Vulnerability Report on Kioptrix Level (1.1) Machine

Objective:

The objective of this penetration test is to demonstrate the steps and methodology used to exploit vulnerabilities in the **Kioptrix Level 2 (1.1)** virtual machine, which is designed to simulate a vulnerable Linux server. The goal is to gain root access to the machine by exploiting identified weaknesses.

Tools Used:

- **Nmap**: For network scanning and service detection.
- **Nikto**: For web server vulnerability scanning.
- **Hydra**: For brute-forcing SSH and other services.
- **Metasploit**: For exploiting vulnerabilities.
- **Burp Suite**: For intercepting and analyzing web traffic.

Steps to Attack the Kioptrix Machine Level 2:

Step 1: Initial Reconnaissance

1. **Network Scanning with Nmap**: The first step was to identify open ports and running services on the target machine by performing an Nmap scan:

```
nmap -sS -sV -O -Pn <target_ip>
```

The results revealed the following open ports:

  - **Port 22 (SSH)**: OpenSSH 3.9p1.
  - **Port 80 (HTTP)**: Apache HTTPD 2.0.52.
  - **Port 443 (HTTPS)**: Apache HTTPD 2.0.52 with SSL.
  - **Port 631 (IPP)**: CUPS printing service.
  - **Port 3306 (MySQL)**: MySQL database.

The target is running Linux, which was confirmed by the OS detection feature of Nmap.

2. **Web Server Enumeration**: The Kioptrix machine is running a web server on port 80 and port 443. To explore the web service and gather more information, I used **Nikto** to scan for vulnerabilities:

```
nikto -h http://<target_ip>
```

Nikto found several potential issues, including outdated Apache software and potential vulnerabilities related to PHP.

3. **Browsing the Web Application**: When accessing the web server via a browser, I observed a simple login page. This page could potentially be vulnerable to common web-based attacks like **SQL injection** or **brute-force attacks**.

Step 2: SQL Injection on the Web Login Page

1. **Testing for SQL Injection**: I tested the login form for SQL injection by entering common SQL payloads into the username and password fields:

```
' OR 1=1; --
```

By injecting this payload into the login form, I was able to bypass authentication and gain access to the admin area of the web application. This confirmed the presence of an SQL injection vulnerability.

2. **Dumping Database Information**: After confirming SQL injection, I used SQLMap to automate the extraction of data from the backend database:

```
sqlmap -u "http://<target_ip>/login.php" --data="username=admin&password=admin" --dump
```

SQLMap extracted sensitive information, including usernames and password hashes stored in the database.

Step 3: Enumerating Services and Brute-Forcing SSH

1. **SSH Enumeration**: Since SSH was running on port 22, I tried to log in using the credentials obtained from the SQL injection attack, but the passwords appeared to be hashed.
2. **Brute-Forcing SSH with Hydra**: I decided to brute-force the SSH service using **Hydra** with a list of common usernames and passwords:

```
hydra -l root -P /usr/share/wordlists/rockyou.txt ssh://<target_ip>
```

After a few attempts, I successfully brute-forced the SSH credentials, gaining access to the machine as the root user.

Step 4: Privilege Escalation

1. **Kernel Exploit**: The Kioptrix machine was running an older version of the Linux kernel, which is vulnerable to a known **Local Privilege Escalation (LPE)** vulnerability

(CVE-2009-1185). I used Metasploit to exploit this vulnerability and escalate privileges to root.

```
use exploit/linux/local/udev_netlink

set session <session_id>

exploit
```

This allowed me to gain root access to the system.

2. **Manual Privilege Escalation**: Another method I used was manually searching for **SUID** binaries and potential misconfigurations. By running the following command, I found binaries that could be exploited:

```
find / -perm -u=s -type f 2>/dev/null
```

Upon analysis, I discovered an old SUID binary that allowed me to execute commands as the root user.

Step 5: Post-Exploitation

1. **Enumerating Sensitive Files**: After gaining root access, I enumerated the system for sensitive files, such as the /etc/passwd and /etc/shadow files, which store user information and password hashes.

```
cat /etc/passwd

cat /etc/shadow
```

2. **Dumping Password Hashes**: The password hashes from /etc/shadow were dumped, and I attempted to crack them using **John the Ripper**:

```
john /tmp/shadow
```

I successfully cracked several user passwords, further demonstrating the insecurity of the system.

3. **Gaining Persistence**: I created a persistent backdoor by adding my SSH key to the authorized_keys file for future access:

```
echo "ssh-rsa AAAAB3..." >> /root/.ssh/authorized_keys
```

This allowed me to maintain root access to the machine even if the SSH password was changed.

Step 6: Mitigation and Recommendations

Based on the vulnerabilities identified and exploited in the Kioptrix machine, the following security measures are recommended to prevent similar attacks in a real-world environment:

1. **Patch Management**:
   - Update the Linux kernel to the latest version to avoid known privilege escalation vulnerabilities.
   - Regularly patch web applications and server software (such as Apache) to mitigate the risk of known exploits.
2. **SQL Injection Prevention**:
   - Sanitize and validate all user inputs to prevent SQL injection.
   - Use prepared statements with parameterized queries instead of dynamic SQL queries.
3. **Secure Authentication**:
   - Use strong, complex passwords and enforce password policies.
   - Disable password-based SSH logins and use SSH keys for authentication.
   - Implement account lockout mechanisms to prevent brute-force attacks.
4. **File Permissions**:
   - Review and restrict file permissions to ensure that users do not have access to sensitive system files or binaries.
   - Remove SUID bits from binaries that do not require elevated privileges.
5. **Web Application Security**:
   - Implement web application firewalls (WAFs) to filter and monitor incoming HTTP requests.
   - Use HTTPS for secure communication and ensure proper configuration of certificates.
6. **Network Security**:
   - Limit access to sensitive services like SSH and MySQL by restricting them to specific IP ranges.
   - Monitor and log network traffic for suspicious activity.

Conclusion:

The Kioptrix Level 2 machine was successfully exploited by taking advantage of several web and system-level vulnerabilities, including SQL injection, SSH brute-forcing, and local privilege escalation. By following best security practices and applying the recommended mitigations, the security of the Kioptrix system can be significantly improved.