

CIS/TCOM 551

# Computer and Network Security

Spring 2005

Lecture 8

# Announcements

---

- Midterm 2 will be held next Thursday: March 31st
  - In class
  - Format similar to first midterm: short answer & problem solving
  - Topics: Principles of secure design, software security, Java stack inspection, worms & viruses, epidemic model, malware detection & prevention

# Recap

---

- Over last few lectures we've seen:
  - Propagation models for worms on the internet that suggest reaction time must be on the order of seconds or minutes.
  - In particular black-listing is less effective than filtering
  - Filtering relies on signatures
  - Current state-of-the-art: signatures generated by hand
  - Only after identifying that there is a problem, which is hard
- There is a demand for automated intrusion detection
- There is a need for automating signature generation
- Today: "Automated Worm Fingerprinting" by Singh, et al.
  - Some of the pictures in this lecture are taken from presentations by members of this group.

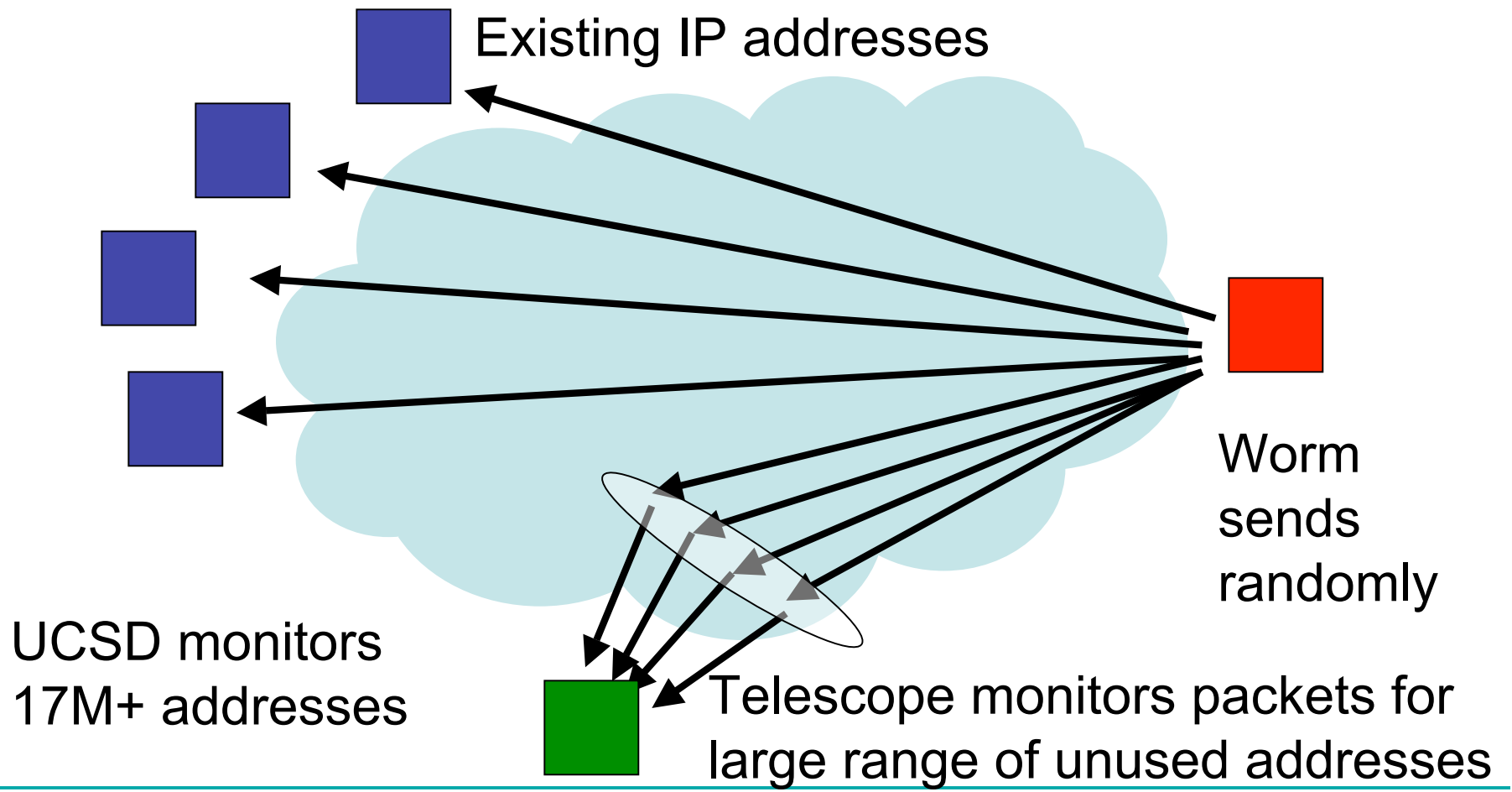
# Signature Generation

---

- Consider a worm that exploits a buffer overflow in the blame server of projects one and two
- What would it look like?
  
- How would you go about isolating a signature?

# Internet Telescopes

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

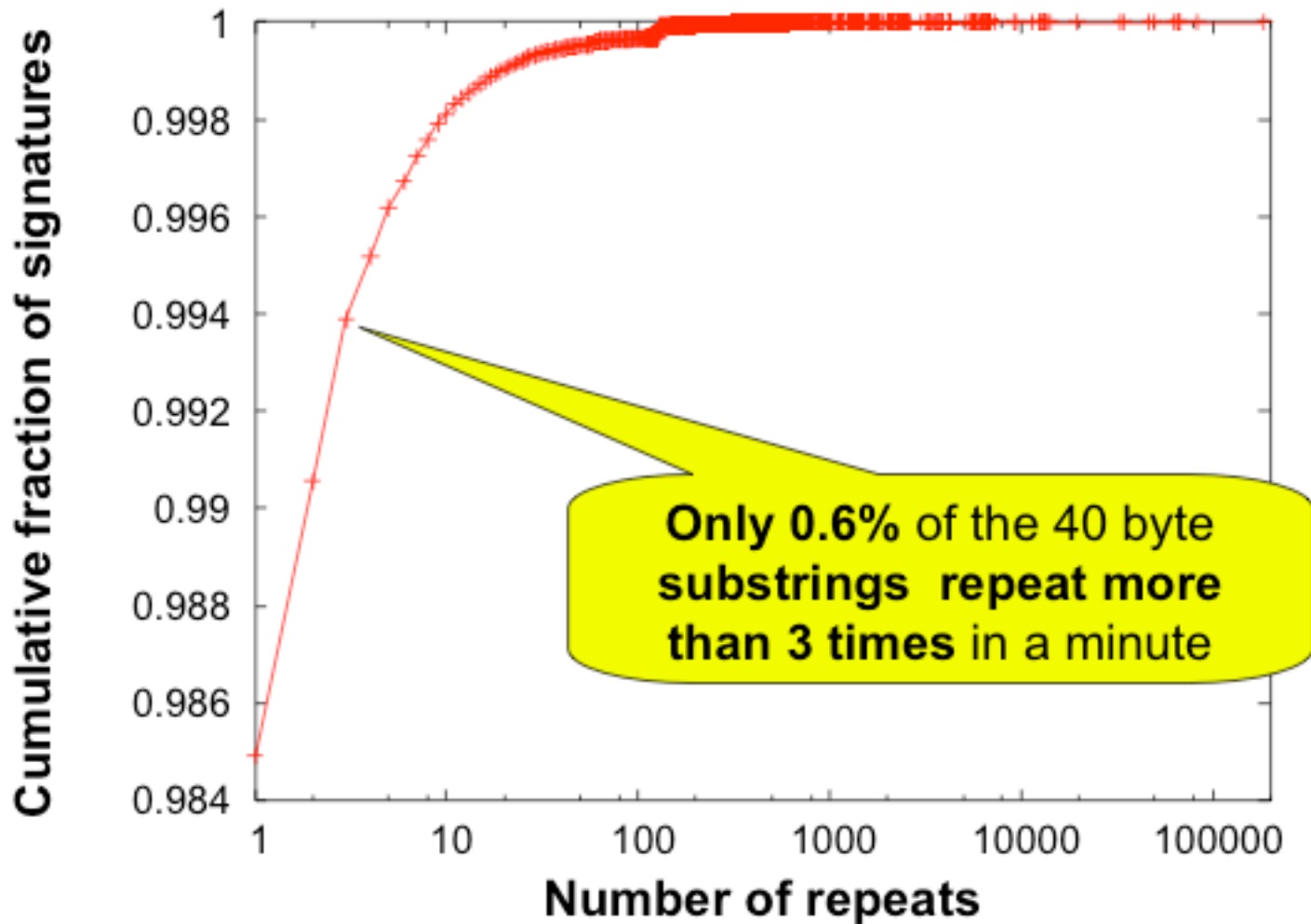


# Automated Worm Fingerprinting

---

- 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

---

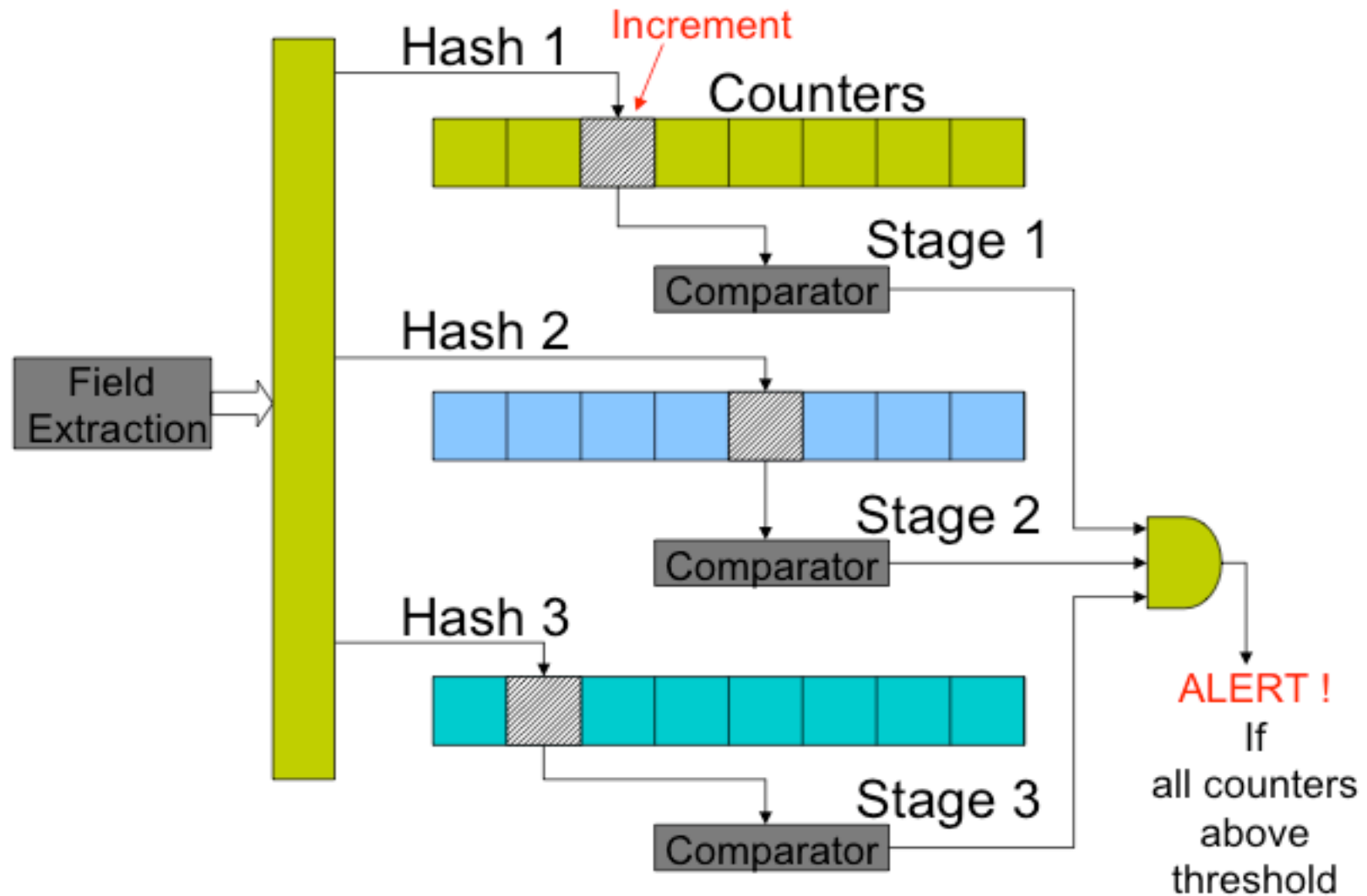
- To support 1Gbps line rate have 12us to process each packet.
- Naïve implementation can easily use 100MB/sec for tables.
- Don't want to just do naïve sampling
  - E.g. don't want to just look at 1/N of the packets because detecting the worm will take N times as long

# Practical Content Sifting

---

- Reduce size of count table by:
  - Hashing the packet content to a fixed size
  - 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

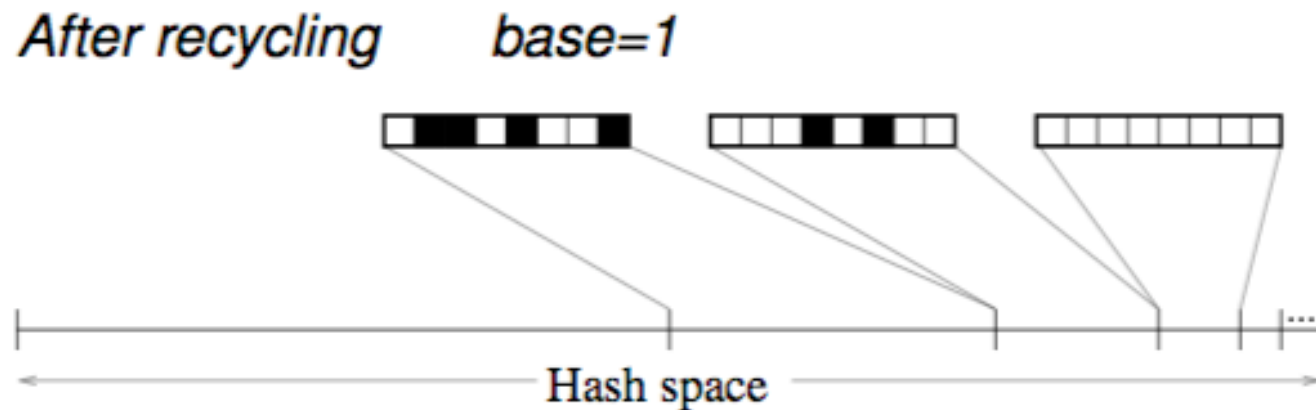
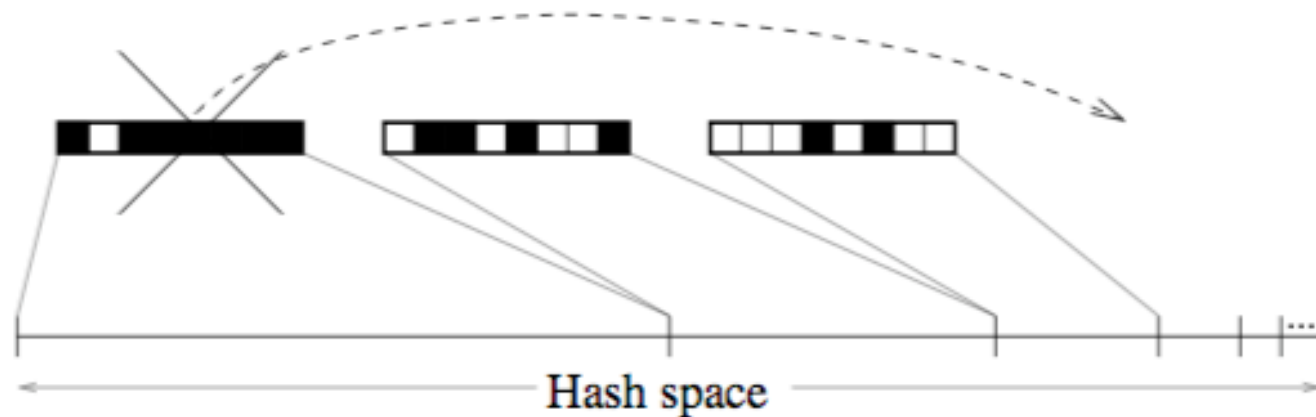
---

- 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

- /



# 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 90 .....M?.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 B9 .....ZJ3.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
  - 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

# 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 phpbb**”)
  - *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.

# 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.