socat 使用手册

目 录

1 引言

1.1 目的

本手册的编写目的是对在 Linux 系统使用 socat 的相关步骤进行总结。

1.2 对象

本文档主要供下列人员使用：

- 实施人员——部署 socat

2 修订历史

日期	版本	说明	作者
2012-10-25	1.0.0	初步整理	Marsleo

3 参考资料

[1].

4 术语与缩写

4.1 术语

序号	术语名称	术语定义

4.2 缩写

序号	缩写	缩写意义

5 socat 简介

socat 是一个多功能的网络工具，名字来由是” Socket CAT”，可以看作是 netcat 的 N 倍加强版，socat 的官方网站：http://www.dest-unreach.org/socat/ 。

Socat 是一个两个独立数据通道之间的双向数据传输的继电器。这些数据通道包含文件、管道、设备（终端或调制解调器等）、插座（Unix，IP4，IP6 - raw，UDP，TCP）、SSL、SOCKS4 客户端或代理 CONNECT。

Socat 支持广播和多播、抽象 Unix sockets、Linux tun/tap、GNU readline 和 PTY。它提供了分叉、记录和进程间通信的不同模式。多个选项可用于调整 socat 和其渠道，Socat 可以作为 TCP 中继（一次性或守护进程），作为一个守护进程基于 socksifier，作为一个 shell Unix 套接字接口，作为 IP6 的继电器，或面向 TCP 的程序重定向到一个串行线。

socat 的主要特点就是在两个数据流之间建立通道；且支持众多协议和链接方式：ip, tcp, udp, ipv6, pipe, exec, system, open, proxy, openssl, socket 等。

6 socat 的安装

6.1 下载软件包

下载 socat 源代码包:http://www.dest-unreach.org/socat/download/ 。

6.2 编译安装

把下载的软件包解压后按照传统的方式编译安装：

```
./configure
make
make install
```

具体的细节可以参考安装文档 README 。

在编译的过程中可能遇到如下错误：

```
FIPSLD_CC=gcc fipsld -O -D_GNU_SOURCE  -Wall -Wno-parentheses -DHAVE_CONFIG_H -I.  -I.   -c -o socat.o socat.c
/bin/sh: fipsld: command not found
```

解决方法有两种：

第一种是禁用 fips，使用如下命令配置：

```
./configure --disable-fips
```

第二种是安装 fips，首先到网站 http://www.openssl.org/source/ 下载 openssl-fips 安装包，然后解压安装：

```
./config
make
make install
```

7 socat 的使用

socat 的具体文档参见网站：http://www.dest-unreach.org/socat/doc/socat.html 。

7.1 工作机理

socat 的运行有 4 个阶段:

7.1.1 初始化

解析命令行以及初始化日志系统。

7.1.2 打开连接

先打开第一个连接，再打开第二个连接。这个单步执行的。 如果第一个连

接失败，则会直接退出。

7.1.3 数据转发

谁有数据就转发到另外一个连接上，read/write 互换。

7.1.4 关闭

其中一个连接掉开，执行处理另外一个连接。

7.2 地址类型

参数由 2 部分组成，第一个连接和第二个连接，最简单的用法就是 socat -- 其效果就是输入什么，回显什么其用法主要在于地址如何描述，下面介绍几个常用的。

7.2.1 TCP

TCP:<host>:<port> 目标机器 host 对应端口 portTCP-LISTEN:<port> 本机监听端口。

7.2.2 UDP

UDP:<host>:<port> 目标机器 host 对应端口 portUDP-LISTEN:<port> 本机监听端口。

7.2.3 OPENSSL

需要一个证书，否则会失败提示：2012/04/06 11:29:11 socat[1614] E SSL_connect(): error:14077410:SSL routines:SSL23_GET_SERVER_HELLO:sslv3 alert handshake failureOPENSSL:<host>:<port> 目标机器 host 对应端口 portOPENSSL-LISTEN:<port> 本机监听端口。

7.2.4 TUN

TUN[:<if-addr>/<bits>] 建立 vpn，双方都需要 root 权限。

7.3 典型使用

7.3.1 连接目标

```
socat - tcp:192.168.1.18:80
```

这个命令等同于 nc 192.168.1.18 80。 socat 里面，必须有两个流，所以第一个参数-代表标准的输入输出，第二个流连接到192.168.1.18的80端口。

```
socat -d -d READLINE,history=$HOME/.http_history TCP4:www.qq.com:80
```

这个例子支持历史记录查询，类似于 bash 的历史记录。

7.3.2 反向连接

再看一个反向 telnet 的例子：

on server:

```
socat tcp-listen:23 exec:cmd,pty,stderr
```

这个命名把 cmd 绑定到端口 23，同时把 cmd 的 Stderr 复位向到 stdout。

on client:

```
socat readline tcp:server:23
```

连接到服务器的 23 端口，即可获得一个 cmd shell。readline 是 gnu 的命令行编辑器，具有历史功能。

7.3.3 向远处端口发数据

```
echo “test” | socat – tcp-connect:127.0.0.1:12345
```

7.3.4 本地开启端口

```
socat tcp-l:7777,reuseaddr,fork system:bash
```

同 nc -l -p 7777 -e bash。

7.3.5 执行 bash 的完美用法

在目标上

```
socat tcp-l:8888 system:bash,pty,stderr
```

本地

```
socat readline tcp:$target:8888
```

用 readline 替代-，就能支持历史功能了。在这个模式下的客户端有本地

一样的效果

7.3.6 文件传递

再看文件传递的例子。nc 也经常用来传递文件，但是 nc 有一个缺点，就是不知道文件什么时候传完了，一般要用 Ctrl+c 来终止，或者估计一个时间，用-w 参数来让他自动终止。用 socat 就不用这么麻烦了：

on host 1:

```
socat -u open:myfile.exe,binary tcp-listen:999
```

on host 2:

```
socat -u tcp:host1:999 open:myfile.exe,create,binary
```

这个命令把文件 myfile.exe 用二进制的方式，从 host 1 传到 host 2。-u 表示数据单向流动，从第一个参数到第二个参数，-U 表示从第二个到第一个。文件传完了，自动退出。

7.3.7 转发

7.3.7.1 本地端口转向远程主机

```
socat TCP4-LISTEN:8888 TCP4:www.qq.com:80
```

如果需要使用并发连接，则加一个 fork,如下:

```
socat TCP4-LISTEN:8888,fork TCP4:www.qq.com:80
```

本地监听 8888 端口，来自 8888 的连接重定向到目标 www.qq.com:80

7.3.7.2 端口映射

再来一个大家喜欢用的例子。在一个 NAT 环境，如何从外部连接到内部的一个端口呢？只要能够在内部运行 socat 就可以了。

外部：

```
socat tcp-listen:1234 tcp-listen:3389
```

内部：

```
socat tcp:outerhost:1234 tcp:192.168.12.34:3389
```

这样，你外部机器上的 3389 就映射在内部网 192.168.12.34 的 3389 端口上。

7.3.7.3 VPN

服务端

```
socat -d -d TCP-LISTEN:11443,reuseaddr TUN:192.168.255.1/24,up
```

客户端

```
socat TCP:1.2.3.4:11443 TUN:192.168.255.2/24,up
```

7.3.8 重定向

```
socat TCP4-LISTEN:80,reuseaddr,fork TCP4:192.168.123.12:8080
```

TCP4-LISTEN：在本地建立的是一个 TCP ipv4 协议的监听端口；

reuseaddr：绑定本地一个端口；

fork：设定多链接模式，即当一个链接被建立后，自动复制一个同样的端口再进行监听

socat 启动监听模式会在前端占用一个 shell，因此需使其在后台执行。

```
socat -d -d tcp4-listen:8900,reuseaddr,fork tcp4:10.5.5.10:3389 # 端口转发
```

或者

```
socat -d -d -lf /var/log/socat.log TCP4-LISTEN:15000,reuseaddr,fork,su=nobody TCP4:static.5iops.com:15000
```

“-d -d -lf /var/log/socat.log”是参数，前面两个连续的-d -d 代表调试信息的输出级别，-lf 则指定输出信息的保存文件。

“TCP4-LISTEN:15000,reuseaddr,fork,su=nobody”是一号地址，代表在 15000 端口上进行 TCP4 协议的监听，复用绑定的 IP，每次有连接到来就 fork 复制一个进程进行处理，同时将执行用户设置为 nobody 用户。

“TCP4:static.5iops.com:15000″是二号地址,代表将 socat 监听到的任何请求，转发到 static.5iops.com:15000 上去。

7.3.9 读写分流

socat 还具有一个独特的读写分流功能，比如：

```
socat open:read.txt!!open:write.txt,create,append tcp-listen:80,reuseaddr,fork
```

这个命令实现一个假的 web server，客户端连过来之后，就把 read.txt 里面的内容发过去，同时把客户的数据保存到 write.txt 里面。”!!”符号用户合并读写流，前面的用于读，后面的用于写。

7.3.10 通过 openssl 来加密传输过程

7.3.10.1 证书生成

```
FILENAME=server openssl genrsa -out $FILENAME.key 1024openssl req -new -key $FILENAME.key -x509 -days 3653 -out $FILENAME.crtcat $FILENAME.key $FILENAME.crt >$FILENAME.pem
```

在当前目录下生成 server.pem server.crt

7.3.10.2 使用

在服务端

```
socat openssl-listen:4433,reuseaddr,cert=srv.pem,cafile=srv.crt system:bash,pty,stderr
```

在本地

```
socat readline openssl:localhost:4433,cert=srv.pem,cafile=srv.crt
```