-ATHARVA KATKAR

Table of Contents

9. **SQL Injection Attack Using Burp Suite**
   9.1 Using Burp Suite Intruder
   9.2 Using Burp Suite Repeater

10**. Cross-Site Scripting (XSS) Attack**
10.1 XSS Concept and Working
10.2 XSS Attack Using Burp Suite Intruder
10.3 XSS Attack Using Burp Suite Repeater

11. **How to Prevent Web Application Attacks / Hacking**

12. **How to Prevent Browser Exploitation**

13. SQL Injection (Concept & Examples)
4.1 Basic SQL Injection Examples
4.2 SQL Injection Cheat Sheet

14. Perform SQL Injection Using Havij Tool
5.1 Key Features of Havij
5.2 Legitimate Use Cases of Havij
5.3 Target Website Testing
5.4 Database Extraction using Havij

15. Perform SQL Injection Using SQLMap Tool

16. How to Prevent From SQL Injection Attacks

Web Application Hacking

Web Application Hacking is the process of identifying, exploiting, and documenting security vulnerabilities in web-based applications. The primary goals are:

- To discover flaws that can be exploited by attackers.
- To understand how attackers manipulate web applications.
- To assist developers in fixing security weaknesses to improve application security.

How Web Applications Work (Basic Architecture)

A web application follows a **client–server model**, where:

Component	Description
Client	The web browser used to send requests to the server.
Server	Processes the client requests and sends back responses.
Protocol	Communication between client and server happens through **HTTP/HTTPS**.

Example HTTP Request

GET /index.html HTTP/1.1

Host: example.com

In this example, the browser requests the webpage index.html from the server example.com, and the server returns the requested content.

Web Application Hacking Process (Step-by-Step)

Step	Description	Example
1. Information Gathering (Footprinting)	Collect publicly available information about the target.	Discovering subdomains, server type, and directory listings.
2. Scanning	Detect technologies, open ports, and directories.	Identifying Apache server running PHP.
3. Vulnerability Detection	Locate potential security weaknesses.	Detecting **SQL Injection** or **Insecure File Upload** flaws.
4. Exploitation	Actively use techniques to verify vulnerabilities.	Extracting data using SQL injection.
5. Post Exploitation	Gain deeper access after initial compromise.	Uploading a **web shell** for persistent access.
6. Reporting	Document vulnerabilities, proof-of-concept, and fixes.	Providing remediation steps to the development team.

How Vulnerabilities Work (Example: SQL Injection)

SQL Injection Example

If a web application directly uses user inputs in SQL queries without validation, it becomes vulnerable.

User Input

' OR '1'='1

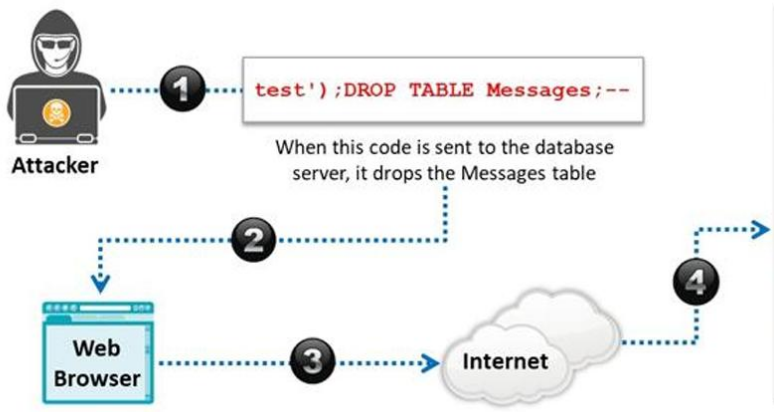
Vulnerable SQL Query

SELECT * FROM users WHERE username = ' ' OR '1'='1';

This condition ('1'='1') always evaluates to **true**, which allows attackers to bypass login authentication and gain unauthorized access.

✓ Key Takeaway

Web application hacking helps uncover weaknesses before attackers exploit them, ensuring better security and safer applications.



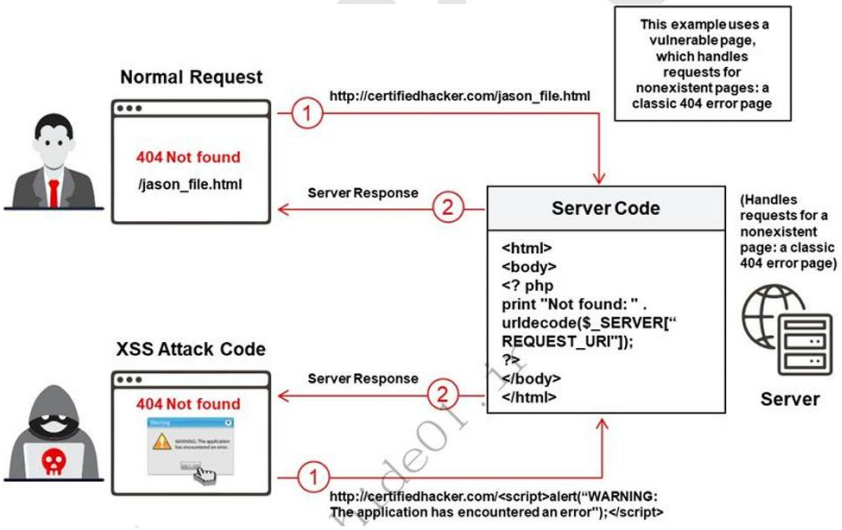

```
01 <?php
02 function save_email($user, $message)
03 {
04   $sql = "INSERT INTO Messages (
05             user, message
06         ) VALUES  (
07           '$user', '$message'
08         )";
09   return mysql_query($sql);
10 }
11 ?>
```

SQL Injection vulnerable server code

- **XSS Example**

Input Field: <script>alert('XSS')</script>

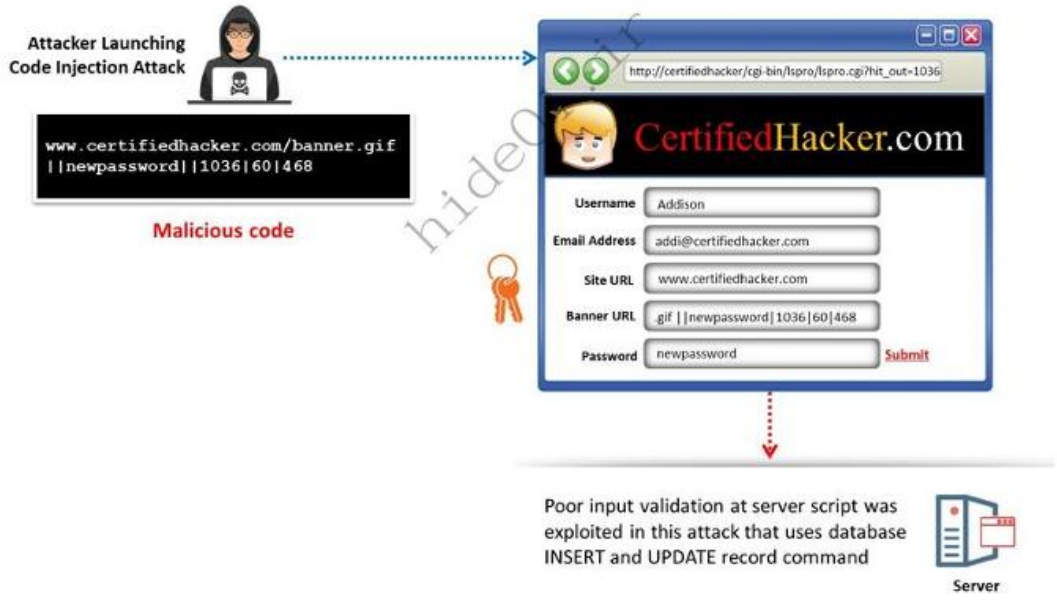
When submitted, this script will execute in another user's browser if input is not sanitized.



- **Command Injection Example**

Input Field: test; ls -la

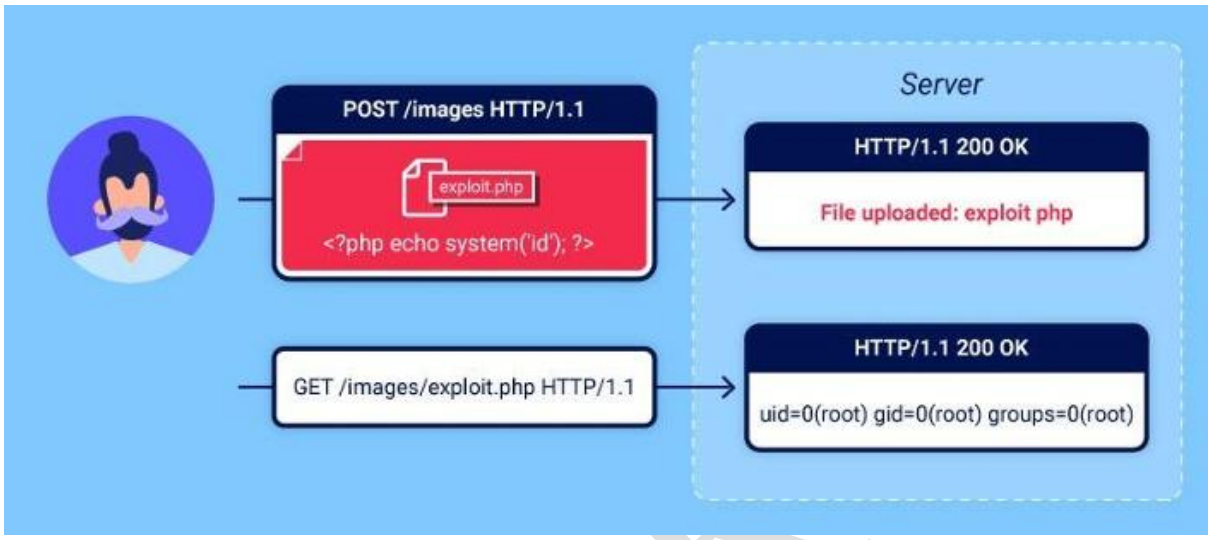
If the input is directly passed to the system shell, it will list all files.



- **File Upload Vulnerability Example**

Attacker uploads: shell.php.jpg (disguised)

If the server doesn't validate the file type properly, the attacker can execute the file.



HTTP Methods (Essential for Hacking)

• GET: Request data from a server (visible in URL)

• POST: Send data to a server (not visible in URL)

• PUT: Update a resource

• DELETE: Remove a resource

• OPTIONS: List allowed HTTP methods

WASP Top 10 (Latest Web Application Security Risks)

The OWASP Top 10 outlines the most critical security risks in web applications.

Category	Description	Example
A01 – Broken Access Control	Access restrictions are not enforced properly.	Changing a user ID in the URL to view someone else's data.
A02 – Cryptographic Failures	Sensitive data is not protected.	Passwords transmitted using **HTTP** instead of **HTTPS**.
A03 – Injection	Untrusted data is used in commands or queries.	Running SQL commands through user input (**SQL Injection**).
A04 – Insecure Design	Application security is not considered from the start.	Missing validation checks during architecture design.
A05 – Security Misconfiguration	Incorrect or insecure settings.	Default admin accounts still active, detailed error messages exposed.
A06 – Vulnerable Components	Use of outdated or weak components.	Old JavaScript libraries with known vulnerabilities.
A07 – Identification & Authentication Failures	Weak authentication mechanisms.	Weak passwords allowed, no account lockout after multiple attempts.

Category	Description	Example
A08 – Software & Data Integrity Failures	Untrusted data or code is used without verification.	Application accepts unsigned or unknown updates.
A09 – Security Logging & Monitoring Failures	Insufficient logging and alerting.	Attacks occur without detection because logs are disabled.
A10 – Server-Side Request Forgery (SSRF)	Server is tricked into accessing internal resources.	Attacker forces the server to access internal-only endpoints.

Key Web Hacking Concepts

These concepts are fundamental for understanding how modern web attacks work and how to defend against them:

- **Cookies & Sessions:**
  Track user authentication; attackers target session tokens to hijack accounts.
- **CSRF Tokens (Cross-Site Request Forgery Protection):**
  Unique security tokens used to prevent unauthorized actions performed on behalf of a user.
- **Input Validation:**
  Ensures only safe and expected data is processed to prevent attacks like **XSS** and **SQL injection**.
- **Encoding:**
  Converts user input into safe formats (HTML encoding, URL encoding) to neutralize malicious payloads.
- **Authentication Mechanisms:**
  Methods for verifying identity such as passwords, access tokens, and multi-factor authentication (MFA).

Footprinting the Web Infrastructure

Attacker Machine-: Kali linux

Target Web application-: http://testfire.net/

1. Footprinting Using Whois

WHOIS Website is a public online database that provides detailed information about who owns a domain name or IP address.

OR

Footprinting using WHOIS lookup means gathering information about a target domain or IP address by querying the WHOIS database, which stores ownership and administrative details about registered domains.

How to use it -:

- Open Browser and search whois and click on official whois website.

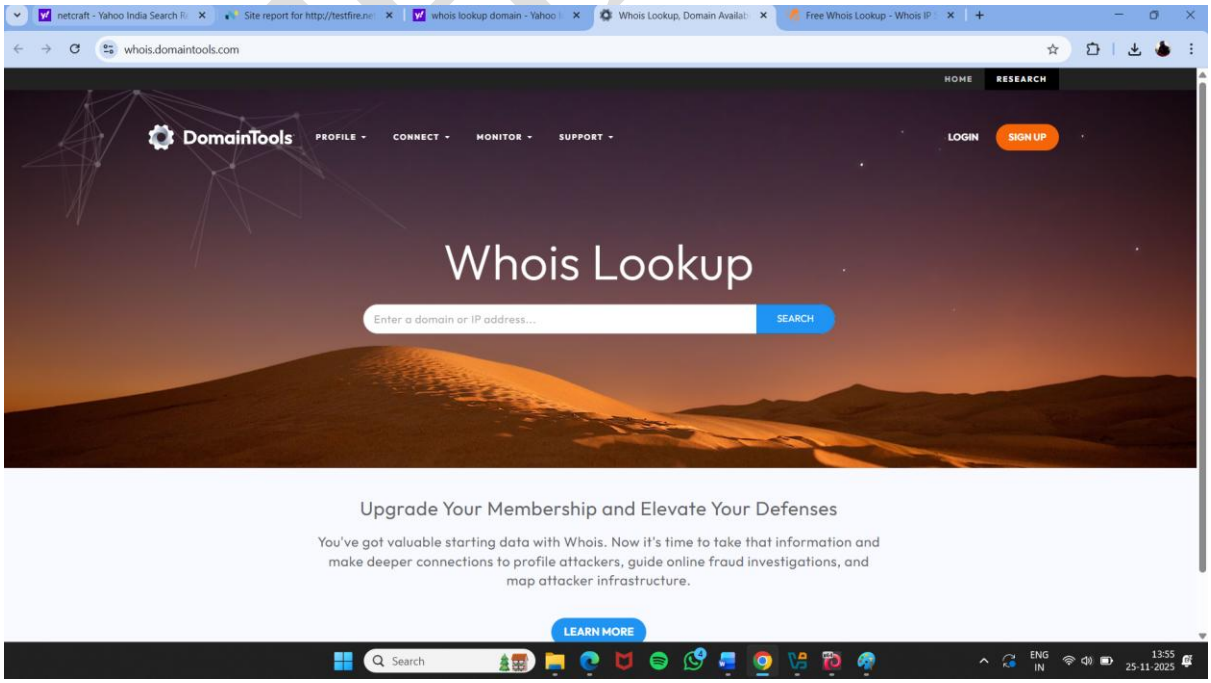


Figure 1

- Enter target website URL & click on search
  - ➢ Here it finds some Name server and other information
  - ➢ Registrar Information and Registrant Contact

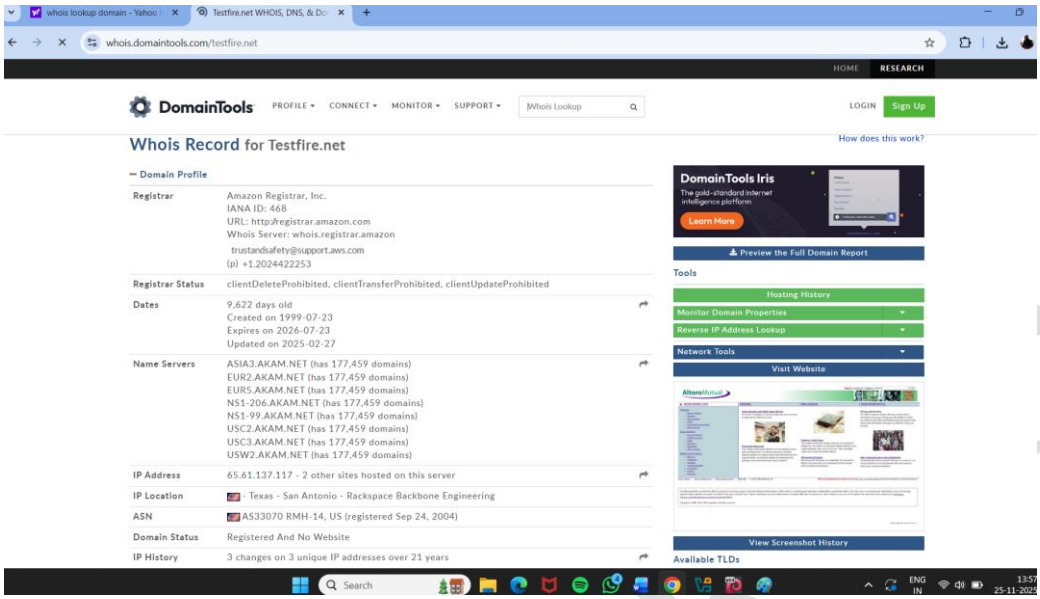


Figure 2

2. Footprinting Using DNS Lookup Website

A DNS Lookup Website is an online tool that helps you find the IP address, DNS records, and server details of a domain name.

- Enter Target Website & click on find.

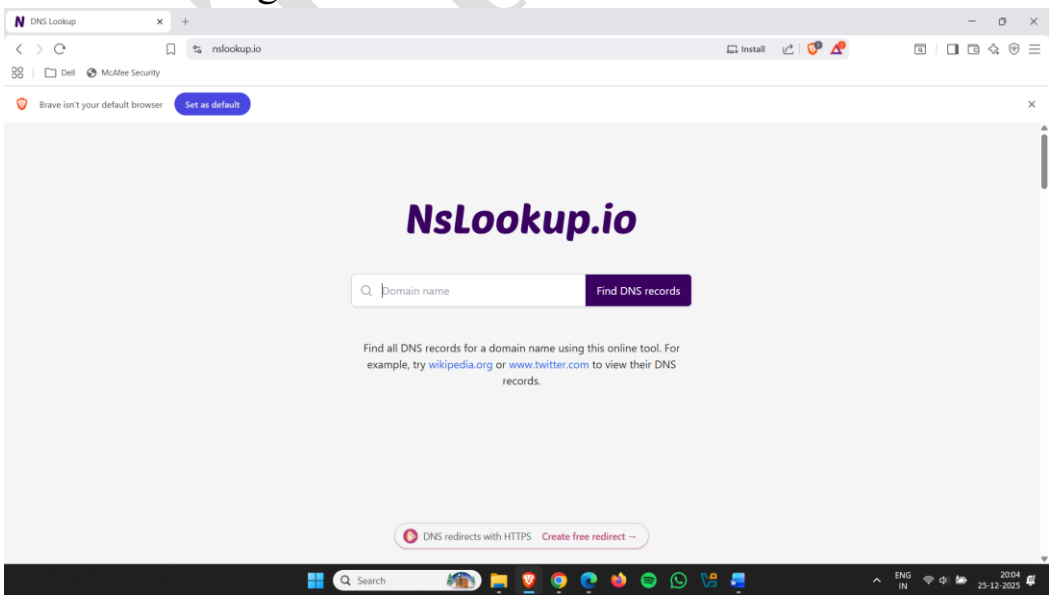


Figure 3

- Below is the Result.

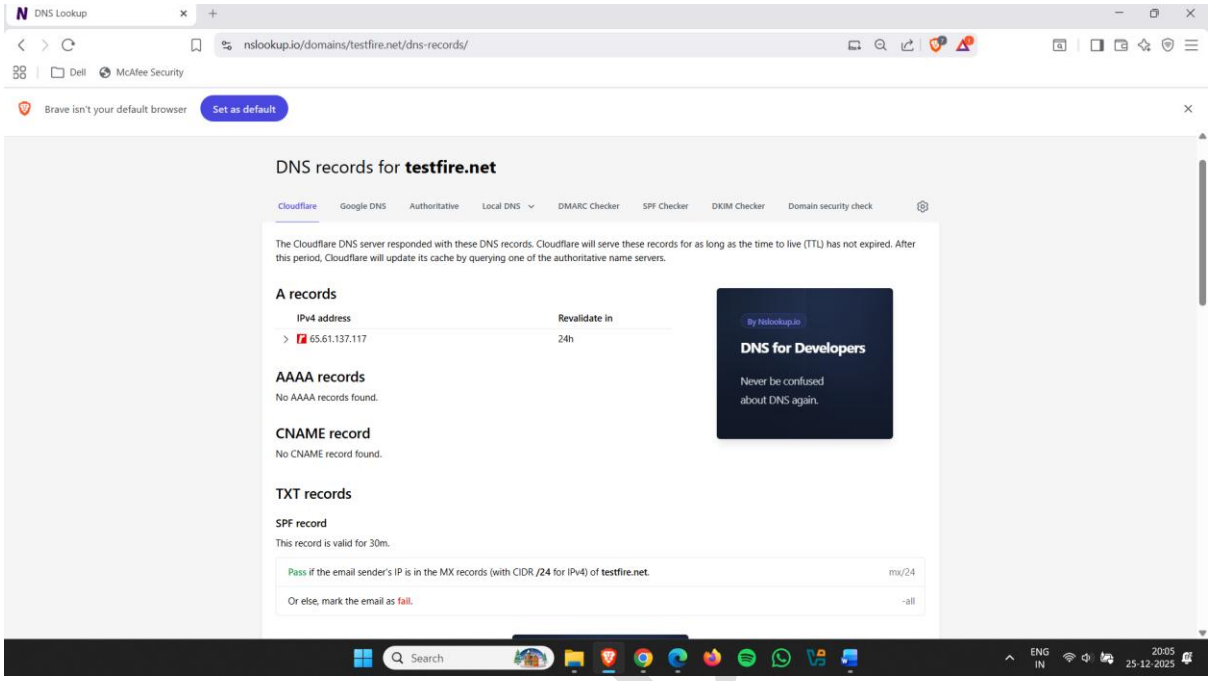


Figure 4

- Below are the Ns Records

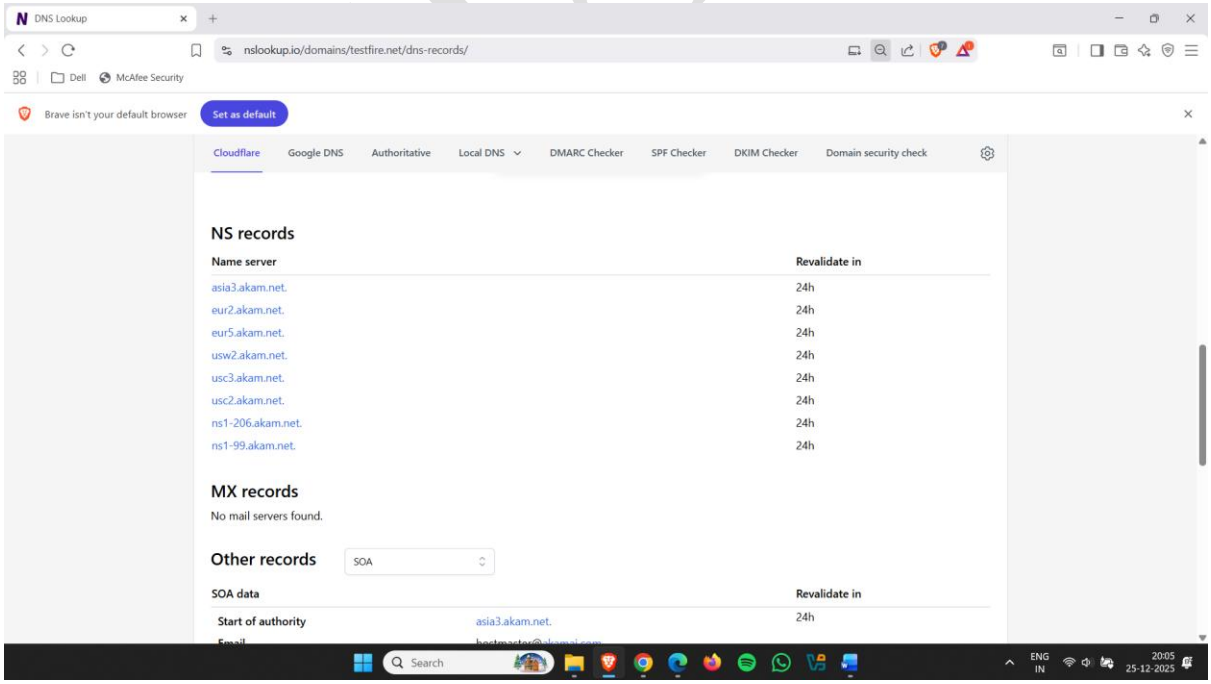


Figure 5

3. Footprinting Using NetCraft

Netcraft is a well-known cybersecurity and internet data-collection service used to analyze websites, detect phishing, and gather hosting/domain information. It is widely used by security researchers, penetration testers (for defensive purposes), and companies to protect against online threats.

What Netcraft Provides:

Feature
Site Report
Phishing Detection
Anti-Phishing Toolbar
Uptime Monitoring
Web Hosting Survey
Takedown Service
Browser Extension

- Go to https://www.netcraft.com.

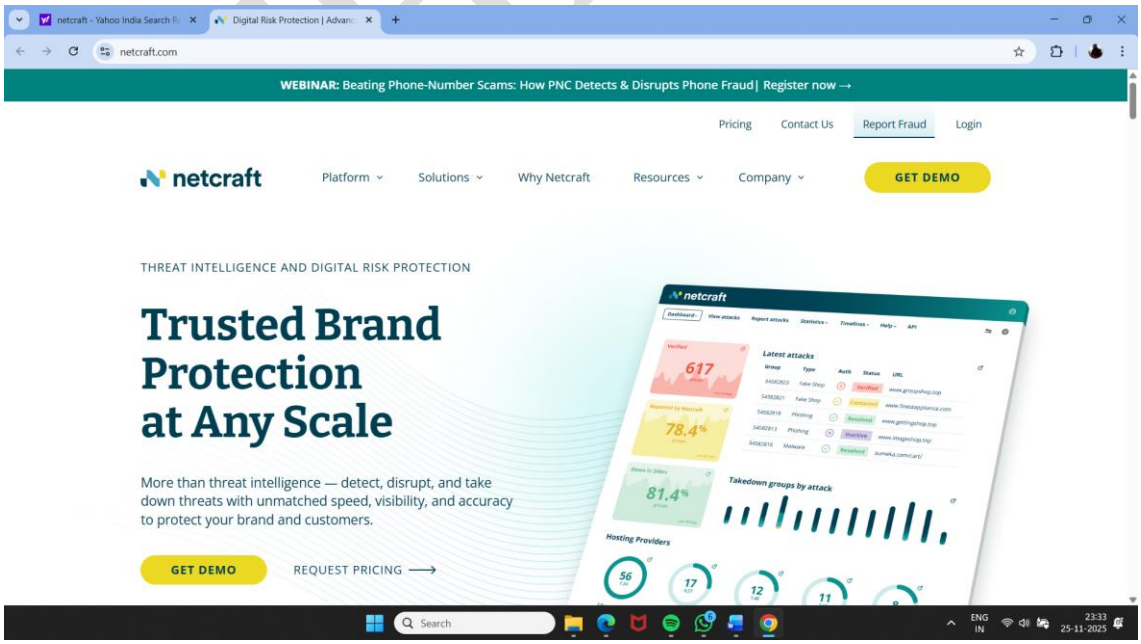


Figure 6

- Click on Resources and click on research tools.

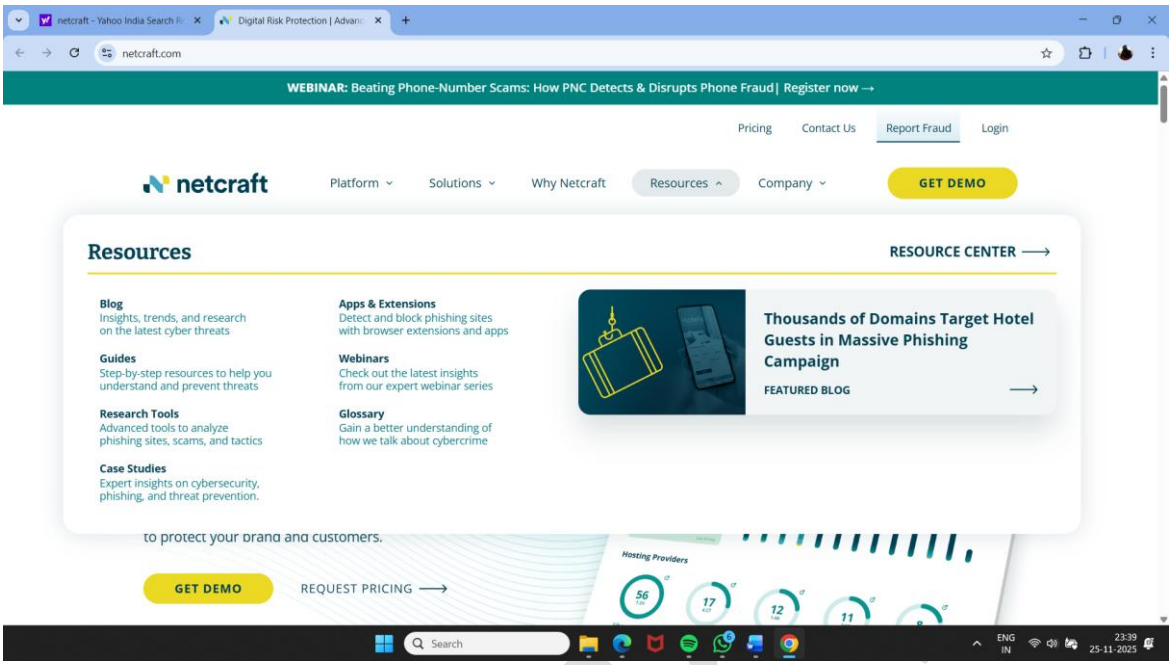


Figure 7

- Click on site report.

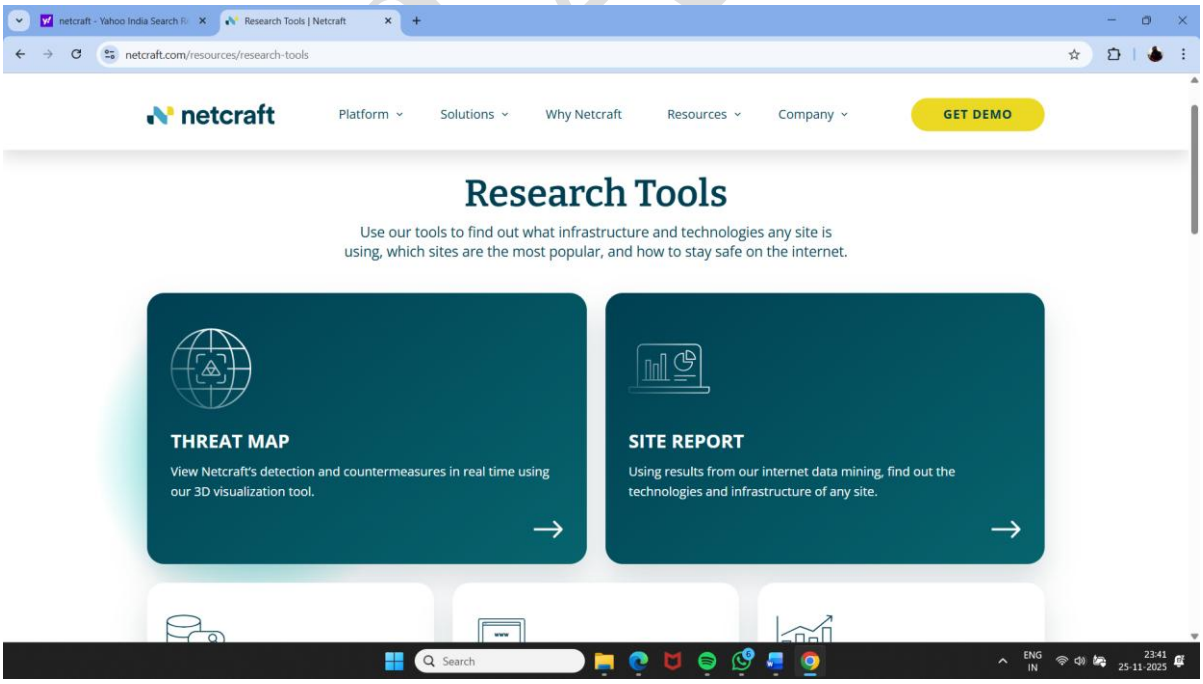


Figure 8

- Provide Domain name & click on lookup.

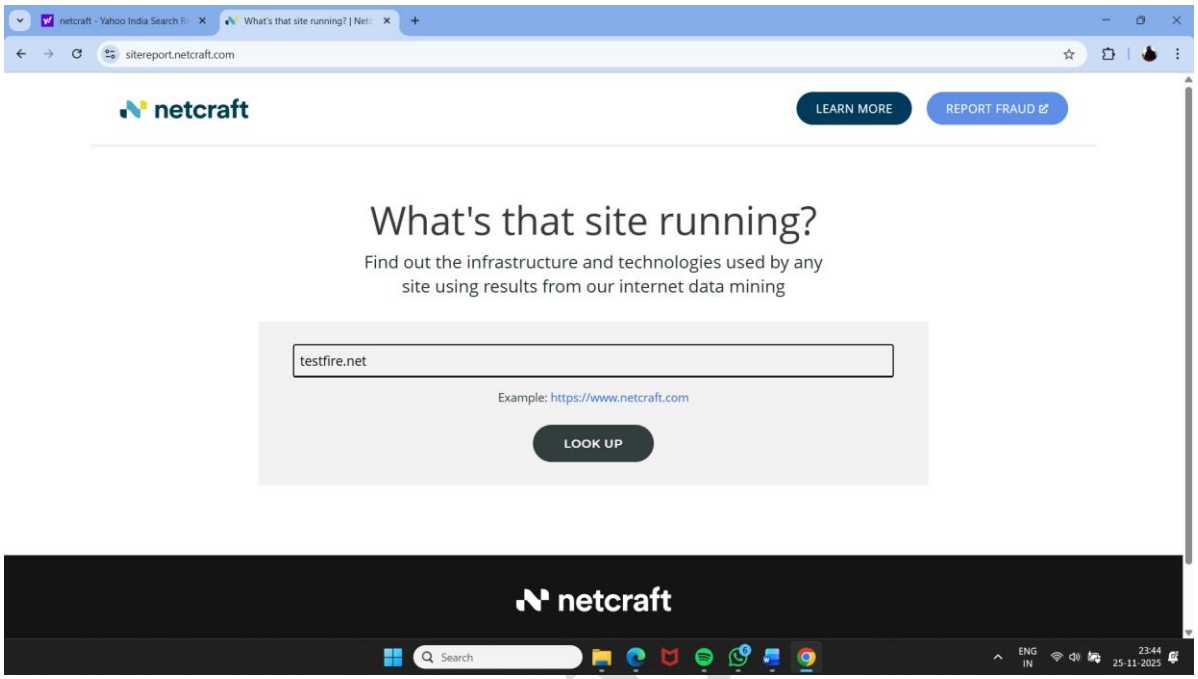


Figure 9

- It will give you full site report.

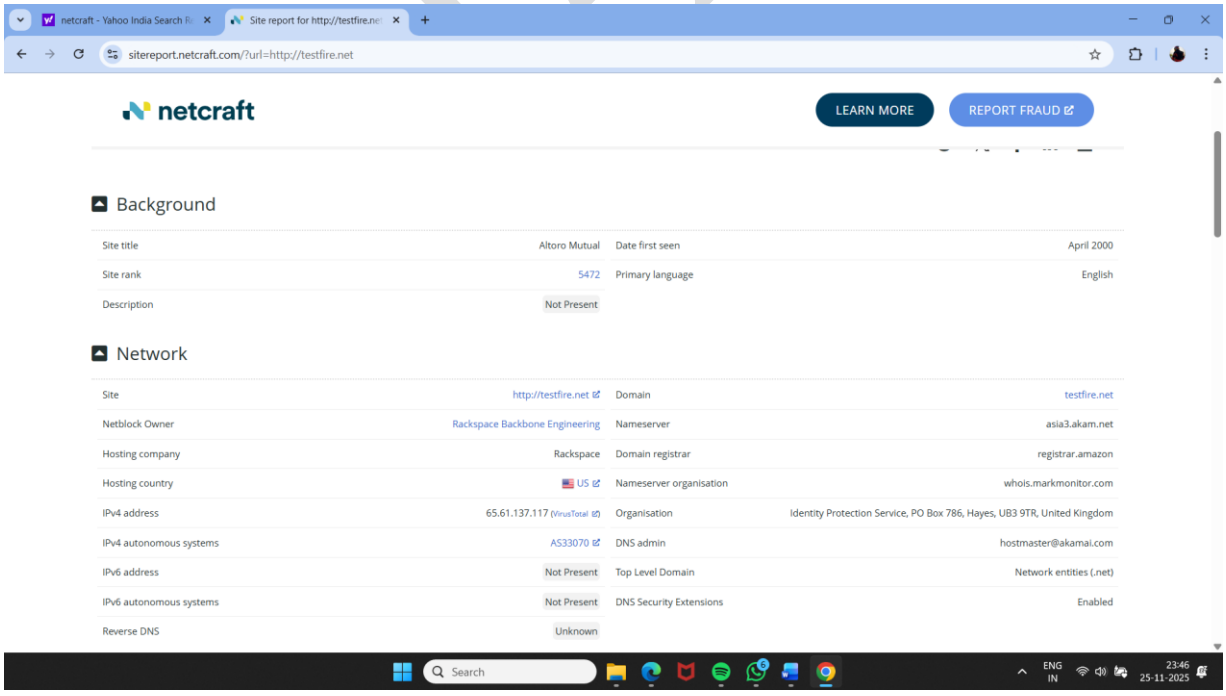


Figure 10

Scanning the Web Infrastructure

Attacker Machine-: Kali linux

Target Web application-: http://testfire.net/

Scanning is the process of actively identifying live hosts, open ports, running services, and potential vulnerabilities in a target system or network.

How to do it -:

- Open kali linux terminal and type below command.
- You will get open ports

1. **Command -: nmap -v -sS -sV<Target Website>**

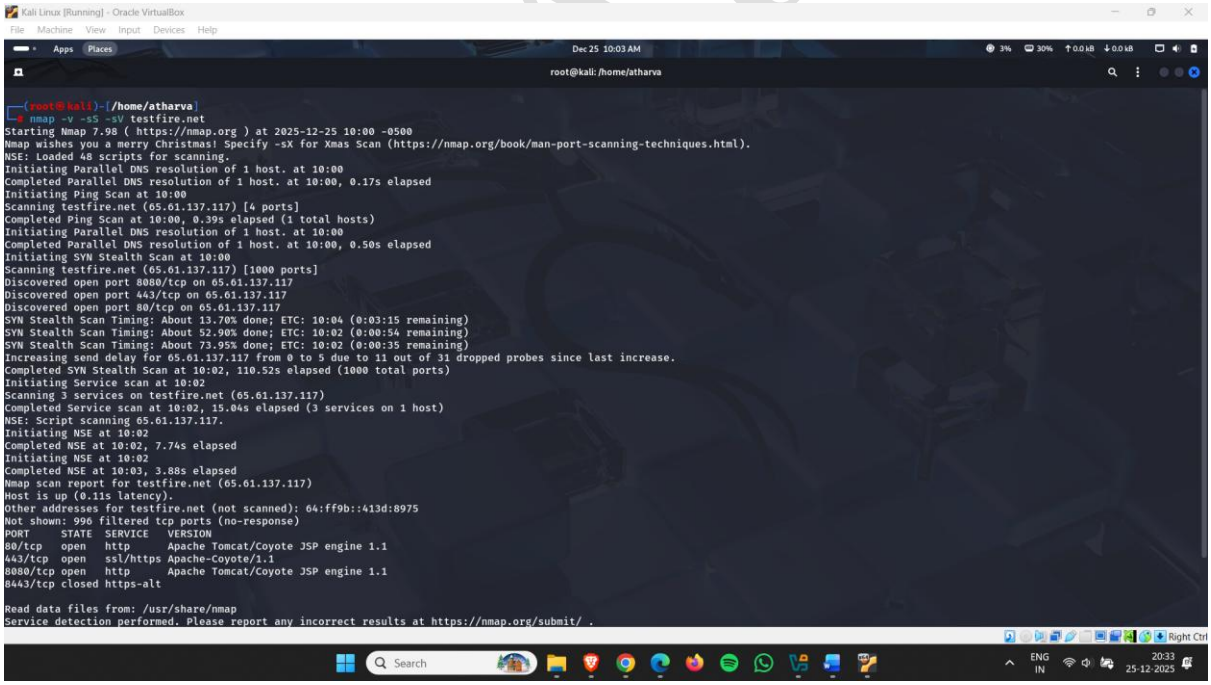


Figure 11

2. Command -: nmap -v --sS -p80 –script=http-enum

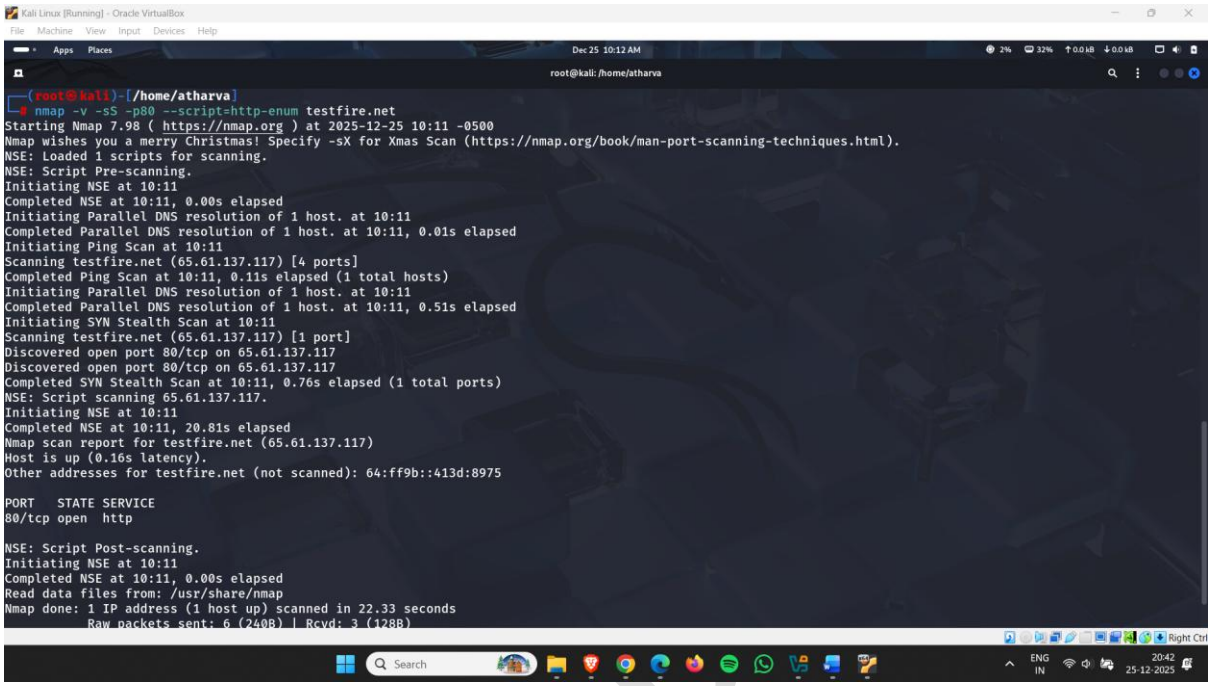


Figure 12

3. Command-: nmap -T4 -A -v testfire.net

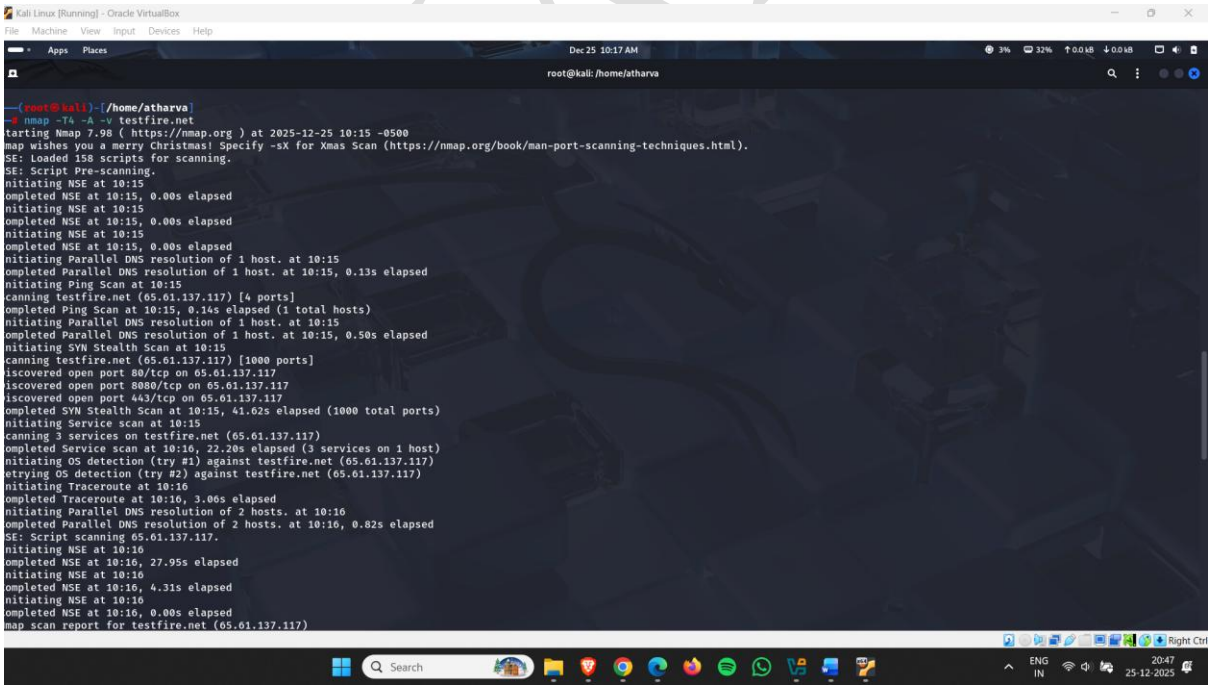


Figure 13

Web Infrastructure Vulnerability Assessment

A Web Infrastructure Vulnerability Assessment is a systematic process of identifying, analyzing, and prioritizing security weaknesses across all components that support a web application or website. This includes servers, databases, network devices, DNS configurations, SSL/TLS settings, web services, APIs, operating systems, and hosting environments. The objective of this assessment is to detect potential vulnerabilities—such as misconfigurations, outdated software, insecure protocols, weak authentication mechanisms, or exposed services—that could be exploited by attackers to compromise availability, confidentiality, or integrity of data..

1. **Vulnerability Analysis using nikto**
   Command -: nikto -h<target website>

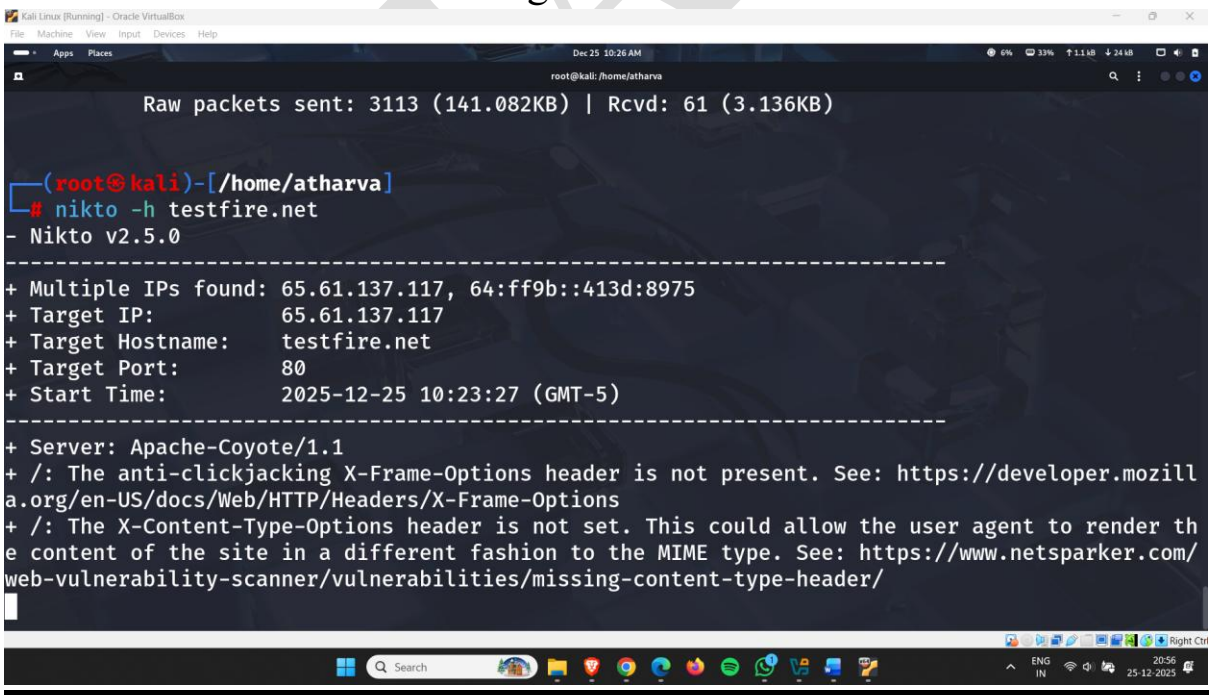


Figure 14

2. Vulnerability Analysis using Burp Suite

- Open kali linux terminal and start burp suite

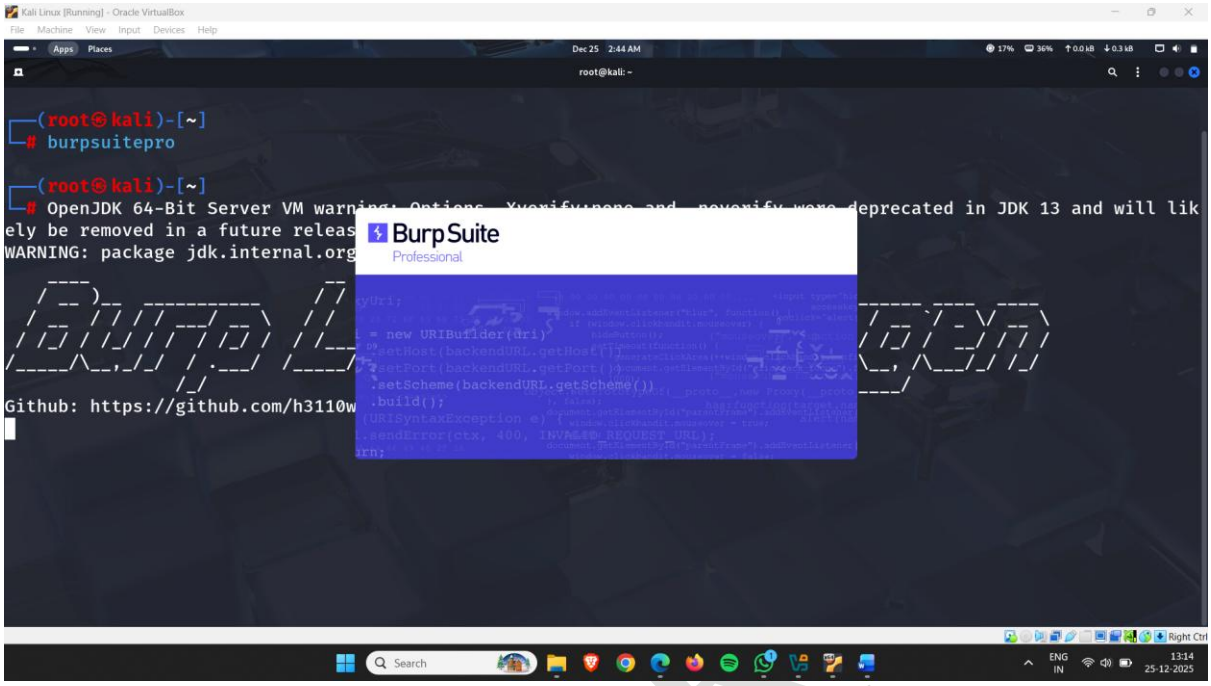


Figure 15

- Now,set a proxy.

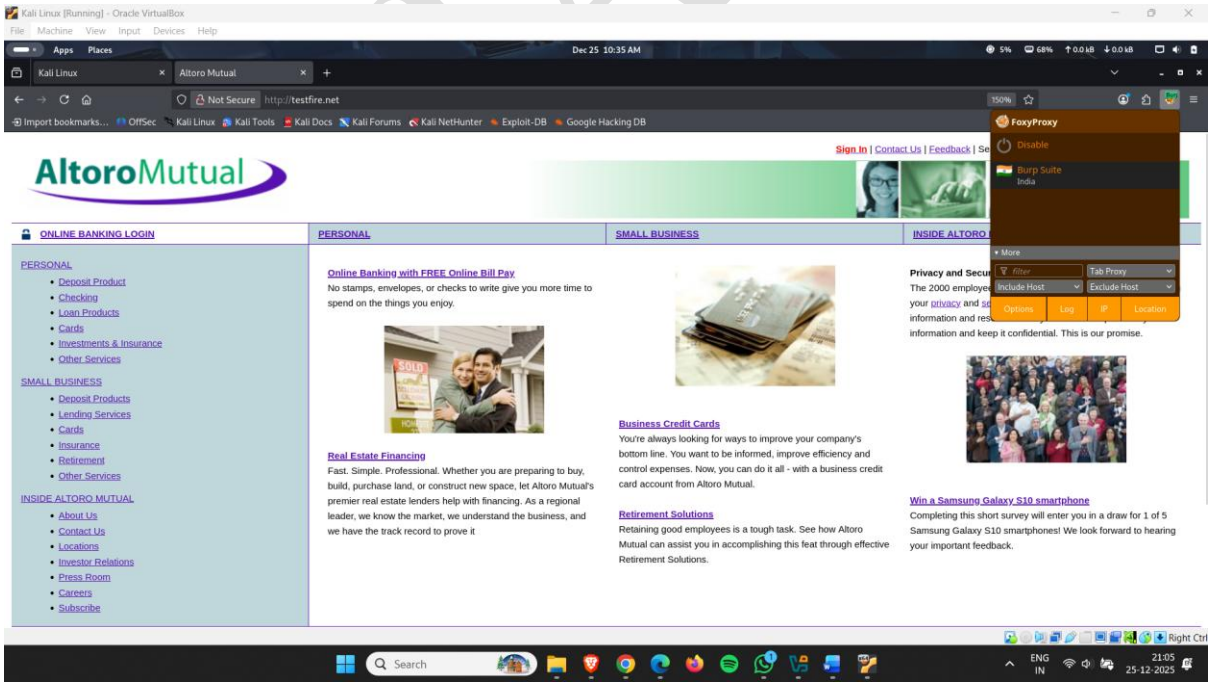


Figure 16

- Open Burpsuite and turn on interception
- After turning on interception refresh browser
- You will get a request

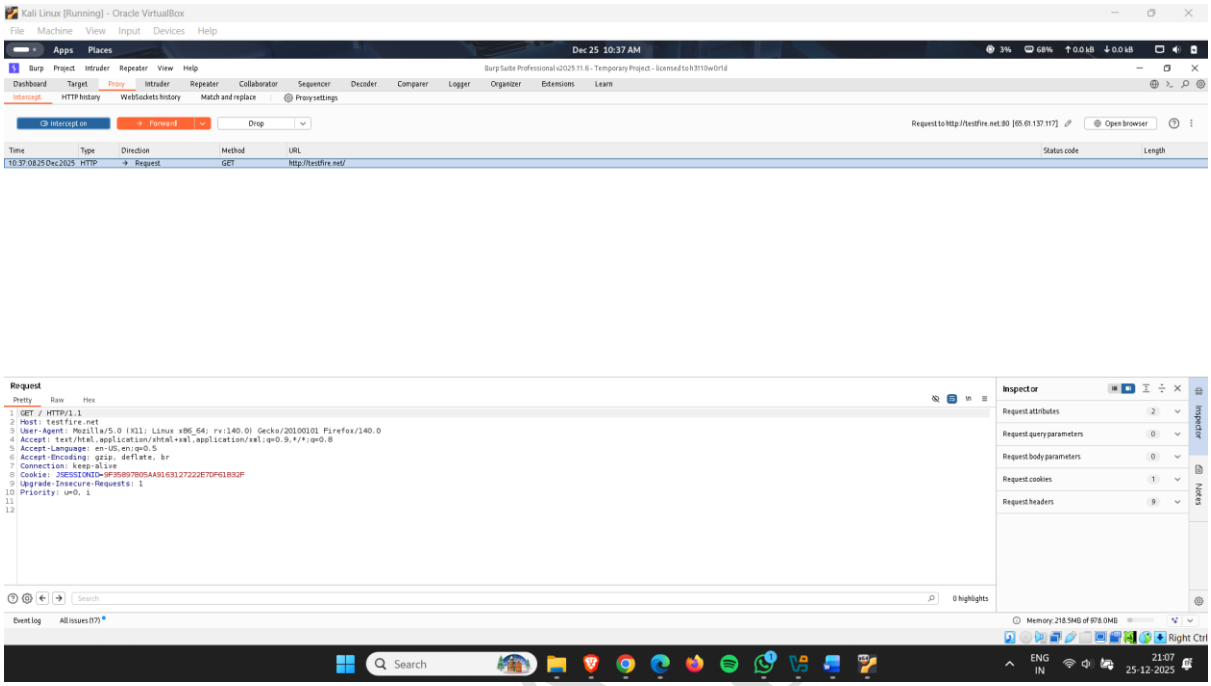


Figure 17

- Right click on scan & click on Scan option

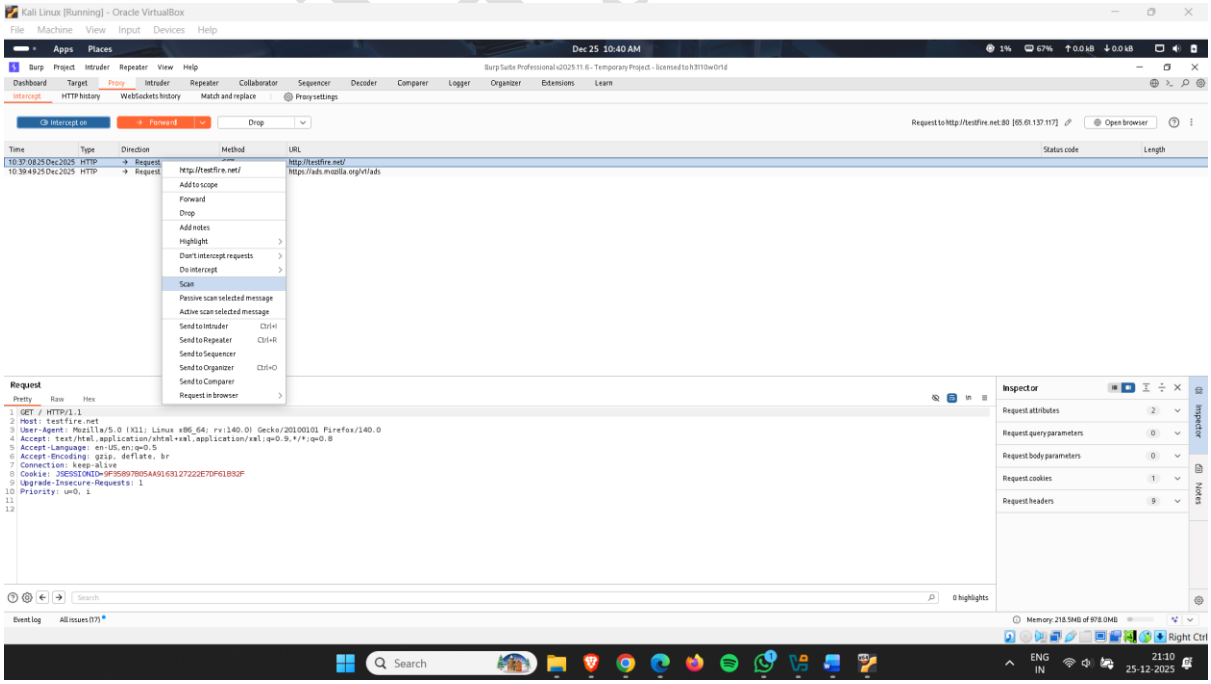


Figure 18

- After click on scan select the options, as per your requirement & click on next

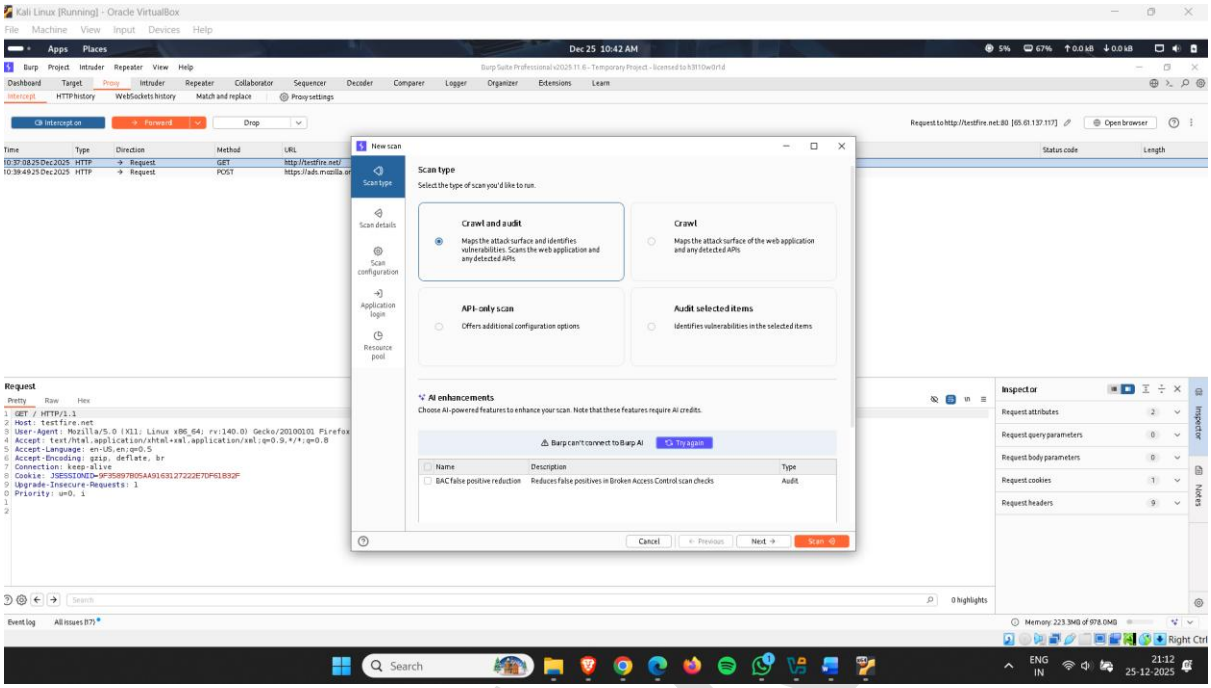


Figure 19

- After click on scan select the options, as per your requirement & click on next

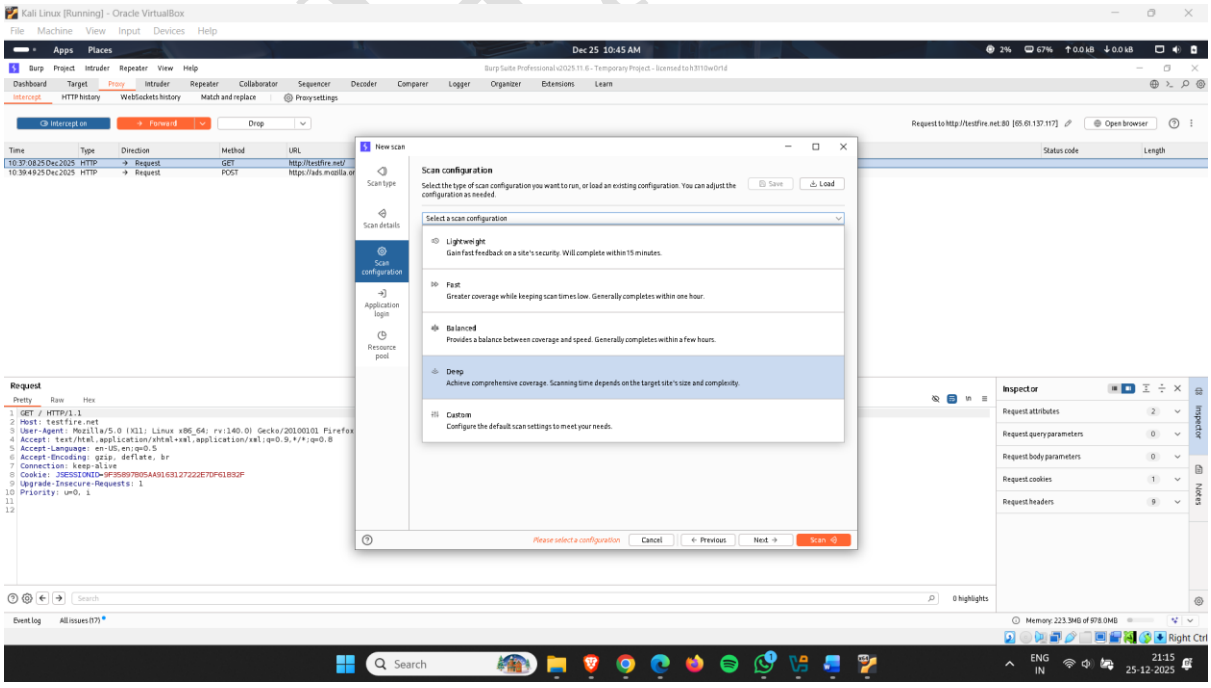


Figure 20

- Scanning started
- Finding pages in website

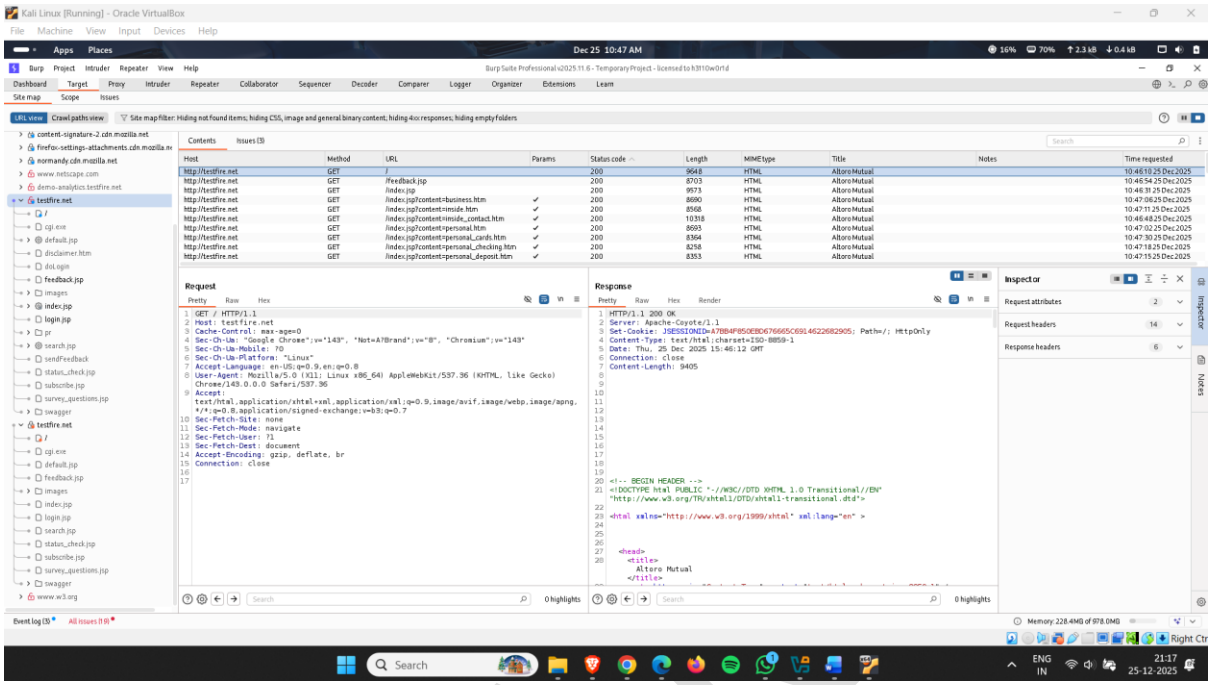


Figure 21

- Started Finding vulnerabilities

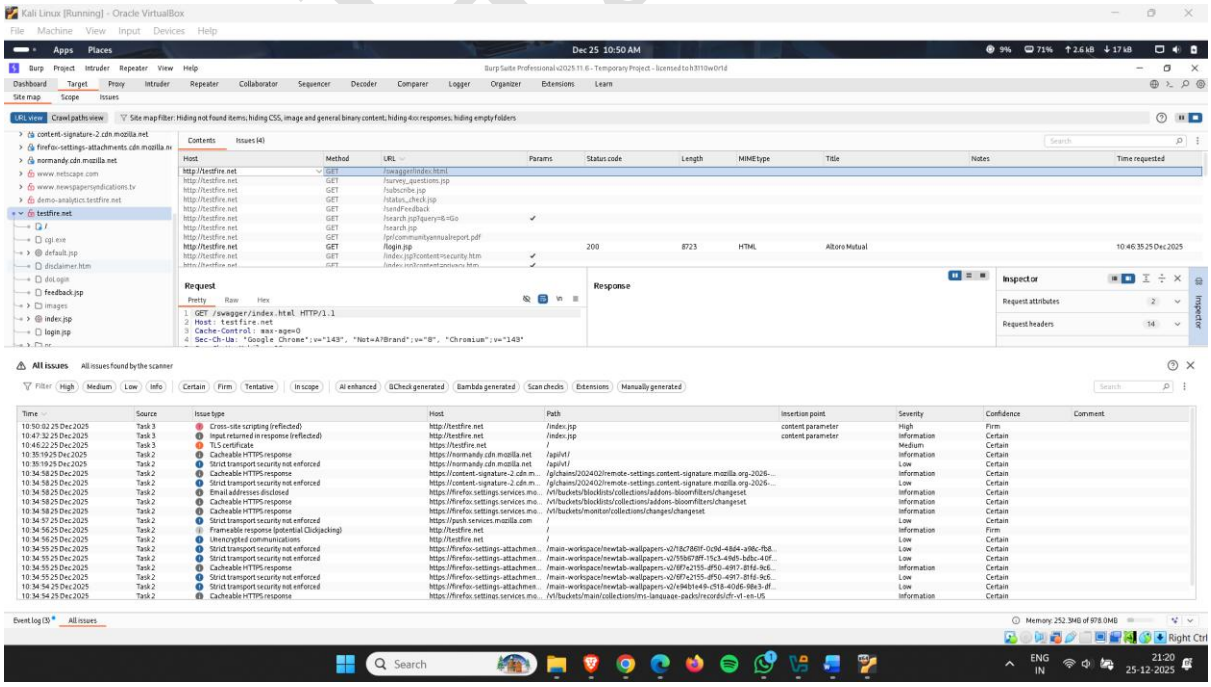


Figure 22

3. Vulnerability Analysis using Wapiti

How to use it -:

- Open kali linux terminal and type following command
- Command -: wapiti -u
- Id not installed type sudo apt install wapiti

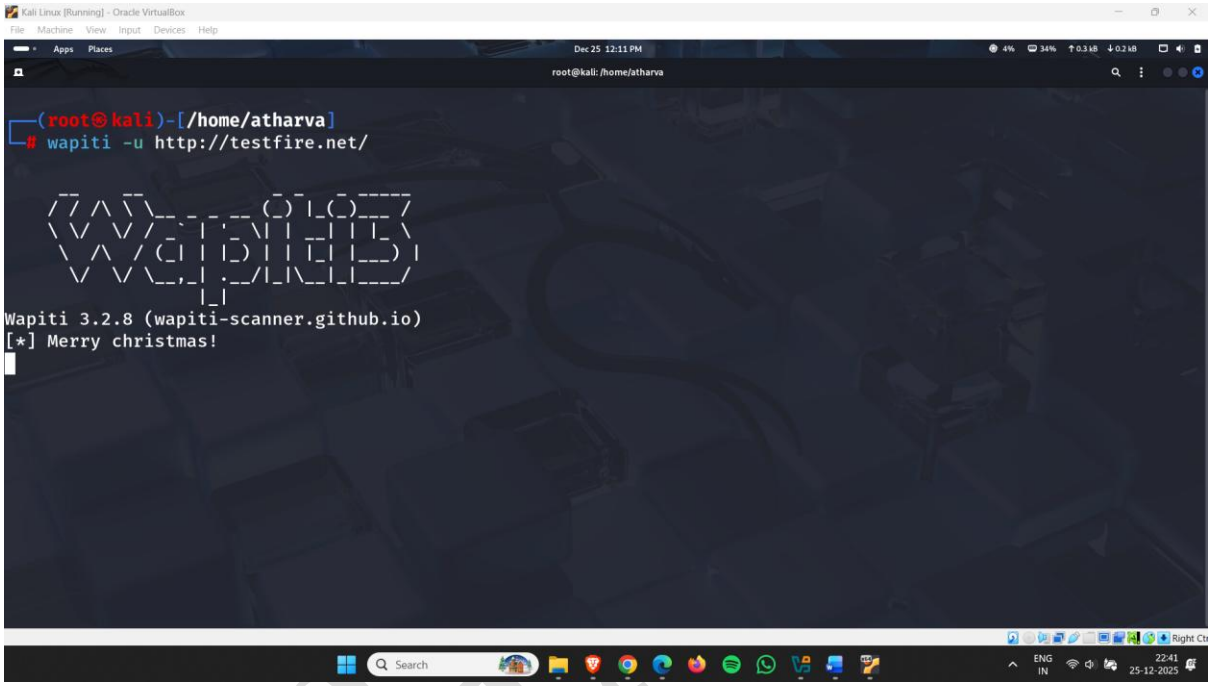


Figure 23

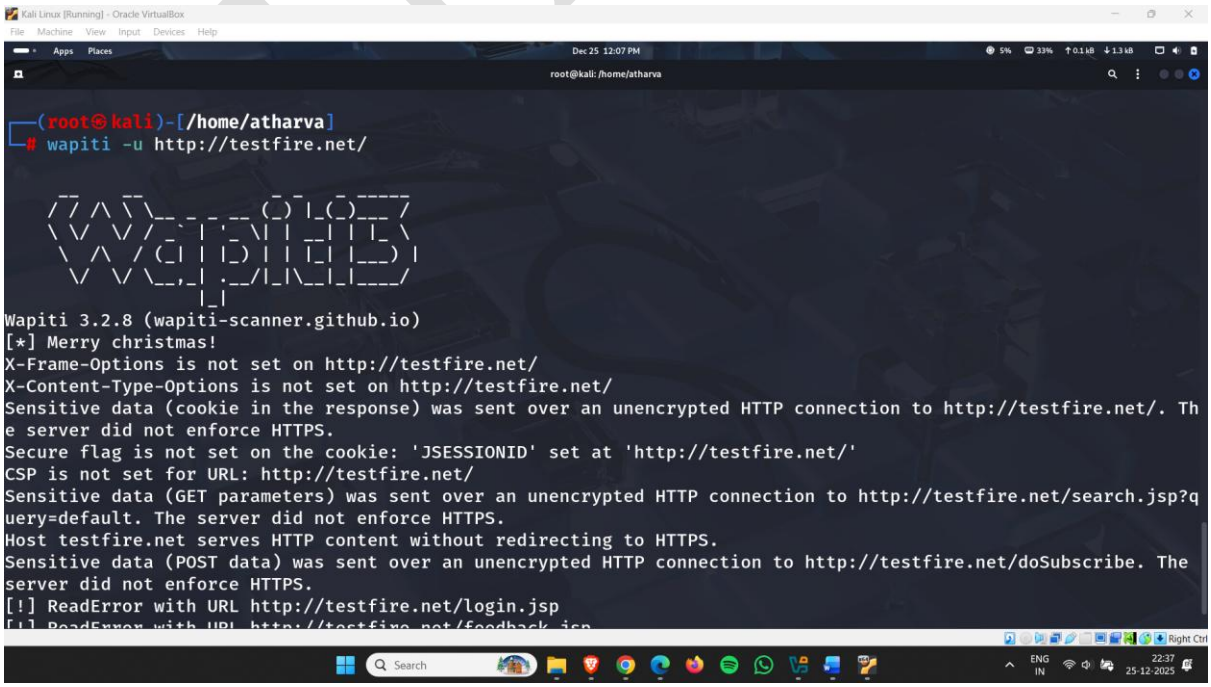


Figure 24

Brute force attack using Burp Suite

Burp Suite is a widely used and powerful web vulnerability scanner and penetration testing framework. It is commonly used by cybersecurity professionals, ethical hackers, and bug bounty hunters to identify security weaknesses in web applications.

Key Features of Burp Suite

1. Intercepting Proxy

- Captures, inspects, and modifies HTTP/S requests and responses between the browser and the web server.

2. Spider (Site Map / Crawler)

- Automatically maps the structure and content of a web application by crawling its pages and links.

3. Scanner (Professional Version)

- Performs automated scanning to detect common web vulnerabilities such as SQL Injection, Cross-Site Scripting (XSS), Server-Side Request Forgery (SSRF), and more.

4. Intruder

- Executes automated customized attacks, including brute-force attacks and fuzzing, to test input validation and authentication mechanisms.

5. Repeater

- Allows manual editing and re-sending of individual HTTP requests to analyze application behavior and test specific inputs.

6. Sequencer

- Evaluates the randomness and unpredictability of tokens, such as session cookies, to detect weak session management.

7. Decoder

- Encodes and decodes data formats (e.g., Base64, URL encoding, HTML encoding) to assist in analysing or crafting payloads.

8. Comparer

- Compares two pieces of data (requests, responses, or strings) to highlight differences useful for debugging or testing.

🧩 Burp Suite Editions

Edition	Price	Features
Community	Free	Manual testing, Proxy, Repeater, Decoder
Professional	Paid	Automated scanner, Intruder speed, Collaborator, BApps
Enterprise	Paid	Large-scale automated scanning for organizations

How to do it:

- Open kali linux terminal and start burp suite professional

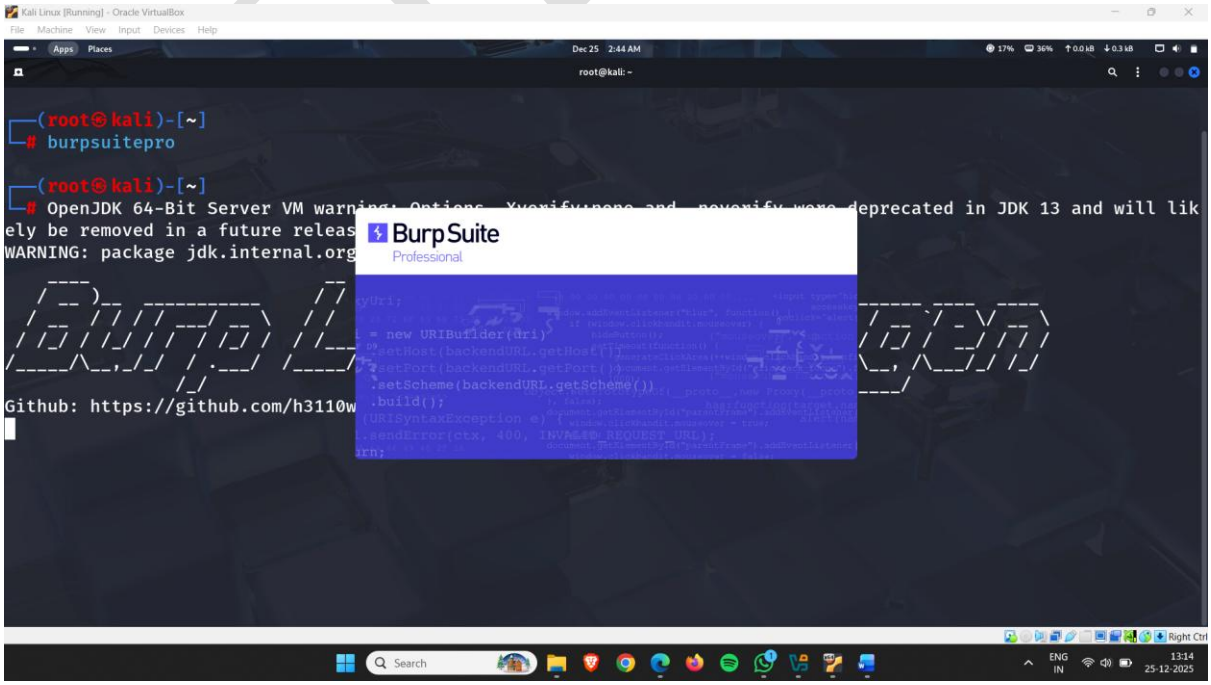


Figure 25

Target Website

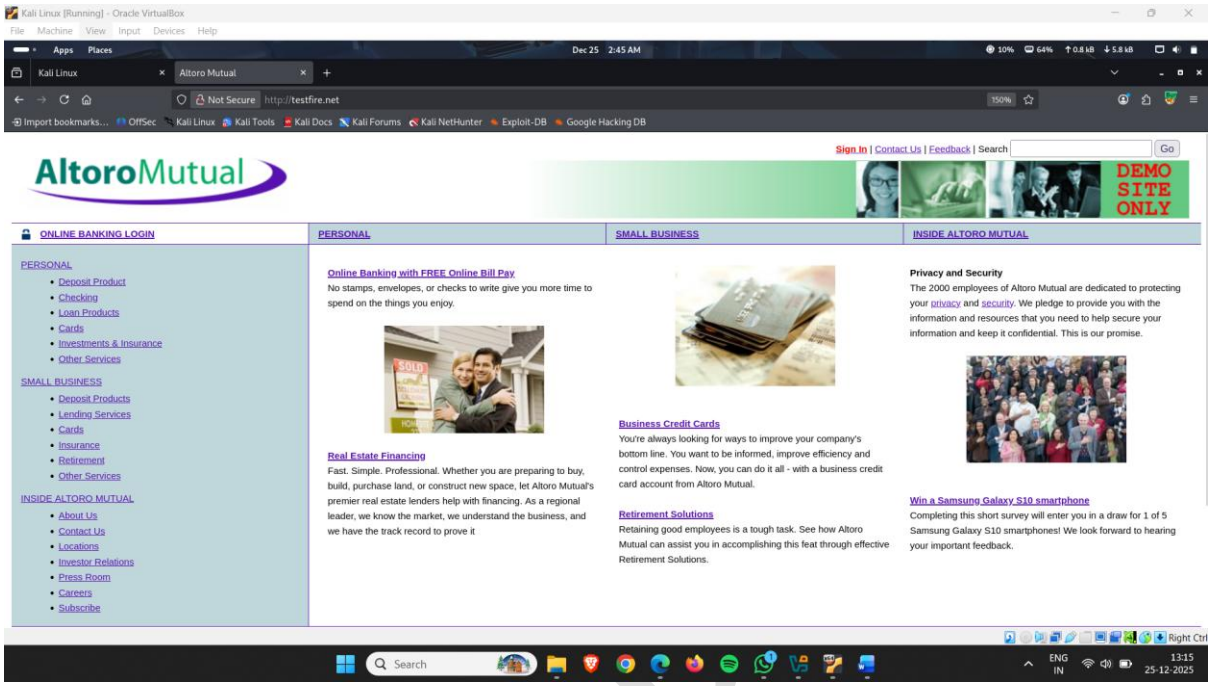


Figure 26

- Open Target Website Login page

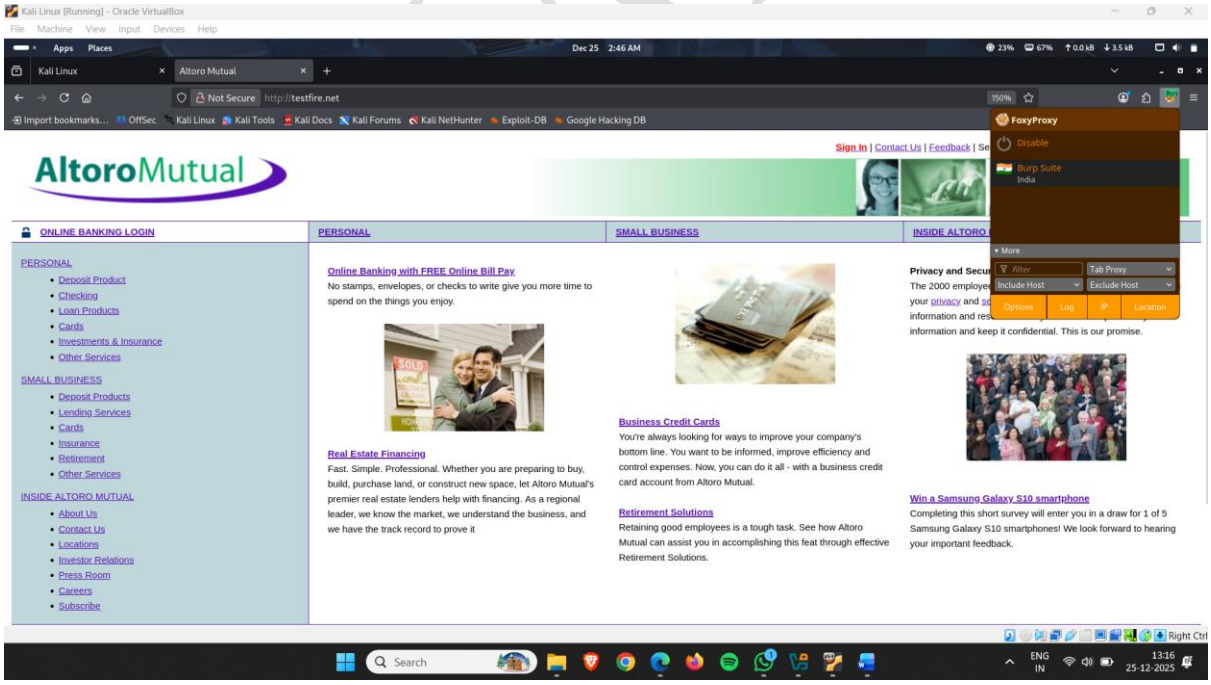


Figure 27

- Set up proxy (Foxy Proxy extension)

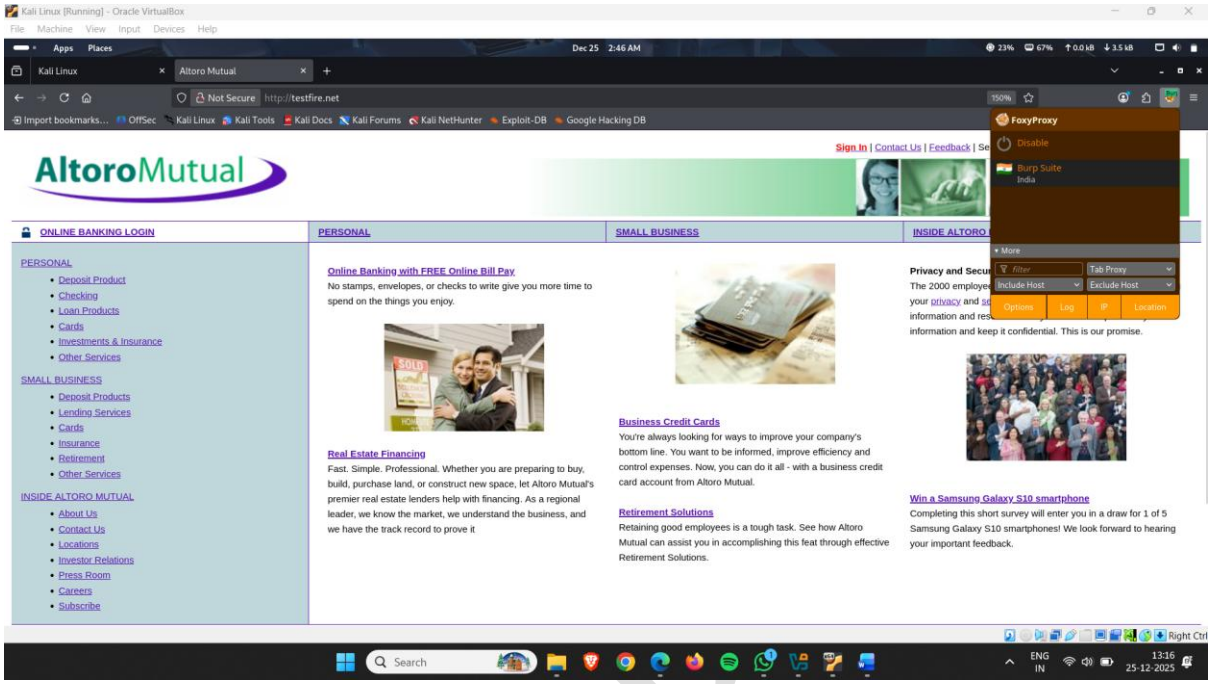


Figure 28

- Start interception in burp suite for intercepting the request

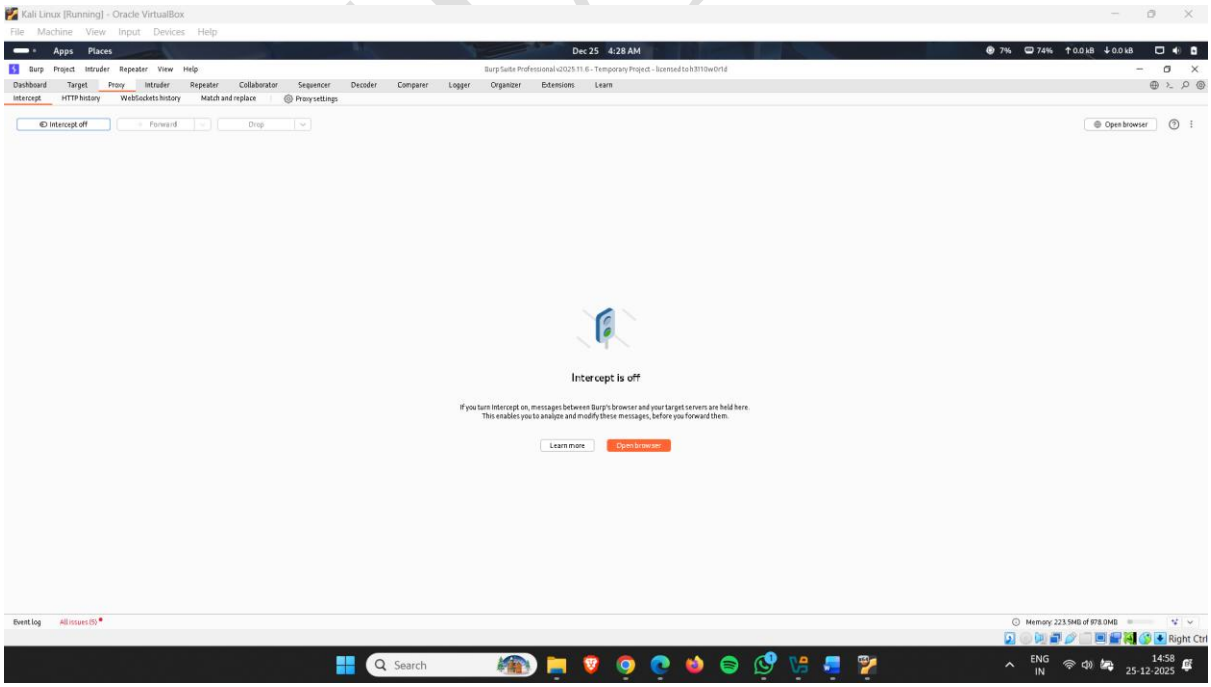


Figure 29

- Enter credentials

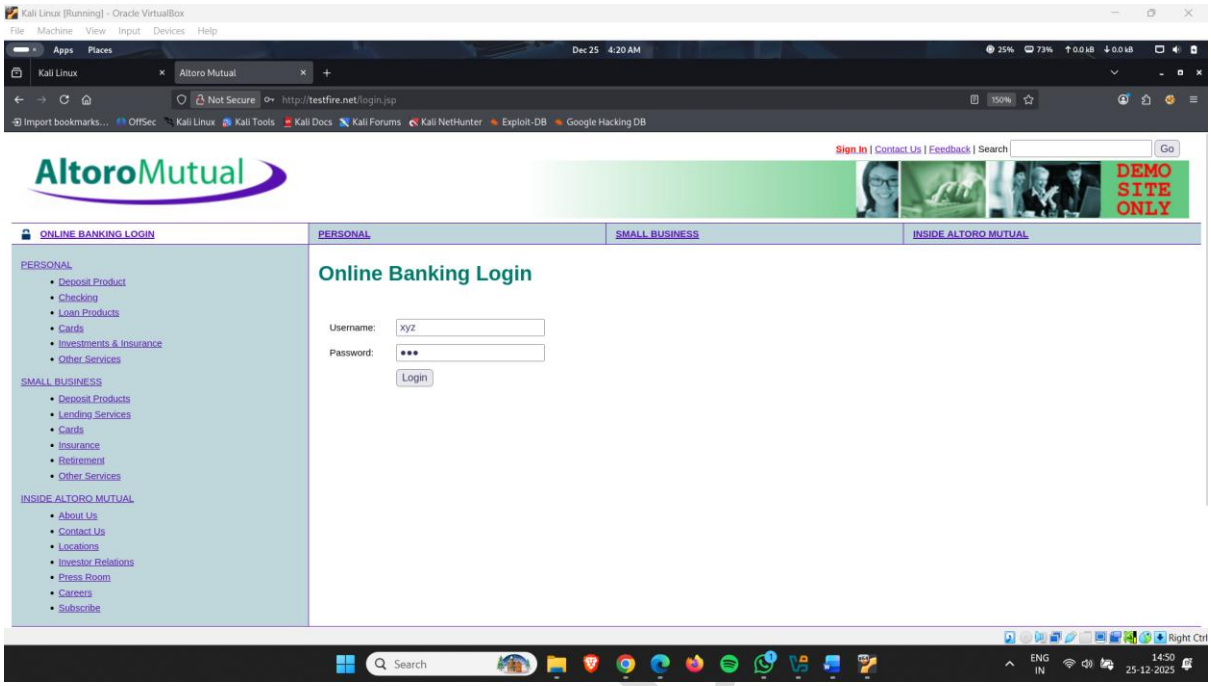


Figure 30

- Login Request intercept
- Now, right click on request and send it to the intruder

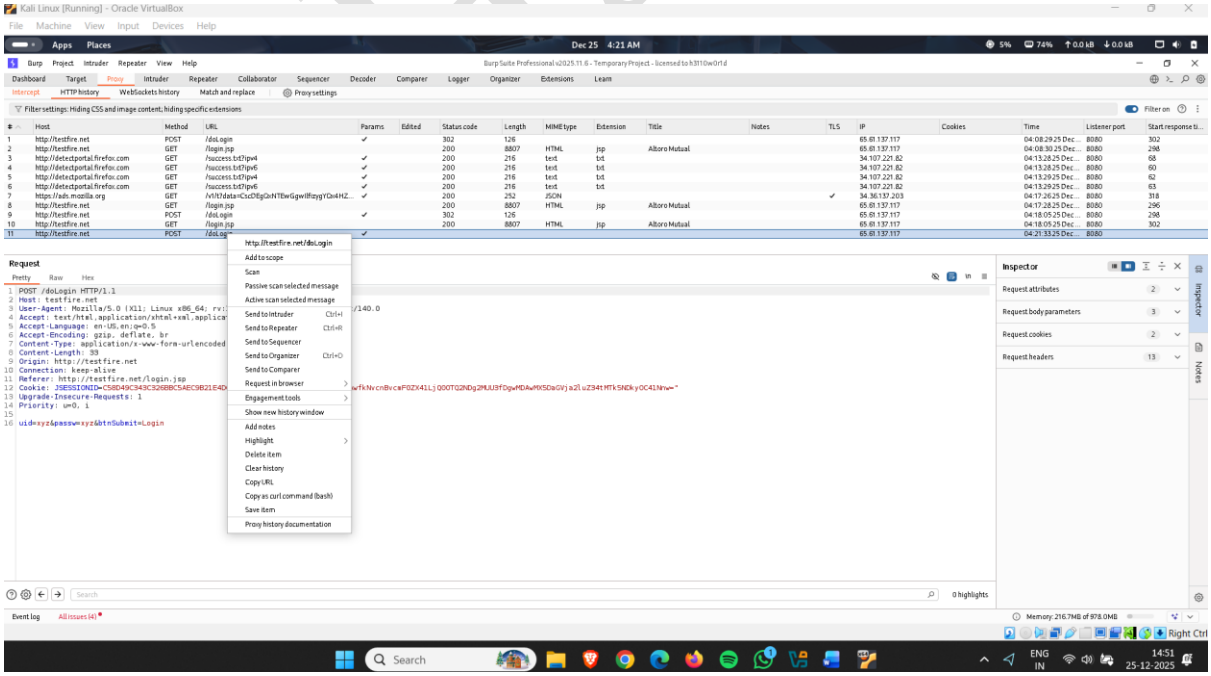


Figure 31

- Open Intruder interface

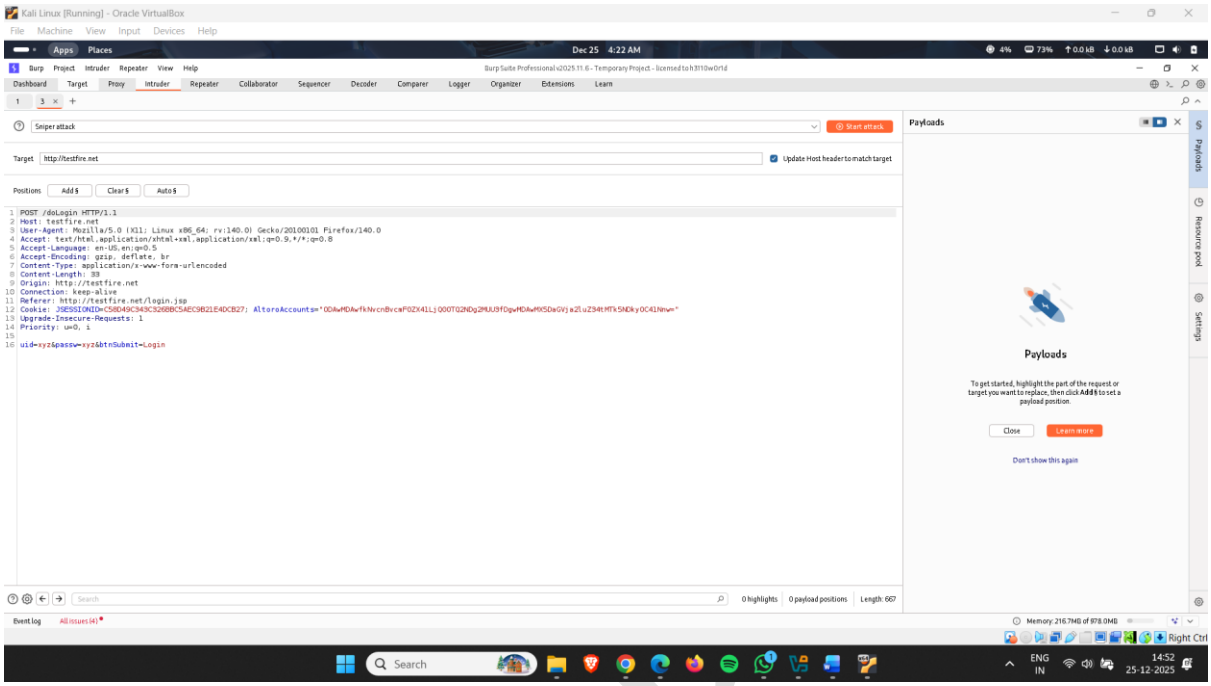


Figure 32

- Click on sniper attack a dropdown list appears
- Select cluster bomb attack

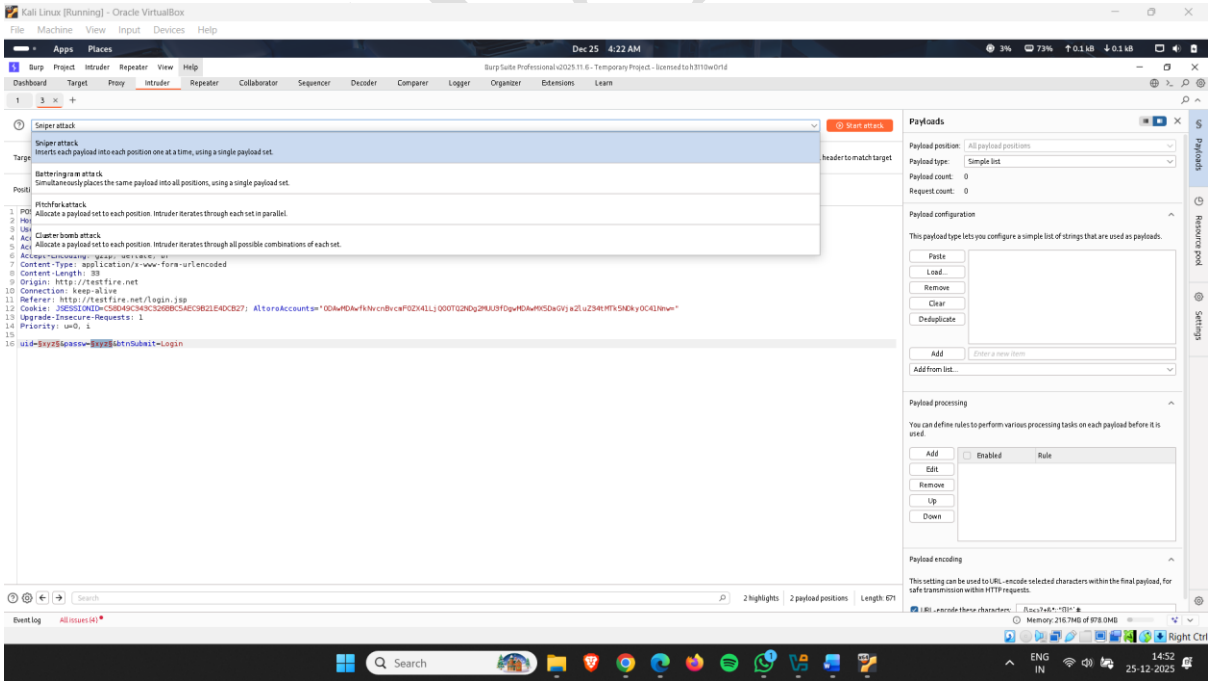


Figure 33

- Select the username section and click on Add$
- Select the Password section and click on Add$
- In payload position, select position (1) i.e. username
    - ➢ Provide a username in payload configuration section
- Now select position second, i.e. password
    - ➢ add passwords in payload configuration section

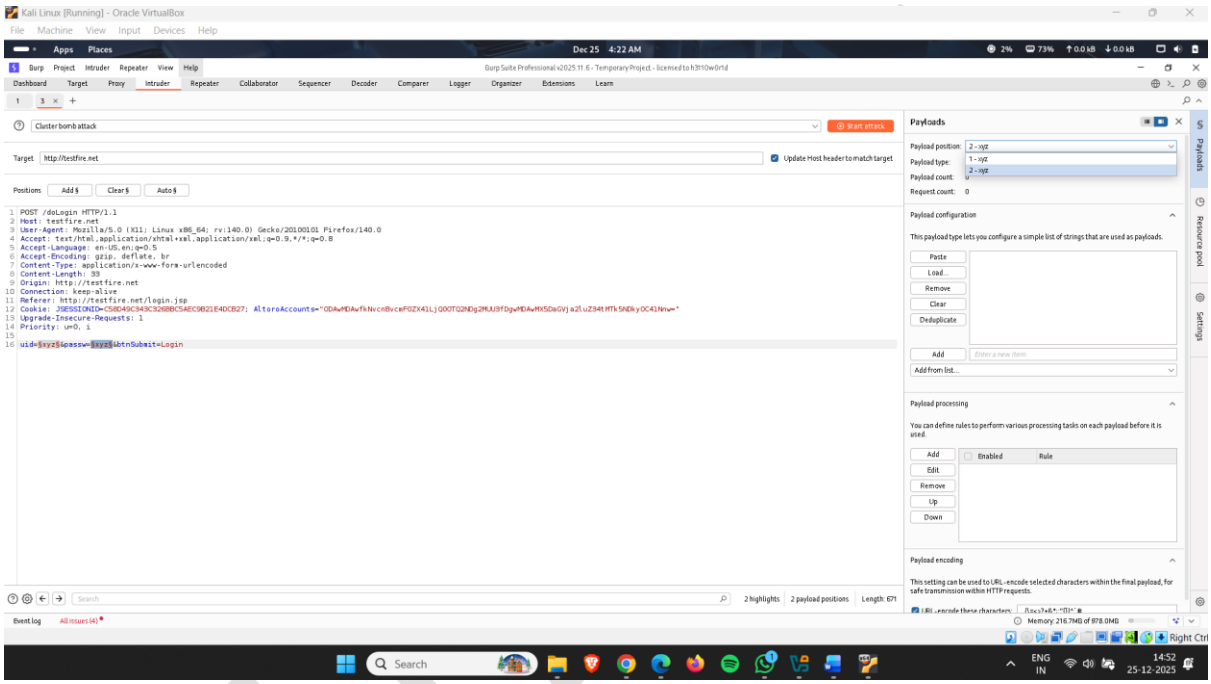


Figure 34

- Now, click on start attack button (Brute force attack will start)

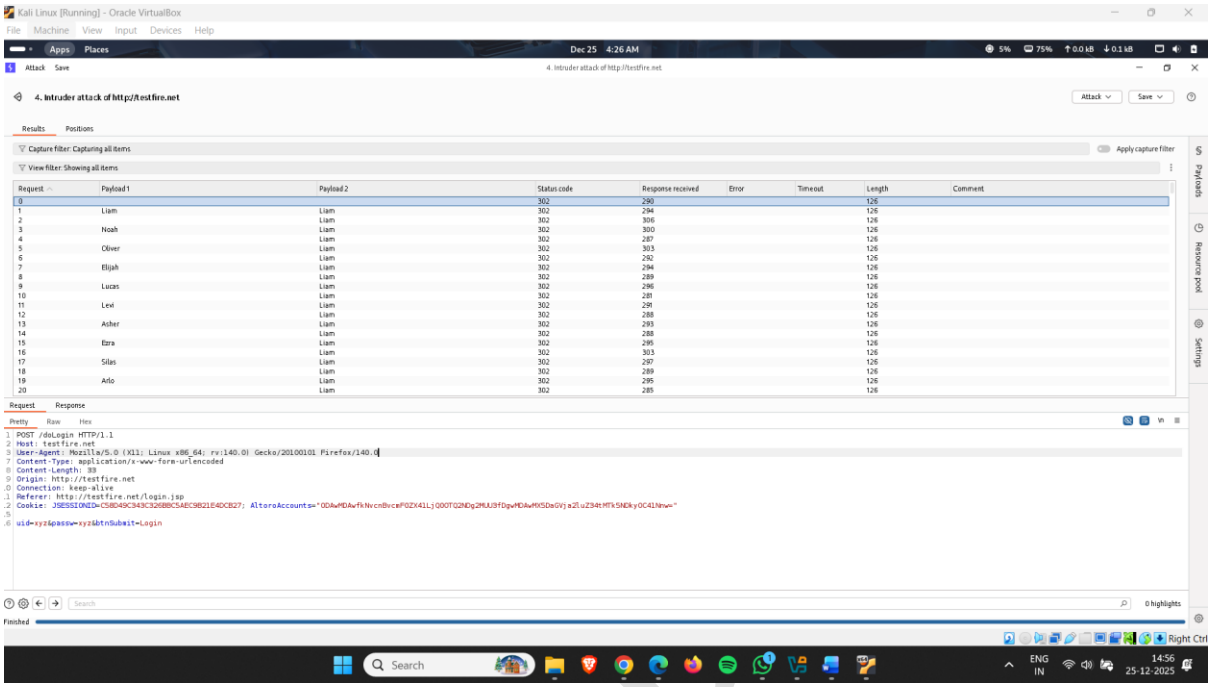


Figure 35

- Here we found password.

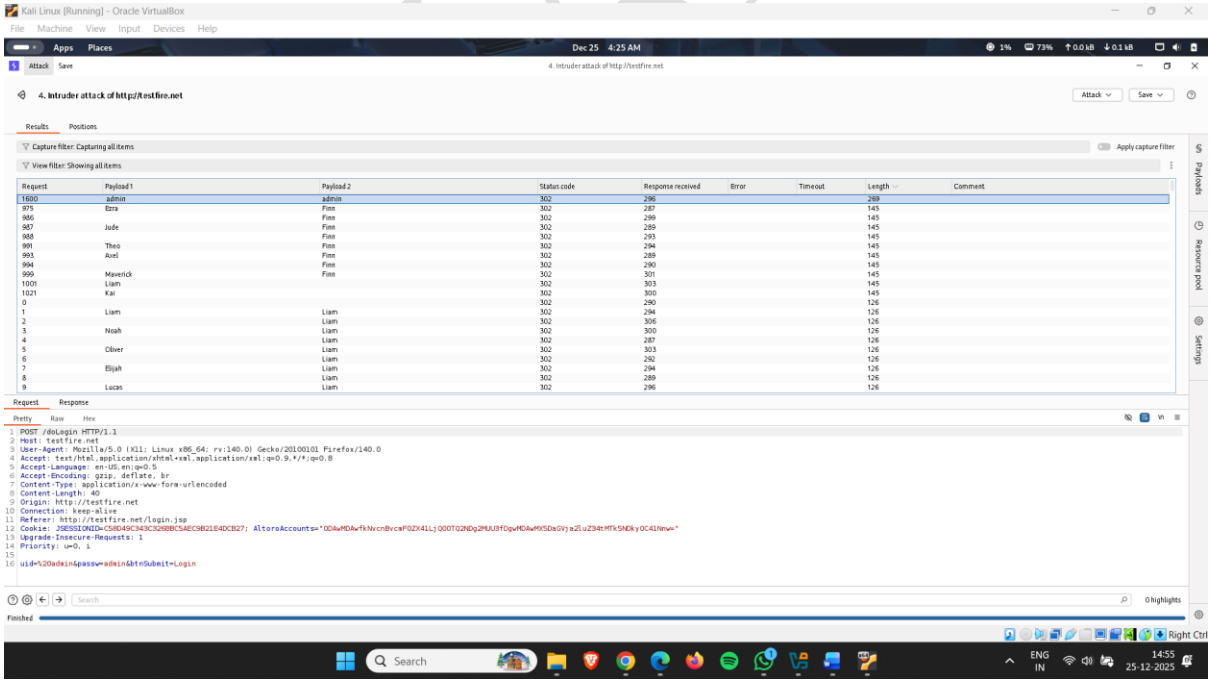


Figure 36

SQL Injection Attack Using Burp Suite Intruder

Burp Suite Intruder is an automated request generation, and fuzzing tool used in web application penetration testing. It allows a tester to insert multiple payloads into HTTP requests to observe how the server responds and identify weaknesses. Intruder works by selecting positions in a captured request, choosing payloads, and running different attack types to test how inputs affect application behavior.

Intruder supports four attack types:

- **Sniper:** Tests one parameter at a time with different payloads.
- **Battering Ram:** Sends the same payload to all positions simultaneously.
- **Pitchfork:** Sends parallel payloads to multiple positions at once.
- **Cluster Bomb:** Tests all combinations of payloads across multiple parameters.

Intruder is commonly used for input validation testing, user enumeration, IDOR checks, rate-limit testing, and hidden parameter discovery.
The Professional edition runs faster, while the Community edition is throttled, limiting speed.

In practice, Intruder is widely used for:

- enumeration (usernames, IDs, parameters),
- testing input validation,
- checking rate limits,
- and identifying business logic flaws.

How to do it:

- Open kali Linux terminal and start burp suite

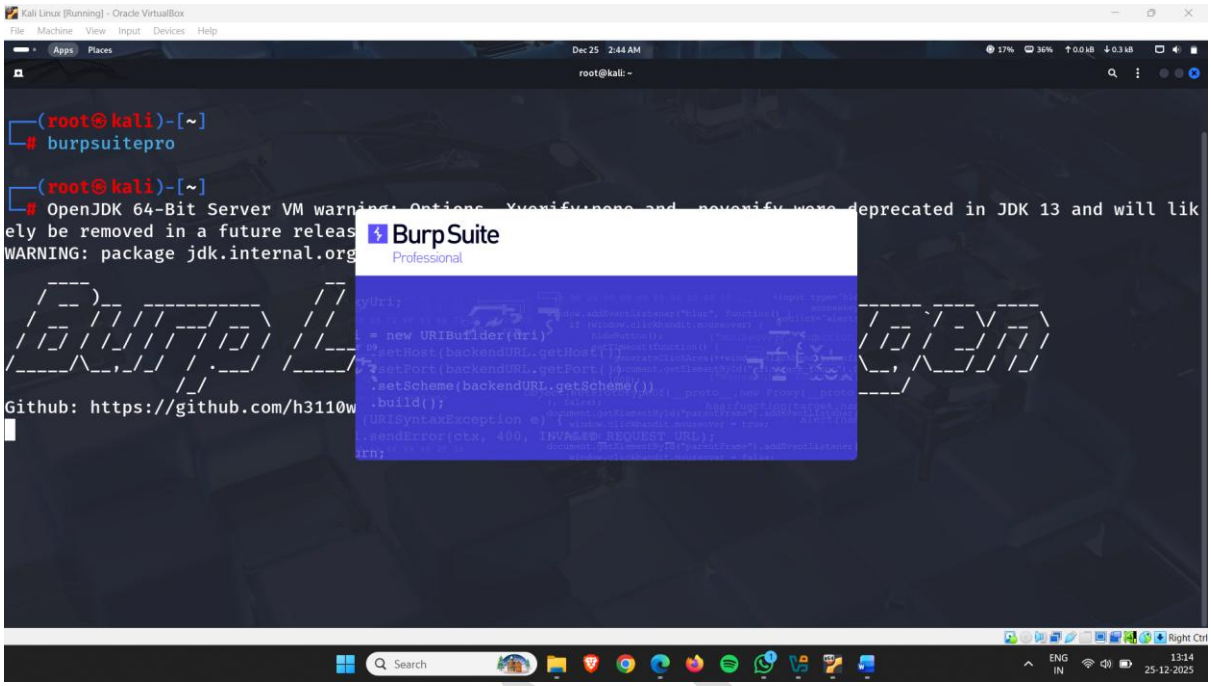


Figure 37

- Target website

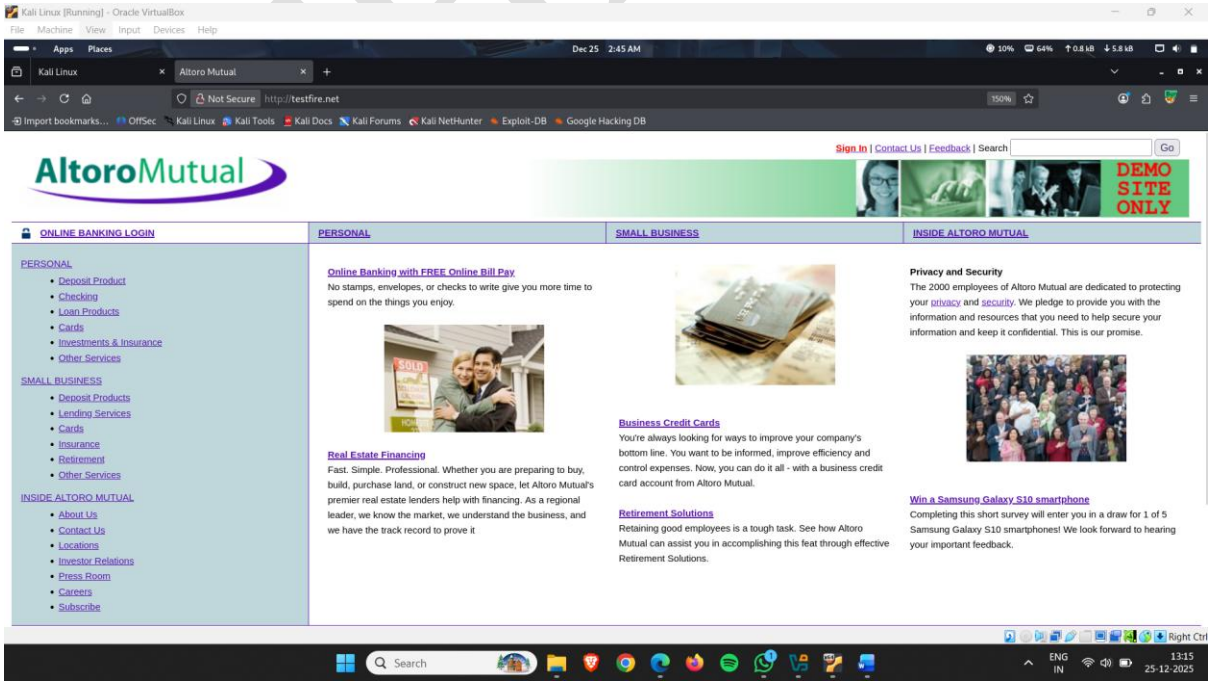


Figure 38

- Set proxy

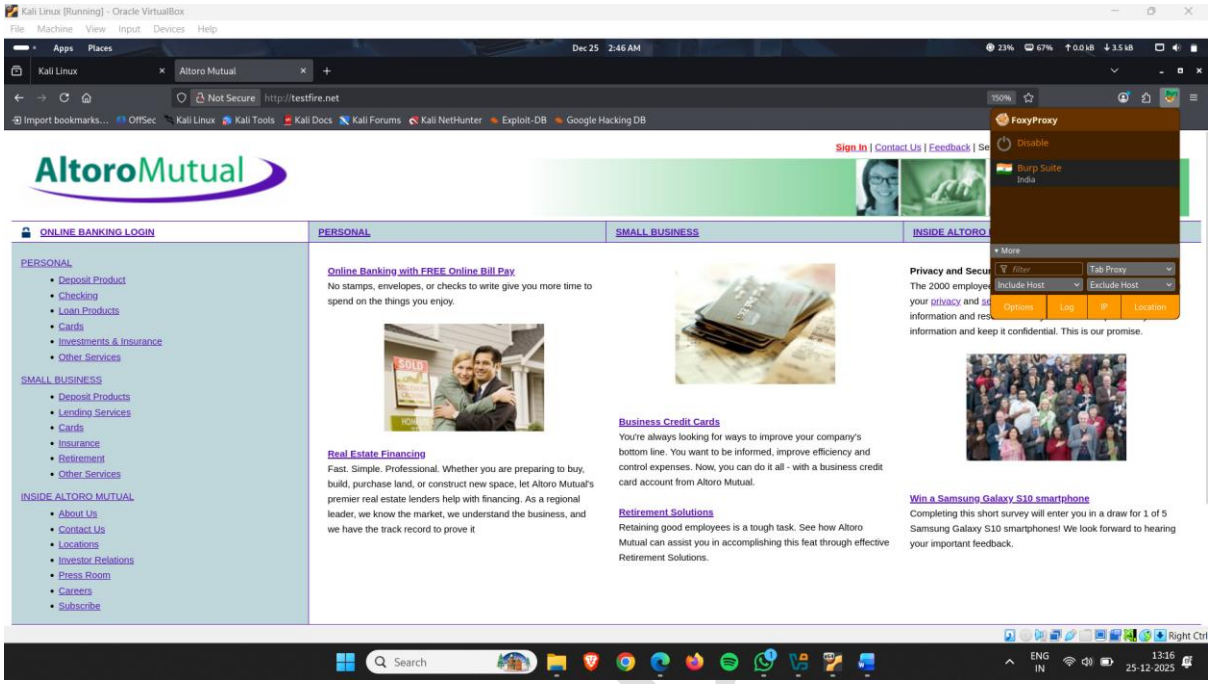


Figure 39

- Turn on interception

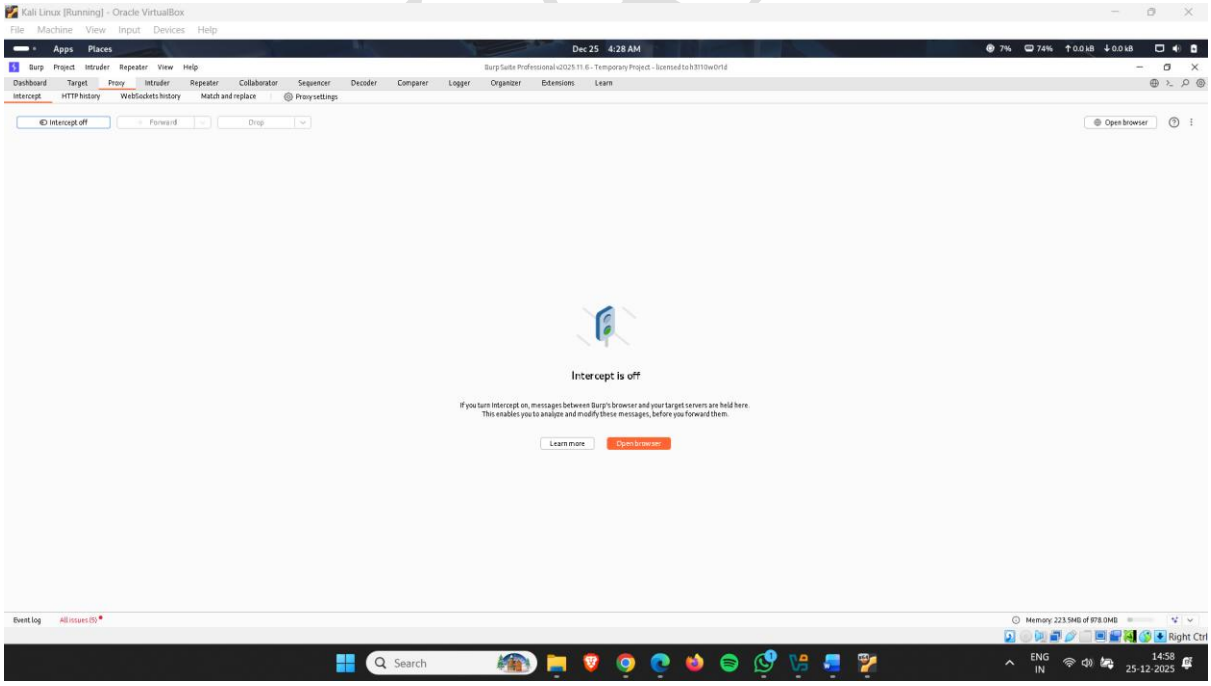


Figure 40

- Enter invalid username and password

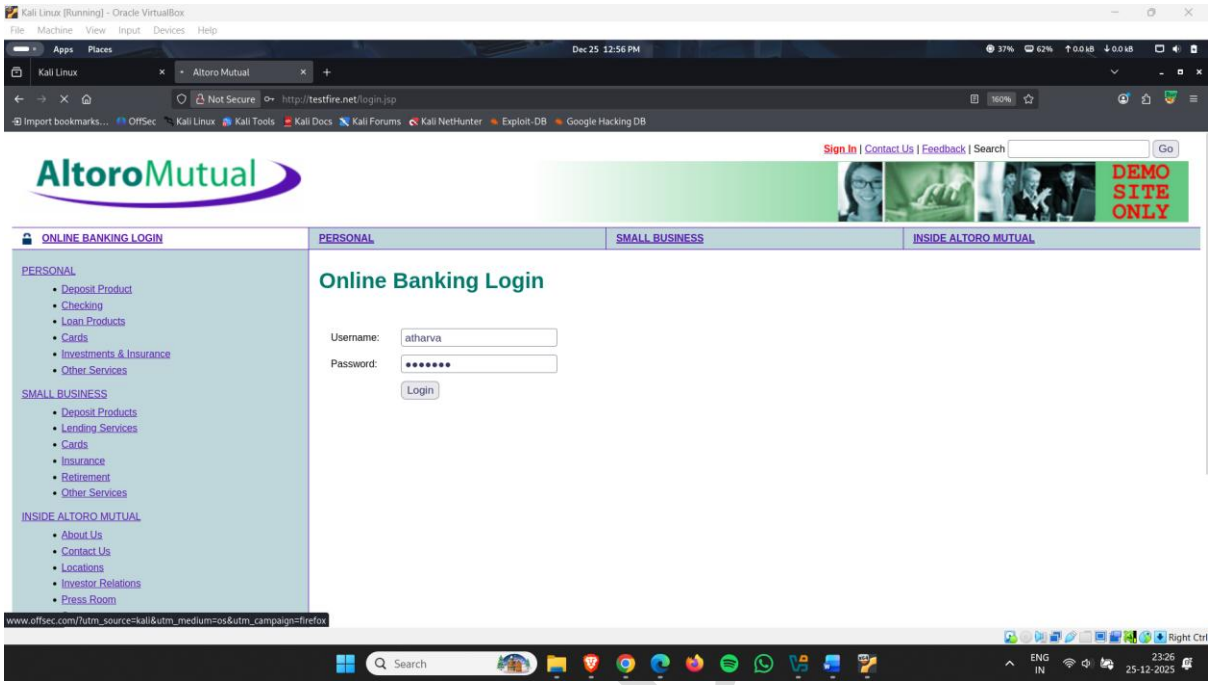


Figure 41

- Request captured
- Send this request to the burp intruder

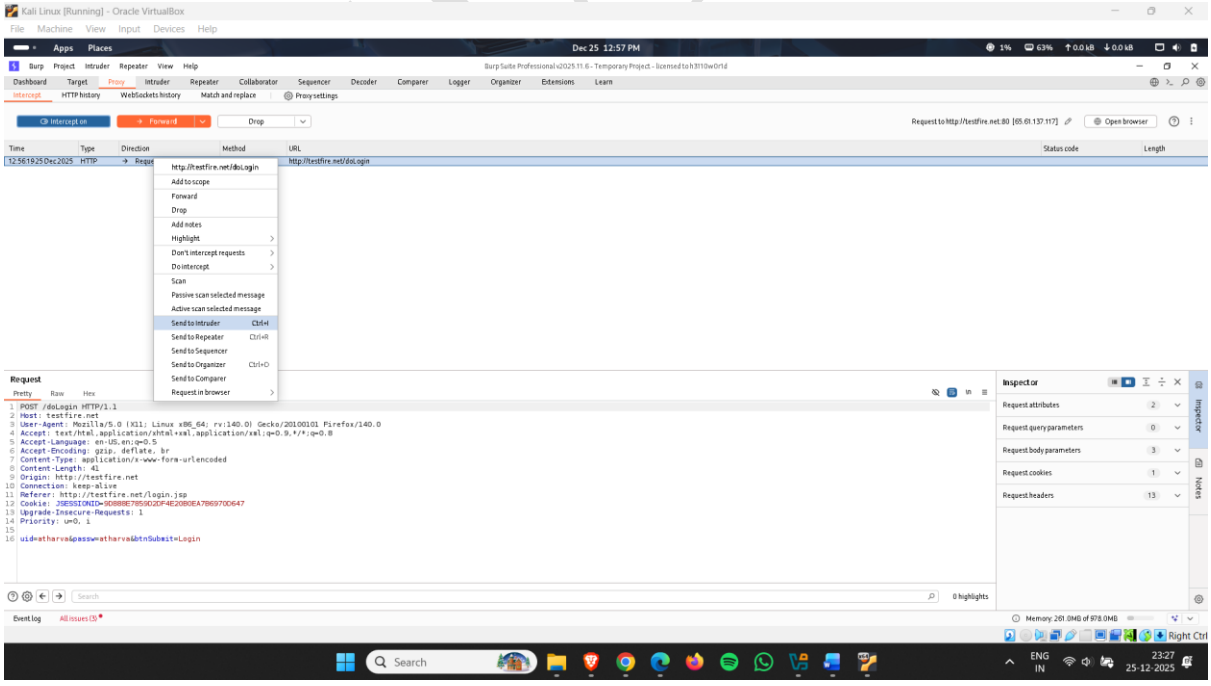


Figure 42

- In Intruder, select username section & click on Add$

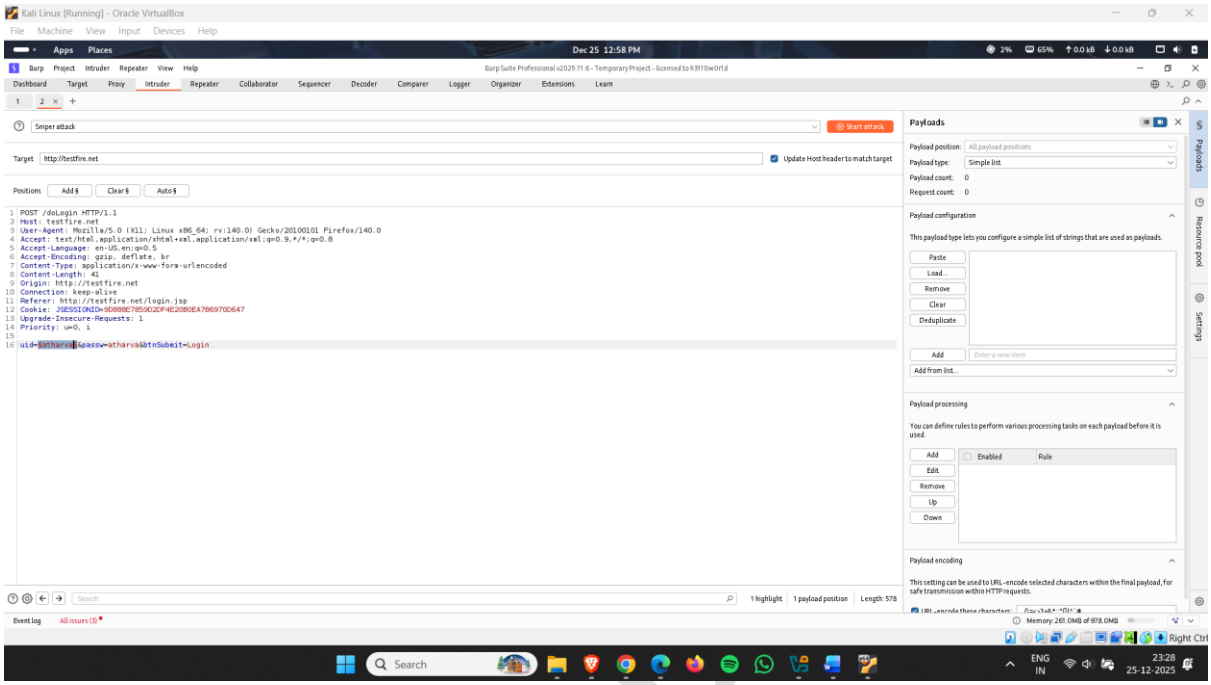


Figure 43

- Use sniper attack

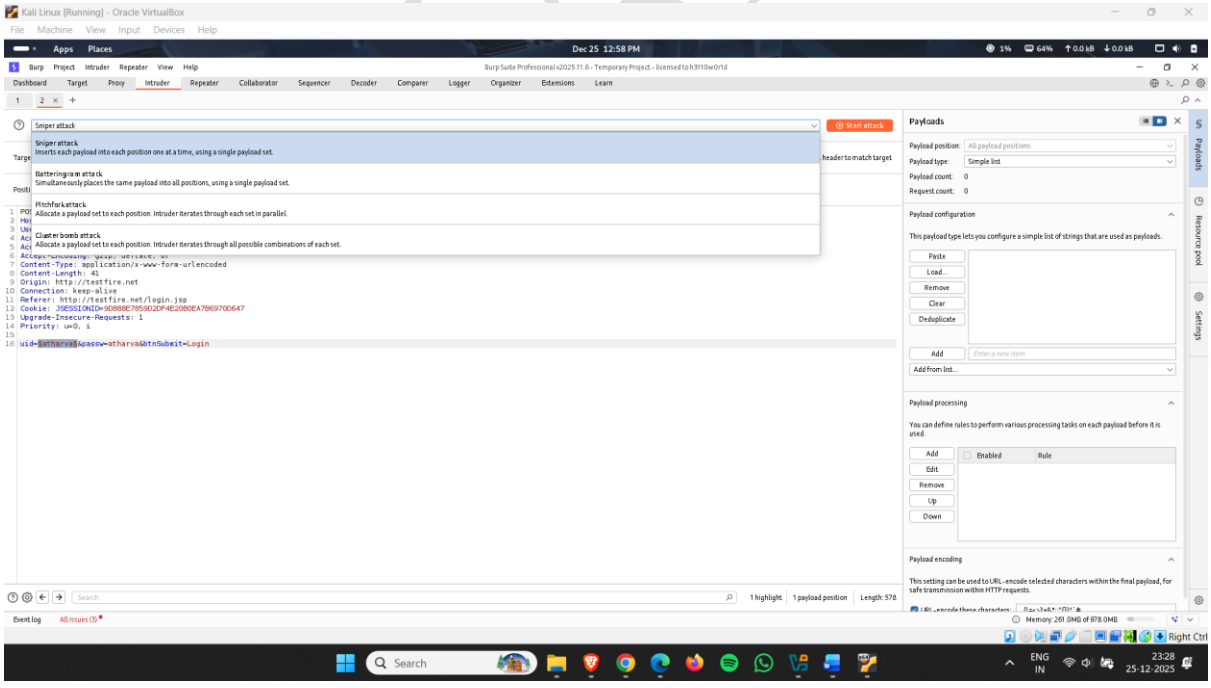


Figure 44

- Now in payload Configuration, click on Add from list.
- Select Fuzzing – sql injection
- Click on start attack

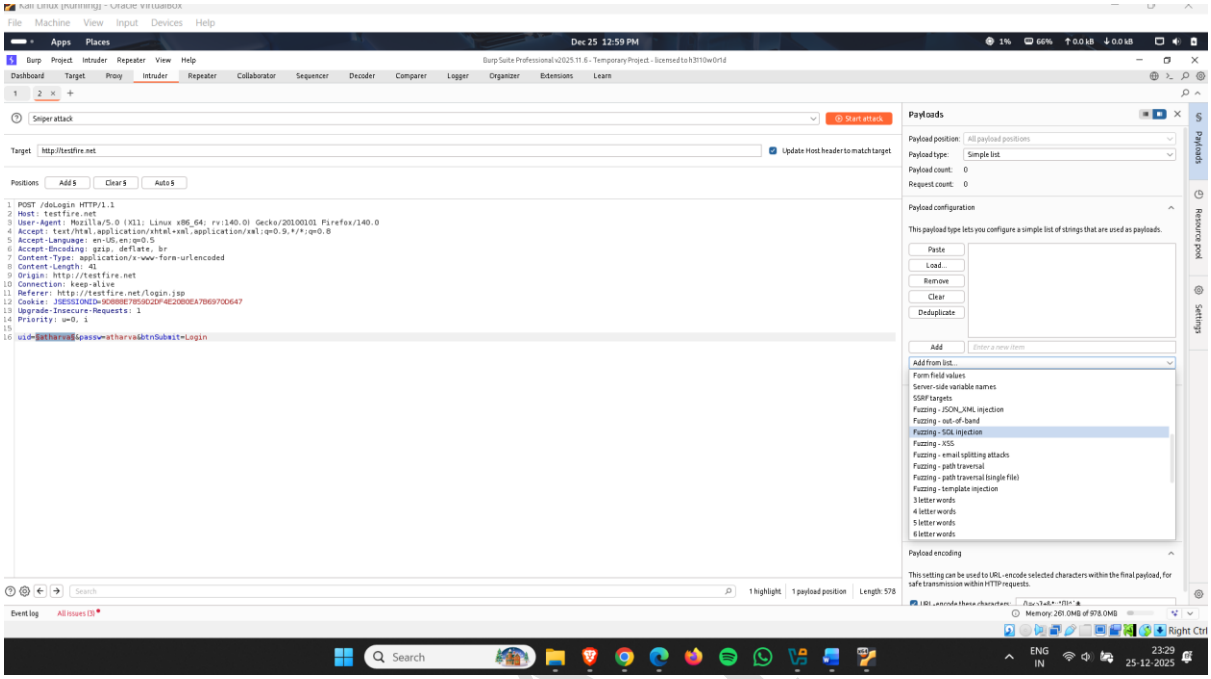


Figure 45

- Here, sql injection attack started.

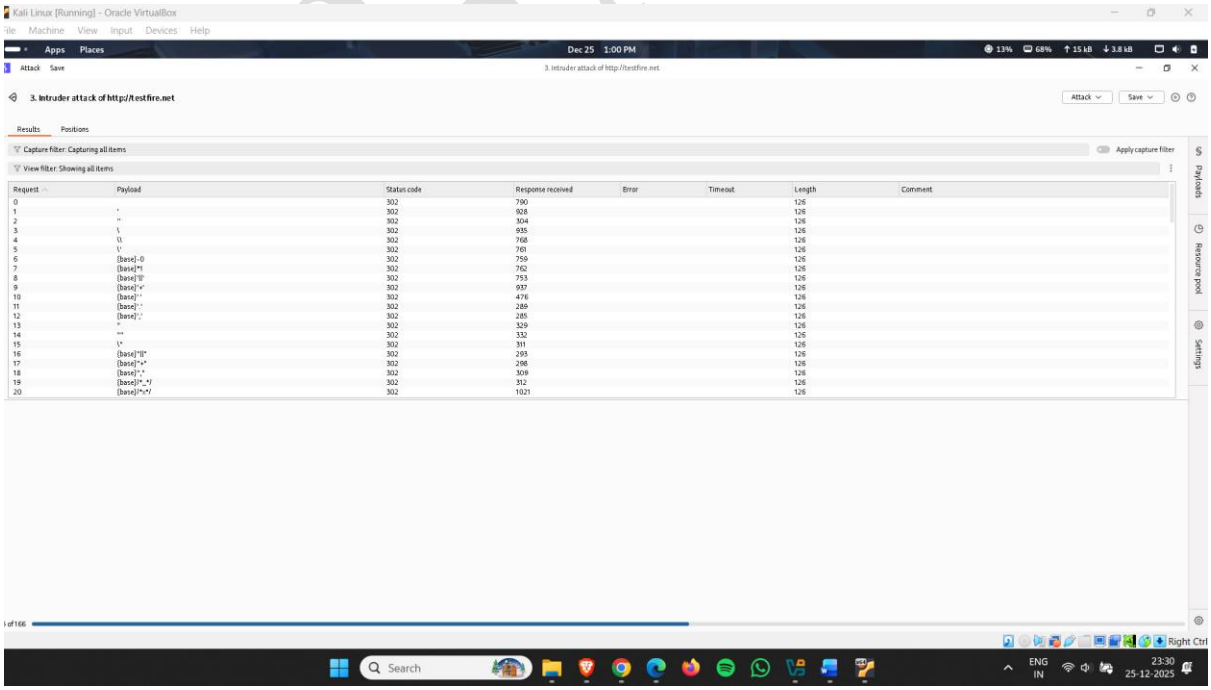


Figure 46

- Now open first payload
- Copy Sql injection Payload

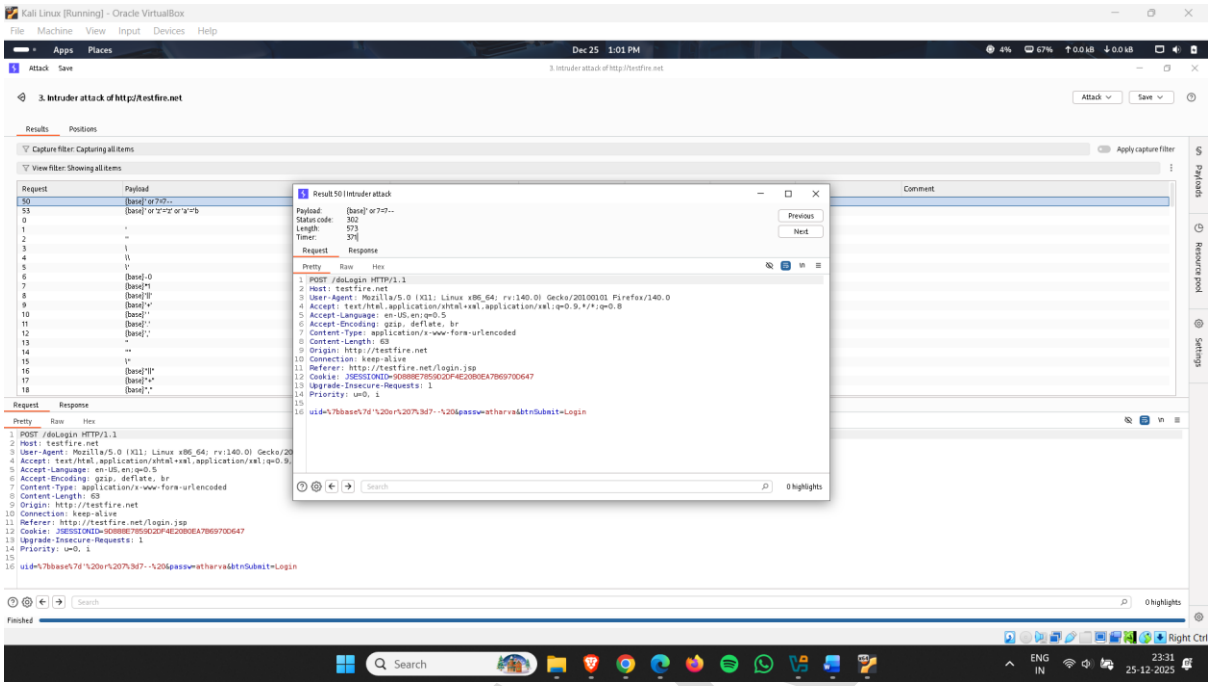


Figure 47

- Go to the target website
- And paste payload on both username and password section, and click on login.

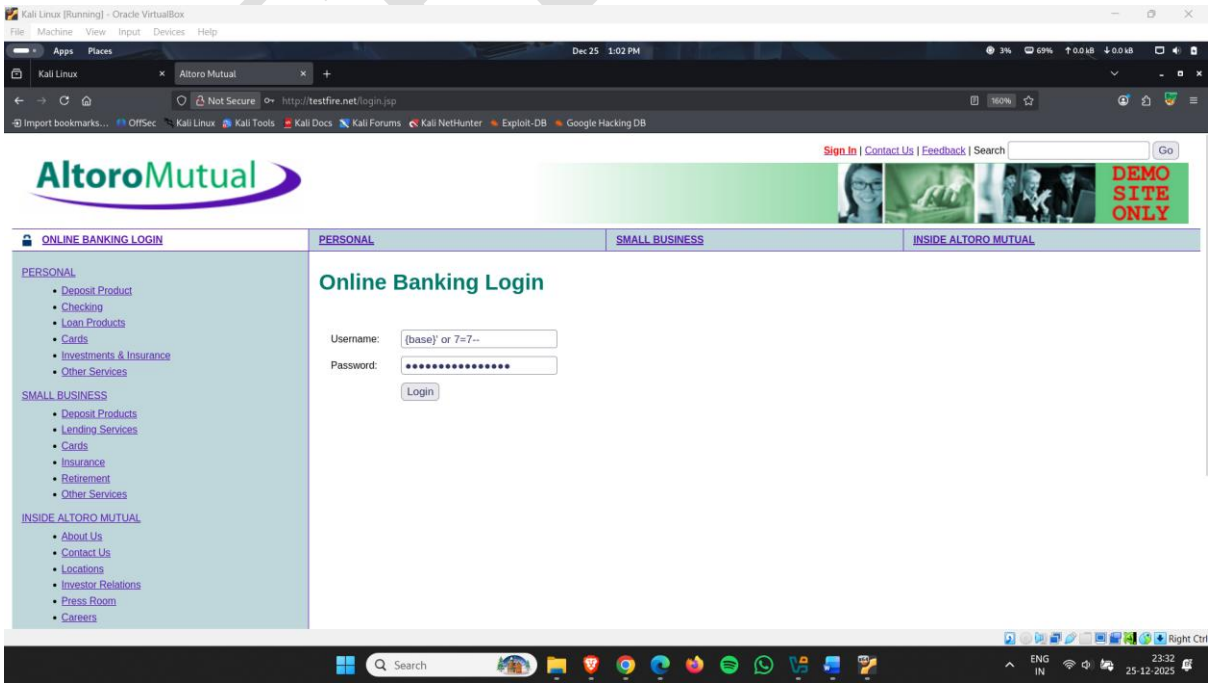


Figure 48

- Here, we have logged in successfully without username and password.

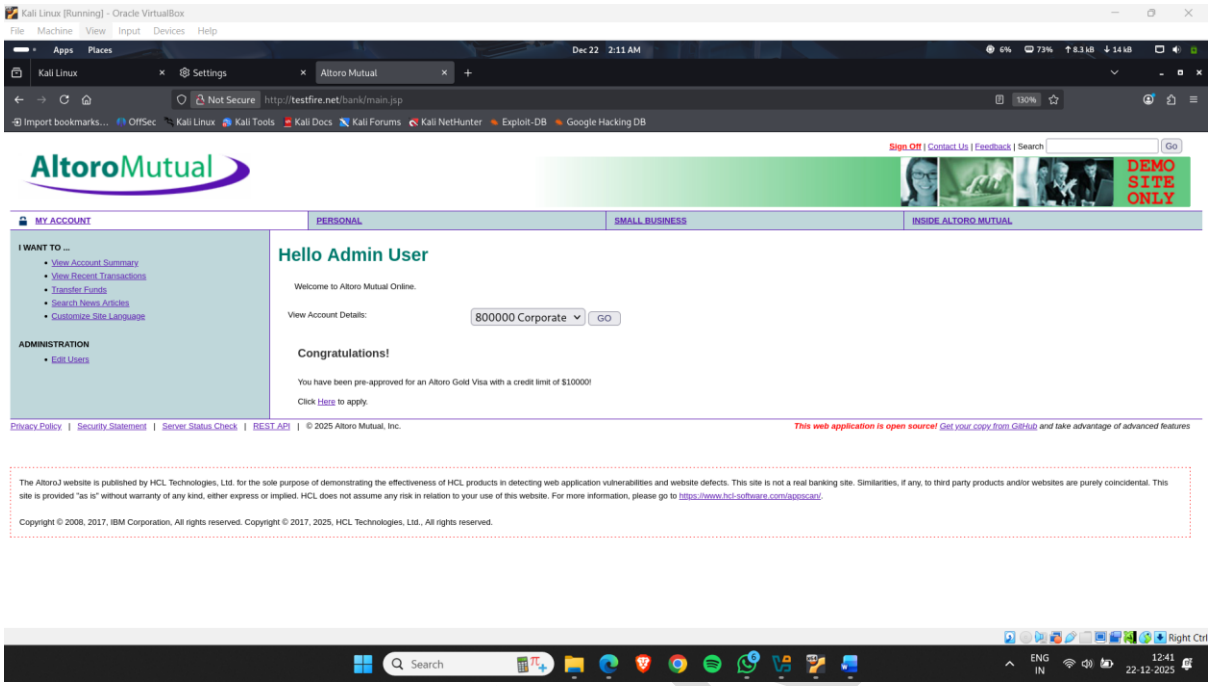


Figure 49

SQL Injection Attack Using Burp Suite Repeater

- Setup Proxy

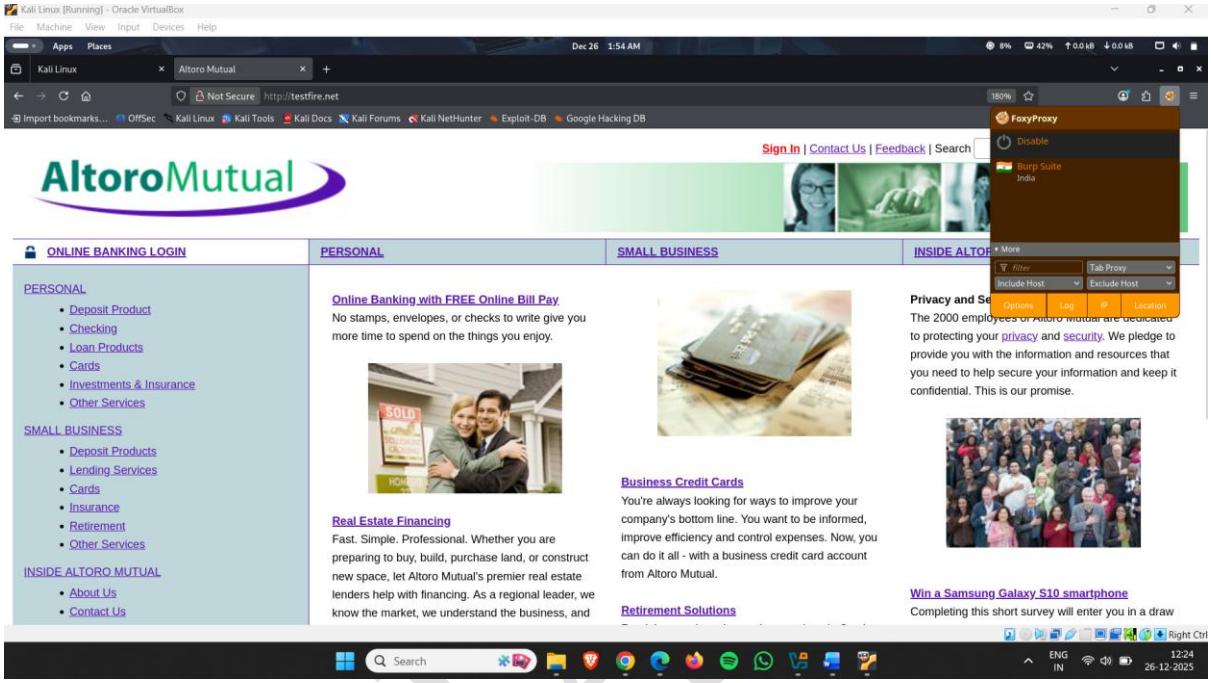


Figure 50

- Turn on intercept

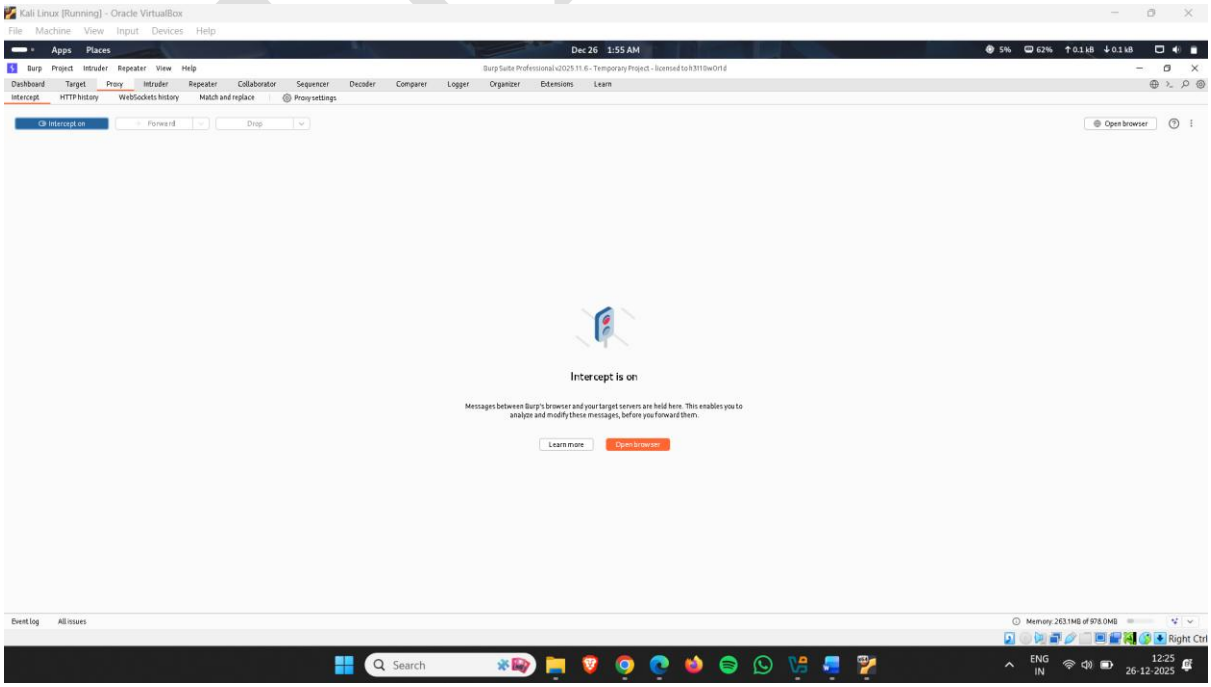


Figure 51

- Enter invalid username and password

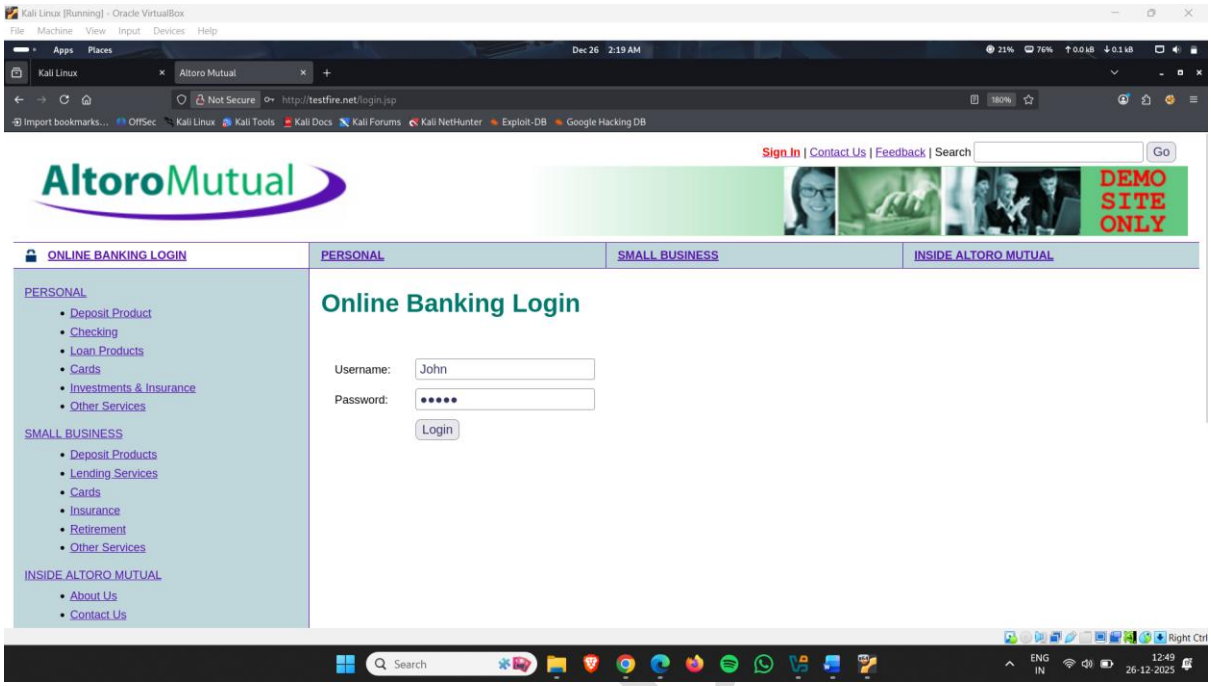


Figure 52

- Request captured

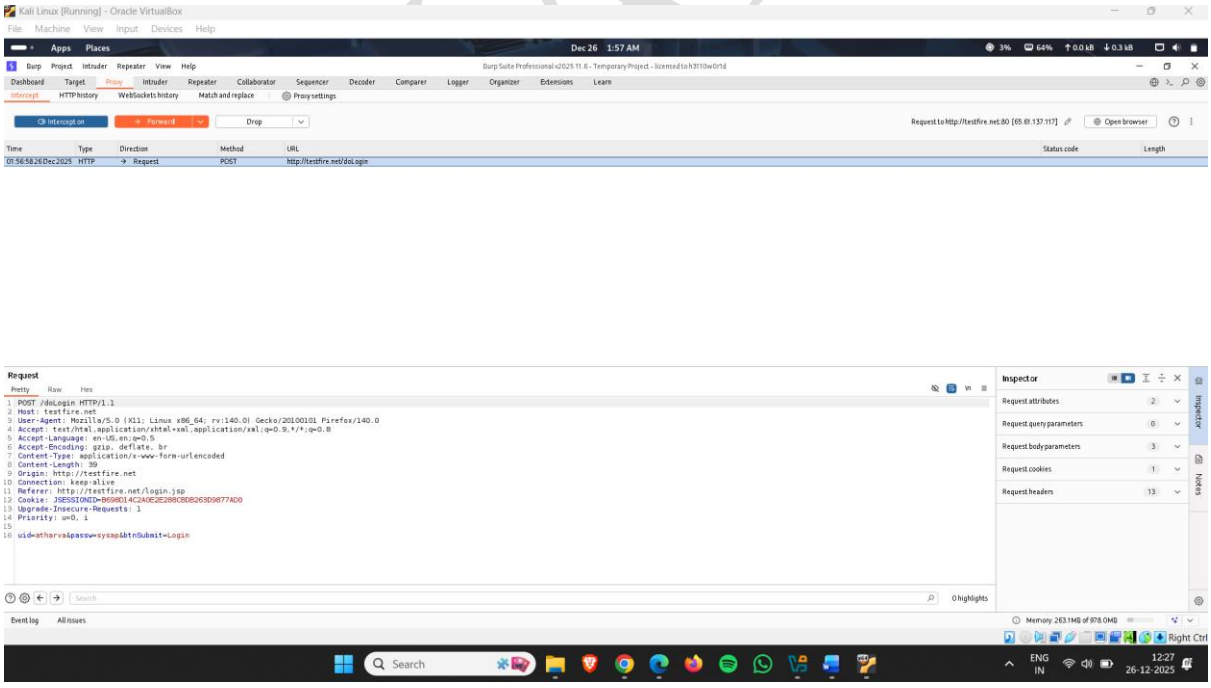


Figure 53

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

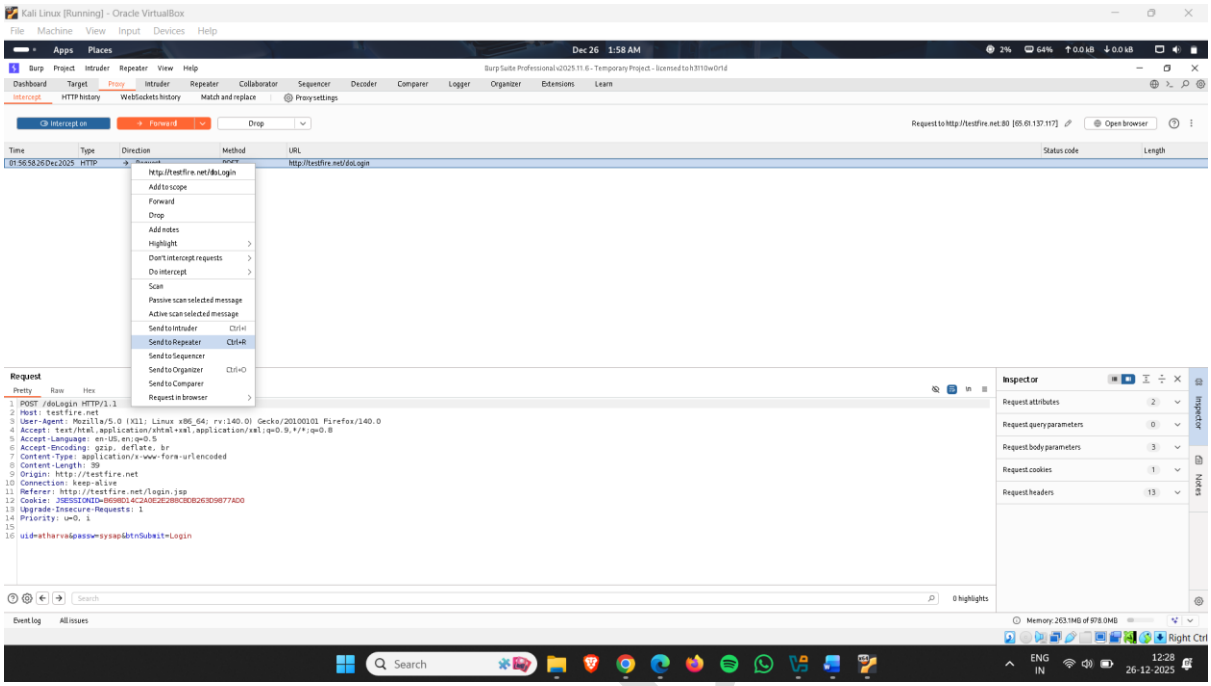


Figure 54

- Now Click on send the request to see the response

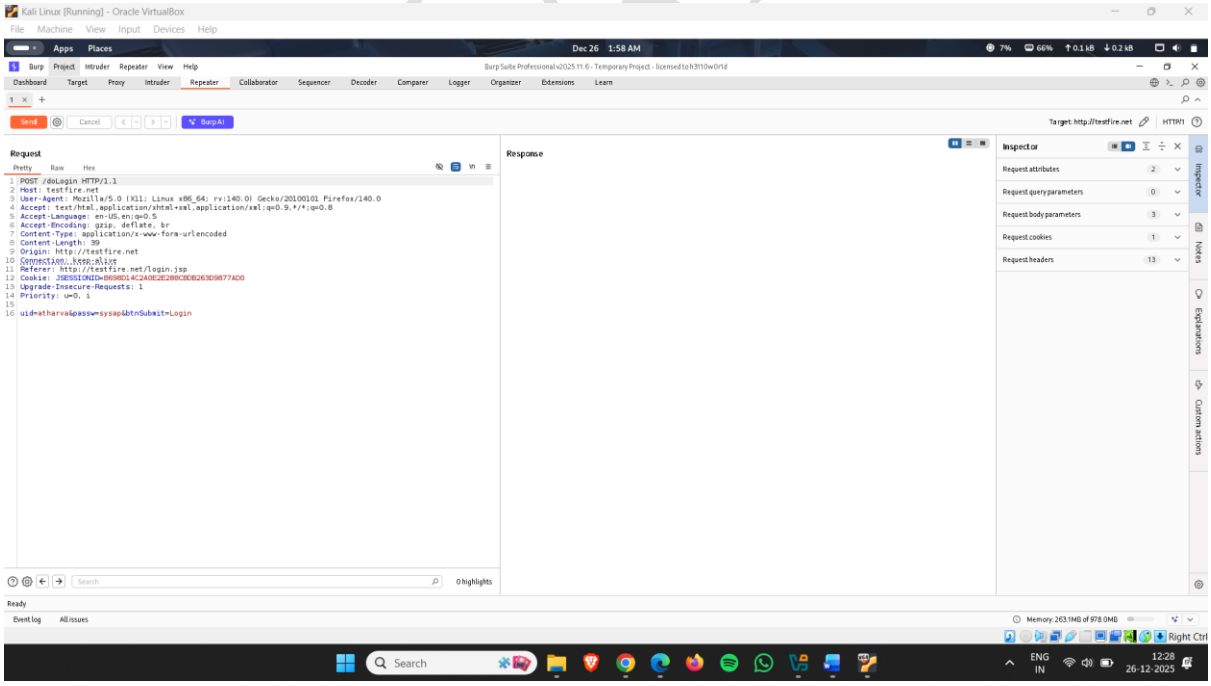


Figure 55

- Response 302 received
- Change username sysap to admin'--

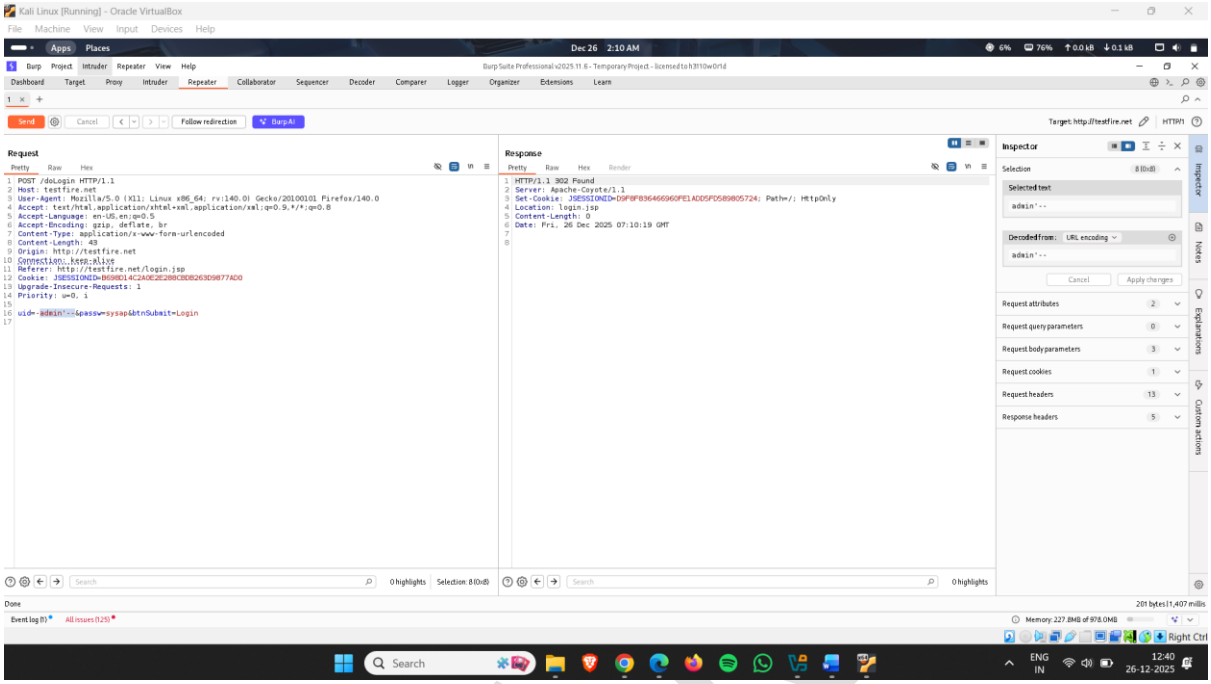


Figure 56

- Now go to the target website login page
- Enter sql injection on both username and password section and click on login
- Logged in successfully

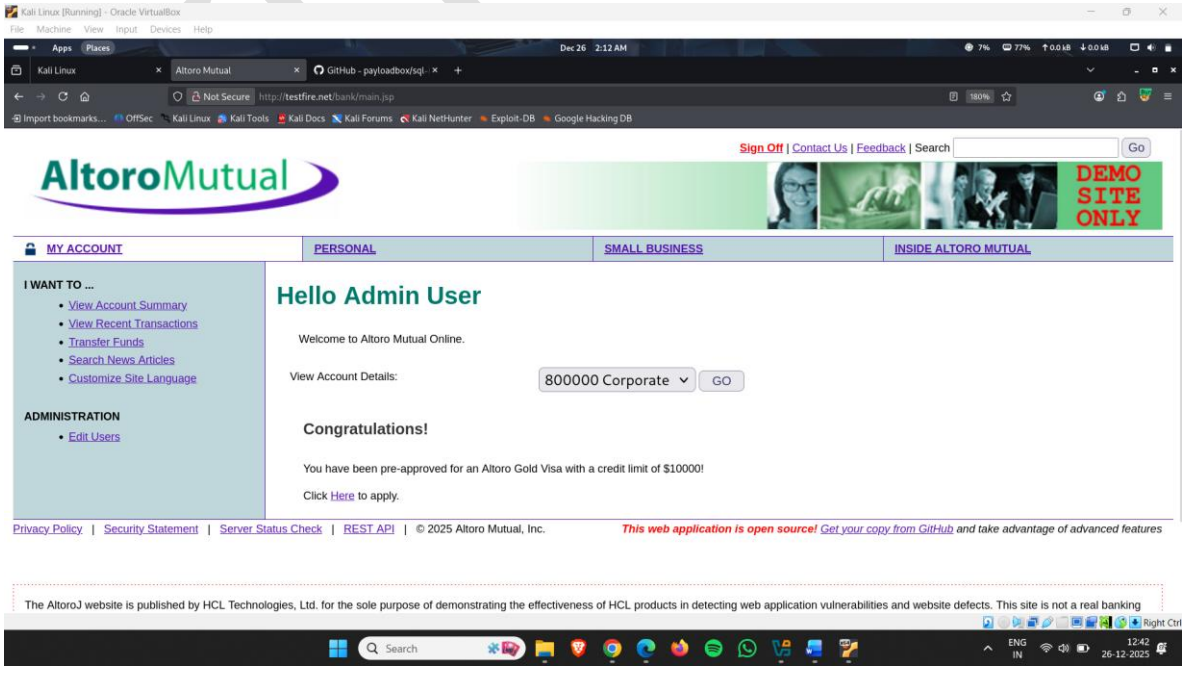


Figure 57

- **Use another sql injection: '%20or%20'z'%3d'z**
- Send the request

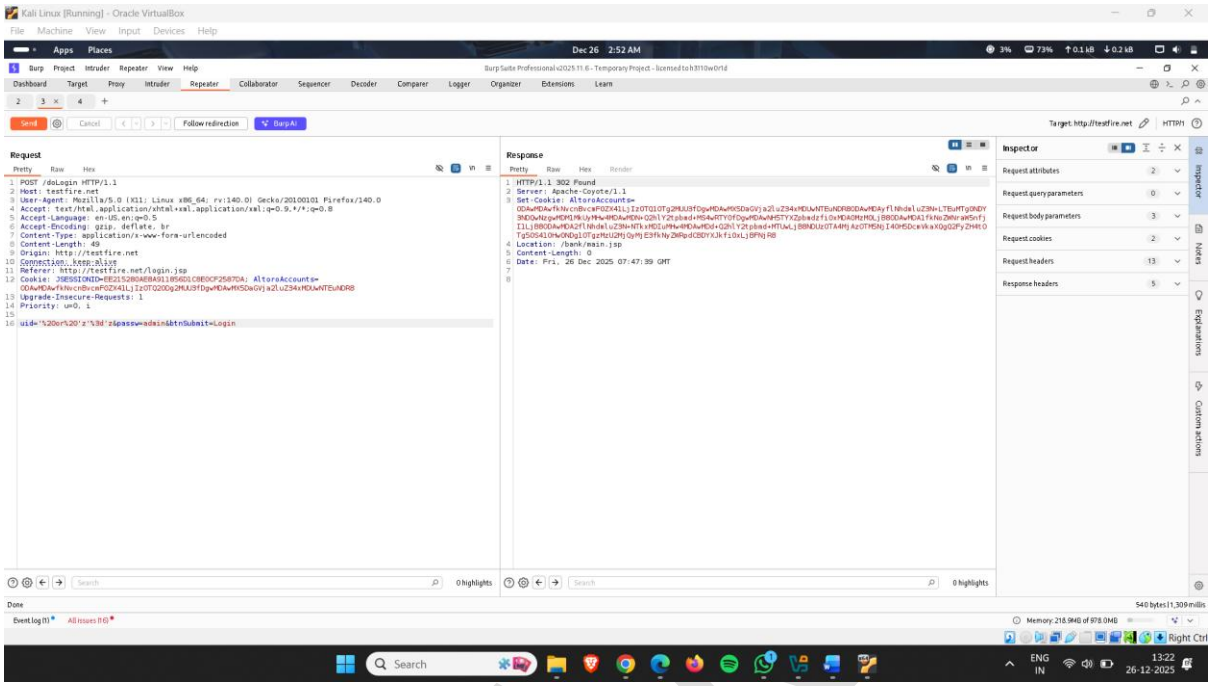


Figure 58

- Enter Sql injection on both username and password

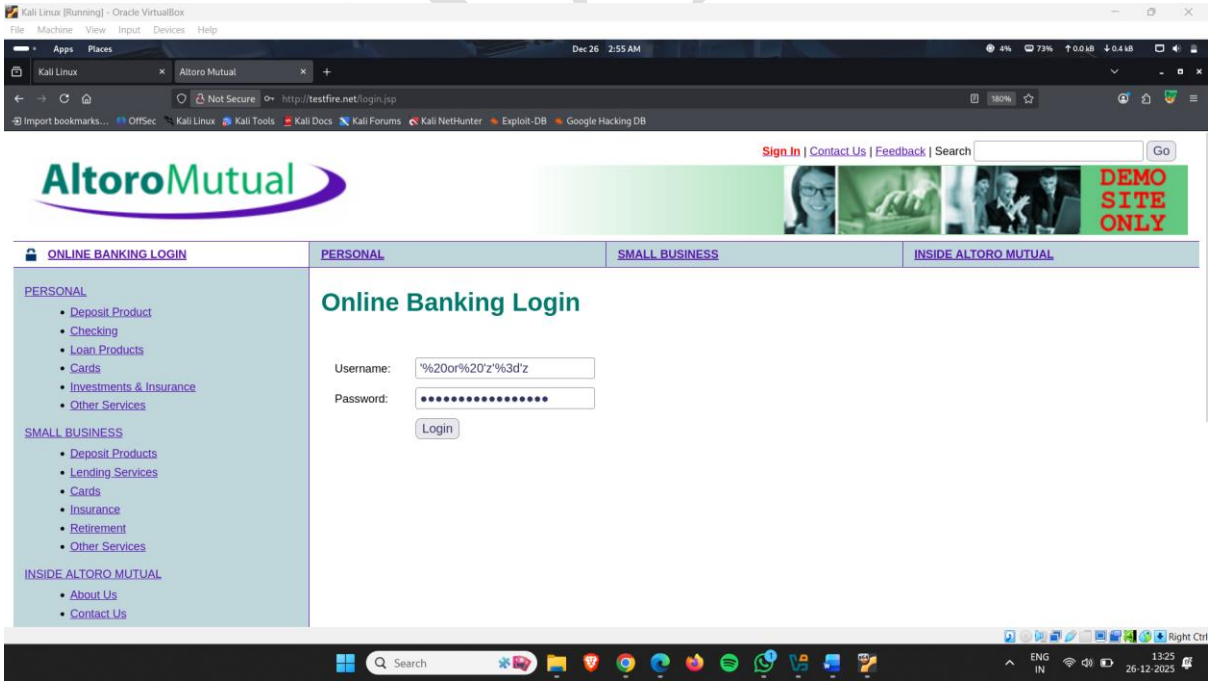


Figure 59

- Click on Login
- Logged in successfully

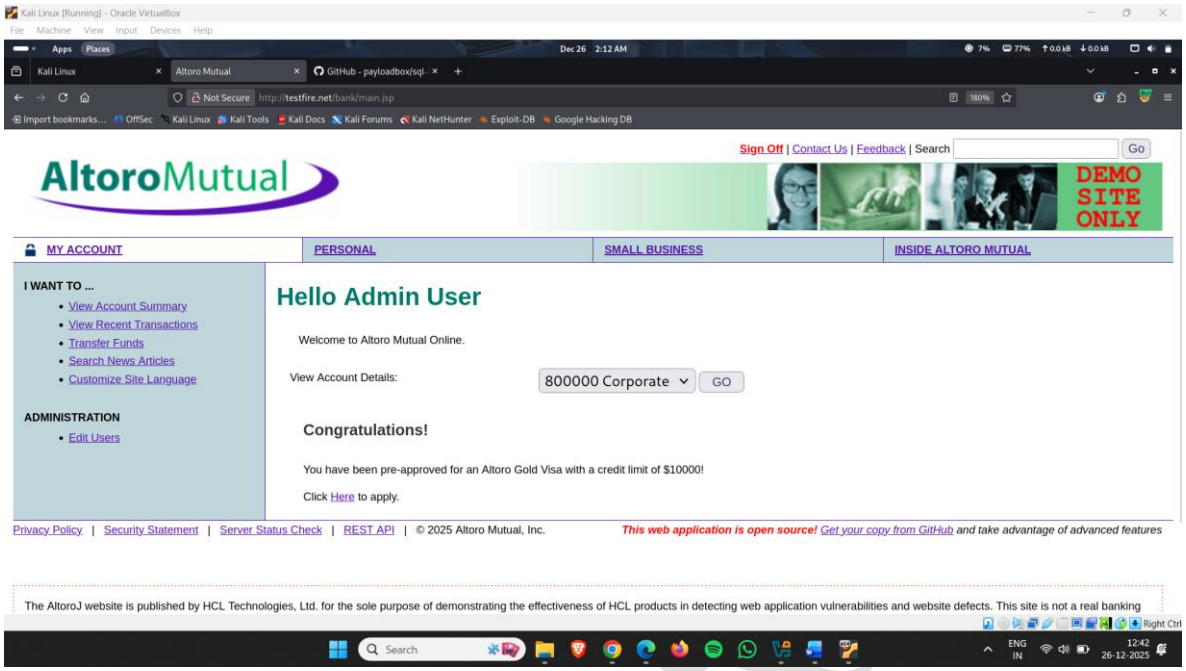


Figure 60

Cross-Site Scripting (XSS) Attack

Cross-Site Scripting (XSS) is a type of web application vulnerability that allows an attacker to inject malicious scripts (usually JavaScript) into web pages viewed by other users. When the victim accesses the page, the injected script executes inside their browser without their knowledge or consent, potentially leading to data theft, session hijacking, or other malicious actions.

How XSS Works

1. **Identify an Input Field**
   The attacker finds a user input area such as a search bar, comment box, login form, or URL parameter where user input is not properly validated or sanitized.
2. **Inject Malicious Script**
   The attacker inserts harmful JavaScript code into the input field.
   Example:<script>alert('XSS')</script>
3. **Script is Returned by the Website**
   The vulnerable web application reflects or stores the injected script on a webpage.
4. **Execution in the User's Browser**
   When another user (or even the attacker) loads that webpage, the malicious script executes in their browser, performing unauthorized actions on behalf of the user.

Types of XSS:

1. **Stored XSS (Persistent XSS)**
2. **Reflected XSS (Non-Persistent XSS)**
3. **DOM-Based XSS (Document Object Model XSS)**

Cross Site Scripting (XSS) Attack Using Burp Suite Intruder

How to do it -:

- Turn on burp suite Intercept

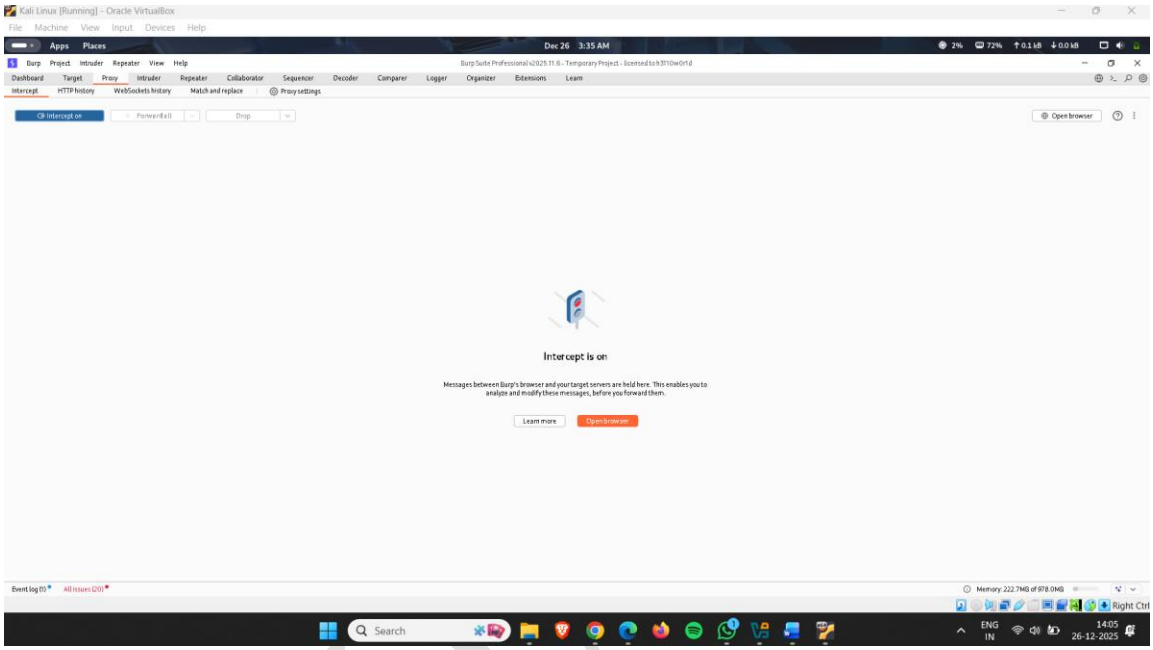


Figure 61

- Set up proxy

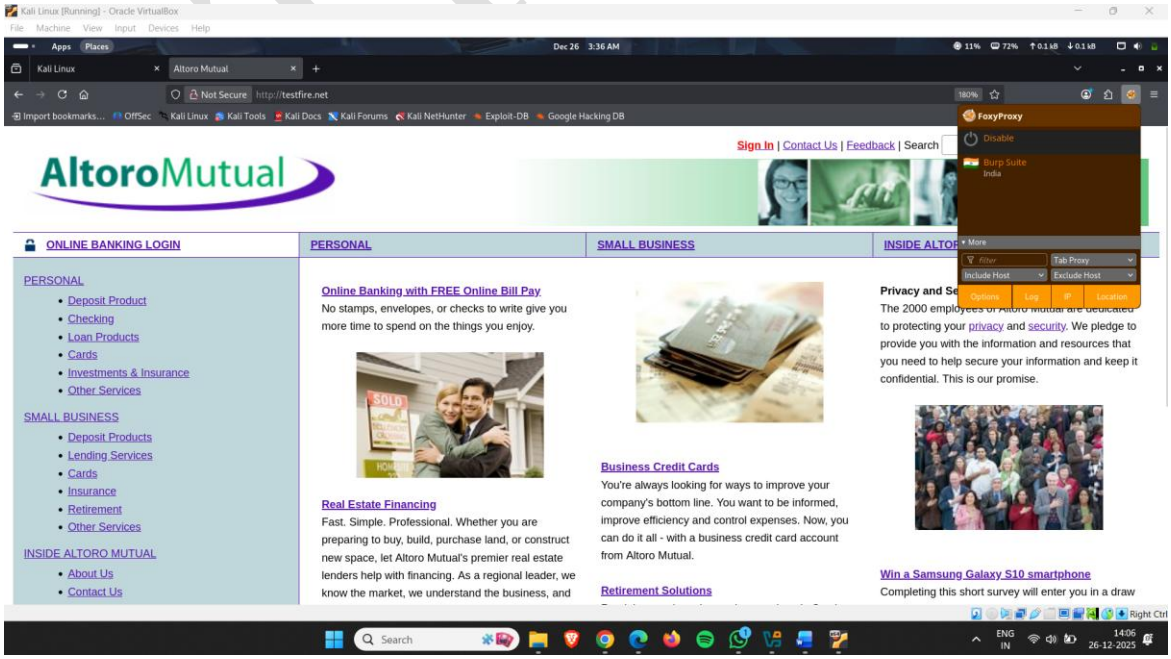


Figure 62

- Target website login page

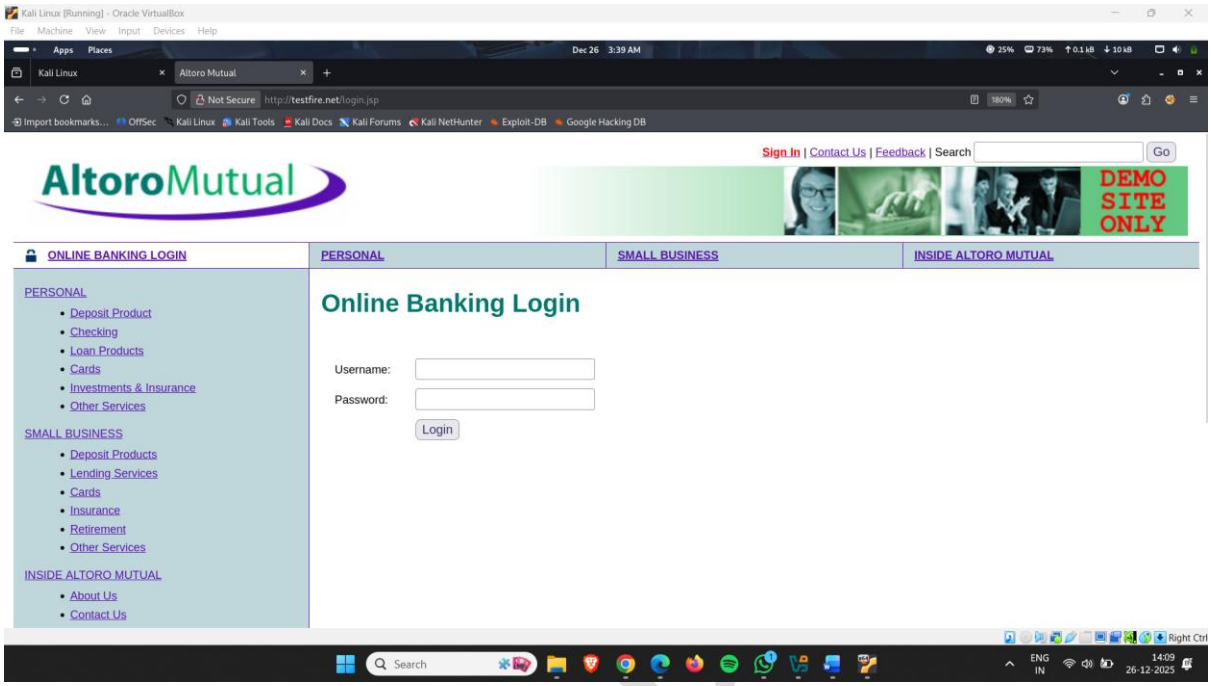


Figure 63

- Enter anything in search section and click on go

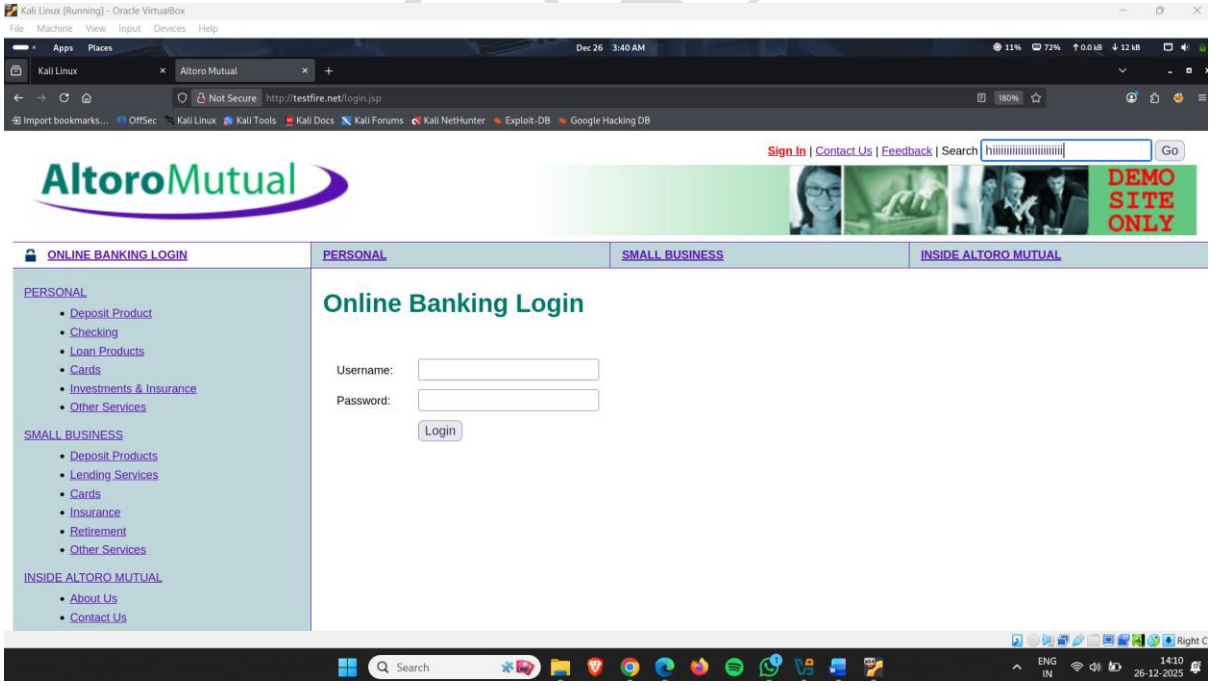


Figure 64

- Request captured

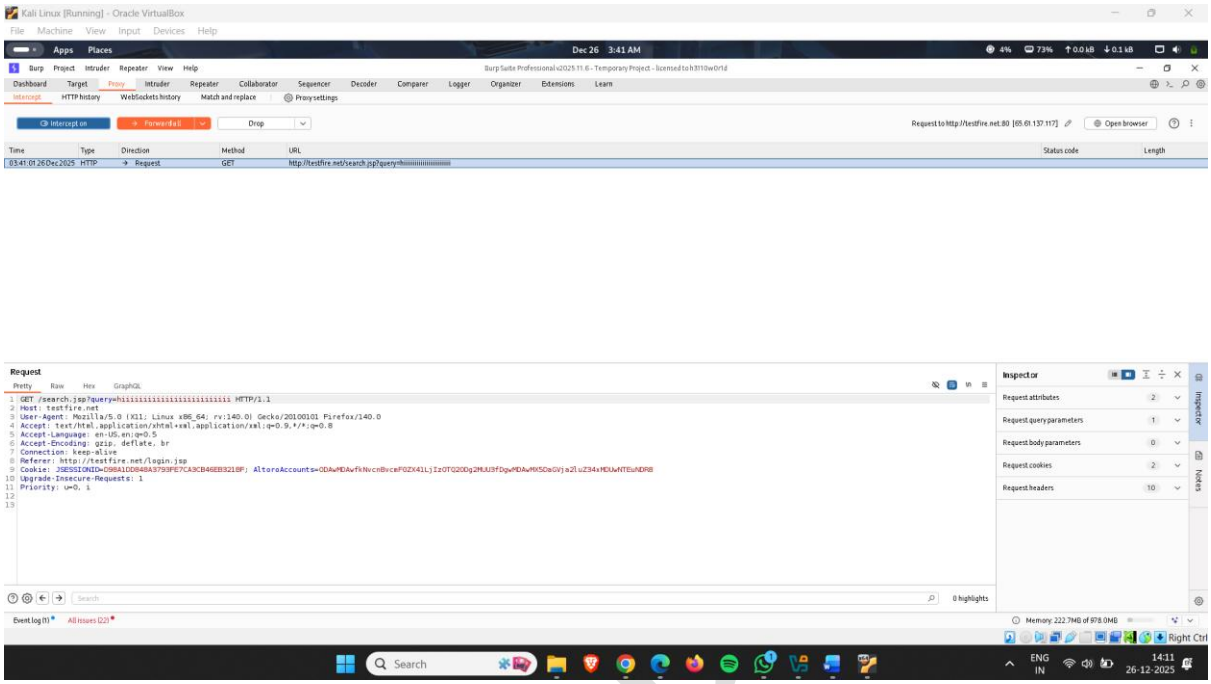


Figure 65

- Right click on request and send It to intruder

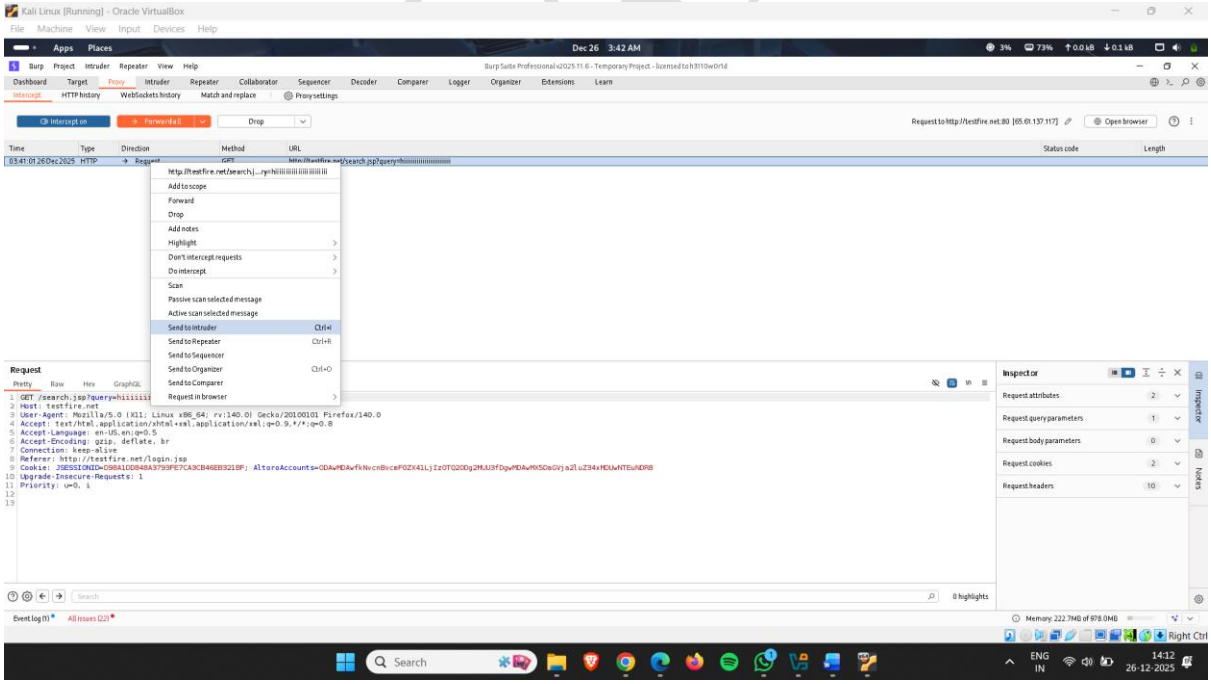


Figure 66

- Now select the hiiiiiiiiiiiiiiiiiiiiiii parameter and click on Add$

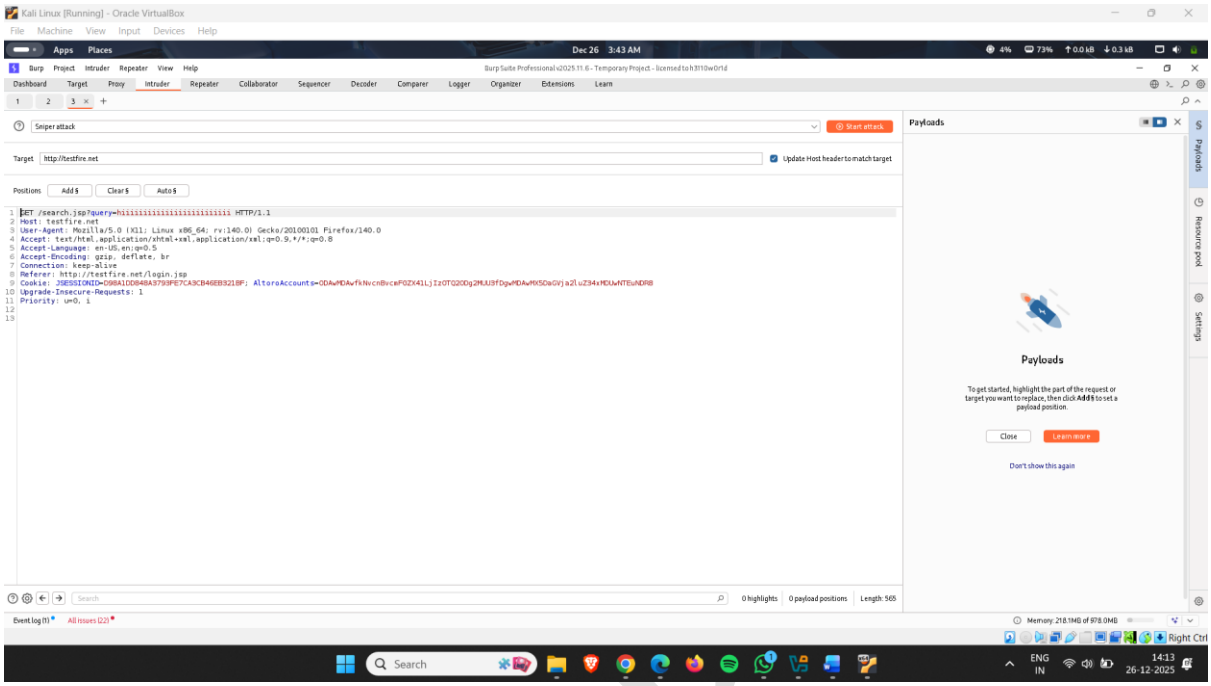


Figure 67

- Now, in Payload Configuration section
- Select Fuzzing-XSS

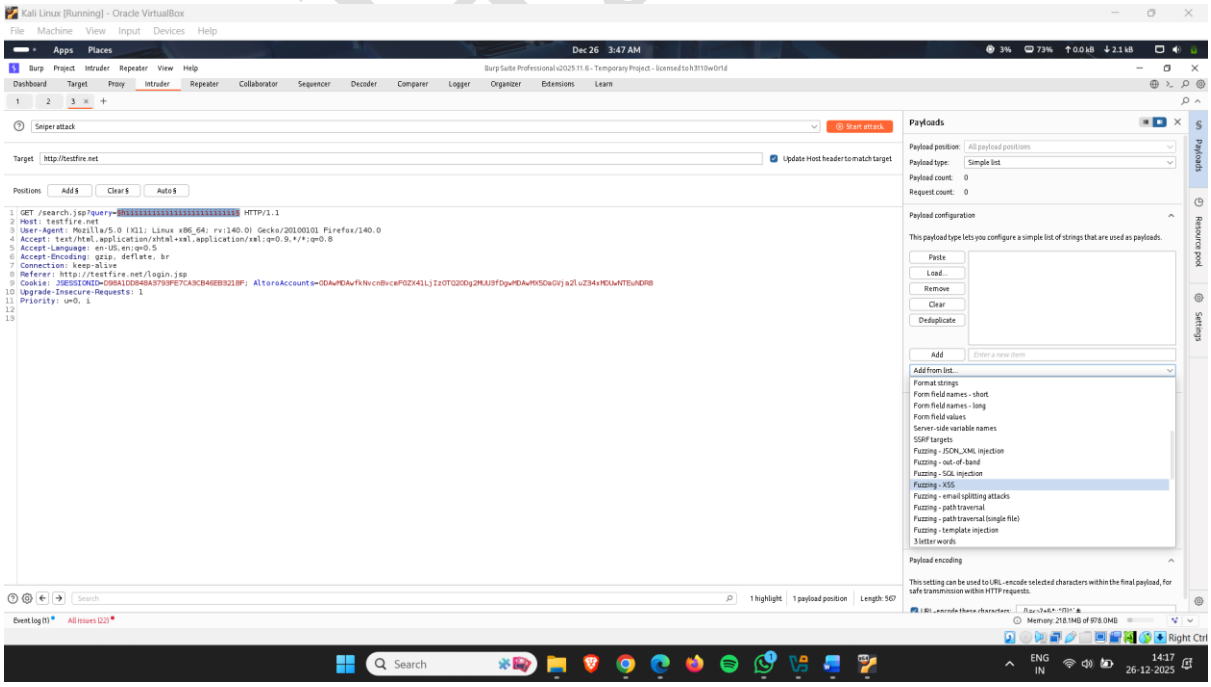


Figure 68

- Attack Started

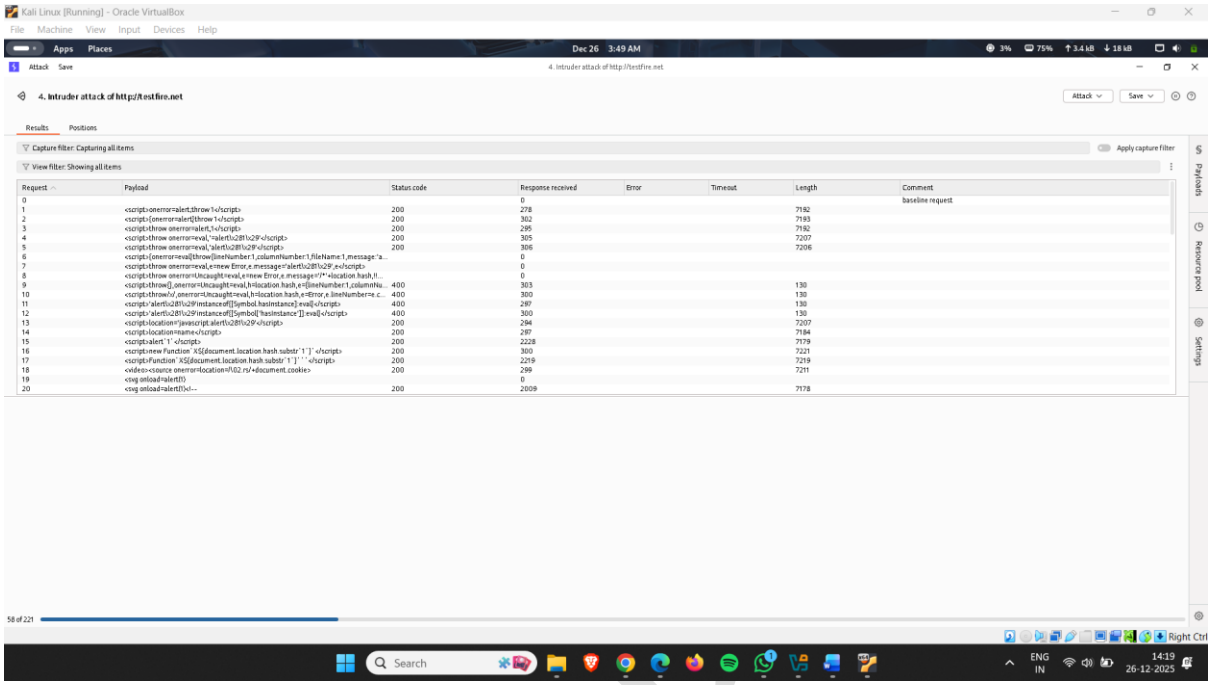


Figure 69

- Now copy payload

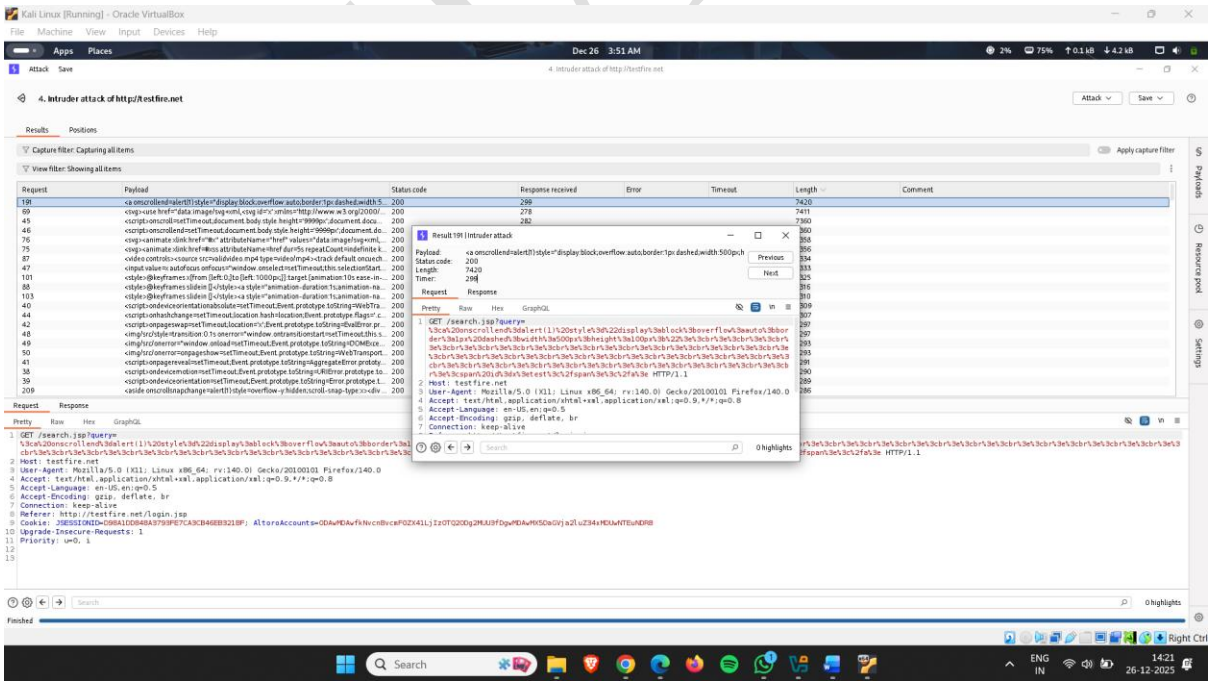


Figure 70

- Now copy any payload and paste in search section on target website and click go.

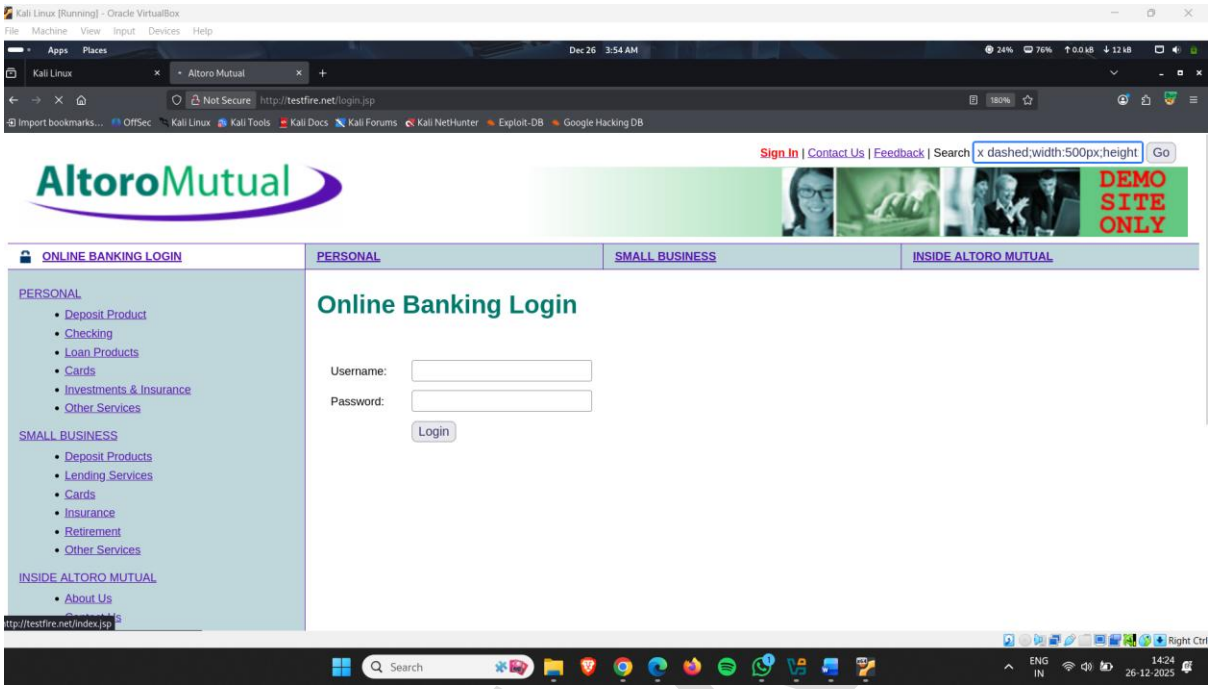


Figure 71

- Here a Pop up arrives

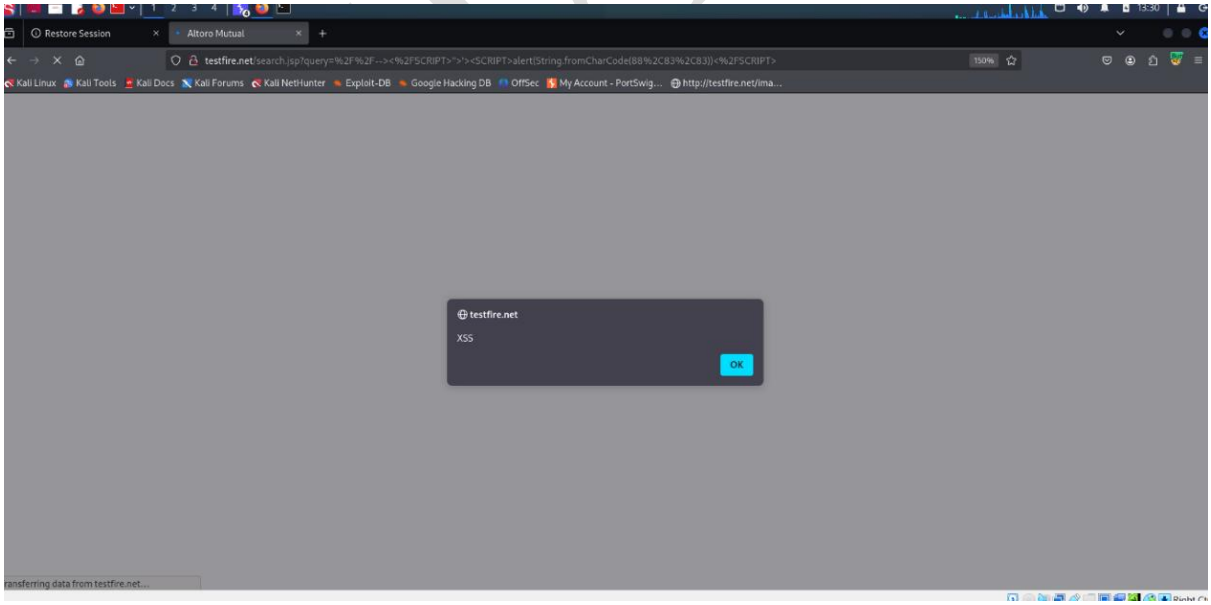


Figure 72

What Does the Pop-Up Mean?

When a pop-up window appears during XSS testing (commonly using alert('XSS')), it confirms that the injected JavaScript code has successfully executed in the victim's browser.
This pop-up acts as a Proof of Concept (PoC) demonstrating that the attacker can run arbitrary scripts within the context of the webpage.

Why Is This Serious?

The pop-up itself is harmless and is typically used only for testing. However, successful execution proves that the attacker can perform far more dangerous actions, such as:

- **Steal session cookies** → hijack user accounts
- **Modify webpage content** → display fake login forms (phishing)
- **Redirect users to malicious websites**
- **Perform actions as the logged-in user** → unauthorized transactions or data changes

---

Key Takeaway

When a pop-up appears during XSS testing:

- It confirms that **XSS vulnerability exists**
- It proves that **JavaScript execution is possible inside the webpage**
- It indicates that **malicious scripts could be used instead of simple alerts**, leading to serious security risks

Cross Site Scripting (XSS) Attack Using Burp Suite Repeater

How to do it -:

- Same process as you before used
- Search anything and click on go

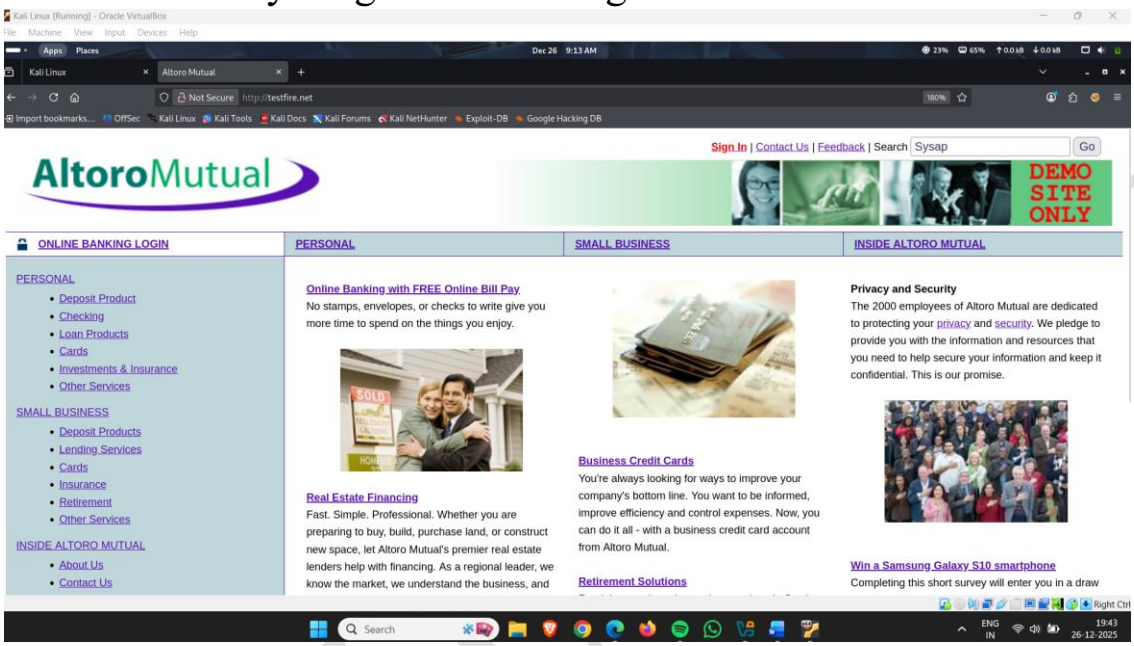


Figure 73

- Request Captured

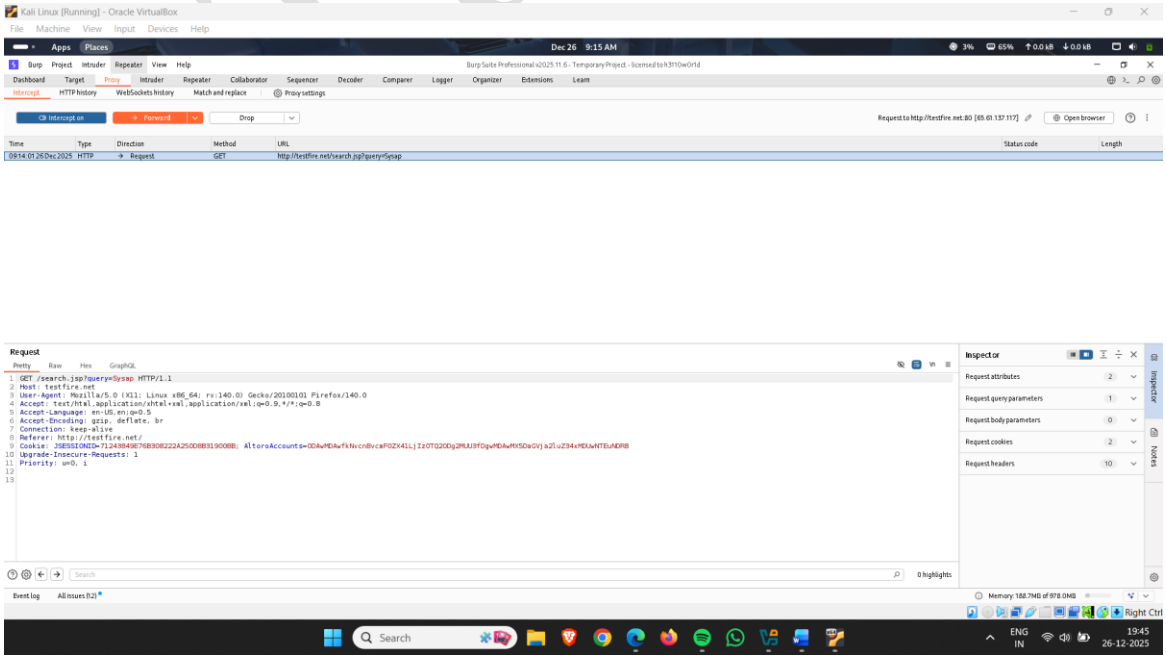


Figure 74

- Send request to repeater

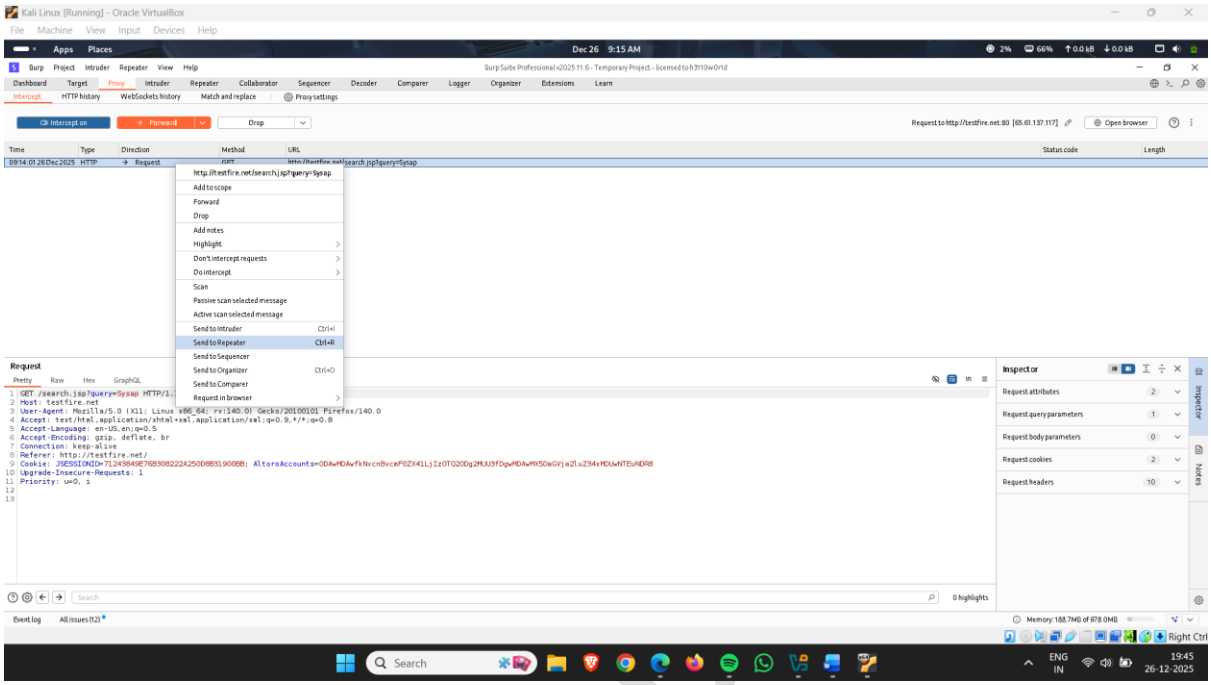


Figure 75

- In repeater, send the request to see the response of this request

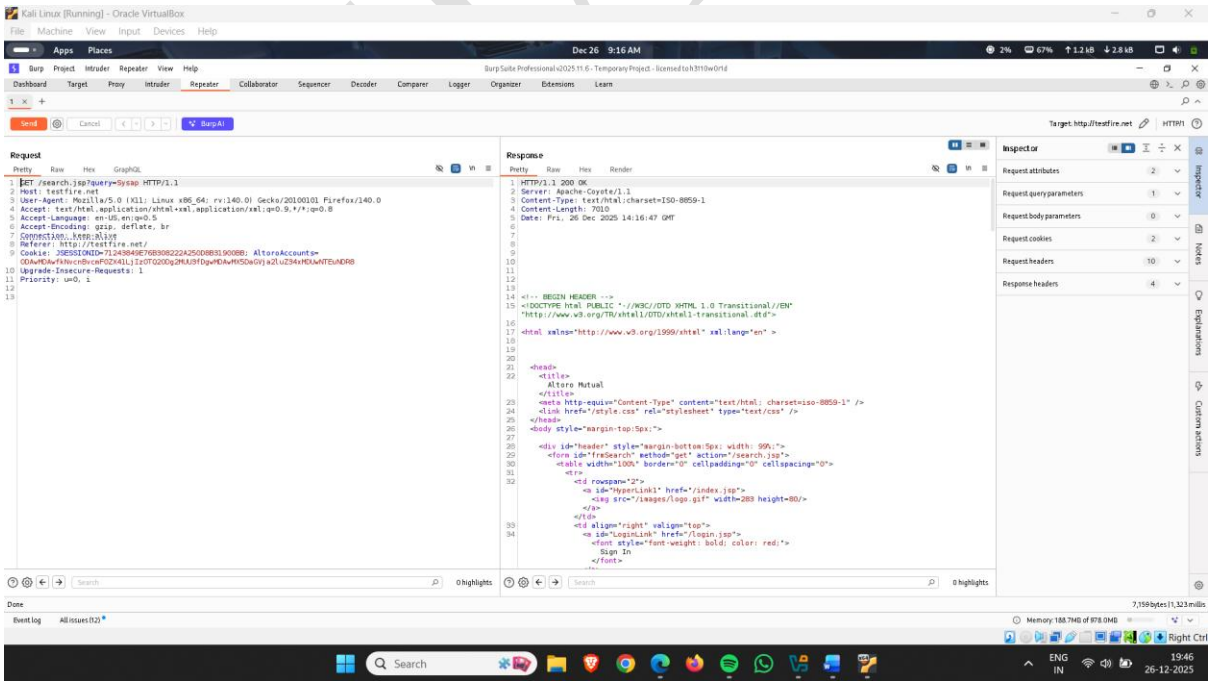


Figure 76

- Now, change the query parameter Sysap to the xss script

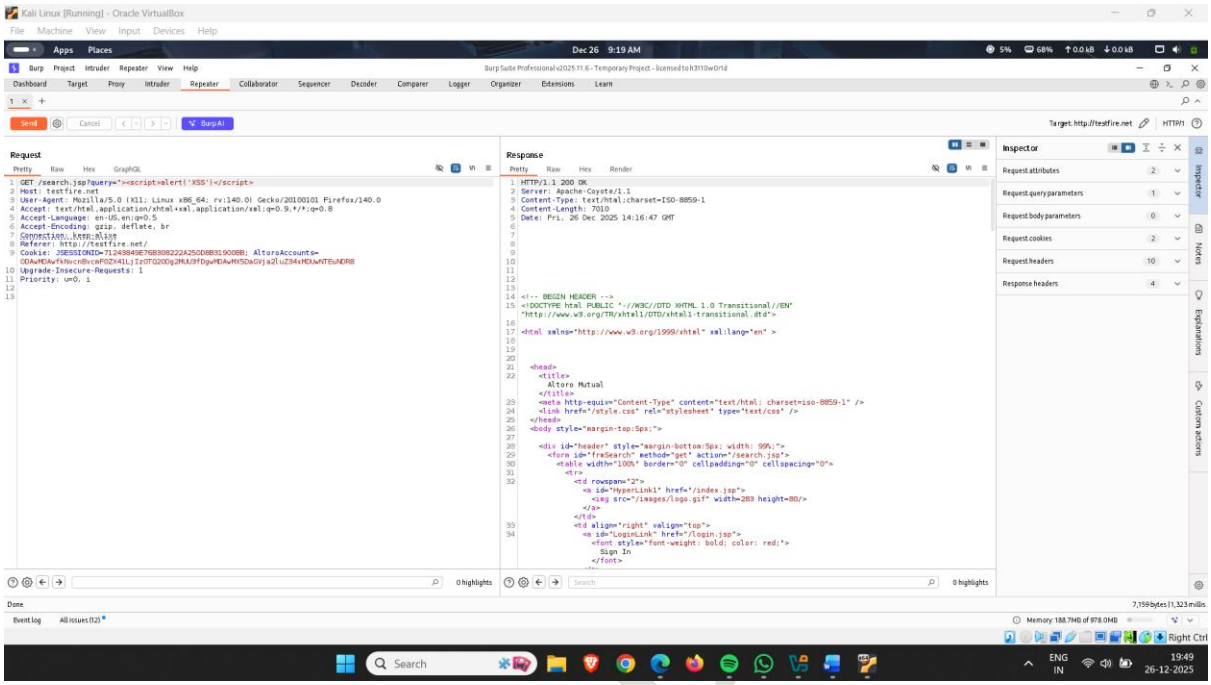


Figure 77

- Send this request

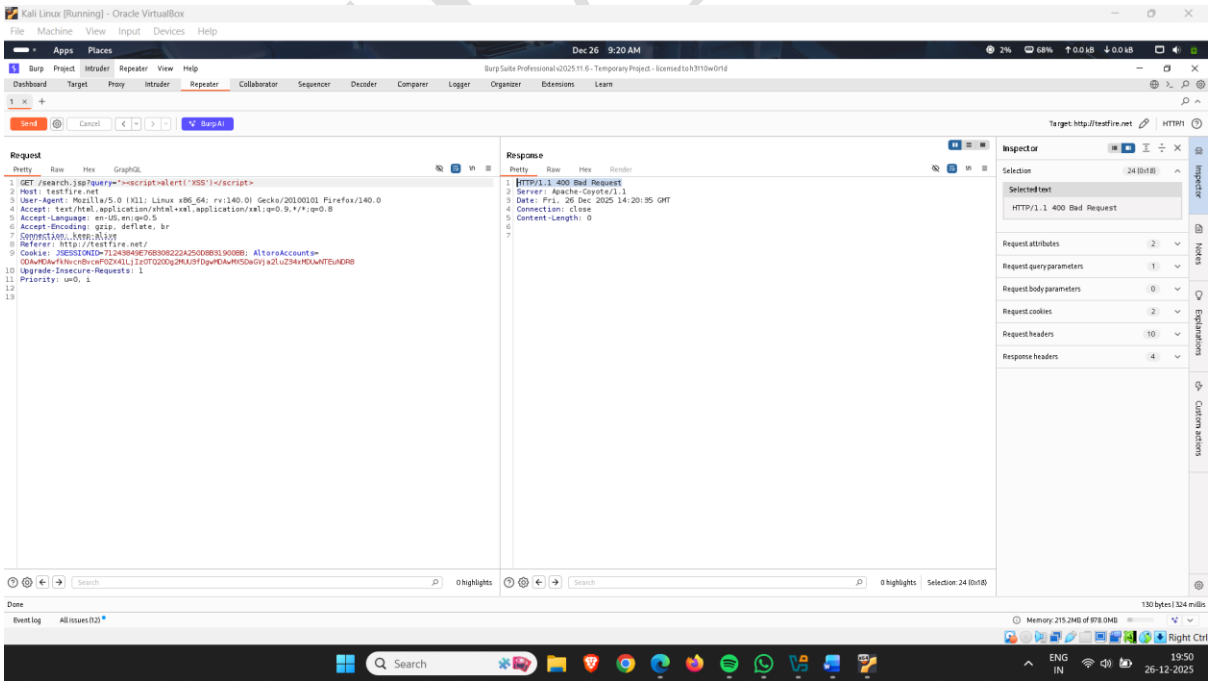


Figure 78

Response:
HTTP/1.1 400 Bad Request

• Status Code 400 means the server rejected your request because it was malformed or contained something it didn't accept.
• The server closed the connection and didn't process your request further.

What This Means:

• The testfire.net server likely has security controls that:
o Block requests with suspicious characters like < > ' ".

How You Can Try Again:

1. Try encoding the script:

/search.jsp?query=%3Cscript%3Ealert('XSS')%3C%2Fscript%3E

What is Encoding?

Encoding means replacing special characters with their safe ASCII representation so that the request can pass through filters.

Why Encoding? Why Do We Encode Scripts in XSS?

Encoding is very important in web security because many modern web applications and firewalls block special characters like:

• < > ' " /

These characters are directly related to HTML tags and JavaScript execution.

When you send them without encoding, the server may reject your request (like you saw: HTTP 400 Bad Request).

- Enter the encoding script & send it.

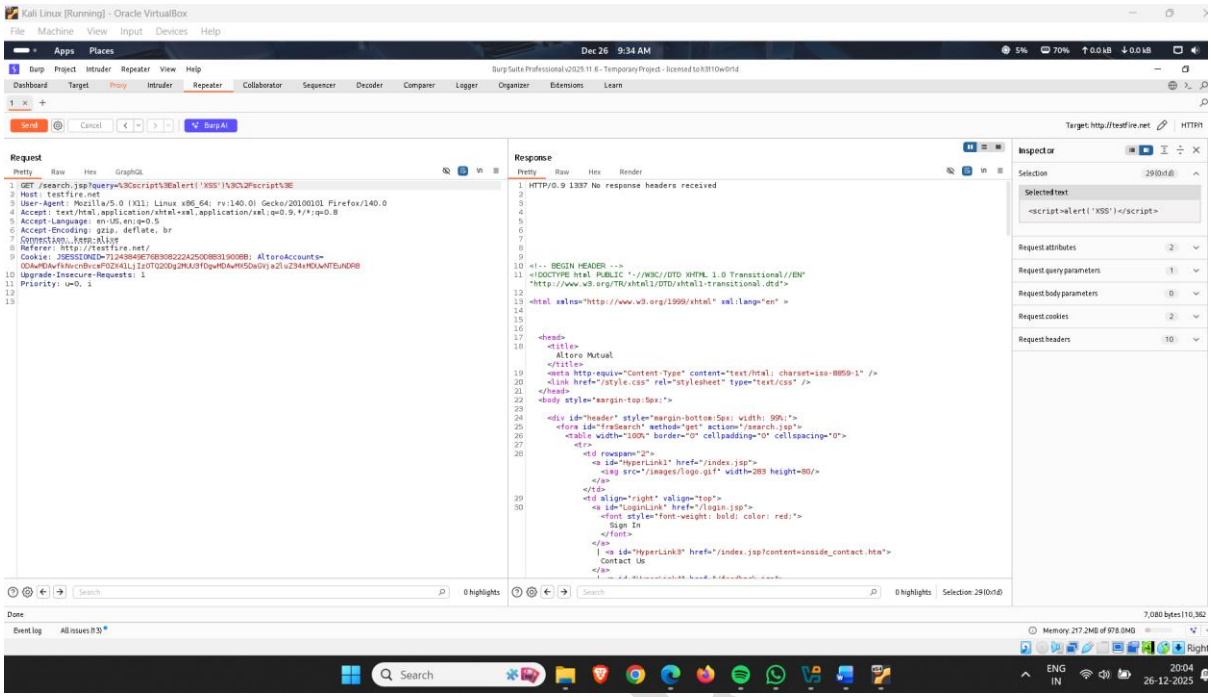


Figure 79

- Response received.

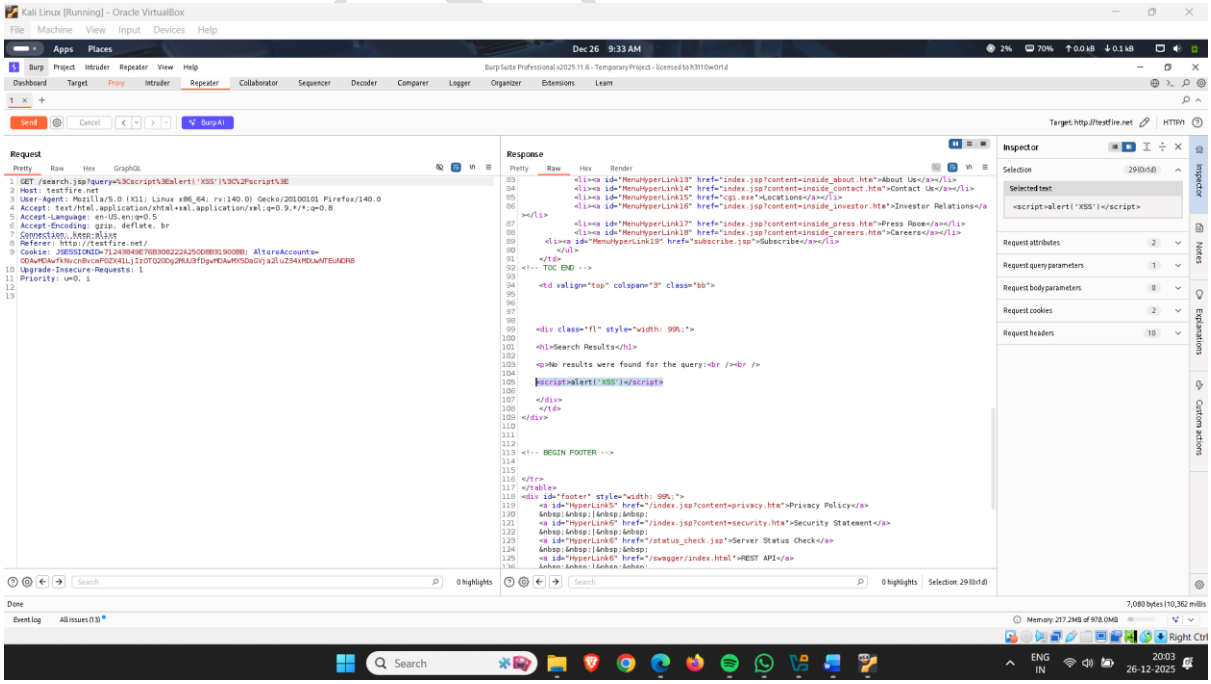


Figure 80

- Now check this script on browser
- Simply right click on request and copy URL

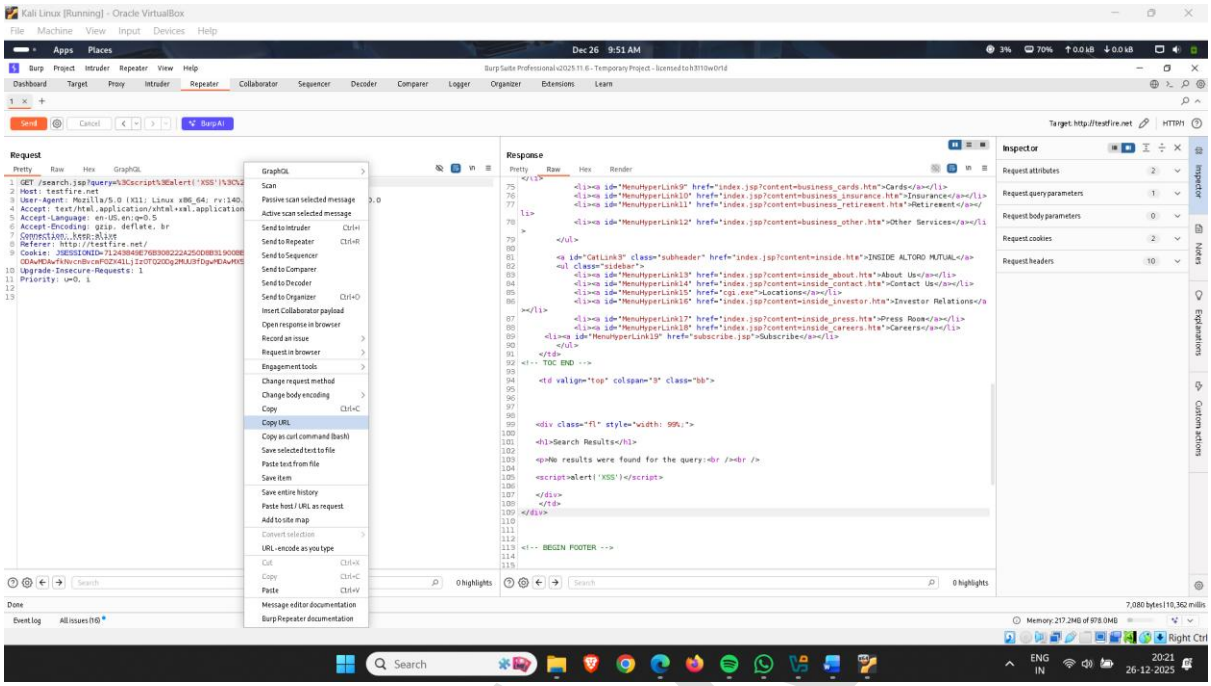


Figure 81

- Paste URL in browser URL section

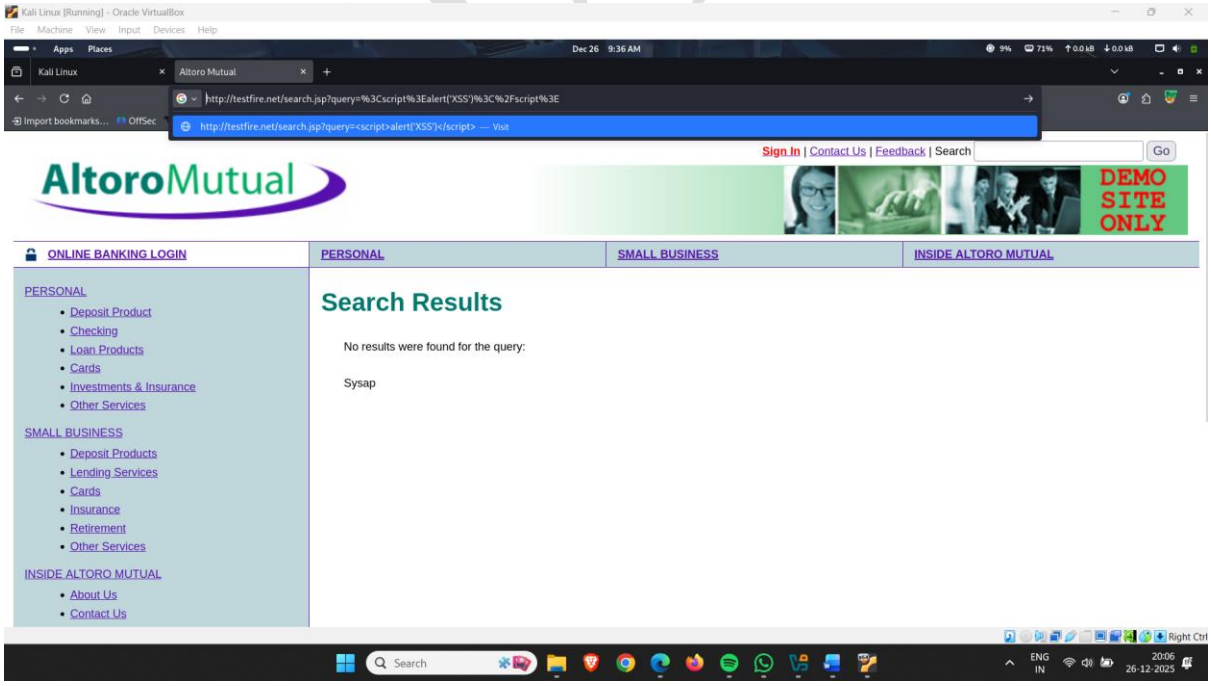
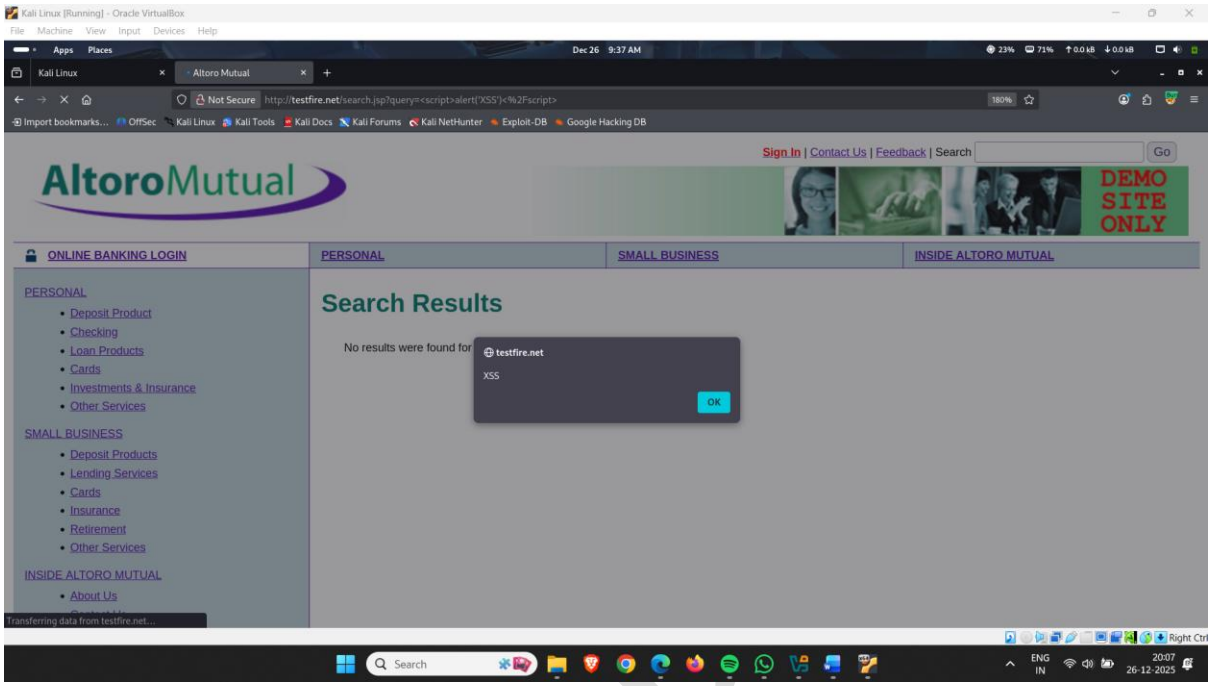


Figure 82

- Result



NOTE: Why the Alert Box Appears

- **The alert box confirms that JavaScript code was executed in the browser.**
- **This indicates a Reflected Cross-Site Scripting (XSS) vulnerability.**
- **Even though the page says, *"No results were found"*, the injected script still runs.**

How To Prevent from Web Application Attacks / Hacking

---

1. Input Validation and Sanitization

• Never trust user input.
• Validate and sanitize all forms, URLs, cookies, and API inputs.
• Block dangerous characters like:
' “; -- < > { } [ ]

---

2. Use Parameterized Queries

• Always use prepared statements to prevent SQL Injection.
• Example in PHP (safe way):

$stmt = $pdo->prepare ('SELECT * FROM users WHERE id = ?');

$stmt->execute([$id]);

---

3. Use Strong Authentication

• Enforce strong passwords.
• Use multi-factor authentication (MFA).
• Limit login attempts (brute force protection).
• Use secure session management (auto logout on inactivity).

---

4. Prevent Cross-Site Scripting (XSS)

• Always encode outputs (HTML, JavaScript, URLs).
• Use security headers like:
Content-Security-Policy: default-src 'self';
• Validate input to reject scripts like:
<script>alert('XSS’) </script>

5. Prevent Cross-Site Request Forgery (CSRF)

• Use unique CSRF tokens in all forms.
• Validate CSRF tokens on the server.
• Use Same Site cookie attribute.

---

6. Secure File Uploads

• Accept only allowed file types.
• Rename uploaded files.
• Store files outside the web root.
• Scan uploaded files for malware.

---

7. Use HTTPS Everywhere

• Use a valid SSL certificate.
• Force HTTPS using HTTP Strict Transport Security (HSTS).

---

8. Apply Proper Access Controls

• Restrict user permissions.
• Protect admin pages using strong authentication.
• Use role-based access.

---

9. Use Security Headers

Add these headers in your web server:
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
Strict-Transport-Security: max-age=31536000; includeSubdomains
Content-Security-Policy: default-src 'self';

---

10. Regular Security Testing

• Perform:
o Vulnerability scanning
o Penetration testing
o Code reviews

• Use tools like:
o Burp Suite
o OWASP ZAP
o SQL Map (for developer testing only)

---

11. Update and Patch Everything

• Keep:
o Servers
o Databases
o Frameworks
o CMS (like WordPress, Joomla)
Up to date and patched.

---

12. Deploy Web Application Firewall (WAF)

• Protects against SQLi, XSS, CSRF, brute force, etc.
• Filters malicious traffic.

---

13. Secure Error Handling

• Show generic error messages.
• Do not reveal server/database info.
• Log detailed errors on the backend only.

---

14. Follow OWASP Top 10 Best Practices

Always protect against:
• SQL Injection
• Broken Authentication
• Sensitive Data Exposure
• XSS
• CSRF
• Insecure Deserialization
• Security Misconfiguration
• Using Components with Known Vulnerabilitie

How To Prevent from Browser Exploitation

Keep Your Browser Fully Updated

BeEF often targets outdated browsers with unpatched security flaws.
• Always use the latest version of your browser.
• Enable automatic updates.

Disable or Control JavaScript Execution

BeEF needs JavaScript to run on the victim's browser.
• Use browser extensions like:
o NoScript (Firefox) – Blocks all JavaScript by default.
o ScriptSafe (Chrome) – Allows you to block scripts selectively.
• Only allow JavaScript from trusted websites.
• If possible, disable JavaScript on unknown sites.

Set a Strong Content Security Policy (CSP) for Your Websites

If you are a developer, you can stop BeEF hooks by blocking unauthorized scripts.
• Example CSP:
Content-Security-Policy: default-src 'self'; script-src 'self';
• This prevents external scripts (like BeEF's hook.js) from loading.

Use a Web Application Firewall (WAF)

For organizations or developers:
Deploy WAF to filter malicious scripts and XSS payloads.
• WAF helps stop attacks before they reach the browser.

Enable Strong Security Headers

These headers harden your browser's security posture:
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
Referrer-Policy: no-referrer
Content-Security-Policy: default-src 'self'; script-src 'self'
Strict-Transport-Security: max-age=31536000; includeSubDomains

---

Avoid Suspicious Links and Phishing Sites

BeEF attacks commonly start by tricking you to click on a malicious link.
• Best Practices:
o Don't click unknown links.
o Verify sender email addresses.
o Be careful when scanning QR codes.

Clear Browser Cache and Cookies Regularly

BeEF uses persistent sessions.
• Regularly clear:
o Cache
o Cookies
o Local storage

Use private browsing when visiting untrusted websites.

Block Third-Party Scripts and Trackers

BeEF often hides in external scripts.
• Use extensions like:
o uBlock Origin – Blocks ads, trackers, and malicious scripts.
o Privacy Badger – Blocks third-party tracking.

Use HTTPS Everywhere

Always use secure, encrypted connections.
• Install the HTTPS Everywhere extension to force HTTPS.
• Avoid using websites that still use HTTP.

Use Browser Sandboxing (If Available)

Some browsers (like Google Chrome) use sandboxing to isolate tabs.
• Always enable sandboxing to limit attack impact.

Avoid Browser Extensions from Untrusted Sources

Some extensions can be compromised or malicious.
• Only install extensions from official web stores.
• Review permissions carefully.
• Remove unused extensions.

User Awareness

BeEF often depends on social engineering.
• Stay aware:
o Don't open suspicious attachments.
o Don't trust pop-ups asking to enable JavaScript or download plugins.


1-==-3EL-1
Union Selarct
SAGL
Sagare
PROTCTION
ESIPLODATION
SQL INJECTION

SQL INJECTION

SQL Injection is a web security vulnerability that allows an attacker to interfere with the queries an application makes to its database.

In simple terms:

An attacker can inject malicious SQL commands into input fields to:

- View hidden data
- Bypass logins
- Modify or delete data
- Execute OS commands (in advanced cases)

Basic Example:

Suppose this URL:
• http://example.com/product.php?id=10

Normal SQL Query:
• SELECT * FROM products WHERE id = 10;

If attacker changes URL to:
• http://example.com/product.php?id=10 OR 1=1

Query becomes:
• SELECT * FROM products WHERE id = 10 OR 1=1;

This forces the query to return all products because 1=1 is always true.

SQL Injection Cheat Sheet:

Basic Tests:

' OR 1=1 --

" OR 1=1 --

' OR 'a'='a

" OR "a"="a

Order By Test:

' ORDER BY 1 --

' ORDER BY 2 --

' ORDER BY 100 -- (error if too high)

Union Injection:

' UNION SELECT null, username, password FROM users –

Boolean-Based Blind:

' AND 1=1 -- (True)

' AND 1=2 -- (False)

Time-Based Blind:

' OR IF(1=1, SLEEP(5), 0) -- (Delays page if true)

Perform Sql Injection Using Havij tool

Havij is an automated SQL Injection tool developed by ITSecTeam. It is a Graphical User Interface (GUI)-based tool that allows attackers and penetration testers to easily detect and exploit SQL injection vulnerabilities in web applications.

Key Features of Havij:

- **Fully Automated SQL Injection Tool**
- **Supports:**
    - Database Detection (MySQL, MSSQL, Oracle, PostgreSQL, MS Access, etc.)
    - o Database Dumping (Extract tables, columns, and data)
    - o Command Execution on the target OS (if the database supports it)
    - o Reading/Writing Files on the database server
    - o Admin Login Bypass (automatic)
- **Supports GET, POST, Cookies parameter testing.**
- **Can perform multi-threaded scanning for faster results.**
- **Graphical reports for extracted data.**

Where Havij is used (Legitimate Scenarios):

Environment	Purpose
Company security audit	Testing internal web apps for vulnerabilities
Bug bounty programs	Finding weaknesses legally for rewards
Academic cybersecurity labs	Demonstrating how SQLi works
Red-team engagements	Showing realistic attack paths
Defensive research	Understanding attack tools to build protections

Target Website:

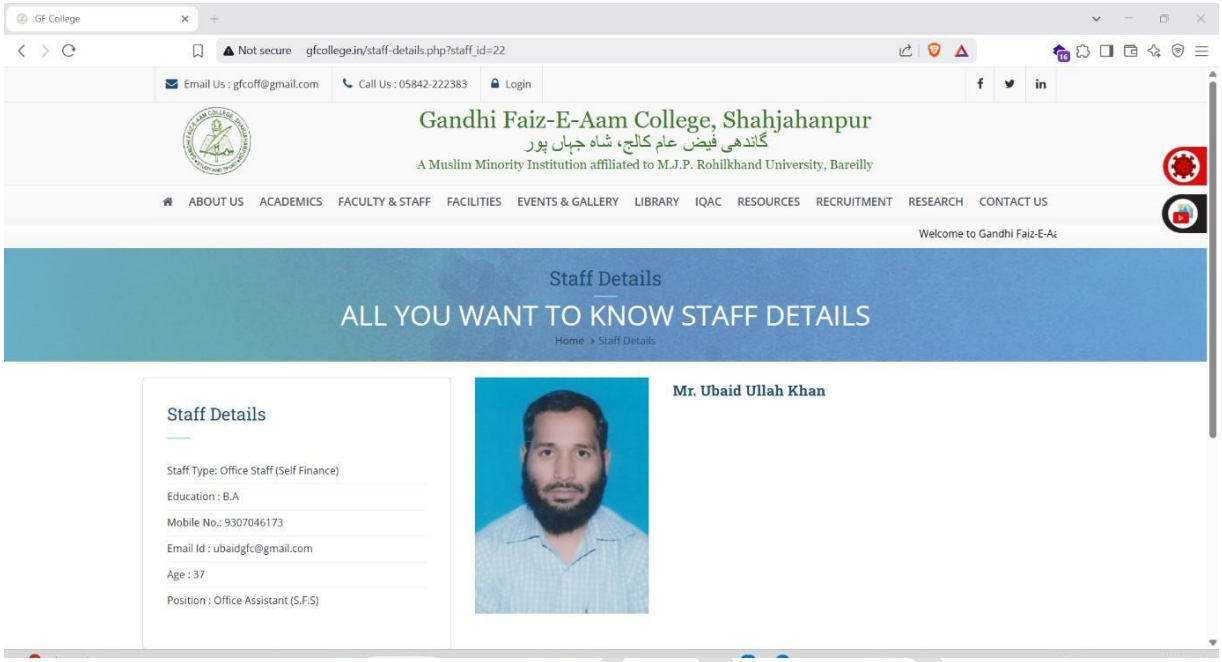


Figure 83

- Paste Url in target Section

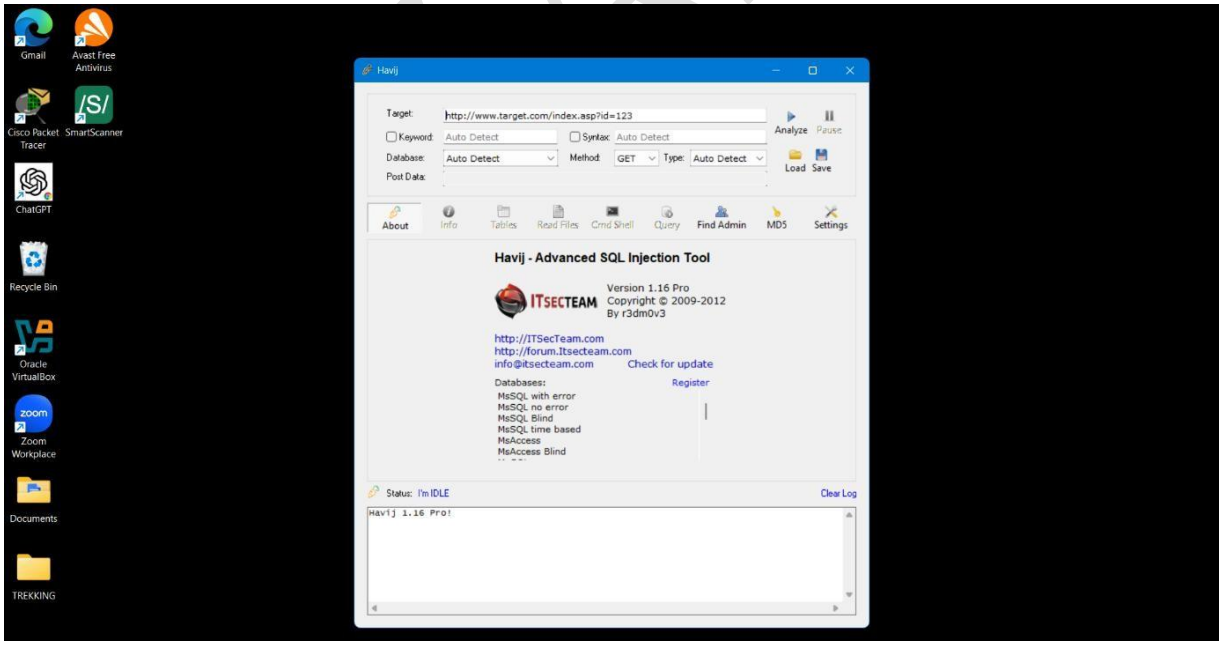


Figure 84

- Click on analyse button

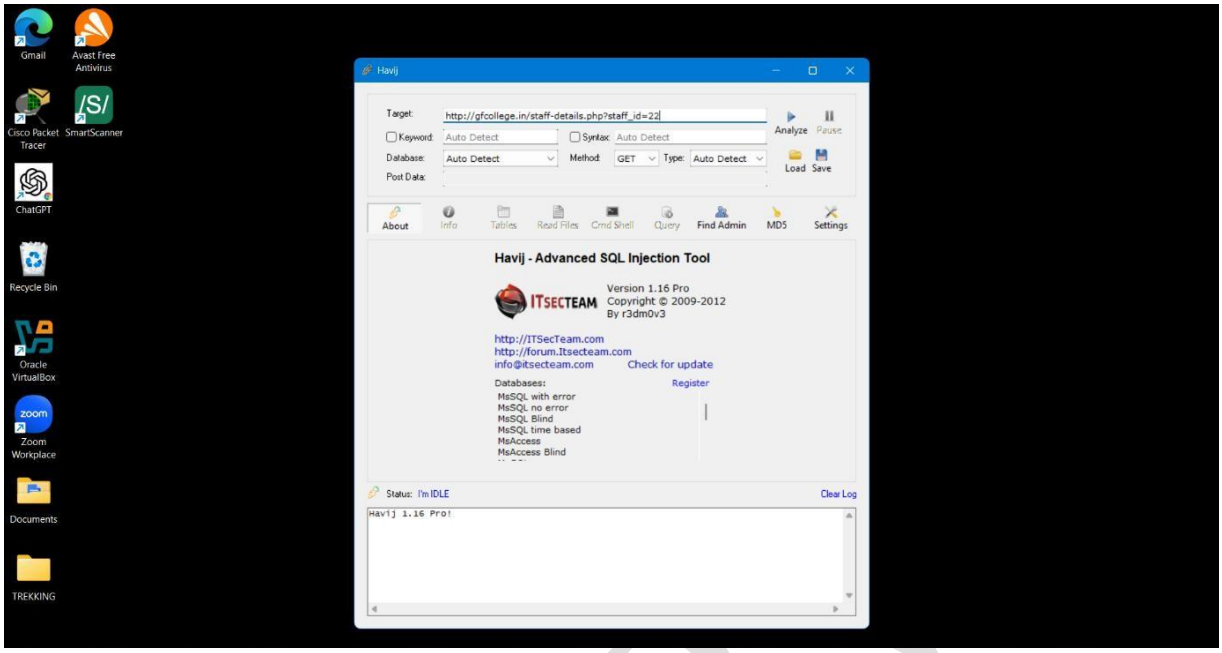


Figure 85

- Havij Started his working

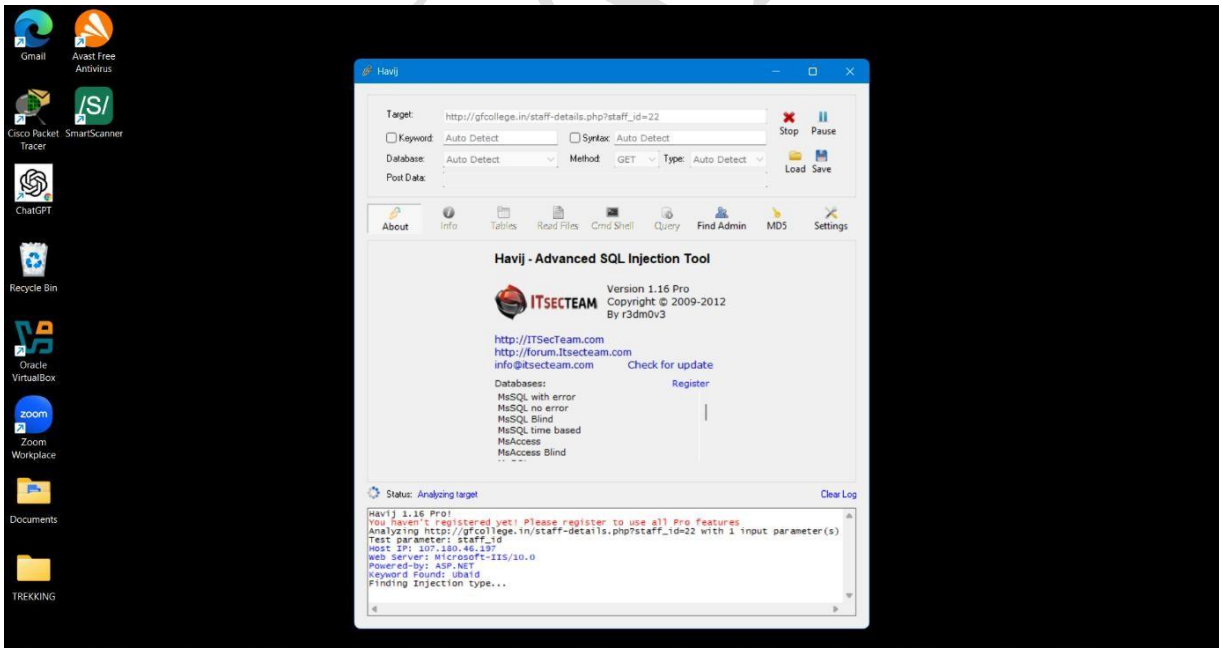


Figure 86

- Here, it finds Database of- gfcollege

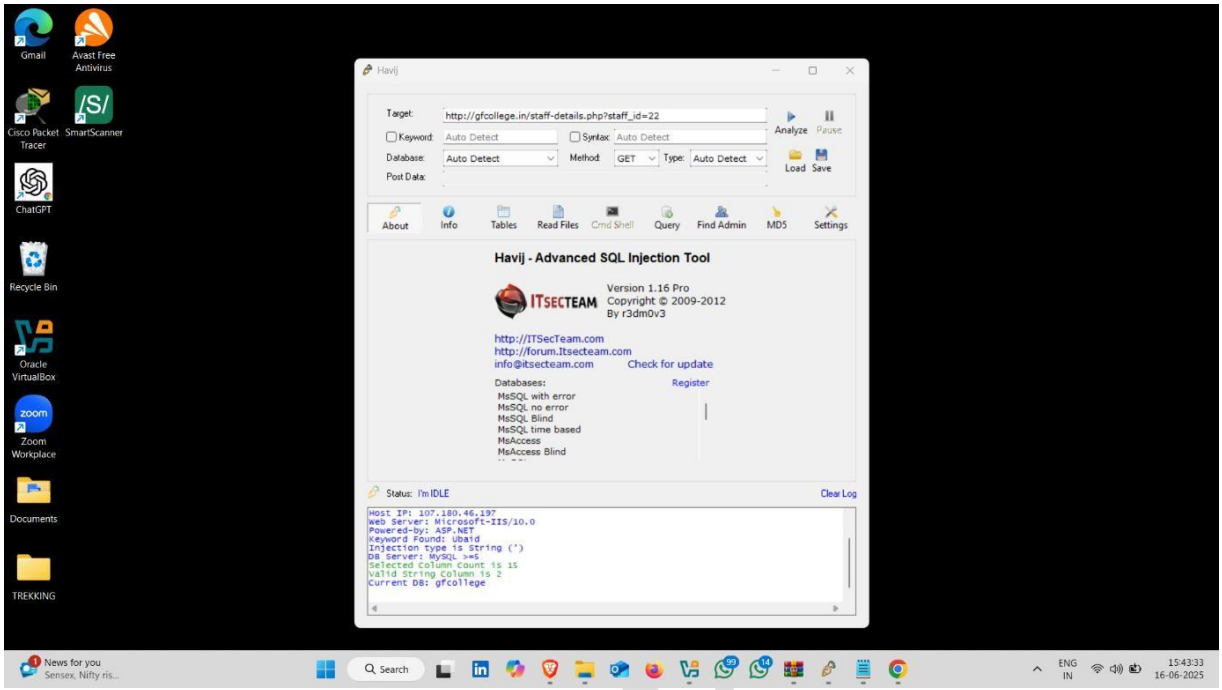


Figure 87

- Now click on Tables

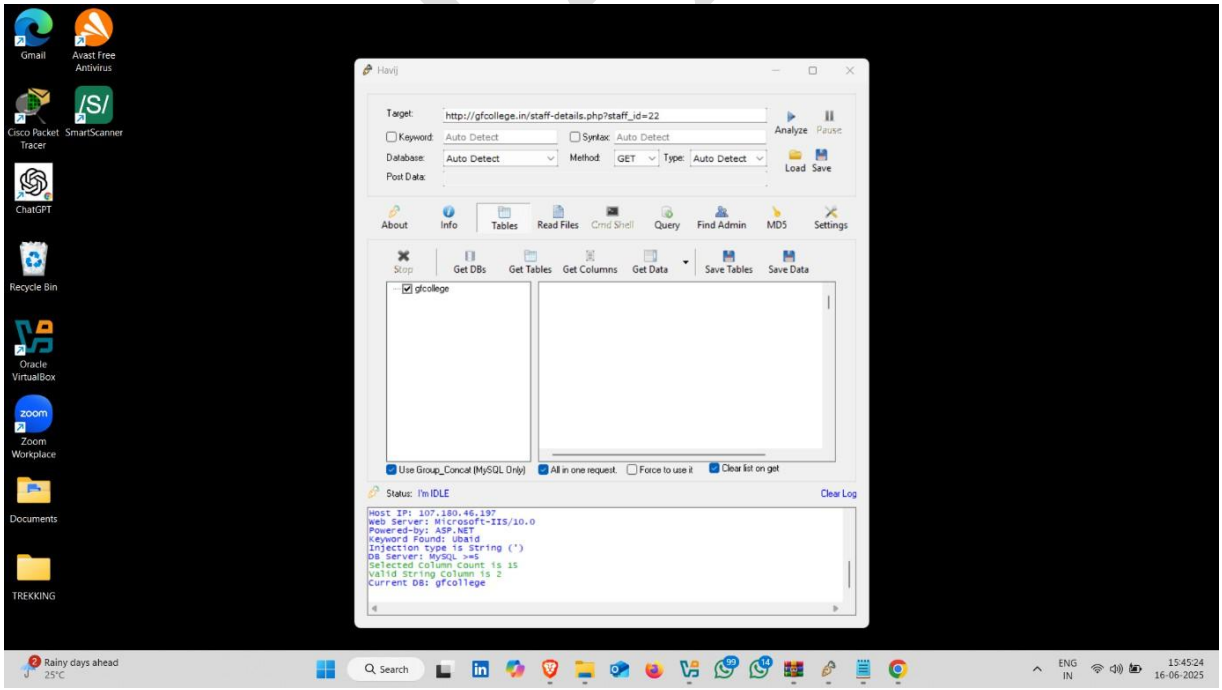


Figure 88

- Here it finds the tables

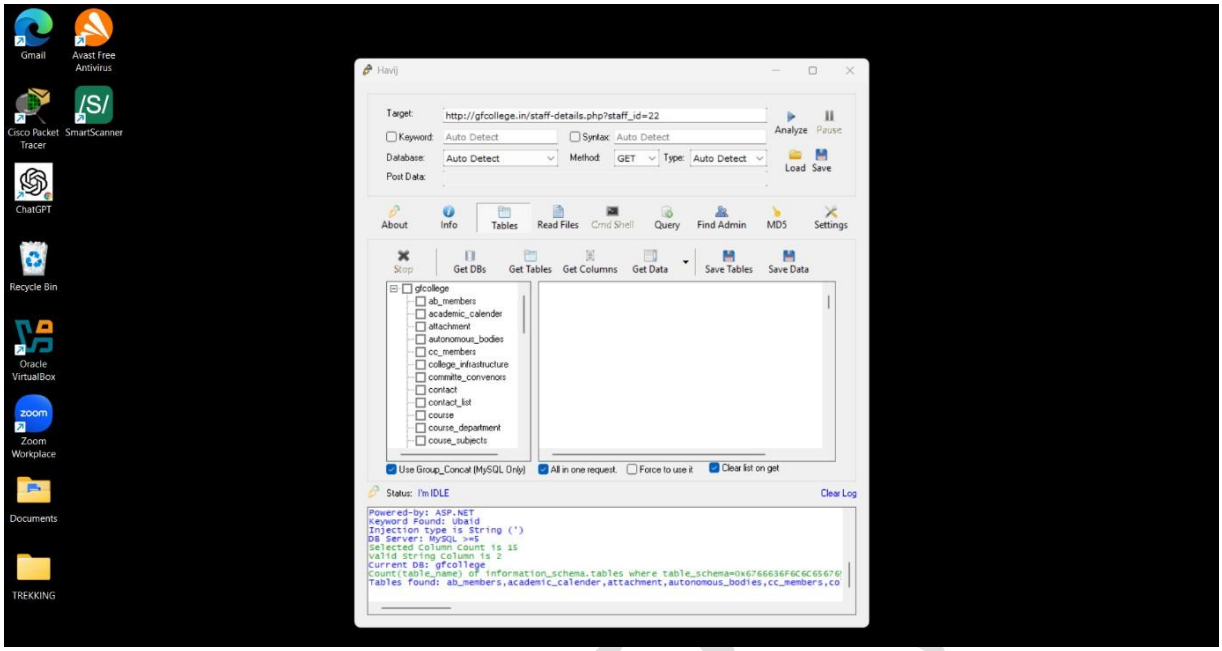


Figure 89

- Now select the tables and click on get columns

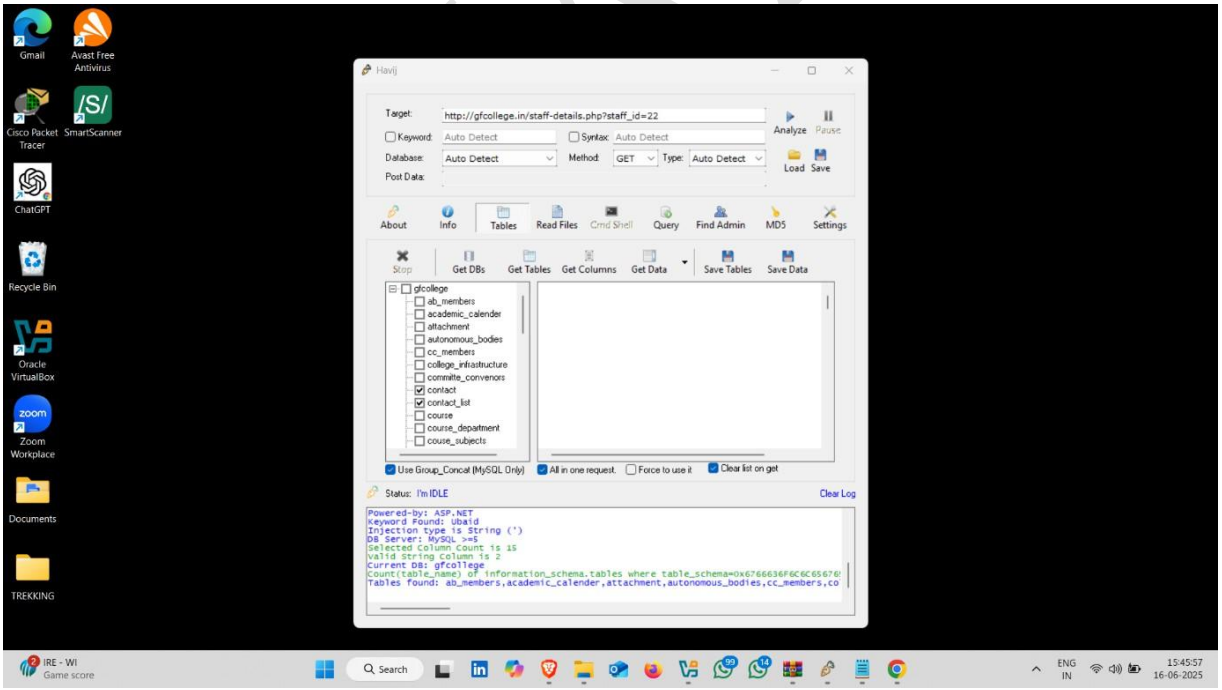


Figure 90

- It finds the columns

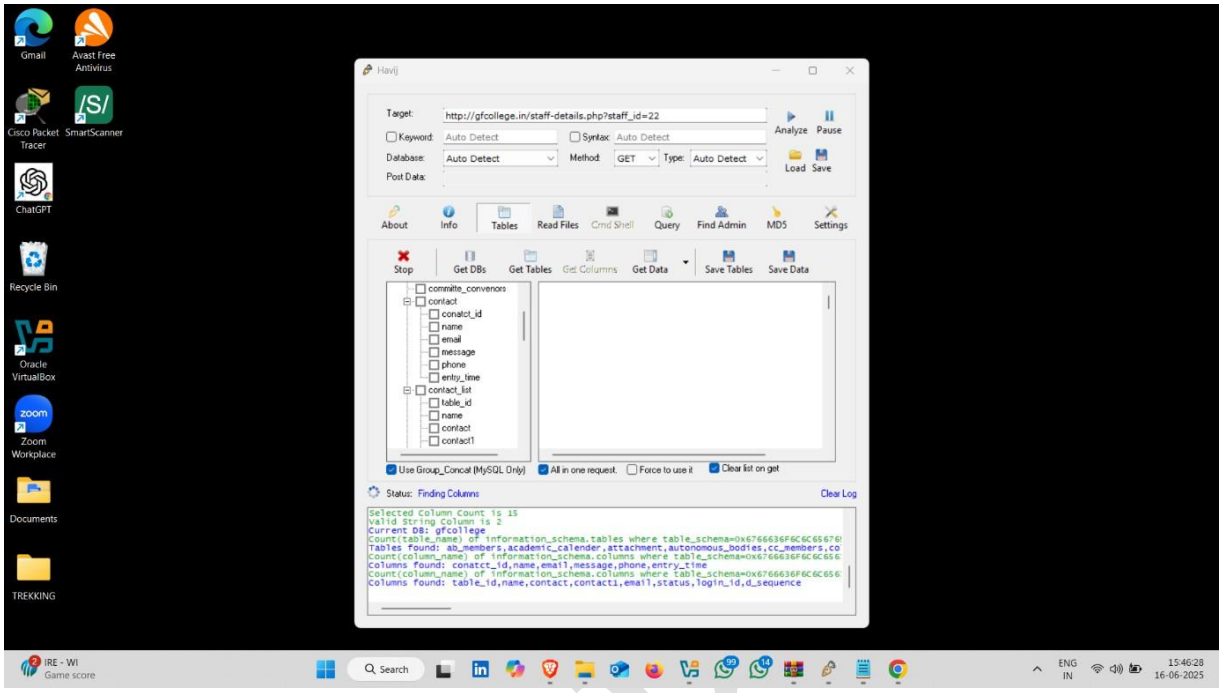


Figure 91

- Select the columns that you want to get data and click on get data option.
- You will get the data

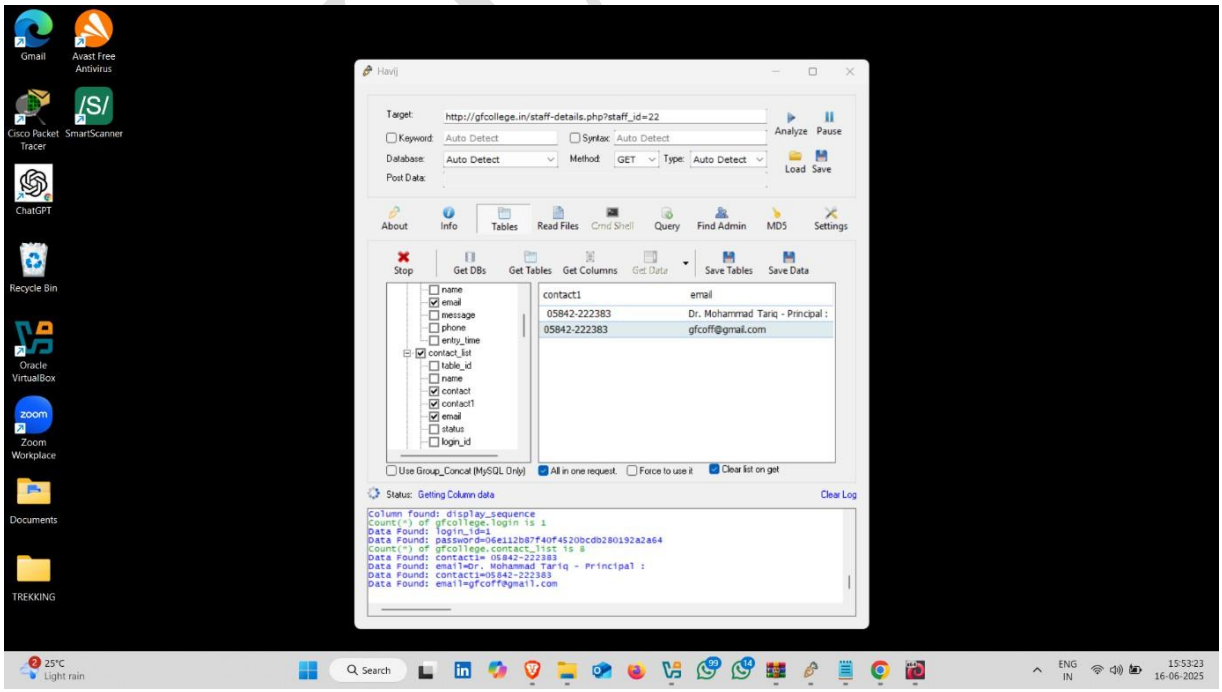


Figure 92

Perform SQL Injection Using SQLMap Tool

SQLMap is an open-source penetration testing tool that automates the process of detecting and exploiting SQL Injection (SQLi) vulnerabilities in web applications.

It can:
• Detect SQLi vulnerabilities.
• Extract sensitive database information.
• Perform database fingerprinting.
• Access the underlying file system.
• Execute remote commands on the database server (if possible).

How SQLMap Works:

Targeting:

SQLMap sends specially crafted payloads to URL parameters, POST data, cookies, or HTTP headers.

Detection:

It analyzes server responses to identify SQL Injection vulnerabilities.

Exploitation:

Once found, it can:

o Enumerate databases, tables, columns.

o Extract data like usernames, passwords.

o Read and write files on the server.

o Gain OS-level access (in advanced scenarios).

Key Features of SQLMap:

Supports GET, POST, Cookie, and HTTP Header injection.

• Automated database fingerprinting.

• Full database dumping (tables, columns, data).

• Password hash retrieval and cracking.

• File system access (read/write files).

Support for most DBMS: MySQL, Oracle, PostgreSQL, MSSQL, SQLite, etc.

• Bypasses WAF (Web Application Firewall) with tamper scripts.

• Supports time-based, error-based, union-based, Boolean-based, and stacked queries.

Primary Objectives of SQLMap

- Identify SQL Injection vulnerabilities
- Fingerprint databases and detect DBMS type
- Enumerate databases, tables, and columns
- Dump or extract sensitive data
- Access the server file system (if permissions allow)
- Execute system commands indirectly through the database engine

How to use it:

- Open kali linux terminal and type following command

Command -: sqlmap -u --random-agent

- sqlmap → Runs the SQLMap tool.
- -u → Specifies the target URL.
- <target website URL> → The website you want to test for SQL Injection.
- --random-agent → Uses a random browser identity to bypass detection.

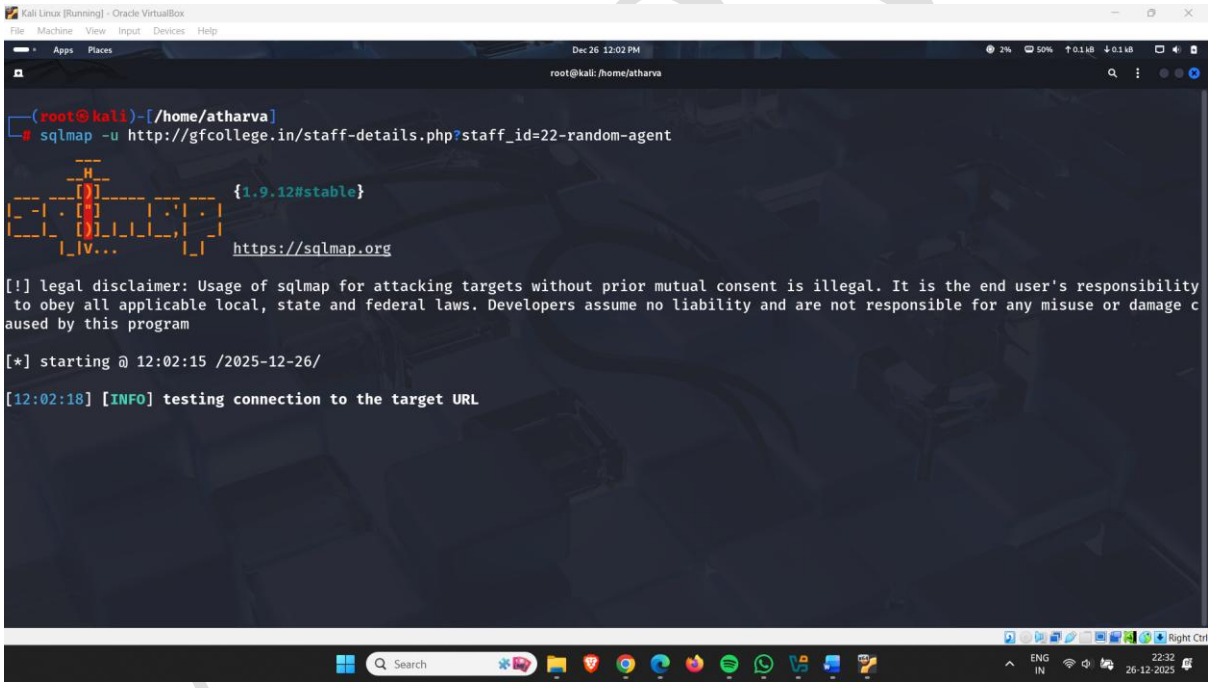


Figure 93

- Sqlmap started

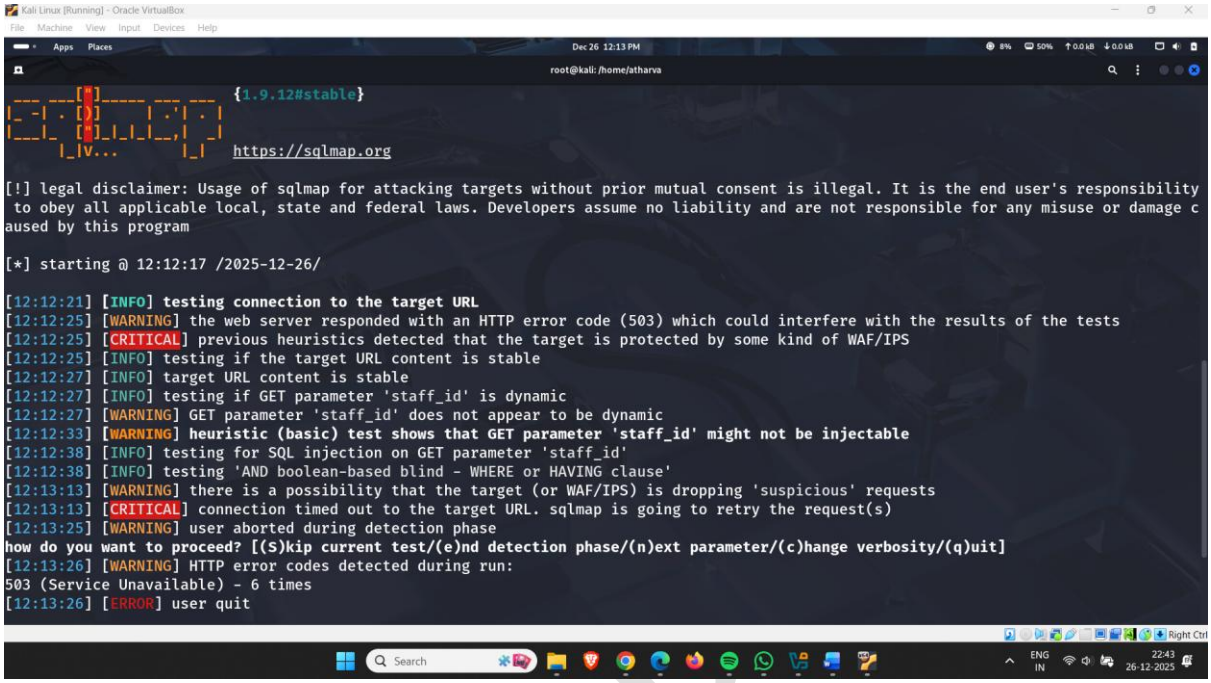


Figure 94

Sqlmap tried to check if this parameter is *SQL injectable*, but the target website is protected by a WAF / IPS and is returning HTTP 503 errors, so sqlmap cannot reliably detect an injection.

How to Prevent From SQL Injection Attacks

1️⃣ Use Parameterized Queries (Prepared Statements)

- Most important defense.
- Prepared statements separate SQL logic from user input.

Example (Safe Code):

$stmt = $pdo->prepare("SELECT * FROM users WHERE username = ?");

$stmt->execute([$username]);

User input is treated as data, not as part of the SQL command.

2️⃣ Use Stored Procedures (Safely)

- Stored procedures can help if used properly with parameterized inputs.

Example:

```
CREATE PROCEDURE GetUserByID (@UserID INT)
AS
BEGIN
  SELECT * FROM Users WHERE UserID = @UserID
END
```

---

3 Never Build SQL Queries with String Concatenation

Vulnerable Code:

$query = "SELECT * FROM users WHERE username = '" . $_POST['username'] . "'";

- Never directly insert user input into SQL queries.

---

4 Input Validation (Allow-List Input)

- Validate type, length, format, and range of user input.

Example:

- Phone Number: Only digits, max length 10.
- Username: Allow only alphanumeric characters.

---

5 Use ORM (Object Relational Mapping) Tools

- ORM tools like Hibernate, Sequelize, and Entity Framework automatically use parameterized queries.

---

6 Implement Least Privilege Database Access

- The database user account used by the application should have minimum required permissions.

Example:

- SELECT, INSERT only.
- No DROP, DELETE, or ALTER if not needed.

7️⃣ Use Web Application Firewalls (WAF)

- WAF can block common SQL Injection patterns.

Tools: AWS WAF, Cloudflare WAF, ModSecurity

8️⃣ Escape User Inputs (Last Layer of Defense)

- Escape dangerous characters (like ' " ; --) if parameterization is impossible.
- But this is not a primary defense — parameterization is better.

9️⃣ Error Handling and Logging

- Never show raw SQL errors to users.

Avoid:

- Displaying detailed SQL errors to users.

Do:

- Show generic error messages.
- Log detailed errors on the backend only.

🔟 Regular Security Testing

Perform:

- Penetration Testing
- Code Reviews
- Automated Scans (e.g., **OWASP ZAP, Burp Suite**)

1️⃣1️⃣ Follow OWASP SQL Injection Guidelines

- Reference: **OWASP SQL Injection Prevention Cheat Sheetsss**