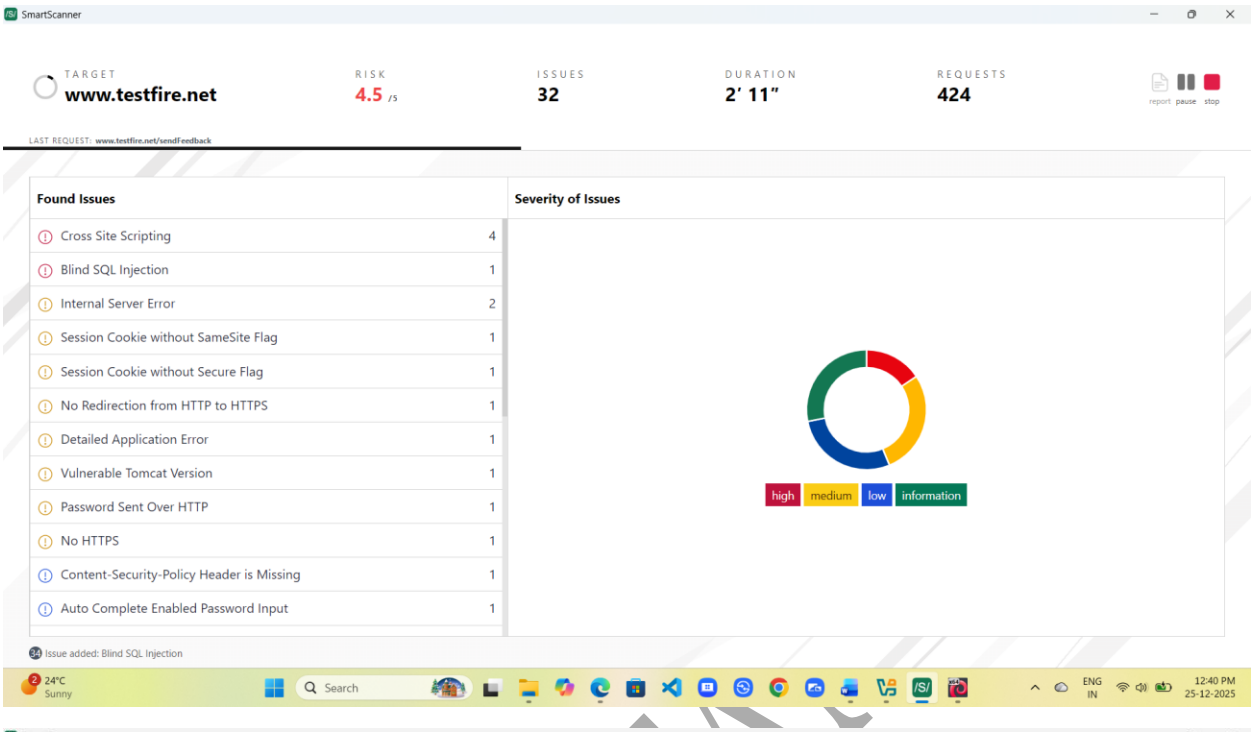
Alerts

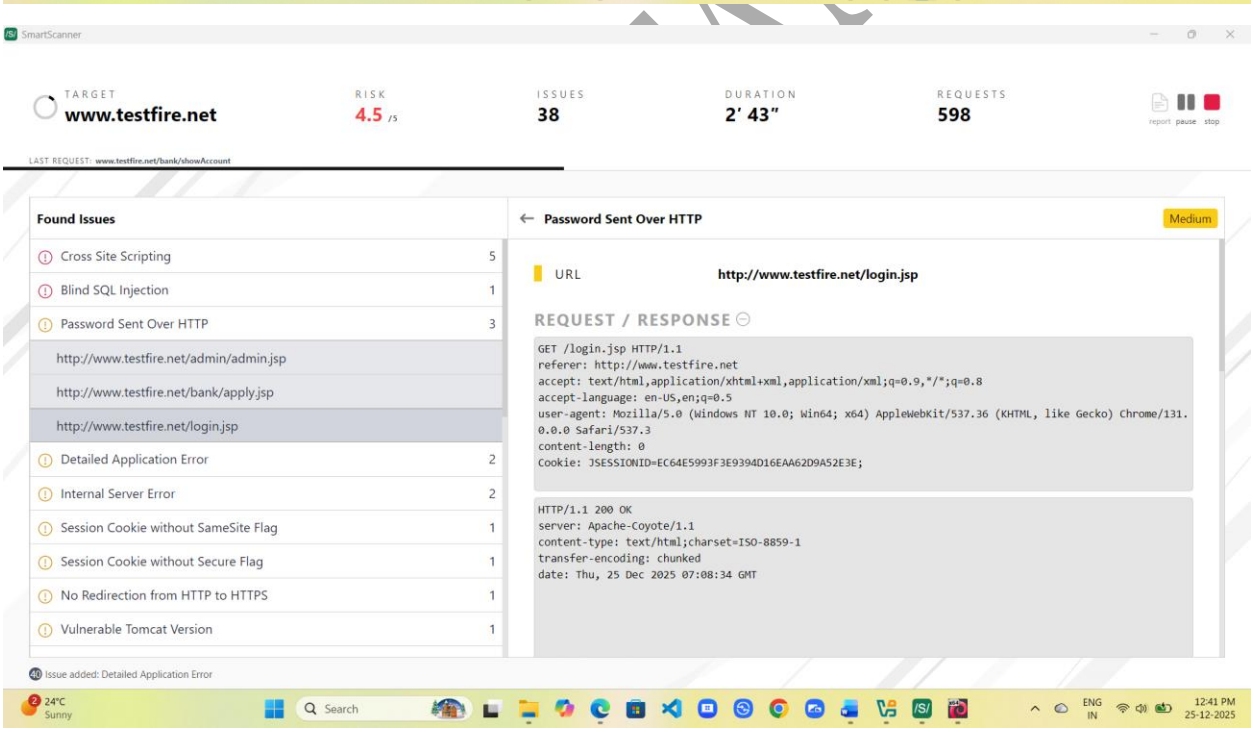


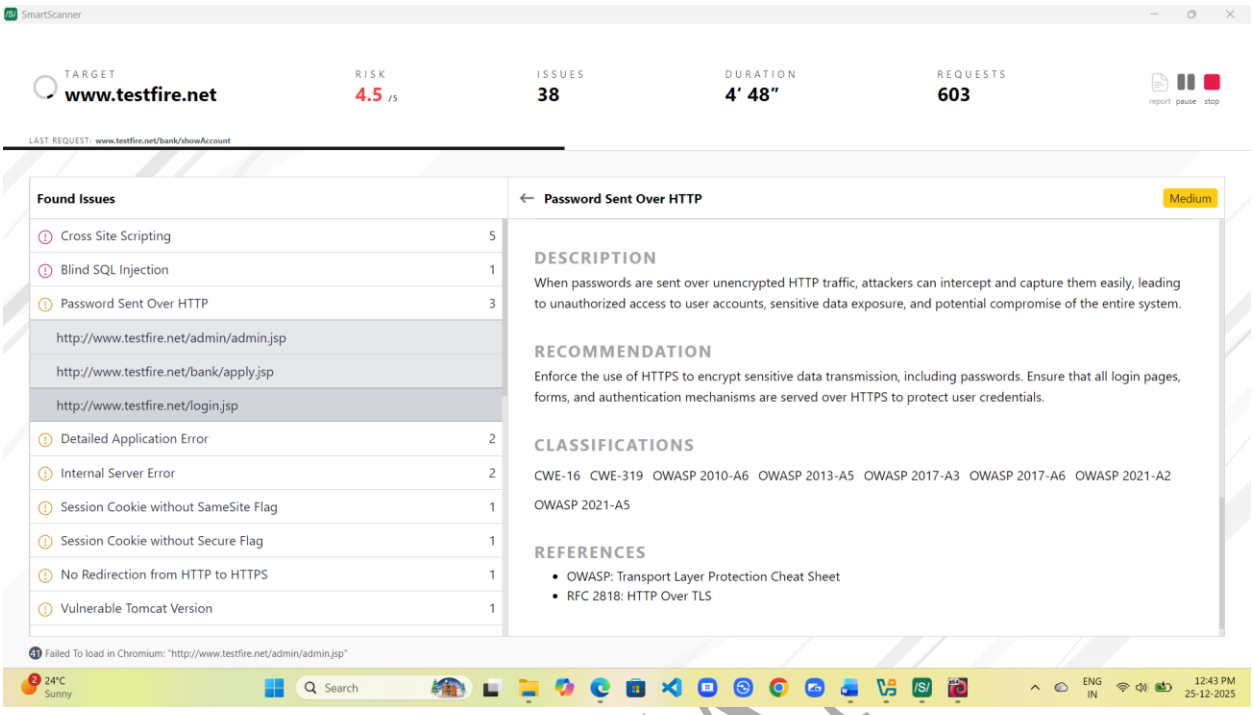
Task 3: Perform Web Application Vulnerability Scanning using Smart Scanner

SmartScanner leverages machine learning (ML) and artificial intelligence (AI) techniques to adapt its methodologies to the behavior of the target. This integration allows SmartScanner to minimize false positives. It uses Al for identifying vulnerable pages, detecting 404 custom pages, identifying input vectors, fingerprinting the target and calculating the security risk.

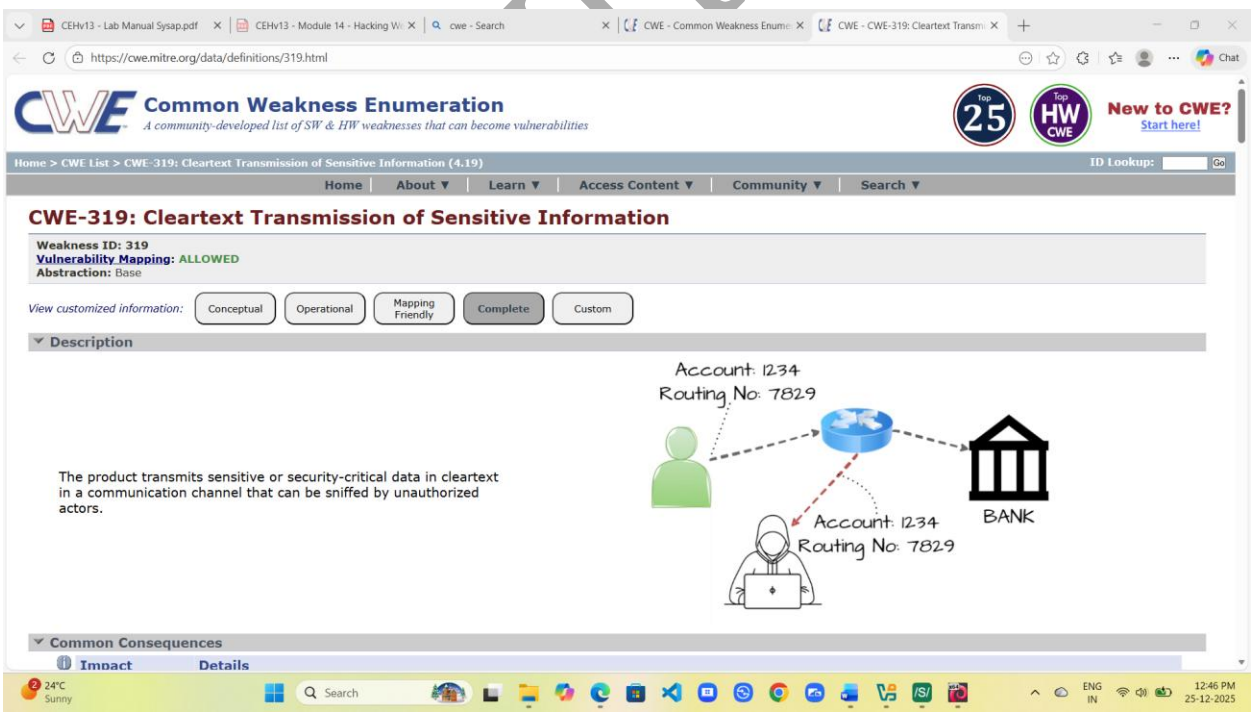


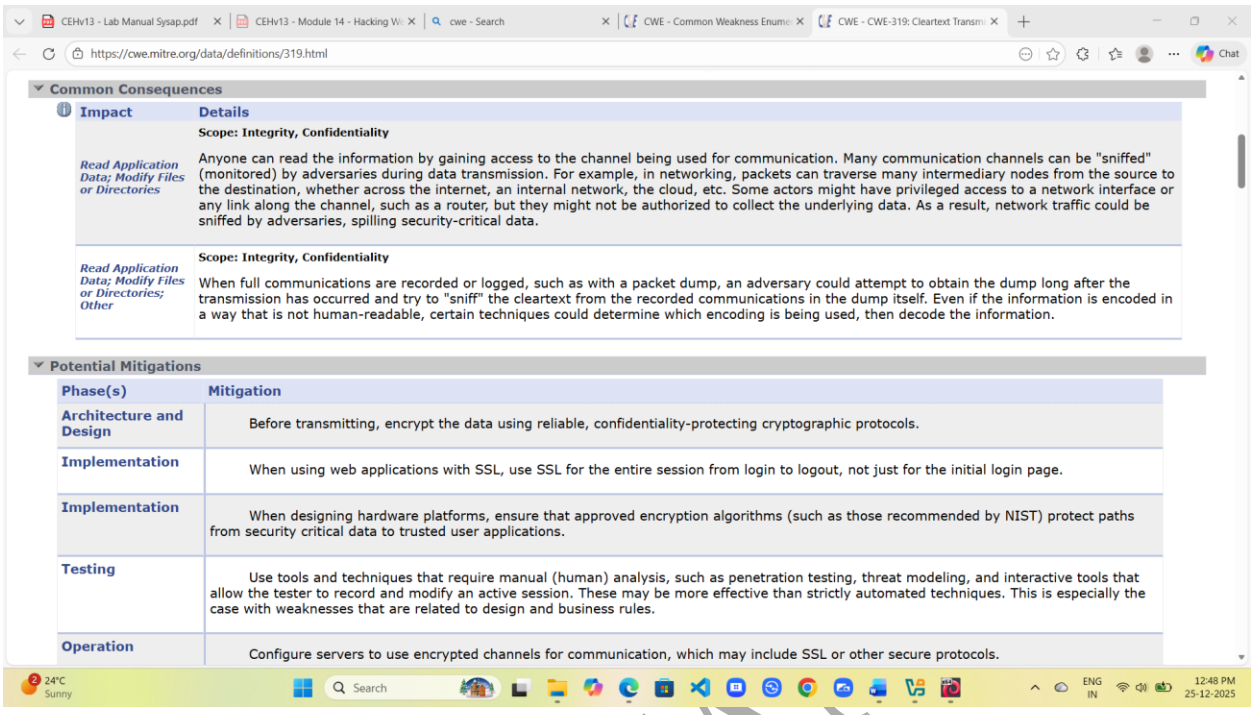

SmartScanner
TARGET
www.testfire.net
RISK
4.5 /5
ISSUES
32
DURATION
2' 11"
REQUESTS
424
report
pause
stop
LAST REQUEST: www.testfire.net/sendFeedback
Found Issues
Severity of Issues
Cross Site Scripting 4
Blind SQL Injection 1
Internal Server Error 2
Session Cookie without SameSite Flag 1
Session Cookie without Secure Flag 1
No Redirection from HTTP to HTTPS 1
Detailed Application Error 1
Vulnerable Tomcat Version 1
Password Sent Over HTTP 1
No HTTPS 1
Content-Security-Policy Header is Missing 1
Auto Complete Enabled Password Input 1
high
medium
low
information
Issue added: Blind SQL Injection
24°C
Sunny
Search
ENG
IN
12:40 PM
25-12-2025


SmartScanner
TARGET
www.testfire.net
RISK
4.5 /5
ISSUES
38
DURATION
2' 43"
REQUESTS
598
report
pause
stop
LAST REQUEST: www.testfire.net/bank/showAccount
Found Issues
Cross Site Scripting 5
Blind SQL Injection 1
Password Sent Over HTTP 3
http://www.testfire.net/admin/admin.jsp
http://www.testfire.net/bank/apply.jsp
http://www.testfire.net/login.jsp
Detailed Application Error 2
Internal Server Error 2
Session Cookie without SameSite Flag 1
Session Cookie without Secure Flag 1
No Redirection from HTTP to HTTPS 1
Vulnerable Tomcat Version 1
Password Sent Over HTTP
Medium
URL
http://www.testfire.net/login.jsp
REQUEST / RESPONSE
GET /login.jsp HTTP/1.1
referer: http://www.testfire.net
accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
accept-language: en-US,en;q=0.5
user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari/537.3
content-length: 0
Cookie: JSESSIONID=EC64E5993F3E9394D16EAA62D9A52E3E;
HTTP/1.1 200 OK
server: Apache-Coyote/1.1
content-type: text/html;charset=ISO-8859-1
transfer-encoding: chunked
date: Thu, 25 Dec 2025 07:08:34 GMT
Issue added: Detailed Application Error
24°C
Sunny
Search
ENG
IN
12:41 PM
25-12-2025


SmartScanner
TARGET
www.testfire.net
RISK
4.5 /5
ISSUES
38
DURATION
4' 48"
REQUESTS
603
report pause stop
LAST REQUEST: www.testfire.net/bank/showAccount
Found Issues
Cross Site Scripting 5
Blind SQL Injection 1
Password Sent Over HTTP 3
http://www.testfire.net/admin/admin.jsp
http://www.testfire.net/bank/apply.jsp
http://www.testfire.net/login.jsp
Detailed Application Error 2
Internal Server Error 2
Session Cookie without SameSite Flag 1
Session Cookie without Secure Flag 1
No Redirection from HTTP to HTTPS 1
Vulnerable Tomcat Version 1
Password Sent Over HTTP
Medium
DESCRIPTION
When passwords are sent over unencrypted HTTP traffic, attackers can intercept and capture them easily, leading to unauthorized access to user accounts, sensitive data exposure, and potential compromise of the entire system.
RECOMMENDATION
Enforce the use of HTTPS to encrypt sensitive data transmission, including passwords. Ensure that all login pages, forms, and authentication mechanisms are served over HTTPS to protect user credentials.
CLASSIFICATIONS
CWE-16 CWE-319 OWASP 2010-A6 OWASP 2013-A5 OWASP 2017-A3 OWASP 2017-A6 OWASP 2021-A2
OWASP 2021-A5
REFERENCES
OWASP: Transport Layer Protection Cheat Sheet
RFC 2818: HTTP Over TLS
Failed To load in Chromium: "http://www.testfire.net/admin/admin.jsp"
24°C Sunny
Search
ENG IN
12:43 PM 25-12-2025

CWE


https://cwe.mitre.org/data/definitions/319.html
Common Weakness Enumeration
A community-developed list of SW & HW weaknesses that can become vulnerabilities
Top 25
Top HW CWE
New to CWE?
Start here!
Home > CWE List > CWE-319: Cleartext Transmission of Sensitive Information (4.19)
ID Lookup:
Home About Learn Access Content Community Search
CWE-319: Cleartext Transmission of Sensitive Information
Weakness ID: 319
Vulnerability Mapping: ALLOWED
Abstraction: Base
View customized information:
Conceptual
Operational
Mapping Friendly
Complete
Custom
Description
The product transmits sensitive or security-critical data in cleartext in a communication channel that can be sniffed by unauthorized actors.
Account: 1234
Routing No: 7829
BANK
Account: 1234
Routing No: 7829
Common Consequences
Impact
Details
24°C Sunny
Search
ENG IN
12:46 PM 25-12-2025


CEHv13 - Lab Manual Sysap.pdf
CEHv13 - Module 14 - Hacking W
cwe - Search
CWE - Common Weakness Enume
CWE - CWE-319: Cleartext Transmi
https://cwe.mitre.org/data/definitions/319.html
Chat
Common Consequences
Impact
Details
Scope: Integrity, Confidentiality
Read Application Data; Modify Files or Directories
Anyone can read the information by gaining access to the channel being used for communication. Many communication channels can be "sniffed" (monitored) by adversaries during data transmission. For example, in networking, packets can traverse many intermediary nodes from the source to the destination, whether across the internet, an internal network, the cloud, etc. Some actors might have privileged access to a network interface or any link along the channel, such as a router, but they might not be authorized to collect the underlying data. As a result, network traffic could be sniffed by adversaries, spilling security-critical data.
Scope: Integrity, Confidentiality
Read Application Data; Modify Files or Directories; Other
When full communications are recorded or logged, such as with a packet dump, an adversary could attempt to obtain the dump long after the transmission has occurred and try to "sniff" the cleartext from the recorded communications in the dump itself. Even if the information is encoded in a way that is not human-readable, certain techniques could determine which encoding is being used, then decode the information.
Potential Mitigations
Phase(s)
Mitigation
Architecture and Design
Before transmitting, encrypt the data using reliable, confidentiality-protecting cryptographic protocols.
Implementation
When using web applications with SSL, use SSL for the entire session from login to logout, not just for the initial login page.
Implementation
When designing hardware platforms, ensure that approved encryption algorithms (such as those recommended by NIST) protect paths from security critical data to trusted user applications.
Testing
Use tools and techniques that require manual (human) analysis, such as penetration testing, threat modeling, and interactive tools that allow the tester to record and modify an active session. These may be more effective than strictly automated techniques. This is especially the case with weaknesses that are related to design and business rules.
Operation
Configure servers to use encrypted channels for communication, which may include SSL or other secure protocols.
24°C
Sunny
Search
ENG
IN
12:48 PM
25-12-2025

Perform Web Application Attacks

An expert ethical hacker or pen tester must implement various techniques to launch web application attacks on the target organization's website.

Lab Scenario

For an ethical hacker or pen tester, the next step after gathering required information about the target web application is to attack the web application. They must have the required knowledge to perform web application attacks to test the target network's web application security infrastructure.

Task 1: Perform a Brute-force Attack using Burp Suite

Burp Suite

Burp Suite is a **widely used web application security testing platform** employed by cybersecurity professionals, ethical hackers, and bug bounty hunters to **identify, analyze, and exploit vulnerabilities** in web applications. It acts as a bridge between the browser and the web server, allowing complete visibility and control over HTTP/HTTPS traffic.

Burp Suite is powerful because it lets testers **see what normally stays hidden**—requests, responses, parameters, cookies, and tokens. Attackers abuse this. Defenders must understand it.

Key Features of Burp Suite

1. Intercepting Proxy

This is the heart of Burp Suite.

- Captures HTTP/HTTPS requests and responses
- Allows inspection and modification of data in transit
- Helps test authentication, session handling, and input validation

If you can intercept it, you can break it—or protect it.

---

2. Spider (Crawler)

Automatically explores the web application.

- Maps pages, links, and parameters
- Helps understand application structure
- Saves time during reconnaissance

Old-school recon, just automated.

---

3. Scanner (Professional Version)

Performs automated vulnerability scanning.

- Detects SQL Injection, XSS, CSRF, etc.
- Identifies security misconfigurations
- Generates detailed reports

Fast, effective—but never a replacement for human logic.

---

4. Intruder

Used for automated attacks and fuzzing.

- Brute-force login pages
- Test parameter manipulation
- Identify weak input validation

Precision chaos. Controlled damage.

---

5. Repeater

Manual testing at its finest.

- Modify and resend HTTP requests
- Observe server responses in real time
- Ideal for validating vulnerabilities

This is where real hackers slow down and think.

6. Sequencer

Analyzes randomness of tokens.

- Tests session IDs
- Detects predictable tokens
- Identifies weak session management

Weak randomness = stolen sessions.

---

7. Decoder

Encodes and decodes data.

- Base64
- URL encoding
- HTML encoding

Because attackers speak many languages—machines included.

---

8. Comparer

Compares two pieces of data.

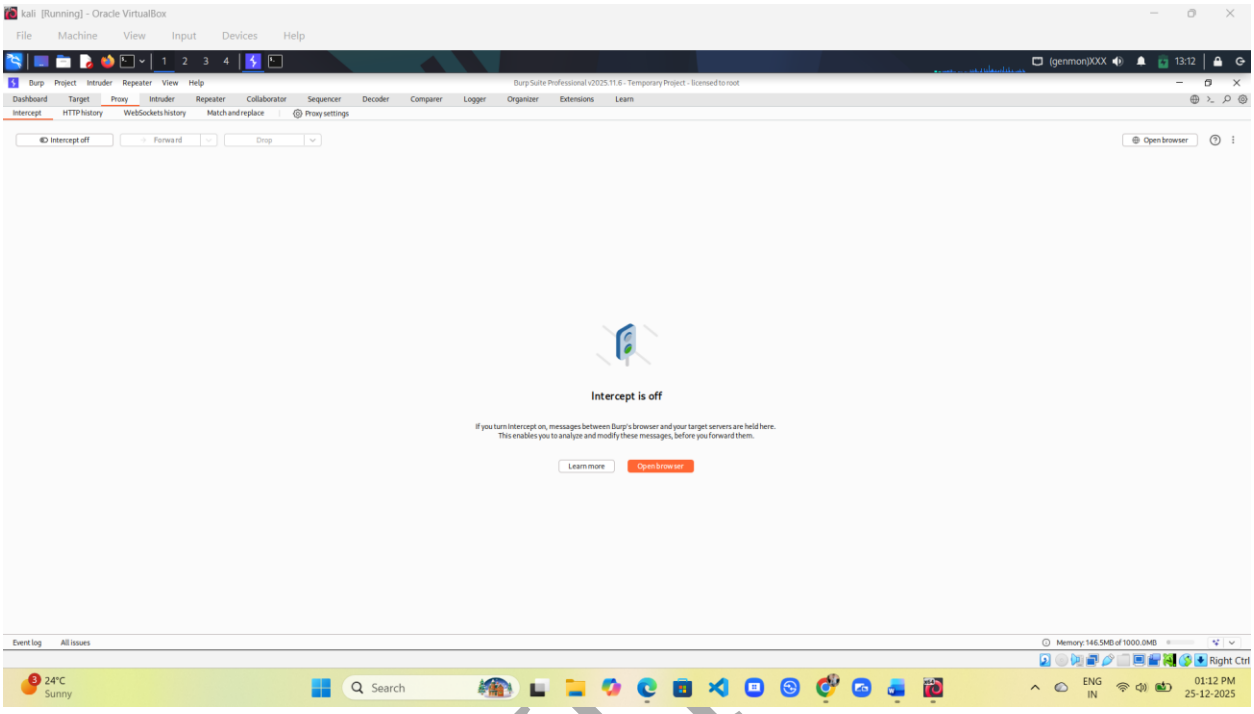
- Requests vs requests
- Responses vs responses

How to Use Burp Suite

1. Open **Kali Linux**
2. Launch terminal
3. Start Burp Suite:
4. `burpsuite`
5. Configure browser proxy (usually 127.0.0.1:8080)
6. Enable **Intercept ON**
7. Browse the target web application
8. Analyze requests using Burp tools

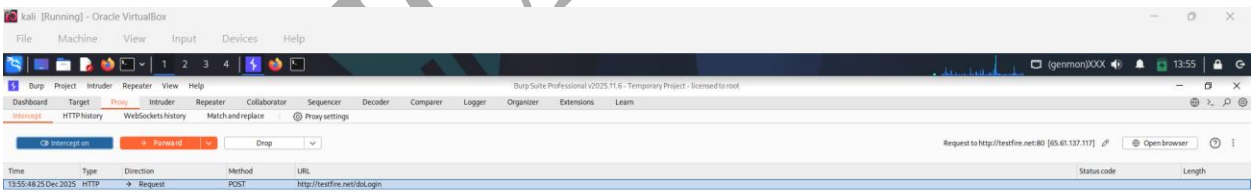
INTRUDER

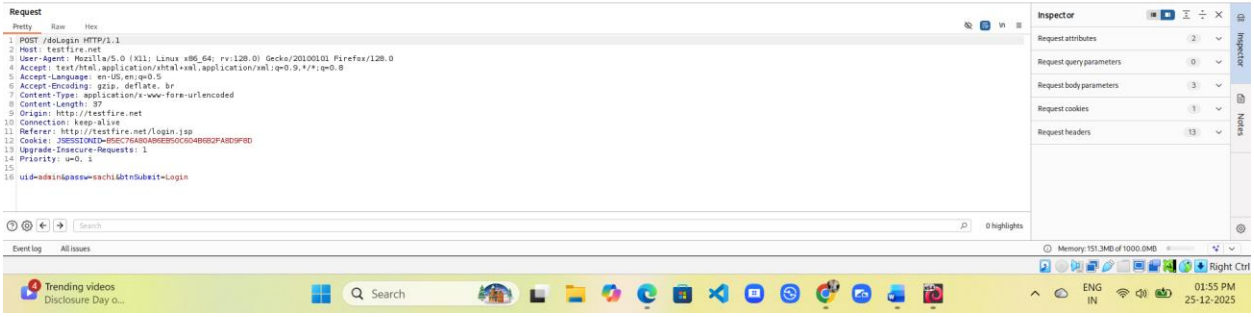
- start interception in burp suite for intercept the request



• Login Request intercept

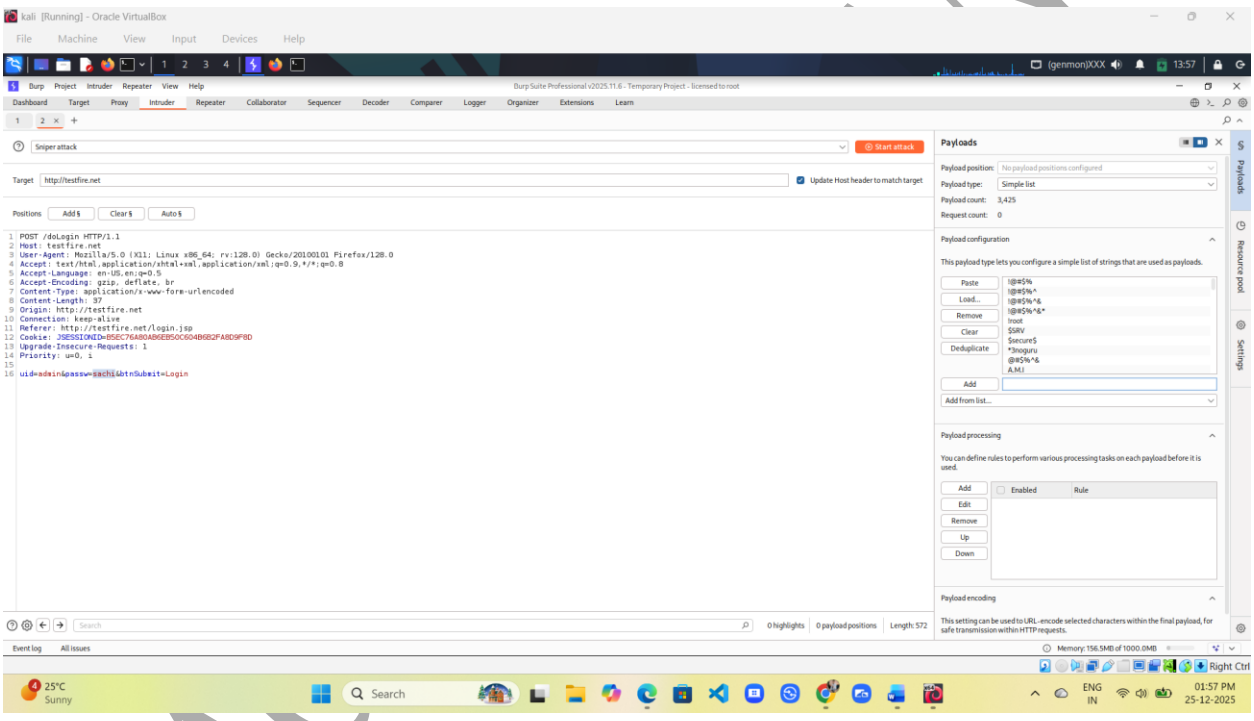
• Now , right click on request and send it to the intruder



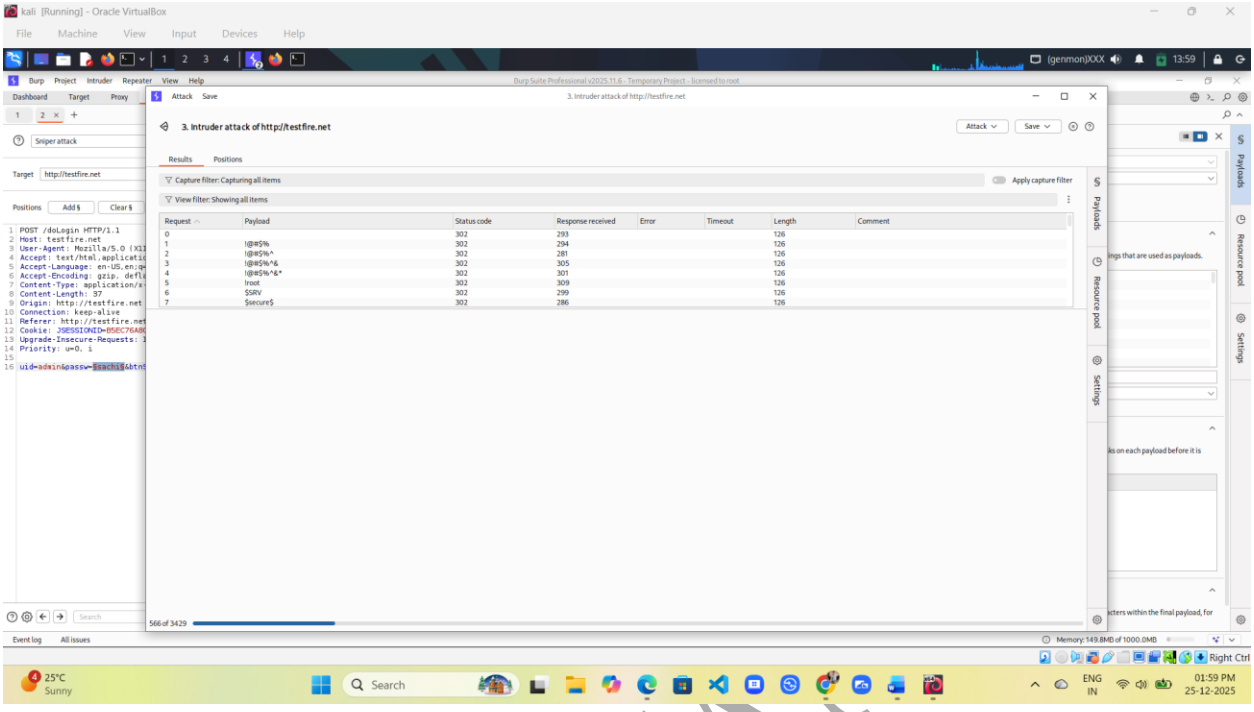


1. Sniper attack

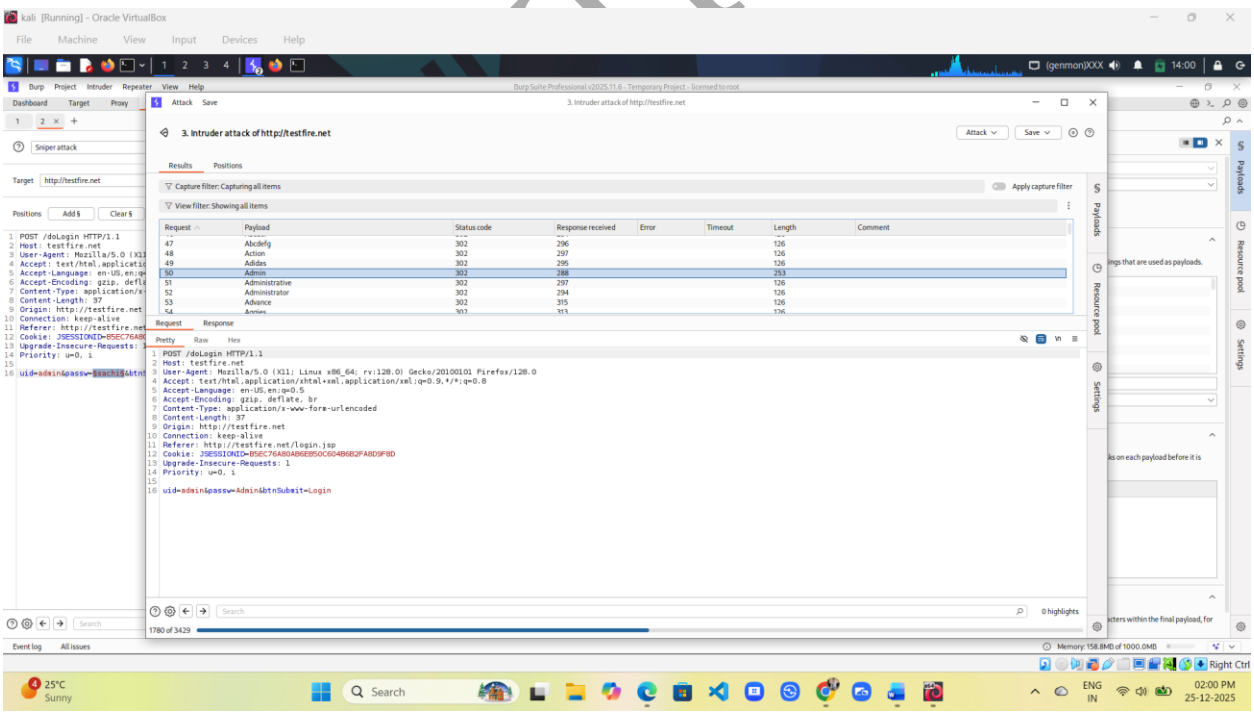
- Intruder interface
- Now select the username section and click on Add$
- Same step on password
- Provide a usernames in payload configuration section
- And add passwords in payload configuration section

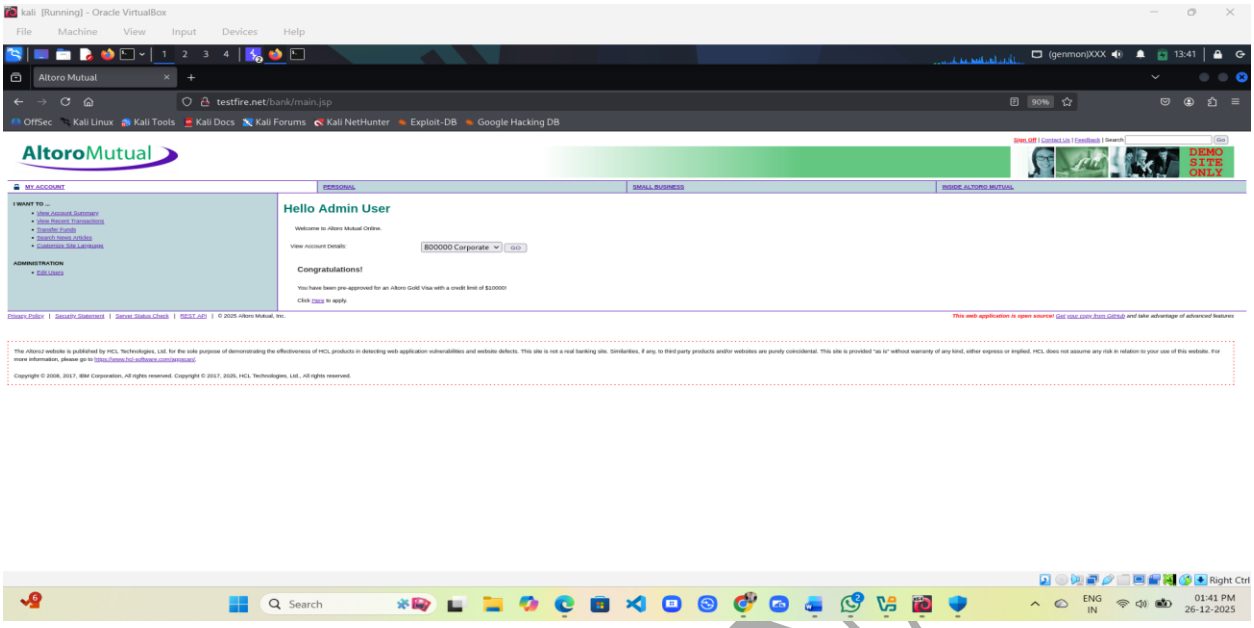


- Now , click on start attack button
- Here , brute force attack started



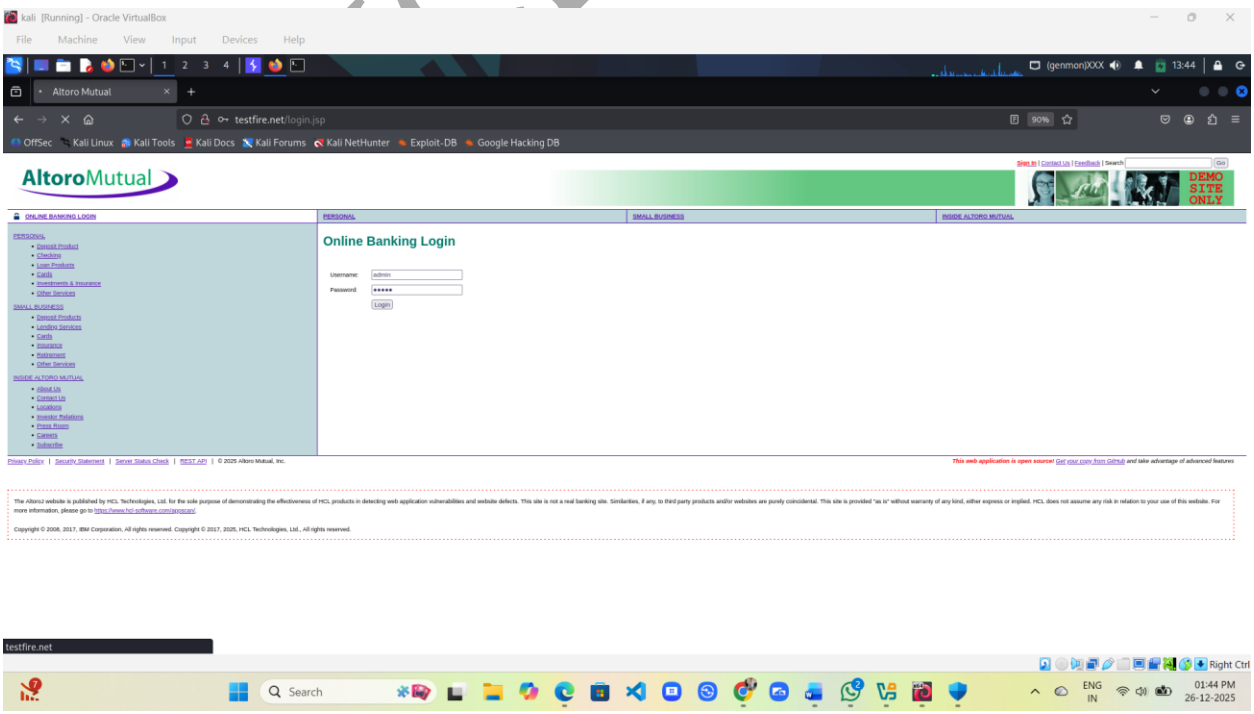
• Here password find



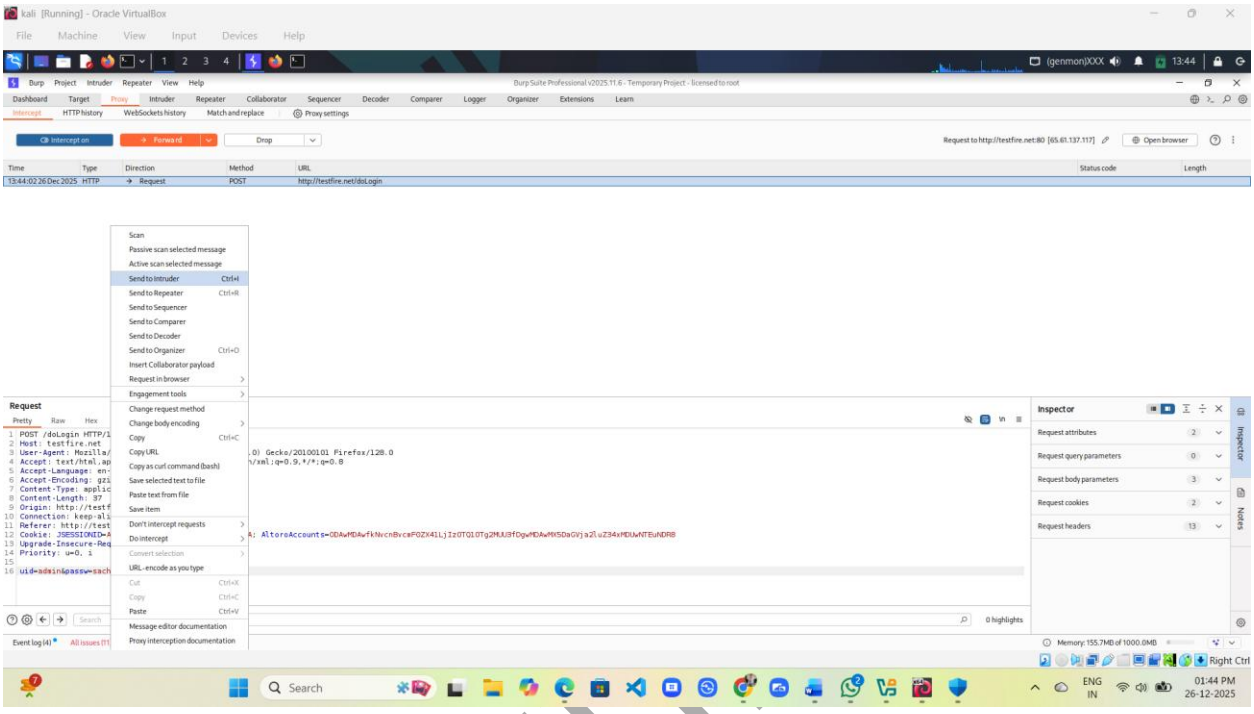


2. Battering ram attack

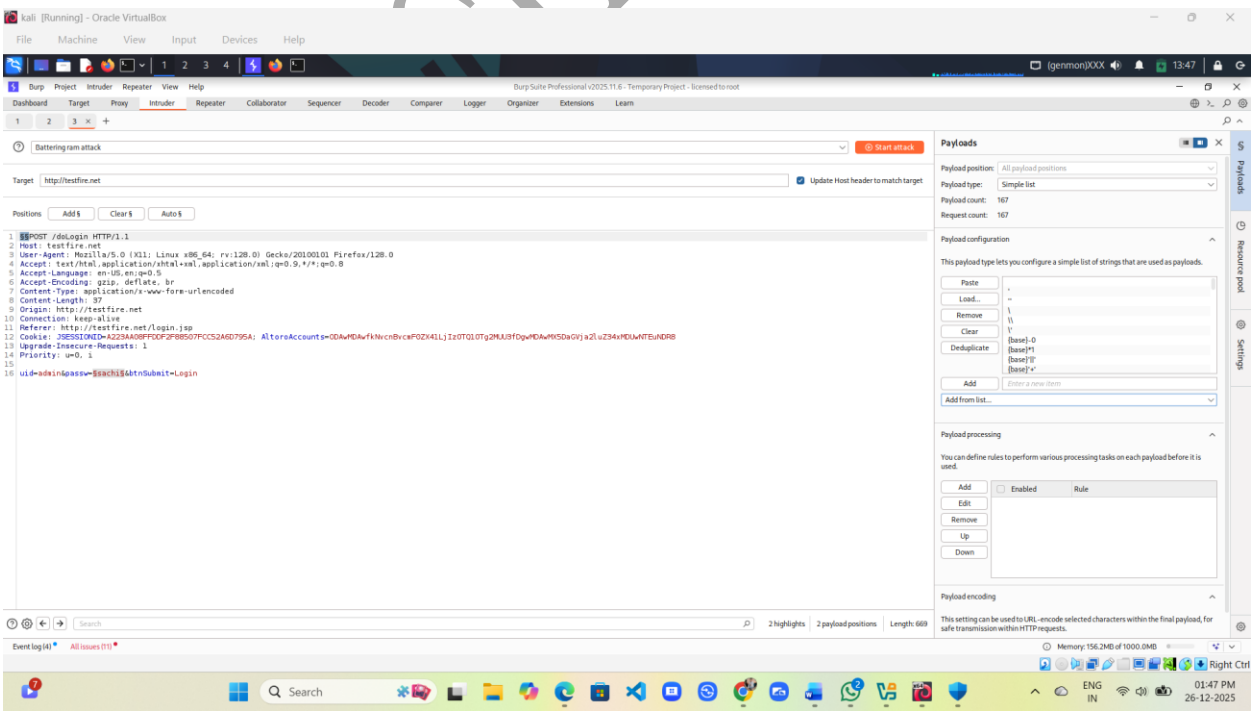
- Target Website
- Enter username and password and click on login
- Account Login



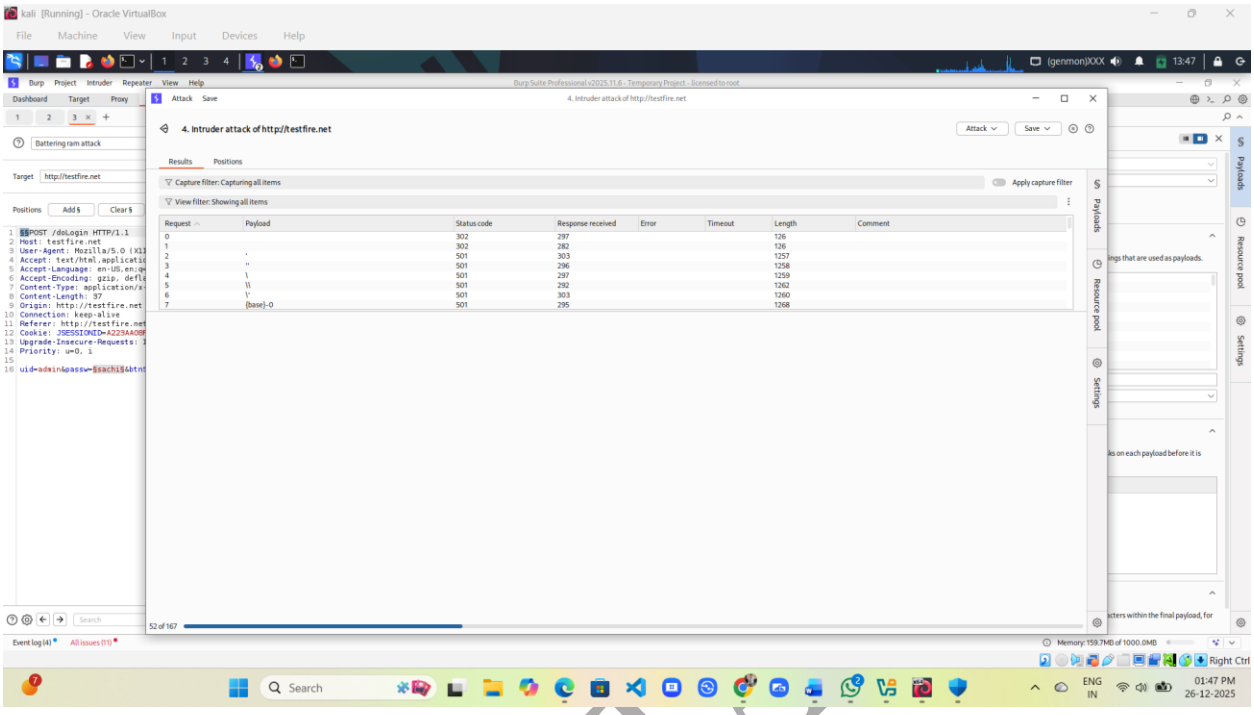
- Open burp suite and click on target option
- Send to Intruder



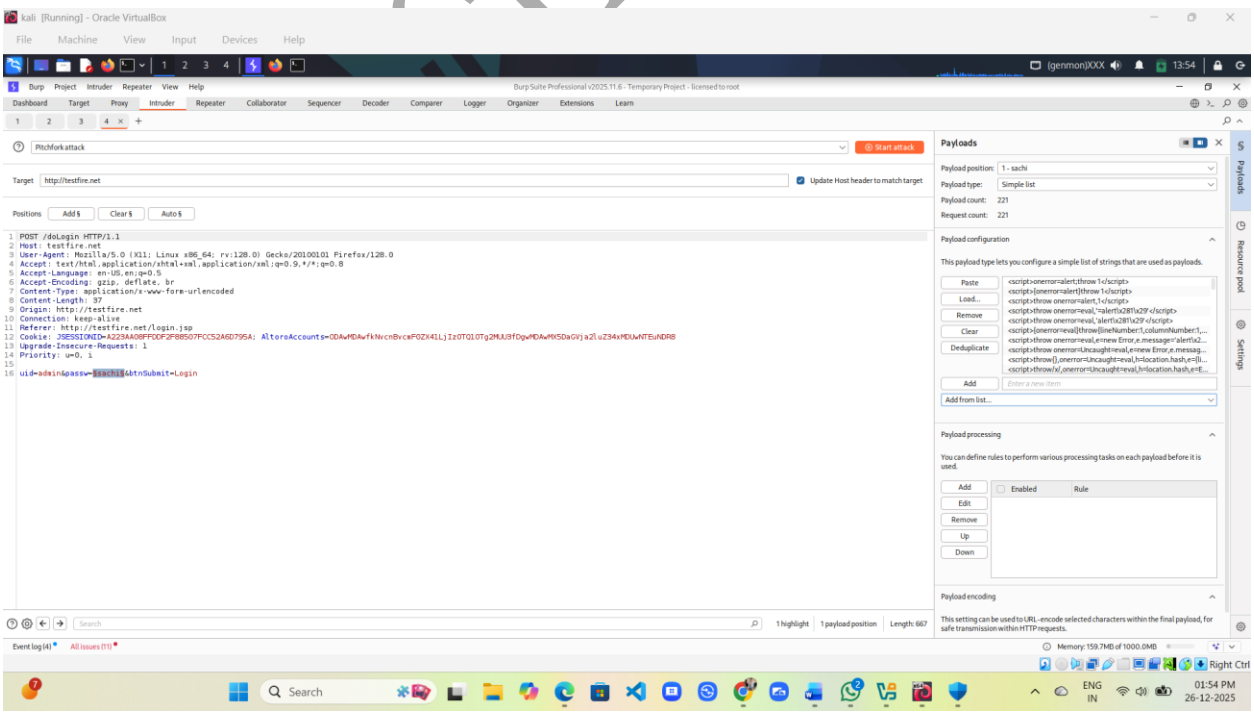
- Add Payloads

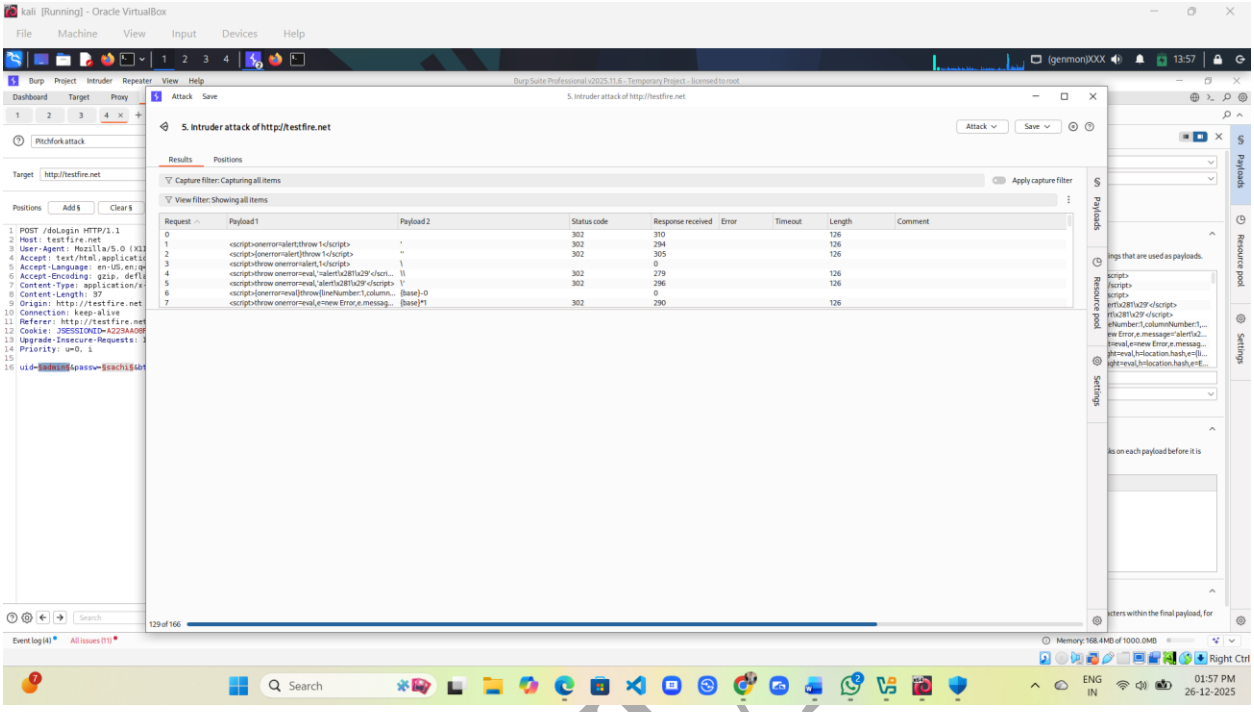


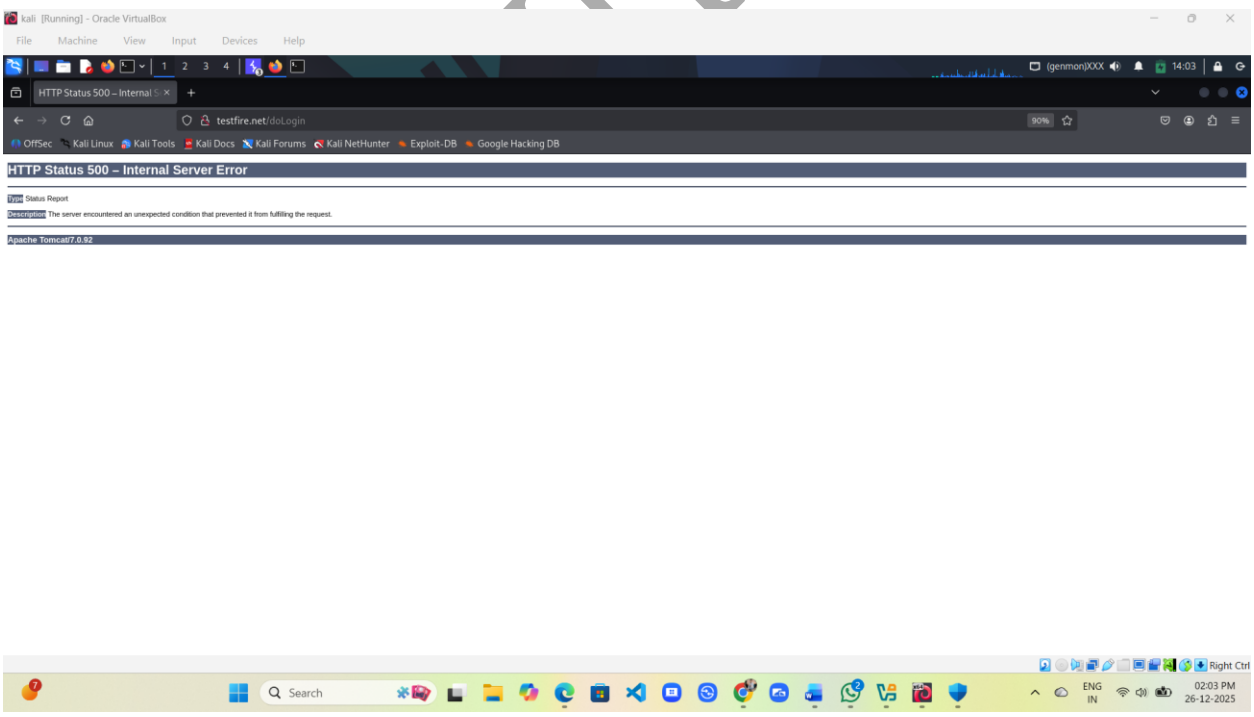
Attack started



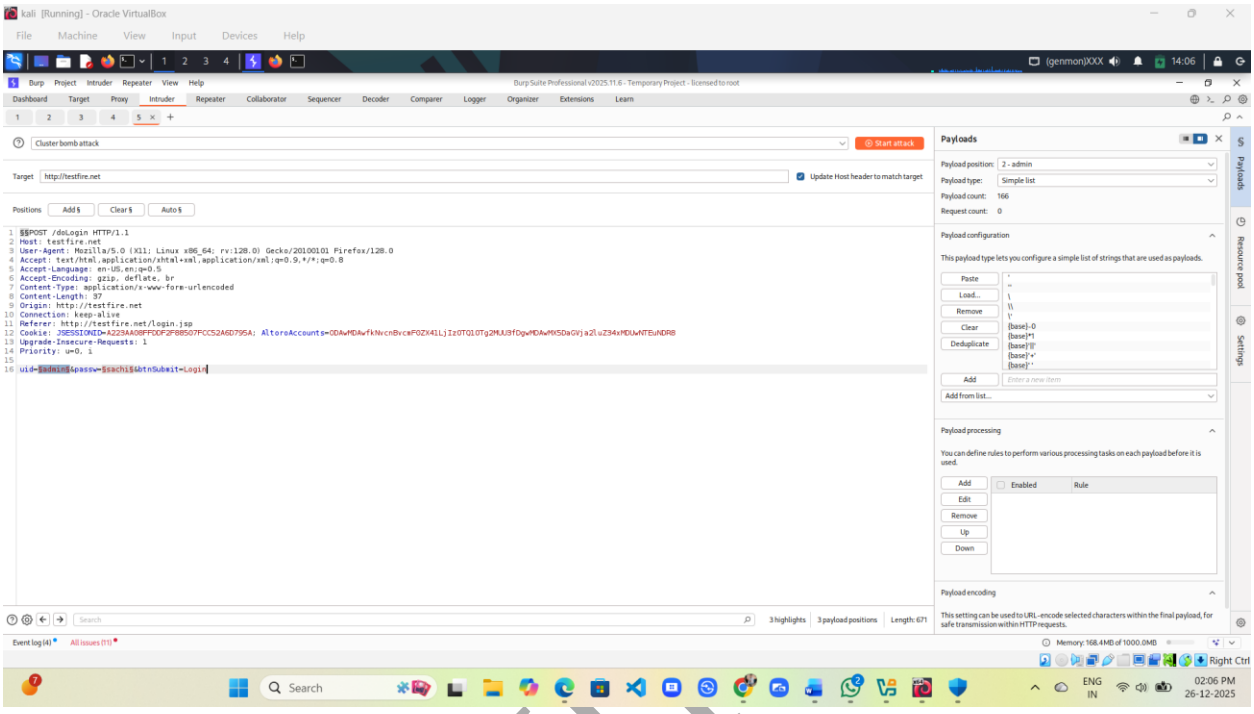
3. Pitchfork attack

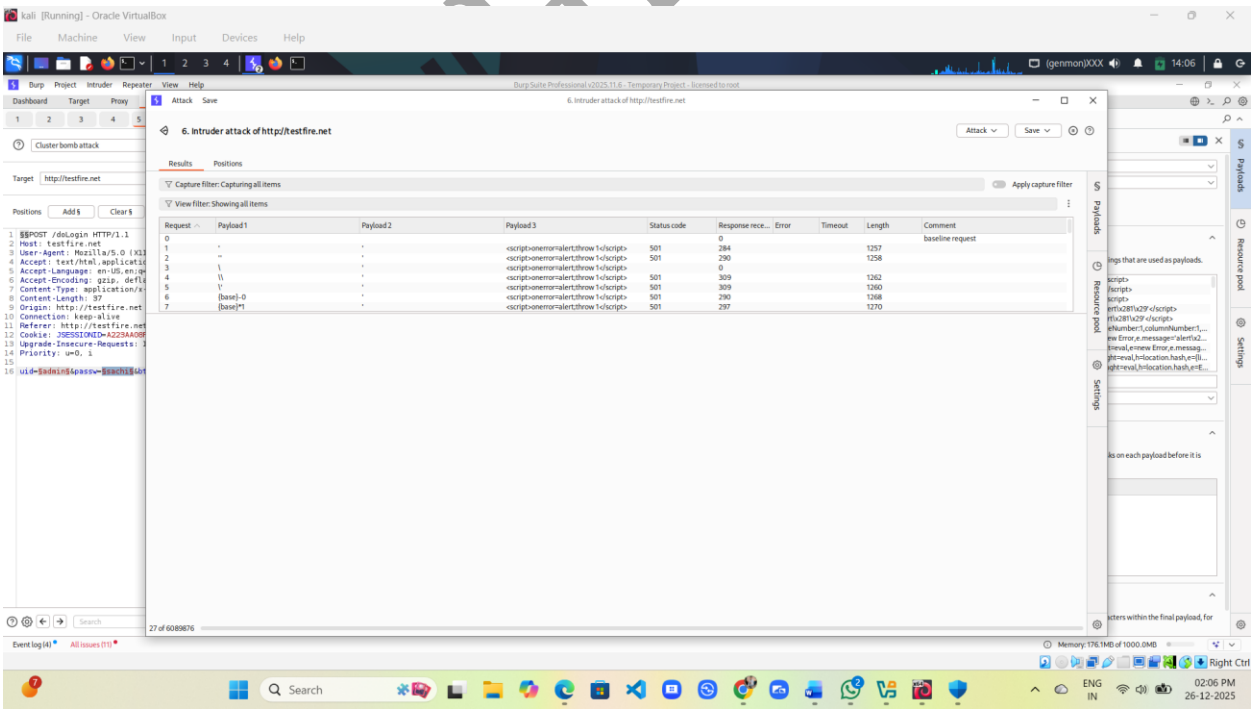


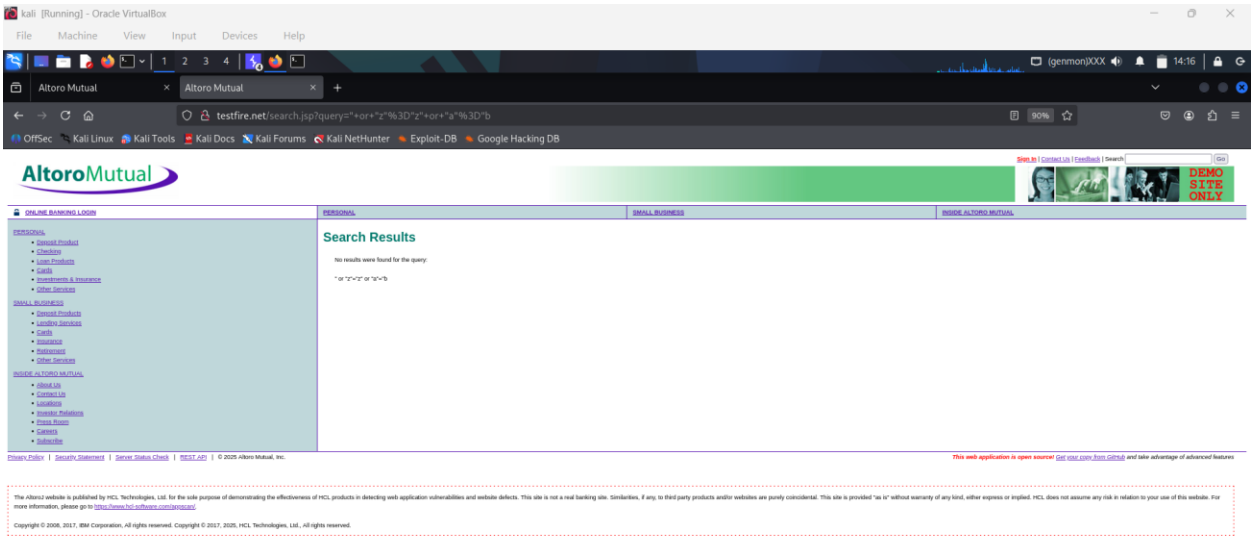

kali [Running] - Oracle VirtualBox
5. Intruder attack of http://testfire.net
Request
Payload 1
Payload 2
Status code
Response received
Error
Timeout
Length
Comment
01:57 PM
26-12-2025


kali [Running] - Oracle VirtualBox
HTTP Status 500 – Internal Server Error
Type Status Report
Description The server encountered an unexpected condition that prevented it from fulfilling the request.
Apache Tomcat/7.0.92
02:03 PM
26-12-2025

4. Cluster bomb attack

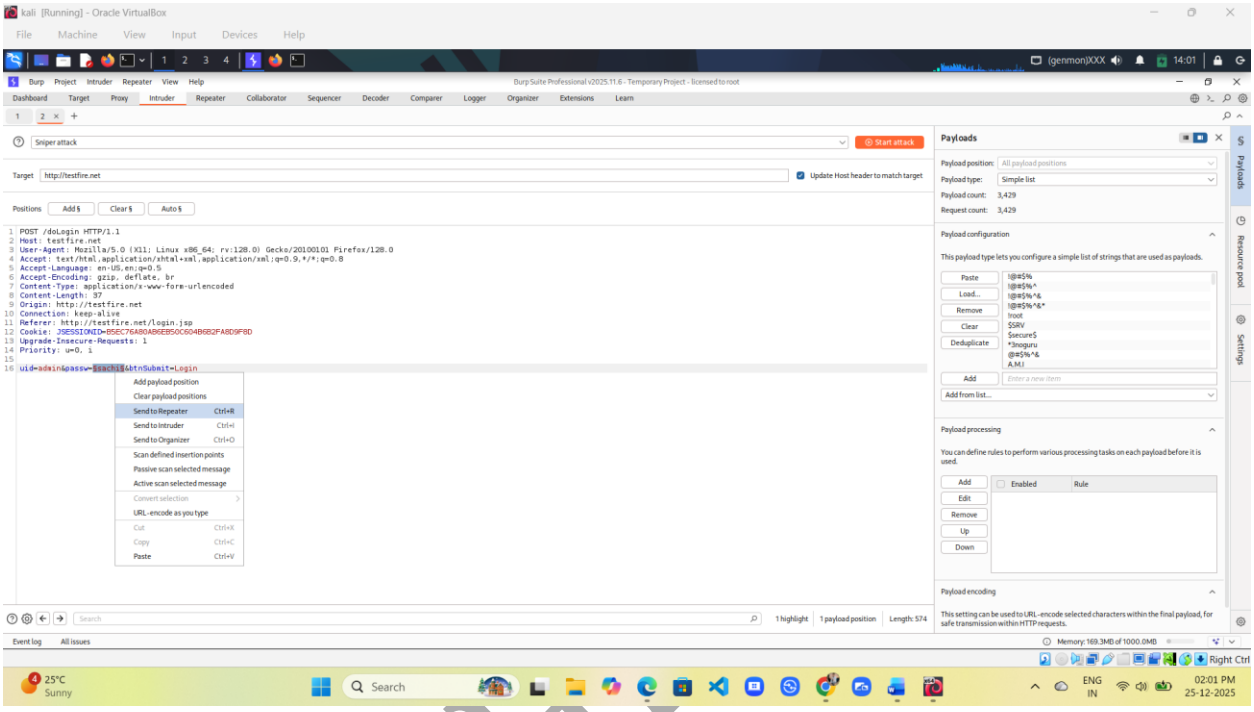


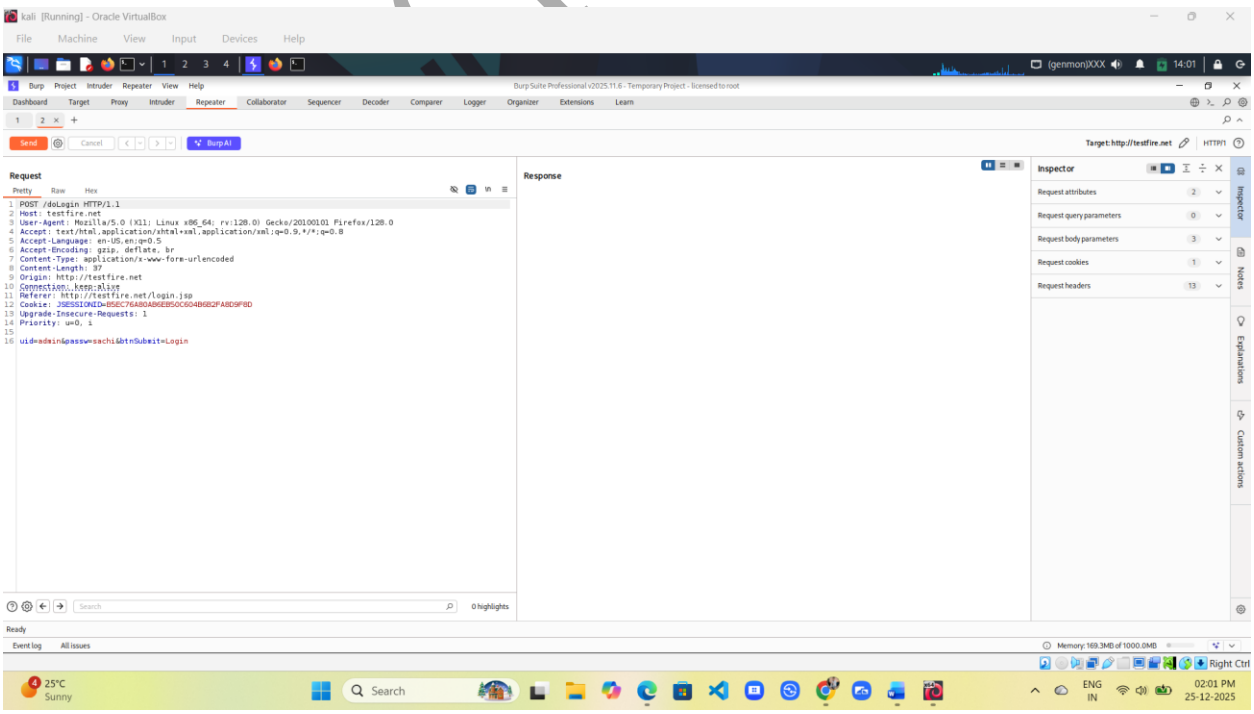


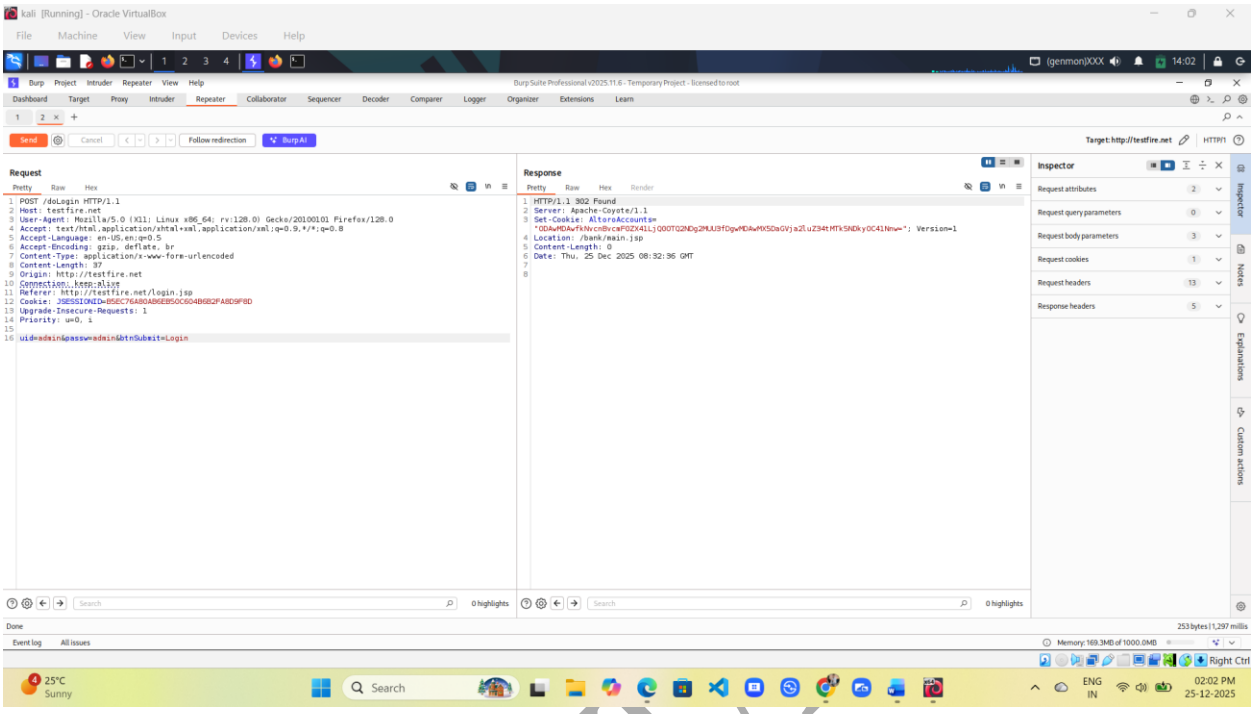


REPEATER

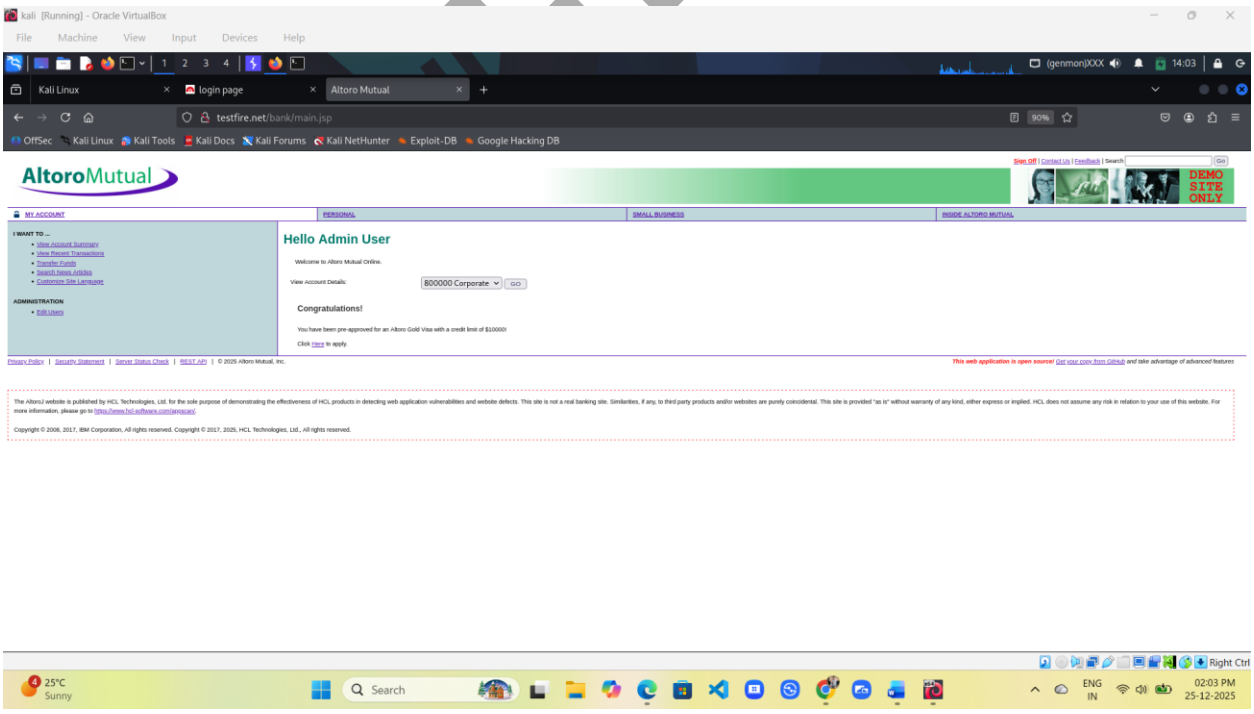
• Now , right click on request and send it to the repeater





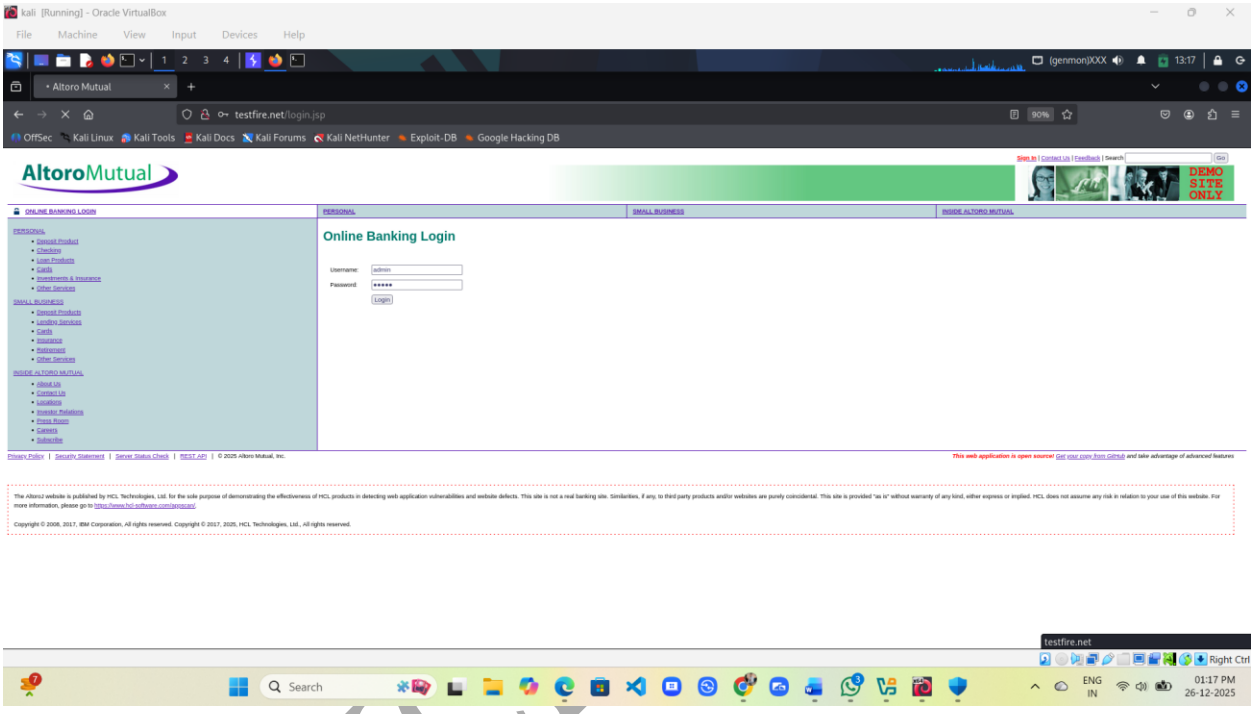


- Now Login using password

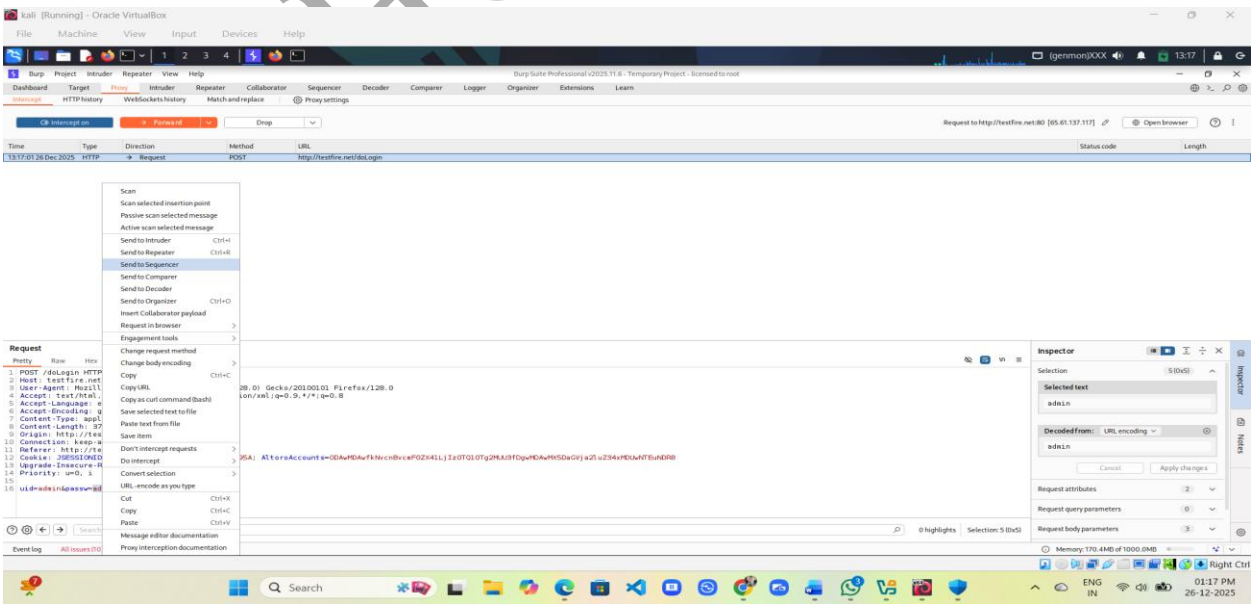


SEQUENCER

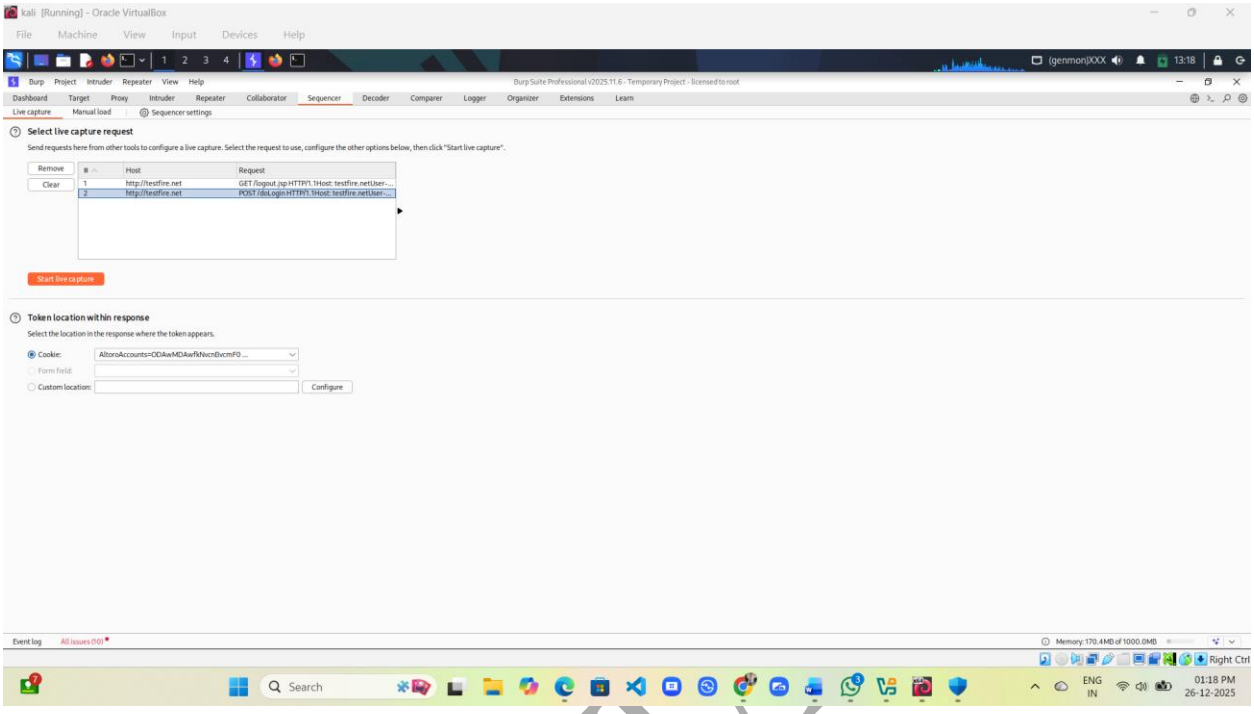
- Target Website
- Enter username and password and click on login
- Account Login



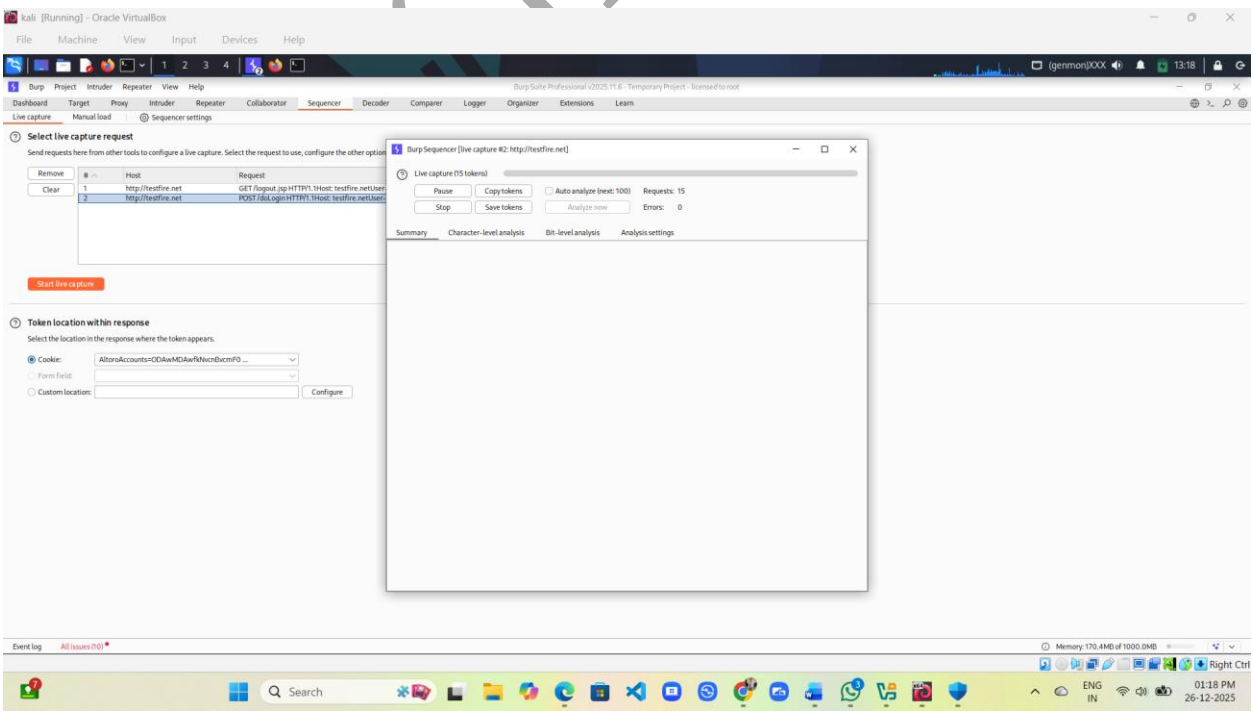
- Send to Sequencer

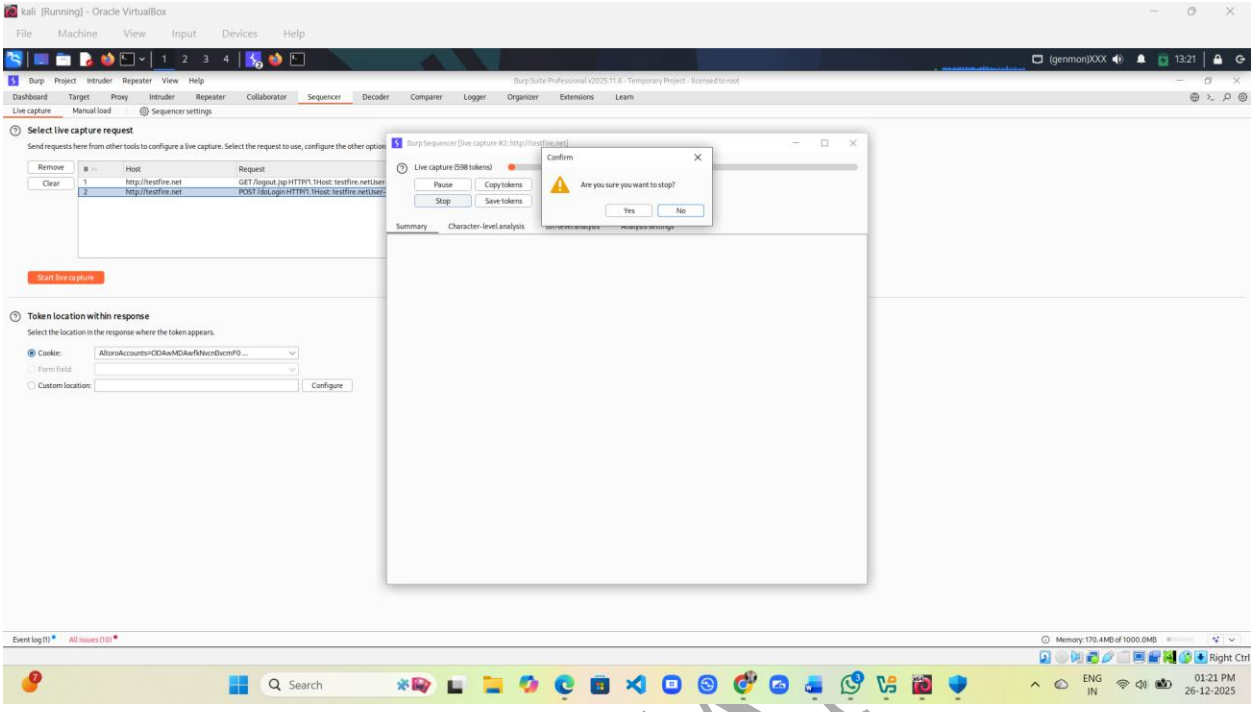


- Click on Start live capture

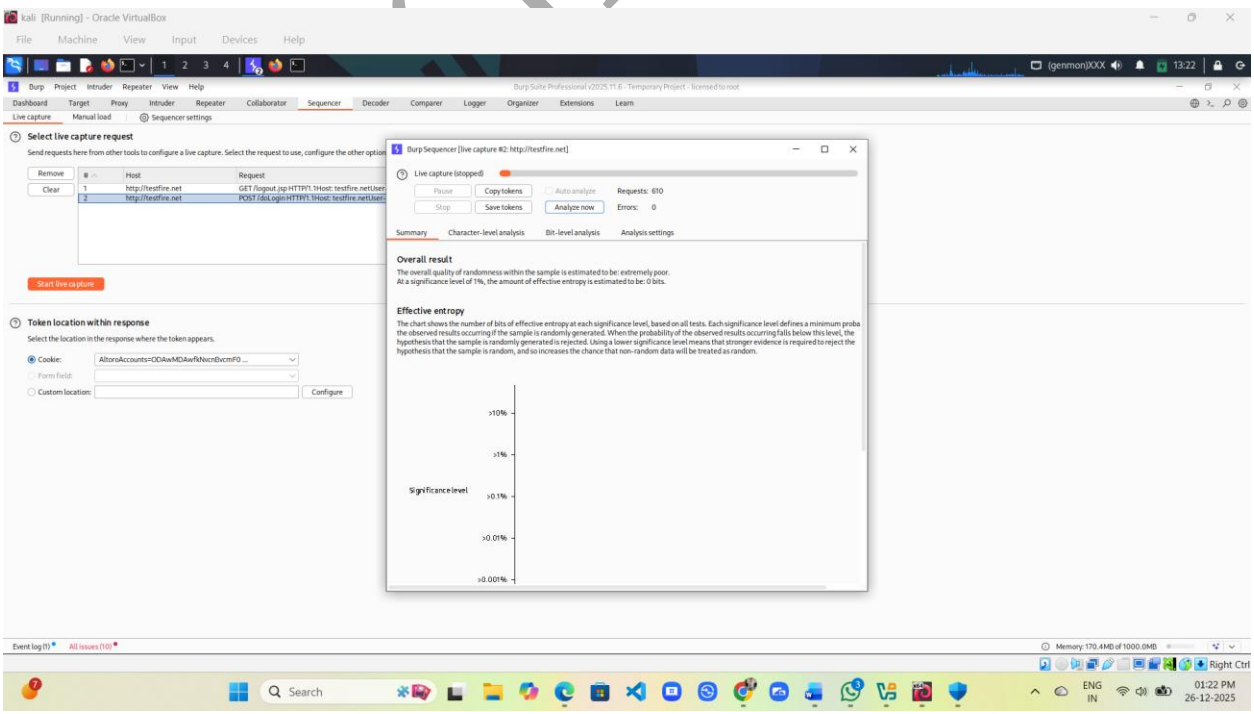


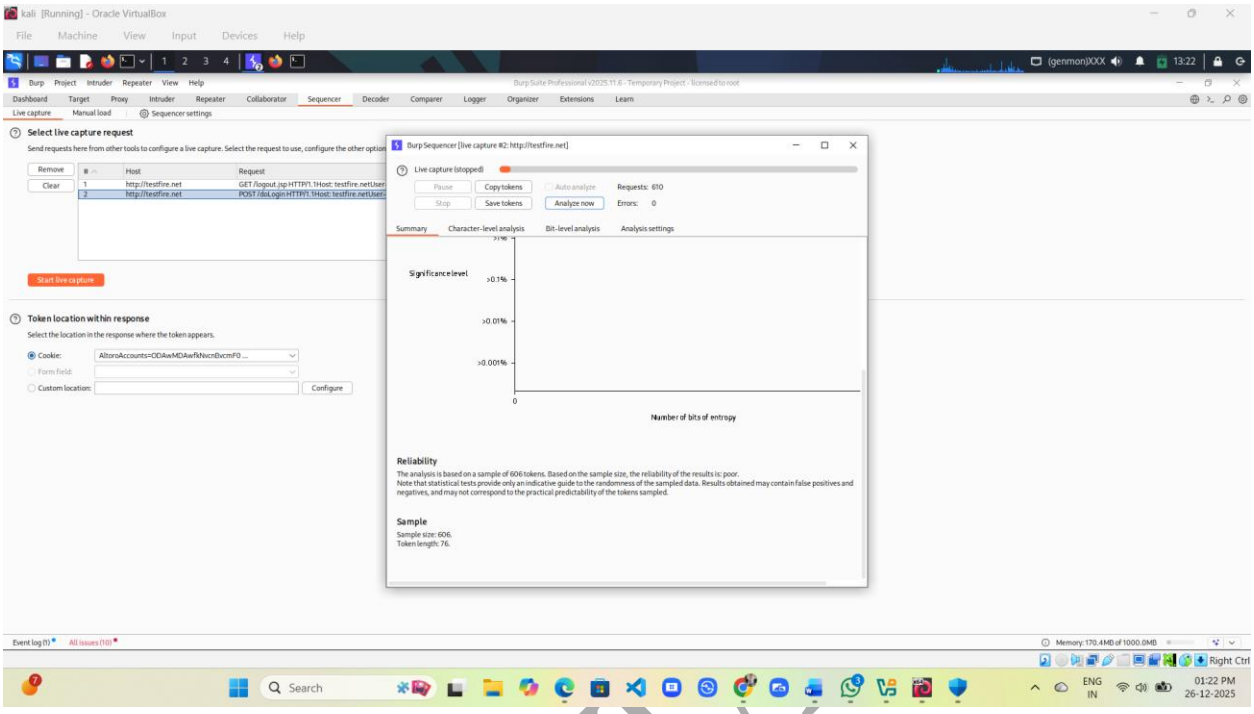
- Started





- Now click on Analyze now
- Here , result




kali [Running] - Oracle VirtualBox
Burp Sequencer [live capture #2: http://testfire.net]
Select live capture request
Token location within response
Reliability
Sample
Search

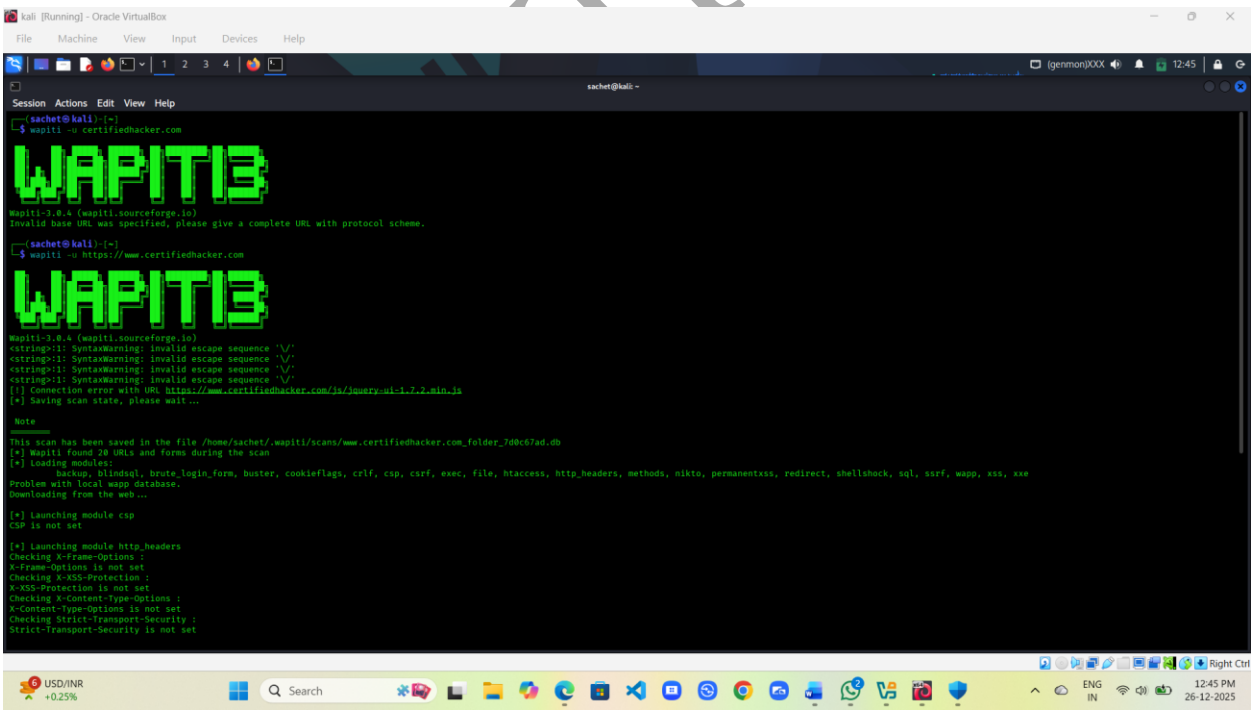
Detect Web Application Vulnerabilities using Various Web Application Security Tools

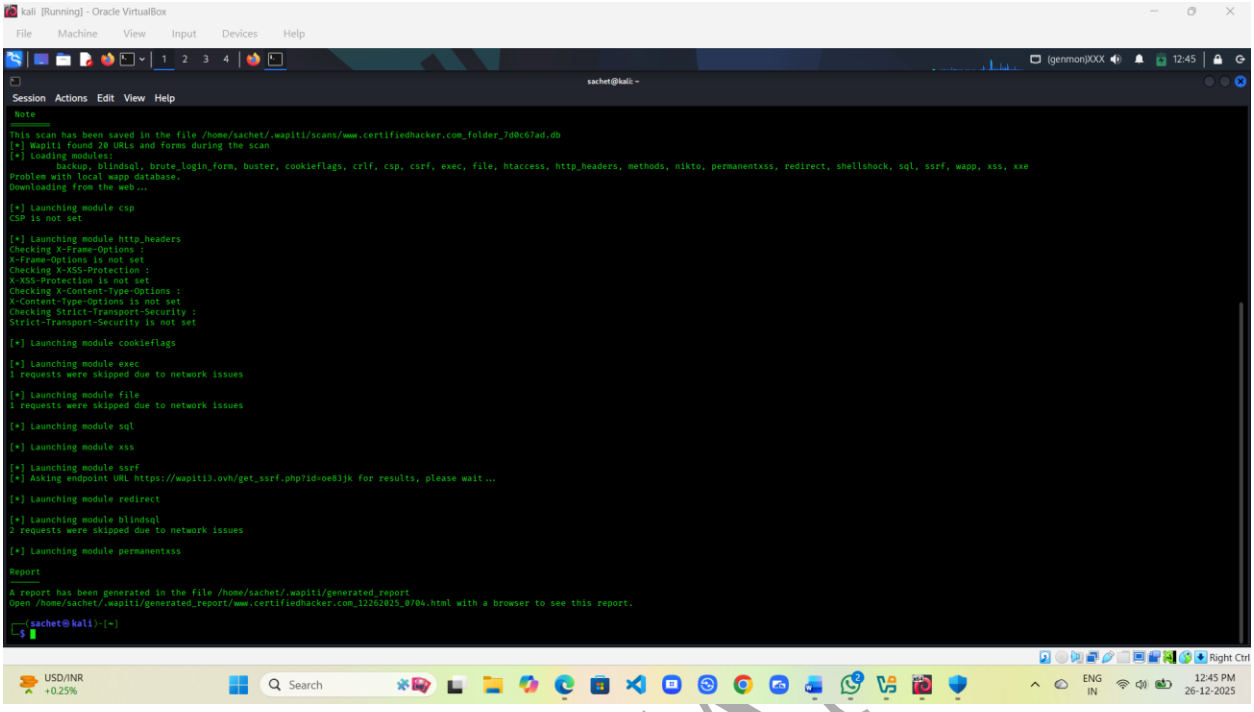
Ethical hackers and pen testers are aided in the discovery of web application vulnerabilities with the help of various tools that make the detection of web application vulnerabilities an easy task.

Lab Scenario

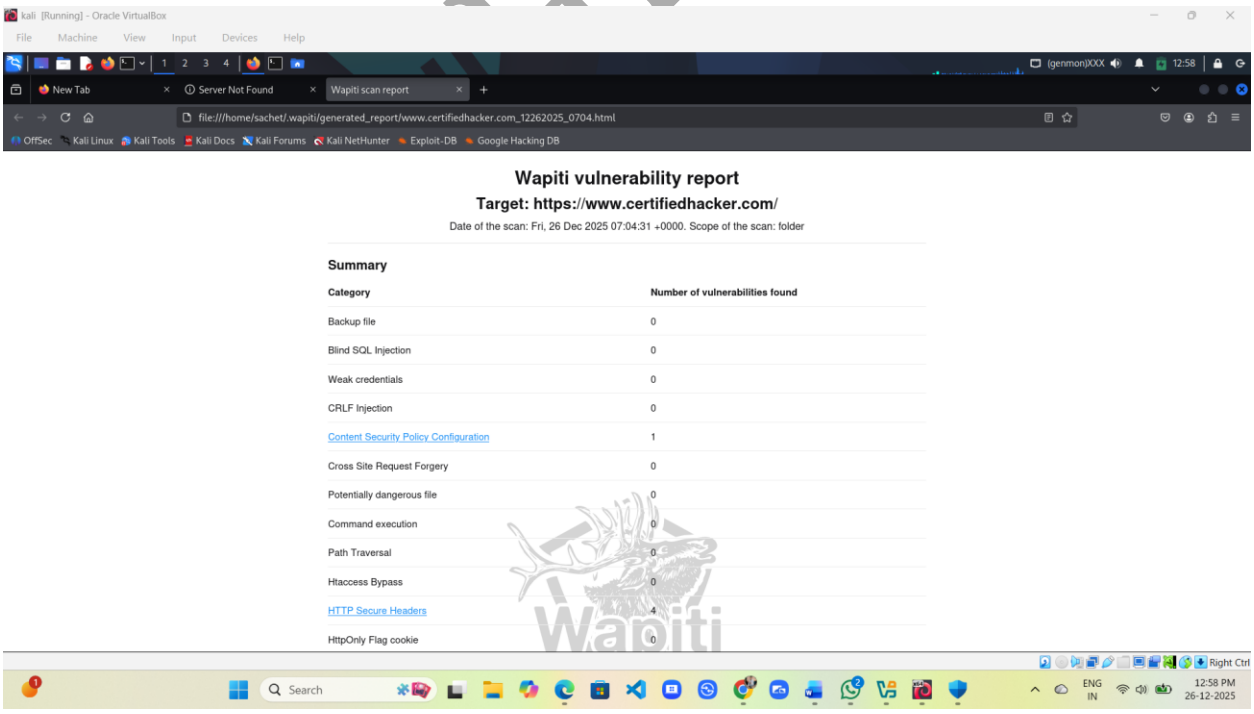
When talking about web applications, organizations consider security to be a critical component, because web applications are a major source of attacks. Attackers try various application-level attacks to compromise the security of web applications to commit fraud or steal sensitive information.

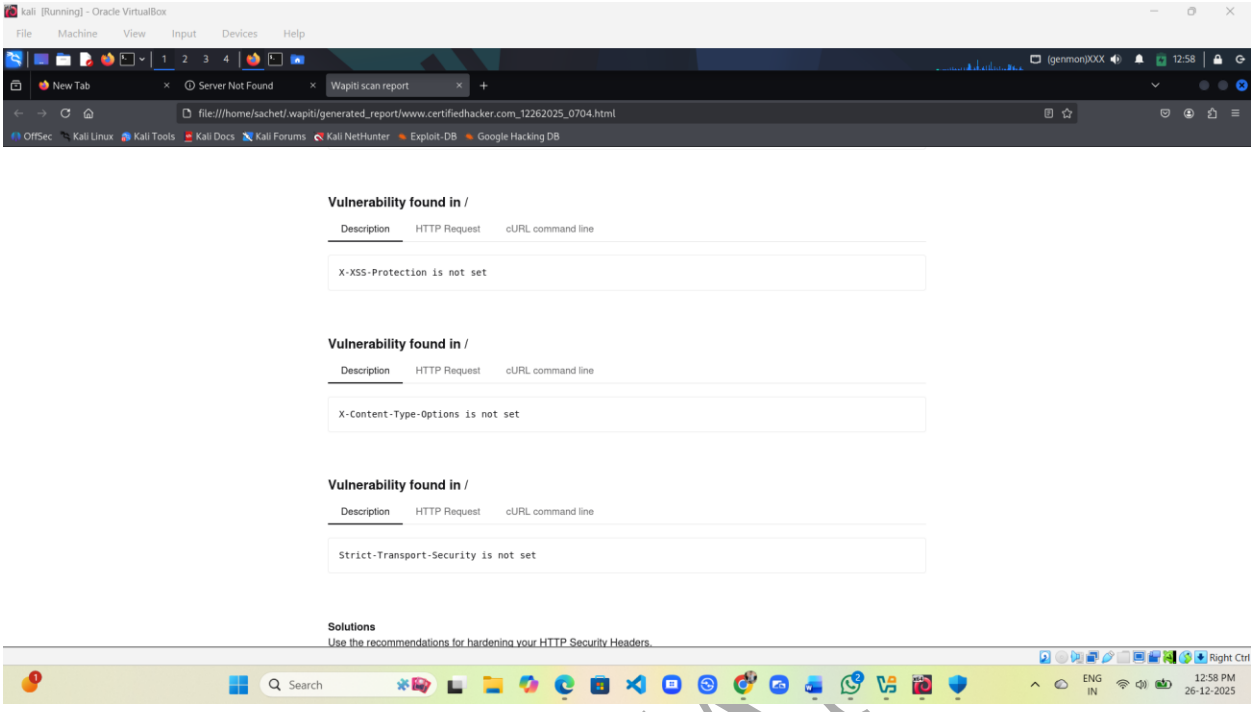
wapiti -u https://www.certifiedhacker.com




kali [Running] - Oracle VirtualBox
File Machine View Input Devices Help
sachet@kali: ~
Session Actions Edit View Help
Note
This scan has been saved in the file /home/sachet/.wapiti/scans/www.certifiedhacker.com_folder_7d0c67ad.db
[+] Wapiti found 20 URLs and forms during the scan
[+] Loading modules:
backup, blindsql, brute_login_form, buster, cookieflags, crlf, csp, csrf, exec, file, htaccess, http_headers, methods, nikto, permanentxss, redirect, shellshock, sql, ssrf, wapp, xss, xxe
Problem with local wapp database.
Downloading from the web ...
[+] Launching module csp
CSP is not set
[+] Launching module http_headers
Checking X-Frame-Options :
X-Frame-Options is not set
Checking X-XSS-Protection :
X-XSS-Protection is not set
Checking X-Content-Type-Options :
X-Content-Type-Options is not set
Checking Strict-Transport-Security :
Strict-Transport-Security is not set
[+] Launching module cookieflags
[+] Launching module exec
1 requests were skipped due to network issues
[+] Launching module file
1 requests were skipped due to network issues
[+] Launching module sql
[+] Launching module xss
[+] Launching module ssrf
[+] Asking endpoint URL https://wapiti3.ovh/get_ssrf.php?id=oe83jk for results, please wait ...
[+] Launching module redirect
[+] Launching module blindsql
2 requests were skipped due to network issues
[+] Launching module permanentxss
Report
A report has been generated in the file /home/sachet/.wapiti/generated_report
Open /home/sachet/.wapiti/generated_report/www.certifiedhacker.com_12262025_0704.html with a browser to see this report.
(sachet㉿kali)-[~]
$
Right Ctrl
USD/INR +0.25%
Search
ENG IN
12:45 PM 26-12-2025

Report


kali [Running] - Oracle VirtualBox
File Machine View Input Devices Help
New Tab
Server Not Found
Wapiti scan report
file:///home/sachet/.wapiti/generated_report/www.certifiedhacker.com_12262025_0704.html
OffSec Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB
Wapiti vulnerability report
Target: https://www.certifiedhacker.com/
Date of the scan: Fri, 26 Dec 2025 07:04:31 +0000. Scope of the scan: folder
Summary
Category
Number of vulnerabilities found
Backup file 0
Blind SQL Injection 0
Weak credentials 0
CRLF Injection 0
Content Security Policy Configuration 1
Cross Site Request Forgery 0
Potentially dangerous file 0
Command execution 0
Path Traversal 0
Htaccess Bypass 0
HTTP Secure Headers 4
HttpOnly Flag cookie 0
Wapiti
Right Ctrl
Search
ENG IN
12:58 PM 26-12-2025


kali [Running] - Oracle VirtualBox
File Machine View Input Devices Help
New Tab
Server Not Found
Wapiti scan report
file:///home/sachet/.wapiti/generated_report/www.certifiedhacker.com_12262025_0704.html
OffSec Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB
Vulnerability found in /
Description HTTP Request cURL command line
X-XSS-Protection is not set
Vulnerability found in /
Description HTTP Request cURL command line
X-Content-Type-Options is not set
Vulnerability found in /
Description HTTP Request cURL command line
Strict-Transport-Security is not set
Solutions
Use the recommendations for hardening your HTTP Security Headers.
Search
Right Ctrl
ENG IN
12:58 PM 26-12-2025

Perform Web Application Hacking using AI

Conducting web application hacking with Al involves leveraging machine learning algorithms to automate the detection and exploitation of vulnerabilities, enhancing the efficiency and effectiveness of penetration testing.

Lab Scenario

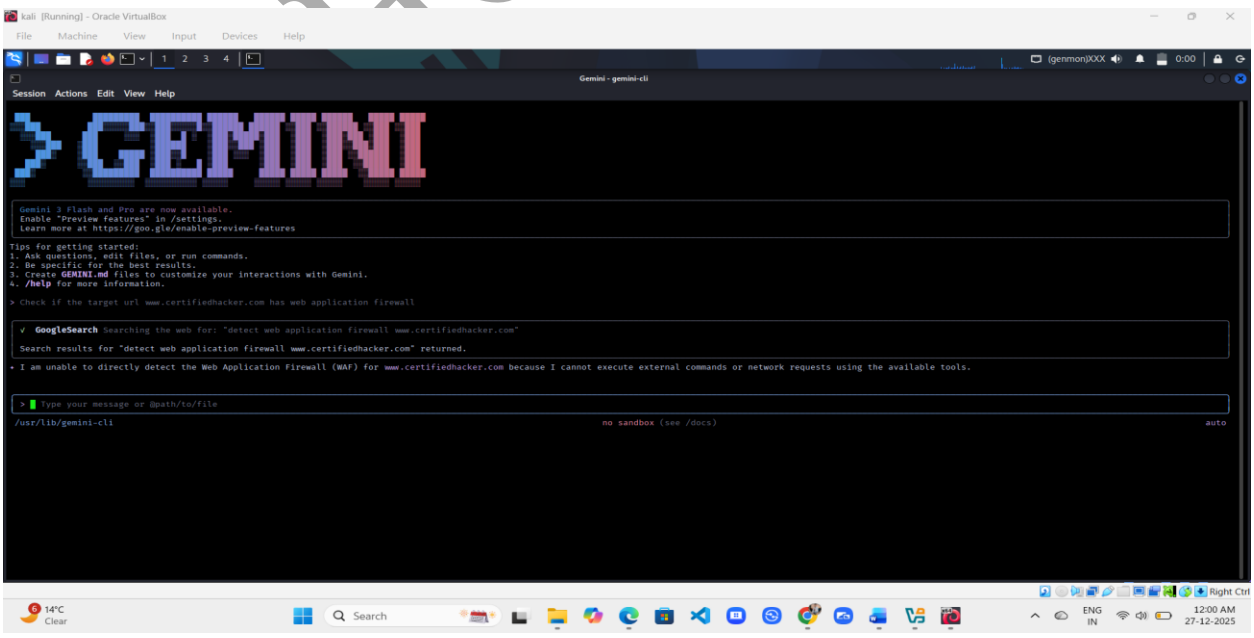
Hacking web applications using Al involves leveraging advanced machine learning techniques to exploit vulnerabilities in web applications. This approach can automate and enhance the traditional methods of penetration testing and vulnerability assessment.

The labs in this exercise demonstrate how to perform web application hacking using Al.

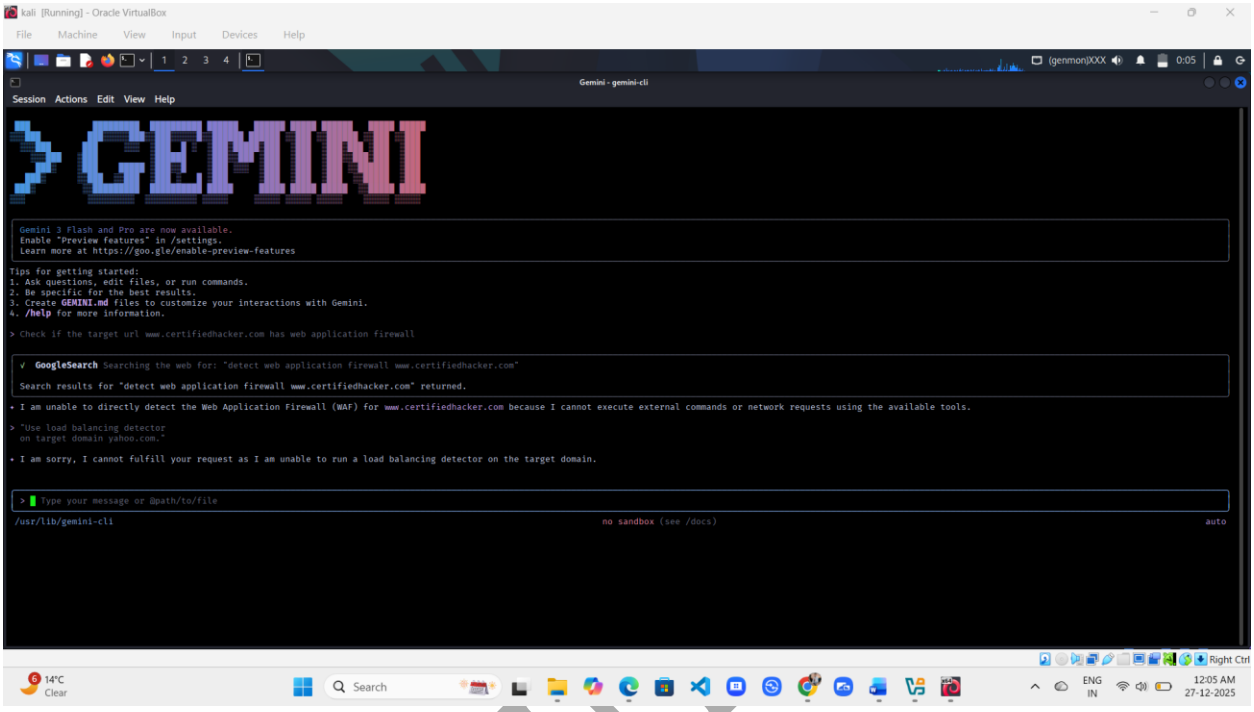
Lab Objectives

Perform web application hacking using Gemini-CLI

Prompt - "Check if the target url www.certifiedhacker.com has web application firewall"



Prompt - "Use load balancing detector on target domain yahoo.com."



Countermeasures Against Web Application Hacking

1. Strong Input Validation (The First Line of Truth)

If user input isn't trusted, attackers starve.
Every single input—forms, headers, cookies, URLs—must be:

- Validated on the **server-side**
- Checked for type, length, format, and range

Client-side checks are cute. Server-side checks are law.

---

2. Proper Output Encoding (XSS Kryptonite)

Never send raw user input back to the browser.

- Encode output for HTML, JavaScript, URL, and CSS contexts
- Use secure templating engines

XSS exists because developers believed input was "harmless." History disagrees.

---

3. Use Secure Authentication & Session Management

Sessions are identity. Protect them like crown jewels.

- Strong passwords + hashing (bcrypt, Argon2)
- Multi-Factor Authentication (MFA)
- Secure, HttpOnly, SameSite cookies
- Session timeout and regeneration

If sessions are weak, attackers don't brute-force—they borrow.

---

4. Enforce Access Control Properly

Never trust URLs. Never trust roles sent from the client.

- Validate authorization on every request

- Apply **least privilege** principle
- Block IDOR by checking ownership server-side

If changing a number in a URL grants access—congrats, you built a vulnerability.

---

5. Protect Against SQL Injection

Databases should never understand user intent.

- Use **prepared statements / parameterized queries**
- Avoid dynamic SQL
- Use ORM frameworks where possible

String concatenation with SQL is how legends fall.

---

6. Secure File Upload Mechanisms

File uploads are a hacker's favorite hobby.

- Validate file type using MIME + content inspection
- Rename uploaded files
- Store uploads outside the web root
- Disable execution permissions

If users can upload files, attackers will upload ambition.

---

7. Use HTTPS Everywhere

No excuses. None.

- Encrypt data in transit
- Prevent credential sniffing
- Enable HSTS

HTTP is nostalgia. HTTPS is survival.

---

8. Security Configuration Hardening

Default settings are attacker-friendly.

- Disable unnecessary services
- Remove default accounts
- Hide detailed error messages
- Keep debug mode OFF in production

Silence errors. Log them. Attackers don't need tutorials.

---

9. Patch and Update Regularly

Old components are public enemies.

- Update frameworks, libraries, servers
- Remove unused dependencies
- Monitor CVEs

Unpatched systems aren't unlucky—they're predictable.

---

10. Implement Security Logging & Monitoring

If you can't see attacks, you're already losing.

- Log authentication attempts
- Monitor abnormal behavior
- Use alerts and SIEM tools

Attacks should feel loud, not invisible.

---

11. Web Application Firewalls (WAF)

Not a replacement—but a shield.

- Filters malicious payloads
- Blocks known attack patterns
- Buys time during active attacks

WAFs don't fix bad code, but they slow bad actors.

---

12. Regular Security Testing

Security is not “set and forget.”

- Vulnerability scanning
- Penetration testing
- Code reviews
- Bug bounty programs

Web Application Hacking – Module Summary

Web applications are the backbone of modern digital services, but their constant exposure to the internet makes them prime targets for attackers. Web Application Hacking focuses on identifying, exploiting, and responsibly reporting vulnerabilities present in web-based systems to improve overall security.

A web application operates on a **client–server architecture**, where the browser sends requests using HTTP or HTTPS, the server processes them, interacts with backend databases, and returns responses. Any weakness in this flow—especially improper handling of user input—can be exploited.

The **web application hacking process** follows a structured methodology: information gathering to understand the target, scanning to identify technologies and exposed services, vulnerability detection to find security flaws, exploitation to validate those flaws, post-exploitation to assess impact, and reporting to document findings with remediation steps.

Common vulnerabilities include **SQL Injection**, **Cross-Site Scripting (XSS)**, **Command Injection**, and **File Upload flaws**, all of which arise mainly due to poor input validation, insecure design, and weak server-side controls. Understanding HTTP methods such as GET, POST, PUT, DELETE, and OPTIONS is essential, as attackers often abuse them to manipulate application behavior.

The **OWASP Top 10** highlights the most critical web application risks, including broken access control, cryptographic failures, injection flaws, insecure design, security misconfiguration, vulnerable components, authentication failures, integrity issues, lack of logging and monitoring, and server-side request forgery (SSRF). These risks remain relevant because the same development mistakes continue to repeat.

To defend against web application hacking, strong **countermeasures** must be implemented. These include proper input validation and output encoding, secure authentication and session management, strict access control, use of parameterized queries, secure file upload handling, HTTPS enforcement, hardened configurations, regular patching, effective logging and monitoring, deployment of web application firewalls, and continuous security testing.

In conclusion, web application security is not dependent on tools alone but on disciplined development practices and regular testing. Most attacks succeed not because attackers are highly skilled, but because basic security principles are ignored. Understanding both attack techniques and countermeasures is essential for building, testing, and securing modern web applications.

THANK YOU