

每天一个Linux命令

Curl

基于URL 规则进行文件传输工作

curl 参数 网址 URL 文件名

-a	追加写入到指定文件
-A	设置用户代理标头信息
-b	设置用户 Cookie 信息
-B	使用 ASCII 文本传输
-C	支持断点续传
-d	以 HTTP POST 方式传送数据
-D	把头部信息写入指定文件
-e	设置来源网址 URL
-f	连接失败时不显示报错
-o	设置新的本地文件名
-a	追加写入到指定文件
-O	保留远程文件的原始名
-G	以 GET 方式传送数据
-H	自定义头信息
-I	显示网站的响应头信息
-K	读取指定配置文件

- 获取指定网站的网页源码： curl https://www.linuxcool.com
- 下载指定网站中的文件： curl -O https://www.linuxprobe.com/docs/LinuxProbe.pdf
- 打印指定网站的 HTTP 响应头信息： curl -I https://www.linuxcool.com

```
bk201@LAPTOP-QK5GU6SJ:/mnt/c/Users/lenovo$ curl -I https://www.linuxcool.com
HTTP/2 200
date: Fri, 05 Jan 2024 14:45:51 GMT
content-type: text/html; charset=UTF-8
vary: Accept-Encoding
x-powered-by: PHP/7.4.33
link: <https://www.linuxcool.com/wp-json/>; rel="https://api.w.org/"
cf-cache-status: DYNAMIC
report-to: {"endpoints":[{"url":"https://a.nel.cloudflare.com/report/v3?s=lKy0Sb4UKsNCfPrVhIVsK83t1qavYWLrkfkBoFodl14Wi0z7ULY83%2Fnd06S%2F%2Fa3Ti96r93Ff4qcTkMsTcWK22JH91WlpuqgSW0tfwk7lLt9zf4a2Mrt18zdqa1%2BcjJm3mWFGXg%3D%3D"}],"group":"cf-nel","max_age":604800}
nel: {"success_fraction":0,"report_to":"cf-nel","max_age":604800}
server: cloudflare
cf-ray: 840c81dddca6c36c-SEA
alt-svc: h3=":443"; ma=86400
```

- 下载指定文件服务器中的文件（用户名:密码): `curl -u linuxprobe:redhat ftp://www.linuxcool.com/LinuxProbe.pdf`

Netcat

`nc` 参数 域名或 IP 地址

-g	设置路由器通信网关	-r	设置本地与远程主机的端口
-h	显示帮助信息	-s	设置本地主机送出数据包的 IP 地址
-i	设置时间间隔	-u	使用 UDP 传输协议
-l	使用监听模式	-v	显示执行过程详细信息
-n	使用 IP 地址，而不是域名	-w	设置等待连线的时间
-o	设置文件名	-z	使用输入或输出模式
-p	设置本地主机使用的端口		

- 扫描指定主机的端口（默认为 TCP）：`nc -nvv 192.168.10.10 80` / `nc -z -v -w 1 google.com 442-444`
- 扫描指定主机的 1~1000 端口，指定为 UDP：`nc -u -z -w2 192.168.10.10 1-1000`

Creating a Client-Server Setup

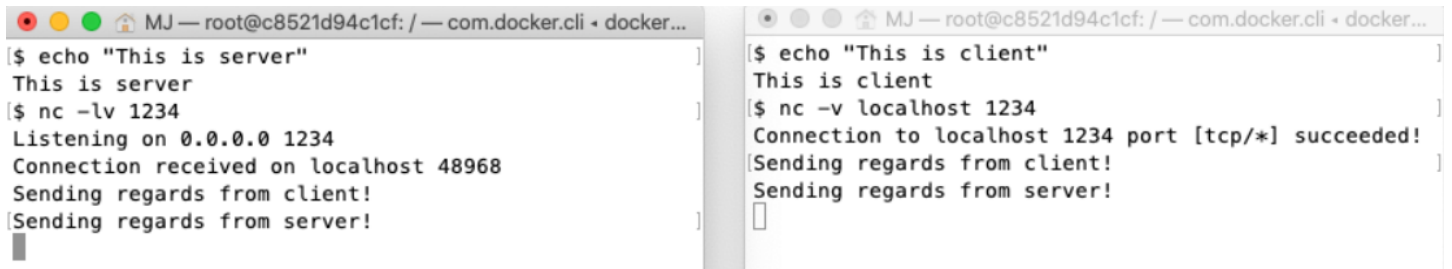
打开一个终端会话启动一个 netcat 服务器进程, 一旦执行, 进程将无限期监听, 直到被杀死。

```
1 $nc -lv 1234
2 Listening on 0.0.0.0 1234
```

打开一个新的终端会话，连接到服务器进程

```
1 $ nc -v localhost 1234
2 Connection to localhost 1234 [tcp/*] succeeded
```

在客户端进程中，可以输入一些文本，然后按回车键，马上就能在服务器进程的标准输出上看到准确的文本。同样，也可以从服务器进程向客户端进程发送内容。



```
MJ — root@c8521d94c1cf: / — com.docker.cli • docker...
$ echo "This is server"
This is server
$ nc -lv 1234
Listening on 0.0.0.0 1234
Connection received on localhost 48968
Sending regards from client!
Sending regards from server!

MJ — root@c8521d94c1cf: / — com.docker.cli • docker...
$ echo "This is client"
This is client
$ nc -v localhost 1234
Connection to localhost 1234 port [tcp/*] succeeded!
Sending regards from client!
Sending regards from server!

```

默认情况下，只要连接终止，服务器和客户端 netcat 进程都会返回。我们可以使用 -k 标志来保持服务器 netcat 进程

```
1 $nc -l -v -k localhost 1234
```

Setting up a Minimal Web Server

创建一个 index.html 文件，服务器将向所有连接的客户端提供该文件

```
1 $cat - > index.html <<<EOF
2 <!DOCTYPE html>
3 <html>
4     <head>
5         <title>Simple Netcat Server</title>
6     </head>
7     <body>
8         <h1>Welcome to simple netcat server!</h1>
9     </body>
10    </body>
11 <html>
12 EOF
```

`cat - > index.html << index.html`，它可以用于多行输入，也可以用于传递多行字符串给其他命令或文件，如果想输入多行需要使用 `cat > index.html <<EOF` 这种语法，它表示从标准输入读取一段文本，直到遇到 `EOF` 为止

然后启动一个 netcat 进程，该进程监听 1234 端口，并在客户端连接到服务器时提供文件

```
1 $echo -e "HTTP/1.1 200 OK\n\n$(cat index.html)" | nc -l 1234
```

打开浏览器，访问 localhost:1234 即可看到 index.html

但是这个服务器有两个问题：

- 即使数据传输完成，连接也不会终止
- 服务器只为单个 client 提供服务

可以使用 `-w` 标志可以指定超时值。上面的命令只要连接空闲时间超过一秒，就会被终止。然后将命令包装成一个 `while` 循环。这样，每当命令终止时，就会重启进程。具体来说，只要有 client 连接到 server，netcat 进程就会返回 HTTP 响应。空闲一秒后，进程终止并返回。最后，`while` 循环将重启进程，再次启动在 1234 端口监听的 netcat 进程。

Reverse Proxy With *netcat*

假设有一个服务正在监听 4321 端口。但是，外部流量只能通过 1234 端口访问主机。通过 netcat，我们可以设置一个反向代理，将流量从 1234 端口重定向到 4321 端口，反之亦然。

创建一个 named pipe：

```
1 $mkfifo /tmp/rp
```

然后创建反向代理

```
1 $nc -lv 1234 < /tmp/rp | nc localhost 4321 > /tmp/rp
```

通过该命令，我们创建了两个 netcat 进程。第一个进程称为 external router，第二个进程称为 internal router。

当 1234 端口有 incoming traffic 时，external router pipes the traffic to the internal router.

另一方面，当有来自 4321 端口的 outgoing traffic 时，the internal router will pipe it to the named pipe `/tmp/rp`. Then, the external router will read and send the content of `/tmp/rp` to the client.

Encryption

用 OpenSSL 加密一个文件，然后从服务器 A 发送：

```
1 $ openssl enc -aes-256-cbc -salt -in sensitive_file.txt -out encrypted_file.enc
2 $ nc -v -w 2 192.168.1.101 12345 < encrypted_file.enc
```

然后，接收端的服务器 B 必须使用 OpenSSL 接收并解密文件：

```
1 $ nc -l -p 12345 | openssl enc -aes-256-cbc -d -salt -out received_data.txt
```

iperf3

iperf 是一种网络吞吐量测量工具，可以测试 UDP 或 TCP 的吞吐量。我们还可以用它来验证 UDP 的连接性。需要同时建立客户端和服务端才能使用它。

首先，我们在服务器端使用 -s（服务器）选项启动 iperf：

```
1 $ iperf3 -s
2 -----
3 Server listening on 5201
4 -----
```

然后需要在客户端启动该工具，目标是服务器的 IP。我们使用 -u 来指定 UDP，而 -c 选项则表示这是客户端

```
1 $ iperf3 -u -c 172.16.38.137
2 Connecting to host 172.16.38.137, port 5201
3 [ 4] local 172.16.38.136 port 38369 connected to 172.16.38.137 port 5201
4 [ ID] Interval          Transfer    Bandwidth      Total Datagrams
5 [ 4]  0.00-1.00 sec    120 KBytes  983 Kbits/sec    15
6 [ 4]  1.00-2.00 sec    128 KBytes  1.05 Mbits/sec   16
7 [ 4]  2.00-3.00 sec    128 KBytes  1.05 Mbits/sec   16
8 [ 4]  3.00-4.00 sec    128 KBytes  1.05 Mbits/sec   16
9 [ 4]  4.00-4.52 sec    80.0 KBytes 1.26 Mbits/sec   10
10                                     ...
```