Announcements

• Project 2 is due Today.

• Homework 2 has been assigned:
  – It's due on Monday, November 6th.
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.
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.
Rabin Fingerprints

- Given string of length $n$
  - Write as sequence of bytes: $t_0 \ t_1 \ t_2 \ \ldots \ \ t_n$
- Check all possible substrings of length $k$
- Choose constants $p$ (a prime) and $M$ (modulus)
- Fingerprint for substrings are:
  - $F_1 = (t_0 \cdot p^{k-1} + t_1 \cdot p^{k-2} + \ldots + t_k) \mod M$
  - $F_2 = (t_1 \cdot p^{k-1} + t_2 \cdot p^{k-2} + \ldots + t_{k+1}) \mod M$
    = $(F_1 \cdot p + t_{k+1} - t_1 \cdot p^k) \mod M$
  - $F_3 = (F_2 \cdot p + t_{k+2} - t_2 \cdot p^k) \mod M$
  - $F_i = (F_{i-1} \cdot p + t_{k+i-1} - t_{i-1} \cdot p^k) \mod M$
- For efficiency, precompute table of $x \cdot p^k$
Multistage Filters, Pictorially

Field Extraction

Hash 1

Increment

Counters

Stage 1

Comparator

Hash 2

Stage 2

Comparator

Hash 3

Stage 3

Comparator

ALERT!
If all counters above threshold
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
Analysis

• False Positives:
  – SPAM
    • No solution yet
  – BitTorrent (35% of Internet traffic?!)
    • 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
Attacks

• 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:
    \[ y = x + x \quad \text{vs.} \quad y = 2 \times x \]
  – 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.
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*)