ITC 502 Computer Network Security M6L3

g) Access controls
   i. ACLs on routers
   ii. Firewalls
h) Intrusion Detection Systems: alarms and alerts
i) Honeypots
j) Traffic flow security
k) Review of network security controls

g) Access controls (1)

- Before user is allowed access to network resources, must know:
  - Who needs access => authentication
  - What and how will be accessed => *access controls*

- Access controls include:

1) ACLs (Access Control Lists) on router
2) Firewalls

h) Intrusion Detection Systems: alarms & alerts

- Example of 2-layer network protection
  - Provided by router (Layer 1) AND firewall (Layer 2)
  - Fig. 7-33, p. 452

- We can add one more layer of protection: *intrusion detection systems* (*IDS*) = device placed within protected network for monitoring for illegitimate actions in order to detect attacks in progress (beginning, advanced) or after they have occurred
    - E.g.: Can detect reconaissance & *alert* sysadmin or secadmin, raise *alarm*, thus preventing „real" attack

    OR
    - Can detect that attack has already occurred & raise *alarm*, starting system recovery actions
  - IDS is a.k.a. IPS = *intrusion protection system*
    - A marketing gimmick?
  - IDS can be Layer 3 of layered network protection
  - To be discussed in detail soon

i) Honeypots

- Honeypot – system built as a bait attracting attackers
  - Once attackers take the bait:
    - They are observed to learn how they behave/operate
      - New attacks / Prefered targets / ...
    - They are traced to catch them or scare them off
      - Or at least trace enough to be able to threaten them with identifying them if they don’t stop
    - They are diverted from really valuable attack targets
      - E.g., diverted to *phony* credit card database while *real* credit card database remains obscure to them

- User lessons learned (thanks to honeypots) to build better countermeasures

j) Traffic flow security (1)

- Threat: attacker infering occurrence/location of some event / structure from intensity of encrypted network traffic

  (If not encrypted, no need to infer – attacker can simply read all)
  - Example 1: Inference that traffic between Thinges Corp. and bankruptcy lawyer precedes declaration of bankruptcy by Thinges
  - Example 2 (old): Battlefield network: Busiest network node is at enemy's HQs

- Solution 1: Masking by steady traffic volume
  - X and Y always send the same volume of *encrypted* traffic between them
  - If X has nothing to communicate to Y, X sends meaningless padding packets to Y (Y behaves analogously)

k) Review of network security controls

- Table 7-4, p. 426 provided classification of network vulnerabilities (during our earlier discussion of threats)
- Table 7-7, p. 454 lists controls for each of these classes of network vulnerabilities — it shows that:
  - There are many great network security controls
  - Most are used also in environments other than networks (including applications and OSs)
  - Three of these controls are specific to networks:
    - Firewalls / IDSs / encrypted e-mail

    We shall discuss them in some detail next
- Table 7-7 is a great reference for network security controls!
  - Use it often
    - If you can get copyright permission from publisher, I'd advise you to copy it and post above your desk
    - Otherwise, make your own notes based on it

End of Class

© by Leszek T. Lilien, 2005

7. Security in Networks

...

7.3. Networks Security Controls

...

d) Encryption
e) Message content integrity controls
f) Strong authentication

---

g) Access controls
   i. ACLs on routers
   ii. Firewalls
h) Intrusion Detection Systems: alarms and alerts
i) Honeypots
j) Traffic flow security
k) Review of network security controls

7.4. Network Security Tools

7.4.1. Firewalls

a) Introduction
b) What is a firewall
c) Firewall design
d) Types of firewalls
e) Comparison of firewall types
f) Example firewall configurations
g) What firewalls can—and can't—block

7.4. Network Security Tools

[Fig: B. Endicott-Popovsky]

- Network security tools

  7.4.1. Firewalls

  7.4.2. Intrusion Detection Systems

  7.4.3. Secure E-Mail

6.2.1. Firewalls

- Outline
  - a) Introduction
  - b) What is a firewall
  - c) Firewall design
  - d) Types of firewalls
    - i. Packet filters
      - (i-1) Simple packet filters
      - (i-2) Stateful packet filters
    - ii. Application proxies
      - (ii-1) Guards (“top model” subcategory)
    - iii. Personal firewalls
  - e) Comparison of firewall types
  - f) Example firewall configurations
  - g) What firewalls can—and can’t—block

--[OPTIONAL]-- a. Introduction

- Firewalls
  - Invented in the early 1990’s
  - But idea related to *reference monitors* from 1970’s
    - What is reference monitor?
      - OS includes *kernel* / *core* / *nucleous* – responsible for lowest-level functions
        - Incl. synchronization, inter-process communication, msg passing, interrupt handling
      - *Security kernel* – provides security mechanisms for entire OS
        - Incl. security interfaces among h/w, OS, other parts of computing system
      - Typically, security kernel is a part of OS kernel
      - *Reference monitor* is portion of security kernel that controls access to objects (controls „references” to objects)

b. What is a firewall (1)

© by Leszek T. Lilien, 2005

- Firewall = device (h/w), or software, or combination of both designed to prevent unauthorized users from accessing network and/or single workstation

  Wall between protected local (sub)net & outside global net

  - Inspect each individual inbound or outbound packet of data sent to / from protected system
  - Check if it should be blocked or allowed to enter

- Firewalls keep „bad things" *out*, keep sensitive info *in*
  - Security policy specifies what are „bad things"
    - E.g., requires that traceroute & ping -o can't see internal hosts
  - F. protect against security threats from external network
  - F. are effective in protecting local subnet incl. its sensitive info

What is a firewall (2)

- Examples of *security policy requirements* w.r.t. firewalls:
  - Block any access *from* the outside, allow all accesses *to* the outside
  - Allow *"from"* accesses only for certain activities OR only to/from certain subnets/hosts/apps/users
    - E.g., prevent outside access to subnet hosts except for mail server accesses

- Choice of *default firewall behavior*

  1) *Default permit*
     - „That which is not expressly forbidden is allowed"

  2) *Default deny*
     - „That which is not expressly allowed is forbidden"

  - Users prefer *default permit*, security experts prefer *default deny*
  - Sysadmin must make the choice

c. Firewall design (1)

- Firewall design principles:
  - Small / simple enough for rigorous analysis
    - KISS principle (= „Keep It Simple, Stupid”)
    - Simple firewall functionality
  - Tamperproof
    - Typically well isolated (=> highly immune to modifications)
      - On a separate computer
      - With direct connections only to the outside networks and to the inside network
  - Designed to be always invoked
    - Efficient enough not too be a bottleneck
    - Placed strategically
      - All network accesses that we want to control pass through it

--[OPTIONAL]-- Firewall design (2)

- General firewall techniques:
  1) Service control
     - Type of service: inbound or outbound
  2) Traffic filtering — based on IP address & TCP port nr
     - Provide proxy software to receive or interpret service request before passing it on
     - Could also host server software (e.g. Web or mail service)
       - Not recommended
         - Complicates it (more code => more vulnerabilities)
  3) User Control
     - Control access to service using ACLs
  4) Behavior Control
     - E.g. filter e-mail for spam

--[OPTIONAL]-- Firewall design (3)

- Basic firewall characteristics
  - All traffic (incoming / outgoing) must pass thru firewall
  - Only authorized traffic allowed to pass
  - Firewall itself must be immune to penetration
    - I.e. must use trusted system w/ secure OS (min. size/complexity)
    - Usually implemented on dedicated device
      - Dedicated = only firewall functions performed on this device
    - Firewall *code* must be very well protected

- Basic kinds of firewalls:
  - Hardware firewalls
    - More common
    - Implemented on router level
      - More expensive / more difficult to configure
  - Software firewalls
    - Used in single workstations
    - Less expensive / Easier to configure

d. Types of firewalls (1)

- Types of firewalls
  - i. *Packet filters* / packet filtering firewalls
    - (i-1) *Simple* packet filters / (simple) packet filtering gateways / screening routers
    - (i-2) *Stateful* packet filters / stateful inspection firewalls
  - ii. *Application proxies* / proxy firewalls / application-level gateways
    - (ii-1) Guards (a special case of app proxies)
  - iii. Personal firewalls

Types of firewalls (2)

- Firewall properties:
  - Packet filter properties:
    - Transparent
    - Does not change traffic, only passes it
  - Proxy properties:
    - Active
    - Intercepts traffic & acts as an intermediary

- Different firewall types needed for different needs

  „Different strokes for different floks" — e.g.:

  - Simple packet filters / screening routers – implement simplistic security policies
    - Simple is best if sufficient to counter exisiting threats well
  - App proxies – much richer capabilities

--[OPTIONAL]-- Types of firewalls (3)

- Firewall is a type of host

  Even some routers are host-based

  To have better tools available (editors, programming tools)

  - Programmable
  - Minimal functionality
    - Reduces vulnerabilities
      - Small = > less complex => fewer vulnerabilities
    - Reduces motivation for attacks
      - No password files to steal, etc.

(i) Packet filters (1)

- *Packet filters* — a.k.a. *packet filtering firewalls*
  - (i-1) *Simple* packet filters („memoryless”)
  - (i-2) *Stateful* packet filters (with „memory”)

- *Basis* for packet filtering
  1) Packet IP addresses
     - Filtering based on both source/destination addresses
  2) Port number determines TCP transport protocol type
     - Recall “port→protocol” mapping in TCP: 21→FTP, 23→Telnet, 25→SMTP, 80→HTTP, 110→POP, 161→ SNMP, etc.
     - Filtering based on port nr

- Packet filtering firewalls do not „see” other packet fields
  - See only IP address ‘ transport protocol type
  - E.g., can not allow only some Telnet commands OR exclude only some other Telnet commands

--[OPTIONAL]-- (i) Packet filters (2)

- Examples of packet filtering – see text

  1a) Packet address filtering (cf. Fig. 7-35, p. 459)
  Can block traffic from specific subnets and/or allow traffic from specific subnets

  1b) Packet address filtering (cf. Fig. 7-36, p. 460)
  Can block traffic from specific IP addresses and/or allow traffic from specific IP addresses

  2) Filtering based on transport protocol (cf. Fig. 7-35, p. 459)
  Can block traffic using Telnet protocol (port 23) but allow HTTP traffic (port 80)

- To avoid overburdening router, firewall can run on device behind router (on subnet side)

© by Leszek T. Lilien, 2005

(i-1) Simple packet filters (1)

- *Simple packet filters* / *(simple) packet filtering gateways* / *screening routers* — simplest firewall type
- Simple packet filters (PFs) are *memoryless*
  => can *not* perform attack detections that require remembering *state* (history/context) of ≥ N last pkts
  - E.g., can *not* see that prev. & curr. pkt indicate attack
  - "*Attack signature*" (i.e., attack pattern) *would* be clearly visible if both pkts were put together
    - Example: Certain attack script known to use Telnet (port 23) and then SNMP (port 161)

      The same source address in previous pkt, using port 23, and in current packet, using port 161, constitutes attack signature
  - Why need to remember ≥ N last pkts?
    - TCP pkts arrive in order different than sending order
      => remembering ≤ N last pkts is not enough

--[OPTIONAL]-- (i-1) Simple packet filters (2)

- Cheating simple (memoryless) PF:
  - Attacker divides pkt (including attack signature) into many v. short pkts
    - Attack signature (pattern) *would* be visible in original long pkt
      - Even memoryless simple PF *would* detect it
    - Pattern of attack is completely invisible in any *single* short pkt
      - => memoryless simple PF is unable to detect attack
    - Additionally, TCP pkts arrive in order different than their sending order

      => remembering just last packet would not be enough – must remember N last packets

--[OPTIONAL]-- (i-1) Simple packet filters (3)

- One very important task for simple packet filtering gateways: Validating inside IP addresses
  - Inside hosts trust more other inside host
  - Simple filtering assures that no external source can masquerade as internal source
    - Blocks any packet coming from outside network that claims to be sent by internal host
      - Cf. Fig. 7-37, p. 460

--[OPTIONAL]-- (i-1) Simple packet filters (4)

- Simplicity of inspection is both disadvantage & advantage
  - Disadvantage because of high granularity
    - E.g., can block all Telnet coomands, but can *not* block only selected telnet commands
  - Advantage beacuse reduces complexity
    - Filtering rules to block, e.g., only selected Telnet traffic would have to be much more detailed

      => more complex

      => more vulnerable

(i-2) Stateful packet filters

- *Stateful packet filters* — a.k.a. *stateful inspection firewalls*
  - Keep state/history/context of ≥ N previously seen pkts

    => stateful packet filters *have* memory
  - This allows detection of some attacks that simple PFs can *not* detect
  - Still limited to detection based on IP address & TCP transport protocol type (port nr)

(ii) Application proxies (1)

- *Application proxies* / proxy firewalls / *application-level gateways* / *application proxy gateways*

  Note: The term *bastion host* (used in text) should *not* be used as a synonym. Bastion host is a host that serves as a *platform* for app proxy or circuit-level proxy [Stallings, Crypto&Net.Sec, p.625].

- Application proxies include — as a special case

  (ii-1) Guards

- App proxy firewalls fix basic problem with packet filtering firewalls because they:
  - See all pkt data (not just IP adresses and port #s)
  - (In addition, they are stateful => can analyze multiple pkts)

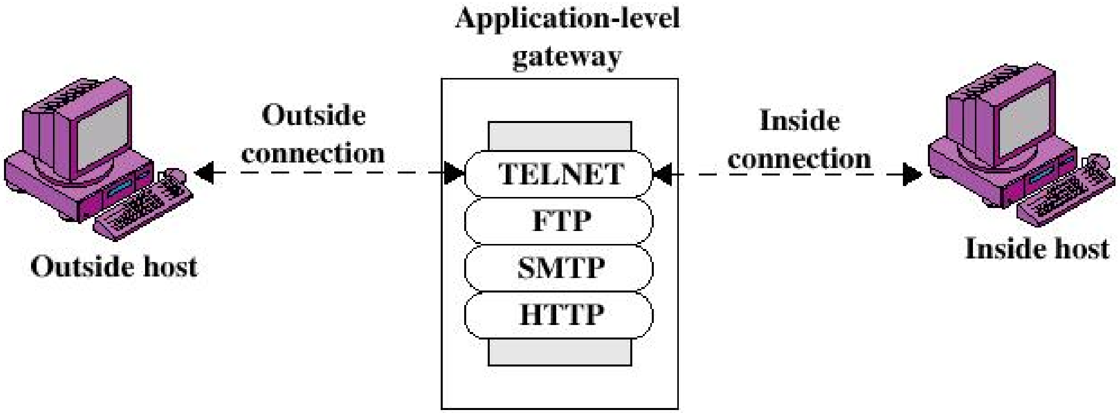
  => can detect/derail more sophisticated attacks
  - Can filter out harmful commands in pkt stream

--[OPTIONAL]-- (ii) Application proxies (2)

- For example, app proxies can prevent:
  - Use of back door open to pkts inbound to SMTP port (Port 25)
  - Flawed application run by user U (e.g., an e-mail agent) has all U’s privileges => can cause damage

(ii) Application proxies (3)

- Act as mediators/censors (!) of app-level traffic – like benevolent „woman-in-the middle” ☺ (not an official term!)
  - They “censor” insecure actions
  - Maybe a rare case of a truly benevolent censor
- Ex. scenario of using app proxy gateway G: [cf. ibid, p.624]
  - Extern. user U tries to Telnet to host H protected by G
  - G intercepts U’s packets
  - G acts as H would: asks U for id+pwd
  - U replies w/ id+pwd
  - G logs in into H on behalf of U
  - G relays H’s msgs to U
  - Etc., etc.



--[OPTIONAL]-- (ii) Application proxies (4)

- Examples of app proxy activities
  - Preventing outsiders from modifying company’s online price list
  - More - see bulleted list on p. 462
- App proxy must implement code for given app (e.g., for Telnet) to be able to perform service to this app
- Netadmin can configure app proxy to support only selected features of an app
  - Unsupported features are considered too risky

    => not available

--[OPTIONAL]-- (ii) Application proxies (5)

© by Leszek T. Lilien, 2005

- App proxies provide higher level of security than packet filters (PFs)
  - PFs try to deal with *all potentially* deployable applications that could use TCP/IP (*default permit* philosophy)
  - App proxy considers only *few* allowable apps among ones *actually* deployed in a given system (*default deny* philosophy)
  - App proxy can easily log/audit traffic at app level (vs. transport level for PFs)

- Prime disadvantage of app proxies: Processing overhead for each app-level connection
  - 1 connection split into 2 *logical* connections
    - With "woman-in-the-middle" ☺
    - Circuit-level gateways (another proxy subcategory) splits 1 TCP connection into 2 TCP connections

(ii) Application proxies (6)

(ii-1) Guards = most sophisticated category of app proxies ("top model")

- Limited only by what is computable (& by human creativity)
- No sharp boundary between app proxies and guards
  - At some point of upgrading app proxy, it becomes a guard

--------[OPTIONAL]------->

- Examples of guard activities
  - Limiting nr of msgs (or nr of msg characters) that a student may e-mail per week
    - Easiest if done by gurad monitoring mail transfer protocol
  - More - see bulleted list on p. 464

(iii) Personal firewalls

- Regular firewalls protects *subnetworks*
  Personal firewalls protect *single hosts*
  - For small business / home office / home
  - Can be used to complement conventional firewall
    - Next line of defense
    - Customized to user(s) of particular host
  - Firewall capabilities at a lower price

- Personal firewall is application program
  - Products include: Norton Personal Firewall (Symantec), McAfee Personal Firewall, Zone Alarm (Zone Labs)

- Personal firewall also enforces certain security policy
  - E.g., if you're using default personal firewall's policy on your computer, see its rules
  - Combine it with antivirus software for more effective protection & with automatic (or very frequent manual) OS and antivirus s/w updates

--[OPTIONAL]-- e. Comparison of firewall types

- Comparison of firewall types
  - See Table 7-8, p. 465
  - Criteria:
    - Complexity
    - Part of packets visible to firewall
    - Diffculty of auditing
    - Basis for screening
    - Difficulty of configuring

f. Example firewall configurations

- Example firewall configurations
  - Subnet with screening router (simple packet filtering)
    — Fig. 7-39, p. 466
  - Subnet with proxy gateway (app proxy)
    — Fig. 7-40, p. 467
  - Subnet with simple PF & app proxy
    — Fig. 7-41, p. 467
    - Note:
      The LAN between outer firewall ("screening router" in the fig) and the inner firewall ("proxy firewall" in the fig) constitutes *DMZ* (*demilitarized zone*)

g. What firewalls can—and can't—block

- Firewalls are not a panacea - only a *perimeter* protection
- Points 2 remember about firewalls — see text, p.466-467
  - Can protect environment only if control its whole perimeter
  - Do not protect data outside the perimeter
  - Are most visible subnet component – attractive attack targets
  - Must be correctly configured, & config must be periodically updated
  - Firewall platforms should not havye any s/w that could help attacker who penetrates firewall in subsequent exploits
  - Firewalls exercise very limited control over content they let in
    - Other means of verifying/enforcing accuracy/correctness must be used inside perimeter

End