

CSE331: Introduction to Networks and Security

Lecture 18
Fall 2006



Announcements

- Project 2 is due next Weds.
- Homework 2 has been assigned:
 - It's due on Monday, November 6th.

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

The screenshot shows the Nmap Front End v3.49 application window. The target is set to `www.insecure.org`. The scan type is `SYN Stealth Scan`. The scanned ports are set to `Most Important [fast]`. The scan extensions include `OS Detection` and `Version Probe`. The output shows the following results:

```
Starting nmap 3.49 ( http://www.insecure.org/nmap/ ) at 2003-12-19 14:28 PST
Interesting ports on www.insecure.org (205.217.153.53):
(The 1212 ports scanned but not shown below are in state: filtered)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 3.1p1 (protocol 1.99)
25/tcp    open  smtp     qmail smtpd
53/tcp    open  domain   ISC Bind 9.2.1
80/tcp    open  http     Apache httpd 2.0.39 ((Unix) mod_perl/1.99_07-dev Perl/v5.6.1)
113/tcp   closed auth
Device type: general purpose
Running: Linux 2.4.X|2.5.X
OS details: Linux Kernel 2.4.0 - 2.5.20
Uptime 212.119 days (since Wed May 21 12:38:26 2003)

Nmap run completed -- 1 IP address (1 host up) scanned in 33.792 seconds
```

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

Kinds of Auditing done

- Nessus web pages:
 - Backdoors
 - CGI abuses
 - Denial of Service
 - Finger abuses
 - Firewalls
 - FTP
 - Gain a shell remotely
 - Gain root remotely
 - Netware
 - NIS
 - Port scanners
 - Remote file access
 - RPC
 - Settings
 - SMTP problems
 - SNMP
 - Useless services
 - Windows
 - Windows : User management
- Doing this kind of auditing by hand is complex and error prone
- These tools aren't fool proof or complete.

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?

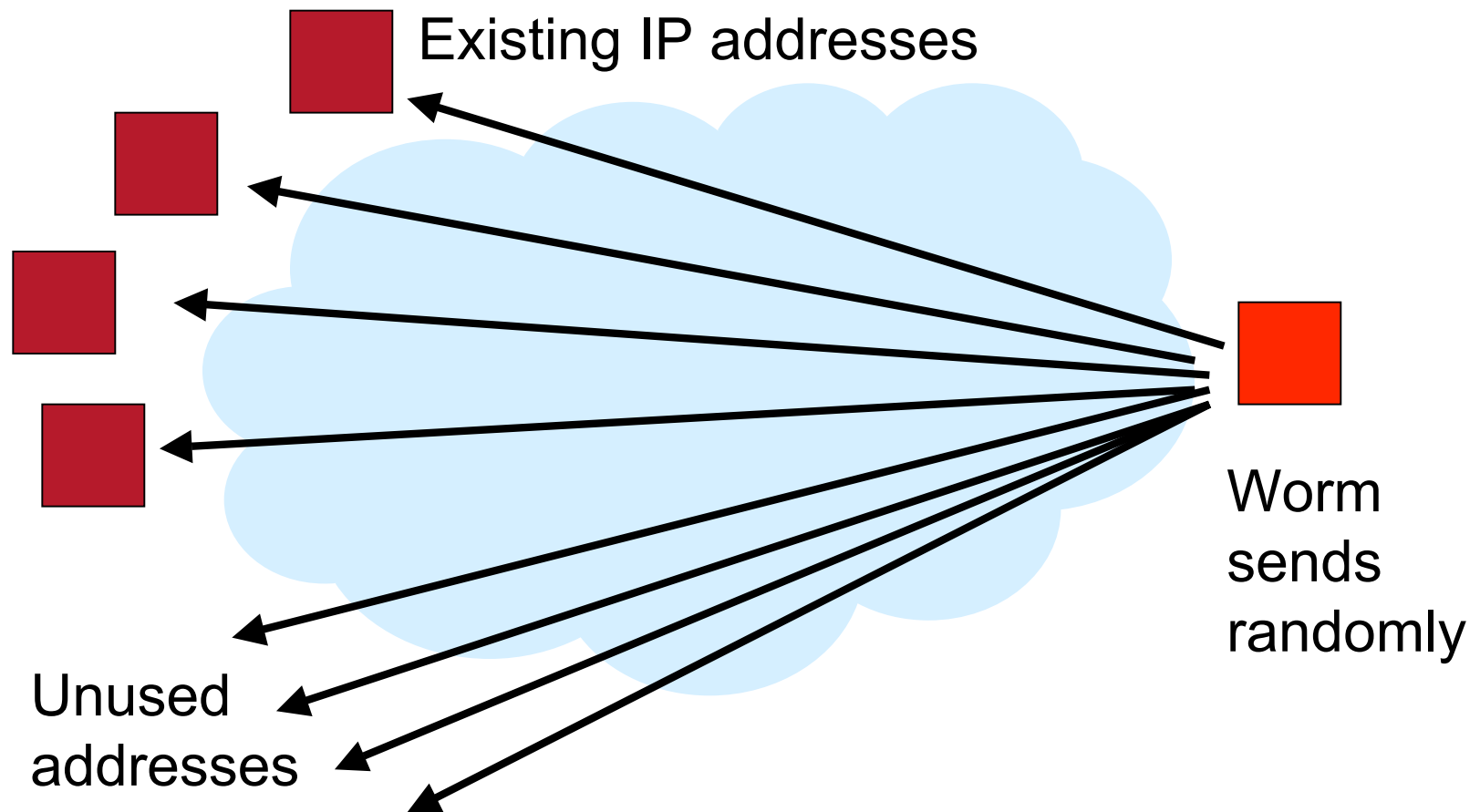


Detection & Prevention Recap

- Many strategies for intrusion detection
 - So far, the techniques we've seen are local to a machine or local network.
- What about large scale behavior?
- Virus/worm scanners work well if known signatures are available
 - Constructing signatures can be hard
 - Reaction time must be very quick

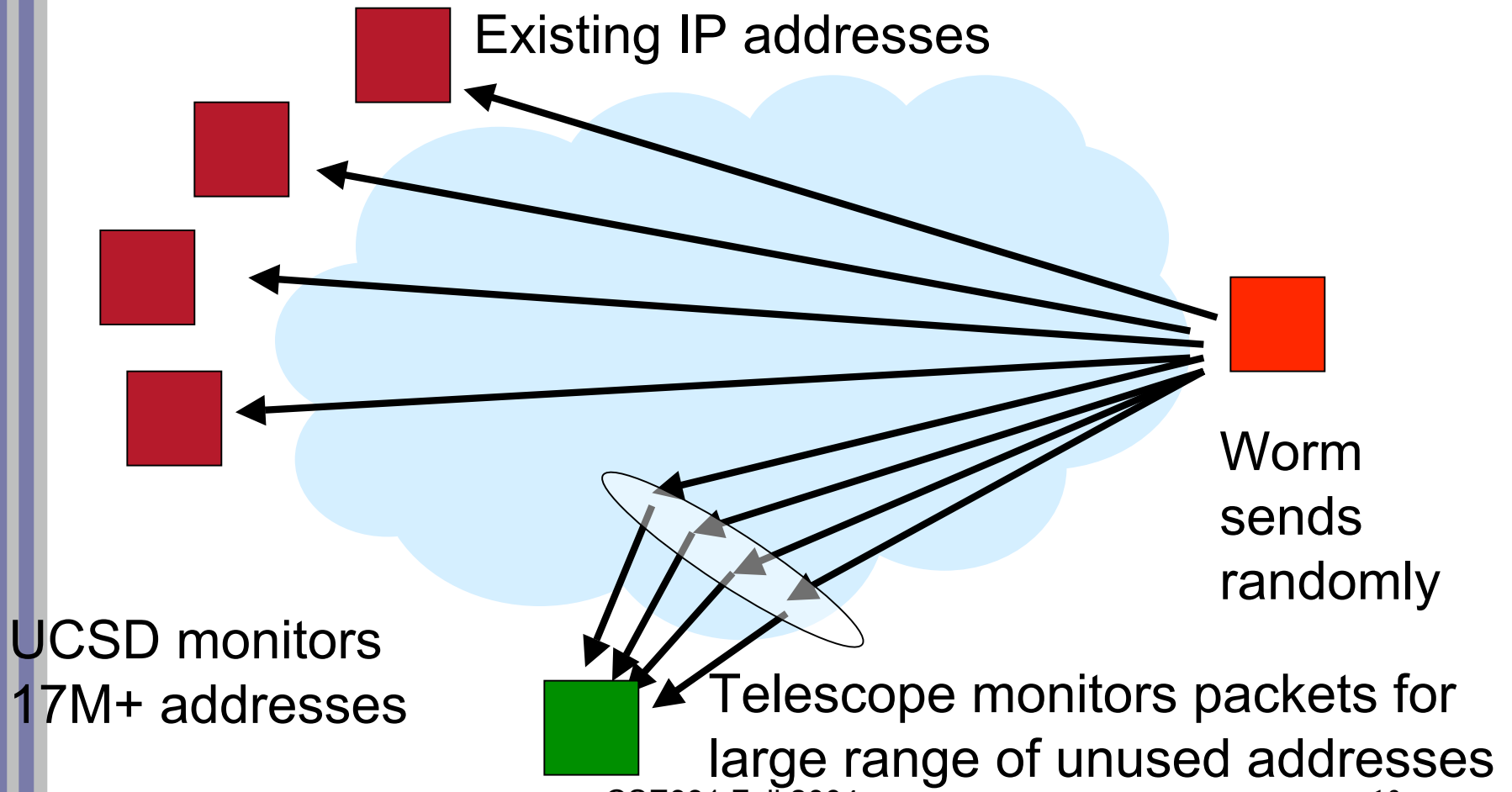
Internet Telescopes

- 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.

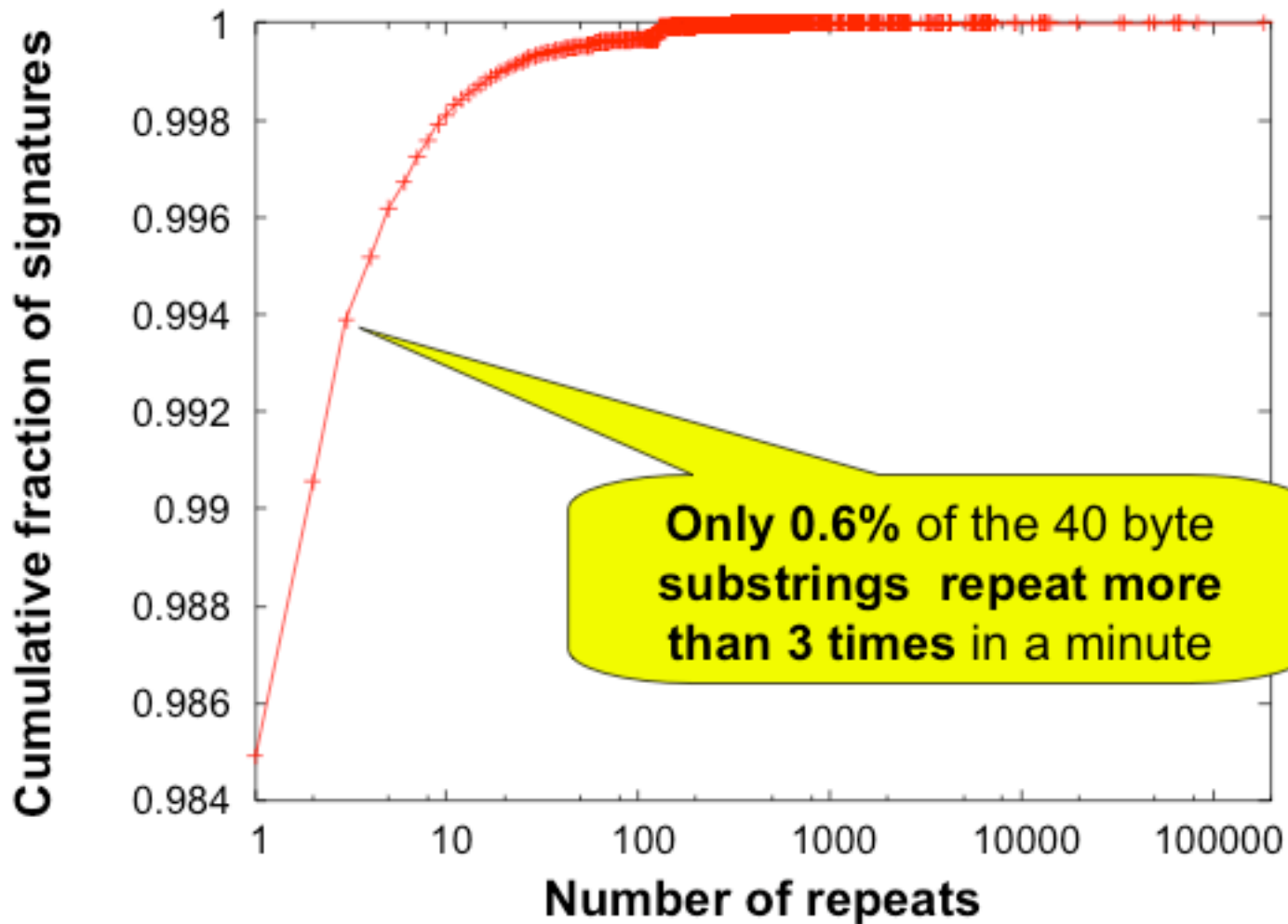




Automated Worm Fingerprinting

- 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
 - Large # of packet sources and destinations
- Other possibilities
 - Destination addr. distribution will be roughly uniform (unlike regular traffic that tends to be clustered)
 - Doesn't work for topologically spreading malware (like e-mail viruses)

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

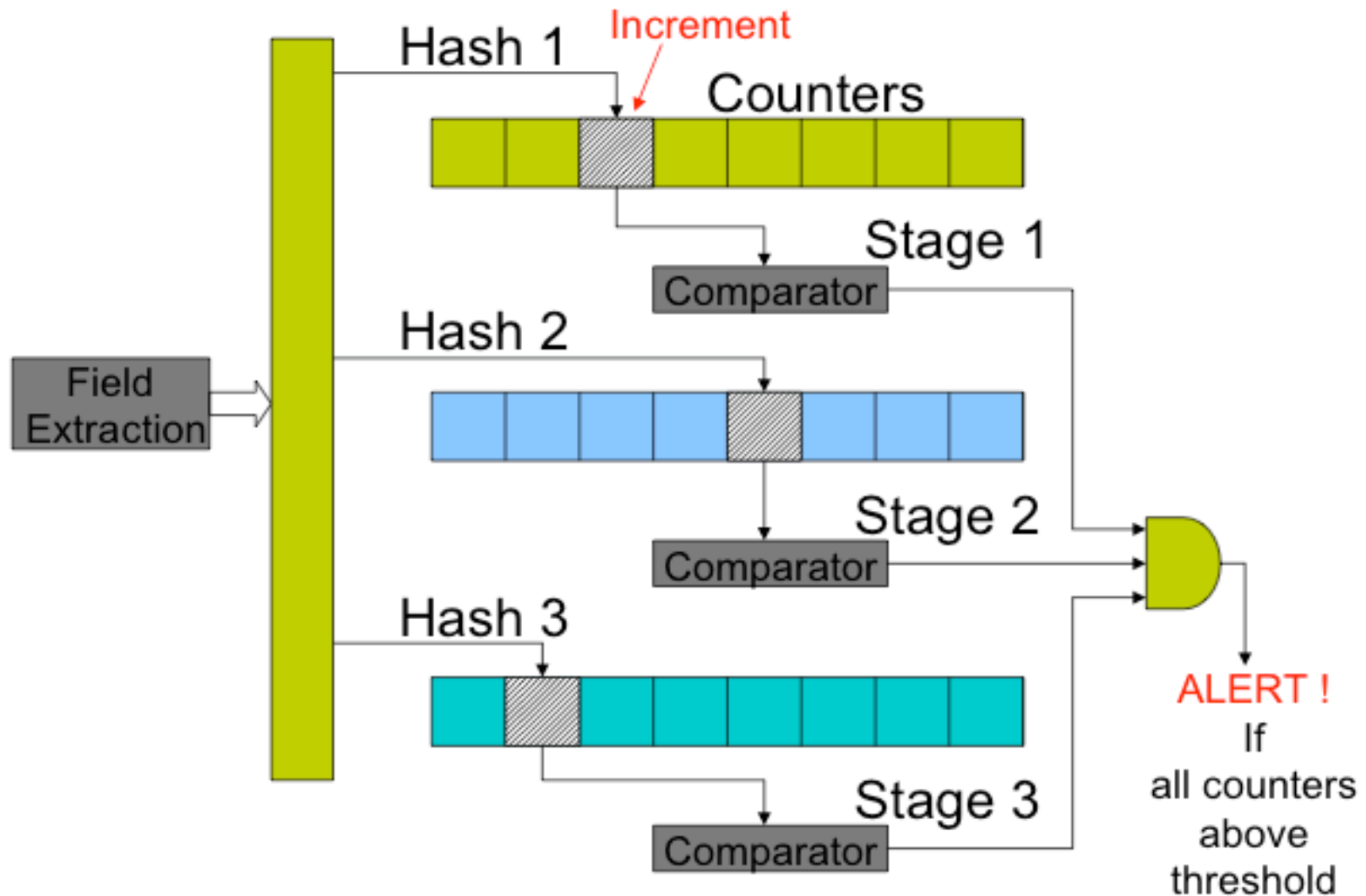
---

- 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)
  - 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

## PACKET HEADER

SRC: 11.12.13.14.3920 DST: 132.239.13.24.5000 PROT: TCP

## PACKET PAYLOAD (CONTENT)

```
00F0 90 90 90
0100 90 90 90M?.w
0110 90 90 90 cd.....
0120 90 90 90 90 90
0130 90 90 90 90 90 90 90 EB 10 5A 4A 33 C9 66 B9ZJ3.f.
0140 66 01 80 34 0A 99 E2 FA EB 05 E8 EB FF FF FF 70 f..4.....p
...
```

**Kibvu.B signature captured by  
Earlybird on May 14<sup>th</sup>, 2004**

# Analysis

---

- False Positives:
  - SPAM
    - No solution yet
  - BitTorrent
    - Replicates packets, so it actually looks like worm traffic
  - 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

# 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*)
- What about violating the assumptions?
  - Invariant content
  - Worm propagates randomly
  - Worm propagates quickly