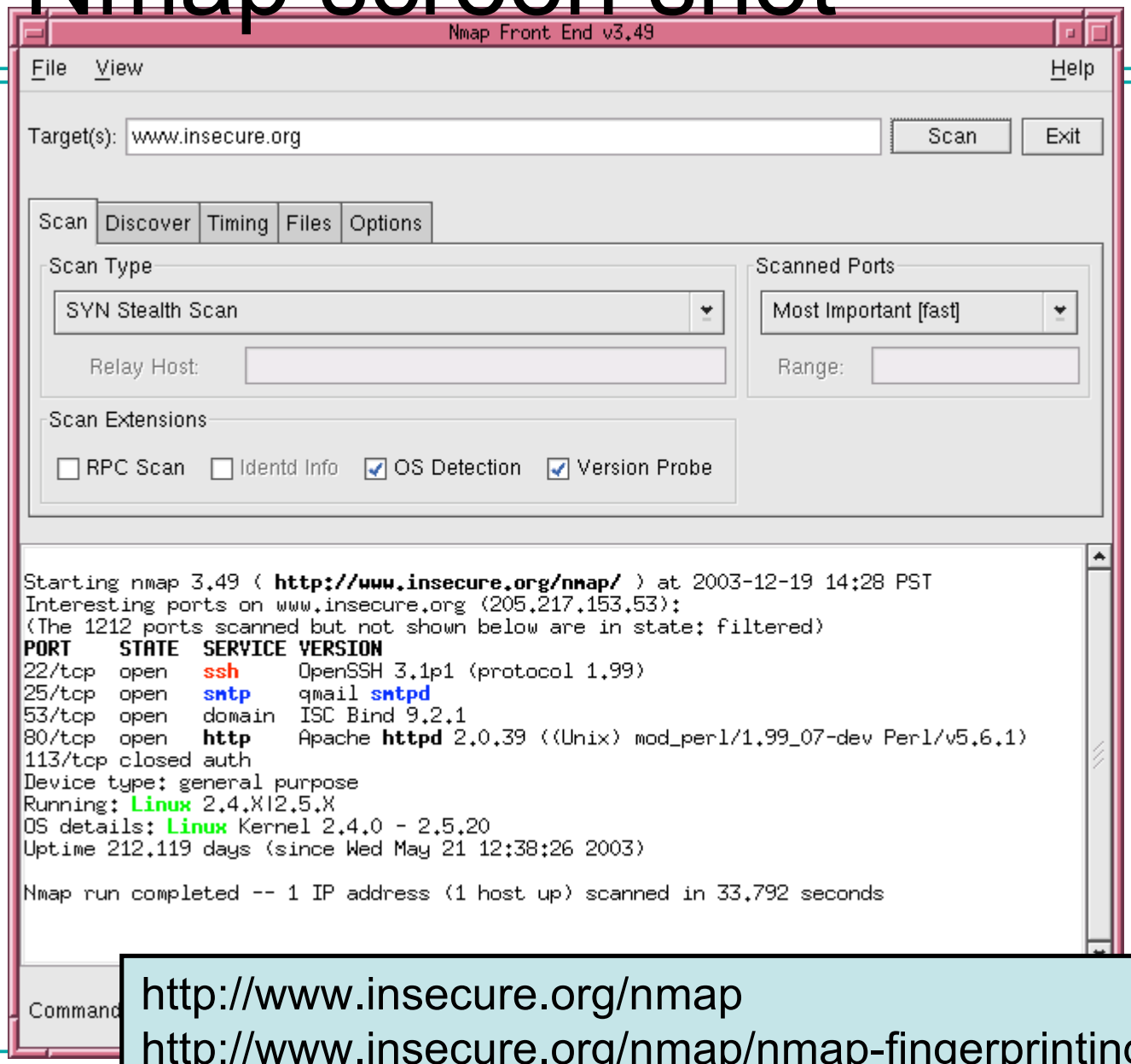
CIS 551 / TCOM 401

Computer and Network Security

Spring 2006

Lecture 22

Nmap screen shot



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.

Defenses

- Disable ping service, unneeded ports
  - Not always possible (you do want some connectivity!)
- Firewalls and NATs
  - Filter out inappropriate packets
  - Scanners use broadcast, multicast packets, clever port/flag combinations to thwart firewall filters
- Keep audit logs of requests
  - Generates a lot of data, hard to sift through
  - Clever port scan packages use a “drip scan” approach, sprinkling their scan packets sparsely across several hours or days
- Intrusion detection

Intrusion Detection

- The process of monitoring computers and networks to analyze and detect signs of security threats.
- Layered security is good
  - More layers means more chances to thwart attacks and more places where the attack might be discovered
- Basic detection alone is insufficient
  - Localization (i.e. What machines are infected.)
  - Identification (i.e. What is the nature of the attack.)
  - Assessment (i.e. What is the extent of the damage, what action to take.)
- Evaluation of intrusion detection scheme based on:
  - False positives: alerts to situations that aren't attacks
  - False negatives: attacks that are missed

Intrusion Detection Systems

- Network-based IDSs:
  - Use raw packet data to look for possible attacks
  - Run network adapter in promiscuous mode
- Host-based IDSs:
  - Log OS specific events (e.g. system calls)
- Looking for:
  - Known attack patterns (e.g. versus SMTP or port scans)
  - Frequency statistics (to detect DoS)
  - Anomalous behavior (based on some profile of “usual” behavior)

- Example: ZoneAlarm software can detect port scans.

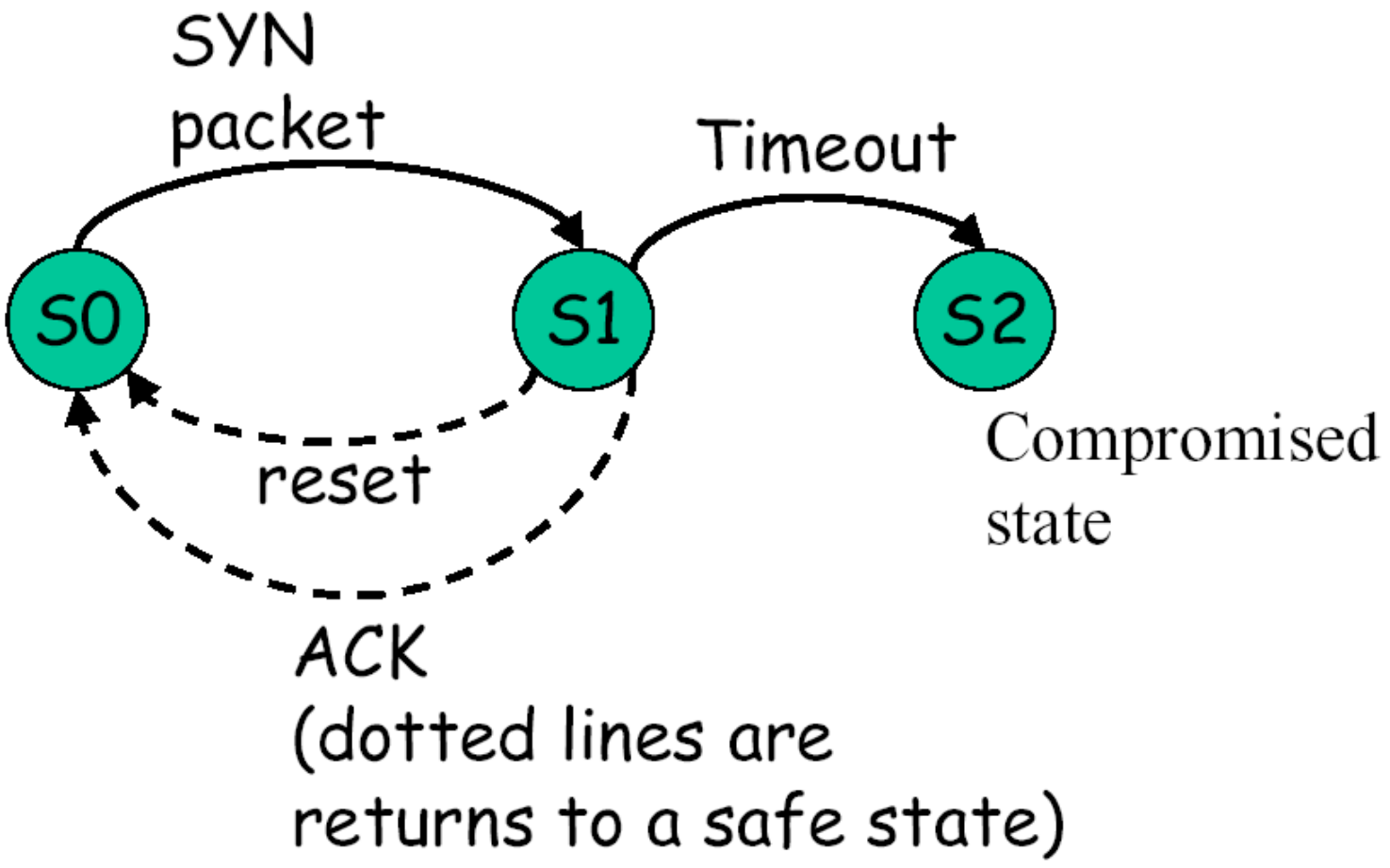
Intrusion Detection

- Known attack signatures
  - As for worms and viruses
- Anomaly detection:
  - Build a profile of “typical” or “normal” behavior
  - What programs are usually running
  - Where do people usually log in from
  - What is the normal timing behavior of your keystrokes
  - How do you arrange your computer desktop…

State Transition Analysis Techniques

- STAT: developed by Eckmann, Vigna, and Kemmerer.
- Defines attack scenarios as sequence of transitions in the security state
  - "state" captures information relevant to security
  - e.g. file ownership, userids, protocol stack states, etc.
- Transitions are the necessary actions that lead to compromise

STAT state diagram example



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?

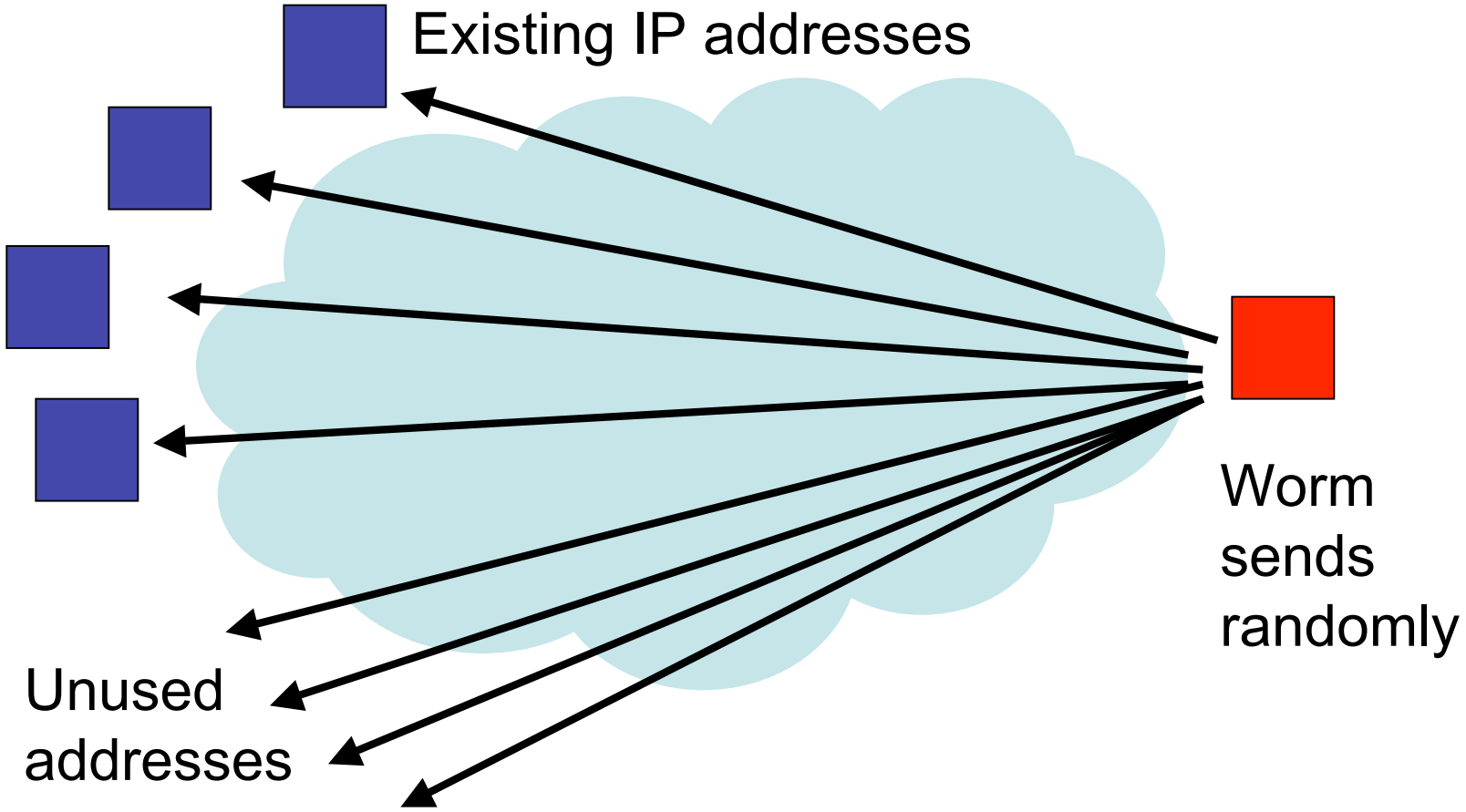
Detection & Prevention Recap

- 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

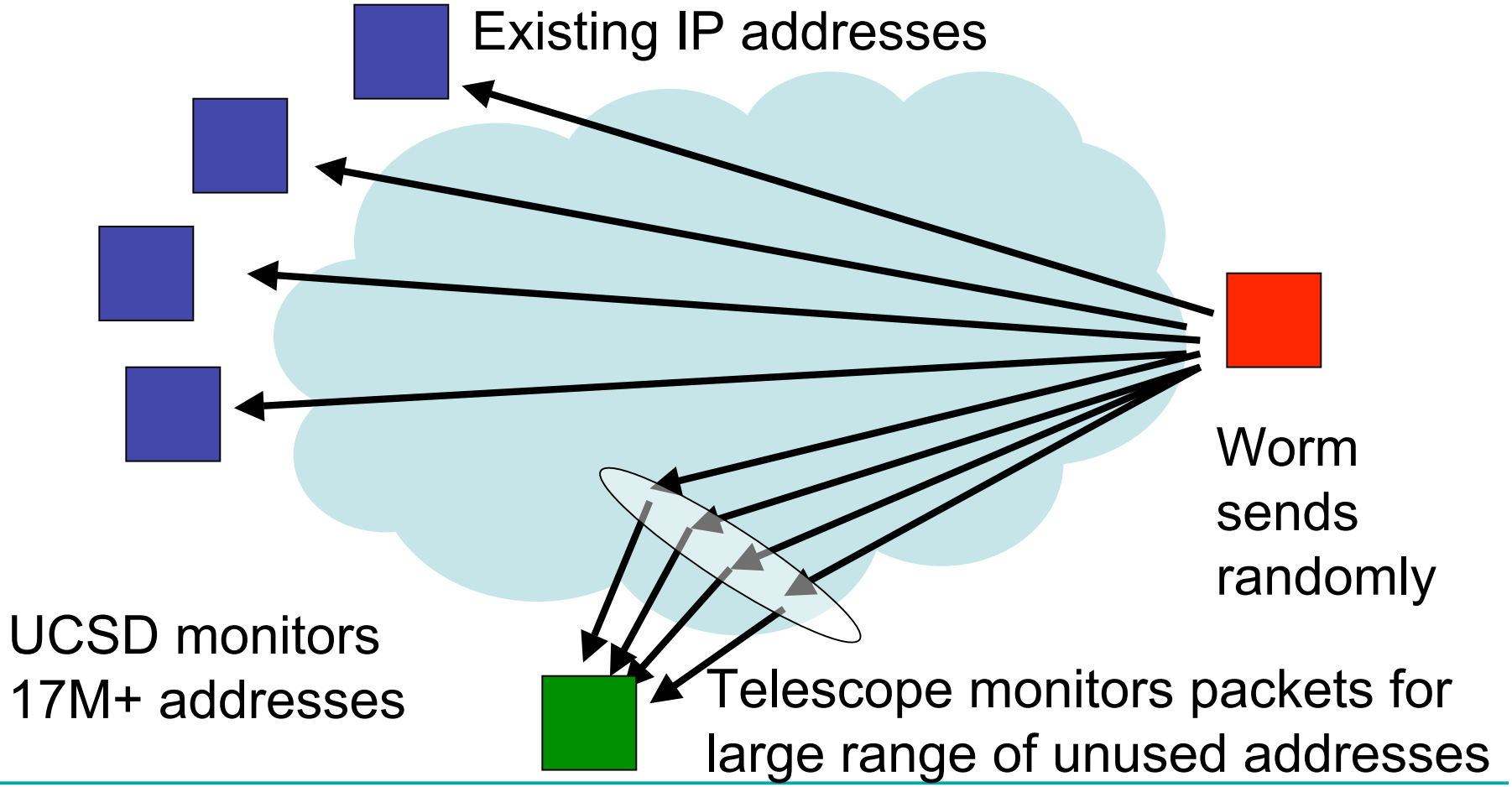
Internet Telescopes

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



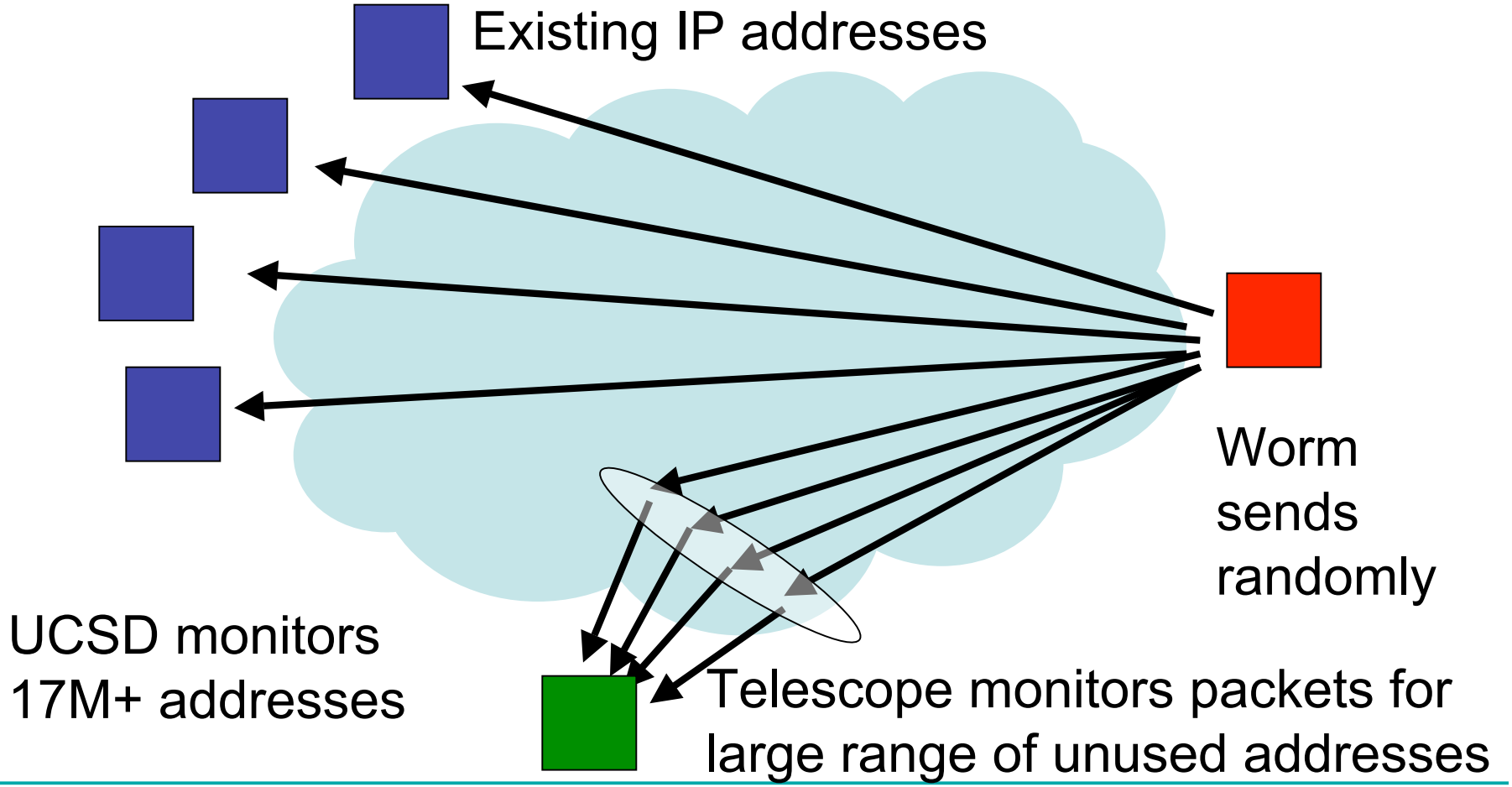
Internet Telescopes

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



Internet Telescopes

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

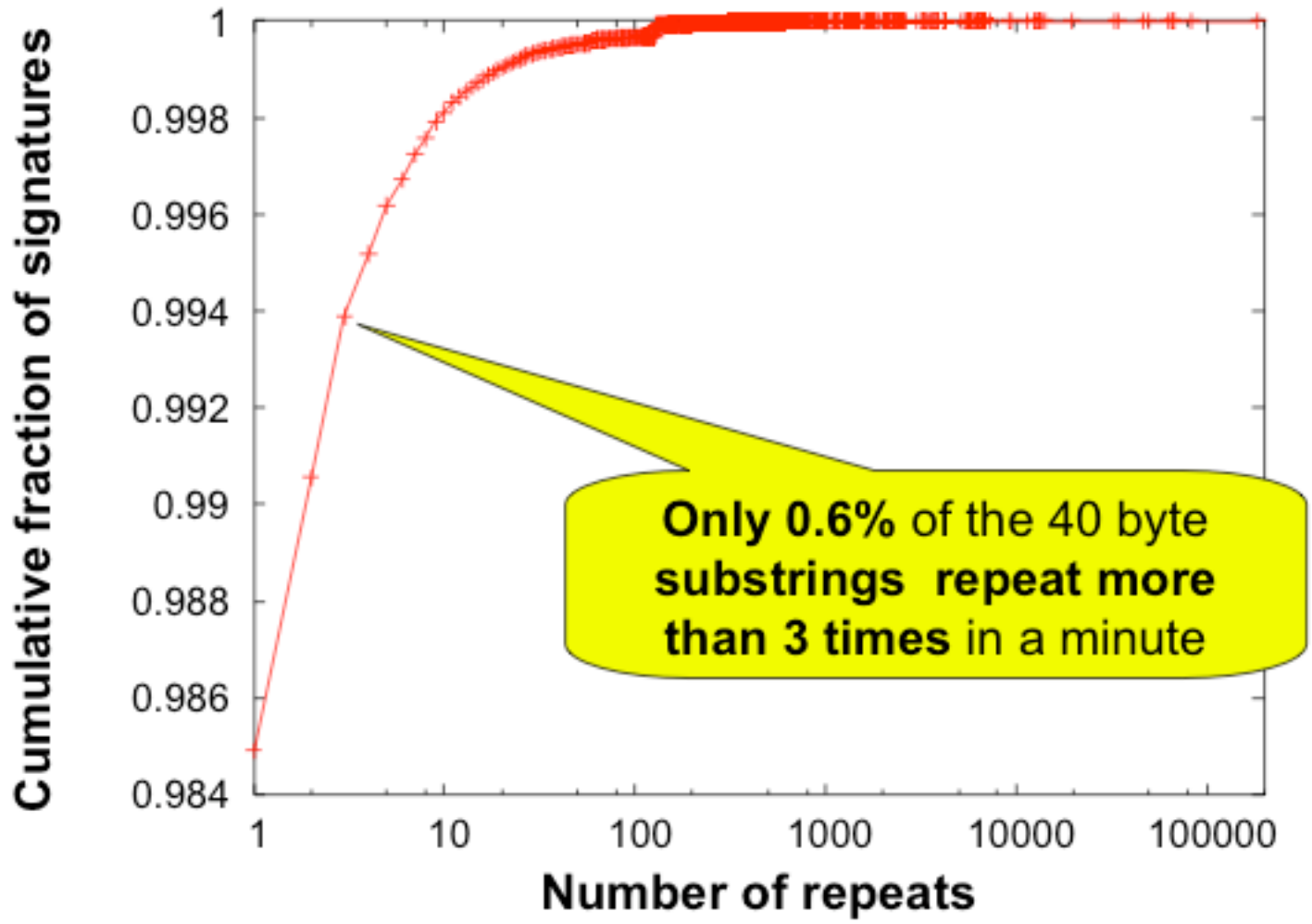


Automated Worm Fingerprinting

- 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!

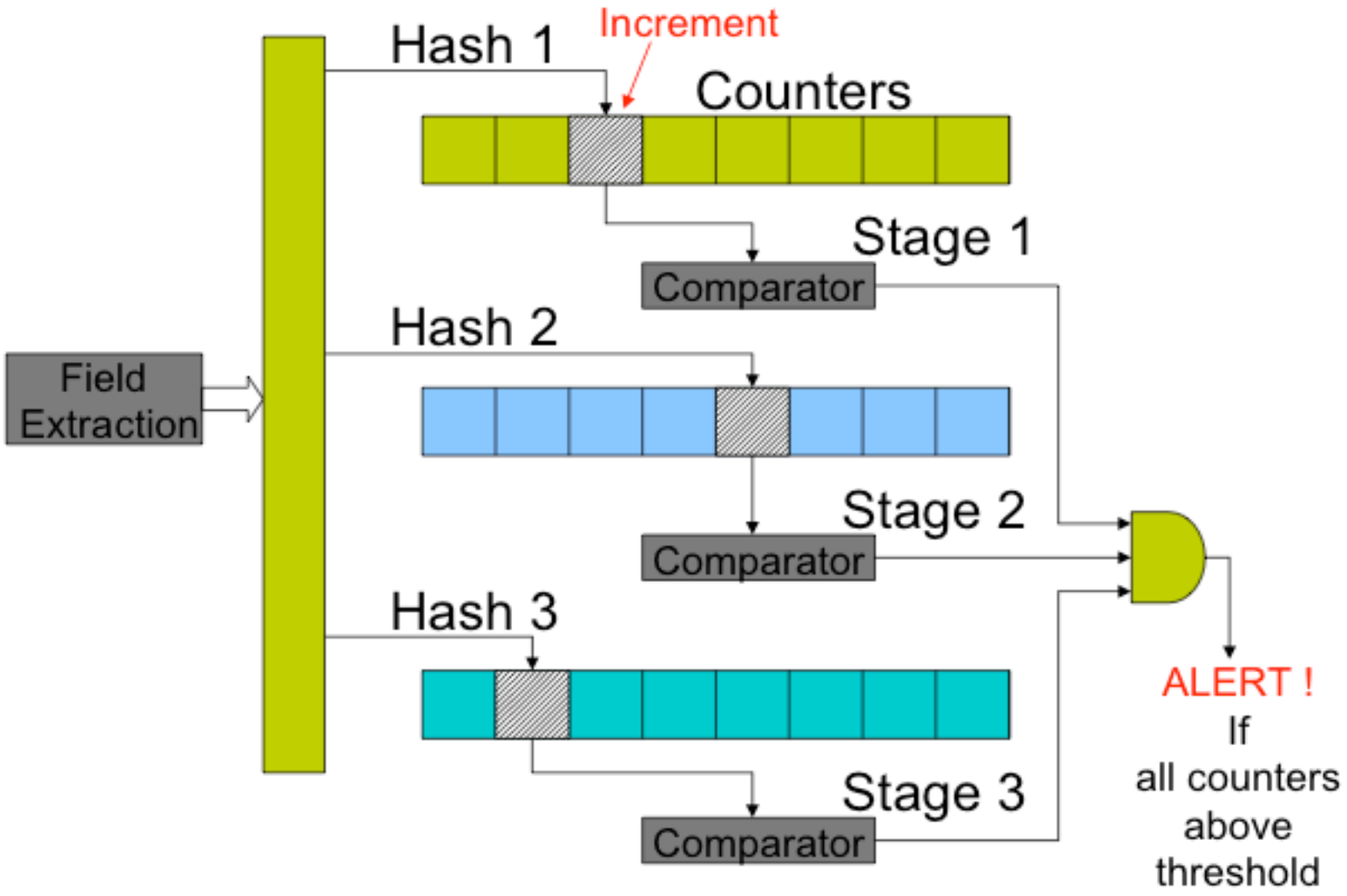
Engineering Challenges

- 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.

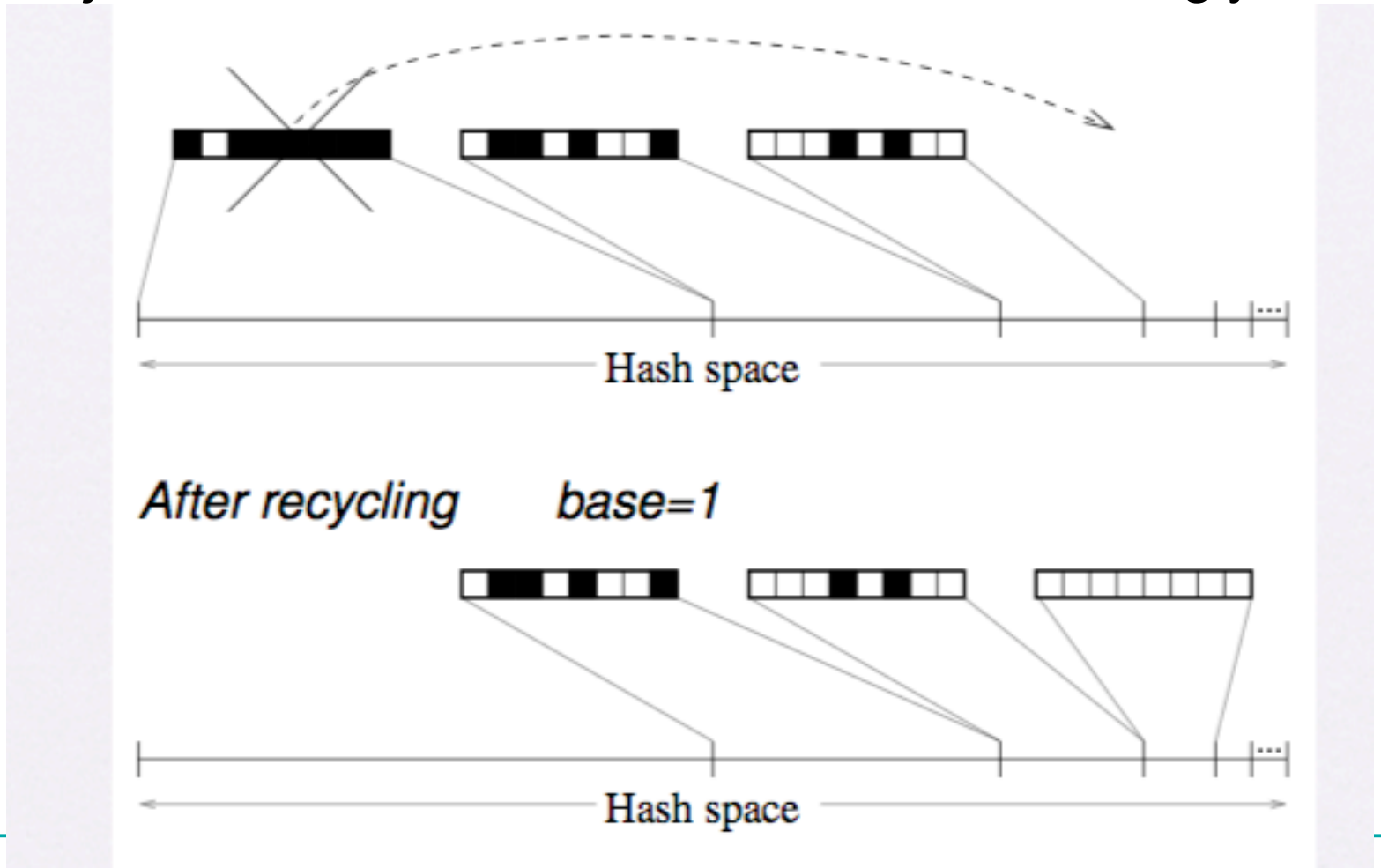
- Instead: Keep an *approximate* count
- Scalable bitmap counters
  - Reduce memory requirements by 5x

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)
  - To estimate actual number of addresses, multiply the number of bits set in the bitmap by 2.

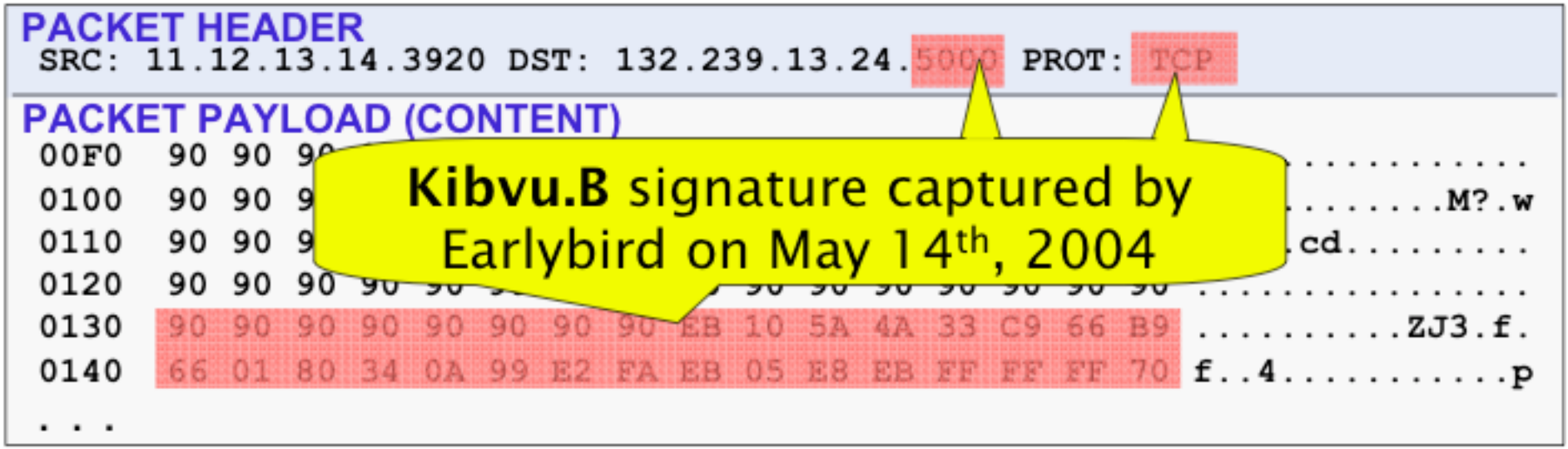
Multiple Bitmaps, Pictorially

- Recycle bitmaps after they fill up
- Adjust the scale factors on the counts accordingly



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

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

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 ala’ 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.