



Network Security Technology

网络安全技术

第三章 网络监听与防御技术

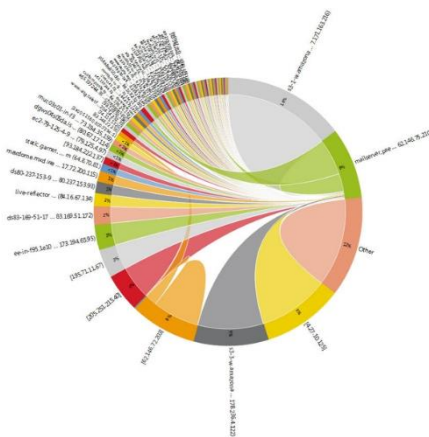
主讲：李强

E-mail: dr_qiangli@163.com

Office: 明实楼（3号楼）A410

内容介绍

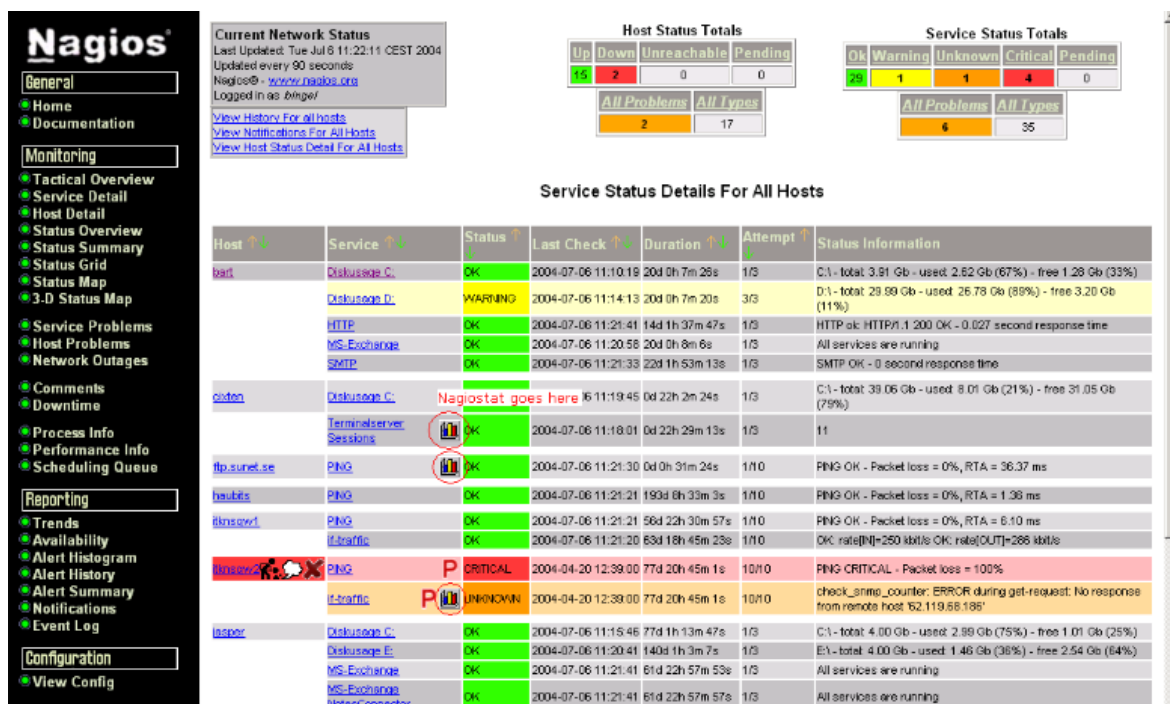
- **3.1** 网络监听概述
- **3.2** 监听技术
- **3.3** 监听的防御
- **3.4** 小结



3.1 网络监听概述

□ 3.1.1 基础知识与实例

□ 3.1.2 网络监听技术的发展情况





3.1.1 基础知识与实例

□ 1. 网络监听的概念

- 网络监听技术又叫做网络嗅探技术(Network Sniffing)，顾名思义，这是一种在他方未察觉的情况下捕获其通信报文或通信内容的技术。
- 在网络安全领域，网络监听技术对于网络攻击与防范双方都有着重要的意义，是一把双刃剑。对网络管理员来说，它是了解网络运行状况的有力助手，对黑客而言，它是有效收集信息的手段。
- 网络监听技术的能力范围目前已经突破局域网。

3.1.1 基础知识与实例

□ 2. 相关网络基础

网络传输技术：广播式和点到点。

- 广播式网络传输技术：仅有一条通信信道，由网络上的所有机器共享。信道上传输的分组可以被任何机器发送并被其他所有的机器接收。
- 点到点网络传输技术：点到点网络由一对对机器之间的多条连接构成，分组的传输是通过这些连接直接发往目标机器，因此不存在发送分组被多方接收的问题。

3.1.1 基础知识与实例

□ 3. 网卡的四种工作模式

- (1) 广播模式：该模式下的网卡能够接收网络中的广播信息。
- (2) 组播模式：该模式下的网卡能够接受组播数据。
- (3) 直接模式：在这种模式下，只有匹配目的MAC地址的网卡才能接收该数据帧。
- (4) 混杂模式：（**Promiscuous Mode**）在这种模式下，网卡能够接受一切接收到的数据帧，而无论其目的MAC地址是什么。

3.1.1 基础知识与实例

□ 4 实例：用Ethereal嗅探sina邮箱密码

Filter: http

No.	Time	Source	Destination	Protocol	Info
2	0.462914	192.168.1.118	219.133.48.109	HTTP	Continuation or non-HTTP traffic
6	0.844122	219.133.48.109	192.168.1.118	HTTP	Continuation or non-HTTP traffic
12	2.272498	192.168.1.118	219.133.48.109	HTTP	Continuation or non-HTTP traffic
16	2.632527	219.133.48.109	192.168.1.118	HTTP	Continuation or non-HTTP traffic
21	4.069930	192.168.1.118	219.133.48.109	HTTP	Continuation or non-HTTP traffic
24	4.430122	219.133.48.109	192.168.1.118	HTTP	Continuation or non-HTTP traffic
33	4.561609	192.168.1.118	202.108.43.230	HTTP	POST /cgi-bin/login.cgi HTTP/1.1 (application/x-www-form-urlencoded)
36	4.767126	202.108.43.230	192.168.1.118	HTTP	HTTP/1.1 302 Found
43	5.875617	192.168.1.118	219.133.48.109	HTTP	Continuation or non-HTTP traffic
46	6.261084	219.133.48.109	192.168.1.118	HTTP	Continuation or non-HTTP traffic
49	7.670024	192.168.1.118	219.133.48.109	HTTP	Continuation or non-HTTP traffic

Raw data of the selected packet (Frame 184 bytes):

```

0000  00 0f 7a 02 00 4b 00 0d 60 36 bd 05 08 00 45 00  ..Z..K.. 6....E.
0010  00 aa d3 b6 40 00 80 06 6e 26 c0 a8 01 76 ca 6c  ..@... n&...v.l
0020  2b e6 24 ea 00 50 f1 a3 0f f9 7d 11 f1 21 50 18  +$.P.. ..}..!P.
0030  ff ff 96 6d 00 00 43 6f 6e 74 65 6e 74 2d 54 79  ...m..Co ntent-Ty
0040  70 65 3a 20 61 70 70 6c 69 63 61 74 69 6f 6e 2f  pe: appl ication/
0050  78 2d 77 77 77 2d 66 6f 72 6d 2d 75 72 6c 65 6e  x-www-fo rm-urlen
0060  63 6f 64 65 64 0d 0a 43 6f 6e 74 65 6e 74 2d 4c  coded..C ontent-L
0070  65 6e 67 74 68 3a 20 35 39 0d 0a 0d 0a 6c 6f 67  length: 5 9....log
0080  69 6e 74 79 70 65 3d 75 69 64 26 75 3d 68 61 63  intype=u id&u=hac
0090  6b 5f 74 65 73 74 69 6e 67 26 70 73 77 3d 68 61  k_testin g&psw=ha
00a0  63 6b 74 65 73 74 69 6e 67 26 3d 25 42 35 25 43  cktestin g&=%B5%C
00b0  37 2b 25 43 32 25 42 43 7+ %C2%BC
  
```

```

..Z..K.. 6....E.
....@... n&...v.l
+$.P.. ..}..!P.
...m..Co ntent-Ty
pe: appl ication/
x-www-fo rm-urlen
coded..C ontent-L
length: 5 9....log
intype=u id&u=hac
k_testin g&psw=ha
cktestin g&=%B5%C
7+ %C2%BC
  
```

U=hack_tesing
Psw=hacktesting

3.1.1 基础知识与实例

□ 4 实例：xx届 学生实验编写的 的sniffer， 嗅探FTP用户 名和密码

数据包内容：

```
时间：10:13:27,协议：TCP  
传递：192.168.1.19--->192.168.1.34; 00-01-6C-A4-BE-77--->00-01-6C-2F-21-23  
源端口：21,目的端口：1946  
数据大小：103  
数据：220 Serv-U FTP Server v6.3 for WinSock ready...
```

FTP Server Version is Serv-U V6.3

数据包内容：

```
时间：10:13:27,协议：TCP  
传递：192.168.1.34--->192.168.1.19; 00-01-6C-2F-21-23--->00-01-6C-A4-BE-77  
源端口：1946,目的端口：21  
数据大小：65  
数据：USER test
```

USER test

数据包内容：

```
时间：10:13:27,协议：TCP  
传递：192.168.1.19--->192.168.1.34; 00-01-6C-A4-BE-77--->00-01-6C-2F-21-23  
源端口：21,目的端口：1946  
数据大小：90  
数据：331 User name okay, need password.
```

数据包内容：

```
时间：10:13:27,协议：TCP  
传递：192.168.1.34--->192.168.1.19; 00-01-6C-2F-21-23--->00-01-6C-A4-BE-77  
源端口：1946,目的端口：21  
数据大小：65  
数据：PASS test
```

PASS test

数据包内容：

```
时间：10:13:27,协议：TCP  
传递：192.168.1.19--->192.168.1.34; 00-01-6C-A4-BE-77--->00-01-6C-2F-21-23  
源端口：21,目的端口：1946  
数据大小：84  
数据：230 User logged in, proceed.
```

Logged in ok



3.1 网络监听概述

□ **3.1.1** 基础知识与实例

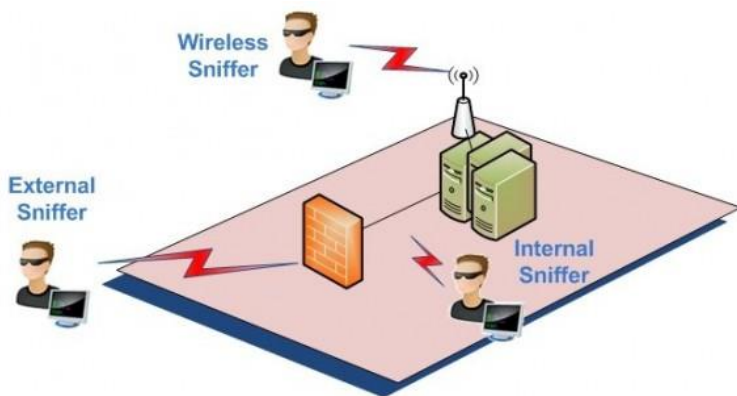
□ **3.1.2** 网络监听技术的发展情况

3.1.2 网络监听技术的发展情况

□ 1. 网络监听（**Sniffer**）的发展历史

Sniffer这个名称最早是一种网络监听工具的名称，后来其也就成为网络监听的代名词。在最初的时候，它是作为网络管理员检测网络通信的一种工具。

□ 网络监听器分**软**、**硬**两种



3.1.2 网络监听技术的发展情况

□ 1. 网络监听（**Sniffer**）的发展历史

- **软件嗅探器**便宜易于使用，缺点是功能往往有限，可能无法抓取网络上所有的传输数据(比如碎片)，或效率容易受限；
- **硬件嗅探器**通常称为协议分析仪，它的优点恰恰是软件嗅探器所欠缺的，处理速度很高，但是价格昂贵。
- 目前主要使用的嗅探器是**软件**的。

3.1.2 网络监听技术的发展情况

□ 2. Sniffer软件的主要工作机制

□ **驱动程序支持**：需要一个直接与网卡驱动程序接口的驱动模块，作为网卡驱动与上层应用的“中间人”，它将网卡设置成混杂模式，捕获数据包，并从上层接收各种抓包请求。

□ **分组捕获过滤机制**：对来自网卡驱动程序的数据帧进行过滤，最终将符合要求的数据交给上层。

链路层的网卡驱动程序上传的数据帧就有了两个去处：一个是**正常的协议栈**，另一个就是**分组捕获过滤模块**，对于非本地的数据包，前者会丢弃（通过比较目的**IP**地址），而后者则会根据上层应用的要求来决定上传还是丢弃。

3.1.2 网络监听技术的发展情况

□ 2. Sniffer软件的主要工作机制

- 许多操作系统都提供这样的“中间人”机制，即分组捕获机制。在UNIX类型的操作系统中，主要有3种：BSD系统中的BPF(Berkeley Packet Filter)、SVR4中的DLPI(Date Link Interface)和Linux中的SOCK_PACKET类型套接字。在Windows平台上主要有NPF过滤机制。
- 目前大部分Sniffer软件都是基于上述机制建立起来的。如Tcpdump、Wireshark等。

3.1.2 网络监听技术的发展情况

□ 3. 网络监听的双刃性

现在的监听技术发展比较成熟，可以协助网络管理员测试网络数据通信流量、实时监控网络状况。

然而事情往往都有两面性，**Sniffer**的隐蔽性非常好，它只是“被动”的接收数据，所以在传输数据的过程中，根本无法察觉到有人在监听。网络监听给网络维护提供便利同时，也给网络安全带来了很大隐患。



3.2 监听技术

- **3.2.1** 局域网中的硬件设备简介
- **3.2.2** 共享式局域网的监听技术
- **3.2.3** 交换式局域网的监听技术
- **3.2.4** 网络监听工具举例

3.2.1 局域网中的硬件设备简介

□ 1. 集线器

(1) 集线器的原理:

集线器（又称为**Hub**）是一种重要的网络部件，主要在局域网中用于将多个客户机和服务器连接到中央区的网络上。

集线器工作在局域网的物理环境下，其主要应用在**OSI**参考模型第一层，属于物理层设备。它的内部采取电器互连的方式，当维护**LAN**的环境是逻辑总线或环型结构时，完全可以用集线器建立一个物理上的星型或树型网络结构。

3.2.1 局域网中的硬件设备简介

□ 1. 集线器

(2) 集线器的工作特点

依据**IEEE 802.3**协议，集线器功能是随机选出某一端口的设备，并让它独占全部带宽，与集线器的上联设备(交换机、路由器或服务器等)进行通信。集线器在工作时具有以下两个特点：

- 首先是集线器只是一个多端口的信号放大设备；
- 其次集线器只与它的上联设备(如上层Hub、交换机或服务器)进行通信，同层的各端口之间不会直接进行通信，而是通过上联设备再将信息广播到所有端口上。

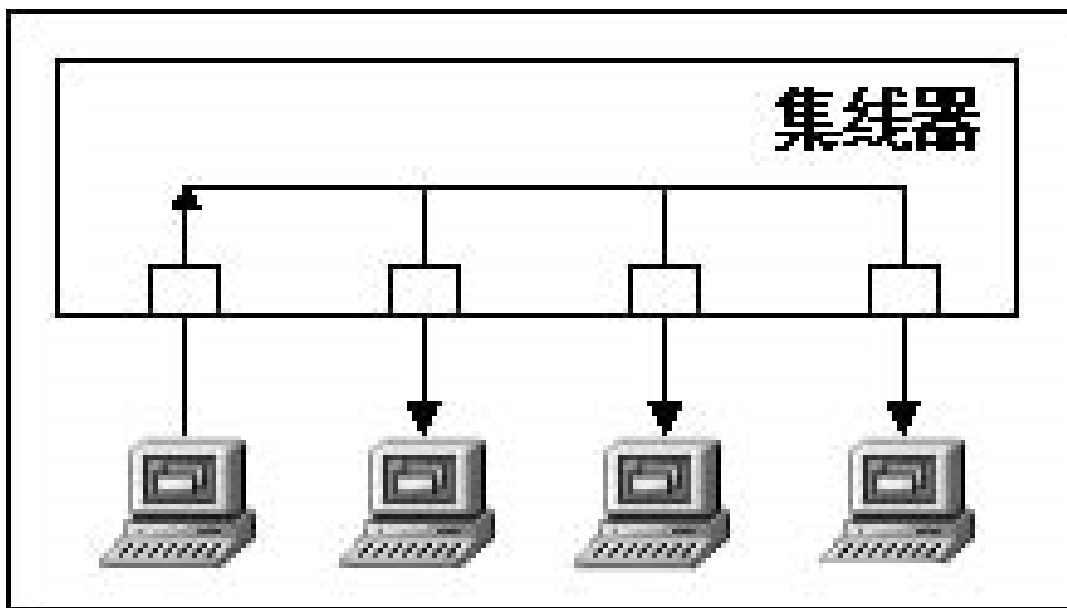


D-LINK DES-1024D 24PORT

3.2.1 局域网中的硬件设备简介

□ 1. 集线器

(3) 用集线器组建的局域网示意图



3.2.1 局域网中的硬件设备简介

□ 2. 交换机

(1) 交换机的原理:

交换机是一种网络开关 (**Switch**), 也称交换器, 由于和电话交换机对出入线的选择有相似的原理, 因此被人称为交换机。

交换机在局域网的环境下, 工作在比集线器更高一层链路层上。交换机被定义成一个能接收发来的信息帧, 加以暂时存储, 然后发到另一端的网络部件, 其本质上就是具有流量控制能力的多端口网桥。

3.2.1 局域网中的硬件设备简介

□ 2. 交换机

(2) 交换机的工作特点

- 把每个端口所连接的网络分割为独立的LAN，每个LAN成为一个独立的冲突域。
- 每个端口都提供专用的带宽。这是交换机与集线器的本质区别，集线器不管有多少端口，都是共享其全部带宽。
- 转发机制。交换机维护有每个端口对应的地址表，其中保存与该端口连接的各个主机的MAC地址。

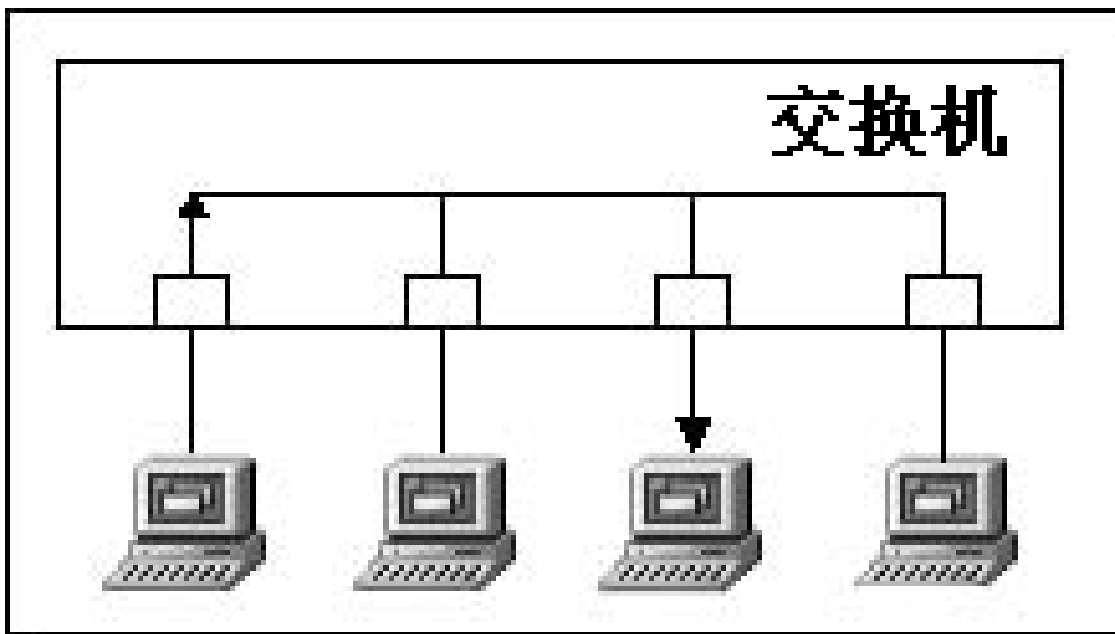


CISCO WS-C2950-24交换机

3.2.1 局域网中的硬件设备简介

□ 2. 交换机

(2) 用交换机组建的局域网示意图



3.2 监听技术

- **3.2.1** 局域网中的硬件设备简介
- **3.2.2** 共享式局域网的监听技术
- **3.2.3** 交换式局域网的监听技术
- **3.2.4** 网络监听工具举例

什么是共享式局域网

- ❑ 共享式局域网就是使用集线器或共用一条总线的局域网，它采用了载波检测多路侦听（**Carries Sense Multiple Access with Collision Detection**，简称**CSMA/CD**）机制来进行传输控制。
- ❑ 共享式局域网是基于广播的方式来发送数据的，因为集线器不能识别帧，所以它就不知道一个端口收到的帧应该转发到哪个端口，它只好把帧发送到除源端口以外的所有端口，这样网络上所有的主机都可以收到这些帧。

共享式局域网的监听原理

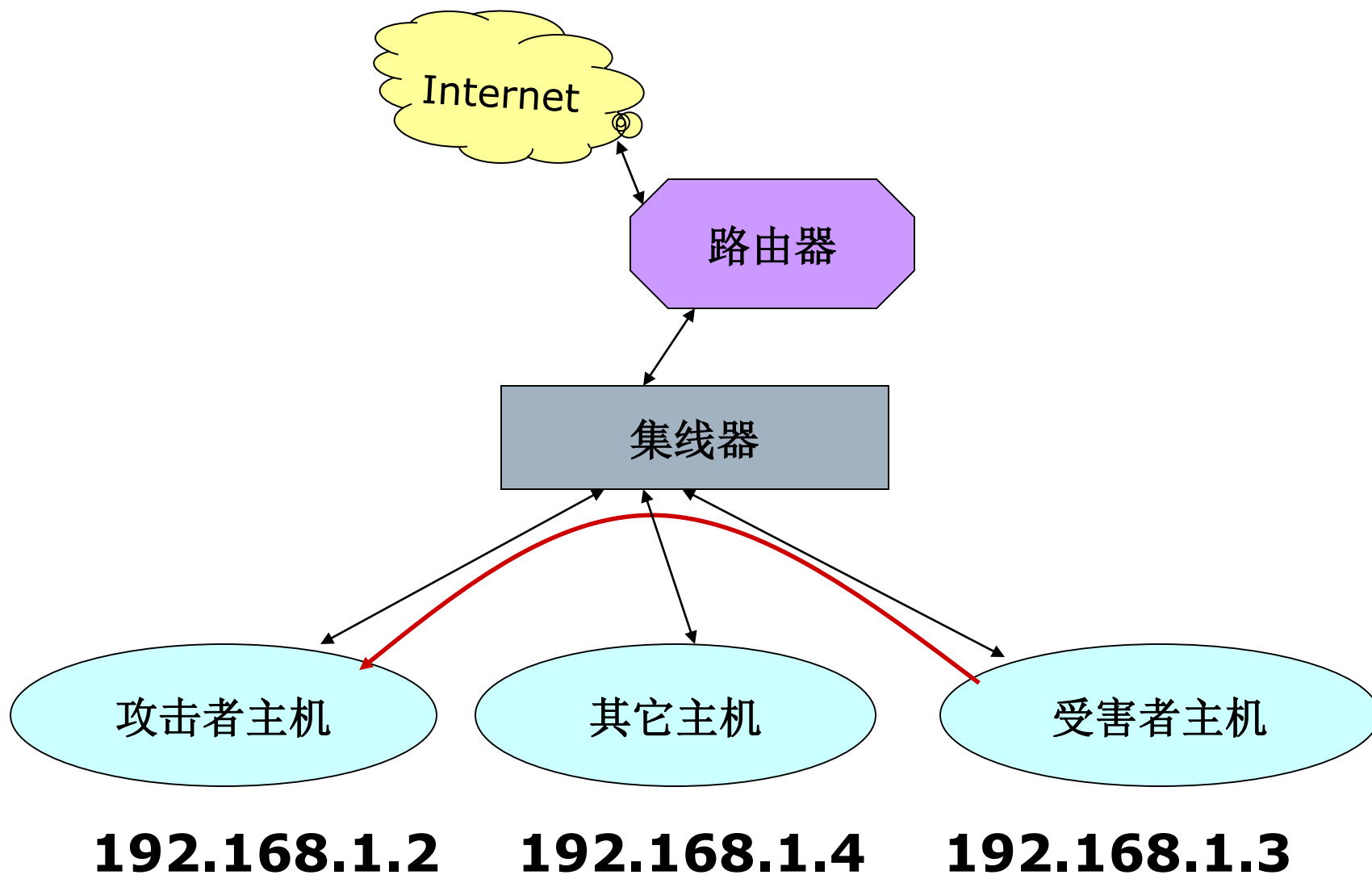
- 在正常的情况下，网卡应该工作在广播模式、直接模式，一个网络接口（网卡）应该只响应这样的两种数据帧：
 - 与自己的**MAC**地址相匹配的数据帧（目的地址为单个主机的**MAC**地址）。
 - 发向所有机器的广播数据帧（目的地址为**0xFFFFFFFF**）。

共享式局域网的监听的工作原理(2)

- 但如果共享式局域网中的一台主机的网卡被设置成混杂模式状态的话，那么，对于这台主机的网络接口而言，任何在这个局域网内传输的信息都是可以被听到的。主机的这种状态也就是监听模式。
- 处于监听模式下的主机可以监听到同一个网段下的其他主机发送信息的数据包。

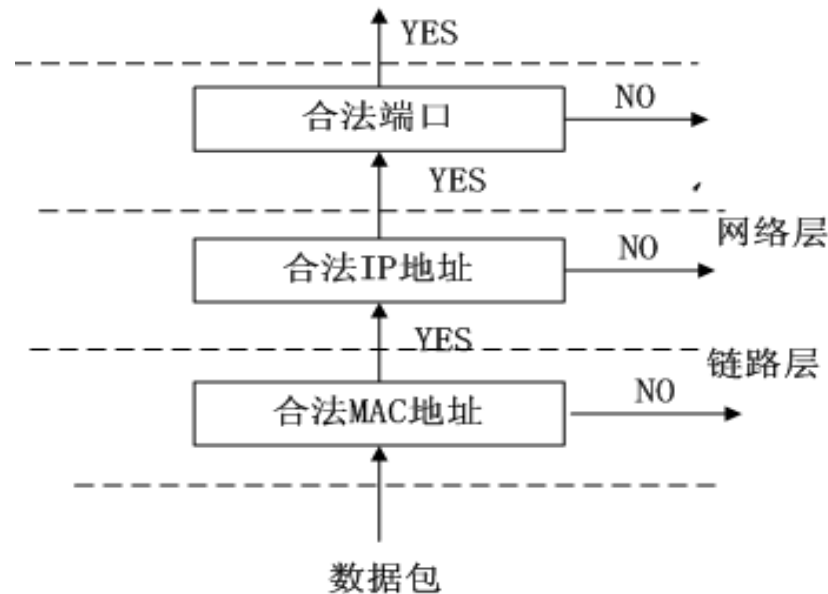
共享式局域网的监听实现方法

- 在共享式局域网中，集线器会广播所有数据，这时，如果局域网中一台主机将网卡设置成混杂模式，那么它就可以接收到该局域网中的所有数据了。
- 网卡在混杂模式工作的情况下，所有流经网卡的数据帧都会被网卡驱动程序上传给网络层。
- 共享式局域网监听示意图见下页。



共享式局域网的监听实现方法(2)

- 正常工作时，应用程序只能接收到以本主机为目标主机的数据包，其他数据包过滤后被丢弃不做处理。
- 该过滤机制可以作用在链路层、网络层和传输层这几个层次，工作流程如图所示：



共享式局域网的监听实现方法(3)

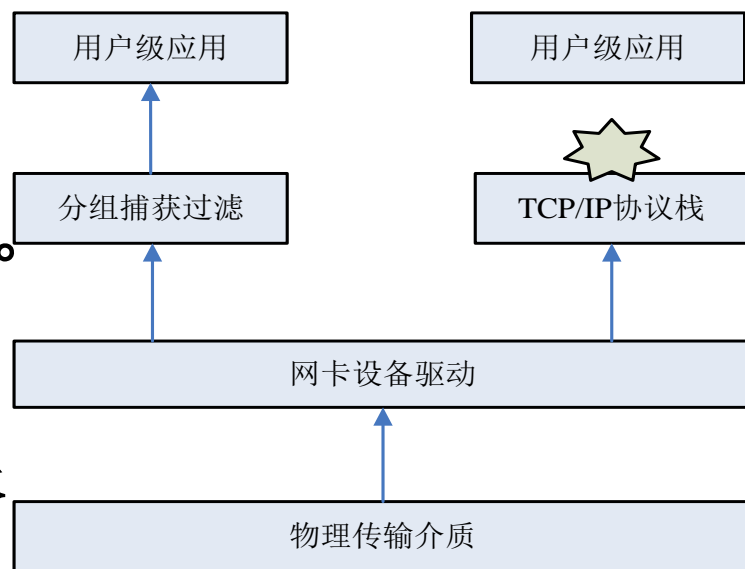
- ❑ 链路层过滤：判断数据包的目的地**MAC**地址。
- ❑ 网络层过滤：判断数据包的目的地**IP**地址。
- ❑ 传输层过滤：判断对应的目的端口是否在本机已经打开。
- ❑ 因而，如果没有一个特定的机制，上层应用也无法抓到本不属于自己的“数据包”。

共享式局域网的监听实现方法(4)

- 需要一个直接与网卡驱动程序接口的驱动模块，它将网卡设置成混杂模式，并从监听软件接收下达的各种抓包请求，对来自网卡驱动程序的数据帧进行过滤，最终将符合监听软件要求的数据返回给监听软件。

共享式局域网的监听实现方法(5)

- 有了驱动模块，链路层的网卡驱动程序上传的数据帧就有了两个去处：一个是正常的协议栈，另一个就是分组捕获即过滤模块。
- 对于非本地的数据包，前者会丢弃（通过比较目的IP地址），而后者则会根据上层应用的要求来决定上传还是丢弃，如图所示。

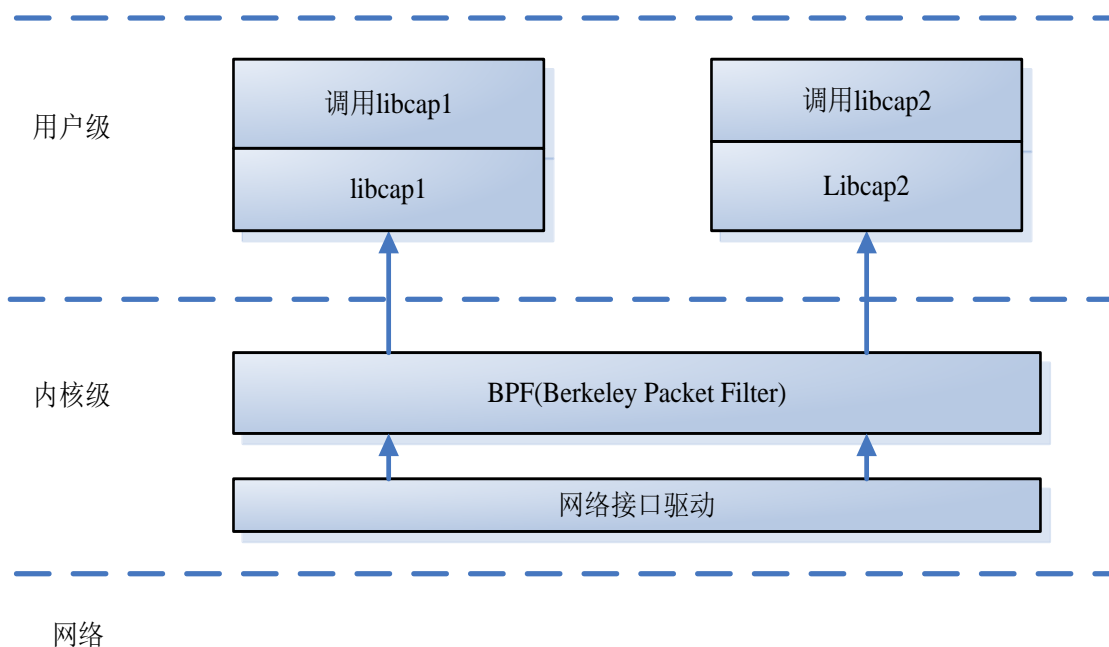


共享式局域网的监听实现方法(6)

- 在实际应用中，监听时存在不需要的数据，严重影响了系统工作效率。网络监听模块过滤机制的效率是该网络监听的关键。
- 信息的过滤包括以下几种：站过滤，协议过滤，服务过滤，通用过滤。
- 同时根据过滤的时间，可以分为两种过滤方式：捕获前过滤、捕获后过滤。

相关开发库

(1) 基于UNIX系统的开发库libpcap



相关开发库

(1) 基于**UNIX**系统的开发库**libpcap**

对开发者而言，网卡驱动程序和**BPF**捕获机制是透明的，需要掌握的是**libpcap**库的使用。**libpcap**隐藏了用户程序和操作系统内核交互的细节，完成了如下工作：

- 向用户程序提供了一套功能强大的抽象接口。
- 根据用户要求生成过滤指令。
- 管理用户缓冲区。
- 负责用户程序和内核的交互。

相关开发库

(2) 基于Windows系统的WinPcap

WinPcap是基于**Windows** 操作系统环境的**Libpcap**，其在监听程序中起的作用和**UNIX** 系统下的**libpcap**类似。但是比**libpcap**多一些功能，如**WinPcap**可以发送数据，但是**libpcap**则不行。

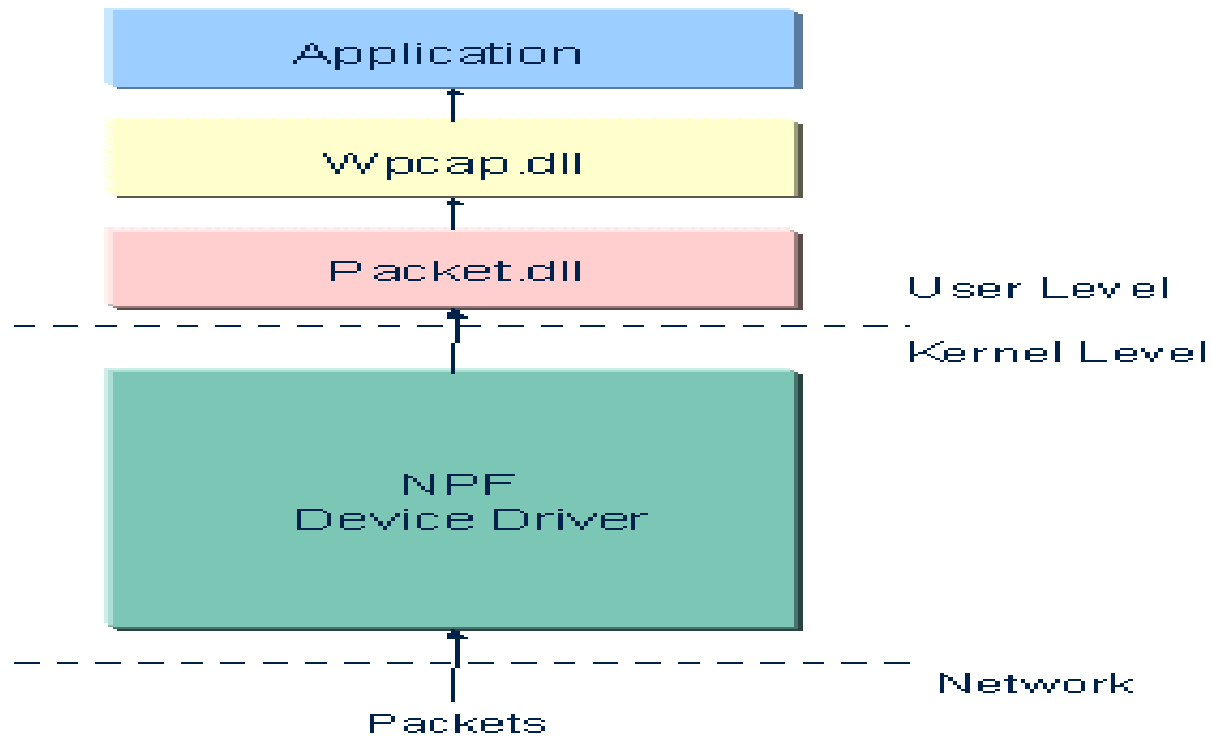
相关开发库

□ WinPcap的架构包括:

- **内核级的数据包监听设备驱动程序NPF:** 把设备驱动增加在Windows, 它直接从数据链路层取得网络数据包不加修改地传递给运行在用户层的应用程序上, 也允许用户发送原始数据包。
- **低级动态链接库packet.dll:** 运行在用户层, 把应用程序和数据包监听设备驱动程序隔离开, 使得应用程序可以不加修改地不同Windows系统上运行。
- **高级系统无关库Wpcap.dll:** 它和应用程序编译在一起, 它使用低级动态链接库提供的服务, 向应用程序提供完善的监听接口, 不同Windows平台上的高级系统无关库是相同的。

相关开发库

□ WinPcap架构图



相关开发库

□ 使用**Winpcap**的流程

打开网卡接口，设置为混杂模式

Pcap_open_live

设置过滤器(捕获前过滤)

Pcap_setfilter

捕获数据

Pcap_next_ex

对捕获到的数据进行处理

MyPacketProcess

Pcap_dump

将捕获到的数据进行存储



3.2 监听技术

- **3.2.1** 局域网中的硬件设备简介
- **3.2.2** 共享式局域网的监听技术
- **3.2.3** 交换式局域网的监听技术
- **3.2.4** 网络监听工具举例

3.2.3 交换式局域网的监听技术

□ 什么是交换式局域网

- 交换式以太网就是用交换机或其它非广播式交换设备组建成的局域网。
- 这些设备根据收到的数据帧中的**MAC**地址决定数据帧应发向交换机的哪个端口。
- 因为端口间的帧传输彼此屏蔽，因此节点就不担心自己发送的帧会被发送到非目的节点中去。

3.2.3 交换式局域网的监听技术

□ 产生交换式局域网的原因：

- 系统管理人员常常通过在本地网络中加入交换设备，来预防sniffer(嗅探器)的侵入。
- 交换机工作在数据链路层，工作时维护着一张MAC地址与端口的映射表。在这个表中记录着交换机每个端口绑定的MAC地址。不同于HUB的报文广播方式，交换机转发的报文是一一对应的。

3.2.3 交换式局域网的监听技术

- 交换式局域网在很大程度上解决了网络监听的困扰。
- 但是交换机的安全性也面临着严峻的考验，随着嗅探技术的发展，攻击者发现了有如下方法来实现现在交换式以太网中的网络监听：
 - 溢出攻击
 - ARP欺骗（常用技术）

3.2.3 交换式局域网的监听技术

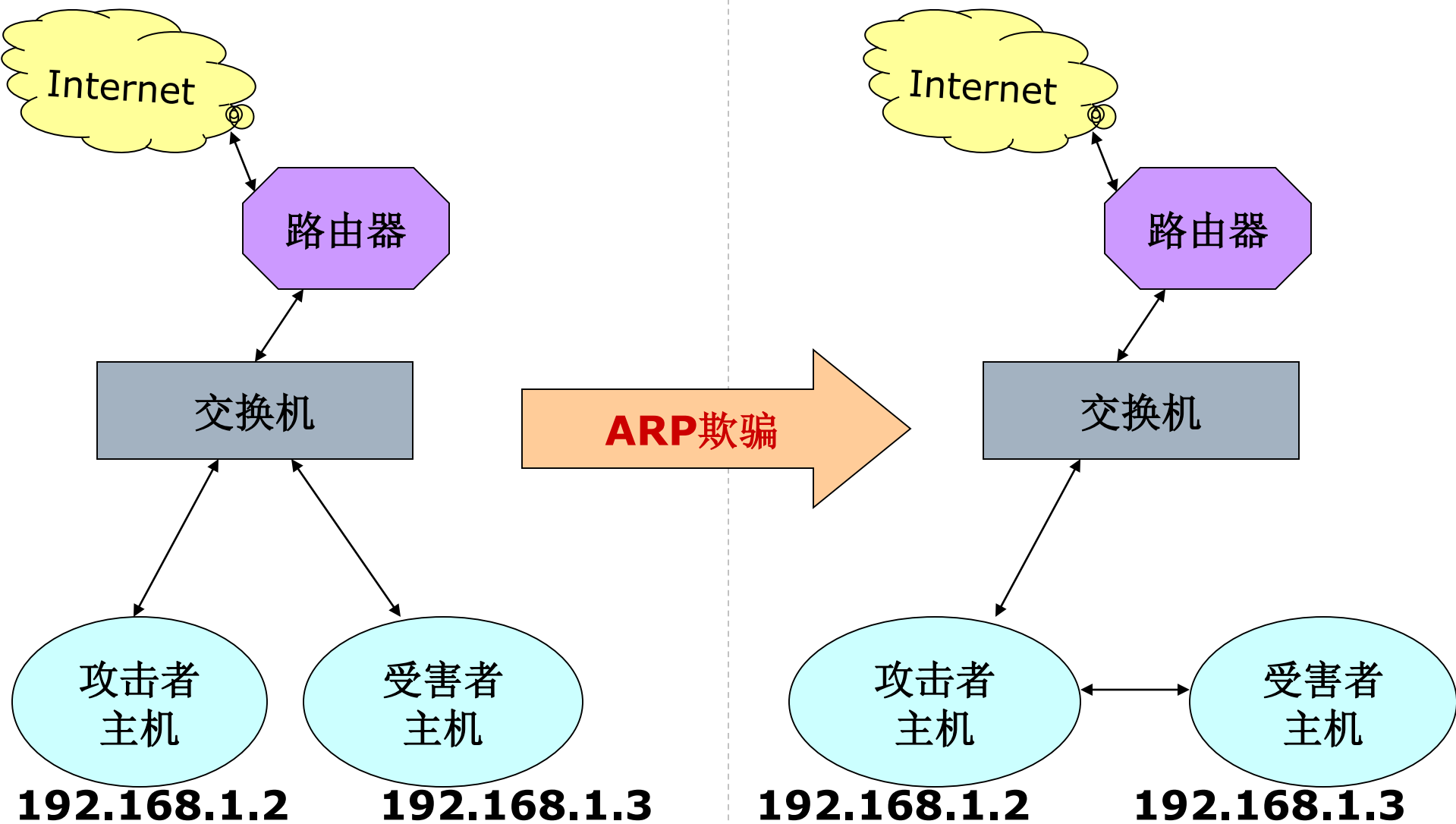
□ 溢出攻击

- 交换机工作时要维护一张**MAC**地址与端口的映射表。
- 但是用于维护这张表的内存是有限的。如用大量的错误**MAC**地址的数据帧对交换机进行攻击，交换机就可能出现溢出。
- 这时交换机就会退回到**HUB**的广播方式，向所有的端口发送数据包，一旦如此，监听就很容易了。

3.2.3 交换式局域网的监听技术

□ ARP欺骗

- 计算机中维护着一个IP-MAC地址对应表，记录了IP地址和MAC地址之间的对应关系。该表将随着ARP请求及响应包不断更新。
- 通过ARP欺骗，改变表里的对应关系，攻击者可以成为被攻击者与交换机之间的“中间人”，使交换式局域网中的所有数据包都流经自己主机的网卡，这样就可以像共享式局域网一样分析数据包了。
- dsniff和parasite等交换式局域网中的嗅探工具就是利用ARP欺骗来实现的。
- ARP欺骗示意图见下页，具体过程会在欺骗攻击章节讲解。





3.2 监听技术

- **3.2.1** 局域网中的硬件设备简介
- **3.2.2** 共享式局域网的监听技术
- **3.2.3** 交换式局域网的监听技术
- **3.2.4** 网络监听工具举例

3.2.4 网络监听工具举例

□ 常用的网络监听工具

- Tcpdump/Windump
- Ngrep
- Ethereal/Wireshark
- Sniffer Pro
- NetXray

□ 网络监听工具的主要功能大都相似，我们以 **Wireshark**为例。

Wireshark简介

- ❑ **Wireshark**是一个免费的开源网络数据包分析工具，可以在**Linux**、**Solaris**、**Windows**等多种平台运行。
- ❑ 它允许用户从一个活动的网络中捕捉数据包并进行分析，详细探究数据包的协议字段信息和会话过程。
- ❑ 帮助网络管理员解决网络问题，帮助网络安全工程师检测安全隐患，开发人员可以用它来测试协议执行情况、学习网络协议。
- ❑ 具有很好的可扩展性，用户能自由地增加插件以实现额外功能。

Wireshark更名的故事

- ❑ 2006年6月8号, **Ethereal**软件的创始人 **Gerald Coombs**宣布离开**NIS**公司(**Ethereal**所属公司), 正式加入**CaceTech**。
- ❑ 由于**Coombs**最终没能与**NIS**公司达成协议, **Coombs**想保留**Ethereal**商标权, 因此将**Ethereal**后续版本更名为**Wireshark**, 属于**CaceTech**公司。
- ❑ **Ethereal**原网站(<http://ethereal.com/>)依旧提供下载服务。

如何获得软件

- **Ethereal**官网：
<http://www.Ethereal.com/>
- **Wireshark**官网（最新稳定版本**1.12**）：
<http://www.wireshark.org/>
- 在安装**Wireshark**时，要同时安装**Winpcap**，它是提供**Windows** 系统所需要的封包捕获驱动程序

Wireshark的特点

- ❑ 支持多种通讯接口（如**Ethernet**、**Token-ring**、**X.25**等）及数据包协议类型（如**ARP**、**TCP**、**UDP**等），可以组合**TCP**上的封包且显示出以**ASCII**或是**EBCDIC**型态的数据（**TCP Stream**），所捕获的封包可以被储存。
- ❑ 支持**Capture Filter**（捕获前过滤）和**Display Filter**（捕获后过滤）功能帮助用户筛选想要的数据包。

Capture filter

- 在捕获数据包之前设定。用于设定捕获数据包时的过滤条件，属于**捕获前过滤**
- 好处:可以让你选择要抓取的数据包
- **Examples:**
 - tcp
 - tcp or udp
 - tcp || udp（此过滤规则是上一条的不同写法）
 - tcp and ip.addr=192.168.1.34

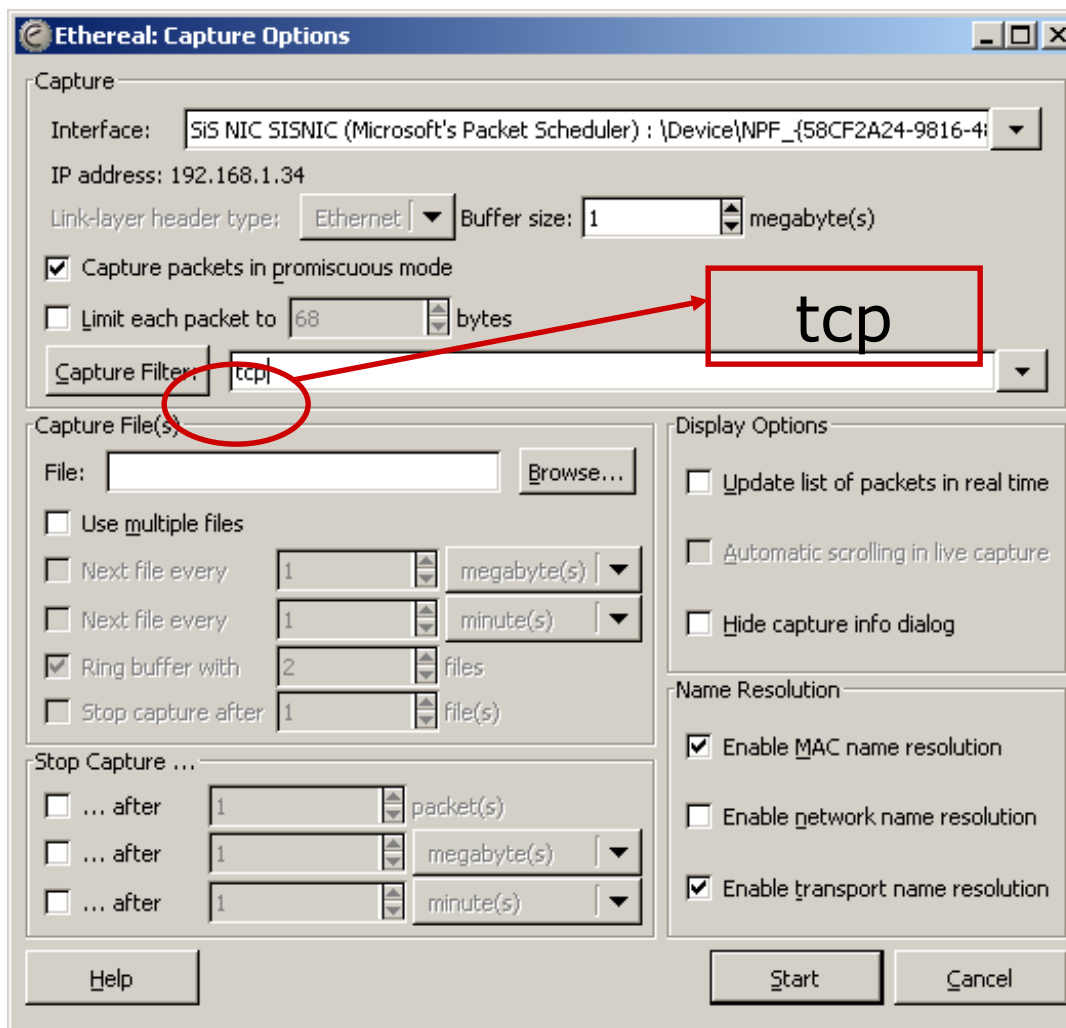
Display filter

- ❑ 在捕获数据包结束后设定。用来设定显示数据包的条件，属于**捕获后过滤**
- ❑ 好处:可以让你选择要看的数据包
- ❑ **Example:**
 - 同Capture filter
 - `tcp.port == 80`
 - `tcp port 80` （此过滤规则是上一条的不同写法）

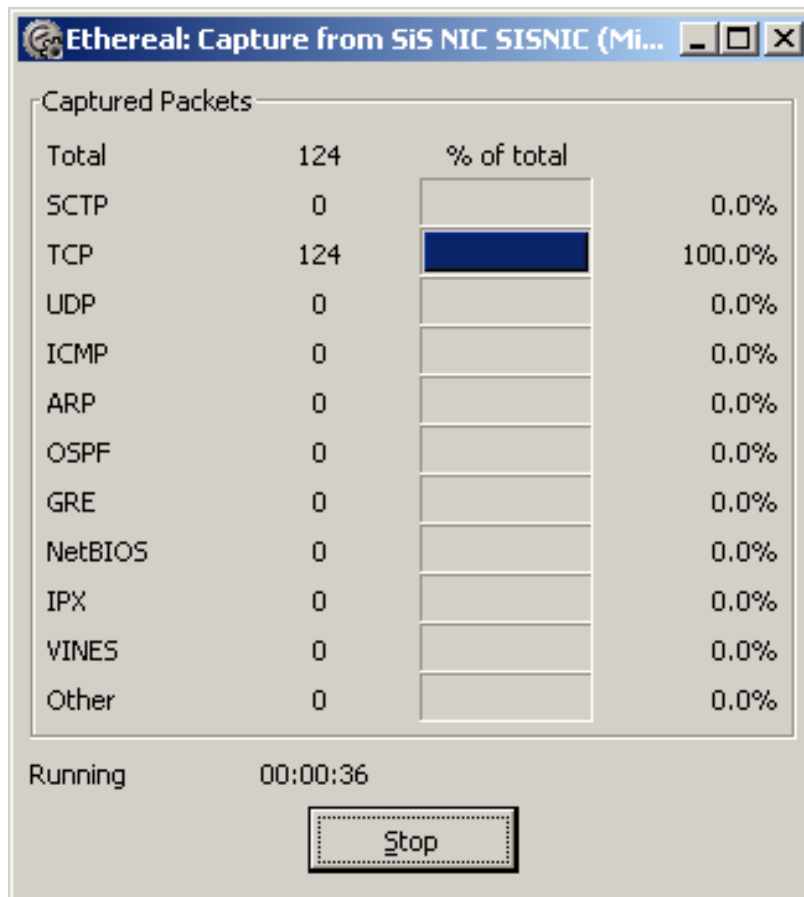
案例一：观察**FTP**数据流

- ❑ 设置**capture filter**，只对**tcp**包进行监测
- ❑ 开始抓取数据包，同时局域网内有**ftp**登陆
- ❑ (隔一小段时间后)
- ❑ 停止抓取封包
- ❑ 观察数据包的格式内容
- ❑ 设置**display filter**，只查看**21**端口与**ftp**主机相关的数据包
- ❑ 使用**follow tcp stream**功能重组数据包，查看**ftp**登陆过程

设置Capture filter



抓包正在进行中，只抓取了**TCP**包

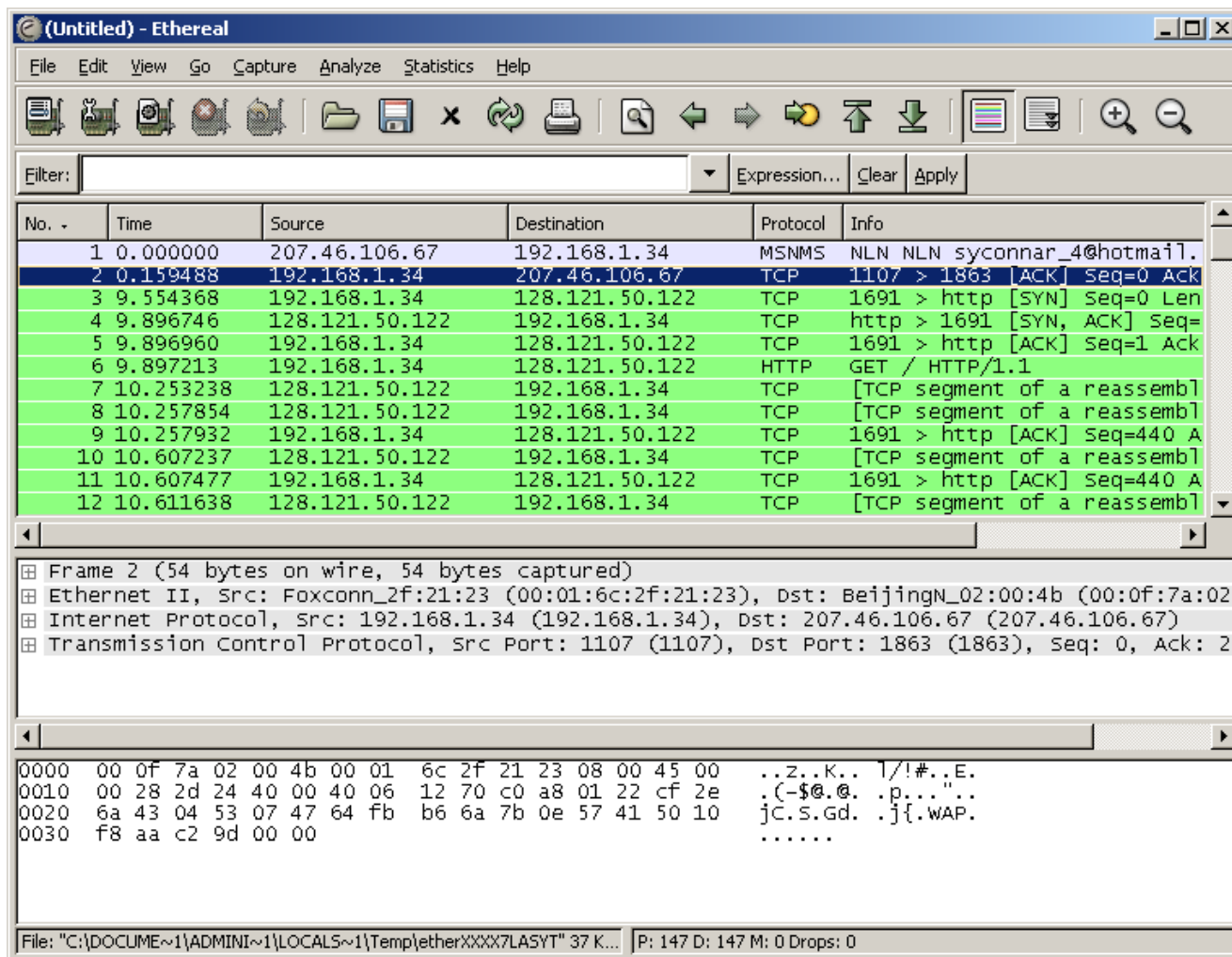


抓包结束，查看封包内容

控制列

封包总览

封包内容



(Untitled) - Ethereal

File Edit View Go Capture Analyze Statistics Help

Filter: Expression... Clear Apply

No.	Time	Source	Destination	Protocol	Info
1	0.000000	207.46.106.67	192.168.1.34	MSNMS	NLN NLN syconnar_4@hotmail.
2	0.159488	192.168.1.34	207.46.106.67	TCP	1107 > 1863 [ACK] Seq=0 Ack
3	9.554368	192.168.1.34	128.121.50.122	TCP	1691 > http [SYN] Seq=0 Len
4	9.896746	128.121.50.122	192.168.1.34	TCP	http > 1691 [SYN, ACK] Seq=
5	9.896960	192.168.1.34	128.121.50.122	TCP	1691 > http [ACK] Seq=1 Ack
6	9.897213	192.168.1.34	128.121.50.122	HTTP	GET / HTTP/1.1
7	10.253238	128.121.50.122	192.168.1.34	TCP	[TCP segment of a reassembl
8	10.257854	128.121.50.122	192.168.1.34	TCP	[TCP segment of a reassembl
9	10.257932	192.168.1.34	128.121.50.122	TCP	1691 > http [ACK] Seq=440 A
10	10.607237	128.121.50.122	192.168.1.34	TCP	[TCP segment of a reassembl
11	10.607477	192.168.1.34	128.121.50.122	TCP	1691 > http [ACK] Seq=440 A
12	10.611638	128.121.50.122	192.168.1.34	TCP	[TCP segment of a reassembl

Frame 2 (54 bytes on wire (54 bytes captured))

Ethernet II, Src: Foxconn_2f:21:23 (00:01:6c:2f:21:23), Dst: BeijingN_02:00:4b (00:0f:7a:02:00:4b)

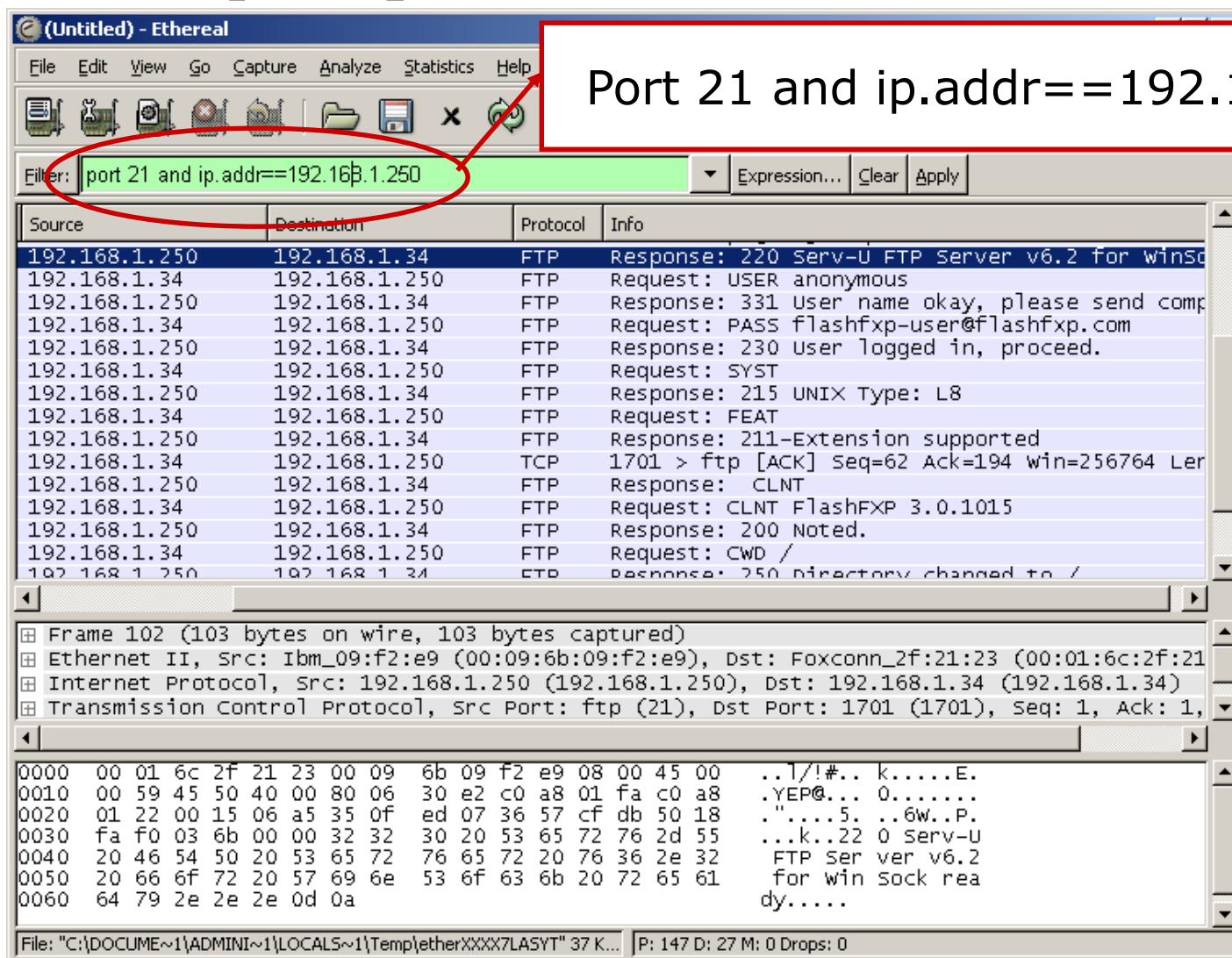
Internet Protocol, Src: 192.168.1.34 (192.168.1.34), Dst: 207.46.106.67 (207.46.106.67)

Transmission Control Protocol, Src Port: 1107 (1107), Dst Port: 1863 (1863), Seq: 0, Ack: 2

0000 00 0f 7a 02 00 4b 00 01 6c 2f 21 23 08 00 45 00 ..Z..K.. 1/!#..E.
0010 00 28 2d 24 40 00 40 06 12 70 c0 a8 01 22 cf 2e .(-\$.@. .p...".
0020 6a 43 04 53 07 47 64 fb b6 6a 7b 0e 57 41 50 10 jC.S.Gd. .j{.WAP.
0030 f8 aa c2 9d 00 00
File: "C:\DOCUME~1\ADMINI~1\LOCALS~1\Temp\etherXXX7LASYT" 37 K... P: 147 D: 147 M: 0 Drops: 0

十六进制码

设置Display filter



The screenshot shows the Wireshark network protocol analyzer interface. The title bar reads "(Untitled) - Ethereal". The menu bar includes File, Edit, View, Go, Capture, Analyze, Statistics, and Help. The toolbar contains icons for file operations and network analysis. The "Filter:" field is highlighted with a red circle and contains the text "port 21 and ip.addr==192.168.1.250". A red arrow points from this filter field to a text box on the right. Below the filter field is a table with columns: Source, Destination, Protocol, and Info. The table lists several FTP and TCP packets. The bottom pane shows the packet details for Frame 102, including Ethernet II, Internet Protocol, and Transmission Control Protocol fields. The bottom status bar shows the file path and packet statistics.

Port 21 and ip.addr==192.168.1.250

Source	Destination	Protocol	Info
192.168.1.250	192.168.1.34	FTP	Response: 220 Serv-U FTP Server v6.2 for winsock
192.168.1.34	192.168.1.250	FTP	Request: USER anonymous
192.168.1.250	192.168.1.34	FTP	Response: 331 User name okay, please send complete login information
192.168.1.34	192.168.1.250	FTP	Request: PASS flashfxp-user@flashfxp.com
192.168.1.250	192.168.1.34	FTP	Response: 230 User logged in, proceed.
192.168.1.34	192.168.1.250	FTP	Request: SYST
192.168.1.250	192.168.1.34	FTP	Response: 215 UNIX Type: L8
192.168.1.34	192.168.1.250	FTP	Request: FEAT
192.168.1.250	192.168.1.34	FTP	Response: 211-Extension supported
192.168.1.34	192.168.1.250	TCP	1701 > ftp [ACK] Seq=62 Ack=194 win=256764 Len=0
192.168.1.250	192.168.1.34	FTP	Response: CLNT
192.168.1.34	192.168.1.250	FTP	Request: CLNT FlashFXP 3.0.1015
192.168.1.250	192.168.1.34	FTP	Response: 200 Noted.
192.168.1.34	192.168.1.250	FTP	Request: CWD /
192.168.1.250	192.168.1.34	FTP	Response: 250 directory changed to /

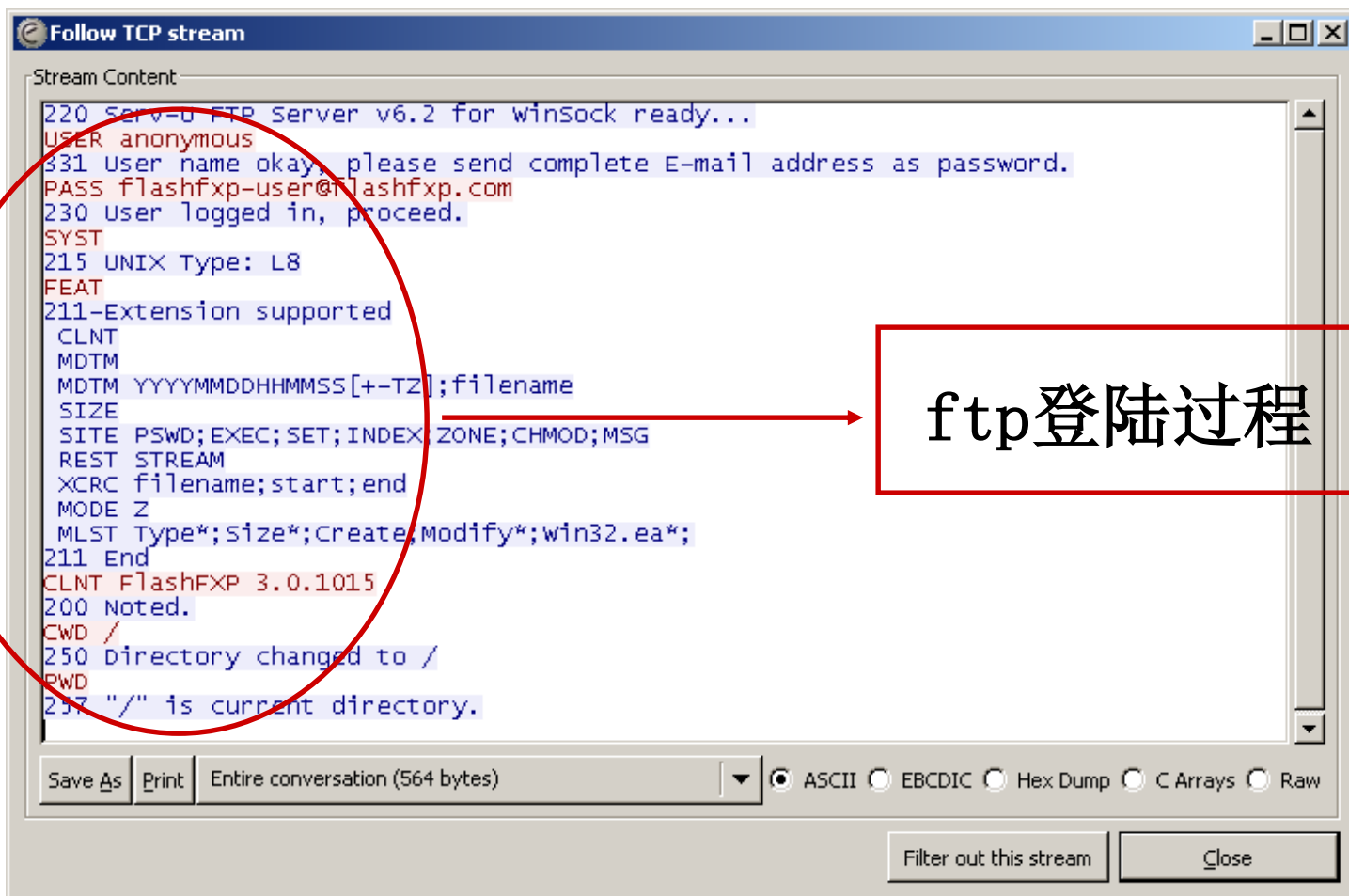
Frame 102 (103 bytes on wire, 103 bytes captured)
Ethernet II, Src: Ibm_09:f2:e9 (00:09:6b:09:f2:e9), Dst: Foxconn_2f:21:23 (00:01:6c:2f:21:23)
Internet Protocol, Src: 192.168.1.250 (192.168.1.250), Dst: 192.168.1.34 (192.168.1.34)
Transmission Control Protocol, Src Port: ftp (21), Dst Port: 1701 (1701), Seq: 1, Ack: 1, Len: 0

0000 00 01 6c 2f 21 23 00 09 6b 09 f2 e9 08 00 45 00 ..1/!#.. k.....E.
0010 00 59 45 50 40 00 80 06 30 e2 c0 a8 01 fa c0 a8 .YEP@... 0.....
0020 01 22 00 15 06 a5 35 0f ed 07 36 57 cf db 50 18 .".5. ..6w..P.
0030 fa f0 03 6b 00 00 32 32 30 20 53 65 72 76 2d 55 ...k..22 0 serv-u
0040 20 46 54 50 20 53 65 72 76 65 72 20 76 36 2e 32 FTP Ser ver v6.2
0050 20 66 6f 72 20 57 69 6e 53 6f 63 6b 20 72 65 61 for win sock rea
0060 64 79 2e 2e 2e 0d 0a dy.....

File: "C:\DOCUME~1\ADMINI~1\LOCAL5~1\Temp\etherXXX7LASYT" 37 K... | P: 147 D: 27 M: 0 Drops: 0

封包重组

选定某一封包内容后，执行**Follow TCP Stream**，即可对与被选中封包相关的所有封包内容进行重组，可更清楚的看到封包中的**Data**。





案例二：监听**TCP**通信过程

- **TCP**通信过程回顾
- 实验环境
- 用**Ethereal**抓包
- 数据包详细分析

TCP通信过程回顾

□ TCP数据报格式

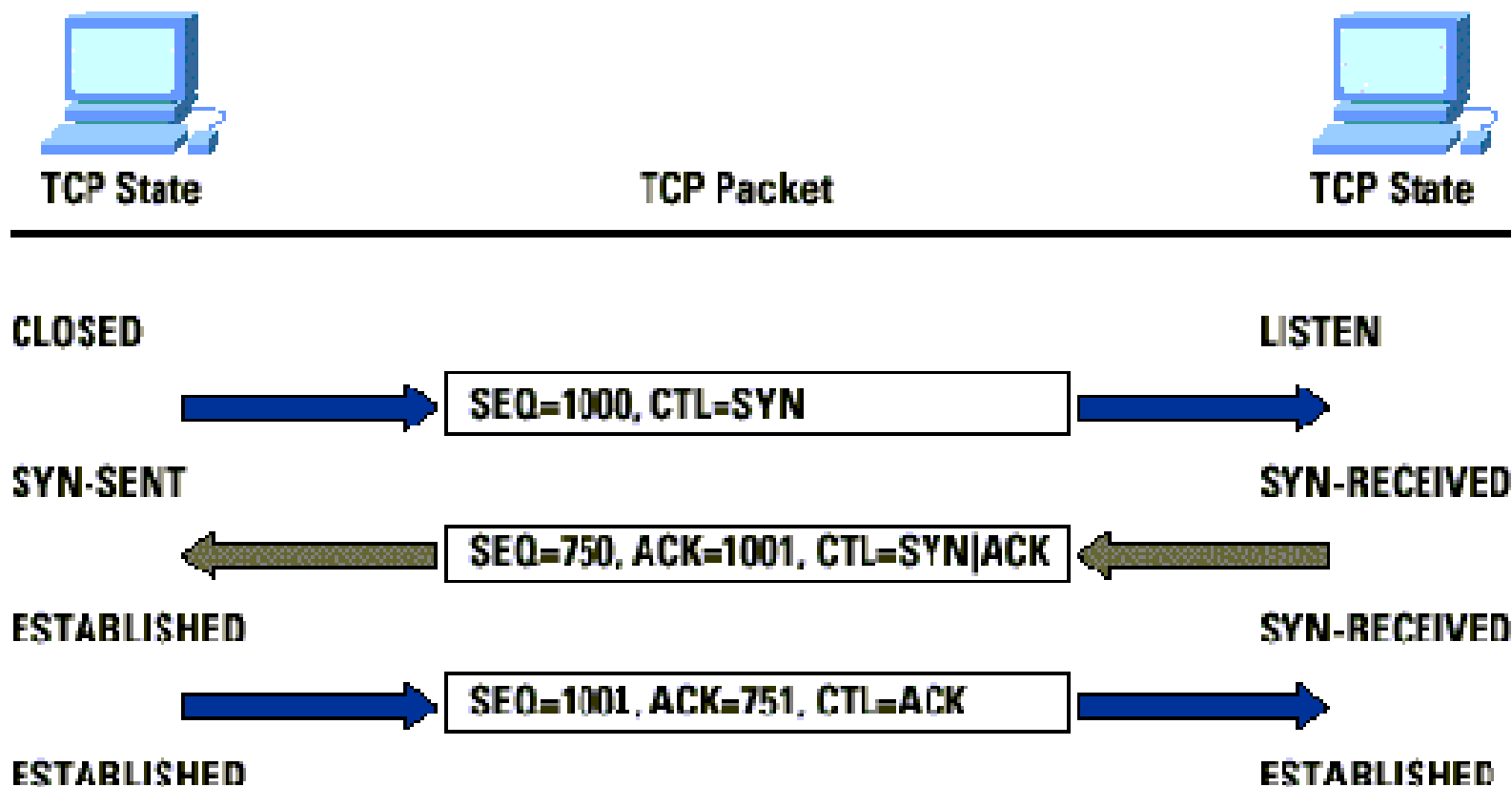
□ 正常TCP通信过程:

- 建立连接
- 数据传输
- 断开连接

TCP数据报格式

源端口（16 位）								目的端口（16 位）							
序号（32 位）															
确认号（32 位）															
TCP 头长 （4 位）	保留位 （6 位）	U R G	A C K	P S H	R S T	S Y N	F I N	窗口大小（16 位）							
校验和（16 位）								紧急指针（16 位）							
可选项（0 或更多的 32 位字）															
数据（可选项）															

TCP连接建立过程



TCP数据传输过程



TCP State

TCP Packet



TCP State

Established

Established

Sending1



SEQ=1001,ACK=751,dataLen=256

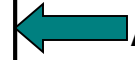


Waiting1

OK1



SEQ=751,ACK=1257



ACK1

Sending2

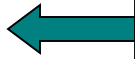


SEQ=1257,ACK=751,dataLen=256

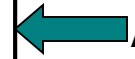


Waiting2

OK2



SEQ=751,ACK=1513



ACK2

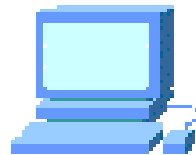
.....

TCP连接断开过程



TCP State

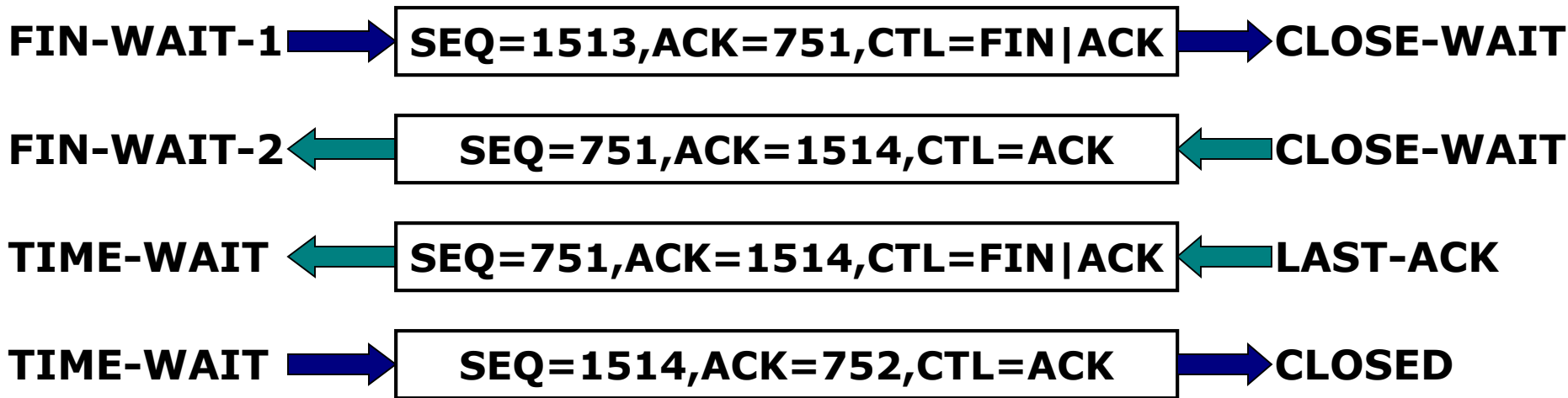
TCP Packet



TCP State

Established

Established



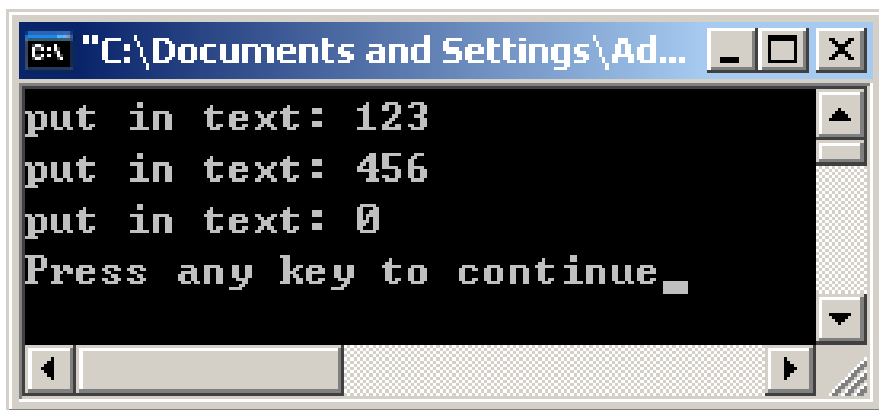
CLOSED

实验环境

- 位于同一局域网内的两台主机，**IP**分别为：
192.168.1.34，**192.168.1.119**
- 自己编写了一个**C/S**模式的程序，实现简单的**TCP**数据发送与接收
- **Client**运行在**192.168.1.34**
- **Server**运行在**192.168.1.119**

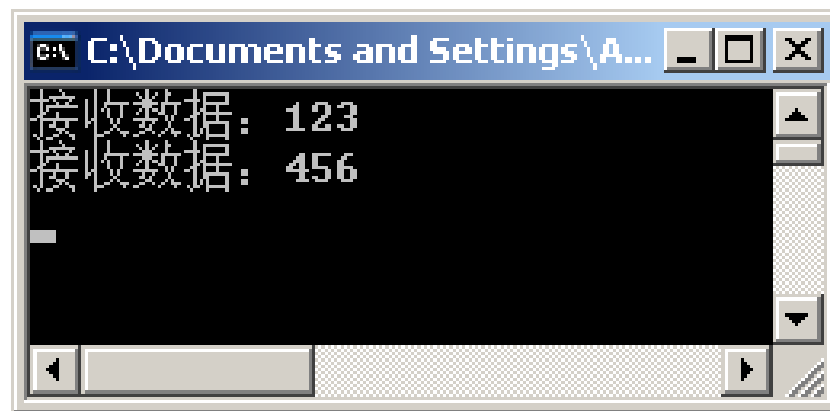
实验环境（2）

- ❑ **Client**发送两次数据，内容分别为**123**和**456**，然后发送**0**结束**TCP**连接。
- ❑ 程序截图如下。



```
C:\Documents and Settings\Ad...  
put in text: 123  
put in text: 456  
put in text: 0  
Press any key to continue_
```

客户端发送数据



```
C:\Documents and Settings\A...  
接收数据: 123  
接收数据: 456
```

服务端接收到数据

捕获数据包

- ❑ 在**Client**发送数据之前，在**192.168.1.34**主机(**Client**)上开启**Ethereal**。
- ❑ 在捕获前不进行过滤，直接捕获所有数据包。
- ❑ 当**Client**结束**TCP**连接之后，停止捕获数据包。
- ❑ 采用**捕获后过滤**的方法，过滤规则是
tcp AND ip.addr==192.168.1.119
其中，**192.168.1.119**是**Server**主机。
- ❑ 过滤后，共得到**11**个数据包，见下页图。



tcp - Ethereal

File Edit View Go Capture Analyze Statistics Help

Filter: `tcp and ip.addr==192.168.1.119` Expression... Clear Apply

Time	Source	Destination	Protocol	Info
4.747917	192.168.1.34	192.168.1.119	TCP	1684 > 4567 [SYN] Seq=0 Len=0 MSS=
4.748564	192.168.1.119	192.168.1.34	TCP	4567 > 1684 [SYN, ACK] Seq=0 Ack=
4.748765	192.168.1.34	192.168.1.119	TCP	1684 > 4567 [ACK] Seq=1 Ack=1 win
7.198946	192.168.1.34	192.168.1.119	TCP	[TCP segment of a reassembled PDU
7.306400	192.168.1.119	192.168.1.34	TCP	4567 > 1684 [ACK] Seq=1 Ack=257 w
9.818699	192.168.1.34	192.168.1.119	TCP	[TCP segment of a reassembled PDU
9.922210	192.168.1.119	192.168.1.34	TCP	4567 > 1684 [ACK] Seq=1 Ack=513 w
11.800602	192.168.1.34	192.168.1.119	TCP	1684 > 4567 [FIN, ACK] Seq=513 Ac
11.801021	192.168.1.119	192.168.1.34	TCP	4567 > 1684 [ACK] Seq=1 Ack=514 w
11.816489	192.168.1.119	192.168.1.34	TCP	4567 > 1684 [FIN, ACK] Seq=1 Ack=
11.816603	192.168.1.34	192.168.1.119	TCP	1684 > 4567 [ACK] Seq=514 Ack=2 w

Frame 4 (66 bytes on wire, 66 bytes captured)

Ethernet II, Src: Foxconn_2f:21:23 (00:01:6c:2f:21:23), Dst: Foxconn_2c:40:d8 (00:15:58:00:01:6c:2f:21:23)

Internet Protocol, Src: 192.168.1.34 (192.168.1.34), Dst: 192.168.1.119 (192.168.1.119)

Transmission Control Protocol, Src Port: 1684 (1684), Dst Port: 4567 (4567), Seq: 0, Len: 0

Source port: 1684 (1684)

Destination port: 4567 (4567)

```

0000  00 15 58 2c 40 d8 00 01 6c 2f 21 23 08 00 45 00  ..X,@... 1/!#..E.
0010  00 34 47 8c 40 00 40 06 6f 4e c0 a8 01 22 c0 a8  .4G.@.@. ON..."..
0020  01 77 06 94 11 d7 28 74 85 73 00 00 00 00 80 02  .w....(t .s.....
0030  ff ff 24 da 00 00 02 04 05 b4 01 03 03 02 01 01  ..$. ....
0040  04 02  ..
  
```

File: "D:\Administrator\...\tcp" 2972 Bytes 00:00:18 | P: 28 D: 11 M: 0

数据包详细分析

- 这**11**个数据包的含义如下：
 - 1~3: 三次握手, 建立连接
 - 4~5: 第一次发送数据
 - 6~7: 第二次发送数据
 - 8~11: 断开连接
- 下面将对这**11**个数据包进行详细分析。

1 C→S SYN

SEQ=X+0

与TCP报文格式相对应

Source port: 1684 (1684)

Destination port: 4567 (4567)

Sequence number: 0 (relative sequence number)

Header Length: 32 bytes

Flags: 0x0002 (SYN)

0... .. = Congestion window Reduced (CWR): Not set

.0.. = ECN-Echo: Not set

..0. = Urgent: Not set

...0 = Acknowledgment: Not set

.... 0... = Push: Not set

.... .0.. = Reset: Not set

.... ..1. = Syn: Set

.... ...0 = Fin: Not set

Window size: 262140 (scaled)

Checksum: 0x24da [correct]

Options: (12 bytes)

2 S→C SYN,ACK

SEQ=Y+0

ACK=X+1

Source port: 4567 (4567)

Destination port: 1684 (1684)

Sequence number: 0 (relative sequence number)

Acknowledgement number: 1 (relative ack number)

Header length: 32 bytes

Flags: 0x0012 (SYN, ACK)

0... .. = Congestion Window Reduced (CWR): Not set

.0... .. = ECN-Echo: Not set

..0. = Urgent: Not set

...1 = Acknowledgment: Set

.... 0... = Push: Not set

.... .0.. = Reset: Not set

.... ..1. = Syn: Set

.... ...0 = Fin: Not set

window size: 65535

checksum: 0x1faa [correct]

options: (12 bytes)

3 C→S ACK

三次握手结束

SEQ=X+1

ACK=Y+1

Source port: 1684 (1684)

Destination port: 4567 (4567)

Sequence number: 1 (relative sequence number)

Acknowledgement number: 1 (relative ack number)

Header length: 20 bytes

Flags: 0x0010 (ACK)

0... .. = Congestion Window Reduced (CWR): Not set

.0... .. = ECN-Echo: Not set

..0. = Urgent: Not set

...1 = Acknowledgment: Set

.... 0... = Push: Not set

.... .0.. = Reset: Not set

.... ..0. = Syn: Not set

.... ...0 = Fin: Not set

Window size: 256960 (scaled)

Checksum: 0x6584 [correct]

4 C→S PSH,ACK

SEQ=X+1, data length=256, next seq=257

ACK=Y+1

Source port: 1684 (1684)

Destination port: 4567 (4567)

Sequence number: 1 (relative sequence number)

[Next sequence number: 257 (relative sequence number)]

Acknowledgement number: 1 (relative ack number)

Header length: 20 bytes

Flags: 0x0018 (PSH, ACK)

0... .. = Congestion Window Reduced (CWR): Not set

.0.. ... = ECN-Echo: Not set

..0. ... = Urgent: Not set

...1 ... = Acknowledgment: Set

.... 1... = Push: Set

.... .0.. = Reset: Not set

.... ..0. = Syn: Not set

.... ...0 = Fin: Not set

Window size: 256960 (scaled)

Checksum: 0x337d [correct]

TCP segment data (256 bytes)

数据内容见下页图

TCP segment data(256 bytes)

这是第一次
发送的数据
123

TCP segment data (256 bytes)																	
0000	00	15	58	2c	40	d8	00	01	6c	2f	21	23	08	00	45	00	..X,@... 1/!#...E.
0010	01	28	47	8e	40	00	40	06	6e	58	c0	a8	01	22	c0	a8	.(G.@.@.nX..."..
0020	01	77	06	94	11	d7	28	74	85	74	54	84	b0	9d	50	18	.w....(t .tT...P.
0030	fa	f0	33	7d	00	00	31	32	33	00	cc	cc	cc	cc	cc	cc	..3}..12 3.....
0040	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc
0050	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc
0060	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc
0070	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc
0080	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc
0090	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc
00a0	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc
00b0	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc
00c0	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc
00d0	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc
00e0	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc
00f0	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc
0100	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc
0110	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc
0120	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc
0130	cc	cc	cc	cc	cc	cc										

5 S→C ACK

SEQ=Y+1

ACK=X+257

第一次传输数据结束

Source port: 4567 (4567)

Destination port: 1684 (1684)

Sequence number: 1 (relative sequence number)

Acknowledgement number: 257 (relative ack number)

Header length: 20 bytes

Flags: 0x0010 (ACK)

0... .. = Congestion Window Reduced (CWR): Not set

.0... .. = ECN-Echo: Not set

..0. = Urgent: Not set

...1..... = Acknowledgment: Set

.... 0... = Push: Not set

.... .0.. = Reset: Not set

.... ..0. = Syn: Not set

.... ...0 = Fin: Not set

Window size: 65279

Checksum: 0x6075 [correct]

6 C→S PSH,ACK

SEQ=X+257, data length=256, next seq=513

ACK=Y+1

Source port: 1684 (1684)

Destination port: 4567 (4567)

Sequence number: 257 (relative sequence number)

[Next sequence number: 513 (relative sequence number)]

Acknowledgement number: 1 (relative ack number)

Header length: 20 bytes

Flags: 0x0018 (PSH, ACK)

0... .. = Congestion window Reduced (CWR): Not set

.0.. .. = ECN-Echo: Not set

..0. = Urgent: Not set

...1 = Acknowledgment: Set

.... 1... = Push: Set

.... .0.. = Reset: Not set

.... ..0. = Syn: Not set

.... ...0 = Fin: Not set

Window size: 256960 (scaled)

Checksum: 0xf946 [correct]

TCP segment data (256 bytes)

数据内容见下页图

TCP segment data(256 bytes)

这是第二次
发送的数据
456

TCP segment data (256 bytes)																
0000	00	15	58	2c	40	d8	00	01	6c	2f	21	23	08	00	45	00
0010	01	28	47	8f	40	00	40	06	6e	57	c0	a8	01	22	c0	a8
0020	01	77	06	94	11	d7	28	74	86	74	54	84	b0	9d	50	18
0030	fa	f0	f9	46	00	00	34	35	36	00	00	00	00	00	00	00
0040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0090	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00a0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00b0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00c0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00d0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00e0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00f0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0100	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0110	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0130	00	00	00	00	00	00										

..X,@... 1/!#..E.
.(G.@.@. nw..."..
.w... (t .tT...P..
...F..45 6.....

7 S→C ACK

SEQ=Y+1

ACK=X+513

第二次传输数据结束

Source port: 4567 (4567)

Destination port: 1684 (1684)

Sequence number: 1 (relative sequence number)

Acknowledgement number: 513 (relative ack number)

Header length: 20 bytes

Flags: 0x0010 (ACK)

0... .. = Congestion Window Reduced (CWR): Not set

.0... .. = ECN-Echo: Not set

..0... .. = Urgent: Not set

...1... .. = Acknowledgment: Set

.... 0... = Push: Not set

.... .0.. = Reset: Not set

.... ..0. = Syn: Not set

.... ...0 = Fin: Not set

window size: 65023

checksum: 0x6075 [correct]

8 C→S FIN,ACK

SEQ=X+513

ACK=Y+1

Source port: 1684 (1684)

Destination port: 4567 (4567)

Sequence number: 513 (relative sequence number)

Acknowledgement number: 1 (relative ack number)

Header length: 20 bytes

Flags: 0x0011 (FIN, ACK)

0... .. = Congestion window Reduced (CWR): Not set

.0.. = ECN-Echo: Not set

..0. = Urgent: Not set

...1 = Acknowledgment: Set

.... 0... = Push: Not set

.... .0.. = Reset: Not set

.... ..0. = Syn: Not set

.... ...1 = Fin: Set

Window size: 256960 (scaled)

Checksum: 0x6383 [correct]

9 S→C ACK

SEQ=Y+1

ACK=X+514

Source port: 4567 (4567)

Destination port: 1684 (1684)

Sequence number: 1 (relative sequence number)

Acknowledgement number: 514 (relative ack number)

Header length: 20 bytes

Flags: 0x0010 (ACK)

0... .. = Congestion Window Reduced (CWR): Not set

.0.. = ECN-Echo: Not set

..0. = Urgent: Not set

...1 = Acknowledgment: Set

.... 0... = Push: Not set

.... .0.. = Reset: Not set

.... ..0. = Syn: Not set

.... ...0 = Fin: Not set

Window size: 65023

Checksum: 0x6074 [correct]

10 S→C FIN,ACK

SEQ=Y+1

ACK=X+514

Source port: 4567 (4567)

Destination port: 1684 (1684)

Sequence number: 1 (relative sequence number)

Acknowledgement number: 514 (relative ack number)

Header length: 20 bytes

Flags: 0x0011 (FIN, ACK)

0... .. = Congestion window Reduced (CWR): Not set

.0.. = ECN-Echo: Not set

..0. = Urgent: Not set

...1 = Acknowledgment: Set

.... 0... = Push: Not set

.... .0.. = Reset: Not set

.... ..0. = Syn: Not set

.... ...1 = Fin: Set

Window size: 65023

Checksum: 0x6073 [correct]

11 C→S ACK

SEQ=X+514

ACK=Y+2

TCP连接已经断开

Source port: 1684 (1684)

Destination port: 4567 (4567)

Sequence number: 514 (relative sequence number)

Acknowledgement number: 2 (relative ack number)

Header length: 20 bytes

Flags: 0x0010 (ACK)

0... .. = Congestion Window Reduced (CWR): Not set

.0... .. = ECN-Echo: Not set

..0. = Urgent: Not set

...1 = Acknowledgment: Set

.... 0... = Push: Not set

.... .0.. = Reset: Not set

.... ..0. = Syn: Not set

.... ...0 = Fin: Not set

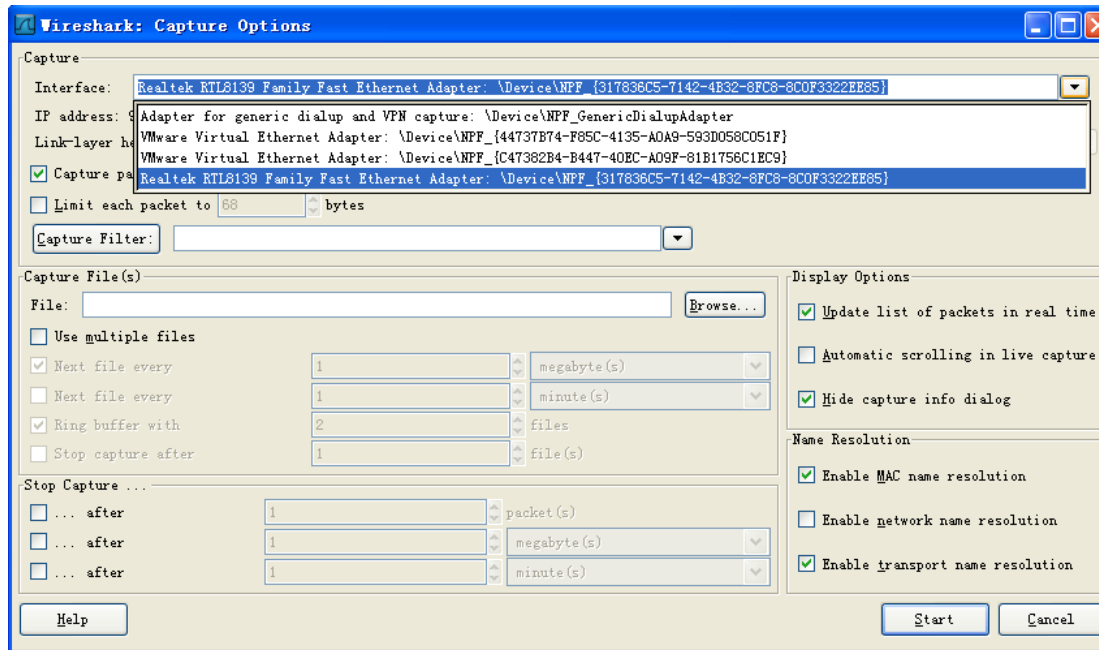
Window size: 256960 (scaled)

Checksum: 0x6382 [correct]

案例三：观察登录BBS过程

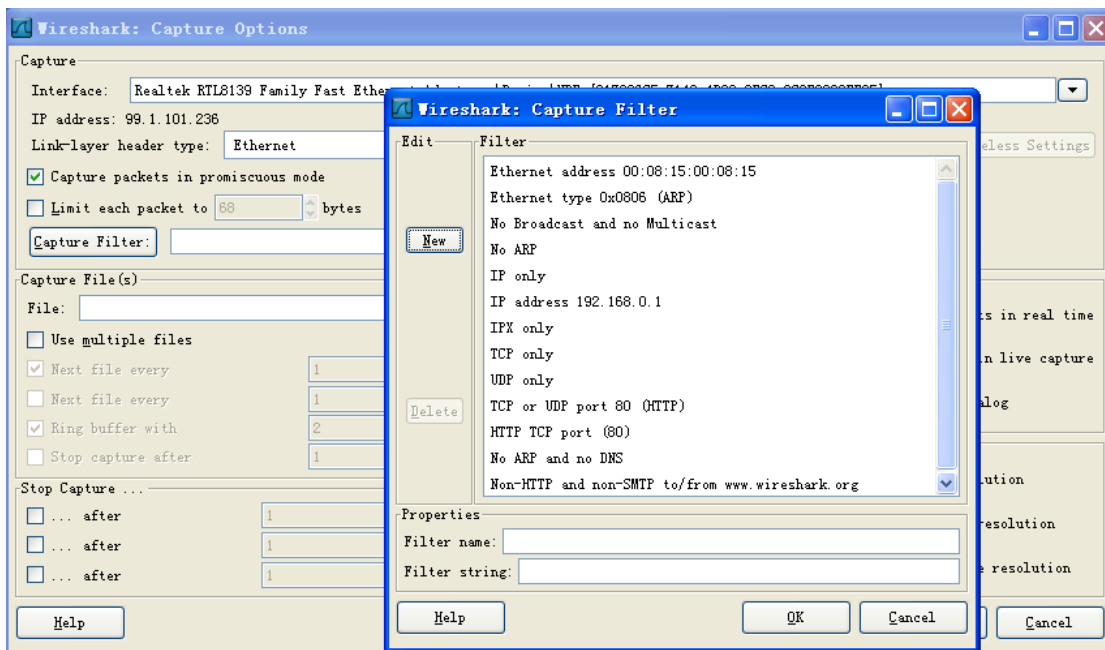
□ 设置网卡

- 如果有多个网络接口（网卡），首先在Capture Options中设置在哪个网络接口上抓包。勾选Capture packets in promiscuous mode选项，将网卡设置成混杂模式。



案例三：观察登录BBS过程

□ 设置过滤条件：捕获前过滤



案例三：观察登录**BBS**过程

- 设置过滤条件：捕获后过滤
- 如果**Filter**框背景显示为绿色，说明所设定的过滤规则合乎**Wireshark**支持的语法规则。



- 如果**Filter**框背景显示为红色，说明所设定的过滤规则不符合语法规则。



主机向服务器发送的POST请求





3.3 监听的防御

- **3.3.1 通用策略**
- **3.3.2 共享网络下的防监听**
- **3.3.3 交换网络下的防监听**



3.3.1 通用策略

- 由于嗅探器是一种被动攻击技术，因此非常难以被发现。
- 完全主动的解决方案很难找到并且因网络类型而有一些差异，但我们可以先采用一些被动但却是通用的防御措施。
- 这主要包括采用安全的网络拓扑结构和数据加密技术两方面。此外要注意重点区域的安全防范。

安全的拓扑结构

- 网络分段越细，嗅探器能够收集的信息就越少。
- **网络分段**：将网络分成一些小的网络，每一个网段的集线器被连接到一个交换器 (**Switch**) 上，所以数据包只能在该网段的内部被网络监听器截获，这样网络的剩余部分（不在同一网段）便被保护了。网络有三种网络设备是嗅探器不可能跨过的：交换机、路由器、网桥。我们可以通过灵活的运用这些设备来进行网络分段。
- **划分VLAN**：使得网络隔离不必要的数据传送，一般可以采用**20**个工作站为一组，这是一个比较合理的数字。网络分段只适应于中小的网络。网络分段需要昂贵的硬件设备。

数据加密

- **数据通道加密**：正常的数据都是通过事先建立的通道进行传输的，以往许多应用协议中明文传输的账号、口令的敏感信息将受到严密保护。目前的数据加密通道方式主要有**SSH**、**SSL(Secure Socket Layer**，安全套接字应用层)**和VPN**。
- **数据内容加密**：主要采用的是将目前被证实的较为可靠的加密机制对对互联网上传输的邮件和文件进行加密。如**PGP**等。



3.3 监听的防御

- **3.3.1** 通用策略
- **3.3.2** 共享网络下的防监听
- **3.3.3** 交换网络下的防监听

3.3.2 共享网络下的防监听

- 虽然共享式局域网中的嗅探很隐蔽，但也有一些方法来帮助判断：
 - 检测处于混杂模式的网卡
 - 网络通讯丢包率非常高
 - 网络带宽出现反常
- 检测技术
 - 网络和主机响应时间测试
 - ARP检测（如AntiSniff 工具）

3.3.2 共享网络下的防监听

□ 1. 网络和主机响应时间测试

- 这种检测已被证明是最有效的，它能够发现网络中处于监听模式的机器，而不管其操作系统是什么。

3.3.2 共享网络下的防监听

□ 1. 网络和主机响应时间测试

- **测试原理**是处于非监听模式的网卡提供了一定的硬件底层过滤机制，即目标地址为非本地(广播地址除外)的数据包将被网卡所丢弃。
- 这种情况下骤然增加目标地址不是本地的网络通讯流量对操作系统的影响很小。
- 而处于混杂模式下的机器则缺乏底层的过滤，骤然增加目标地址不是本地的网络通讯流量会对该机器造成较明显的影响(不同的操作系统/内核/用户方式会有不同)。

3.3.2 共享网络下的防监听

□ 1. 网络和主机响应时间测试

- **实现方法**是利用ICMP ECHO请求及响应计算出需要检测机器的响应时间基准和平均值。
- 在得到这个数据后，立刻向本地网络发送大量的伪造数据包，与此同时再次发送测试数据包以确定平均响应时间的变化值。
- 非监听模式的机器的响应时间变化量会很小，而监听模式的机器的响应时间变化量则通常会有1~4个数量级。

3.3.2 共享网络下的防监听

□ 2. ARP检测

- 地址解析协议(Address Resolution Protocol,ARP)请求报文用来查询硬件地址到IP地址的解析。适用于所有基于以太网的IPV4协议。
- 我们可以使用这类分组来校验网卡是否被设置为混杂模式。

3.3.2 共享网络下的防监听

□ 2. ARP检测

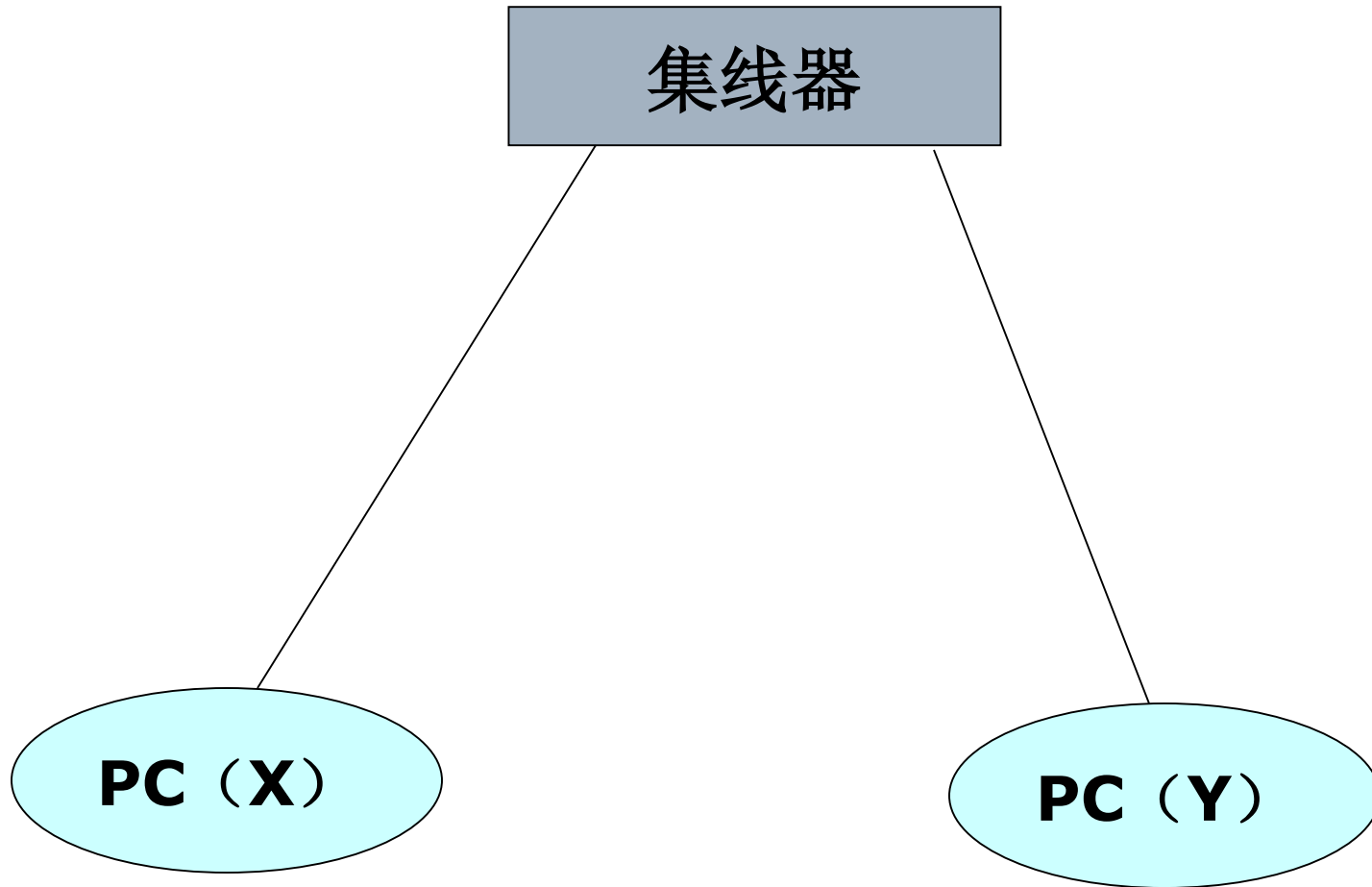
在混杂模式下，网卡不会阻塞目的地址不是自己的分组，而是照单全收，并将其传送给系统内核。然后，系统内核会返回包含错误信息的报文。

基于这种机制，我们可以假造一些**ARP**请求报文发送到网络上的各个节点，没有处于混杂模式的网卡会阻塞这些报文，但是如果某些节点有回应，就表示这些节点的网卡处于混杂模式下。这些处于混杂模式的节点就可能运行嗅探器程序。

3.3.2 共享网络下的防监听

□ ARP检测原理

- 例如：网络上一台IP地址为192.168.1.1的PC(X)以太网地址是00-00-00-00-00-01，这台PC(X)需要向网络上另外一台IP地址为192.168.1.10的PC(Y)发送消息。



IP: 192.168.1.1

MAC: 00-00-00-00-00-01

IP: 192.168.1.10

ARP检测原理(2)

- 在发送之前，**X**首先发出一个**ARP**请求包查询**192.168.1.10**对应的以太网地址。查询包的目的地地址被设置为**FF-FF-FF-FF-FF-FF**(广播)，从而本地网络上的所有节点都可以收到这个包。
- 收到之后，每个节点会检查这个**ARP**包查询的**IP**地址和本机的**IP**地址是否匹配。如果不同，就忽略这个**ARP**包；如果匹配(**Y**)就向**X**发出应答。
- **X**收到应答之后就缓存**Y**的**IP**/硬件地址。然后，**X**就可以向**Y**发送实际的数据。

ARP检测原理(3)

- 进一步设想，如果我们把这个查询包的目的地地址(以太网地址)设置为另外的地址,而不是原来的广播地址又将如何?

ARP检测原理(4)

- ❑ 例如：我们把查询包的目的地地址设置为**00-00-00-00-00-01**会发生什么？
- ❑ 处于正常模式下网络节点的以太网卡会认为这个查询包是发往其它主机的，其硬件过滤器会拒绝接收这个包；然而，如果这个网络节点**(192.168.1.10)**的以太网卡处于混杂模式**(promiscuous mode)**下，那么即使以太网地址不匹配，其硬件过滤器也不进行任何过滤，从而使这个查询包能够进入到系统的内核。
- ❑ 因为这个节点的**IP**地址和查询包的要查询**IP**地址相同，其内核就会认为**ARP**查询包到达，应该作出应答。

ARP检测原理(5)

- 但是，在不同的操作系统中，这个处于混杂模式节点的内核可能不会应答**ARP**查询包。
- 这是因为这个包被系统内核过滤掉了。在这里我们把这叫作软件过滤器。
- 所以这个方法并不是完全奏效，但是此方法简单实用。

ARP检测原理(6)

- 概括地说，我们可以通过区别硬件过滤器和软件过滤器的不同特征来检测处于混杂模式的网络节点。我们需要构造应该被硬件过滤器阻塞，但是却能够通过软件过滤器的报文。如果把这种报文送到各个网络节点，那么处于普通模式下的网络节点将不做应答；而处于混杂模式的节点会进行应答。

3.3 监听的防御

- **3.3.1** 通用策略
- **3.3.2** 共享网络下的防监听
- **3.3.3** 交换网络下的防监听

3.3.3 交换网络下的防监听

- 交换网络下防监听，主要要防止**ARP**欺骗及**ARP**过载。如何防范**ARP**欺骗将在后续章节讲述。**ARP**过载则是指通过发送大量**ARP**数据包使得交换设备出现信息过载，工作于广播模式。
- 交换网络下防范监听的措施主要包括：
 - 不要把网络安全信任关系建立在单一的IP或MAC基础上，理想的关系应该建立在IP-MAC对应关系的基础上。
 - 使用静态的**ARP**或者IP-MAC对应表代替动态的**ARP**或者IP-MAC对应表，禁止自动更新，使用手动更新。
 - 定期检查**ARP**请求，使用**ARP**监视工具，例如**ARPWatch**等监视并探测**ARP**欺骗。
 - 制定良好的安全管理策略，加强用户安全意识。

3.4 小结

最普遍同时也是最致命的安全威胁往往来自内部，其破坏性也远大于外部威胁。

其中网络嗅探对于一般的网络来说，威胁巨大。因此很多黑客也使用嗅探器进行网络入侵渗透。

网络嗅探器对信息安全的威胁来自其被动性和非干扰性，使得网络嗅探具有很强的隐蔽性，往往让网络信息泄密变得不容易被发现。

本节课分析了网络嗅探的原理，并提出一些防范措施。但对于用户而言除了技术手段以外，最重要的还是建立相应的安全意识，注意对隐私信息的加密保护。



Thank you for your attention!

