

第3章 网络监听及防御技术

国家计算机网络入侵防范中心

张玉清



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



引入：

网络监听在安全领域引起人们普遍注意是在**1994**年开始的。在那一年**2**月间，相继发生了几次大的安全事件，一个不知名的人在众多的主机和骨干网络设备上安装了网络监听软件，利用它对美国骨干互联网和军方网窃取了超过**100,000**个有效的用户名和口令。

上述事件可能是互联网上最早期的大规模的网络监听事件了，它使早期网络监听从"地下"走向了公开，并迅速的在大众中普及开来。

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只限于局域网。

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

Frame (184 bytes) | Reassembled TCP (1618 bytes)

File: "etherXXXJ2N4XT" 67 KB 00:00:15 | P: 212 D: 39 M: 0 Drops: 0

..Z..K.. 6....E.
....@... n&...v.1
+.\$..P.. ..}...!P.
...m..Co ntent-Ty
pe: appl ication/
x-www-fo rm-urlen
coded..C ontent-L
ength: 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：上届学生实验编写的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。

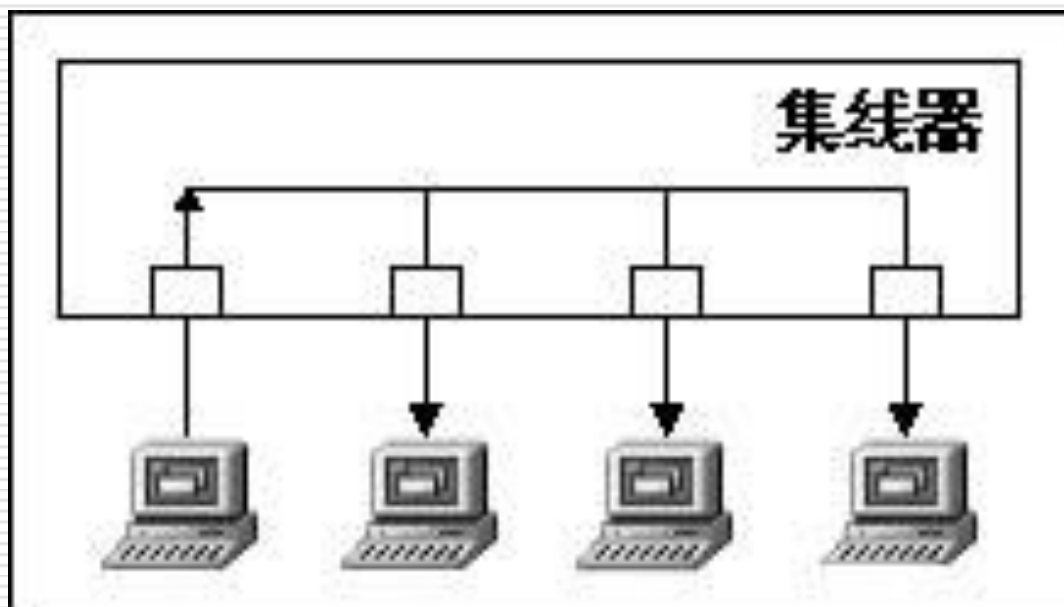


D-LINK DES-1024D 24PORT

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。

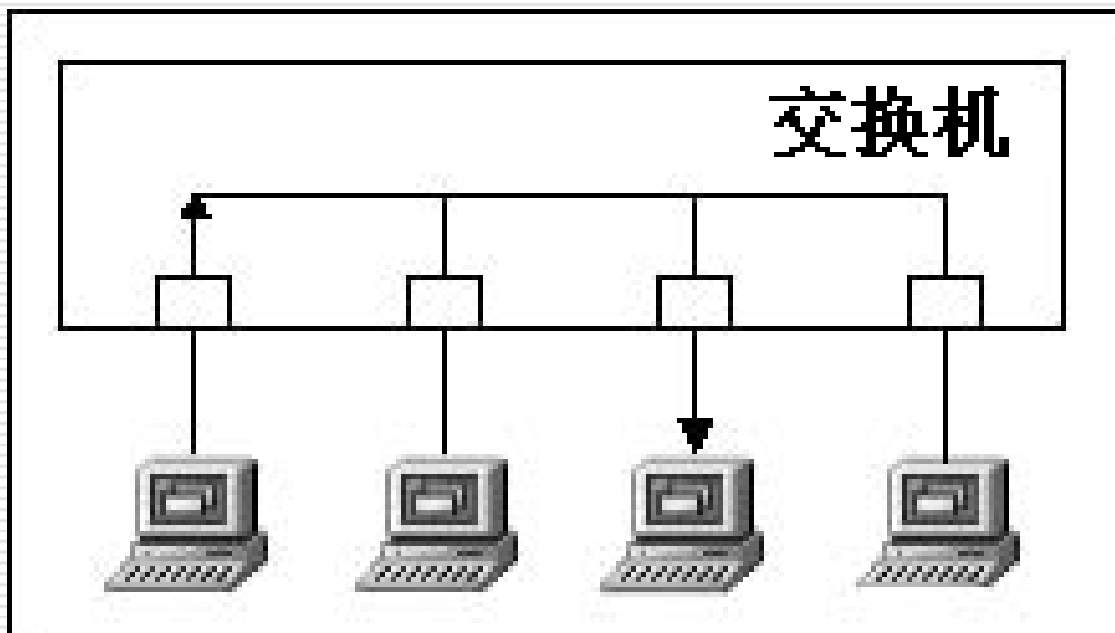


CISCO WS-C2950-24交换机

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）。

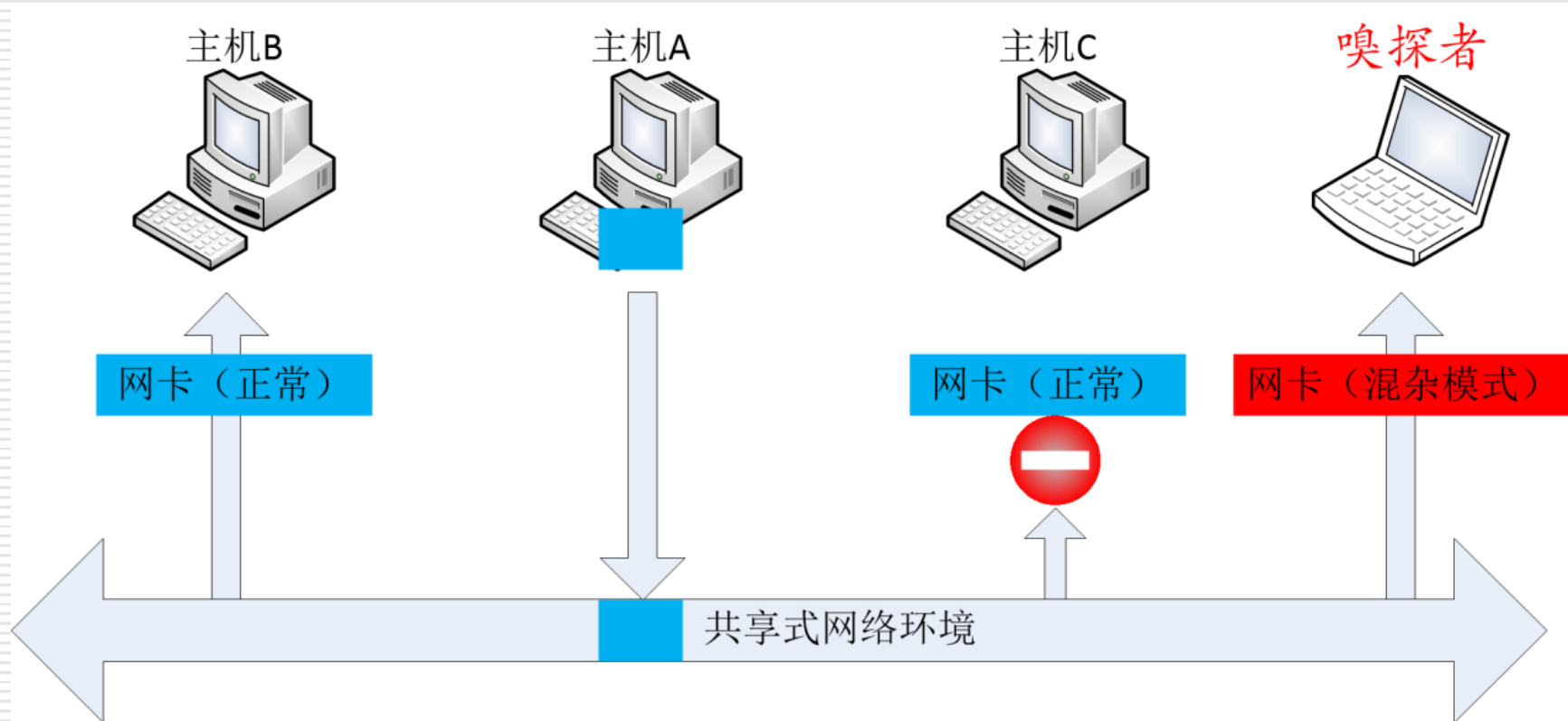
共享式局域网的监听的工作原理(2)

- 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

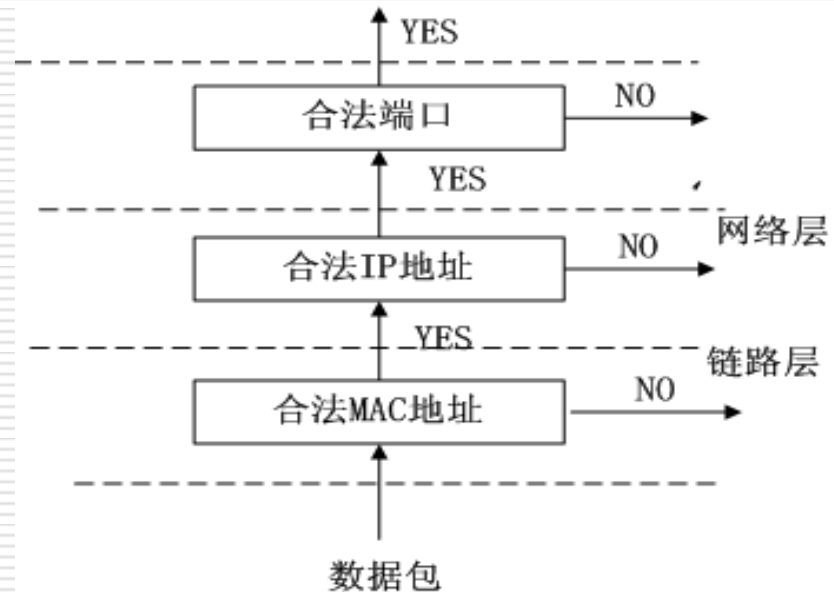
- 在共享式局域网中，集线器会广播所有数据，这时，如果局域网中一台主机将网卡设置成混杂模式，那么它就可以接收到该局域网中的所有数据了。
- 网卡在混杂模式工作的情况下，所有流经网卡的数据帧都会被网卡驱动程序上传给网络层。
- 共享式局域网监听示意图见下页。

共享式局域网监听示意图



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)

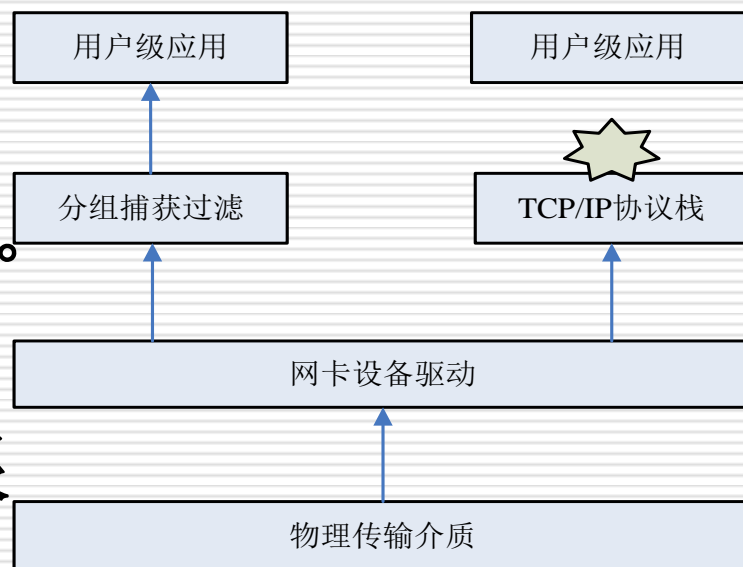
- ❑ 链路层过滤：判断数据包的目的**MAC**地址。
- ❑ 网络层过滤：判断数据包的目的**IP**地址。
- ❑ 传输层过滤：判断对应的目的端口是否在本机已经打开。
- ❑ 因而，如果没有一个特定的机制，上层应用也无法抓到本不属于自己的“数据包”。

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。

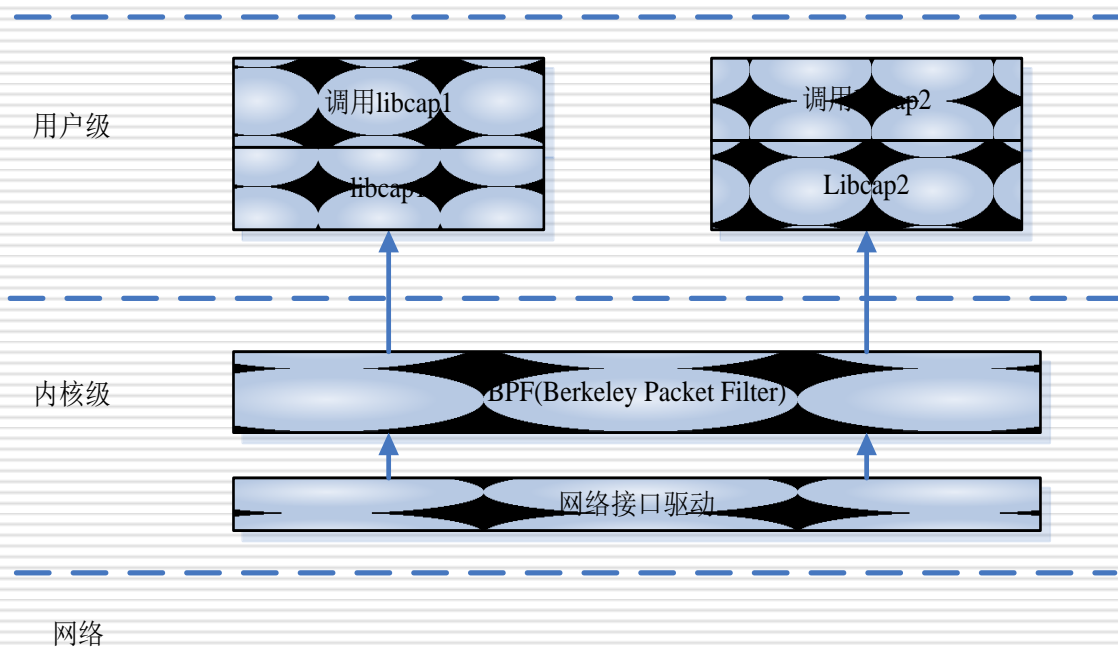


共享式局域网的监听实现方法(6)

- 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。

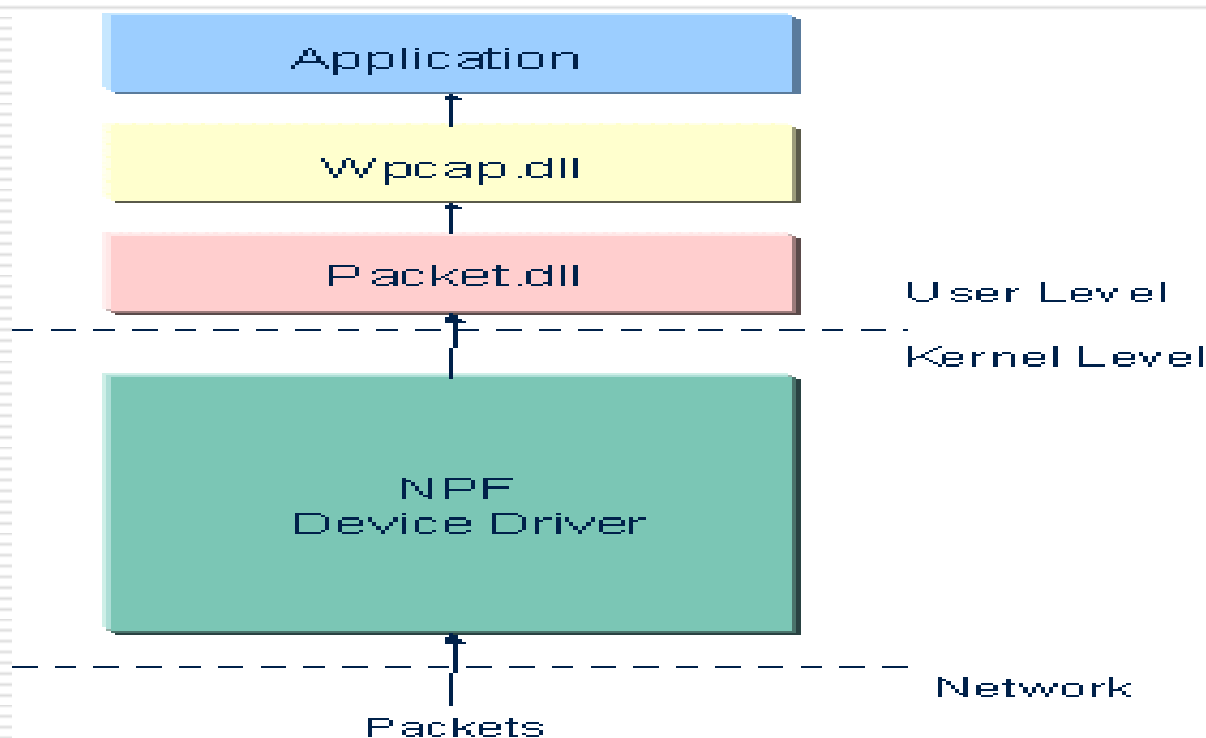
相关开发库

□ 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在不同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

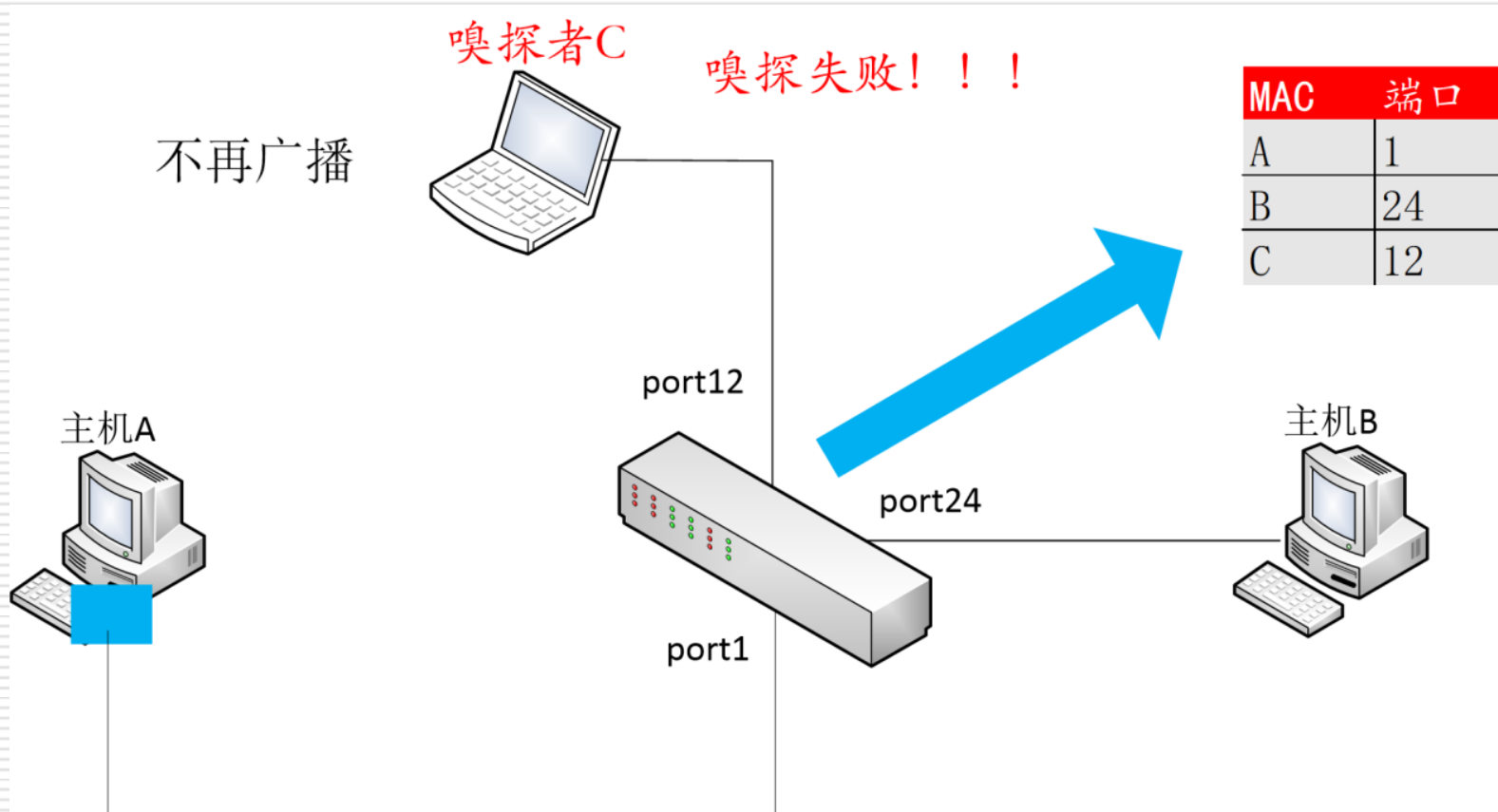
□ 产生交换式局域网的原因：

- 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

- 所有主机连接到交换机，对于发给某个特定主机的数据包会被交换机从特定的端口送出，而不是广播给网络中的所有主机。这种传输形式使得以太网的性能大大提高，但是也破坏了监听的第一条件，即其他的主机即使将网卡设置在混杂模式，也只能收到广播帧和目的地址是本机的帧，因而无法进行监听。
- 交换式网络环境下监听失败示意图见下页。

交换式网络环境下监听失败示意图



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

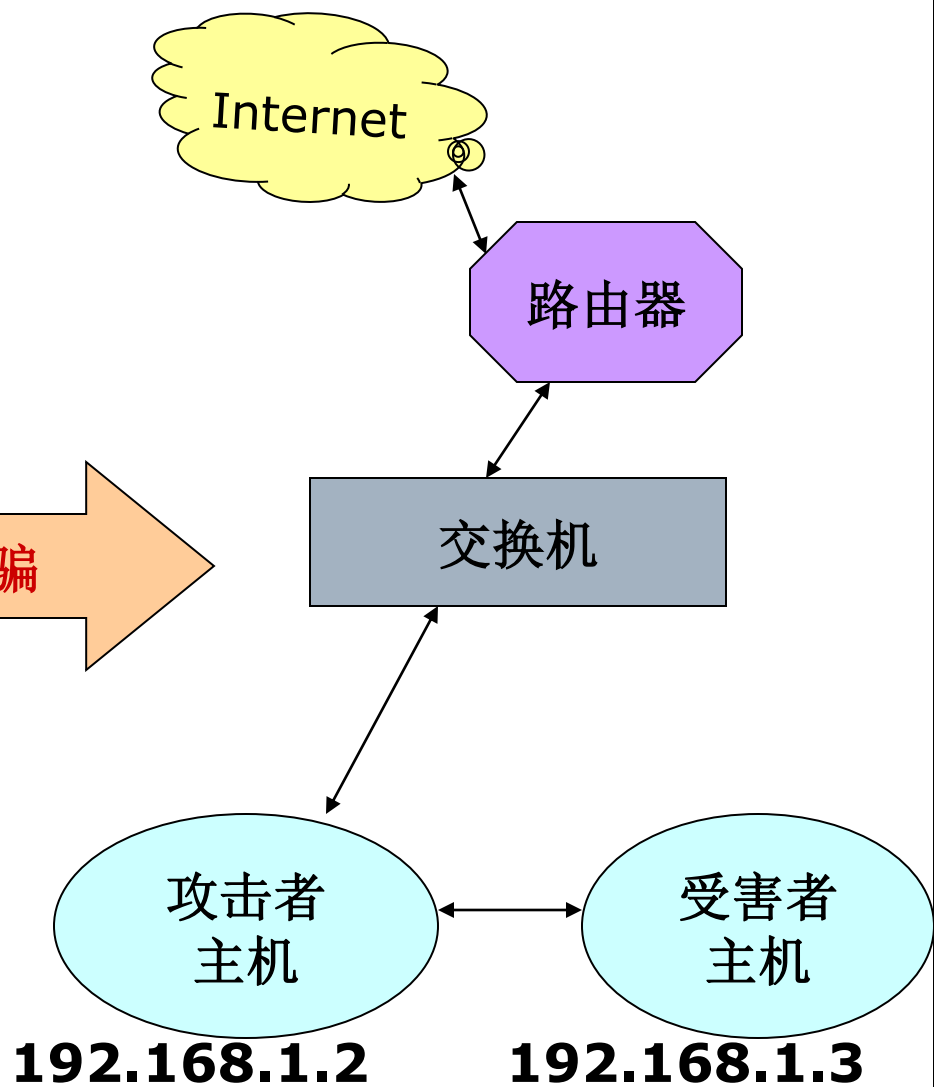
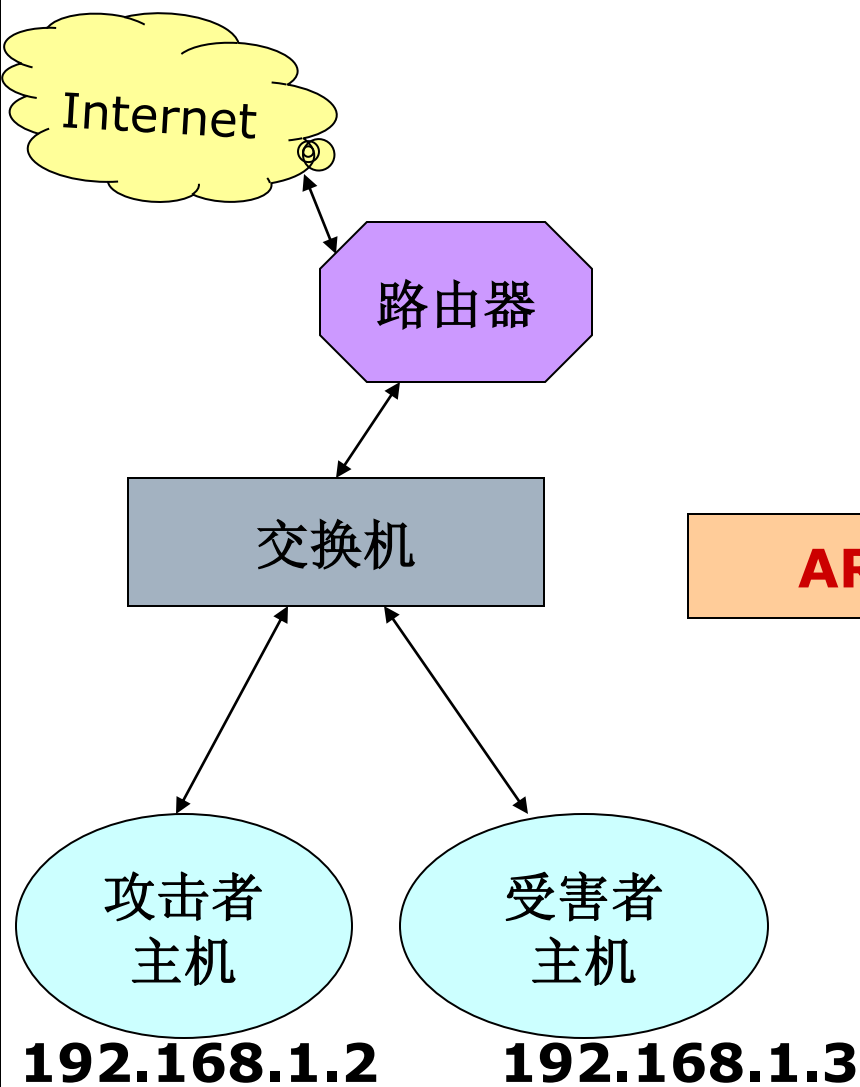
□ 溢出攻击

- 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。
- 主动监听软件 **dsniff**

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（终结版本**0.99.0**）：
<http://www.Ethereal.com/>
- ❑ **Wireshark**官网：
<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

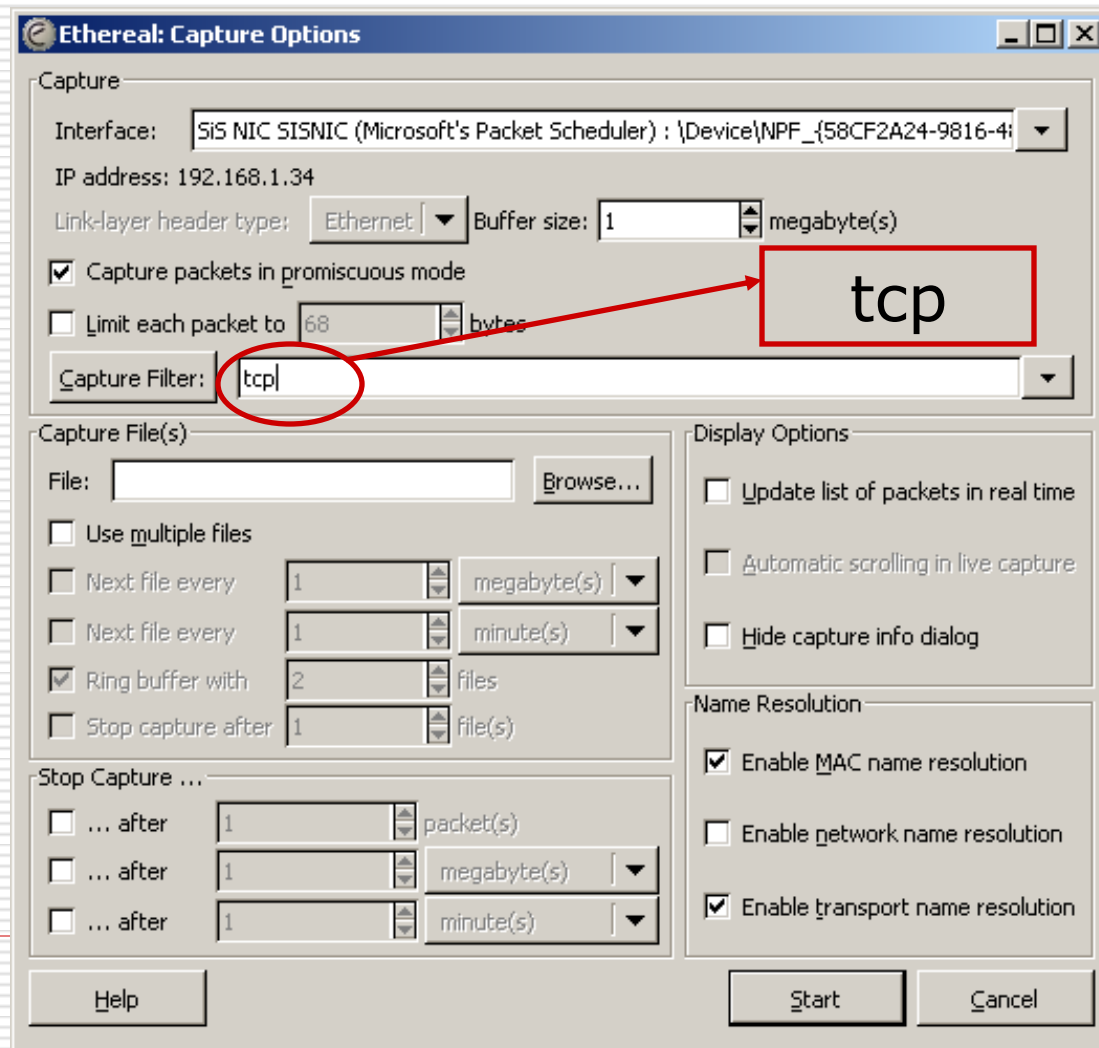
Display filter

- ❑ 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）

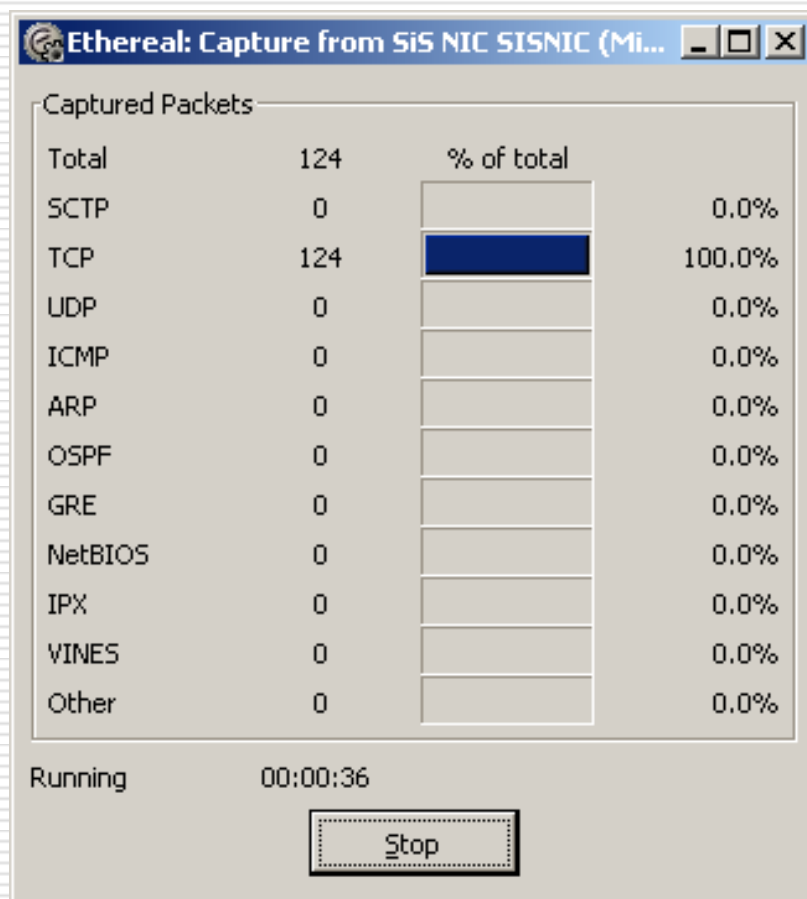
案例一：观察**FTP**数据流

- ❑ 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



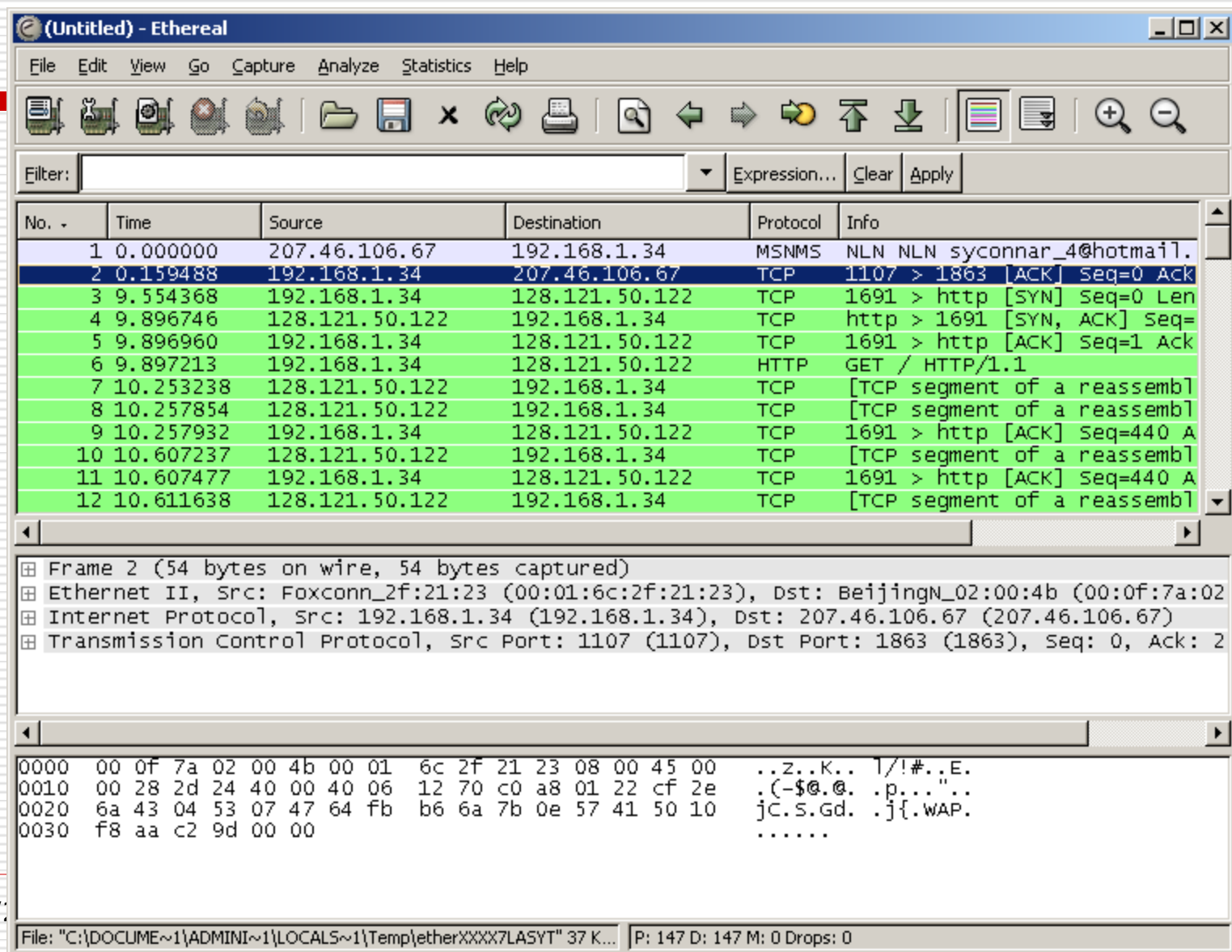
抓包结束，查看封包内容

控制列

封包总览

封包内容

十六进制码



设置Display filter

Port 21 and ip.addr==192.168.1.250

The screenshot shows the Wireshark interface with the display filter 'port 21 and ip.addr==192.168.1.250' applied. The packet list shows several FTP and TCP packets between 192.168.1.250 and 192.168.1.34. The packet details pane shows the selected packet (Frame 102) and its structure: Ethernet II, Internet Protocol, and Transmission Control Protocol. The packet bytes pane shows the raw data in hexadecimal and ASCII.

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)

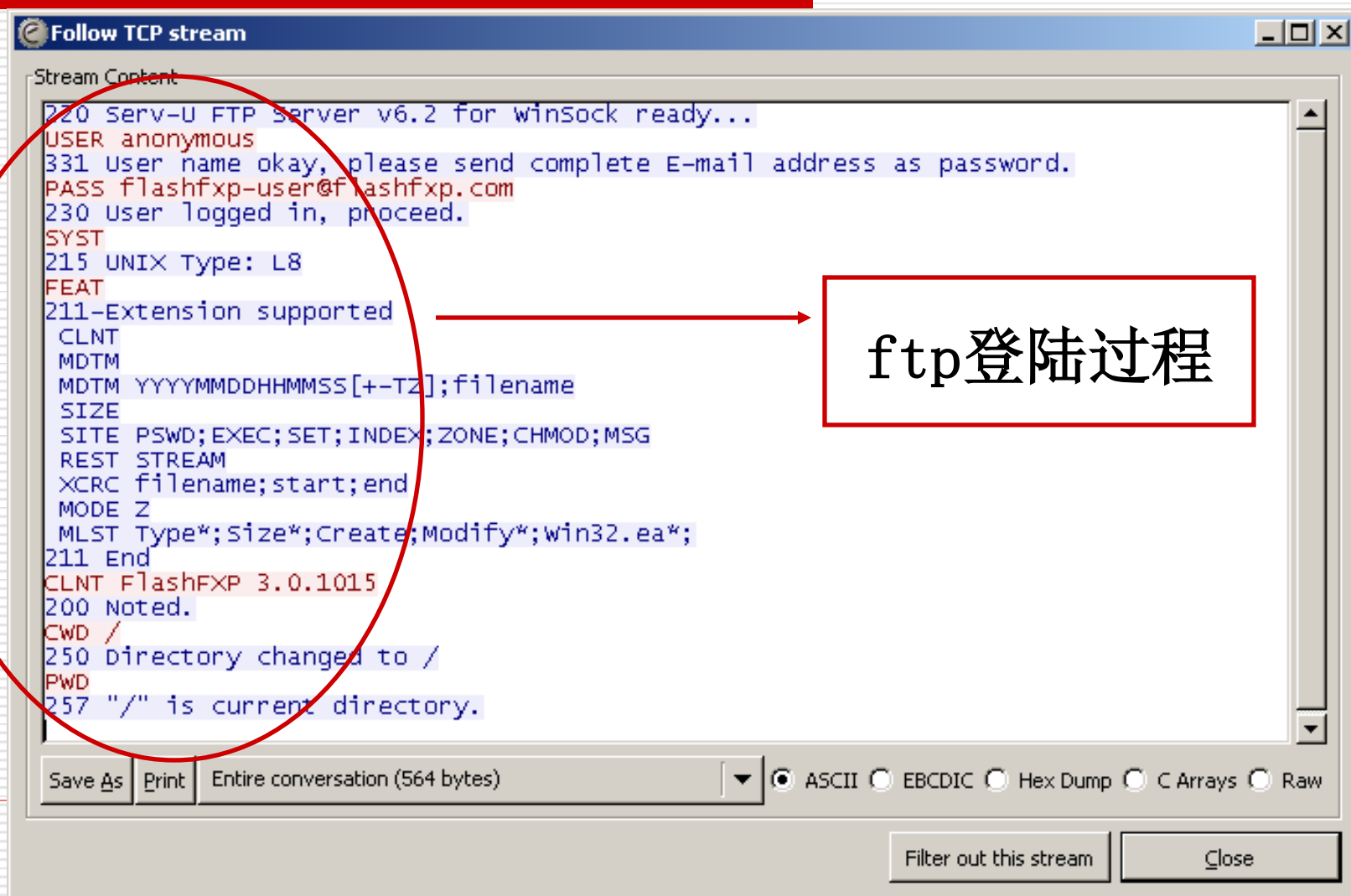
Transmission Control Protocol, Src Port: ftp (21), Dst Port: 1701 (1701), Seq: 1, Ack: 1, Len: 0

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\LOCALS~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**

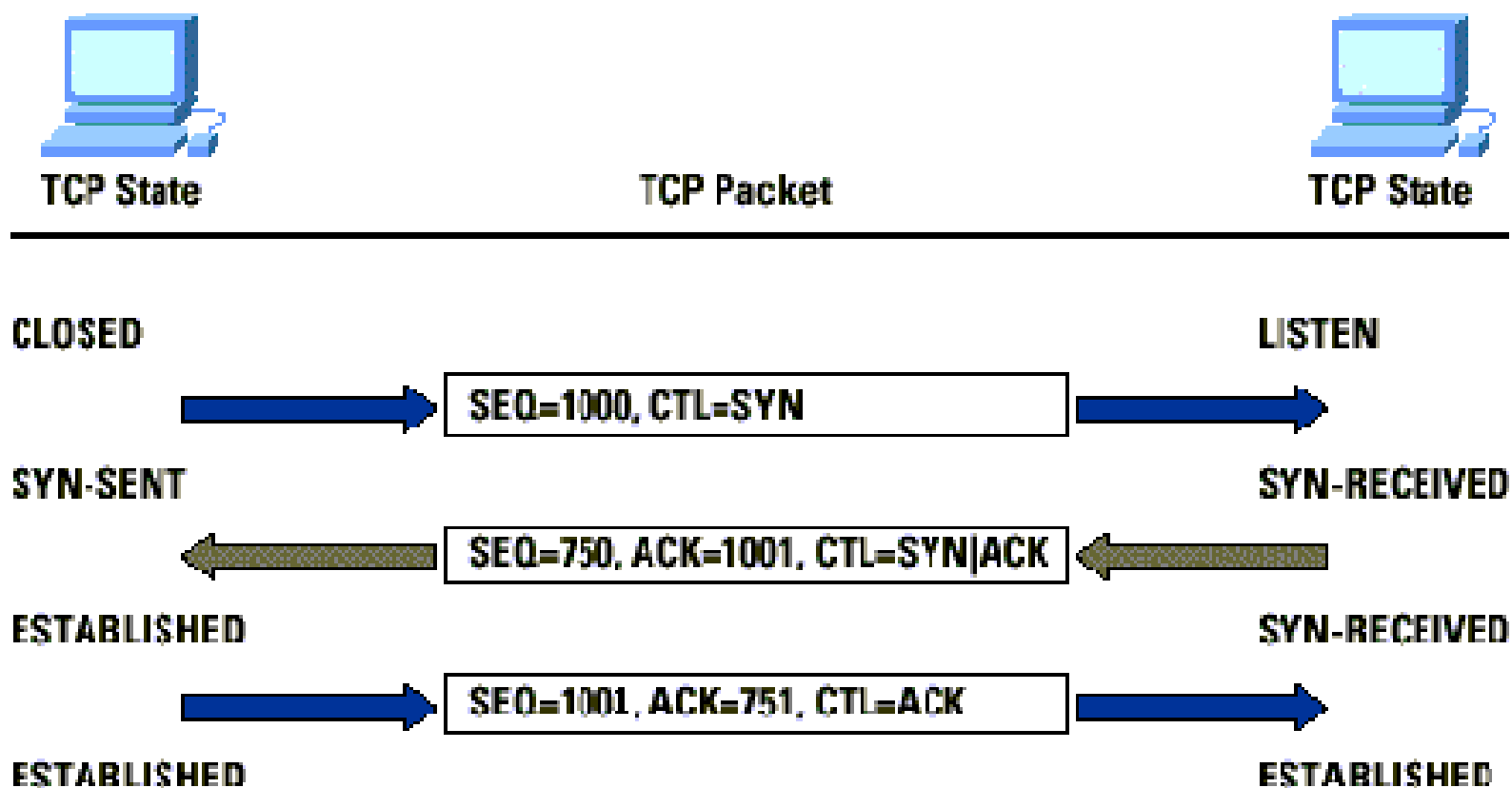
□ **正常TCP通信过程:**

- 建立连接
- 数据传输
- 断开连接

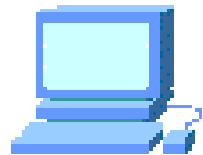
TCP数据报格式

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

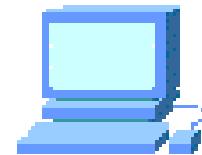


TCP数据传输过程



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

FIN-WAIT-1 → **SEQ=1513,ACK=751,CTL=FIN|ACK** → CLOSE-WAIT

FIN-WAIT-2 ← **SEQ=751,ACK=1514,CTL=ACK** ← CLOSE-WAIT

TIME-WAIT ← **SEQ=751,ACK=1514,CTL=FIN|ACK** ← LAST-ACK

TIME-WAIT → **SEQ=1514,ACK=752,CTL=ACK** → CLOSED

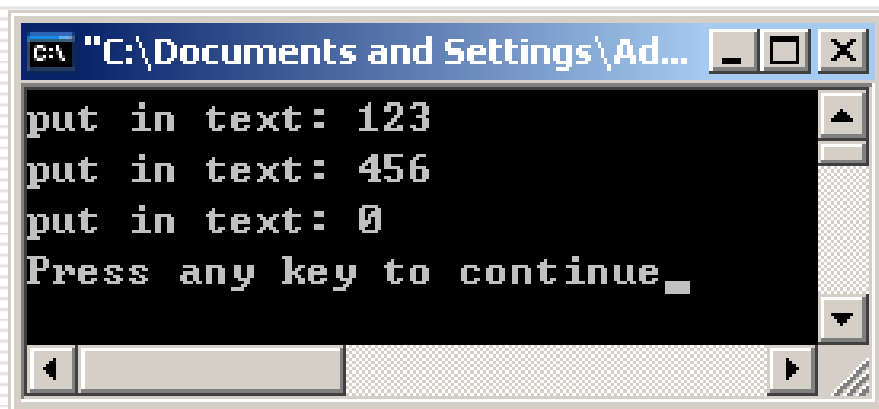
CLOSED

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**

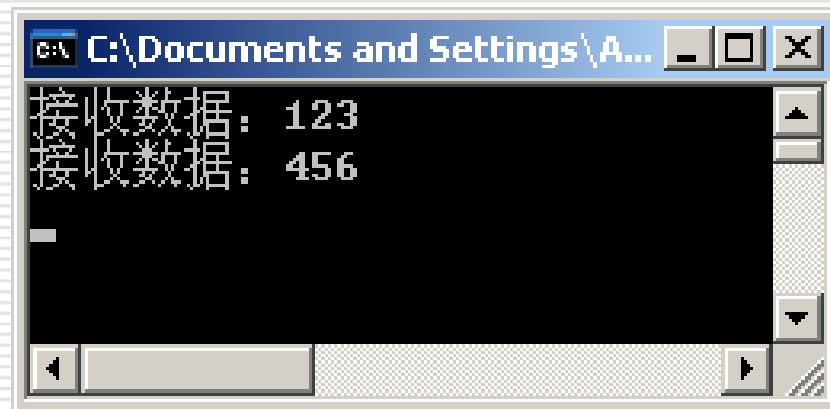
实验环境（2）

- ❑ **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 wir
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 v
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 v
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 v
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 v

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)

[illegible]

```

..X,@... 1/!#..E.
.(G.@.@. nX... "
.w....(t .tT...P.
.3}..12 3.

```

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

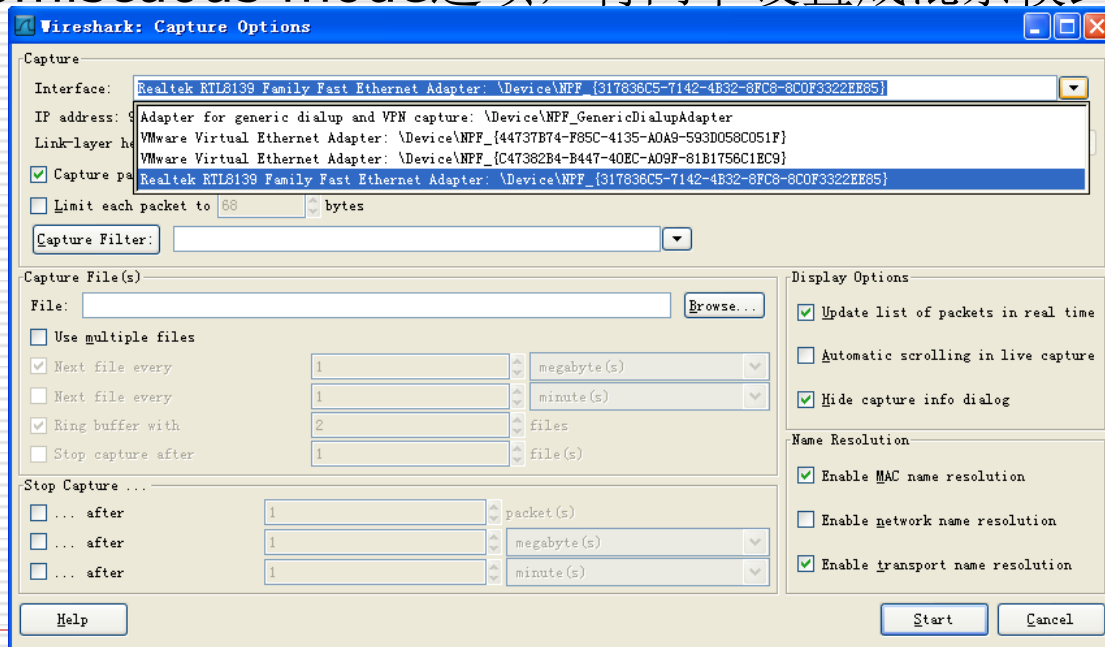
Window size: 256960 (scaled)

Checksum: 0x6382 [correct]

案例三：观察登录BBS过程

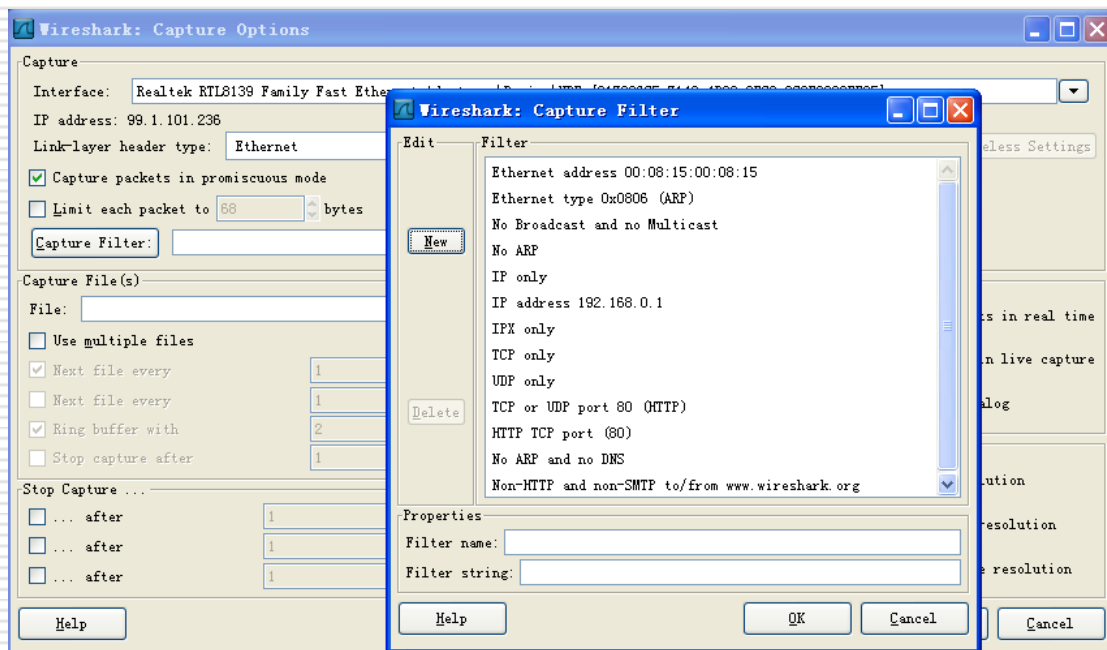
□ 设置网卡

- 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。



[illegible]

215546885.107240654.1216623809.1216623809.
:username=caojq9902&password=chen86751537

dsniff

- ❑ **dsniff**简介：基于**unix**系统网络嗅探工具，是一系列密码嗅探和网络流量分析工具，用来解析不同的应用协议并提取相关信息，是网络安全审计和渗透测试工具集。
 - arpspoof 指定目标的 arp 欺骗重定向
 - dnsspoof 伪造 DNS 响应消息
 - dsniff 口令嗅探
 - filesnarf NFS 文件流截获 dump
 - ...

dsniff

❑ **arpspoof** 指定目标的**arp**欺骗重定向

❑ 下载地址

<https://www.monkey.org/~dugsong/dsniff/>

❑ 命令及其参数

■ `foo$ arpspoof [-i interface] [-t target] host`

■ `-i interface` 网卡接口；

■ `-t target`指明**arp**欺骗的目标主机，如果不指明，则对所有主机有效；**host** 指明有要截获数据包的主机（经常是本地路由）。

dsniff

❑ arpspoof 指定目标的arp欺骗重定向

```
RX errors 0  dropped 0  overruns 0  frame 0
TX packets 38  bytes 2490 (2.4 KiB)
TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0
device interrupt 19  base 0x2000

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
    inet 127.0.0.1  netmask 255.0.0.0
    inet6 ::1  prefixlen 128  scopeid 0x10<host>
    loop txqueuelen 1  (Local Loopback)
    RX packets 33  bytes 2572 (2.5 KiB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 33  bytes 2572 (2.5 KiB)
    TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0

root@kali:~# arpspoof -i eth0 -t 192.168.1.112 192.168.1.1
0:c:29:96:80:d7 ac:b5:7d:51:4c:c0 0806 42: arp reply 192.168.1.1 is-at 0:c:29:96:80:d7
0:c:29:96:80:d7 ac:b5:7d:51:4c:c0 0806 42: arp reply 192.168.1.1 is-at 0:c:29:96:80:d7
0:c:29:96:80:d7 ac:b5:7d:51:4c:c0 0806 42: arp reply 192.168.1.1 is-at 0:c:29:96:80:d7
0:c:29:96:80:d7 ac:b5:7d:51:4c:c0 0806 42: arp reply 192.168.1.1 is-at 0:c:29:96:80:d7
```


dsniff

- ❑ **dsniff:** 一个密码侦测工具，他能够自动分析端口上收到的某些协议的数据包，并获取相应的密码。

```
[root@MidLAMP ~]# dsniff -i eth2
dsniff: listening on eth2
-----
10/25/17 00:58:30 tcp 192.168.3.8.49613 -> 192.168.3.23.21 (ftp)
USER klion
PASS admin
-----
10/25/17 00:59:08 tcp 192.168.3.8.49643 -> 192.168.3.23.23 (telnet)
administrator
```

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**
- **3.3.4 防范ARP欺骗**



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在互联网上传输的邮件和文件进行加密。如**PGP**等。

对可能存在的网络监听的检测

□ 对可能存在的网络监听的检测

- 对于怀疑运行监听程序的机器，用正确的IP地址和错误的物理地址ping，运行监听程序的机器会有响应。这是因为正常的机器不接收错误的物理地址，处理监听状态的机器能接收，但如果他的IPstack不再次反向检查的话，就会响应。
- 向网上发大量不存在的物理地址的包，由于监听程序要分析和处理大量的数据包会占用很多的CPU资源，这将导致性能下降。通过比较前后该机器性能加以判断。这种方法难度比较大。
- 使用反监听工具如antisniffer等进行检测。

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
- **3.3.4** 防范**ARP**欺骗

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

□ 实现**ARP**检测

- 由于在正常模式下，主机只接收目的 **MAC** 地址是自己的数据包，其他将一律丢弃；而在**混杂模式**下，网卡并不检查目的 **MAC** 地址，对所有的数据包都来者不拒。
- 所以，我们只需要构造并发送一个：疑似待排查为混杂模式目标主机的 **IP** 目的地址和与其真实 **MAC** 地址不同的**虚假 **MAC** 目的地址 ICMP 回显请求 (echo request)** 数据包。如果收到应答，则说明该被排查的目标主机网卡当前正工作于混杂模式，即当前网络中可能存在监听者。
- 检测共享式网络环境中的监听者示意图见下页

3.3.2 共享网络下的防监听

□ 实现ARP检测



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
- **3.3.4** 防范**ARP**欺骗

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，内存溢出从而工作于广播模式。
- 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。

3.3 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
- **3.3.4** 防范**ARP**欺骗

3.3.4 防范**ARP**欺骗

- ❑ 针对接入设备的**ARP**攻击三种防范措施
- ❑ **ARP**报文检测措施
- ❑ **ARP**网关过滤保护措施
- ❑ **ARP**报文限速措施

3.3.4 防范ARP欺骗

□ ARP报文检测措施

- ARP报文检测措施是非常有效的防范ARP网关攻击和主机攻击的重要手段之一。
- 原理是当一个VLAN（虚拟局域网）内开启了ARP报文检测功能后，该虚拟局域网内任何端口所接收到的ARP请求或者应答包的报文都会被重定向到主机系统，并对该报文的用户合法性及报文有效性进行全面检测。当检测结果认为ARP报文合法时，则进行下一步转发，否则直接丢弃该ARP报文。
- ARP报文检测措施的主要方法有对ARP报文进行有效性检测、对用户合法性进行检测以及对ARP报文进行强制转发。

3.3.4 防范ARP欺骗

□ ARP网关过滤保护措施

- ARP网关过滤保护措施包括两个部分，一是ARP网关保护，二是ARP过滤保护。这两种保护措施实现起来比较方便，只需通过具有过滤与保护功能的设备进行相关的功能配置即可。

□ ARP报文限速措施

- 由于在ARP欺骗攻击过程中，主要利用了ARP协议的当初设计缺陷，根据ARP欺骗攻击的原理，可以看出光靠对ARP报文的有效性检查还不够，还需要在网络设备端口采取一定的限速措施，使某个端口一旦受到攻击，就暂时采取关闭动作，从而避免网络带宽资源和网络设备的CPU资源被ARP攻击耗尽。

3.3.4 防范**ARP**欺骗

- ❑ 针对网关设备的**ARP**攻击三种防范措施
- ❑ 授权配置**ARP**缓存表的防范措施
- ❑ **ARP**主动确认的防范措施
- ❑ **ARP**报文限速措施。

3.3.4 防范ARP欺骗

□ 授权配置ARP缓存表的防范措施

- 多数情况下，ARP欺骗攻击是通过非法或错误地修改ARP缓存表来造成的。如果将ARP缓存表的修改或配置进行授权，则可以在一定程度上避免这种情况。这种授权配置ARP缓存表的方法适合于采用DHCP协议进行主机IP地址动态分配的网络环境中。

□ ARP主动确认的防范措施

- ARP主动确认的主要目的就是在网关设备对ARP缓存表进行更新时主动确认，防止ARP缓存表更新产生错误。这种主动确认的工作过程首先是对新建ARP缓存表项目时的主动确认，其次是更新已有ARP 缓存表项目时的主动确认。

3.3.4 防范ARP欺骗

□ 源主机IP或MAC的抑制防范措施

- 针对源主机IP或MAC发送的攻击报文进行抑制，是一种配置简单且适用的功能，常采取的抑制措施有如下3种：
 - 进行ARP报文的源主机MAC的一致性检测措施
 - 对IP报文的抑制措施
 - 限制ARP缓存表项目最大数目的抑制措施

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。

谢谢各位!