Networks Security

Information Security Fundamentals

Security in Networks

Network Concepts

a) Introduction
b) Protocols

Threats in Networks

a) Introduction
b) Threat precursors
c) Threats in transit: eavesdropping and wiretapping
d) Protocol flaws

Security in Networks – Part 1 – Outline (2)

Types of attacks
- g-1) Impersonation
- g-2) Spoofing
- g-3) Message confidentiality threats
- g-4) Message integrity threats
- g-5) Web site attacks
- g-6) Denial of service
- g-7) Distributed denial of service
- g-8) Threats to active or mobile code
- g-9) Scripted and complex attacks

Summary of network vulnerabilities

Network Concepts

- Outline
  a) Introduction
  b) The network
  c) Media
  d) Protocols
  e) Types of networks
  f) Topologies
  g) Distributed systems
  h) APIs
  i) Advantages of computing networks

a. Introduction

- We'll review network basics only
  - Emphasis on security
  - Simplifying network complexity (by abstractions)

- Concept of fault tolerance
  - System reliability higher than reliability of its components
    - One way: redundancy

      => elimination of single points of failure

      E.g. a spare in your car
      - E.g., resilient routing in networks

        \- with redundant source-to-destination paths

b. The network (1)

- Simplest network

  workstation <-----------------------------------> host

  (client) communication medium (server)

- More typical networks:

  many clients connected to many servers

- Basic terms:
  - *Node* – can include a number of hosts (computers)
  - *Host*
  - *Link* – connects hosts

Protocols

- *Media independence* – we don't care what media used for communications

- *Protocols* provide abstract view of communications
  - View in terms of users and data
  - The 'how' details are hiden

- *Protocol stack* – layered protocol architecture
  - Each higher layer uses abstract view (what) provided by lower layer (which hides the 'how' details)
  - Each lower layer encapsulates higher layer (in an 'envelope' consisting of header and/or trailer)

- Two popular protocol stacks:
  1) Open Systems Interconnection (OSI)
  2) Transmission Control Protocol / Internet Protocol (TCP/IP)

Protocols (2)

1) ISO OSI Reference Model (ISO = Int'l Standards Organization)

OSI Layer	Name	Activity
7	Application	User-level messages
6	Presentation	Standardized data appearance, blocking, text compression
5	Session	Sessions/logical connections among parts of an app; msg sequencing, recovery
4	Transport	Flow control, end-to-end error detection & correction, priority service
3	Network	Routing, msg → same-sized packets
2	Data Link	Reliable data delivery over physical medium; transmission error recovery, packets → same-sized frames
1	Physical	Actual communication across physical medium; transmits bits

Protocols (3)

- Each layer adds its own service to communication
- Fig. 7-5, p.374
  - OSI stack at sender and at receiver
  - Corresponding layers are peers
- Example: Sending e-mail (p.373 - 376)

  On the sender's end:
  - User writes message
  - Layer 7 (application): Application pgm (e.g., MS Outlokk or Eudora) produces standard e-mail format: [header, body]
  - Layer 6 (presentation): Text compression, char conversion, cryptography
  - Layer 5 (session): No actions (email is 1-way - needs no 2-way session)

Protocols (4)

- Layer 4 (transport): Adds error detection & correction codes
- Layer 3 (network): Adds source address and destination address to msg header (cf. Fig.7-7, p.375) & produces *packets*
  - Packet addresses are in format recognizable by network *routers*
    - Now packets ready to be moved from your computer to your router
    - Then, your router can move packets to your destination’s router (possibly via a chain of routers)
    - Then, your destination’s router can move packets to your destination’s computer

Protocols (5)

- Layer 2 (data): Adds your computer's MAC address (source MAC) and your router's MAC address (destination MAC) (cf. Fig.7-8, p.376) & produces *frames*
  - *MAC address* = Media Access Control address – a *unique physical* address in your local network
  - MAC address identifies a *network interface card* (*NIC*) of the computer/router
- Layer 1 (physical): Device drivers send sequences of bits over physical medium

On the receiver's end:

- Layer 1 (physical): Device drivers receive sequence of bits over physical medium
- Layer 2 (data): NIC card of receiver's computer receives frames addressed to it; removes MAC addresses, reconstructs packets

Protocols (6)

- Layer 3 (network): Checks if packet addressed to it; removes source/dest. Addresses; reorders packets if arrived out-of-order
- Layer 4 (transport): Applies error detection/correction
- Layer 5 (session): No actions (email is 1-way - needs no 2-way session)
- Layer 6 (presentation): Decryption, char conversion, decompression
- Layer 7 (application): Application pgm (e.g., MS Outlokk or Eudora) converts standard e-mail format: [header, body] into user-friendly output

Protocols (7)

- *OSI* is a conceptual model — *not actual implementation*
  - Shows all activities required for communication
  - Would be to slow and inefficient with 7 layers

- An example implementation: TCP/IP

Protocols (8)

2) Transmission Control Protocol/Internet Protocol (TCP/IP)

- Invented for what eventually became Internet
- Defined in terms of protocols not layers

  but can be represented in terms of *four* layers:
  - Application layer
  - Host-to-host (e2e =end-to-end) transport layer
  - Internet layer
  - Physical layer
- Some people use different layer names (e.g. Application, Network, Data Link, and Physical - cf. Wikipedia at: http://en.wikipedia.org/wiki/Internet_protocol_suite)
  - Confusing since Network here corresponds to Transport in OSI, and Data Link here corresponds to Network in OSI)
- Some people use yet different layer names (e.g. Application, Transport, Internet, Network Access - cf. Wikipedia at: http://en.wikipedia.org/wiki/Internet_protocol_suite)

- Actually not TCP/IP but:

  TCP/IP/UDP (user datagram protocol)

Protocols (9)

[cf. B. Endicott-Popovsky and D. Frincke]

- TCP/IP vs. OSI

OSI Layer	Name	Activity
7	*Application*	User-level data
6	Presentation	Standardized data appearance
5	Session	Logical connection among parts
4	*Transport*	Flow control
3	*Internet* ("Network" in OSI)	Routing
2	Data Link	Reliable data delivery
1	*Physical*	Actual communication across physical medium

Protocols (10)

- TCP/IP

Layer	Action	Responsibilities
Application	Prepare messages from user interaction	User interaction, addressing
Transport	Convert messages to packets	Sequencing of packets, reliability (integrity), error correction
Internet	Convert packets to datagrams	Flow control, routing
Physical	Transmit datagrams as individual bits	Actual data communication

Protocols (11)

- TCP packet includes:
  - Sequence #
  - Acknowledgement # connecting packets of a session
  - Flags
  - Source port #
  - Destination port #

- *Port* – # of a *channel* for communication for a particular (type of) application running on a computer
  - Examples of port-application pairs:
    - 23 – Telnet (remote terminal connection)
    - 25 – SMTP (e-mail)
    - 80 – HTTP (web pages)
    - 161 – SNMP (network mngmt)
  - App has a waiting process monitoring its port
    - When port receives data, app performs service on it

Protocols (12)

- UDP - user datagram protocol (connection*less*)
  - Faster and smaller than TCP
    - No error checking/correction
    - 8 bytes of control info (vs. 24 bytes for TCP)
  - Uses IP => actually UDP/IP
  - Applications use application-level protocols
    - which, in turn, use TCP/IP or UDP/IP

    Apps do *not* use TCP/IP or UDP/IP *directly*
    - Examples - cf. Table 7-3, p.379 (shows 4 protocol layers)

      Examples of App Protocols using TCP/IP:
      - SMTP (e-mail) / HTTP (web pages) / FTP (file transfer) / Telnet (remote terminal connection)

      Examples of App Protocols using UDP/IP:
      - SNMP (network mngmt) / Syslog (entering log records) / Time (synchronizing network device time)

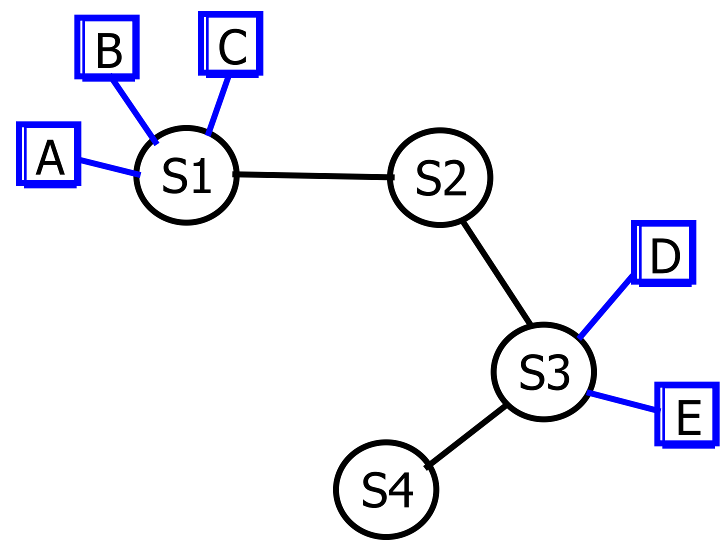
Protocols (13)

- Network *addressing scheme*
  - Address – unique identifier for a single point in the network
  - WAN addressing must be more standardized than LAN addressing
  - LAN addressing:
    - Each node has unique address
      - E.g. = address of its NIC (network interface card)
    - Network admin may choose arbitrary addresses
  - WAN addressing:
    - Most common: Internet addr. scheme – IP addresses
      - 32 bits: four 8-bit groups
      - In decimal: g1.g2.g3.g4 where gi ∈ [0, 255]
        E.g.: 141.218.143.10
      - User-friendly representation
        E.g.: cs.wmich.edu (for 141.218.143.10)

Protocols (14)

- Parsing IP addresses
  - From right to left
  - Rightmost part, known as *top-level domain*
    - E.g., .com, .edu, .net, .org,. gov,
    - E.g., .us, .in, .pl
    - Top-level domain controlled by *Internet Registrars*
      - IRs also control 2nd-level domains (e.g., wmich in wmich.edu)
      - IRs maintain tables of 2nd-level domains within „their" top-level domains
- Finding a service on Internet – e.g., cs.wmich.edu
  - Host looking for a service queries one of tables at IRs for wmich.edu
  - Host finds numerical IP address for wmich.edu
  - Using this IP address, host queries wmich.edu to get from *its* table numerical address for cs.wmich.edu

- Dissemination of routing information
  - Each host knows all other hosts directly connected to it
    - Directly-connected => distance = 1 hop
  - Each host passes information about its directly connected hosts to all its neighbors
  - Example – [Fig. below simplifies Fig. 7-2 p.366]
    - System 1 (S1) informs S2 that S1 is 1 hop away from Clients A, B, and C
    - S2 notifies S3 that S2 is 2 hops away from A, B, C
    - S3 notifes S2 that S3 is 1 hop away from D, E & S4
    - S2 notifies S1 that S2 is 2
    - hops away from D, E & S4
    - Etc., etc.



Threats in Networks (1)

- Outline
  a) Introduction
  b) Network vulnerabilities
  c) Who attacks networks?
  d) Threat precursors
  e) Threats in transit: eavesdropping and wiretapping
  f) Protocol flaws
  g) Types of attacks:
     g-1) Impersonation
     g-2) Spoofing
     g-3) Message confidentiality threats
     g-4) Message integrity threats
     g-5) Web site attacks

- Outline—cont.
  - g) Types of attacks-cont.:
    - g-6) Denial of service
    - g-7) Distributed denial of service
    - g-8) Threats to active or mobile code
    - g-9) Scripted and complex attacks
  - h) Summary of network vulnerabilities

a. Introduction (1)

- We will consider

  threats aimed to compromise C-I-A

  applied against data, software, or hardware

  by nature, accidents, nonmalicious humans, or malicious attackers

Introduction (2)

- From CSI/FBI Report 2002 (survey of ~500 com/gov/edu/org)
  - 90% detected computer security breaches
  - 80% acknowledged financial losses
  - 44% (223) were willing/able to quantify losses: $455M
  - Most serious losses: theft of proprietary info and fraud
    - 26 respondents: $170M
    - 25 respondents: $115M
  - 74% cited *Internet connection* as a frequent point of attack
  - 33% cited *internal systems* as a frequent point of attack
  - 34% *reported* intrusions to law enforcement (up from 16%-1996)

[cf.: D. Frincke]

Introduction (3)

- More from CSI/FBI Report 2002
  - 40% detected external penetration
  - 40% detected DoS attacks
  - 78% detected employee abuse of Internet
  - 85% detected computer viruses
  - 38% suffered unauthorized access on Web sites
  - 21% didn’t know
  - 12% reported theft of information
  - 6% *reported* financial fraud (up from 3%-- 2000)

[cf.: D. Frincke]

Threat precursors (1)

- How attackers prepare for attacks?
  - Investigate and plan

  These are *threat prescursors*

- If we detect threat precursors, we might be able to block attacks before they're launched

- Threat prescursors techniques include:

  1) Port scan
  2) Social engineering
  3) Reconnaissance
  4) OS and application fingerprinting
  5) Using bulletin boards and chats
  6) Getting available documentation

Threat precursors (2)

1) Port scan

Port scanner - pgm that scans port indicated by IP address

- Reports about:
  a) Standard ports/services running and responding
     - Recall (ex.): port 80–HTTP, 25-SMTP(e-mail), 23-Telnet

  b) OS installed on target system

  c) Apps and app versions on target system

  => Can infer which known vulnerabilities present
- Example: **`nmap`**
  - **`nmap -sP 192.168.100.*`**
    - Performs quick (20-30 s) ping scan („P")
    - Notice wild card!
  - **`nmap -sT 192.168.100.102`**
    - Performs much slower (~10 min.) TCP port scan („T")
  - OPTIONAL: more on nmap „Computer Security Lab Manual" (p.199)

Threat precursors (3)

1) Port scan – cont.

- Other port scanning tools:
  - **`netcat`** (free)
  - Many commercial port scanners:
    - Nessus (Nessus Corp.)
    - CyberCop Scanner (Network Associates)
    - Secure Scanner (Cisco)
    - Internet Scanner (Internet Security systems)
    - ...

2) Social engineering

= using social skills and personal interaction to get someone to reveal security-releveant info or do sth that permits an attack

- Impersonates sb inside an organization
  - Person in a high position (works best – by intimidation), co-worker, ...
- Often exploits sense of urgency
  - „My laptop has been stolen and I have an important presentation. Can you help me ....”
- Relies on human tendency to help others when asked politely

Threat precursors (5)

2) Social engineering – cont.

- Example: Phone call asking for system info
  - Never provide system info to a caller
  - Ask for identification
  - Best: Refer to help desk or proper system/security authority
  - If contact with sys/sec auth impossible, you might consider calling back but using phone number known to you from *independent source* (*not* the number given by the caller)
    - Independent source: known beforehand, obtained from company directory, etc.

3) Reconnaissance

= collecting discrete bits of security information from various sources and putting them together

- Reconnaissance techniques include:
  a) Dumpster diving
  b) Eavesdropping
     - E.g., follow employees to lunch, listen in
  c) Befriending key personnel (social engg!)

- Reconnaissance requires little training, minimal investment, limited time

  BUT can give big payoff in gaining background info

4) OS and application fingerprinting

= finding out OS/app name, manufacturer and version by using pecularities in OS/app responses

- Example: Attacker's approach
  - Earlier port scan (e.g., nmap) reveals that port 80 – HTTP is running
  - Attacker uses Telnet to send meaningless msg to port 80
  - Attacker uses response (or a lackof it) to infer which of many possible OS/app it is
    - Each version of OS/app has its fingerprint (pecularities) that reveals its identity (manufacturer, name, version)

Threat precursors (8)

5) Using bulletin boards / chats

- Attackers use them to help each other
  - Exchange info on their exploits, tricks, etc.

6) Getting available documentation

- Vendor documentation can help attackers
  - Esp. 3rd party developer documentation

e. Threats in transit: eavesdropping and wiretapping (1)

- Threats to data in transit:
  1) Eavesdropping
     = overhearing *without any extra effort*
       E.g., admin anyway uses s/w to monitor network traffic to manage the network - in this way she effortlessly eavesdrops on the traffic
  2) Wiretapping
     = overhearing *with some extra effort*
     a) Passive wiretapping
        Pretty similar to eavesdropping but some extra effort
          E.g., starting monitoring s/w usually not used
     b) Active wiretapping – injecting msgs
- Wiretapping technique depends on the communication medium

Threats in transit: eavesdropping and wiretapping (2)

- Wiretapping technique depends on the communication medium

1) Wiretapping cables
   - Via *packet sniffer* for Ethernet or other LAN
     - Msgs broadcast onto Ethernet or other LAN
     - Reads all data packets—not only ones addressed to *this* node
   - By means of *inductance*
     - Using radiation emitted by cable
     - Tap must be close to cable
   - By *splicing* / connecting to cable
     - Can be detected by resistance/impedance change
   - Note: If signal multiplexed (on WANs), wiretapper must extract packets of interest from intercepted data

Threats in transit: eavesdropping and wiretapping (3)

2) Wiretapping microwave

- Signal broadcast thru air, dispersed (cf. Fig. 7-14)

  => accessible to attackers
- Very insecure medium
- Protected by volume —carries a lot of various data, multiplexed

3) Wiretapping satellite links

- Very wide signal dispersion (even k*100 by n*1,000 mi)

  => easy to intercept
- Protected by being highly multiplexed

Threats in transit: eavesdropping and wiretapping (4)

4) Wiretapping optical fiber

- Must be tuned after each new connection made => easy to detect wiretaps (wiretaps destroy „balance")
- Inductive tap impossible (no magnetic radiation for light)
- Easiest to tap at:
  - Repeaters, splices, and taps along the cable
  - Points of connection to computing equipment

5) Tapping wireless

- Typical signal range= interception range: 100-200 ft.
- Wireless communication standards:
  - 802.11b (≤10 Mbps)
  - 802.11a (~ 50 Mbps)
  - 802.11g – most popular currently
  - 802.11n – planned approval: Sept. 2007

cont.

Threats in transit: eavesdropping and wiretapping (5)

- Problem 1: Interception
  - Due to *no* encryption or *weak* encryption standard
  - 85% wireless installations don't provide encryption (!)
  - Standard encryption (WEP) is weak
    - WEP = Wired Equivalent Privacy
    - Stream cipher with 40- or 104-bit key
    - 40-bit key can be broken pretty easily
  - WEP superceded by:
    - WPA (Wi-Fi Protected Access) in 2003
    - Full IEEE 802.11i standard (also known as WPA2) in 2004
- Problem 2: Service theft
  - Popular DHCP protocol (negotiating with client) assigns one-time IP address *without authentication* (of the client)
    - DHCP = Dynamic Host Configuration Protocol
  - Anybody can get free Internet access (after she gets IP)

f. Protocol flaws

- Protocol flaws:
  - Design flaws
    - Proposed Internet protocols posted for public scrutiny
    - Does not prevent protocol design flaws
  - Implementation flaws

g. Types of attacks
g-1. Impersonation (1)

- *Impersonation* = attacker foils authentication and assumes identity of a *valid entity* in a communication
- Impersonation attack may be easier than wiretapping
- Types of impersonation attacks (IA):
  1) IA by guessing
  2) IA by eavesdropping/wiretaping
  3) IA by circumventing authentication
  4) IA by using lack of authentication
  5) IA by exploiting well-known authentication
  6) IA by exploiting trusted authentication

Impersonation (2)

1) Impersonation attacks by guessing

- Ways of guessing:
  - Common word/dictionary attacks
  - Guessing default ID-password pairs
    - E.g., GUEST-guest / GUEST-null / ADMIN-password
  - Guessing weak passwords
- Guessing can be helped by social engg
  - E.g., guess which account might be dead/dormant
    - Read in a college newspaper online that Prof. Ramamoorthy is on sabbatical => guessses that his acct is dormant
  - Social engg: call to help desk to reset password to one given by attacker

Impersonation (3)

2) Impersonation attacks by eavesdropping/wiretaping

- User-to-host or host-to-host authentication must not transmit password in the clear
  - Instead, e.g., transfer hash of a password
  - Correct protocols needed
    - Devil is in the details
  - Example of simple error: Microsoft LAN Manager
    - 14-char password of 67 characters
    - Divided into 2 pieces of 7 chars for transmission
    - Each piece hashed separately
    - To break hash, wiretapper need at most:

      $$67^7 + 67^7 = 2 * 67^7 \text{ attempts}$$

      (as now each 7-char piece can be guessed separately)
    - Should have divided into 2 pieces for transmission *after* hashing, not before (hash 14 not 2 * 7 chrs) => would have 67^{14} possibilities (10 billion times more!)

Impersonation (4)

3) Impersonation attacks by circumventing authentication

- Weak/flawed authentication allows bypassing it
- „Classic" OS flaw:
  - Buffer overflow caused bypassing password comparison
  - Considered it correct authentication!
- Crackers routinely scan networks for OSs with weak/flawed authentication
  - Share this knowledge with each other

4) Impersonation attacks by using lack of authentication

a) Lack of authorization by design

- Example: Unix facilitates host-to-host connection by users already authorized on their primary host
  - .rhosts - list of trusted hosts
  - .rlogin - list of trusted users allowed access w/o authentication
  - Attacker who gained proper id I1 on one host H1, can access all hosts that trust H1 (have H1 and I1 in .rhosts and .rlogin, respectively)

b) Lack of authorization due to administrative decision

- E.g., a bank may give access to public information to anybody under guest-no login account-pasword pair
- „Guest" account can be a foothold for attacker
  - Attacker will try to expand guest privileges to exploit the system

Impersonation (6)

5) Impersonation attacks by exploiting well-known authentic.

- Example: A computer manufacturer planned to use same login-password pair for maintenance account for any of its computers all over the world
- System/network admins often leave default password unchanged
  - Example: „community string” deafult password in SNMP protocol (for remote mgmt of network devices)
- Some vendors still ship computers with one sys admin account installed with a default password

6) Impersonation attacks by exploiting trusted authentication

- Identification *delegated* to trusted source
- E.g., on Unix with .rhosts/.rlogin (see 4a above)
- Each delegation is a potential security hole!
  - Can you really trust the „trusted” source?

E.g., Host A trusts Host B.

User X on Host B can impersonate User Y from Host B.

g-2. Spoofing (1)

- Spoofing — attacker (or attacker's agent) pretends to be a valid entity *without foiling authentication*
  - **Spoof - 1.** To deceive. [...]

    The American Heritage® Dictionary of the English Language: Fourth Edition. 2000
- Don't confuse spoofing with impersonation
  - Impersonation — attacker *foils authentication* and assumes identity of a valid entity
- Three types of spoofing:

1) Masquerading
2) Session hijacking
3) Man-in-the middle (MITM)

Spoofing (2)

1) Masquerading = a host pretends to be another

- Really: attacker sets up the host (host is attacker's agent)
- Masquerading - Example 1:
  - Real web site: Blue-Bank.com for Blue Bank Corp.
  - Attacker puts a masquerading host at: BlueBank.com
    - It mimics the look of original site as closely as possible
  - A mistyping user (who just missed „-") is asked to login, to give password => sensitive info disclosure
  - Can get users to masquerading site by other means
    - E.g., advertise masquerading host with banners on other web sites (banners would just say „Blue Bank"-no „-" there)
- Similar typical masquerades:
  - xyz.org *and* xyz.net masquerade as xyz.com
  - 10pht.com masquerades as lOpht.com (1-I, 0-O)
  - citicar.com masquerades as citycar.com

Spoofing (3)

- Masquerading - Example 2:
  - Attacker exploits web server flaw – modifies web pages
  - Makes no visible changes but „steals" customers
  - E.g., Books-R-Us web site could be changed in a sneaky way:
    - Processing of browsing customers remains unchanged

    BUT
    - Processing of ordering customers modified: (some) orders sent to competing Books Depot
      - Only „some" to mask the masquerade

Spoofing (4)

2) Session hijacking = attacker intercepting and carrying on a session begun by a legitimate entity

- Session hijacking - Example 1
  - Books Depot wiretaps network and intercepts packets
  - After buyer finds a book she wants at Books-R-Us and starts ordering it,

    the order is taken over by Books Depot
- Session hijacking - Example 2
  - Sysadmin starts Telnet session by remotely logging in to his privileged acct
  - Attacker uses hijacking utility to intrude in the session
    - Can send his own commands between admin's commands
    - System treats commands as coming from sysadmin

3) Man-in-the middle (MITM)

*** SKIP "3) Man-in-the middle (MITM)" (this & next slide) – will cover after encryption explained ***

- Similar to hijacking
- Difference: MITM participates in a session from its start

  (session hijacking occurs *after* session established)

...continued....

*** SKIP ***

- MITM – Example: Alice sends encrypted msg to Bob

(a) Correct communication

- Alice requests key distributor for $K_{PUB\text{-}Bob}$
- Key distributor sends $K_{PUB\text{-}Bob}$ to Alice
- Alice encrypts P: $C = E(P, K_{PUB\text{-}Bob})$ & sends C to Bob
- Bob receives C and decrypts it: $P = D(C, K_{PRIV\text{-}Bob})$

(b) MITM attack

- Alice requests key distributor for $K_{PUB\text{-}Bob}$
- MITM intercepts request & sends $K_{PUB\text{-}MITM}$ to Alice
- Alice encr. P: $C = E(P, K_{PUB\text{-}MITM})$ & sends C to Bob
- MITM intercepts C & decrypts it: $P = D(C, K_{PRIV\text{-}MITM})$
- MITM requests key distributor for $K_{PUB\text{-}Bob}$
- Key distributor sends $K_{PUB\text{-}Bob}$ to MITM
- MITM encr. P: $C = E(P, K_{PUB\text{-}Bob})$ & sends C to Bob
- Bob receives C and decrypts it: $P = D(C, K_{PRIV\text{-}Bob})$

Note: Neither Alice not Bob know about MITM attack

g-3. Message confidentiality threats (1)

- Message confidentiality threats include:
  1) Eavesdropping – above
  2) Impersonation – above
  3) Misdelivery
     - Msg delivered to a wrong person due to:
       - Network flaw
       - Human error
         - Email addresses should not be cryptic
           iwalkey@org.com better than iw@org.com
           iwalker@org.com better than 10064,30652@org.com

Message confidentiality threats (2)

4) Exposure

- Msg can be exposed at any moment between its creation and disposal
- Some points of msg exposure:
  - Temporary buffers
  - Switches / routers / gateways / intermediate hosts
  - Workspaces of processes that build / format / present msg (including OS and app pgms)
- Many ways of msg exposure:
  - Passive wiretapping
  - Interception by impersonator at source / in transit / at destination

5) Traffic flow analysis

- Mere existence of msg (even if content unknown) can reveal sth important
  - E.g., heavy msg traffic form one node in a military network might indicate it's headquarters

g-4. Message integrity threats (1)

- Message integrity threats include:
  1) Msg fabrication
  2) Noise

1) Msg fabrication

- Receiver of fabricated msg may be misled to do what fabricated msg requests or demands
- Some types of msg fabrication:
  - Changing part of/entire msg body
  - Completely replacing whole msg (body & header)
  - Replay old msg
  - Combine pieces of old msgs
  - Change apparent msg source
  - Destroy/delete msg

- Means of msg fabrication:
  - Active wiretap
  - Trojan horse
  - Impersonation
  - Taking over host/workstation

2) Noise = unintentional interference

- Noise can distort msg
- Communication protocols designed to detect/correct transmission errors
  - Corrected by:
    - error correcting codes
    - retransmission

g-5. Web site attacks (1)

- Web site attacks – quite common due to:
  - Visibility
    - E.g., web site defacement – changing web site appearance
  - Ease of attack
    - Web site code available to attacker (Menu: View>>Source)
    - A lot of vulnerabilities in web server s/w
      - E.g., 17 security patches for MS web server s/w, IIS v. 4.0 in 18 months

- Common Web site attacks (discussed next):

  1) Buffer overflows
  2) Dot-dot attacks
  3) Exploiting application code errors
  4) Server-side include

1) Buffer overflows

- Attacker feeds pgm much more data than it expects
  - WILL BE DISCUSSED in the "Program Security" Chapter
- iishack - best known web server buffer overflow problem
  - Procedure executing this attack is available

Web site attacks (3)

2) Dot-dot attacks

- In Unix & Windows: '..' points to parent directory
- Example attack: on webhits.dll for MS Index Server
  - Pass the following URL to the server

http://URL/null.htw?CiWebHitsFile=/../../../../../winnt/system32/autoexec.nt

  - Returns *autoexec.nt* file – attacker can modify it
- Other example attacks: Lab Manual – p. 257
  - Using ..%255c.. in URL allows executing arbitrary commands

- Solution to (some) dot-dot attacks:

  1) Have no editors, xterm, telnet, utilities on web server

     => no s/w to be executed by an attacker on web server to help him

  2) Create a fence confining web server

Web site attacks (4)

3) Exploiting application code errors

- Source of problem:
  - Web server may have k*1,000 transactions at a time
  - Might use *parameter fields* (appended to URL) to keep track of transaction status
- Example: exploiting *incomplete mediation* in app (cf. earlier)
  - URL generated by *client's browser* to access web server, e.g.:

http://www.things.com/order/final&custID=101&part=555A&qy=20&price=10&ship=boat&shipcost=5&total=205

  - Instead, *user* edits URL directly, changing price and total cost as follows:

http://www.things.com/order/final&custID=101&part=555A&qy=20&price=1&ship=boat&shipcost=5&total=25

  - User sends forged URL to web server
    - The server takes 25 as the total cost

4) Server-side include

- HTML code for web page can contain *include* commands
- Example
  - Attacker can open telnet session on server (with server's privileges)
  - <!-#exec cmd=/"usr/bin/telnet &"->
- *include exex* (# exec) commands can be used to execute an arbitrary file on the server
- Attacker can execute, e.g., commands such as:
  - chmod – changes access rights
  - sh – establish command shell
  - cat – copy to a file

g-6. Denial of service (attack on avail.) (1)

- Service can be denied:

A) due to (nonmalicious) failures

- Examples:
  - Line cut accidentally (e.g., by a construction crew)
  - Noise on a line
  - Node/device failure (s/w or h/w failure)
  - Device saturation (due to nonmalicious excessive workload/ or traffic)
- Some of the above service denials are short-lived and/or go away automatically (e.g., noise, some device saturations)

B) due to denial-of-service (DoS) attacks = attacks on *availab.*

- DoS attacks include:

1) Physical DoS attacks

2) Electronic DoS attacks

Denial of service (2)

1) Physical DoS attacks – examples:
   - Line cut deliberately
   - Noise injected on a line
   - Bringing down a node/device via h/w manipulation

2) Electronic DoS attacks – examples:

   (2a) Crashing nodes/devices via s/w manipulation
   - Many examples discussed earlier

   (2b) Saturating devices (due to malicious injection of excessive workload/ or traffic)

   Includes:
   (i) Connection flooding
   (ii) SYN flood

   (2c) Redirecting traffic

   Includes:
   (i) Packet-dropping attacks (incl. black hole attacks)
   (ii) DNS attacks

(i) Connection flooding

= flooding a connection with useless packets so it has no capacity to handle (more) useful packets

- ICMP (Internet Control Msg Protocol) - designed for Internet system diagnostic (3rd class of Internet protocols next to TCP/IP & UDP)

  ICMP msgs can be used for attacks

  - Some ICMP msgs:
    - *echo request* – source S requests destination D to return data sent to it (shows that link from S to D is good)
    - *echo reply* – response to echo request sent from D to S
    - *destination unreachable* – msg to S indicating that packet can't be delivered to D
    - *source quench* – S told to slow down sending msgs to D (indicates that D is becoming saturated)

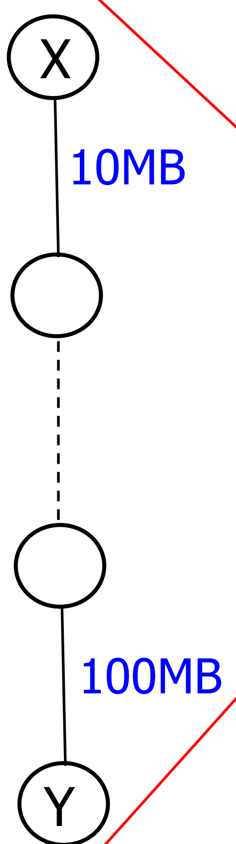
Note: *ping* sends ICMP „echo request" msg to destination D.
If D replies with „echo reply" msg, it indicates that D is reachable/functioning (also shows msg round-trip time).

Denial of service (4) – 2b: Saturating devices – i: Connection flooding

Note: Try ping/echo on MS Windows:
(1) Start>>All Programs>>Accessories>>Command Prompt
(2) ping www.wmich.edu (try: www.cs.wmich.edu, cs.wmich.edu)

- Example attacks using ICMP msgs
  (i1) Echo-chargen attack
  - *chargen* protocol – generates stream of packets; used for testing network
  - Echo-chargen attack example 1:
    (1) attacker uses chargen on server X to send stream of *echo request* packets to Y
    (2) Y sends *echo reply* packets back to X
    This creates endless „busy loop" beetw. X & Y
  - Echo-chargen attack example 2:
    (1) attacker uses chargen on X to send stream of *echo request* packets *to X*
    (2) X sends *echo reply* packets back to itself

Denial of service (5) – 2b: Saturating devices – i: Connection flooding



(i2) Ping of death attack, incl. smurf attack

- Ping of death example :
  - (1) attacker uses ping after ping on X to flood Y with pings (ping uses ICMP echo req./reply)
  - (2) X responds to pings (to Y)

  This creates endless „busy loop" beetw. X & Y

Note: In cases (i1-ex.1) & (i2):

- if X is on 10 MB connection and path to victim Y is 100 MB, X can't flood Y
- if X is on 100 MB connection and path to victim Y is 10 MB, X can easily flood Y

- Smurf attack example:
  - (1) attacker spoofs source address of ping packet sent fr. X – appears to be sent by Z
  - (2) att. broadcasts spoofed pkt to N hosts
  - (3) all N hosts echo to Z – flood it

(ii) SYN flood DoS attack

- Attack is based on properties/implementation of a *session* in TCP protocol suite
- *Session* = virtual connection between protocol peers
  - Session established with *three-way handshake* (S = source, D = destination) as follows:
    - S to D: SYN
    - D to S: SYN+ACK
    - S to D: ACK
    - Now session between S and D is established
  - D keeps *SYN_RECV queue* which tracks connections being established for which it has received no ACK
    - Normally, entry is in SYN_RECV for a short time
    - If no ACK received within time T (usu. a few minutes), entry discarded (connection establ. times out)

Denial of service (7) – 2b: Saturating devices – ii: SYN flooding

    - Normally, size of SYN_RECV (10-20) is sufficient to accommodate all connections under establishment

- SYN flood attack scenario
    - Attacker sends many SYN requests to D (as if starting 3-way handshake)
    - Attacker never replies to D's SYN+ACK packets
    - D puts entry for each unanswered SYN+ACK packet into SYN_RECV queue
    - With many unanswered SYN+ACK packets, SYN_RECV queue fills up
    - When SYN_RECV is full, no entries for legitimate unanswered SYN+ACK packets can be put into SYN_RECV queue on D

      => nobody can establish legitim. connection with D

Denial of service (8) – 2b: Saturating devices – ii: SYN flooding

- Modification 1 of SYN flood attack scenario:
  attacker spoofs sender's address in SYN packets sent to D
  - Question: Why?

Denial of service (9) – 2b: Saturating devices – ii: SYN flooding

- Modification 1 of syn flood attack scenario:
  attacker spoofs sender's address in SYN packets sent to D
  - Question: Why?
  - Answer:
    To mask packet's real source, to cover his tracks

- Modification 2 of SYN flood attack scenario:
  attacker makes each spoofed sender's address in SYN packets different
  - Question: Why?

Denial of service (10) – 2b: Saturating devices – ii: SYN flooding

- ...

- Modification 2 of SYN flood attack scenario:
  attacker makes each spoofed sender's address in SYN packets different
  - Question: Why?
  - Answer:
    If all had the same source, detection of attack would be simpler (too many incomplete connection requests coming from the same source look suspicious)

(2c) Redirecting traffic (incl. dropping redirected packets)

(i) Redirecting traffic by advertising a false best path

- Routers find best path for passing packets from S to D
  - Routers advertise their conections to their neighbors (cf. Disemination of routing info - Slide 28; ALSO: P&P, p.380—Routing Concepts + Fig. 7-2)
- Example of traffic redirection attack:
  - Router R taken over by attacker
  - R advertises (falsely) to all neighbors that it has the best (e.g., shortest) path to hosts H1, H2, ..., Hn
  - Hosts around R forward to R all packets addressed to H1, H2, ..., Hn
  - R drops *some* or *all* these packets

    drops *some* => packet-dropping attack

    drops *all* => black hole attack

    (black hole attack is spec. case of pkt-drop. attack)

(ii) Redirecting traffic by DNS attacks

- Domain name server (DNS)
  - Function: resolving domain name
    = converting domain names into IP addresses
    - E.g., aol.com → 205.188.142.182
  - DNS queries other DNSs (on other hosts) for info on unknown IP addresses
  - DNS caches query replies (addresses) for efficiency
- Most common DNS implementation:
  BIND s/w (BIND = Berkeley Internet Name Domain)
  a.k.a. *named* (named = name daemon)
  - Numerous flaws in BIND
    - Including buffer overflow
- Attacks on DNS (e.g., on BIND)
  - Overtaking DNS / fabricating cached DNS entries
    - Using fabricated entry to redirect traffic

g-7. Distributed denial of service (attack on availability)

[Fig. courtesy of B. Endicott-Popovsky]

- DDoS = distributed denial of service
- Attack scenario:
  1) Stage 1:
     - Attacker plants Trojans on many target machines
       - Target machines controlled by Trojans become *zombies*
  2) Stage 2:
     - Attacker chooses victim V, orders zombies to attack V
     - Each zombie launches a separate DoS attack
       - Different zombies can use different DoS attacks
         - E.g., some use syn floods, other smurf attacks
         - This probes different weak points
       - All attacks together constitute a DDoS
     - V becomes overwhelmed and unavailable

       => DDoS succeeds

g-8. Threats to active or mobile code (1)

- *Active code* / *mobile code* = code pushed by server S to a client C for execution on C
  - Why S doesn't execute all code itself? For efficiency.
    - Example: web site with animation
      - Implementation 1 — S executing animation
        - Each new animation frame must be sent from S to C for display on C
          => uses network bandwidth
      - Implementation 2 — S sends animation code for execution to C
        - C executes animation
        - Each new animation frame is available for dispaly locally on C
      - Implementation 2 is better: saves S's processor time and network bandwidth

Threats to active or mobile code (2)

- Isn't active/mobile code a threat to client's host?
  It definitely is a threat (to C-I-A)!

- Kinds of active code:
  1) Cookies
  2) Scripts
  3) Active code
  4) Automatic execution by type

1) Cookies = data object sent from server S to client C that can cause unexpected data transfers from C to S
   - Note: Cookie is data file not really active code!
   - Cookies typically encoded using S's key (C can't read them)

Threats to active or mobile code (3)

- Example cookies

  a - from google.com, b - from wmich.edu

a)
PREF ID=1e73286f27d23c88:TM=1142049583:LM=1142049583:S=gialJ4YZeKozAsGT
google.com/
1647
2719878336
32222645
3392857739
29856332 *

b)
CPSESSID

wmich.edu/
1647
3757208800
29856325
3542538800
29856325
*
WebCTTicket

wmich.edu/
1647
3757208800
29856325
3542538800
29856325
*

Note: Both cookies are „doctored"
for privacy reasons.

Threats to active or mobile code (4)

- Types of cookies:
  - Per-session cookie
    - Stored in memory, deleted when C's browser closed
  - Persistent cookie
    - Stored on disk, survive termination of C's browser

- Cookie can store anything about client C that browser running on C can determine, including:
  - User's keystrokes
  - Machine name and characteristics
  - Connection details (incl. IP address)
  - ...

Threats to active or mobile code (5)

- Legitimate role for cookies:
  - Providing C's context to S
    - Date, time, IP address
    - Data on current transaction (incl. its state)
    - Data on past transactions (e.g., C user's shopping preferences)
    - ...
- Illegitimate role for cookies:
  - Spying on C
  - Collecting info for impersonating user of C who is target of cookie's info gathering
    - Attacker who intercepts X's cookie can easily impersonate X in interactions with S
- Philosophy behind cookies:

  Trust us, we know what's good for you!

  Hmm... They don't trust you (encode cookie) but want you to trust them.

Threats to active or mobile code (6)

2) Script – resides on server S; when executed on S upon command of client C, allows C to invoke services on S

- Legitimate interaction of browser (run on C) w/ *script* (run by *script interpreter* on S)
  - On C:
    - Browser organizes user input into script params
    - Browser sends string with script name + script params to S (e.g., http://eStore.com/order/custID=97&part=5A&qy=2&...)
  - On S:
    - Named script is executed by script interpreter using provided params, invoking services called by script

- Attacker can intercept interaction of browser w/ script
  - Attacker studies interaction to learn about it
  - Once browser & script behavior is understood, attacker can handcraft string sent fr. browser to script interpreter
    - Falsifies script names/parameters
      - Cf. incomplete mediation example with false price (Slide 80)

Threats to active or mobile code (7)

- Why is it easy to manipulate browser-script interaction?
  - Programmers often lack security knowledge
    - Don't double-check script params
    - Some scripts allow including arbitrary files
    - Some scripts allow execution of arbitrary commands
  - They often assume that no users are malicious
  - Time pressure/management pressure

- Scripting language *CGI* (*Common Gateway Interface*)
  - Enables a client web browser to request data from a program executed on the Web server [Wikipedia]
  - Not really a language – rather standard for passing data between C and S's script interpreter
  - Example CGI string:

http://www.tst.com/cgi-bin/query?%0a/bin/cat%20/etc/passwd

    - %nn represents ASCII special characters
    - E.g., %0a = line feed (new line), %20 = space
    - What is it doing? / Why need %20 to insert a space?

Threats to active or mobile code (8)

- HTTP w/o and with CGI [cf. http://www.comp.leeds.ac.uk/Perl/Cgi/what.html]
  - HTTP without CGI:
    - When Web browser looks up URL, browser contacts HTTP server with this URL
    - HTTP server looks at filename named in URL & that *file* is sent back
    - Browser displays file in the appropriate format
  - HTTP with CGI:
    - When file in certain directory is named in URL (sent by browser), file is not sent back but executed as CGI script (a pgm)
    - Only CGI script *output* is sent back for browser to display.
      - CGI scripts are programs which can generate and send back anything: sound, pictures, HTML documents, and so on

Threats to active or mobile code (9)

- Examples: escape-character attacks
  - Attack 1: CGI string instructs script interpreter to send copy of password file to client C:

http://www.tst.com/cgi-bin/query?%0a/bin/cat%20/etc/passwd

  - Attack 2: CGI string includes substring that instructs script interpreter to remove all files from current dir:
    ...<!-#exec cmd=”rm *”>

- Other scripting solution:
  Microsoft’s active server pages (ASP)

- Conclusions: A server should never trust anything received from a client!
  - Bec. the received string can be fabricated by attacker rather than being generated by a legitimate pgm (e.g.,a browser)

Threats to active or mobile code (10)

3) Active code (Recall: code pushed by S to C for execution on C)

- As demand on server S's computing power grows, S uses client C's computing power
  - S downloads code to C (for execution on C), C executes it

- Two main kinds of active code:

  (a) Java code (Sun Microsystems)

  (b) ActiveX controls (Microsoft)

(a) Java code

- Designed to be truly machine-independent
  - Java pgm: machine-independent *Java bytecode*
  - Java bytecode executed on *Java Virtual Machine* (*JVM*)
    - JVM can be implemented for different platforms & different system components
      - E.g., JVM for Netscape browser

Threats to active or mobile code (11)

- Java security
  - JVM includes *built-in security manager*
  - Java is strongly typed
    - Enforces type checking
  - Java pgms run in a *sandbox*
    - *Sandbox* = restricted resource domain from which pgm can't escape
  - Java 1.2 had some vulnerabilities
    - Some of it security flaws were *not* design flaws
      - Result of security-usability tradeoff
    - Java 1.2 was a response to Java 1.1
      - Java 1.1 very solid but too restrictive for programmers
        - E.g., could not store permanently on disk, limited to procedures put into sandbox by security manager's policy
  - Security flaws in JVM implementations
    - JVM in Netscape browser: no type checking for some data types
    - JVM in MS Internet Explorer: similar flaws

Threats to active or mobile code (12)

- Current (in September 2004): Java 5.0 (internally known as Java 1.5)
- Hostile applet

  = downloadable Java code that can harm client's system

  Can harm because:
  - Not screened for security when dowloaded
  - Typically runs with privileges of invoking user
- Preventing harm by Java applets:
  - Control applets' access to sensitive system resources
  - Protect memory: prevent forged pointers and buffer overflows
  - Clear memory before its reuse by new objects, must perform garbage collection
  - Control inter-aplet communication & applets' effects on environment

Threats to active or mobile code (13)

(b) ActiveX controls

- Allows to download object of arbitrary type from S to C
- Risks of downloading ActiveX controls:

  After object of type T is downloaded:
  - If handler (or viewer) for type T is available,
    it is invoked to present object
    - E.g., after file.doc downloaded, MS Word is invoked to open file.doc ← BIG security risk!
  - If no handler for type T exists on C,
    C asks S for handler for T then uses it to present object
    - E.g., attacker defines type .bomb
      After file.bomb is downloaded by C, C asks S for handler for type .bomb! ← HUGE security risk!

Threats to active or mobile code (14)

- Preventing (some) risks of downloading:
  Prevent arbitrary downloads
  - Authentication scheme to verify code *origin*
    - Downloaded code is *digitally signed* (to be studied)
      - Could use a digital certificate including a signature of a trusted third party (to be studied)
    - Digital signature verified before execution
  - Problems with this scheme:
    - It does *not* verify *correctness* of code
    - Existing vulnerabilities allow ActiveX code to bypass authentication

4) Automatic execution by type

= automatic invocation of file processing program implied by file type

- Two kinds of auto exec by type:
  (a) File type implied by file extension
  - e.g., MS Word automatically invoked for *file.doc*

  (happens also in other cases, e.g., for ActiveX controls)

  (b) File type implied by embedded type
  - File type is specified within the file
  - Example:
    - File named „class28" *without extension* has embedded info that its type is „pdf"
    - Double-clicking on *class28* invokes Adobe Acrobat Reader
- Both kinds of auto exec by type are BIG security risks!

Threats to active or mobile code (16)

- Security risks for auto exec based on file type
  - Text files (without macros!)
  - Files with active content
    - Incl. text files with macros
  - Executable files

Security Risk

- Avoid automatic opening of files by built-in handlers
  - Whether it has extension or not
  - Whether implied by file extension or by embedded type

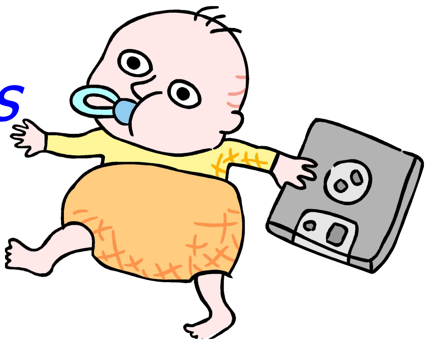
g-9. Scripted and complex attacks

1) Scripted attacks = attacks using *attack scripts*

- Attack scripts created by knowledgeable crackers

BUT

- Can be run even by ignorant *script kiddies*
  - Just download and run script code
    - Script selects victims, launches attack



[Fig. courtesy of B. Endicott-Popovsky]

- Scripted attacks can cause serious damage
  - Even when run by script kiddies

2) Complex attacks = multi-component attacks using miscellanous forms of attacks as its building blocks

- Bldng block example: wiretap for reconaissance, ActiveX attack to install a Trojan, the Trojan spies on sensitive data

- Complex attacks can expand target set & increase damage

h. Summary of network vulnerabilities

- See Table 7-4, p. 426 –
  A classification of network vulnerabilities
  (not quite „clean" taxonomy — overlapping classes)