CIS 551 / TCOM 401

Computer and Network Security

Spring 2009

Lecture 9

Announcements

- Plan for Today:
  - Firewalls
  - Return to content filtering: implementation and countermeasures
- Midterm 1: Next Tuesday
  - 2/17/2009
  - In class, short answer, multiple choice, analysis
- Project 2 will be available soon
  - Due: Friday, March 6th (right before Spring Break)

Kinds of Firewalls

- Personal firewalls
  - Run at the end hosts
  - e.g. Norton, Windows, etc.
  - Benefit: has more application/user specific information
- Network Address Translators
  - Rewrites packet address information
- Filter Based
  - Operates by filtering based on packet headers
- Proxy based
  - Operates at the level of the application
  - e.g. HTTP web proxy

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!

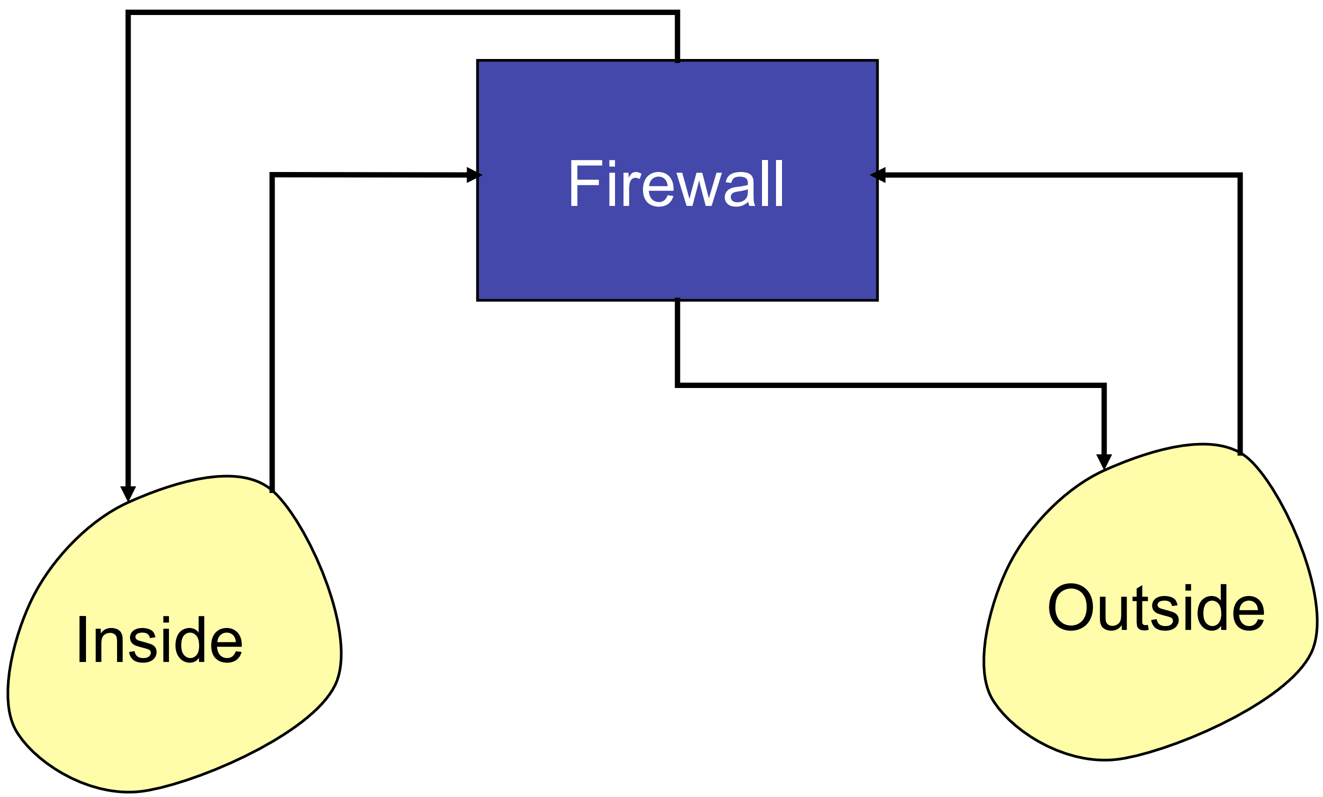
Filter Example Continued

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.

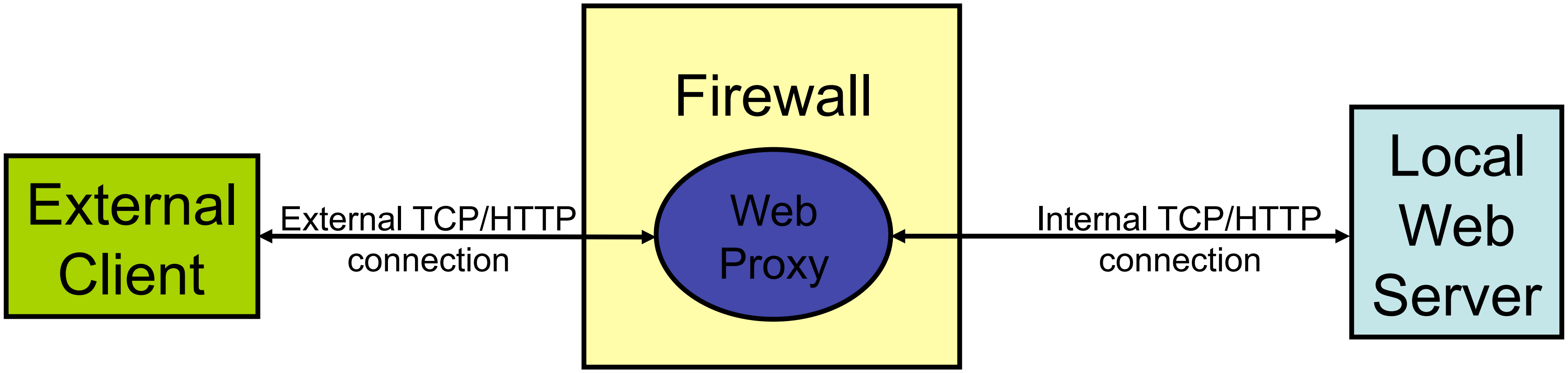
Principles for Firewall Configuration

- General principal: Filter as early as possible
- Least Privileges:
  - 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 firewall config script

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

# Set this to your ip address.
ip="192.100.666.1"
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
…
```

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:
  - Hardware purchase and maintenance
  - Software development or purchase, and update costs
  - Administrative setup and training, and ongoing administrative costs and trouble-shooting
  - Lost business or inconvenience from broken gateway
  - Loss of some services that an open connection would supply.
- False sense of security
  - Firewalls don't protect against viruses…

- Next: Content filtering, revisited

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:

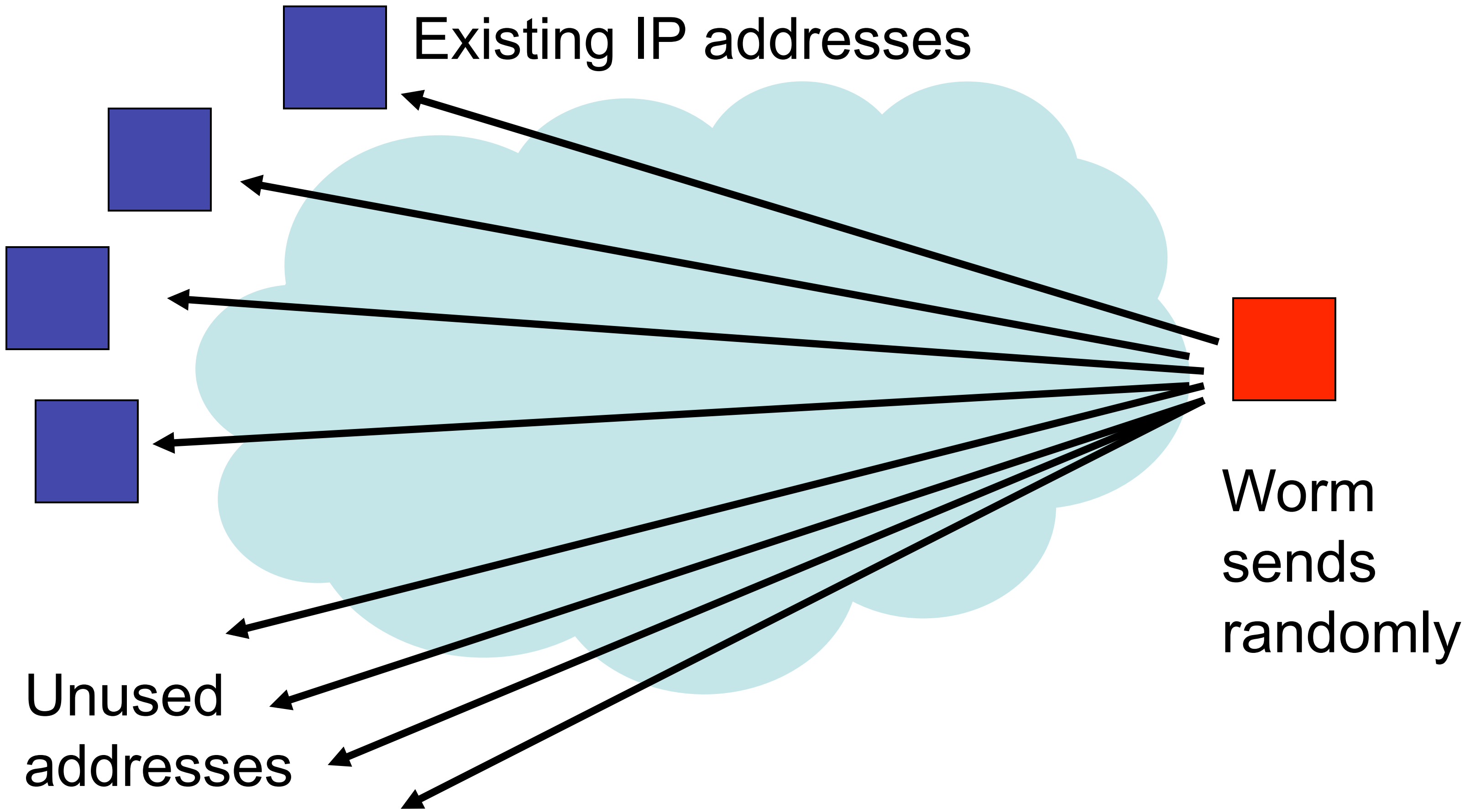
```
alert tcp any any -> 192.168.1.0/24 143 (content:"|E8C0 FFFF FF|/bin/sh"; msg:"New IMAP Buffer Overflow detected!";)
```

  - 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?

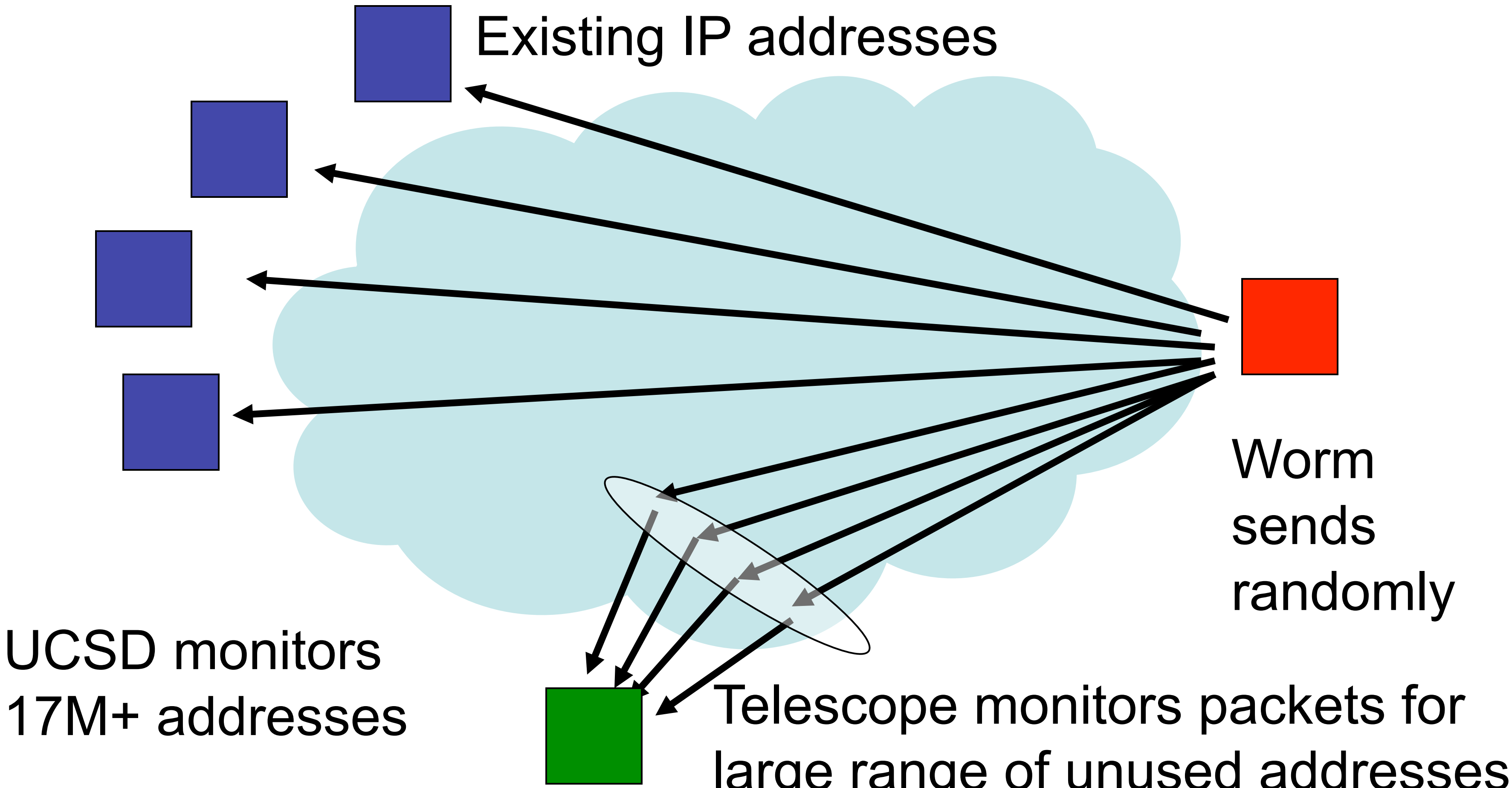
Internet Telescopes

- 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.

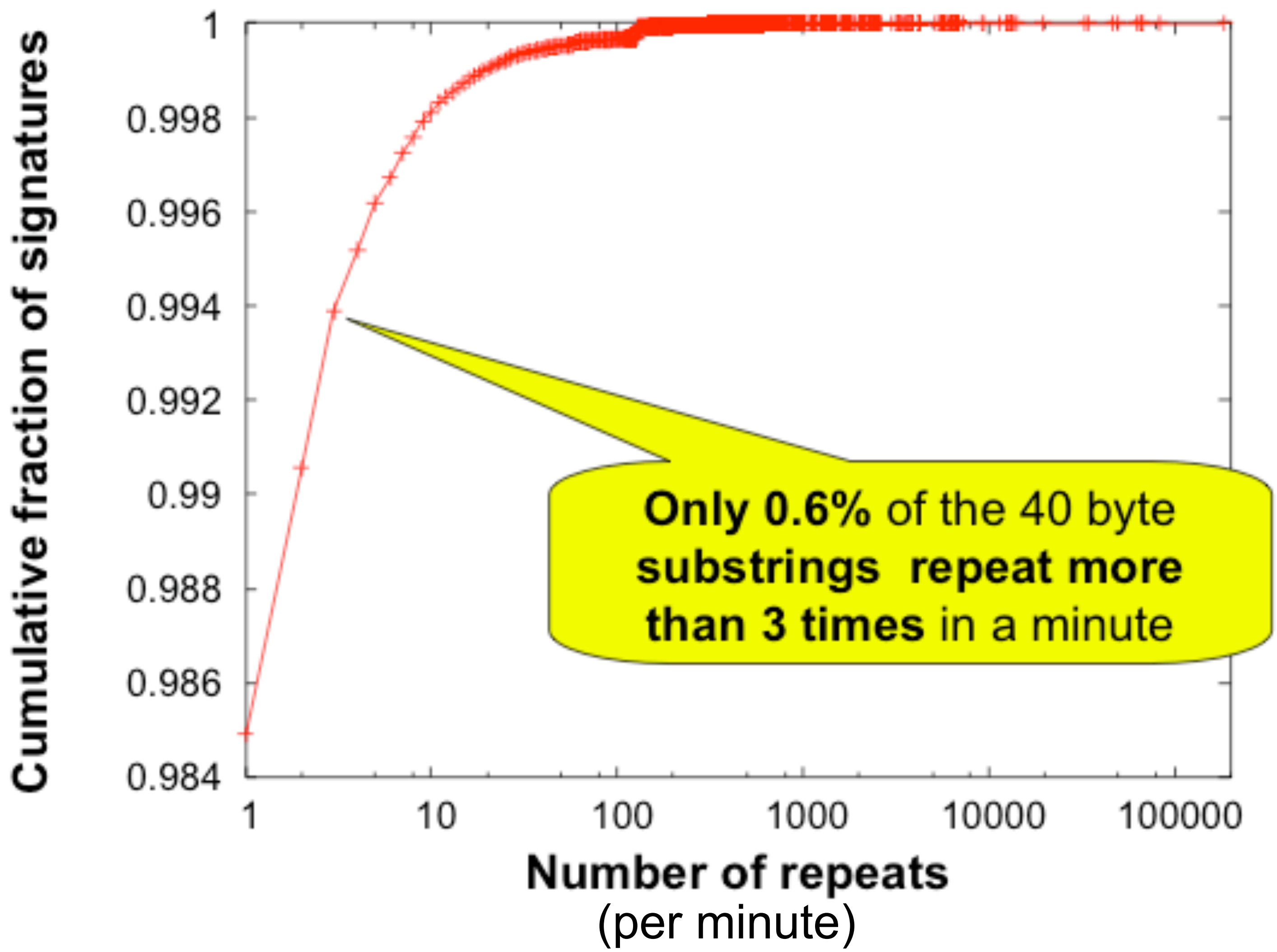


Automated Worm Fingerprinting

- Paper by Singh, Estan, Varghese, and Savage

- Assumptions:
  - All worms have invariant content
  - Invariant packets will appear frequently on the network
    - Worms are trying to propagate, after all
  - Packet sources and destinations will show high variability
    - Sources: over time number of distinct infected hosts will grow
    - Destinations: worms scan randomly
    - Distribution will be roughly uniform (unlike regular traffic that tends to be clustered)

High-prevalence strings are rare



Naïve Content Sifting

- ProcessTraffic(packet, srcIP, dstIP) {

```
  count[packet]++;
  Insert(srcIP, dispersion[packet].sources);
  Insert(dstIP, dispersion[packet].dests);
  if (count[packet] > countThresh
     && size(dispersion[packet].sources) > srcThresh
     && size(dispersion[packet].dests) > dstThresh) {
       Alarm(packet)
  }
}
```

- Tables count and dispersion are indexed by entire packet content.

Problems with Naïve approach

- Frequency count is inaccurate:
  - Misses common substrings
  - Misses shifted content
  - Ideally, would index count and dispersion by all substrings of packet content (of some length)
- Counting every source and destination is expensive.
- Too much data to process every packet.
  - Most packets are going to be uninteresting.
  - Tables count and dispersion will be huge!

Engineering Challenges

- To support 1Gbps line rate have 12us to process each packet.

- Naïve implementation can easily use 100MB/sec for tables.

- Don't want to just do naïve sampling
  - E.g. don't want to just look at 1/N of the packets because detecting the worm will take N times as long

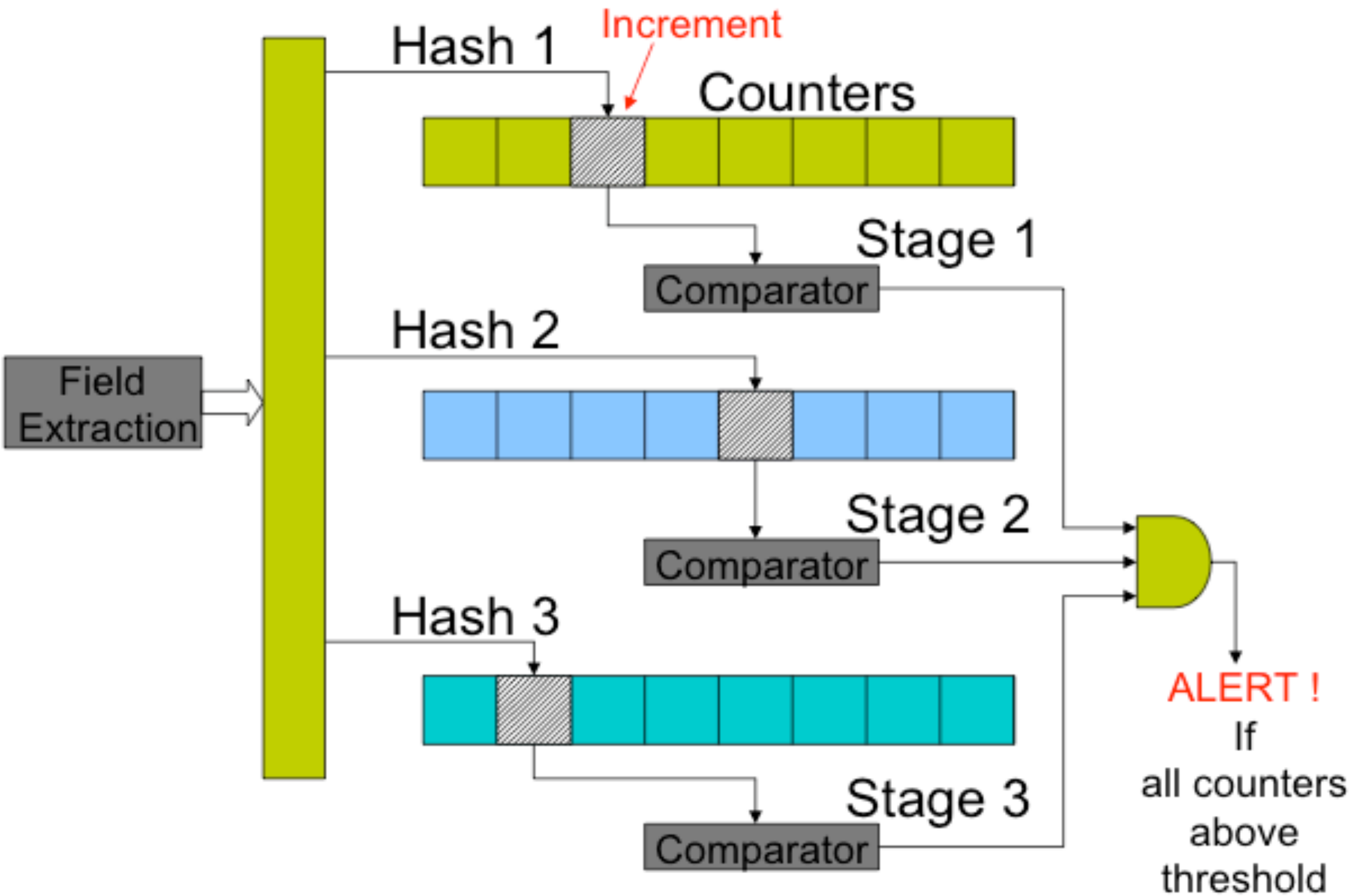
Practical Content Sifting

- Reduce size of count table by:
  - Hashing the packet content to a fixed size (*not* cryptographic hashes)
  - Hash collisions may lead to false positives
  - So, do multiple different hashes (say 3) -- worm content is flagged only if counts along all hashes exceed a threshold

- Include the destination port in the hash of the packet content
  - Current worms target specific vulnerabilities, so they usually aim for a particular port.

- To check for substring matches they propose to use a Rabin fingerprint
  - Probabilistic, incrementally computable hash of substrings of a fixed length.

Multistage Filters, Pictorially



Tracking Address Dispersion

- In this case, we care about the number of distinct source (or destination) addresses in packets that contain suspected worm data.

- Could easily keep an exact count by using a hash table, but that becomes too time and memory intensive.
  – In the limit, need one bit per address to mark whether it has been seen or not.

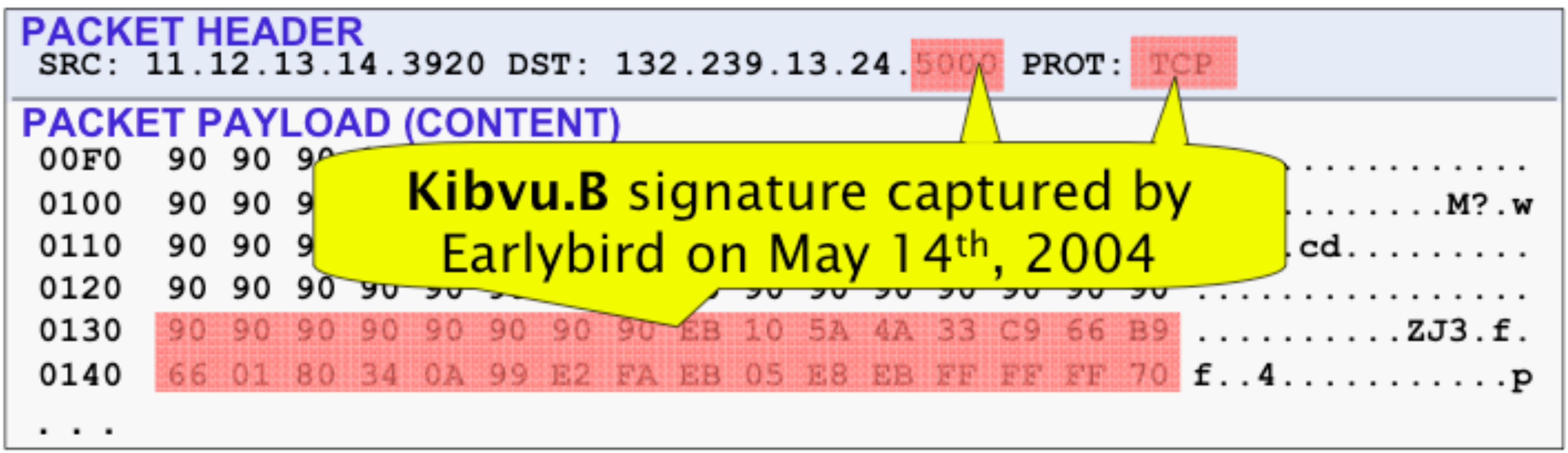
- Instead: Keep an *approximate* count
- Scalable bitmap counters
  – Reduce memory requirements by 5x

Scalable Bitmap Counters

- Suppose there are 64 possible addresses and you want to use only 32 bits to keep track of them.
- High-level idea:
  - Hash the address into a value between 0 and 63
  - Use only the lower 5 bits (yielding 32 possibilities)
  - To estimate actual number of addresses, multiply the number of bits set in the bitmap by 2.

Results

- Earlybird successfully detects and extracts virus signatures from every known recent worm (CodeRed, MyDoom, Sasser, Kibvu.B,…)
- Tool generates content filter rules suitable for use with Snort



Analysis

- False Positives:
  - SPAM
  - BitTorrent
  - Common protocol headers
    - HTTP and SMTP
    - Some P2P system headers
  - Solution: whitelist by hand
- False Negatives:
  - Hard (impossible?) to prove absence of worms
  - Over 8 months Earlybird detected all worm outbreaks reported on security mailing lists

- Countermeasures to content filtering?

Polymorphic Viruses/Worms

- Virus/worm writers know that signatures are the most effective way to detect such malicious code.
- Polymorphic viruses mutate themselves during replication to prevent detection
  - Virus should be capable of generating many different descendents
  - 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 C++ 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

Witty Worm

- Released March 19, 2004.
- Single UDP packet exploits flaw in the *passive analysis* of Internet Security Systems products.
- “Bandwidth-limited” UDP worm like Slammer.
- Vulnerable pop. (12K) attained in 75 minutes.
- Payload: *slowly corrupt random disk blocks*.

Witty, con't

- Flaw had been announced the *previous day*.
- Telescope analysis reveals:
  - Initial spread seeded via a *hit-list*.
  - In fact, targeted a U.S. military base.
  - Analysis also reveals "Patient Zero", a European retail ISP.
- Written by a Pro.
- "Zero-day" exploits are becoming more common

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
  - Via creating artificial heterogeneity

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 detectible 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, tracert
- 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 worms
- 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

```
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:

```
$ ftp ftp.netscape.com 21
Connected to ftp.gftp.netscape.com.
220-36
220 ftpnscp.newaol.com FTP server (SunOS 5.8) ready.
Name (ftp.netscape.com:stevez):
331 Password required for stevez.
Password:
530 Login incorrect.
ftp: Login failed.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> system
215 UNIX Type: L8 Version: SUNOS
ftp>
```

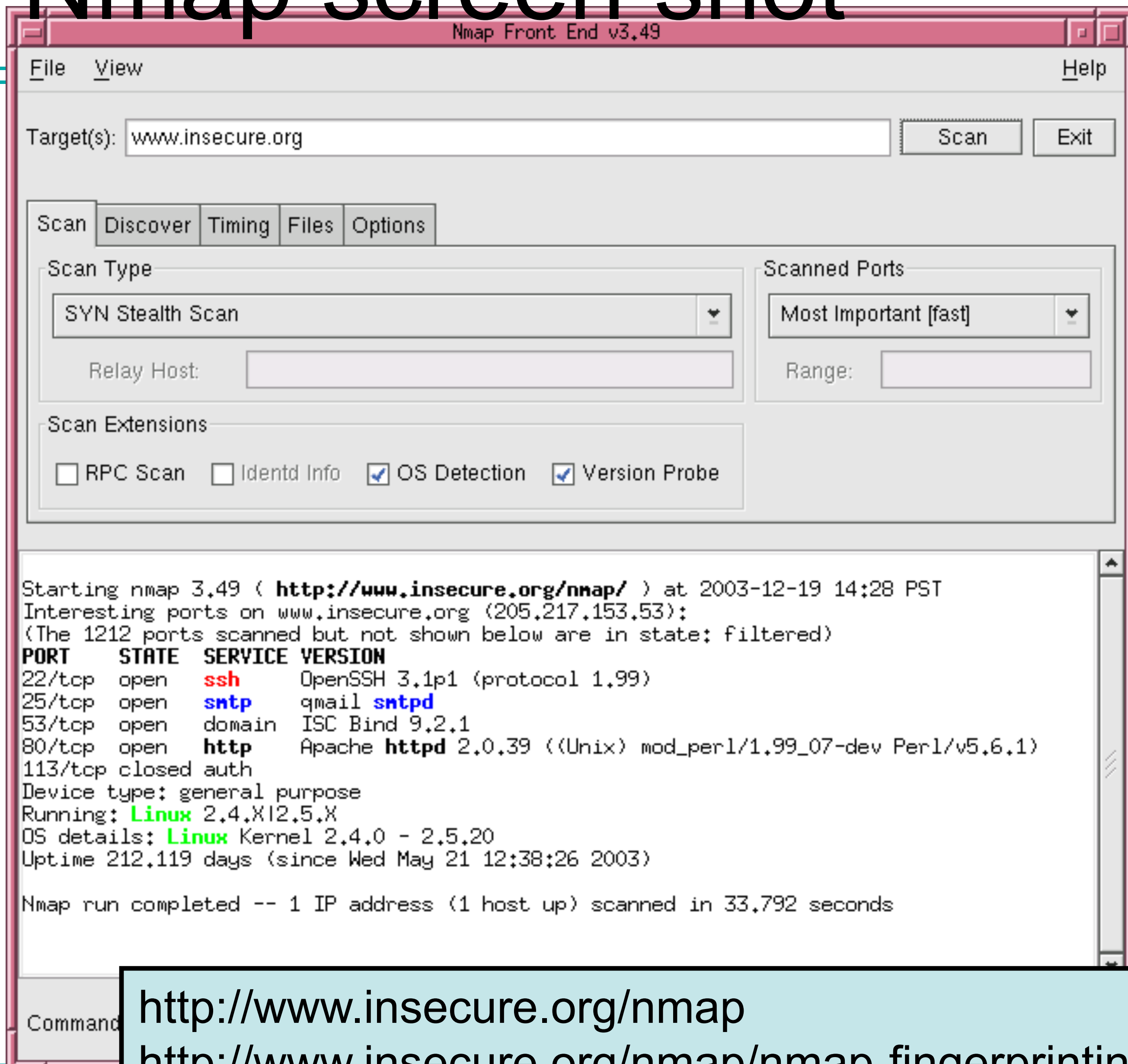
Determining OS

- 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 auditing 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 screen shot



http://www.insecure.org/nmap
http://www.insecure.org/nmap/nmap-fingerprinting-article.html