CS 194-1 (CS 161)
Computer Security

Lecture 10

Secure Channels and Firewalls

October 2, 2006
Prof. Anthony D. Joseph
http://cs161.org/

Goals for Today

- Motivation for Firewalls
- Defining and Enforcing a Security Policy
- Packet Filters and Rulesets
- Reference Monitors
- Virtual Private Network (VPN) Example

The Motivation for Firewalls

- Suppose you are given a machine, and asked to harden it against external attack
  - How do you do it?
- One starting point:
  - Examine network services the machine provides
  - If any services are buggy/have security holes, hacker might penetrate via that application
- Bugs are inevitable and in security-critical applications can lead to security holes
- *Key Observation:*
  - *The more network services your machine runs, the greater the risk*

Least Services Principle

- Simple way to reduce external attack risk
- Turn off unnecessary network services
  - Disable non-essential or insecure (unencrypted) network-accessible apps
  - Or, build stripped-down box running least amount of necessary code
  - Idea: any code you don't run, can't harm you
- For each required network service:
  - Double-check its implementation and config.
  - Take every precaution to render its use safe
- Intuitive, effective approach for 1-2 machines
  - *But, what happens when we scale things up?*

Your Job: Enterprise Security Chief

- Have to protect company's computing infrastructure/networks from external attack
  - How are you going to do it?
- What if company has 1,000's of computers?
  - May have many different OS's and hardware
  - Different users have different needs -> different necessary services
  - Constantly buying/upgrading machines
  - May not have accurate list of all machines (what happens if you miss one?)
- *Sheer management complexity makes hardening each machine individually infeasible*

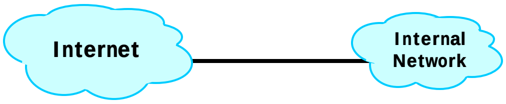
Targeting a Risk Factor

- One big risk factor: the number of network services that are accessible to outsiders
- This suggests a possible defense
  - Reduce risk by blocking, *in the network*, outsiders from being able to access many network services running on company machines
- Exactly the concept behind *firewalls*
  - The firewall is a device designed to block outside (external) access to network services running on company (internal) machines

Two Key Questions

- What is our security policy?
  - Which network services should be externally visible
  - Which ones should be blocked?
  - How do we distinguish insiders from outsiders?
- How will we enforce this security policy?
  - How do we build a firewall that does what we want?
  - What are the implementation issues?
- Need to tackle each question

Security Policy



- How do we decide what is inside, and what is outside?
  - Might trust all company employees, but not trust anyone else (very simple threat model)
    - » Define internal network to contain machines owned by trusted employees, and the external world to include everything else
  - Our link to ISP would be the link between these two worlds

Simple Security Policy: *Outbound-only*

- Distinguish between inbound and outbound conns
  - Inbound connections are attempts by external users to connect to services on internal machines
  - Outbound connections are attempts by internal users to contact external services
- Outbound-only policy permits all outbound connections
  - Reasoning: trust internal users, so let them open connections, but deny all inbound connections
  - Effect: Our network svcs are not externally visible (still accessible to internal users)
- Does this work?

Problems with Outbound-only Policy

- Won't work for large organization – can't run webserver, FTP server, …
- Need more flexibility
  - Think of security policy as a type of access control policy
- Two subjects:
  - Generic inside user (company employee)
  - Anonymous external user (everyone else)
- Objects:
  - Set of services running on inside machines
    - » 1000 machines each running 5 network services yields 5000 objects

Access Control Policy

- Specifies whether subject has permission to access object
- FW enforces simple access control policy:
  - Permit inside users to connect to any service
  - External users restricted:
    - » Permit connections to services intended to be externally visible
    - » Deny connections to services not intended to be externally visible
- Identifying a Security Policy
  - Deciding which svcs external users can access
  - Two philosophies: *Default-allow* and *Default-deny*

Default-Allow

- Default is every network service permitted, unless it is specifically listed as denied
- Start off by allowing outside users access to all internal services, and then mark as blocked those few that are known to be unsafe
- Example: if tomorrow there's a new Slammer II worm, which spreads by exploiting a SQL server vulnerability, we revise our security policy to deny outsiders access to all our SQL servers

Default-Deny

- Default is every network service is denied, unless specifically listed as allowed
- Start with a list of few known servers that need to be externally accessible (and judged to be reasonably safe)
  - External users implicitly denied access to services not the list
  - Wait for complaints…
- User complains that their server isn't externally accessible (e.g., dept's FTP server)
  - We check if they're running a reasonably safe and properly configured FTP server and (if so) add them to the "allow" list

Administrivia

- Midterm #1 in-class on Monday 10/9
  - Two rooms (details Wednesday)
  - Review session in-class on Wednesday
- Moving to new office in RadLab
- Regular office hours this week
  - Mo/Tu 3-4pm 675 Soda
  - No office hours next Monday

Default-Allow versus Default-Deny

- Which policy does Berkeley use?
- Default-allow policy seems more convenient
  - Functional perspective: Everything stays working
  - Security perspective: default-allow is seriously flawed
- What's the problem?
- Default-allow fails open – make any mistake (i.e., forget to add vulnerable svc to "deny" list), result may be security failure
  - In contrast, default-deny fails closed – make a mistake (i.e., safe service mistakenly left off "allow" list), result is just loss of access

Large-Scale Operation

- Which is more likely, errors of omission or errors of commission?
- Thousands of potential services
  - Allow/deny lists have only a few dozen
  - Many more chances to inadvertently omit than add a service to a list…
- Errors of omission much more dangerous in a default-allow policy than in a default-deny policy
  - Cost of security failure is high, so default-deny is much safer

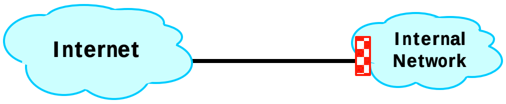
Another Default-Deny Advantage

- May never notice fail-open failures
  - Successful attackers unlikely to notify you
  - Security breaches may go unnoticed for a long time – puts you in an arms race
    - » More hackers than defenders makes this losing proposition...hacker need only win once
- In contrast, fail-closed failures likely to be noticed (user complaints)
- Almost all good firewalls use default-deny
  - Security policy specifies list of "allowed services", and all other services forbidden
  - Risk assessment/cost-benefit analysis applied to every service on allowed list

How to Identify Network Services?

- A TCP service is specified by machine's IP address and TCP port number on it
  - Web server `www.cs.berkeley.edu` (currently) at `169.229.60.105`, port `80`
  - Mail service at `169.229.60.93`, port `25`
  - UDP services similarly identified
- Identify each svc with triplet ***(m,r,p)***:
  - ***m*** is machine's IP addr (`A.B.C.D/[MASK]`)
  - ***r*** is a TCP/UDP protocol identifier
  - ***p*** is the port number
  - Example: official web servers on subnet `1.2.3.x` -> add(`1.2.3.0/24`, `TCP`, `80`) to allowed list

Enforcement: Packet Filters



- Enforce security policy at network chokepoint
  - Add a firewall that blocks any connections denied by security policy
  - Central chokepoint uses single place to easily enforce a security policy on 1,000's of machines
    - » Similar to airport security – few entrances

Packet Filters

- Simplest kind of firewall is a *packet filter*
  - Router with list of access control rules
  - Router checks each received packet against security rules to decide to forward or drop it
  - Each rule specifies which packets it applies to based on a packet's header fields
    - » Specify source and destination IP addrs, port numbers, and protocol names, or wild cards
    - » Each rule also specifies an action for matching packets: ALLOW or DROP
    - » <ACTION> <PRTCL> <SRC:PT> -> <DEST:PT>
  - List of rules is examined one-by-one
    - » First matching rule determines how packet will be handled

Example Ruleset

- What does this ruleset do?
  - drop tcp *:* -> *:23
  - allow * *:* -> *:*
- Answer:
  - Blocks all TCP pkts destined to port 23 (telnet)
    - » Telnet uses cleartext passwords!
  - Forwards all other traffic
- Problems?
- No notion of a connection, or of inbound vs outbound connections
  - Drops outbound telnet connections from inside users
  - This is a default-allow policy!!

Another Example

- Want to allow:
  - Inbound mail connections to our mail server (1.2.3.4:25)
  - All outbound connections
  - Nothing else
  - Consider this ruleset:
    - » allow tcp *:* -> 1.2.3.4:25
    - » allow tcp {*int_hosts*}:* -> *:*
    - » drop * *:* -> *:*
- This policy doesn't work…
  - TCP connections are bidirectional
  - 3-way handshake: send SYN, receive SYN|ACK, send ACK, send DATA w/ACK bit

Problem: Outbound Connections Fail

- Inside host opens TCP connection to port 80 on external machine:
  - Initial SYN packet passed through by rule 2
  - SYN|ACK packet coming back is dropped
    - » Fails rule 1 (not destined for port 25)
    - » Fails rule 2 (source not inside host)
    - » Matches rule 3 -> DROP
- Distinguish between 2 kinds of inbound pkts
  - Allow inbound packets associated with an outbound connection to pass
  - Restrict inbound packets associated with an inbound connection

Inbound versus Outbound Connections

- Key idea: use a feature of TCP!
  - ACK bit set on all packets except first one
  - Recipients discard any TCP packet with ACK bit set, if packet is not associated with an existing TCP connection
- Solution ruleset?
  - allow tcp *:* -> 1.2.3.4:25
  - allow tcp {*int_hosts*}:* -> *:*
  - allow tcp *:* -> {*int_hosts*}:* (if ACK bit set)
  - drop * *:* -> *:*
  - Rules 1 and 3 allow inbound connections to port 25 on machine 1.2.3.4
  - Rules 2 and 3 allow outbound connections to any port

Example Using This Ruleset

- Outside attacker trying to exploit finger service (TCP port 79) vulnerability
  - Tries to open an inbound TCP connection to our finger server
- Attempt #1:Sends SYN pkt to int. machine
  - Pkt doesn't have ACK bit set, so fw rule drops it
- Attempt #2: Sends SYN|ACK pkt to internal machine
  - FW permits pkt, then dropped by TCP stack (ACK bit set but isn't part of existing connection)
- We can specify policies restricting inbound connections arbitrarily

IP Spoofing: Another Security Hole

- IP protocol doesn't prevent attacker from sending pkt with wrong (*spoofed*) src addr
  - Most routers ignore src addrs
- Suppose `1.2.3.7` is an internal host
  - Attacker sends spoofed TCP SYN packet
    - » Src addr `1.2.3.7`, dest addr target internal machine, dest port 79 – rule 2 allows
  - Target replies with SYN|ACK pkt to `1.2.3.7` and waits for ACK (to finish 3-way handshake)
  - Attacker sends spoofed TCP ACK packet
  - Attacker then sends data packet

Attack Analysis

- Attack allows connections to internal hosts
  - Violates of our security policy
  - Allows attacker to exploit any security holes
    - » Ex: finger service vulnerability
  - Caveat:
    - » Attacker has to "guess" Initial Sequence Number set by target in SYN|ACK packet sent to `1.2.3.7` (many ways to guess…)
- Modified Solution
  - Packet filter marks each packet with incoming interface ID, and rules match IDs
    - » Recall: Router has 2+ interfaces, forwards packets from one to another

New Solution

- New ruleset
  - Int. interface: `in`, ext. interface: `out`

```
– allow tcp *:*/out -> 1.2.3.4:25/in
– allow tcp *:*/in  -> *:*/out
– allow tcp *:*/out -> *:*/in  (if ACK bit set)
– drop    *  *:*     -> *:*
```

  - Allows inbound packets only if destined to `1.2.3.4:25` (rule 1), or, if ACK bit set (rule 3)
  - Drops all other inbound packets
- Clean solution: defeats IP spoofing threat
  - Simplifies ruleset admin (no hardcode internal hosts list)

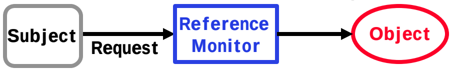
Other Kinds of Firewalls

- Packet filters are quite crude firewalls
  - Network level using TCP, UDP, and IP headers
- Alternative: examine data field contents
  - Application-layer firewalls (application firewalls)
    - » Can enforce more restrictive security policies and transform data on the fly
- For more information on firewalls, read:
  - Cheswick, Bellovin, and Rubin: *Firewalls and Internet Security: Repelling the Wily Hacker*.
- Packet filtering sw available for many OS's:
  - Linux `iptables`, OpenBSD/FreeBSD `PF`, and Windows XP SP2 firewall

BREAK

Principles

- Firewalls embody useful principles that are applicable elsewhere in computer security
  - Optimized for enforcing particular kind of access control policy
  - Chokepoint notion is crucial: makes enforcement possible
- One enforcement mechanism: *reference monitor*
  - Examines every request to access any controlled resource (an object) and determines whether to allow request



Reference Monitor Security Properties

- *Always invoked*
  - *Complete mediation* property: all security-relevant operations must be mediated by RM
  - RM should be invoked on every operation controlled by access control policy
- *Tamper-resistant*
  - Maintain RM integrity (no code/state tampering)
- *Verifiable*
  - Can verify RM correctness (correctly enforces desired access control policy)
    - » Requires extremely simple RM
    - » Can't verify correctness for systems with any appreciable degree of complexity

Firewalls as a RM Instance?

- Always invoked
  - Place Packet Filter on chokepoint link for all internal-external communications
  - Packets are only forwarded across link if packet filter inspects and forwards them

Is PF Really a Chokepoint?

- Thought exercise
  - Paint internal machine and every outgoing wire, red
  - Paint machine connected to red network as red (except for packet filter machine!)
  - Recurse until no more painting to be done
- Check which machines are painted red?
  - PF is the only non-red machine reachable from internal net
  - All red machines are on internal network
  - No external machines are painted red
    - » Red things = resources to be protected
    - » Non-red things = resources we don't have to trust

Potential Problems?

- What if a user hooks up an unsecured wireless access point to their internal machine?
- Anyone who drives by with wireless-enabled laptop can gain access to internal network ("gets painted red")
  - Bypasses packet filter!
- Means that to use a firewall safely, we'd better be sure that we've covered all links between internal and external networks with firewalls
  - Set of links known as the *security perimeter*

RM Property: *Tamper-Resistant*

- Haven't discussed how to make packet filters attack resistant
  - Need to harden as much as possible (single point of failure)
- Choices
  - Desired functionality is relatively simple
  - Could run a non-standard OS without any user-level programs, or network services
- Must also protect packet filter's physical security

RM Property: *Verifiable*

- Current practice:
  - Packet filter software too complex for feasible systematic verification…
- Result:
  - Bugs that allowed attackers to defeat intended security policy by sending unexpected packets that packet filter doesn't handle quite the way it should
- Reference Monitor Summary
  - Notion of a RM recurs over and over, so worth memorizing the three requirements for a secure Reference Monitor

Another Useful Firewall Principle

- Orthogonal Security
  - Transparent security mechanism can more easily be deployed to protect legacy systems
    - » Transparent: A RM that filters requests, dropping disallowed requests but passing allowed requests unchanged
- Can be cascaded in series or in parallel
  - Series: request allowed only if all RMs allow it
    - » Any attack must defeat all the monitors
  - Parallel: allows separation of concerns
    - » One RM handles all TCP traffic, another RM handles all UDP traffic
    - » Unclear what benefit this approach provides

Experience with Firewalls

- Firewalls have been very widely used
  - Success story: R&D to industry tech transfer
    - » First paper published at 1990 conference
    - » Checkpoint firewall vendor founded in 1993, largest fw market share, >$500M/yr revenue
- Why do They Work Well?
  - *Central control – easy administration and update*
    - » Single pt of ctl: update fw to change security policies
    - » Can often block new worms by fw rule changes
  - *Easy to deploy – transparent to end users*
    - » Easy incremental/total deployment to protect 1,000's
  - *Address an important problem*
    - » Security vulnerabilities in network svcs are rampant
    - » Easier to use firewall than to clean up code…

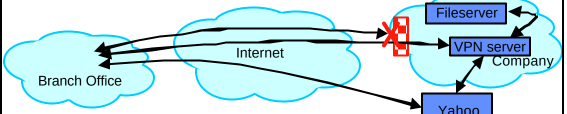
Firewall Failures And Disadvantages?

- *Functionality loss – less connectivity, less risk*
  - May reduce network's usefulness
  - Some applications don't work with firewalls
    - » Two peer-to-peer users behind diff. firewalls
- *The malicious insider problem*
  - Assume insiders are absolutely trusted
    - » Malicious insider (or anyone gaining ctl of an internal machine) can wreak havoc
    - » Defeats physical and network security
  - Firewalls establish *security perimeter*
    - » Bill Cheswick: "crunchy outer coating, with a creamy center"
    - » Threat from travelers with laptop…

Other FW Failures And Disadvantages?

- *"Malicious" applications*
  - Previous properties combine in a very nasty way: app protocol blocked by users' firewalls
- What to do?
  - Tunnel app's connections over HTTP or SMTP
  - Web is killer app, so most firewalls allow it
  - Now firewall can't distinguish real/app traffic
  - Insiders trusted -> their apps trusted -> firewall can't protect against malicious apps
  - More and more traffic goes over port 25/80/…
    - » FWs have less visibility into traffic
    - » FWs become less effective

Secure External Access to Inside Machines



- Often need to provide secure remote access to a network protected by a firewall
  - Remote access, telecommuting, branch offices, …
- Create secure channel (Virtual Private Network) to tunnel traffic from outside host/network to inside network
  - Provides Authentication, Confidentiality, Integrity

Virtual Private Network

- Implementation
  - Virtual network driver forwards traffic over IPSEC or TLS/SSL secure channel
  - Open source clients (OpenVPN)
  - High-performance commercial hardware
- Try it yourself!
  - http://www.net.berkeley.edu/vpn/
- VPN introduces perimeter security issues
  - Compromise remote machine and become trusted insider

VPN Perimeter Security Issues

- Davis-Besse plant used a firewall
- Slammer worm penetrated unsecured network of a Davis-Besse contractor
- Squirms through a VPN into D-B's internal network
- Disables two safety monitoring systems for five to six hours
- Plant was already offline
- Analog systems still online

Ohio's Davis-Besse Nuclear Power Plant (Jan 2003)

SecurityFocus 08/19/03

Summary

- Firewalls provide an easy method for reducing the number of exposed services
- Question of default policy: allow or deny?
  - Allow is transparent, but vulnerable to errors
  - Default-Deny is non-transparent, but safer
- Developing correct rules is hard
  - Need to worry about inbound vs. outbound, established vs. new connections
- Firewalls are an example of Reference Monitor principles
- VPNs make life easy and hard…