CIS551: Computer and Network Security

Jonathan M. Smith

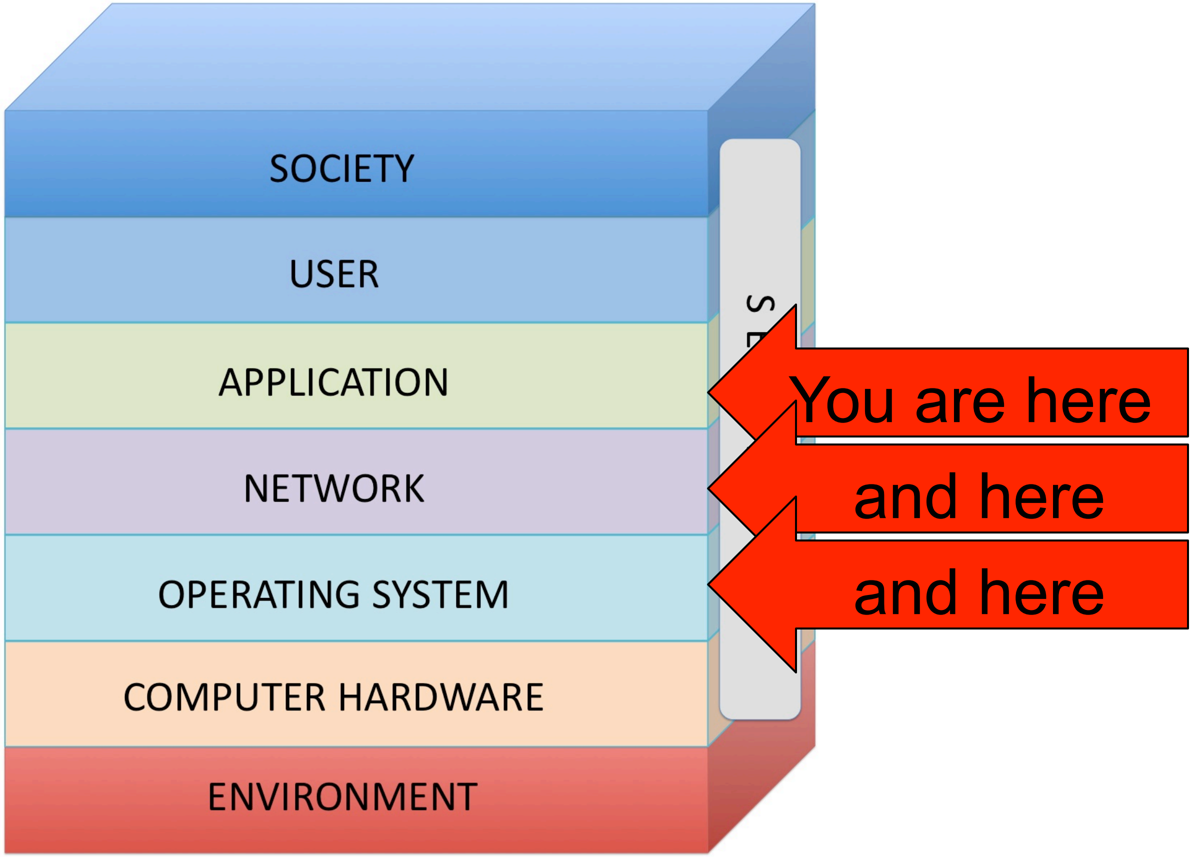
jms@cis.upenn.edu

02/17/2014

CIS551 Topics

- Computer Security
  - Software/Languages, Computer Arch.
  - Access Control, Operating Systems
  - Threats: Vulnerabilities, Viruses
- Computer Networks
  - Physical layers, Internet, WWW, Applications
  - Cryptography in several forms
  - Threats: Confidentiality, Integrity, Availability
- Systems Viewpoint
  - Users, social engineering, insider threats

Sincoskie NIS model



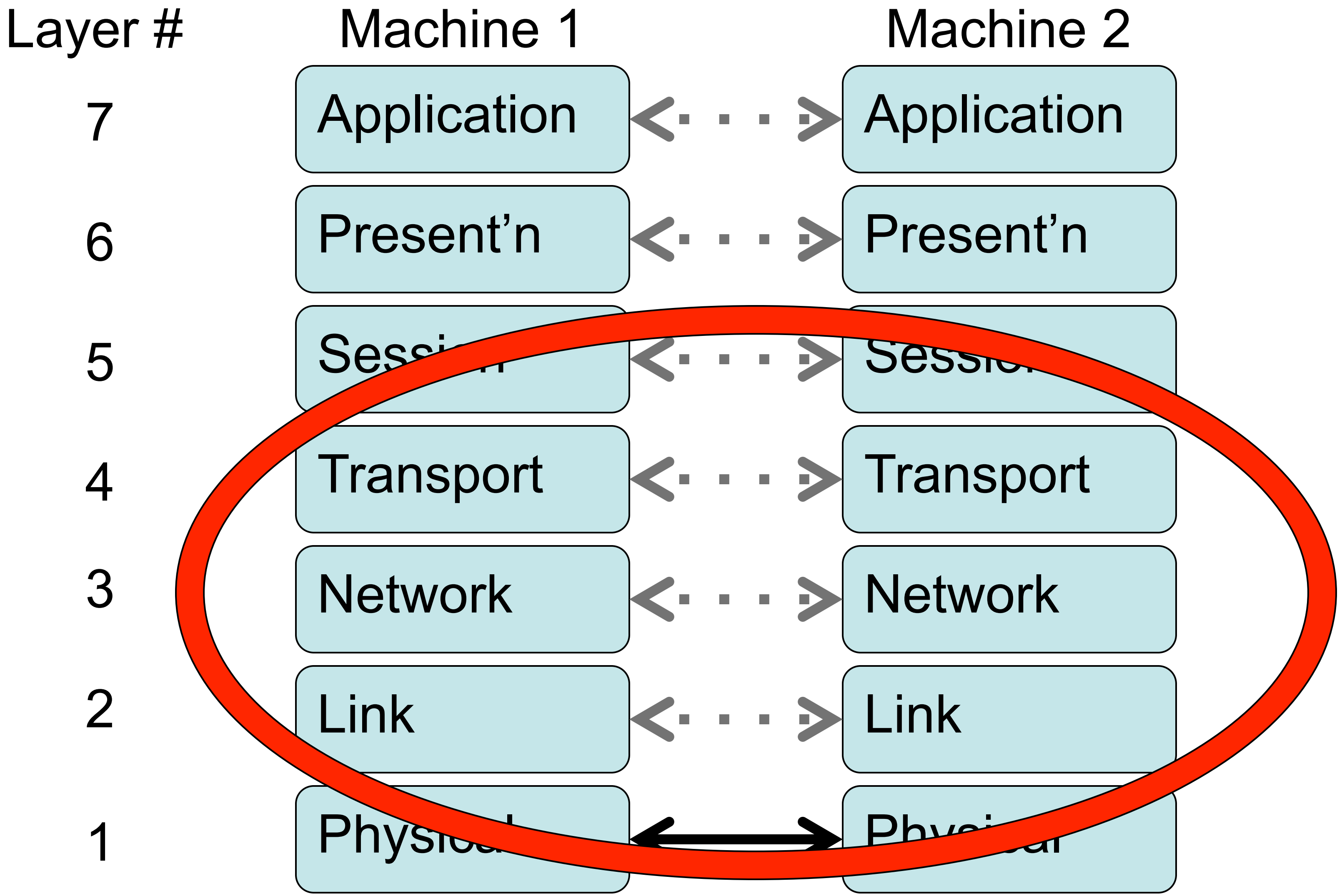
You are here

and here

and here

W.D. Sincoskie, *et al.* “Layer Dissonance and Closure in Networked Information Security” (white paper)

7-layer OSI network model



Layer #	Machine 1	Machine 2
7	Application	Application
6	Present’n	Present’n
5	Session	Session
4	Transport	Transport
3	Network	Network
2	Link	Link
1	Physical	Physical

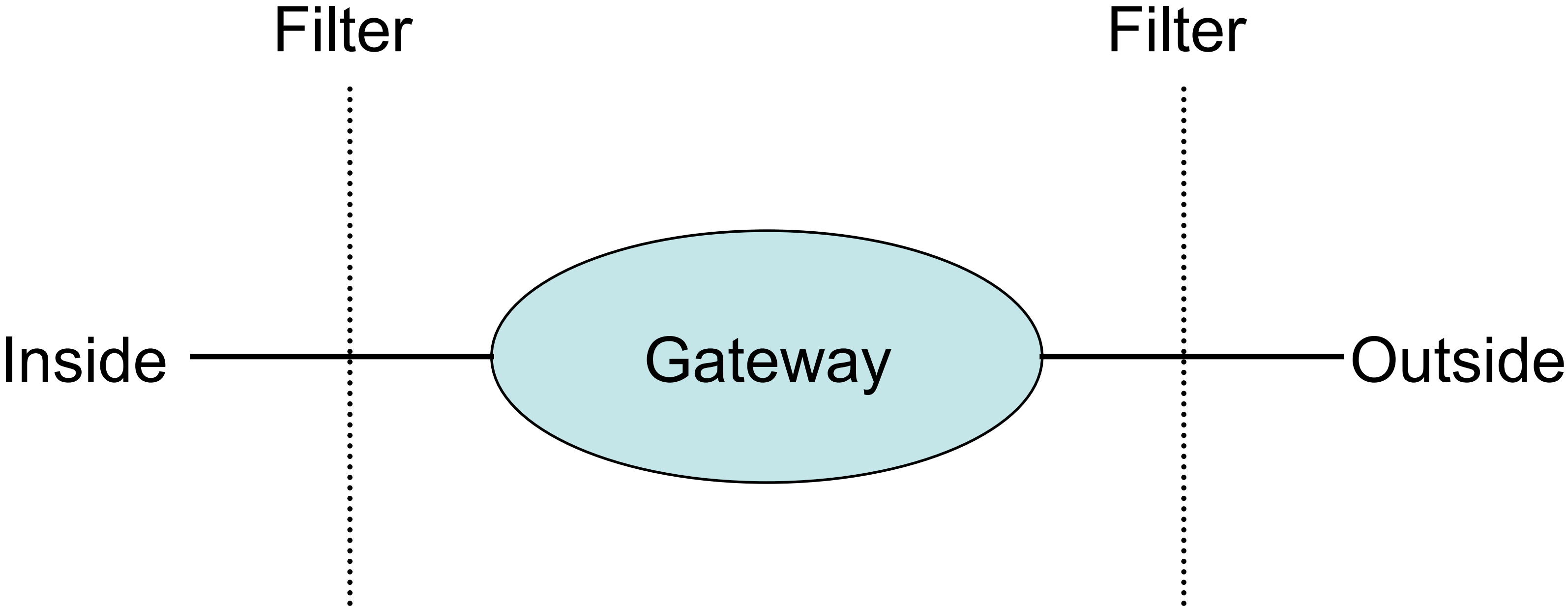
Kinds of Firewalls

- Personal firewalls
  - Run at the end hosts
  - e.g., Norton, Windows, etc.
  - Benefit: has more application/user specific information

- Filter Based
  - Operates by filtering based on packet headers

- Proxy based
  - Operates at the level of the application
  - e.g., HTTP web proxy

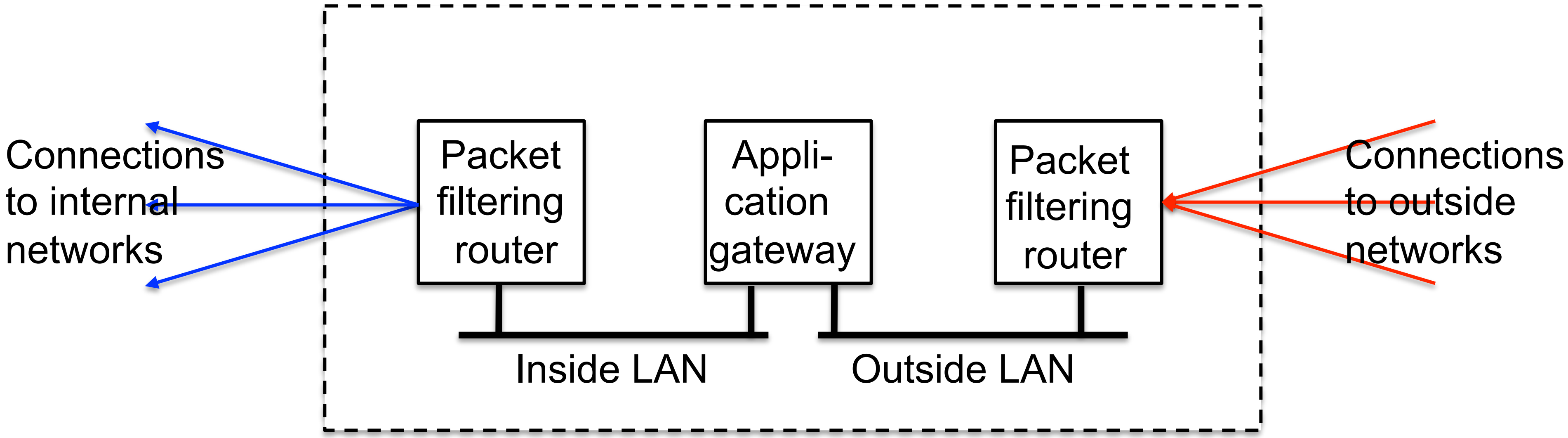
Firewalls



- Filters protect against “bad” packets.
- Protect services offered internally from outside access.
- Provide outside services to hosts located inside.

Firewalls

- Filtering – what to inspect?
  - Packet-filtering gateway (inspects headers)
  - Application-level gateway (inspects contents)



Adapted from Fig. 9-28 in "Distributed Systems", by Tanenbaum and Van Steen

Filtering Firewalls

- Filtering can take advantage of the following information from network and transport layer headers:
  - Source
  - Destination
  - Source Port
  - Destination Port
  - Flags (e.g. ACK)
  - Protocol type (e.g. UDP vs. TCP)

- Some firewalls keep state about open TCP connections
  - Allows conditional filtering rules of the form “if internal machine has established the TCP connection, permit inbound reply packets”

Filter Example

Action	ourhost	port	theirhost	port	comment
block	*	*	BAD	*	untrusted host
allow	GW	25	*	*	allow our SMTP port

Apply rules from top to bottom with assumed *default* entry:

Action	ourhost	port	theirhost	port	comment
block	*	*	*	*	default

Bad entry intended to allow connections to SMTP from inside:

Action	ourhost	port	theirhost	port	comment
allow	*	*	*	25	connect to their SMTP

This allows all connections from port 25, but an outside machine can run *anything* on its port 25!

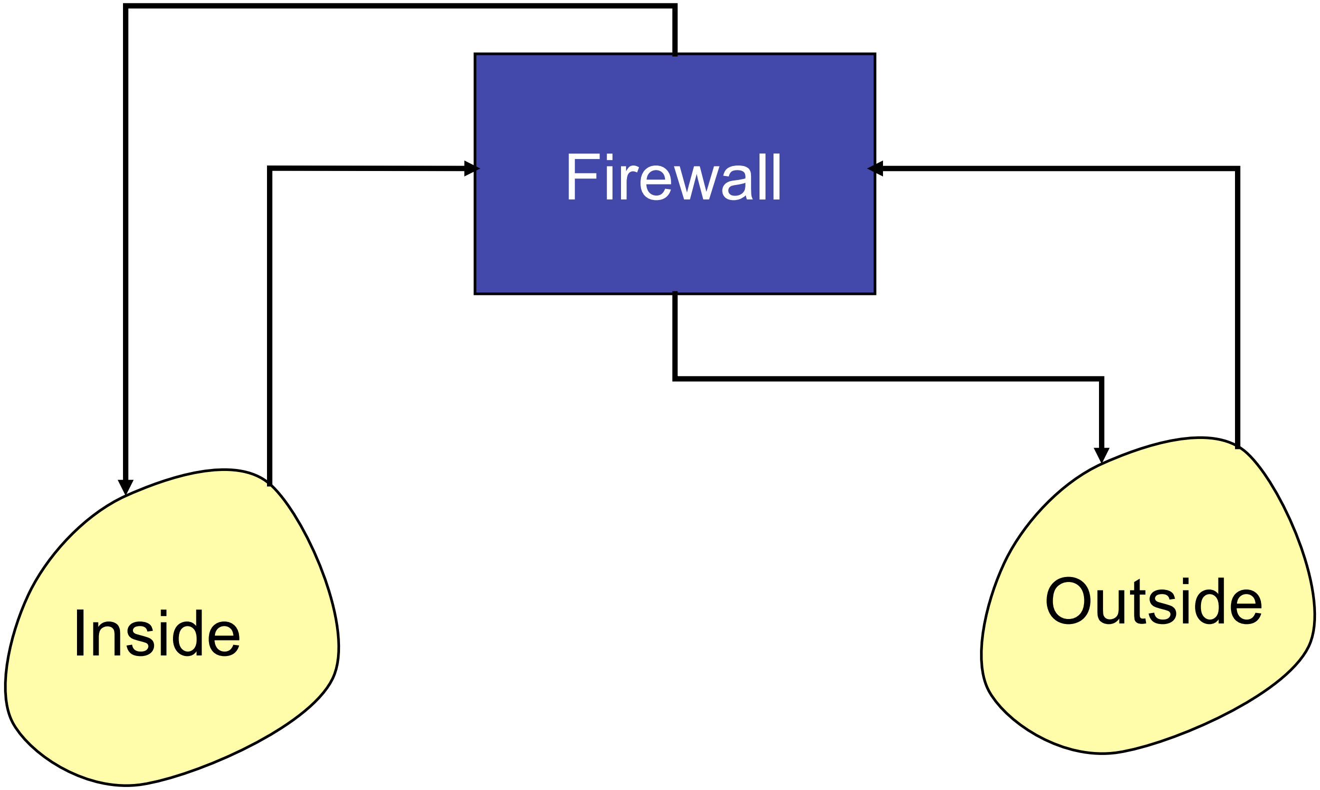
Filter Example Continued

Permit *outgoing* calls to port 25.

Action	src	port	dest	port	flags	comment
allow	123.45.6.*	*	*	25	*	their SMTP
allow	*	25	*	*	ACK	their replies

This filter doesn't protect against IP address spoofing. The bad hosts can "pretend" to be one of the hosts with addresses 123.45.6.* .

When to Filter?



On Input or Output?

- Filtering on *output* can be more efficient since it can be combined with table lookup of the route.
- However, some information is lost at the output stage
  - e.g. the physical input port on which the packet arrived.
  - Can be useful information to prevent address spoofing.
- Filtering on *input* can protect the router itself.

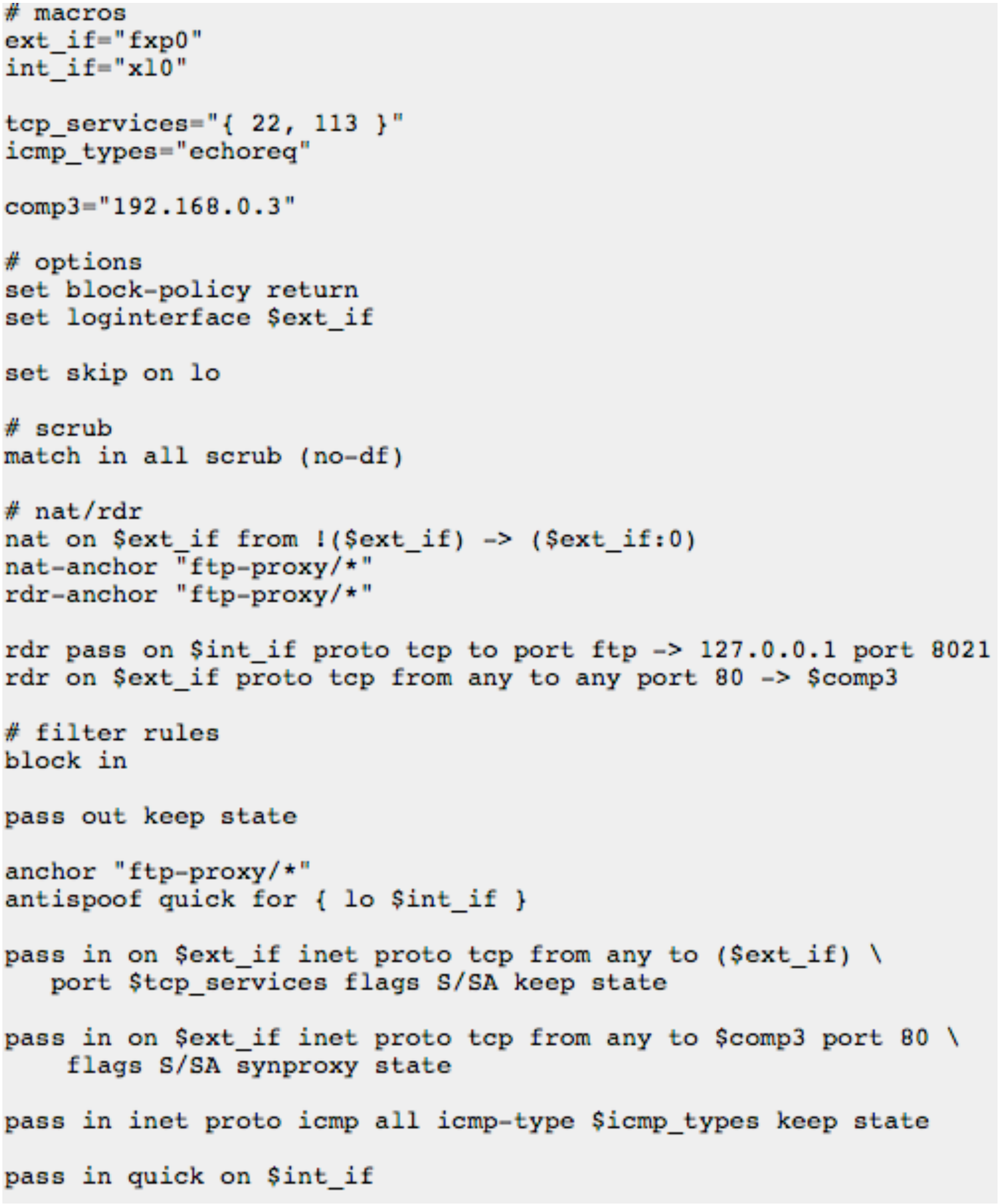
Principles for Firewall Configuration

- General principle: *Filter as early as possible*
- Least Privilege:
  - Turn off everything that is unnecessary (e.g. Web Servers should disable SMTP port 25)
- Failsafe Defaults:
  - By default should reject
  - (Note that this could cause usability problems…)
- Egress Filtering:
  - Filter outgoing packets too!
  - You know the valid IP addresses for machines internal to the network, so drop those that aren’t valid.
  - This can help prevent DoS attacks in the Internet.

Example “real” firewall config script

```
############
# FreeBSD Firewall configuration.
# Single-machine custom firewall setup. Protects somewhat
# against the outside world.
############

# Set this to your ip address.
ip="192.100.66.1"
setup_loopback

# Allow anything outbound from this address.
${fwcmd} add allow all from ${ip} to any out

# Deny anything outbound from other addresses.
${fwcmd} add deny log all from any to any out

# Allow inbound ftp, ssh, email, tcp-dns, http, https, imap, imaps,
# pop3, pop3s.
${fwcmd} add allow tcp from any to ${ip} 21 setup
${fwcmd} add allow tcp from any to ${ip} 22 setup
${fwcmd} add allow tcp from any to ${ip} 25 setup
${fwcmd} add allow tcp from any to ${ip} 53 setup
${fwcmd} add allow tcp from any to ${ip} 80 setup
${fwcmd} add allow tcp from any to ${ip} 443 setup
…
```

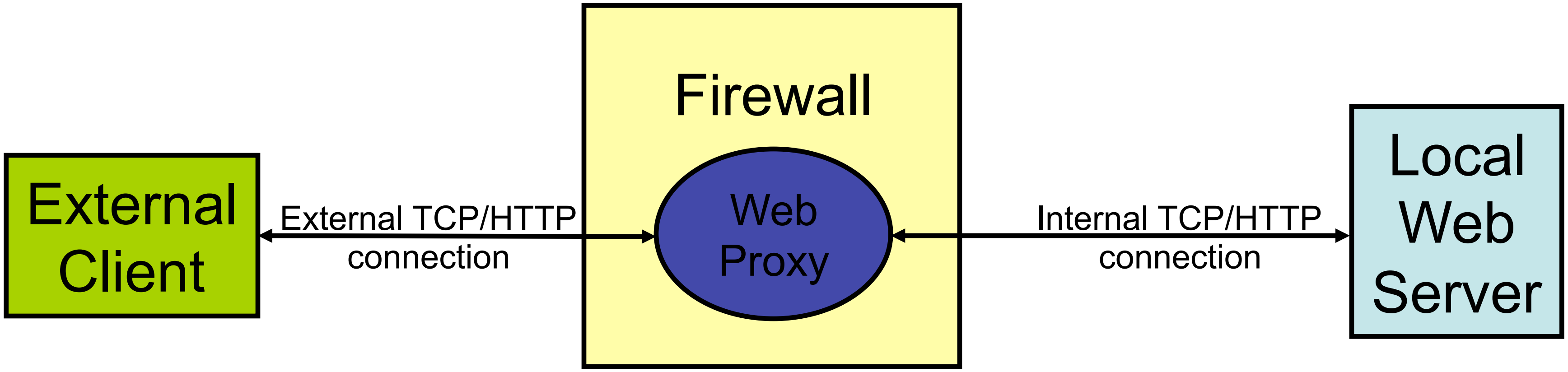
Example real packet filter rules



```
# macros
ext_if="fxp0"
int_if="xl0"

tcp_services="{ 22, 113 }"
icmp_types="echoreq"

comp3="192.168.0.3"

# options
set block-policy return
set loginterface $ext_if

set skip on lo

# scrub
match in all scrub (no-df)

# nat/rdr
nat on $ext_if from !($ext_if) -> ($ext_if:0)
nat-anchor "ftp-proxy/*"
rdr-anchor "ftp-proxy/*"

rdr pass on $int_if proto tcp to port ftp -> 127.0.0.1 port 8021
rdr on $ext_if proto tcp from any to any port 80 -> $comp3

# filter rules
block in

pass out keep state

anchor "ftp-proxy/*"
antispoof quick for { lo $int_if }

pass in on $ext_if inet proto tcp from any to ($ext_if) \
   port $tcp_services flags S/SA keep state

pass in on $ext_if inet proto tcp from any to $comp3 port 80 \
    flags S/SA synproxy state

pass in inet proto icmp all icmp-type $icmp_types keep state

pass in quick on $int_if
```

Example “pf” rules from openbsd.org website

Proxy-based Firewalls



- Proxy acts like *both* a client and a server.
- Able to filter using application-level info
  - For example, permit some URLs to be visible outside and prevent others from being visible.
- Proxies can provide other services too
  - Caching, load balancing, etc.
  - FTP and Telnet proxies are common too

Benefits of Firewalls

- Increased security for internal hosts.
- Reduced amount of effort required to counter break ins.
- Possible added convenience of operation within firewall (with some risk).
- Reduced legal and other costs associated with hacker activities.

Drawbacks of Firewalls

- Costs:
  - HW purchase and maintenance
  - SW development or purchase, and update costs
  - Administrative setup and training, and ongoing administrative costs and trouble-shooting
  - Lost business/inconvenience from broken gateway
  - Loss of some services that an open connection would supply.
- False sense of security
  - Firewalls don’t protect against viruses, port 80 must be kept open, …

Snort

- Snort is a lightweight intrusion detection system:
  - Real-time traffic analysis
  - Packet logging (of IP networks)
- Rules based logging to perform content pattern matching to detect a variety of attacks and probes:
  - such as buffer overflows, stealth port scans, CGI attacks, SMB probes, etc.
- Example Rule:

```
alert tcp any any -> 192.168.1.0/24 143 (content:"|E8C0
FFFF FF|/bin/sh"; msg:"New IMAP Buffer Overflow
detected!";)
```

  - Generates an alert on all inbound traffic for port 143 with contents containing the specified attack signature.
- The Snort web site:
  - http://www.snort.org/docs/

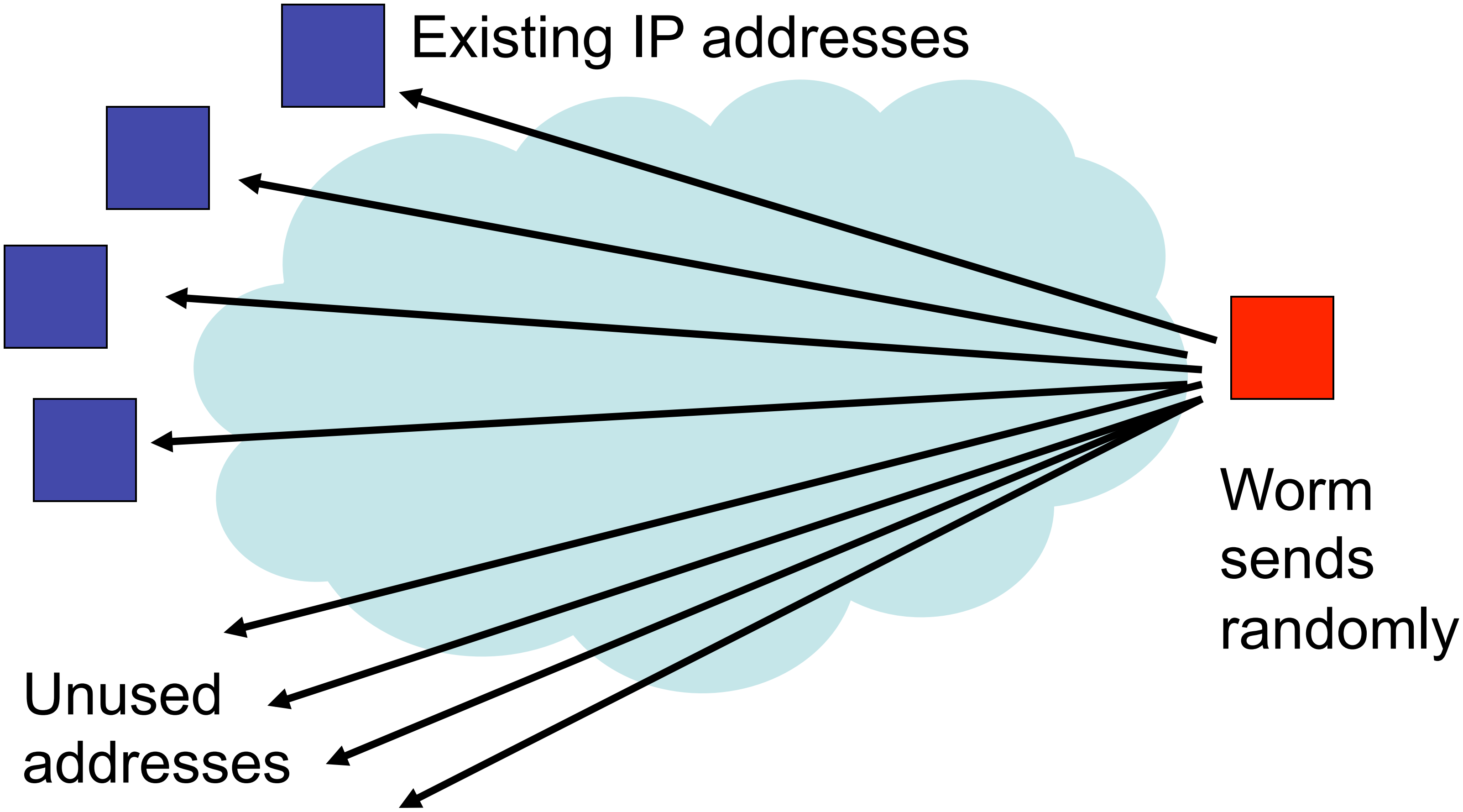
- Question: How do you come up with the filter rules?

Capturing packets

- pcap library; accessible with -lpcap
- pcap_create() – online capture
- pcap_open_offline() – saved data
- pcap_compile() – BPF compiler
- pcap_setfilter() – install compiled filter
- pcap_next() – get next packet
- Will do in-class demo

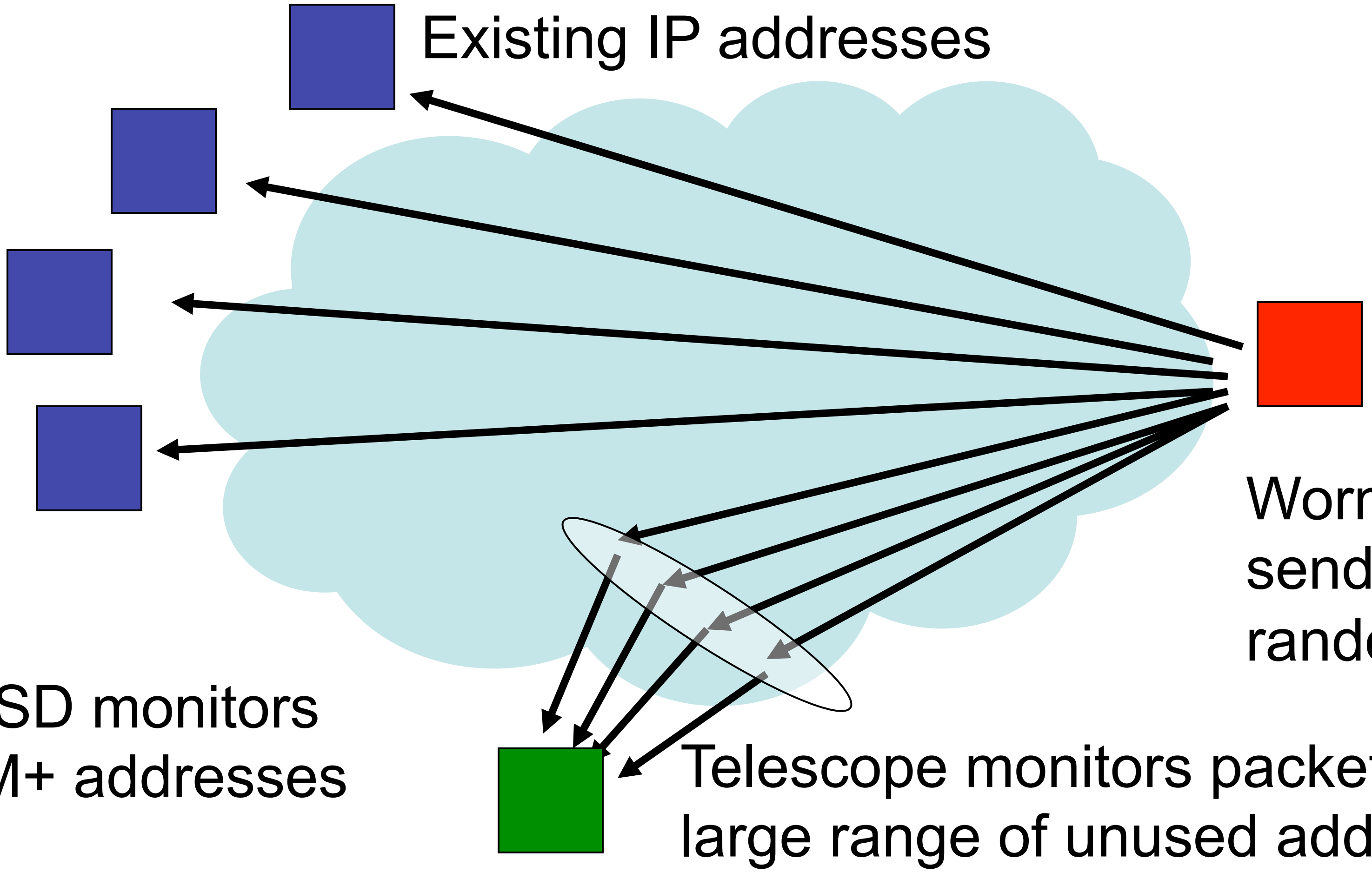
“Internet Telescopes”

- Can be used to detect large-scale, wide-spread attacks on the internet.



“Internet Telescopes”

- Can be used to detect large-scale, wide-spread attacks on the internet.



Existing IP addresses

Worm sends randomly

UCSD monitors 17M+ addresses

Telescope monitors packets for large range of unused addresses

Challenge: *Polymorphic* Viruses/Worms

- Virus/worm writers know that signatures are used to detect such malicious code.
- Polymorphic viruses *mutate* themselves during replication to prevent detection
  - Virus should be capable of generating many different descendants
  - Simply embedding random numbers into virus code is not enough

Strategies for Polymorphic Viruses

- Change data:
  - Use different subject lines in e-mail

- Encrypt most of the virus with a random key
  - Virus first decrypts main body using random key
  - Jumps to the code it decrypted
  - When replicating, generate a new key and encrypt the main part of the replica

- Still possible to detect decryption portion of the virus using virus signatures
  - This part of the code remains unchanged
  - Worm writer could use a standard self-decompressing executable format (like ZIP executables) to cause confusion (many false positives)

Advanced Evasion Techniques

- “Randomly” modify the *code* of the virus/worm by:
  - Inserting no-op instructions: subtract 0, move value to itself
  - Reordering independent instructions
  - Using different variable/register names
  - Using equivalent instruction sequences:

    y = x + x   vs.   y = 2 * x
  - These viruses are sometimes called "metamorphic" viruses in the literature.

- There exist libraries that, when linked against an appropriate executable, automatically turn it into a metamorphic program.

- Sometimes vulnerable software itself offers opportunities for hiding bad code.
  - Example: ssh or SSL vulnerabilities may permit worm to propagate over encrypted channels, making content filtering impossible.
  - If IPSEC becomes popular, similar problems may arise with it.

Other Evasion Techniques

- Observation: worms don't need to scan randomly
  - They won't be caught by internet telescopes

- *Meta-server* worm: ask server for hosts to infect (e.g., Google for “**powered by php**”)

- *Topological* worm: fuel the spread with local information from infected hosts (web server logs, email address books, config files, SSH “known hosts”)
  - No scanning signature; with rich inter-connection topology, potentially very fast.
- Propagate slowly: "trickle" attacks
  - Also a very subtle form of denial of service attacks

Broader View of Defenses

- Prevention -- *make the monoculture hardier*
  - Get the code right in the first place …
    - … or figure out what’s wrong with it and fix it
  - Lots of active research (static & dynamic methods)
  - Security reviews now taken seriously by industry
    - E.g., ~$200M just to *review* Windows Server 2003
  - But very expensive
  - And very large “installed base” problem

- Prevention -- *diversify the monoculture*
  - Via exploiting existing heterogeneity (Windows, MacOS, OpenBSD)
  - Via creating artificial heterogeneity (stack randomization, etc.)

Broader View of Defenses, con' t

- Prevention -- *keep vulnerabilities inaccessible*
  - Cisco' s *Network Admission Control*
    - Examine hosts that try to connect, block if vulnerable
  - Microsoft' s *Shield*
    - Shim-layer blocks network traffic that fits known *vulnerability* (rather than known *exploit*)

Detecting Attacks

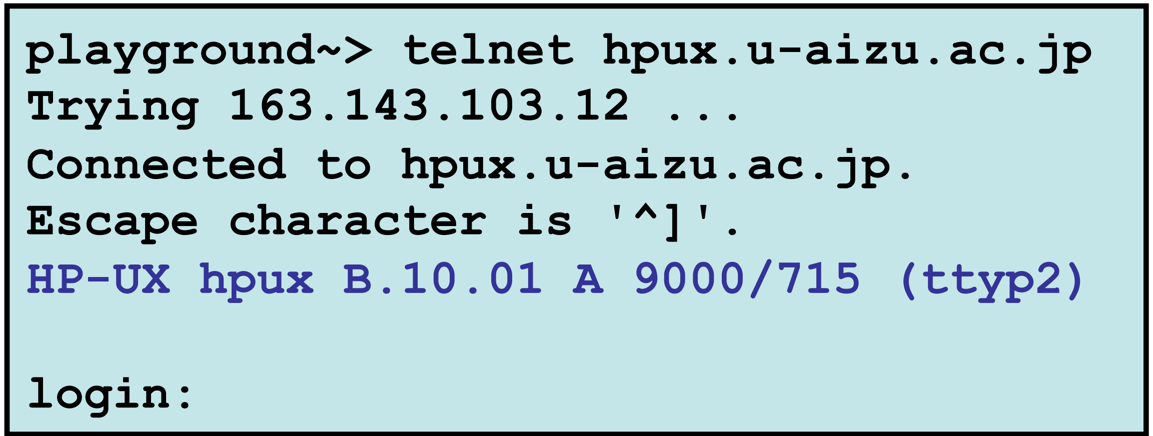
- Attacks (against computer systems) usually consist of several stages:
  - Finding software vulnerabilities
  - Exploiting them
  - Hiding/cleaning up the exploit
- Attackers care about finding vulnerabilities:
  - What machines are available?
  - What OS / version / patch level are the machines running?
  - What additional software is running?
  - What is the network topology?
- Attackers care about not getting caught:
  - How detectable will the attack be?
  - How can the attacker cover her tracks?
- Programs can automate the process of finding/exploiting vulnerabilities.
  - Same tools that sys. admins. use to audit their systems…
  - A worm is just an automatic vulnerability finder/exploiter…

Attacker Reconnaissance

- Network Scanning
  - Existence of machines at IP addresses
  - Attempt to determine network topology
  - ping, traceroute
- Port scanners
  - Try to detect what processes are running on which ports, which ports are open to connections.
  - Typical machine on the Internet gets 10-20 port scans per day!
  - Can be used to find hit lists for flash ("Warhol"!) worms such as Slammer/Sapphire
- Web services
  - Use a browser to search for CGI scripts, Javascript, etc.

Determining OS information

- Gives a lot of information that can help an attacker carry out exploits
  - Exact version of OS code can be correlated with vulnerability databases
- Sadly, often simple to obtain this information:
  - Just try telnet! (this example no longer works):



```
playground~> telnet hpux.u-aizu.ac.jp
Trying 163.143.103.12 ...
Connected to hpux.u-aizu.ac.jp.
Escape character is '^]'.
HP-UX hpux B.10.01 A 9000/715 (ttyp2)

login:
```

Determining OS

- Or FTP (tested 3/4/10, 8AM):

```
$ ftp ftp.gftp.netscape.com
Connected to ftp.gftp.netscape.com.
220-d6
220
Name (ftp.gftp.netscape.com:jms): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> system
215 UNIX Type: L8
ftp> exit
221 Goodbye.
$
```

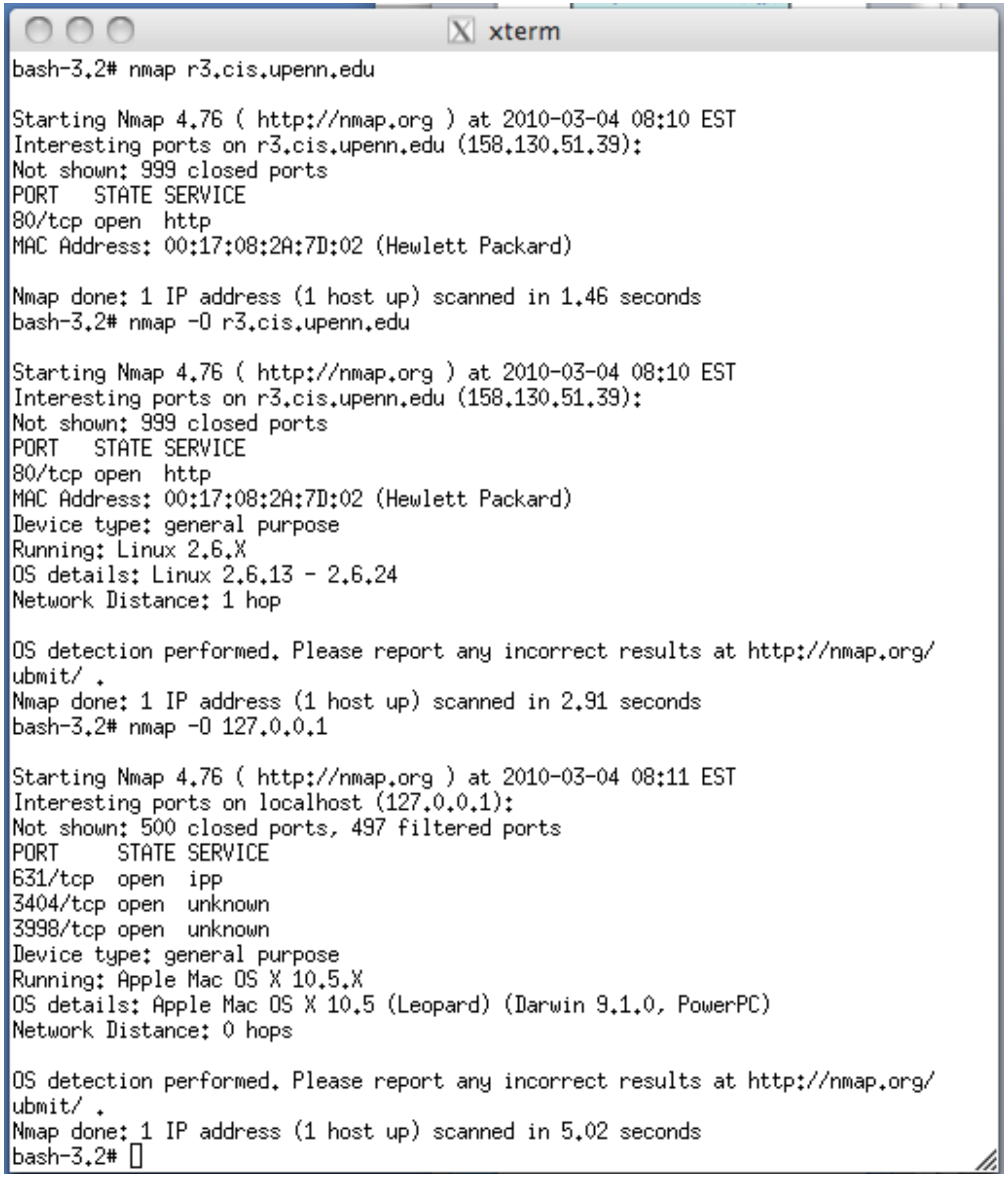
Determining OS

- Exploit different implementations of protocols
  - Different OS’s have different behavior in some cases
- Consider TCP protocol, there are many flags and options, and some unspecified behavior
  - Reply to bogus FIN request for TCP port
    (should not reply, but some OS’s do)
  - Handling of invalid flags in TCP packets
    (some OS’s keep the invalid flags set in reply)
  - Initial values for RWS, pattern in random sequence numbers, etc.
  - Can narrow down the possible OS based on the combination of implementation features
- Tools can automate this process

Auditing: Remote audit tools

- Several utilities available to “attack” or gather information about services/daemons on a system.
  - SATAN (early 1990’s): “Security Administrator Tool for Analyzing Networks”
  - SAINT - Based on SATAN utility
  - SARA - Also based on SATAN
  - Nessus - Open source vulnerability scanner
    - http://www.nessus.org
  - Nmap
- Commercial:
  - ISS scanner
  - Cybercop

Nmap screenshot:



```
bash-3.2# nmap r3.cis.upenn.edu

Starting Nmap 4.76 ( http://nmap.org ) at 2010-03-04 08:10 EST
Interesting ports on r3.cis.upenn.edu (158.130.51.39):
Not shown: 999 closed ports
PORT   STATE SERVICE
80/tcp open  http
MAC Address: 00:17:08:2A:7D:02 (Hewlett Packard)

Nmap done: 1 IP address (1 host up) scanned in 1.46 seconds
bash-3.2# nmap -O r3.cis.upenn.edu

Starting Nmap 4.76 ( http://nmap.org ) at 2010-03-04 08:10 EST
Interesting ports on r3.cis.upenn.edu (158.130.51.39):
Not shown: 999 closed ports
PORT   STATE SERVICE
80/tcp open  http
MAC Address: 00:17:08:2A:7D:02 (Hewlett Packard)
Device type: general purpose
Running: Linux 2.6.X
OS details: Linux 2.6.13 - 2.6.24
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at http://nmap.org/
ubmit/ .
Nmap done: 1 IP address (1 host up) scanned in 2.91 seconds
bash-3.2# nmap -O 127.0.0.1

Starting Nmap 4.76 ( http://nmap.org ) at 2010-03-04 08:11 EST
Interesting ports on localhost (127.0.0.1):
Not shown: 500 closed ports, 497 filtered ports
PORT     STATE SERVICE
631/tcp  open  ipp
3404/tcp open  unknown
3998/tcp open  unknown
Device type: general purpose
Running: Apple Mac OS X 10.5.X
OS details: Apple Mac OS X 10.5 (Leopard) (Darwin 9.1.0, PowerPC)
Network Distance: 0 hops

OS detection performed. Please report any incorrect results at http://nmap.org/
ubmit/ .
Nmap done: 1 IP address (1 host up) scanned in 5.02 seconds
bash-3.2# 
```