编制	编号		文档类型
研发部 / 吕曼			
批准人	日期	版本标识	文档名称
	2009-12-03		

Solaris 和 Linux 环境下

抓包工具的使用

目录

1. Solaris环境下抓包工具的使用

1.1. Snoop的介绍

Snoop 是 Solaris 系统中自带的工具， 是一个用于显示网络通讯的程序， 它可捕获 IP 包并将其显示或保存到指定文件. (限超级用户使用 snoop)
Snoop 可将捕获的包以一行的形式加以总结或用多行加以详细的描述(有调用不同的参数－v -V 来实现). 在总结方式下(-V)，将仅显示最高层的相关协议，例如一个 NFS 包将仅显示 NFS 信息，其低层的 RPC，UDP，IP，Ethernet 帧信息将不会显示，但是当加上相应的参数(-v)，这些信息都能被显示出来.

1.2. Snoop的参数简介

参数简介：
[-a] # Listen to packets on audio
[-d device] # settable to le?, ie?, bf?, tr
[-s snaplen] # Truncate packets
[-c count] # Quit after count packets
[-P] # Turn OFF promiscuous mode
[-D] # Report dropped packets
[-S] # Report packet size
[-i file] # Read previously captured packets
[-o file] # Capture packets in file
[-n file] # Load addr-to-name table from file
[-N] # Create addr-to-name table
[-t r|a|d] # Time: Relative, Absolute or Delta
[-v] # Verbose packet display
[-V] # Show all summary lines
[-p first[,last]] # Select packet(s) to display
[-x offset[,length]] # Hex dump from offset for length
[-C] # Print packet filter code

1.3. Snoop使用的例子说明

由于 snoop 的使用非常灵活，希望能通过下面一些例子的学习来其常见用法.

1. 监听所有以本机为源和目的的包并将其显示出来

#snoop

2. 监听所有以主机 A 为源和目的的包并将其显示出来. (A 为主机名，下同)

snoop A

3. 监听所有 **A** 和 **B** 之间的包并将其保存到文件 **file.**

snoop -o file A B

4. 显示文件 **file** 中指定的包**(99-108)**

snoop - i file -p 99,108

99 0.0027 boutique ->; sunroof NFS C GETATTR FH=8E6C
100 0.0046 sunroof ->; boutique NFS R GETATTR OK
101 0.0080 boutique ->; sunroof NFS C RENAME FH=8E6C MTra00192
to .nfs08
102 0.0102 marmot ->; viper NFS C LOOKUP FH=561E screen.r.13.i386
103 0.0072 viper ->; marmot NFS R LOOKUP No such file or directory
104 0.0085 bugbomb ->; sunroof RLOGIN C PORT=1023 h
105 0.0005 kandinsky ->; sparky RSTAT C Get Statistics
106 0.0004 beeblebrox ->; sunroof NFS C GETATTR FH=0307
107 0.0021 sparky ->; kandinsky RSTAT R
108 0.0073 office ->; jeremiah NFS C READ FH=2584 at 40960 for 8192

5. 详细查看文件 **file** 中第 **101** 个包

snoop - i file - v -p101
ETHER: ----- Ether Header -----
ETHER:
ETHER: Packet 101 arrived at 16:09:53.59
ETHER: Packet size = 210 bytes
ETHER: Destination = 8:0:20:1:3d:94, Sun
ETHER: Source = 8:0:69:1:5f:e, Silicon Graphics
ETHER: Ethertype = 0800 (IP)
ETHER:
IP: ----- IP Header -----
IP:
IP: Version = 4, header length = 20 bytes
IP: Type of service = 00
IP: ..0. .... = routine
IP: ...0 .... = normal delay
IP: .... 0... = normal throughput
IP: .... .0.. = normal reliability
IP: Total length = 196 bytes
IP: Identification 19846
IP: Flags = 0X
IP: .0.. .... = may fragment
IP: ..0. .... = more fragments

6. 查看主机 **A** 和主机 **B** 之间的 **NFS** 包**(**命令中的 **and** 和 **or** 为相应的逻辑运算**)**

```
# snoop - i file rpc nfs and A and B
1 0.0000 A ->; B NFS C GETATTR FH=8E6C
2 0.0046 B ->; A NFS R GETATTR OK
3 0.0080 A ->; B NFS C RENAME FH=8E6C MTra00192 to .nfs08
```

7. 将这些符合条件的包保存到另一文件 **file2** 中

```
# snoop - i file -o file2 rpc nfs A B
```

8. 监听主机 **A** 和主机 **B** 间所有 **TCP 80** 端口或 **UDP80** 端口的包

```
# snoop A and B and (tcp or udp) and port 80
```

9. 监听所有的广播包

```
# snoop broadcast
Using device /dev/hme (promiscuous mode)
10.10.10.50 ->; BROADCAST UDP D=177 S=2541 LEN=35
10.10.10.50 ->; BROADCAST UDP D=177 S=2541 LEN=35
10.10.10.50 ->; BROADCAST UDP D=177 S=2541 LEN=35
```

10. 监听所有的多播包，并显示详细内容

```
#snoop –v multicast
ETHER: ----- Ether Header -----
ETHER:
ETHER: Packet 1 arrived at 12:33:2.16
ETHER: Packet size = 69 bytes
ETHER: Destination = ff:ff:ff:ff:ff:ff, (broadcast)
ETHER: Source = 0:4:76:46:8f:50,
ETHER: Ethertype = 0800 (IP)
ETHER:
IP: ----- IP Header -----
IP:
IP: Version = 4
IP: Header length = 20 bytes
IP: Type of service = 0x00
IP: xxx. .... = 0 (precedence)
IP: ...0 .... = normal delay
IP: .... 0... = normal throughput
IP: .... .0.. = normal reliability
IP: Total length = 55 bytes
IP: Identification = 14658
IP: Flags = 0x0
```

IP: .0.. .... = may fragment
IP: ..0. .... = last fragment
IP: Fragment offset = 0 bytes
IP: Time to live = 128 seconds/hops
IP: Protocol = 17 (UDP)
IP: Header checksum = ed38
IP: Source address = 10.10.10.50, 10.10.10.50
IP: Destination address = 255.255.255.255, BROADCAST
IP: No options
IP:
UDP: ----- UDP Header -----
UDP:
UDP: Source port = 2541
UDP: Destination port = 177
UDP: Length = 35
UDP: Checksum = 8E35
UDP:
ETHER: ----- Ether Header -----
ETHER:
ETHER: Packet 2 arrived at 12:33:12.16
ETHER: Packet size = 69 bytes
ETHER: Destination = ff:ff:ff:ff:ff:ff, (broadcast)
ETHER: Source = 0:4:76:46:8f:50,
ETHER: Ethertype = 0800 (IP)
ETHER:
IP: ----- IP Header -----
IP:
IP: Version = 4
IP: Header length = 20 bytes
IP: Type of service = 0x00
IP: xxx. .... = 0 (precedence)
IP: ...0 .... = normal delay
IP: .... 0... = normal throughput
IP: .... .0.. = normal reliability
IP: Total length = 55 bytes
IP: Identification = 14985
IP: Flags = 0x0
IP: .0.. .... = may fragment
IP: ..0. .... = last fragment
IP: Fragment offset = 0 bytes
IP: Time to live = 128 seconds/hops

IP: Protocol = 17 (UDP)
IP: Header checksum = ebf1
IP: Source address = 10.10.10.50, 10.10.10.50
IP: Destination address = 255.255.255.255, BROADCAST
IP: No options
IP:
UDP: ----- UDP Header -----
UDP:
UDP: Source port = 2541
UDP: Destination port = 177
UDP: Length = 35
UDP: Checksum = 8E35
UDP:

11. 监听所有的 NTP 协议包

snoop |grep – i NTP
Using device /dev/hme (promiscuous mode)
ts1 ->; 224.0.1.1 NTP broadcast (Tue Jul 23 12:48:50 2002)
ts1 ->; 224.0.1.1 NTP broadcast (Tue Jul 23 12:49:54 2002)
ts1 ->; 224.0.1.1 NTP broadcast (Tue Jul 23 12:50:58 2002)
ts1 ->; 224.0.1.1 NTP broadcast (Tue Jul 23 12:52:02 2002)
ts1 ->; 224.0.1.1 NTP broadcast (Tue Jul 23 12:53:06 2002)
ts1 ->; 224.0.1.1 NTP broadcast (Tue Jul 23 12:54:10 2002)
这里我们也可看到 NTP server 每隔约一分钟即向多播地址广播一次.

12.我的常用 snoop -ta -xa 192.168.0.1

18:28:41.61227 192.168.0.1 -> 192.168.0.255 NBT NS Query Request for WWW.6688.cc[0], Success

0: ffff ffff ffff 0002 557c 5bb0 0800 4500 .....U|[...E.
16: 004e cd24 0000 8011 eb29 c0a8 0001 c0a8 .N.$.....).....
32: 00ff 0089 0089 003a 5c23 907d 0110 0001 .....:\#.}....
48: 0000 0000 0000 2046 4846 4846 4843 4f44 ..... FHFHFHCOD
64: 4244 4744 4443 4f45 4445 5045 4e43 4143 BDGDDCOEDEPENCAC
80: 4143 4143 4141 4100 0020 0001 ACACAAA.. ..

18:28:42.36233 192.168.0.1 -> 192.168.0.255 NBT NS Query Request for WWW.6688.cc[0], Success

```
 0: ffff ffff ffff 0002 557c 5bb0 0800 4500       ........U|[...E.
16: 004e ce75 0000 8011 e9d8 c0a8 0001 c0a8       .N.u............
32: 00ff 0089 0089 003a 5c23 907d 0110 0001       .......:\#.}....
48: 0000 0000 0000 2046 4846 4846 4843 4f44       ...... FHFHFHCOD
64: 4244 4744 4443 4f45 4445 5045 4e43 4143       BDGDDCOEDEPENCAC
80: 4143 4143 4141 4100 0020 0001                 ACACAAA.. ..
```

1.4. 本人使用的例子说明

实际使用时候的例子：

1.snoop -o messagetest.txt 135.191.35.194

这句话的意思是把从 IP 为 135.191.35.194 的主机上发来的包，抓下来，放到文件 messagetest.txt。

然后我们解析文件里面的东西

2.snoop -i messagetest.txt

```
  1    0.00000 135.191.35.194 -> qhydcn1       TCP D=8800 S=40117 Syn Seq=1954221959 Len=0 Win=49640 Options=<mss 1460,nop,wscale 0,nop,nop,sackOK>
  2    0.00008 qhydcn1 -> 135.191.35.194 TCP D=40117 S=8800 Syn Ack=1954221960 Seq=2262478164 Len=0 Win=32850 Options=<mss 1460,nop,wscale 4,nop,nop,sackOK>
  3    0.00403 135.191.35.194 -> qhydcn1       TCP D=8800 S=40117 Ack=2262478165 Seq=1954221960 Len=0 Win=49640
  4    0.00038 135.191.35.194 -> qhydcn1       TCP D=8800 S=40117 Push Ack=2262478165 Seq=1954221960 Len=49 Win=49640
  5    0.00005 qhydcn1 -> 135.191.35.194 TCP D=40117 S=8800 Ack=1954222009 Seq=2262478165 Len=0 Win=32850
  6    0.00098 qhydcn1 -> 135.191.35.194 TCP D=40117 S=8800 Fin Ack=1954222009 Seq=2262478165 Len=0 Win=32850
  7    0.00082 135.191.35.194 -> qhydcn1       TCP D=8800 S=40117 Ack=2262478166 Seq=1954222009 Len=0 Win=49640
  8    0.00012 135.191.35.194 -> qhydcn1       TCP D=8800 S=40117 Fin Ack=2262478166 Seq=1954222009 Len=0 Win=49640
  9    0.00003 qhydcn1 -> 135.191.35.194 TCP D=40117 S=8800 Ack=1954222010 Seq=2262478166 Len=0 Win=32850
 10    2.59268 135.191.35.194 -> qhydcn1       TCP D=8800 S=40120 Syn Seq=1955389308 Len=0 Win=49640 Options=<mss 1460,nop,wscale
```

0,nop,nop,sackOK>

11 0.00013 qhydcn1 -> 135.191.35.194 TCP D=40120 S=8800 Syn Ack=1955389309 Seq=2263960914 Len=0 Win=32850 Options=<mss 1460,nop,wscale 4,nop,nop,sackOK>

12 0.00071 135.191.35.194 -> qhydcn1 TCP D=8800 S=40120 Ack=2263960915 Seq=1955389309 Len=0 Win=49640

13 0.00012 135.191.35.194 -> qhydcn1 TCP D=8800 S=40120 Push Ack=2263960915 Seq=1955389309 Len=49 Win=49640

14 0.00016 qhydcn1 -> 135.191.35.194 TCP D=40120 S=8800 Ack=1955389358 Seq=2263960915 Len=0 Win=32846

15 0.00083 qhydcn1 -> 135.191.35.194 TCP D=40120 S=8800 Fin Ack=1955389358 Seq=2263960915 Len=0 Win=32850

16 0.00075 135.191.35.194 -> qhydcn1 TCP D=8800 S=40120 Ack=2263960916 Seq=1955389358 Len=0 Win=49640

17 0.00011 135.191.35.194 -> qhydcn1 TCP D=8800 S=40120 Fin Ack=2263960916 Seq=1955389358 Len=0 Win=49640

18 0.00004 qhydcn1 -> 135.191.35.194 TCP D=40120 S=8800 Ack=1955389359 Seq=2263960916 Len=0 Win=32850

前面的序列号就是说第几个包

我们发现前 3 个包的包长是 0，第 4 个包长为 49。我们想看一下第 4 个包里面的数据

3.snoop -i messagetest.txt -v -x 54 -p4

```
ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 4 arrived at 10:28:4.16
ETHER:  Packet size = 103 bytes
ETHER:  Destination = 0:3:ba:55:c0:29,
ETHER:  Source      = 0:90:b:9:7f:e5,
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
```

```
IP:             .... ..0. = not ECN capable transport
IP:             .... ...0 = no ECN congestion experienced
IP:     Total length = 89 bytes
IP:     Identification = 25547
IP:     Flags = 0x4
IP:             .1.. .... = do not fragment
IP:             ..0. .... = last fragment
IP:     Fragment offset = 0 bytes
IP:     Time to live = 60 seconds/hops
IP:     Protocol = 6 (TCP)
IP:     Header checksum = 10a3
IP:     Source address = 135.191.35.194, 135.191.35.194
IP:     Destination address = 192.168.94.7, qhydcn1
IP:     No options
IP:
TCP:    ----- TCP Header -----
TCP:
TCP:    Source port = 40117
TCP:    Destination port = 8800
TCP:    Sequence number = 1954221960
TCP:    Acknowledgement number = 2262478165
TCP:    Data offset = 20 bytes
TCP:    Flags = 0x18
TCP:            0... .... = No ECN congestion window reduced
TCP:            .0.. .... = No ECN echo
TCP:            ..0. .... = No urgent pointer
TCP:            ...1 .... = Acknowledgement
TCP:            .... 1... = Push
TCP:            .... .0.. = No reset
TCP:            .... ..0. = No Syn
TCP:            .... ...0 = No Fin
TCP:    Window = 49640
TCP:    Checksum = 0xca2d
TCP:    Urgent pointer = 0
TCP:    No options
TCP:
```

```
0: 0000 0000 0000 0000 434c 5230 3030 3030      ........CLR00000
16: 0000 2020 2020 3330 3136 3400 3030 3030     ..      30164.0000
32: 3000 3051 4859 442c 7072 6f78 7931 3233     0.0QHYD,proxy123
```

48: 3b ;

黄色的部分，就是包里面的数据。

还有可以拿工具 Ethereal 看文件 messagetest.txt 里面的内容。Linux 环境下抓包工具的使用

2. Linux环境下抓包工具的使用

2.1. Tcpdump工具的介绍

linux 下抓包封信工具 Tcpdump 详解

tcpdump 采用命令行方式，它的命令格式为：

tcpdump [-adeflnNOpqStvx] [-c 数量] [-F 文件名]
[-i 网络接口] [-r 文件名] [-s snaplen]
[-T 类型] [-w 文件名] [表达式]

2.2. Tcpdump 的选项介绍

-a 将网络地址和广播地址转变成名字；
-d 将匹配信息包的代码以人们能够理解的汇编格式给出；
-dd 将匹配信息包的代码以 c 语言程序段的格式给出；
-ddd 将匹配信息包的代码以十进制的形式给出；
-e 在输出行打印出数据链路层的头部信息；
-f 将外部的 Internet 地址以数字的形式打印出来；
-l 使标准输出变为缓冲行形式；
-n 不把网络地址转换成名字；
-t 在输出的每一行不打印时间戳；
-v 输出一个稍微详细的信息，例如在 ip 包中可以包括 ttl 和服务类型的信息；
-vv 输出详细的报文信息；
-c 在收到指定的包的数目后，tcpdump 就会停止；
-F 从指定的文件中读取表达式,忽略其它的表达式；
-i 指定监听的网络接口；
-r 从指定的文件中读取包(这些包一般通过-w 选项产生)；
-w 直接将包写入文件中，并不分析和打印出来；
-T 将监听到的包直接解释为指定的类型的报文，常见的类型有 rpc （远程过程调用）和 snmp（简单网络管理协议；）

//

第一种是关于类型的关键字，主要包括 host，net，port，例如 host 210.27.48.2，指明 210.27.48.2 是一台主机，net 202.0.0.0 指明 202.0.0.0 是一个网络地址，port 23 指明端口号是 23。如果没有指定类型，缺省的类型是 host.

第二种是确定传输方向的关键字，主要包括 src , dst ,dst or src, dst and src ,这些关键字指明了传输的方向。举例说明，src 210.27.48.2 ,指明 ip 包中源地址是 210.27.48.2 , dst net 202.0.0.0 指明目的网络地址是 202.0.0.0 。如果没有指明方向关键字，则缺省是 src or dst 关键字。

第三种是协议的关键字，主要包括 fddi,ip,arp,rarp,tcp,udp 等类型。Fddi 指明是在 FDDI(分布式光纤数据接口网络)上的特定的网络协议，实际上它是 "ether"的别名，fddi 和 ether 具有类似的源地址和目的地址，所以可以将 fddi 协议包当作 ether 的包进行处理和分析。其他的几个关键字就是指明了监听的包的协议内容。如果没有指定任何协议，则 tcpdump 将会监听所有协议的信息包。

除了这三种类型的关键字之外，其他重要的关键字如下： gateway, broadcast,less,greater,还有三种逻辑运算，取非运算是 'not ' '! ', 与运算是'and','&&';或运算是'or' ,'||'；这些关键字可以组合起来构成强大的组合条件来满足人们的需要，下面举几个例子来说明。

普通情况下，直接启动 tcpdump 将监视第一个网络界面上所有流过的数据包。

```
# tcpdump
tcpdump: listening on fxp0
11:58:47.873028 202.102.245.40.netbios-ns > 202.102.245.127.netbios-ns: udp 50
11:58:47.974331 0:10:7b:8:3a:56 > 1:80:c2:0:0:0 802.1d ui/C len=43
                0000 0000 0080 0000 1007 cf08 0900 0000
                0e80 0000 902b 4695 0980 8701 0014 0002
                000f 0000 902b 4695 0008 00
11:58:48.373134 0:0:e8:5b:6d:85 > Broadcast sap e0 ui/C len=97
                ffff 0060 0004 ffff ffff ffff ffff ffff
                0452 ffff ffff 0000 e85b 6d85 4008 0002
                0640 4d41 5354 4552 5f57 4542 0000 0000
                0000 00
```

使用-i 参数指定 tcpdump 监听的网络界面，这在计算机具有多个网络界面时非常有用，

使用-c 参数指定要监听的数据包数量，

使用-w 参数指定将监听到的数据包写入文件中保存

A 想要截获所有 210.27.48.1 的主机收到的和发出的所有的数据包：

```
#tcpdump host 210.27.48.1
```

B 想要截获主机 210.27.48.1 和主机 210.27.48.2 或 210.27.48.3 的通信，使用命令：（在命令行中适用 括号时，一定要

```
#tcpdump host 210.27.48.1 and \ (210.27.48.2 or 210.27.48.3 \)
```

C 如果想要获取主机 210.27.48.1 除了和主机 210.27.48.2 之外所有主机通信的 ip 包，使用命令：

```
#tcpdump ip host 210.27.48.1 and ! 210.27.48.2
```

D 如果想要获取主机 210.27.48.1 接收或发出的 telnet 包，使用如下命令：

```
#tcpdump tcp port 23 host 210.27.48.1
```

E 对本机的 udp 123 端口进行监视 123 为 ntp 的服务端口

```
# tcpdump udp port 123
```

F 系统将只对名为 hostname 的主机的通信数据包进行监视。主机名可以是本地主机，也可以是网络上的任何一台计算机。下面的命令可以读取主机 hostname 发送的所有数据：
#tcpdump -i eth0 src host hostname
G 下面的命令可以监视所有送到主机 hostname 的数据包：
#tcpdump -i eth0 dst host hostname
H 我们还可以监视通过指定网关的数据包：
#tcpdump -i eth0 gateway Gatewayname
I 如果你还想监视编址到指定端口的 TCP 或 UDP 数据包，那么执行以下命令：
#tcpdump -i eth0 host hostname and port 80
J 如果想要获取主机 210.27.48.1 除了和主机 210.27.48.2 之外所有主机通信的 ip 包，使用命令：
#tcpdump ip host 210.27.48.1 and ! 210.27.48.2
K 想要截获主机 210.27.48.1 和主机 210.27.48.2 或 210.27.48.3 的通信，使用命令：（在命令行中适用 括号时，一定要
#tcpdump host 210.27.48.1 and \ (210.27.48.2 or 210.27.48.3 \)
L 如果想要获取主机 210.27.48.1 除了和主机 210.27.48.2 之外所有主机通信的 ip 包，使用命令：
#tcpdump ip host 210.27.48.1 and ! 210.27.48.2
M 如果想要获取主机 210.27.48.1 接收或发出的 telnet 包，使用如下命令：
#tcpdump tcp port 23 host 210.27.48.1
第三种是协议的关键字，主要包括 fddi,ip ,arp,rarp,tcp,udp 等类型
除了这三种类型的关键字之外，其他重要的关键字如下：gateway, broadcast,less, greater,还有三种逻辑运算，取非运算是 'not ' '! ', 与运算是'and','&&';或运算 是'or' ,'||';
第二种是确定传输方向的关键字，主要包括 src , dst ,dst or src, dst and src ,
如果我们只需要列出送到 80 端口的数据包，用 dst port；如果我们只希望看到返回 80 端口的数据包，用 src port。
#tcpdump –i eth0 host hostname and dst port 80 目的端口是 80
或者
#tcpdump –i eth0 host hostname and src port 80 源端口是 80 一般是提供 http 的服务的主机
如果条件很多的话 要在条件之前加 and 或 or 或 not
#tcpdump -i eth0 host ! 211.161.223.70 and ! 211.161.223.71 and dst port 80
如果在 ethernet 使用混杂模式 系统的日志将会记录
May 7 20:03:46 localhost kernel: eth0: Promiscuous mode enabled.
May 7 20:03:46 localhost kernel: device eth0 entered promiscuous mode
May 7 20:03:57 localhost kernel: device eth0 left promiscuous mode
tcpdump 对截获的数据并没有进行彻底解码，数据包内的大部分内容是使用十六进制的形式直接打印输出的。显然这不利于分析网络故障，通常的解决办法是先使用带-w 参数的 tcpdump 截获数据并保存到文件中，然后再使用其他程序进行解码分析。当然也应该定义

过滤规则，以避免捕获的数据包填满整个硬盘。