第九章

网络安全数据集简介及采集

目录

数据集简介

1. DARPA入侵检测评估数据集
2. KDD Cup 99与NSL-KDD数据集
3. Honeynet数据集
4. Challenge 2013数据集
5. Adult数据集
6. 恶意软件数据集

DARPA入侵检测评估数据集

• 迄今为止网络入侵检测领域的标准数据集

- DARPA 98：收集了9周时间的 TCPDUMP网络连接和系统审计数据，仿真各种用户类型、各种不同的网络流量和攻击手段。
- DARPA 99：包括覆盖了Probe、DoS、R2L、U2R和Data等5大类58种典型攻击方式，是目前最为全面的攻击测试数据集。
- DARPA 2000：一种深度测试，集中地测试入侵检测系统对于某一种攻击的检测效果，对检测算法和检测机制可以进行深入的分析。

DARPA入侵检测评估数据集

- 迄今为止网络入侵检测领域的标准数据集

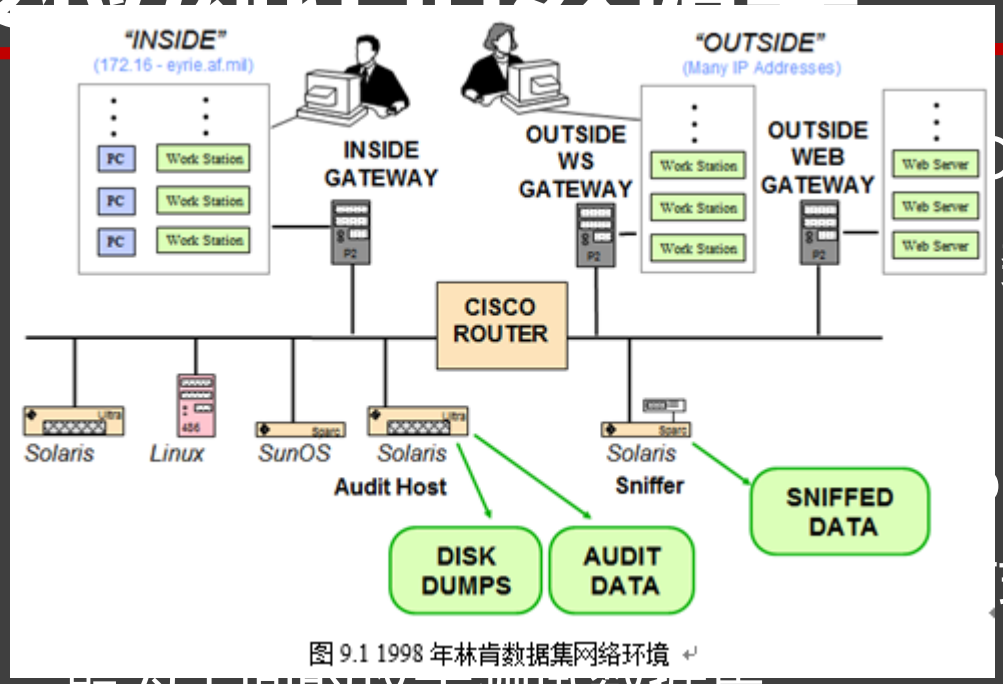


图 9.1 1998 年林肯数据集网络环境

UMP网络
类型、各种
S、R2L、
式，是目前
最为全面的攻击测试数据集。

- DARPA 2000：一种深度测试，集中地测试入侵检测系统对于某一种攻击的检测效果，对检测算法和检测机制可以进行深入的分析。

DARPA入侵检测评估数据集

- 迄今为止网络入侵检测领域的标准数据集

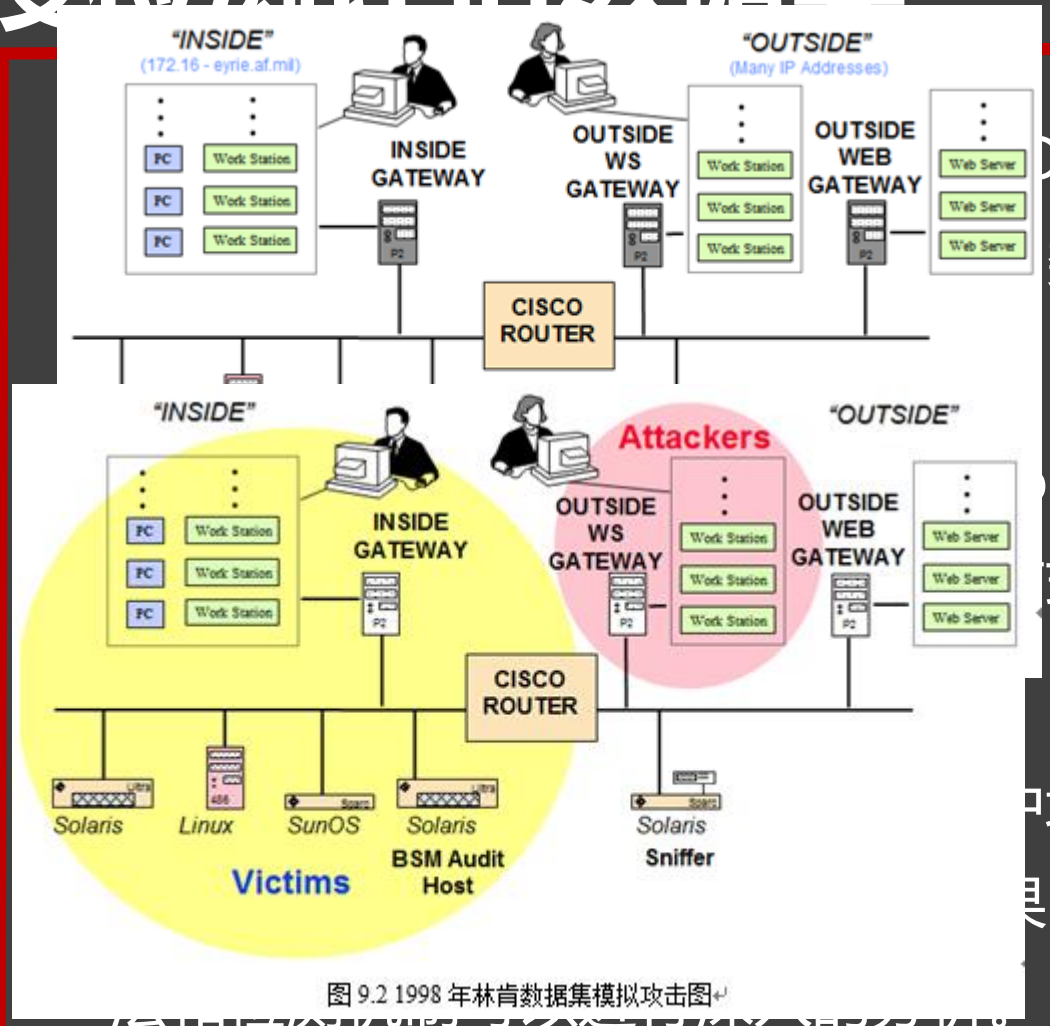


图 9.2 1998 年林肯数据集模拟攻击图

UMP网络
类型、各种
S、R2L、
式，是目前
中地测试入侵
果，对检测算

DARPA入侵检测评估数据集

- 迄今为止
入侵检测领
标准数据集

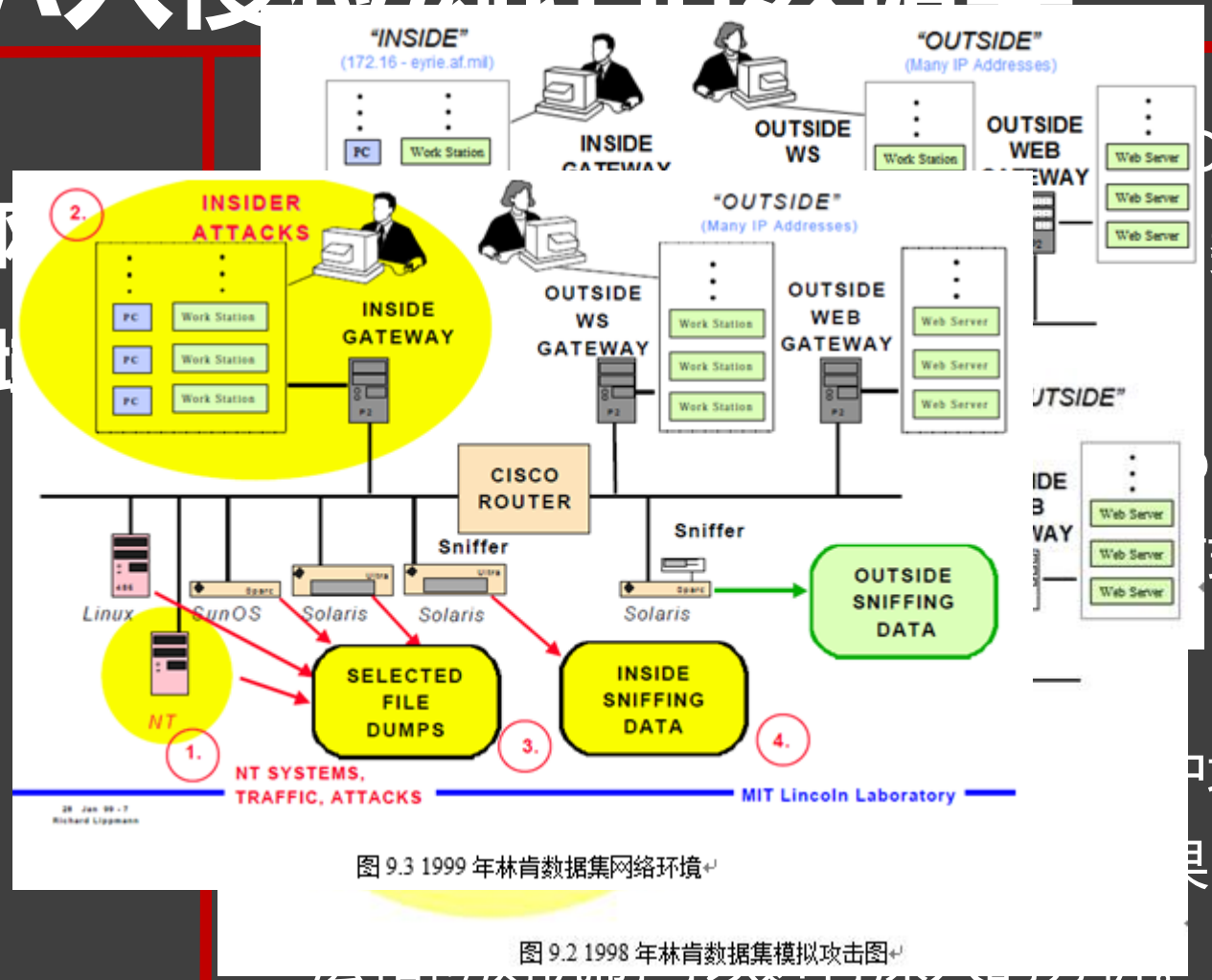


图 9.3 1999 年林肯数据集网络环境

图 9.2 1998 年林肯数据集模拟攻击图

UMP网络
类型、各种
S、R2L、
式，是目前
地测试入侵
果，对检测算

KDD Cup 99与NSL-KDD数据集

- **KDD Cup 99数据集**
- **NSL-KDD数据集**

- KDD Cup 99数据集
  - 数据来源
    - KDD Cup 99数据集是采用数据挖掘等技术对DARPA 98和DARPA 99数据集进行特征分析和数据预处理，形成的新数据集。
  - 数据范例
    - 数据集中每个连接（*）用41个特征来描述，例如：2, tcp, smtp, SF, 1684, 363, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0.00, 0.00, 0.00, 0.00, 1.00, 0.00, 0.00, 104, 66, 0.63, 0.03, 0.01, 0.00, 0.00, 0.00, 0.00, 0.00, normal.

KDD Cup 99数据集

- **KDD Cup 99数据集特征分类**

TCP连接基本特征（共9种）	TCP连接的内容特征（共13种）	基于时间的网络流量统计特征（共9种，23～31）	基于主机的网络流量统计特征（共10种，32～41）
• 基本连接特征包含了一些连接的基本属性，如连续时间，协议类型，传送的字节数等信息泄露；	• 为了检测U2R和R2L之类的嵌入在数据包数据负载里面的攻击，从数据内容里面抽取了部分可能反映入侵行为的内容特征，如登录失败的次数等。	• 统计当前连接记录与之前一段时间内的连接记录之间存在的某些联系。分为两种集合：“same host”特征和“same service”特征	• 按照目标主机进行分类，使用一个具有100个连接的时间窗，统计当前连接之前100个连接记录中与当前连接具有相同目标主机的统计信息。

NSL-KDD数据集

- **对KDD CUP 99的改进**

- 除去了KDD CUP 99数据集中冗余的数据，克服了分类器偏向于重复出现的记录，学习方法的性能受影响等问题。
- 对正常和异常的数据比例进行了合适选择，测试和训练数据数量更合理，因此更适合在不同的机器学习技术之间进行有效准确的评估。

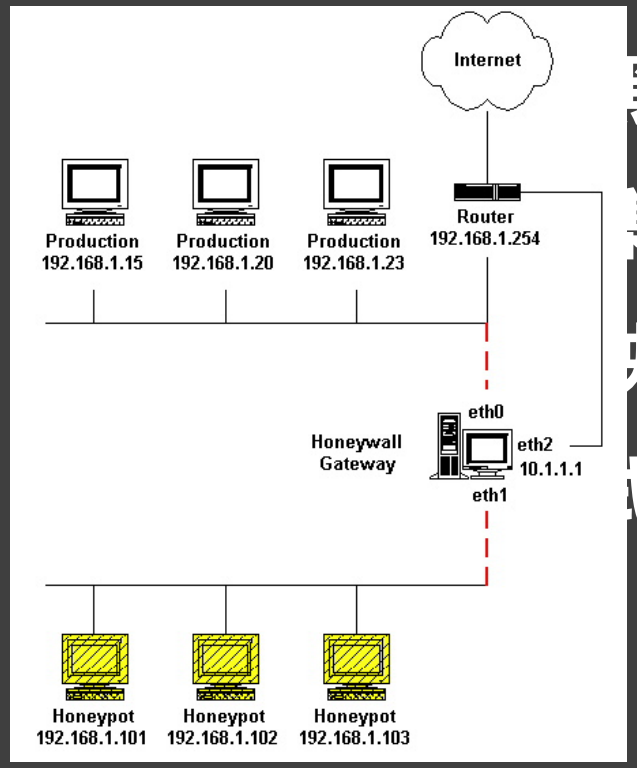
Honeynet数据集

- **由HoneyNet组织收集的黑客攻击数据集，能较好地反映黑客攻击模式**

- 数据来源
  - 包括从2000年4月到2011年2月，累计11个月的Snort报警数据，每月大概60-3000多条Snort报警记录，其网络由8个IP地址通过ISDN连接到ISP。
- 数据范例
  - Apr 16 07:17:06 lisa snort[7483]: IDS128/web-cgi-phf: 200.190.8.220:55220 -> 172.16.1.107:80，其内容分别是日期、时间、触发的Snort规则号、报警内容、源IP、源端口、目的IP和目的端口。
- 网络环境图

Honeynet数据集

- 由HoneyNet



- 数据来源
  - 包括从2000年4月到2011年2月，累计11个月的Snort报警数据，每月大概60-3000多条Snort报警记录，其网络由8个IP地址通过ISDN连接到ISP。
- 数据范例
  - Apr 16 07:17:06 lisa snort[7483]: IDS128/web-cgi-phf: 200.190.8.220:55220 -> 172.16.1.107:80，其内容分别是日期、时间、触发的Snort规则号、报警内容、源IP、源端口、目的IP和目的端口。
- 网络环境图

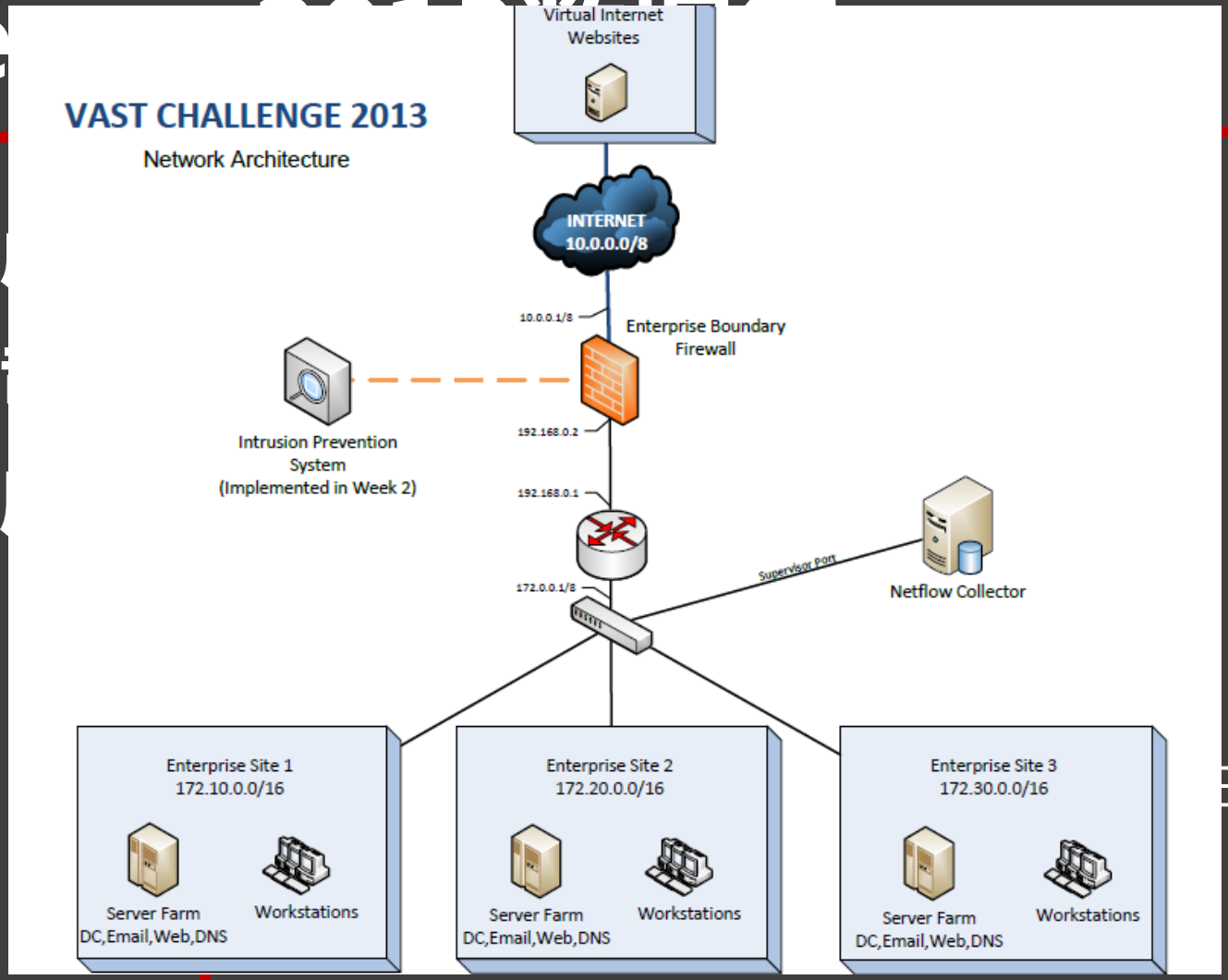
Challenge 2013数据集

- **提供了某虚构的跨国公司内部网络两周的运行日志**

- 日志类型（3种）
  - 网络流量Netflow日志数据
  - Big Brother 网络健康数据
  - Big Brother 网络状态数据
- 日志内容
  - 第一、二周的Netflow和Big Brother日志
  - 第二周的入侵预防系统日志数据
- 网络环境图

Challe

- 提供了某
  的跨国公
  部网络两
  运行日志



志

- 网络环境图

Challenge 2013数据集

- **导入MySQL数据库过程**

- 数据格式
  - 与原始数据集相比，经过了数据清洗和时间同步，加入了统一的时间戳，数据都已经通过入库程序导入了MySQL数据库，所以这儿提供的数据集是通过SQL语句从MySQL数据库中导出的，并且同时提供数据表结构。（请注意使用MyISAM的表格式，预计需要数据库磁盘空间30G，文件名后缀中a表示第一周，b表示第二周）
- 导入步骤
  - 先创建对应表，然后导入数据文件，导入数据文件的MySQL语句参考：Load data infile 'c:/netflow-origin.txt' into table netflow1。

Adult数据集

- **该数据集适用于机器学习、数据挖掘和隐私保护等**

- 数据来源
  - 该数据集来自UCI，又名人口调查数据集，来自于美国1994年人口调查数据库，共有记录48842条，格式为TEXT，包含14个属性，分别为Age，workclass，fnlwgt，education，education-num，marital-status，occupation，relationship，race，sex，capital-gain，capital-loss，hours-per-week，native-country。
- 数据范例
  - 39,State-gov,77516, Bachelors, 13, Never-married, Adm-clerical, Not-in-family, White, Male, 2174, 0, 40, United-States, <=50K

恶意软件数据集

- **该数据集由West Virginia University的Yanfang Ye提供**

- 用于恶意软件检测用于恶意软件检测
  - 包含50000个实例，一半是恶意软件中提取的特征，一半是良性文件中提取的特征。
    - 通过该数据集，可以在数据挖掘和大数据建模技术的基础上，通过Win API调用提取特征集进行恶意软件检测，而我们人看不见的，听不见的，感觉不到的事物或者关系同样是数据，而且很多关键的数据正是隐藏在某些关系之中。
- 用于基于文件说明的恶意软件聚类
  - 包含69,165个文件样本，其中3095个是恶意软件，22,583个是良性文件，其余45,487个是未知文件。

网络数据包采集与回放

1. TCPDUMP抓包
2. Wireshark抓包
3. 网络数据包回放
4. 网络抓包编程

TCPDUMP抓包

- **TCPDUMP是一个用于截取网络分组，并输出分组内容的工具，简单说就是数据包抓包工具。**

- 概述
  - 顾名思义，TCPDUMP可以将网络中传送的数据包的“头”完全截获下来提供分析。它支持针对网络层、协议、主机、网络或端口的过滤，并提供and、or、not等逻辑语句来帮助你去掉无用的信息。
  - 不带参数的TCPDUMP会收集网络中所有的信息包头，数据量巨大，必须过滤。

TCPDUMP抓包

- **选项介绍**

参数	简单介绍
-A	以ASCII格式打印出所有分组，并将链路层的头最小化
-c	在收到指定的数量的分组后，TCPDUMP就会停止
-C	在将一个原始分组写入文件之前，检查文件当前的大小是否超过了参数file_size 中指定的大小。如果超过了指定大小，则关闭当前文件，然后在打开一个新的文件。参数 file_size 的单位是兆字节（是1,000,000字节，而不是1,048,576字节）
-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
-D	打印出系统中所有可以用TCPDUMP截包的网络接口
-e	在输出行打印出数据链路层的头部信息

TCPDUMP抓包

- **选项介绍**

参数	简单介绍
--E	用spi@ipaddr algo:secret解密那些以addr作为地址，并且包含了安全参数索引值spi的IPsec ESP分组
-f	将外部的Internet地址以数字的形式打印出来
-F	从指定的文件中读取表达式，忽略命令行中给出的表达式
-i	指定监听的网络接口
-l	使标准输出变为缓冲行形式，可以把数据导出到文件
-L	列出网络接口的已知数据链路
-m	从文件module中导入SMI MIB模块定义。该参数可以被使用多次，以导入多个MIB模块
-M	如果tcp报文中存在TCP-MD5选项，则需要用secret作为共享的验证码用于验证TCP-MD5选选项摘要（详情可参考RFC 2385）

TCPDUMP抓包

- 选项介绍

参数	简单介绍
-b	在数据-链路层上选择协议，包括ip、arp、rarp、ipx都是这一层的
-n	不把网络地址转换成名字
-nn	不进行端口名称的转换
-N	不输出主机名中的域名部分。例如，‘nic.ddn.mil’只输出‘nic’
-t	在输出的每一行不打印时间戳
-O	不运行分组分组匹配（packet-matching）代码优化程序
-P	不将网络接口设置成混杂模式
-q	快速输出。只输出较少的协议信息
-r	从指定的文件中读取包(这些包一般通过-w选项产生)
-S	将tcp的序列号以绝对值形式输出，而不是相对值

TCPDUMP抓包

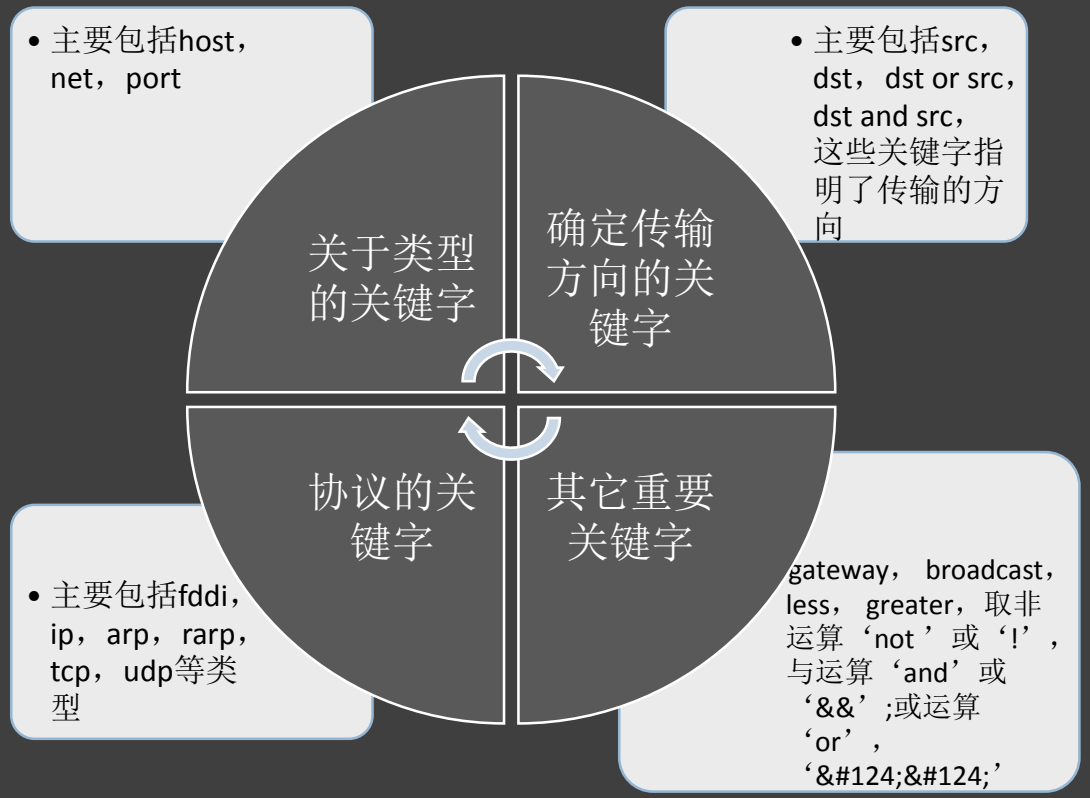
- 选项介绍

参数	简单介绍
-s	从每个分组中读取最开始的snaplen个字节，而不是默认的68个字节
-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；）
-t	不在每一行中输出时间戳
-tt	在每一行中输出非格式化的时间戳
-ttt	输出本行和前面一行之间的时间差
-tttt	在每一行中输出由date处理的默认格式的时间戳
-u	输出未解码的NFS句柄
-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
-w	直接将分组写入文件中，而不是不分析并打印出来

TCPDUMP抓包

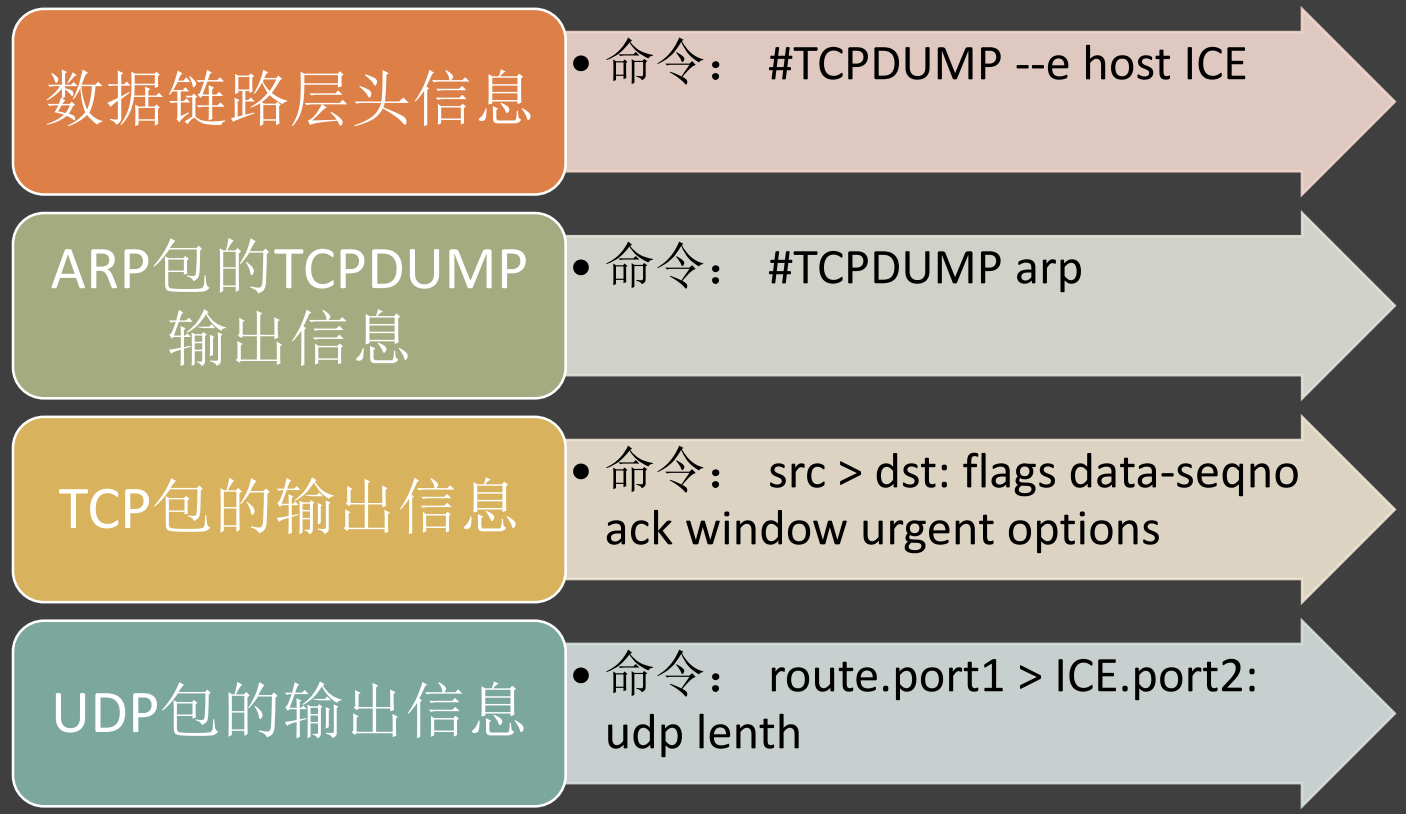
- **表达式介绍**

- 表达式是一个正则表达式，其中包含的关键字类型如图



TCPDUMP抓包

- 输出结果介绍



TCPDUMP抓包

- **举例**

- 想要截获所有210.27.48.1 的主机收到的和发出的所有的分组
  - #TCPDUMP host 210.27.48.1
- 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（注意：括号前的反斜杠是必须的）
  - #TCPDUMP host 210.27.48.1 and (210.27.48.2 or 210.27.48.3)
- 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
- 如果想要获取主机192.168.228.246接收或发出的ssh包，并且不转换主机名使用如下命令
  - #TCPDUMP -nn -n src host 192.168.228.246 and port 22 and tcp

TCPDUMP抓包

- **举例**

- 获取主机192.168.228.246接收或发出的ssh包，并把mac地址也一同显示
  - # TCPDUMP -e src host 192.168.228.246 and port 22 and tcp -n -nn
- 过滤的是源主机为192.168.0.1与目的网络为192.168.0.0的报头
  - TCPDUMP src host 192.168.0.1 and dst net 192.168.0.0/24
- 过滤源主机物理地址为XXX的报头
  - TCPDUMP ether src 00:50:04:BA:9B and dst……
    （为什么ether src后面没有host或者net？物理地址当然不可能有网络）。
- 过滤源主机192.168.0.1和目的端口不是telnet的报头，并导入到tes.t.txt文件中
  - TCPDUMP src host 192.168.0.1 and dst port not telnet -l > test.txt

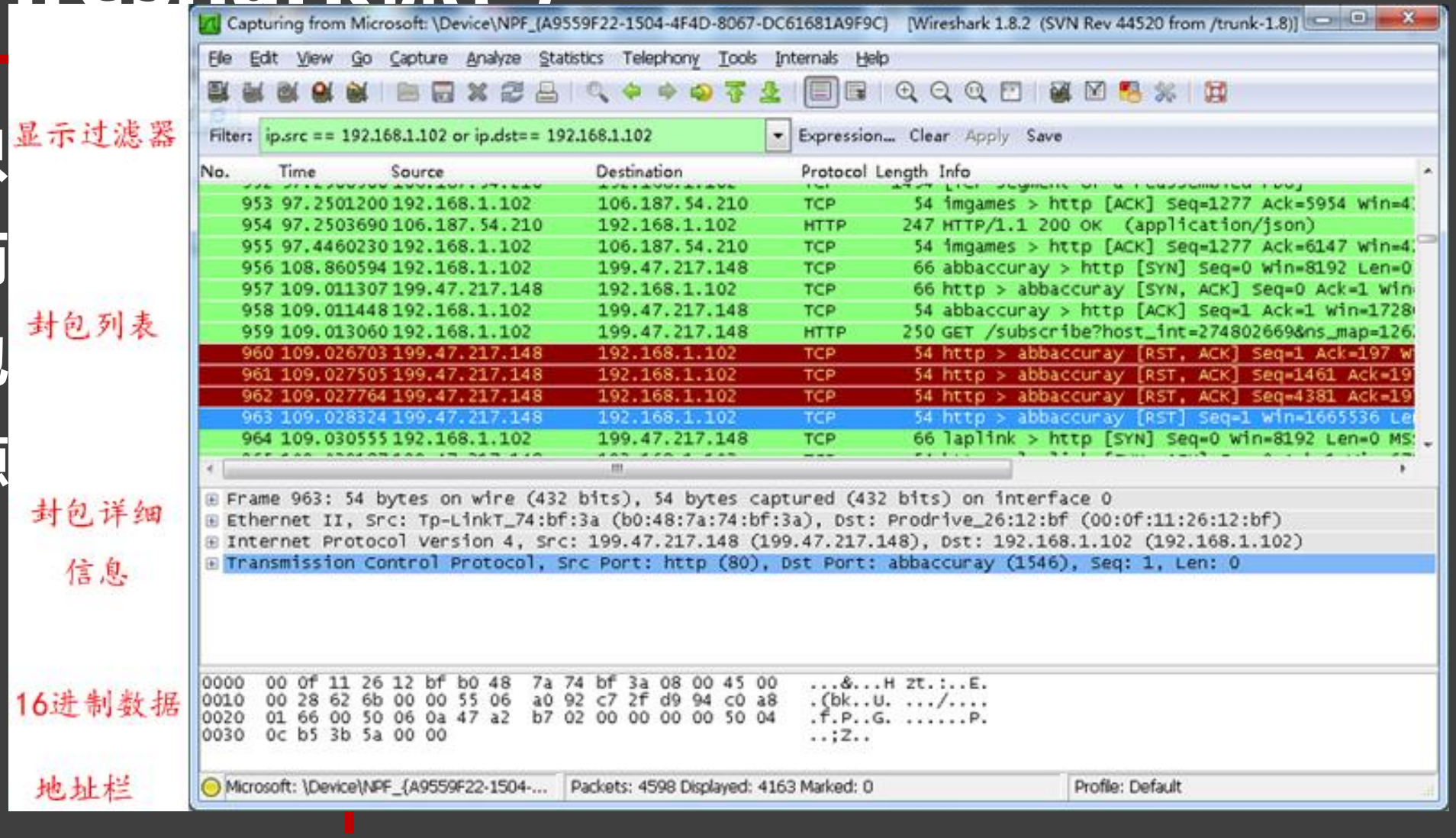
Wireshark抓包

- **只能查看封包，而不能修改封包的内容的开源软件**

- Wireshark窗口（五部分）
  - Display Filter(显示过滤器)，用于过滤
  - Packet List Pane(封包列表)，显示编号，时间戳，源地址，目标地址，协议，长度，以及封包信息。可以看到不同的协议用了不同的颜色显示，也可以修改这些显示颜色的规则
  - Packet Details Pane(封包详细信息)，显示封包中的字段
  - Dissector Pane(16进制数据)
  - Miscellanous(地址栏，杂项)

Wireshark抓包

- 只
- 而
- 包
- 源

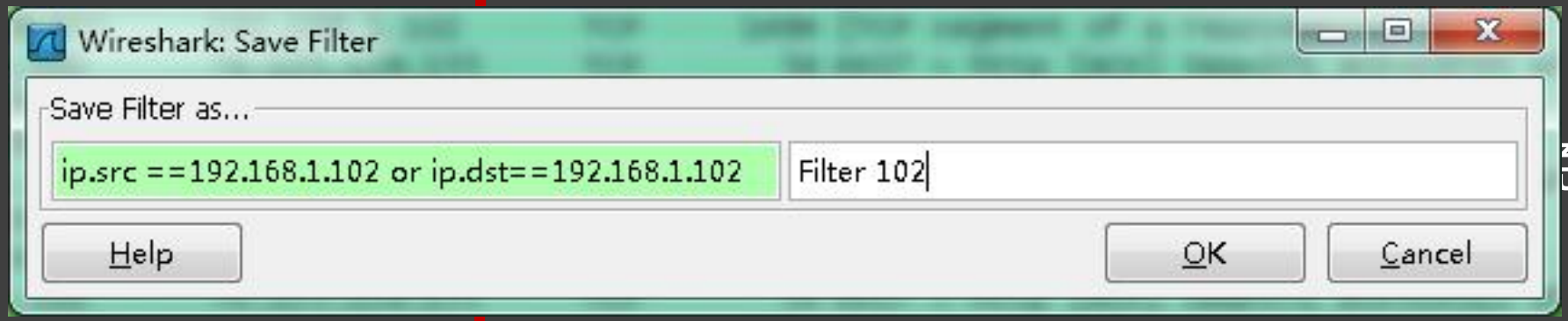


Wireshark抓包

- **过滤器（两种）**
  - 显示过滤器，即主界面显示的，用来在捕获的记录中找到所需要的记录
  - 捕获过滤器，用来过滤捕获的封包，以免捕获太多的记录

Wireshark抓包

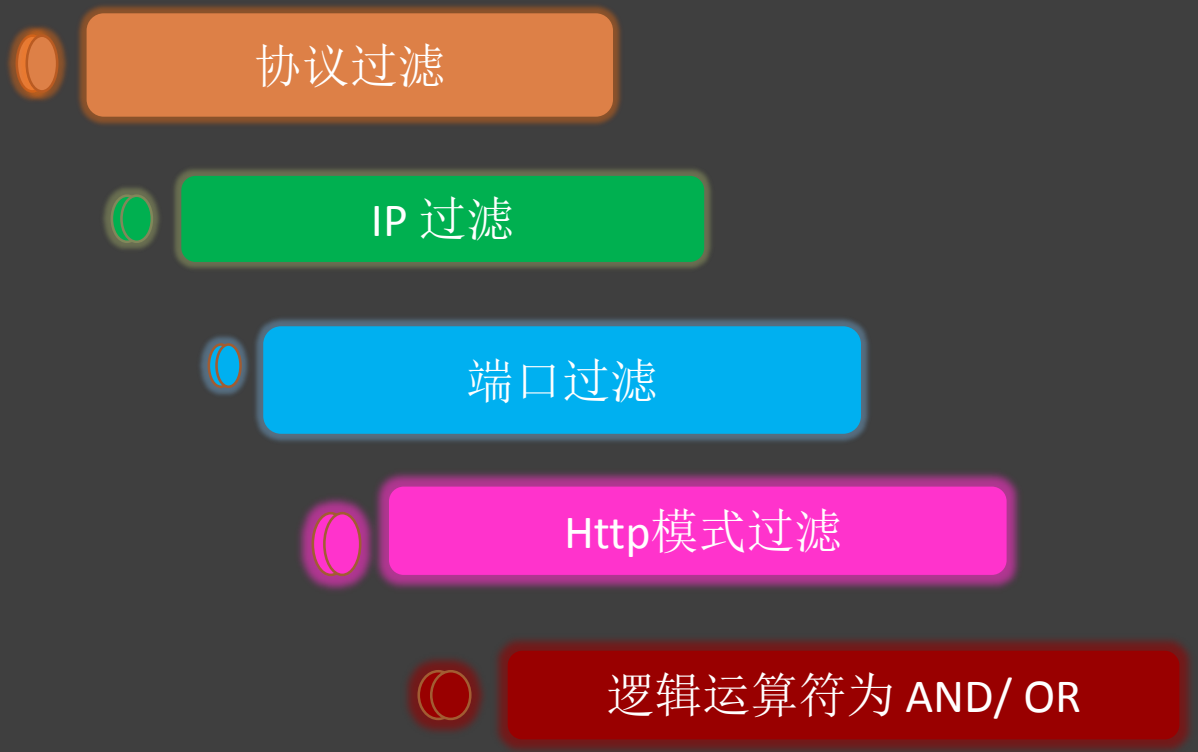
- **过滤器（两种）**
  - 显示过滤器，即主界面显示的，用来在捕获



色捕

Wireshark抓包

- 过滤表达式规则



Wireshark抓包

- **封包信息**

Frame

- 物理层的数据帧概况

Ethernet II

- 数据链路层以太网帧头部信息

Internet Protocol Version 4

- 互联网层IP包头部信息

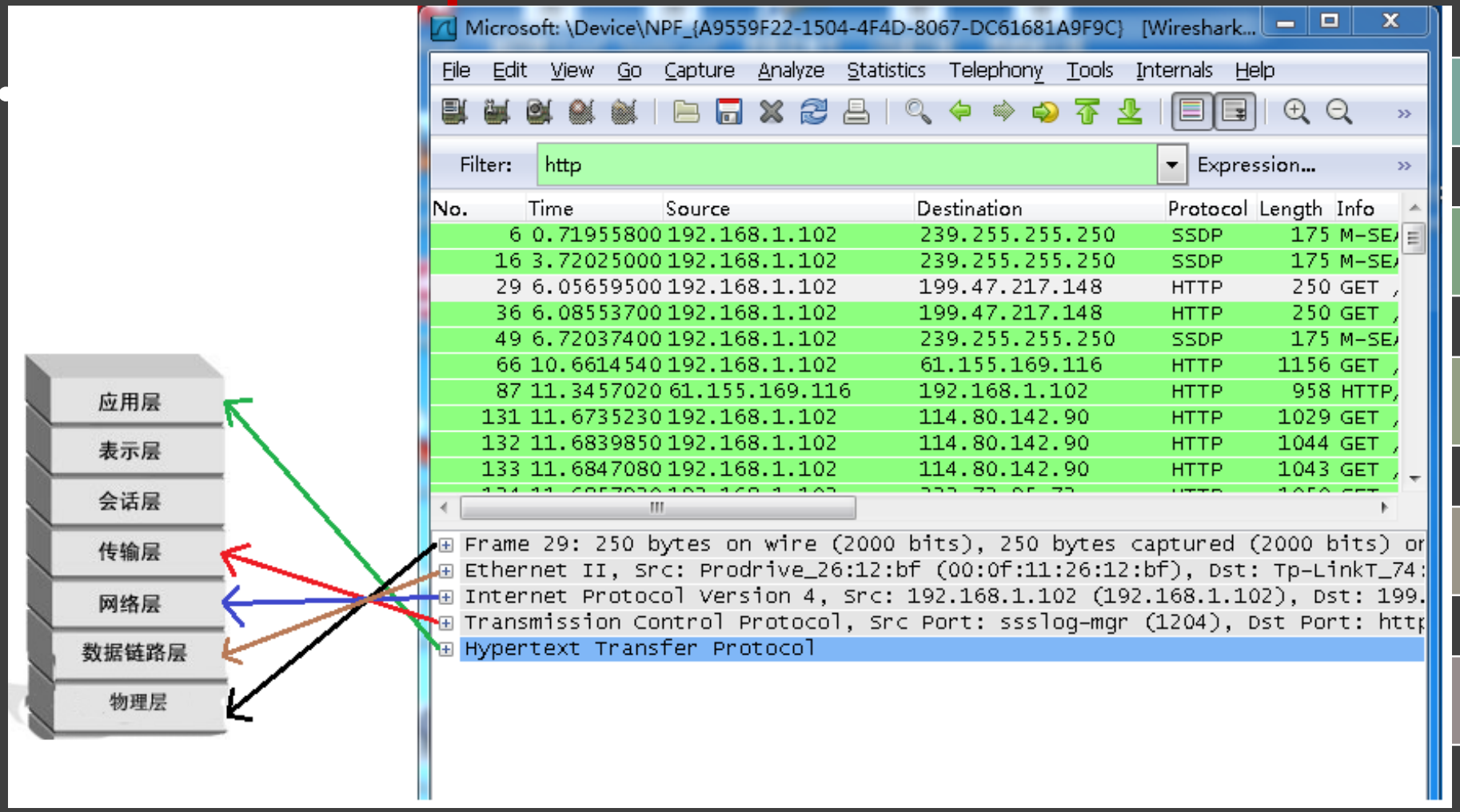
Transmission Control Protocol

- 传输层T的数据段头部信息，此处是TCP

Hypertext Transfer Protocol

- 应用层的信息，此处是HTTP协议

Wireshark抓包

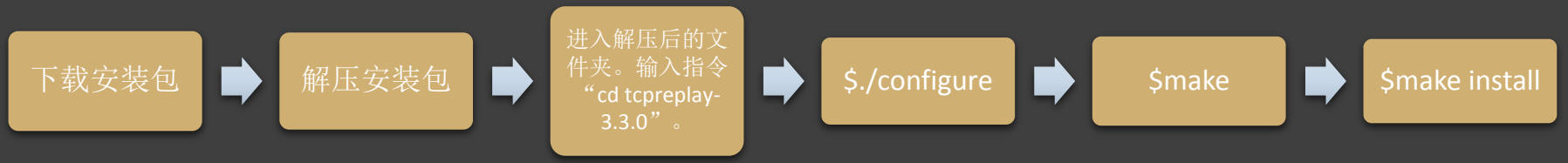


网络数据包回放

- **Tcpreplay包含的工具**

- tcpprep: 这个工具的作用就是划分客户端和服务器，区分pcap数据包的流向，即划分那些包是client的，哪些包是server的，一会发包的时候client包从一个网卡发，另一个server的包可能从另一个网卡发。
- tcprewrite：这个工具的作用就是来修改报文，主要修改2层，3层，4层报文头，即MAC地址，IP地址和PORT地址。
- tcpreplay: 这是最终真正发包使用的工具，可以选择主网卡、从网卡、发包速度等。

Tcpreplay安装过程



网络数据包回放

- **常用指令**

- 采用port-spllit模式来处理http.pcap文件（区分http.pcap中的客户端和服务器），然后将处理结果存到cache_test_cache文件中
  - $tcpprep –-port –cachefile=cache_test.cache --pcap=http.pcap
- 改写数据包的的内容
  - $tcprewrite --endpoints=192.168.0.1:192.168.0.2 --cachefile=cache_test.cache \--infile=http.pcap --outfile=http_rewrite.pcap
- 发送收集的数据包
  - $tcpreplay –intf1=eth0 –intf2=eth0 –t –cachefile=cache_test.cache http_rewrite_pcap

网络抓包编程

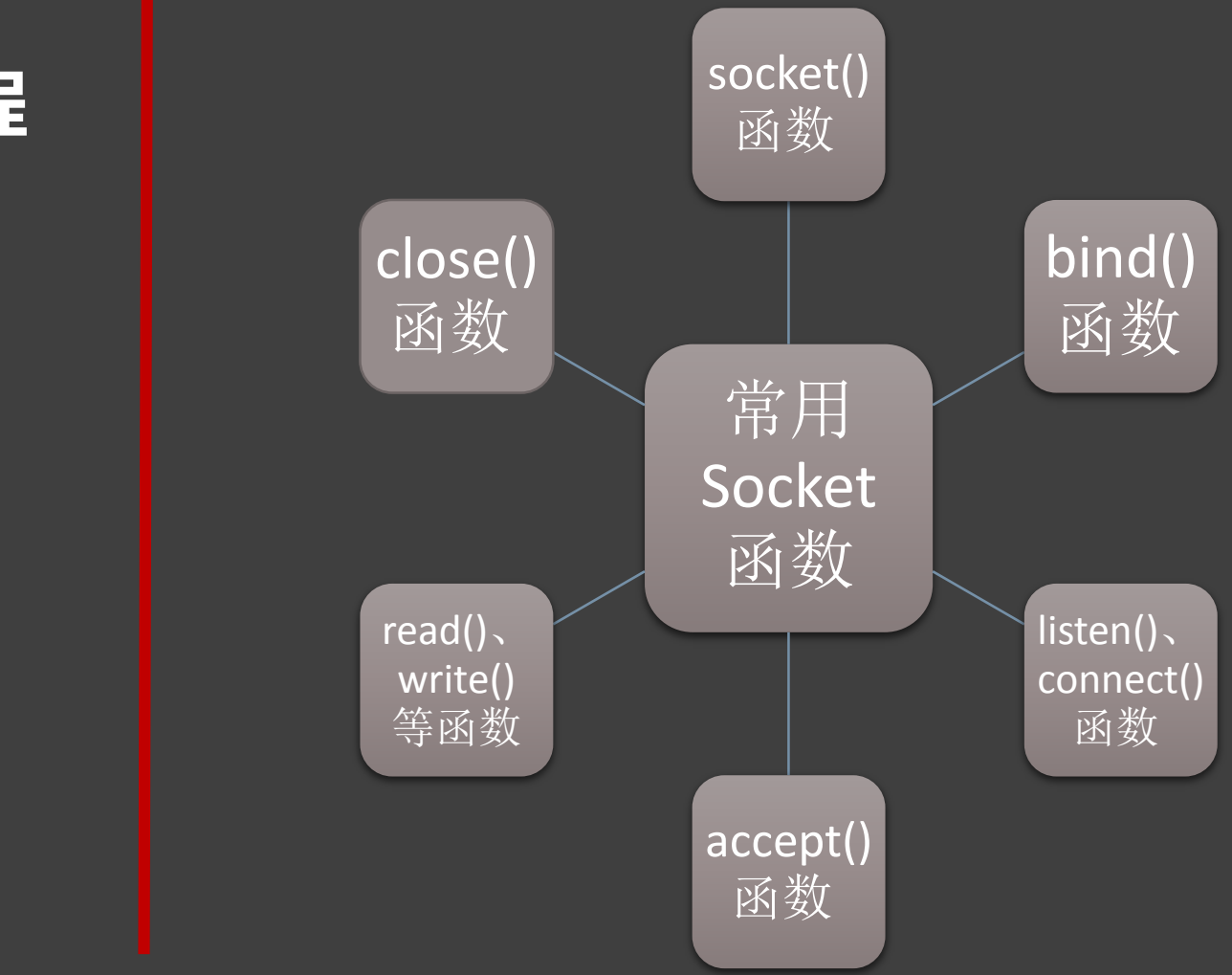
- **LIBPCAP抓包**

- 查找用于捕获数据包的缺省设备
  - char *pcap_lookupdev(char *errbuf);
- 打开用于捕获数据包的网络设备
  - pcap_t *pcap_open_live(const char *device, int snaplen, int promisc, int to_ms, char *errbuf);
- 捕获下一个数据包
  - const u_char *pcap_next(pcap_t *p, struct pcap_pkthdr *h);
- 捕获下一个数据包
  - typedef void (*pcap_handler)(u_char *user, const struct pcap_pkthdr *h, const u_char *bytes);
- const u_char *pcap_loop(pcap_t *p, int cnt, pcap_handler callback, u_char *user);
- 创建过滤器
  - int pcap_compile(pcap_t *p, struct bpf_program *fp, char *str, int optimize, bpf_u_int32 netmask);
- 安装过滤器
  - int pcap_setfilter(pcap_t *p, struct bpf_program *fp);

网络抓包编程

- Socket编程

  抓包



网络抓包编程

- **Socket()函数**

- int socket(int protofamily, int type, int protocol);//返回sockfd
  - protofamily：即协议域，又称为协议族（family）
  - type：指定socket类型
  - protocol：故名思意，就是指定协议

网络抓包编程

- **Socket()函数**

- 注意：并不是上面的type和protocol可以随意组合的，如SOCK_STREAM不可以跟IPPROTO_UDP组合。当protocol为0时，会自动选择type类型对应的默认协议。
- 当我们调用socket创建一个socket时，返回的socket描述字它存在于协议族（address family，AF_XXX）空间中，但没有一个具体的地址。如果想要给它赋值一个地址，就必须调用bind()函数，否则就当调用connect()、listen()时系统会自动随机分配一个端口。

网络抓包编程

- **Bind()函数**

- int bind(int sockfd, const struct sockaddr *addr, socklen_t addrlen);
  - sockfd：即socket描述字，它是通过socket()函数创建了，唯一标识一个socket。
  - addr：一个const struct sockaddr *指针，指向要绑定给sockfd的协议地址。
  - addrlen：对应的是地址的长度。

网络抓包编程

- **Bind()函数**

- 字节序，顾名思义字节的顺序，就是大于一个字节类型的数据在内存中的存放顺序，一个字节的数据没有顺序的问题了。
- 主机字节序就是我们平常说的大端和小端模式：不同的CPU有不同的字节序类型，这些字节序是指整数在内存中保存的顺序，这个叫做主机序。
- 网络字节序：4个字节的32 bit值以下面的次序传输：首先是0～7bit，其次8～15bit，然后16～23bit，最后是24~31bit。这种传输次序称作大端字节序。由于TCP/IP首部中所有的二进制整数在网络中传输时都要求以这种次序，因此它又称作网络字节序。

网络抓包编程

- **listen()、connect()函数**

- 如果作为一个服务器，在调用socket()、bind()之后就会调用listen()来监听这个socket，如果客户端这时调用connect()发出连接请求，服务器端就会接收到这个请求。

网络抓包编程

- **listen()、connect()函数**

- int listen(int sockfd, int backlog);
  - sockfd：要监听的socket描述字
  - backlog：相应socket可以排队的最大连接个数
- int connect(int sockfd, const struct sockaddr *addr, socklen_t addrlen);
  - sockfd：客户端的socket描述字
  - addr：服务器的socket地址
  - addrlen ：socket地址的长度

网络抓包编程

- **accept()函数**

- int accept(int sockfd, struct sockaddr *addr, socklen_t *addrlen); //返回连接connect_fd
- sockfd:上面解释中的监听套接字
- addr：结果参数，用来接受一个返回值，这返回值指定客户端的地址
- len：结果的参数，用来接受上述addr的结构的大小

网络抓包编程

- **accept()函数**

- 注意：
- accept默认会阻塞进程，直到有一个客户连接建立后返回，它返回的是一个新可用的套接字，这个套接字是连接套接字。
- 此时我们需要区分两种套接字：
  - 监听套接字: 监听套接字正如accept的参数sockfd，它是监听套接字，在调用listen函数之后，是服务器开始调用socket()函数生成的，称为监听socket描述字(监听套接字)
  - 连接套接字：一个套接字会从主动连接的套接字变身为一个监听套接字；而accept函数返回的是已连接socket描述字(一个连接套接字)，它代表着一个网络已经存在的点点连接。

网络抓包编程

- **read()、write()等函数**

- read函数是负责从fd中读取内容.当读成功时，read返回实际所读的字节数，如果返回的值是0表示已经读到文件的结束了，小于0表示出现了错误。
- write函数将buf中的nbytes字节内容写入文件描述符fd.成功时返回写的字节数。失败时返回-1，并设置errno变量。

网络抓包编程

- **close()函数**

- int close(int fd);
- close一个TCP socket的缺省行为时把该socket标记为以关闭，然后立即返回到调用进程。该描述字不能再由调用进程使用，也就是说不能再作为read或write的第一个参数。
- 注意：close操作只是使相应socket描述字的引用计数-1，只有当引用计数为0的时候，才会触发TCP客户端向服务器发送终止连接请求。

Thanks!