COSC 734: Network Security

Chapter 10 – Malicious Software

Dr. Wei Yu

Dept. of Computer and Information Sciences

Towson University

Email: wyu@towson.edu

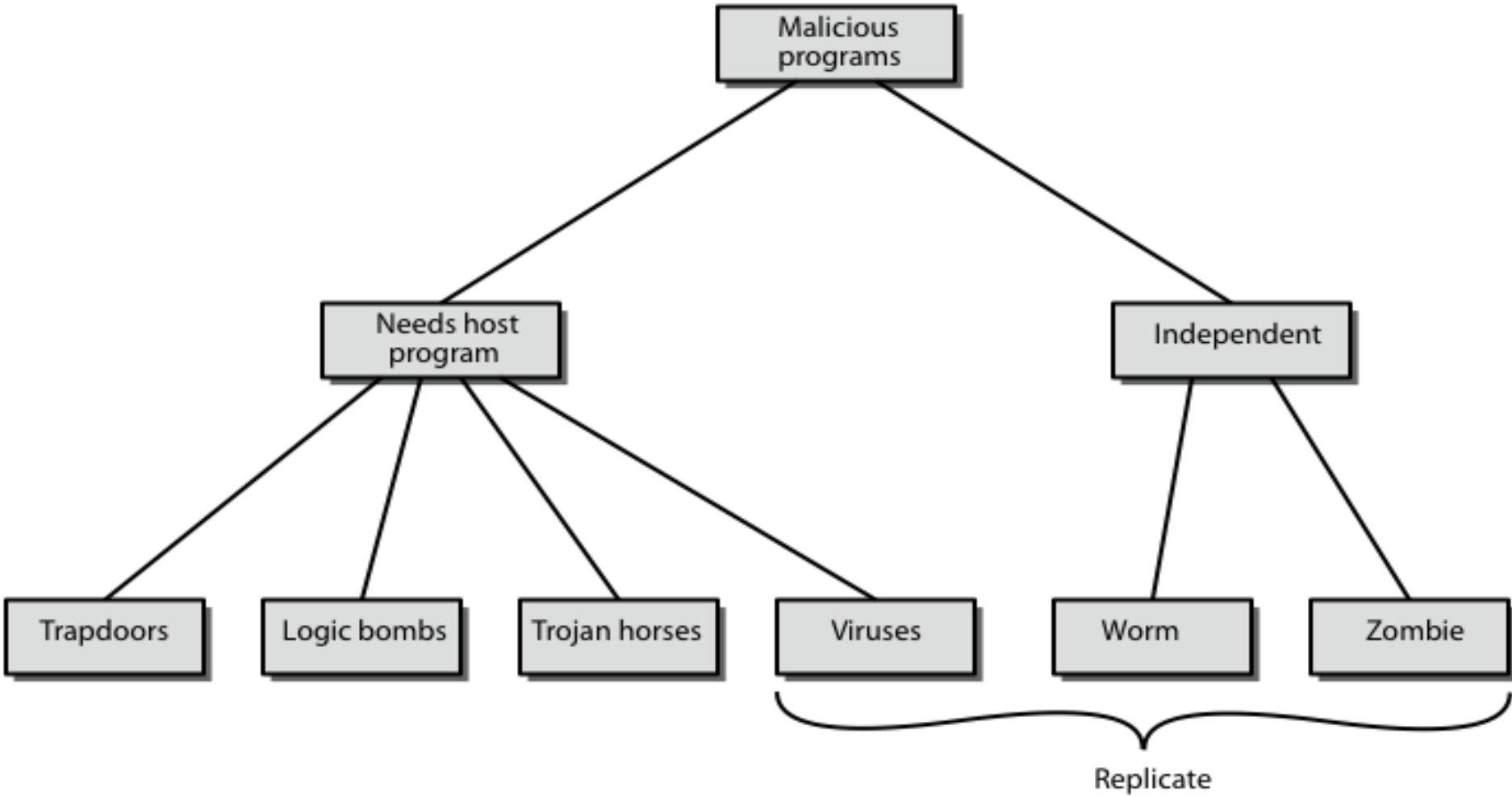
Outline

- Virus
- Worms
  - Modeling
  - Countermeasures
- Botnets
- Distributed denial-of-service (DDoS)

Viruses and Other Malicious Content

- Computer viruses have got a lot of publicity
- One of a family of **malicious software**
- Effects usually obvious
- Have figured in news reports, fiction, movies (often exaggerated)
- Getting more attention than deserve
- Are a concern though

Malicious Software



Backdoor or Trapdoor

- Secret entry point into a program
- Allows those who know access bypassing usual security procedures
- Have been commonly used by developers
- A threat when left in production programs allowing exploited by attackers
- Very hard to block in O/S
- Requires good s/w development & update

Logic Bomb

- One of oldest types of malicious software
- Code embedded in legitimate program
- Activated when specified conditions met
  - presence/absence of some file
  - particular date/time
  - particular user
- When triggered typically damage system
  - modify/delete files/disks, halt machine, etc

Trojan Horse

- Program with hidden side-effects
- Which is usually superficially attractive
  - e.g., game, s/w upgrade etc
- When run performs some additional tasks
  - Allows attacker to indirectly gain access they do not have directly
- Often used to propagate a virus/worm or install a backdoor
- Or simply to destroy data

Mobile Code

- Program/script/macro that runs unchanged
  - On heterogeneous collection of platforms
  - On large homogeneous collection (Windows)
- Transmitted from remote system to local system & then executed on local system
- Often to inject virus, worm, or Trojan horse or to perform own exploits
  - Unauthorized data access, root compromise
  - Popular vehicles for mobile code include Java applets, ActiveX, JavaScript, and VBScript

Multiple-Threat Malware

- Malware may operate in multiple ways
  - **Multipartite** virus infects in multiple ways
    - e.g., is capable of infecting multiple types of files, so that virus eradication must deal with all of the possible sites of infection.
- **Blended** attack uses multiple methods of infection or transmission
  - To maximize speed of contagion and severity
  - May include multiple types of malware
    - e.g., Nimda has worm, virus, mobile code
  - Can also use instant message and peer-to-peer file sharing

Viruses

- Piece of software that infects programs
  - Modifying them to include a copy of the virus
  - So it executes secretly when host program is run
- Specific to operating system and hardware
  - Taking advantage of their details and weaknesses
- Once a virus is executing, it can perform any function, such as erasing files and programs
- A typical virus goes through phases of:
  - Dormant
  - Propagation
  - Triggering
  - Execution

Viruses

- **Dormant phase:** The virus is idle. The virus will eventually be activated by some event, such as a date, the presence of another program or file, or the capacity of the disk exceeding some limit. Not all viruses have this stage.
- **Propagation phase:** The virus places an identical copy of itself into other programs or into certain system areas on the disk. Each infected program will now contain a clone of the virus, which will itself enter a propagation phase
- **Triggering phase:** The virus is activated to perform the function for which it was intended. As with the dormant phase, the triggering phase can be caused by a variety of system events, including a count of the number of times that this copy of the virus has made copies of itself
- **Execution phase:** The function is performed, which may be harmless, e.g. a message on the screen, or damaging, e.g. the destruction of programs and data files

Virus Structure

- Components:
  - Infection mechanism - enables replication
  - Trigger - event that makes payload activate
  - Payload - what it does, malicious or benign

- A virus can be prepended or postpended to an executable program, or it can be embedded in some other fashion. When infected program invoked, executes virus code then original program code\

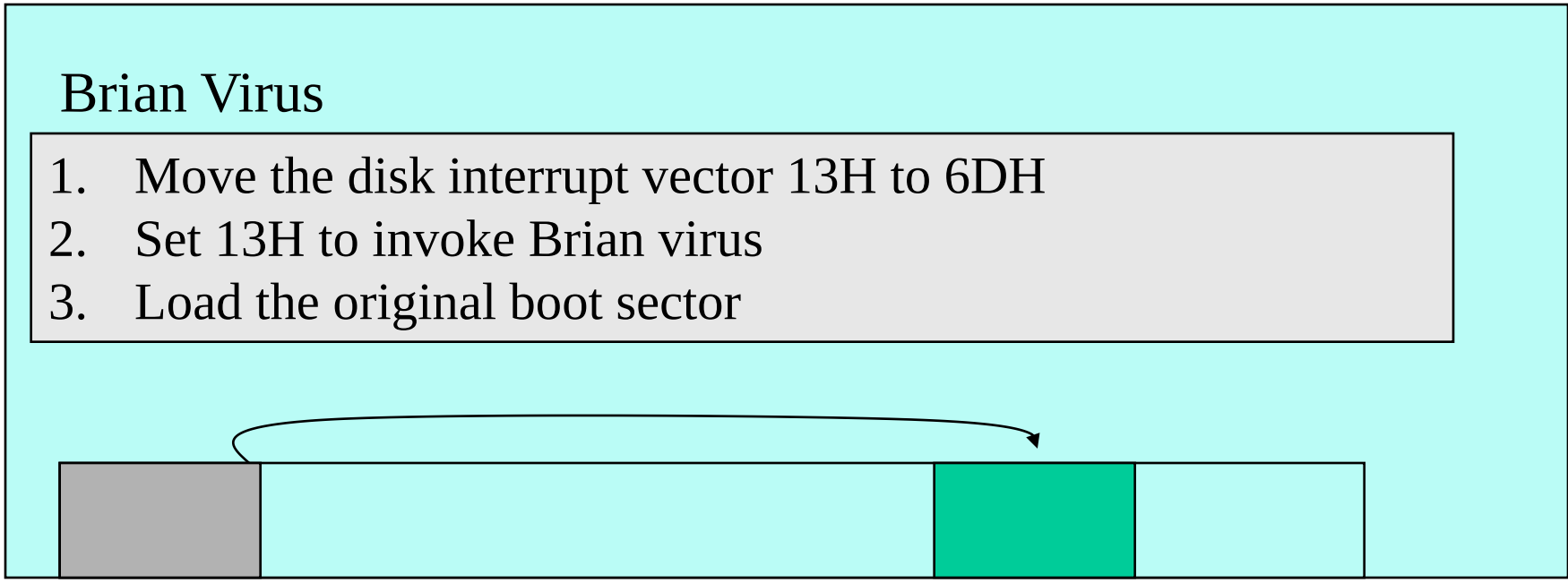
- Can block initial infection (difficult) or propagation (with access controls)

Computer Virus Classification

- **Boot sector infector:** Infects a master boot record or boot record and spreads when a system is booted from the disk containing the virus.
- File infector: Infects files that operating system or shell consider to be executable.
- **Macro virus:** Infects files with macro code that is interpreted by an application. A virus classification by concealment strategy includes the following categories:
- **Encrypted virus:** the virus creates a random encryption key, stored with the virus, and encrypts the remainder of the virus. When an infected program is invoked, the virus uses the stored random key to decrypt the virus. When the virus replicates, a different random key is selected.
- **Stealth virus:** A form of virus explicitly designed to hide itself from detection by antivirus software. Thus, the entire virus, not just a payload is hidden.
- Polymorphic virus: A virus that mutates with every infection, making detection by the “signature” of the virus impossible.
- **Metamorphic virus:** As with a polymorphic virus ,a metamorphic virus mutates with every infection. The difference is that a metamorphic virus rewrites itself completely at each iteration, increasing the difficulty of detection. Metamorphic viruses may change their behavior as well as their appearance.

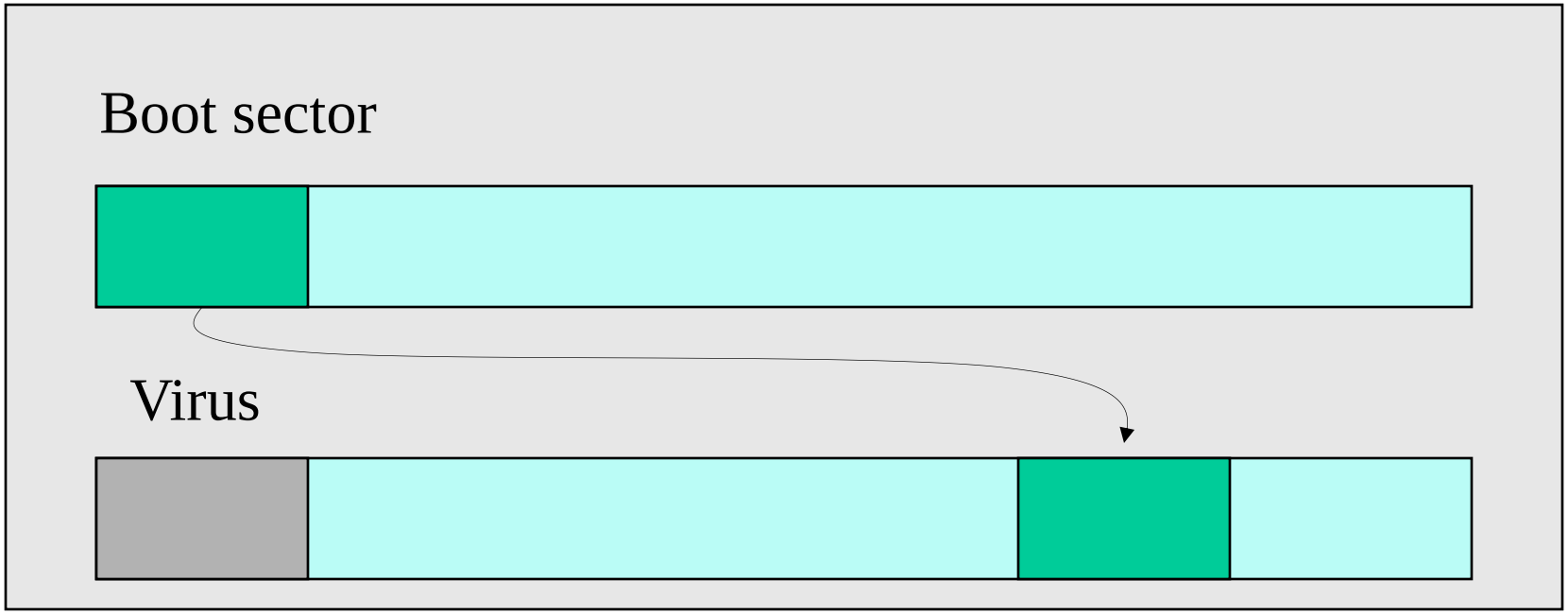
Computer Virus (Cont'd)

- Boot sector infectors
  - The boot sector is the part of a disk used to bootstrap the system
  - Code in a boot sector is executed when the system "sees" the disk for the first time



Boot Sector Infector (Cont.)

Infecting disks



1. Copy the old boot sector to alternative place;
2. Insert itself into the boot sector.

Macro Virus

- Became very common in mid-1990s since
  - Platform independent
  - Infect documents
  - Easily spread
- Viruses composed of instructions that are interpreted, rather than executed.
- Exploit macro capability of office apps
  - Executable program embedded in office doc
  - Often a form of Basic
- More recent releases include protection
- Recognized by many anti-virus programs

E-Mail Viruses

- More recent development (e.g., Melissa)
  - Exploits MS Word macro in attached doc
  - If attachment opened, macro activates
  - Sends email to all on users address list and does local damage
- Then saw versions triggered reading email hence much faster propagation

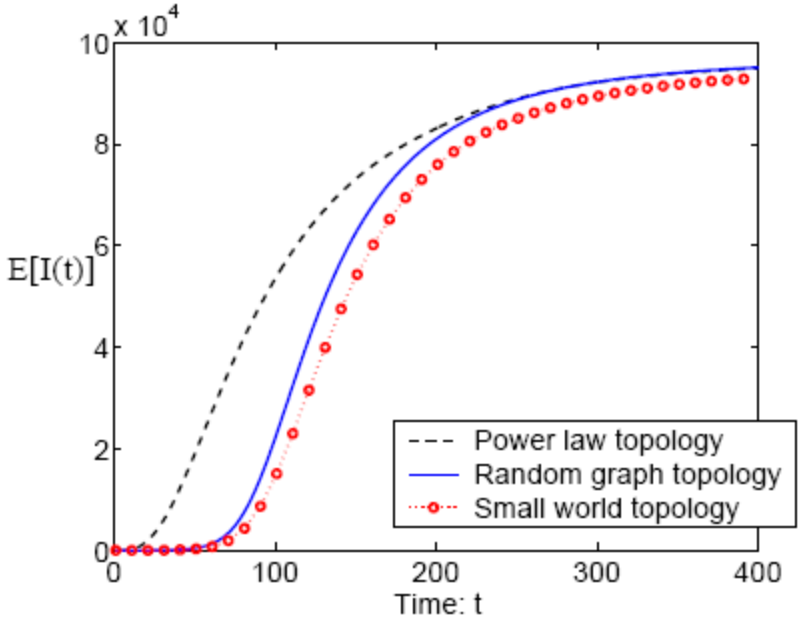
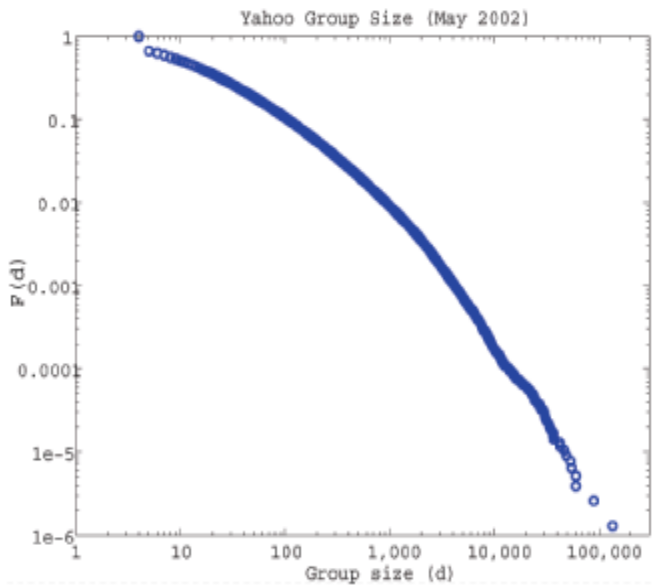


Fig. 9. Effect of topology on email worm propagation



Complementary cumulative distr. of Yahoo! group size (in May 2002)

Viruse Variation

- Stealth viruses
  - Conceal the infection of files
  - Make itself difficult to detect
- Polymorphic viruses
  - Encrypt itself with a random key
  - Avoid detection by anti-virus programs, which search for patterns of viruses.
- Metamorphic viruses
  - Change its form each time it inserts itself into another program.

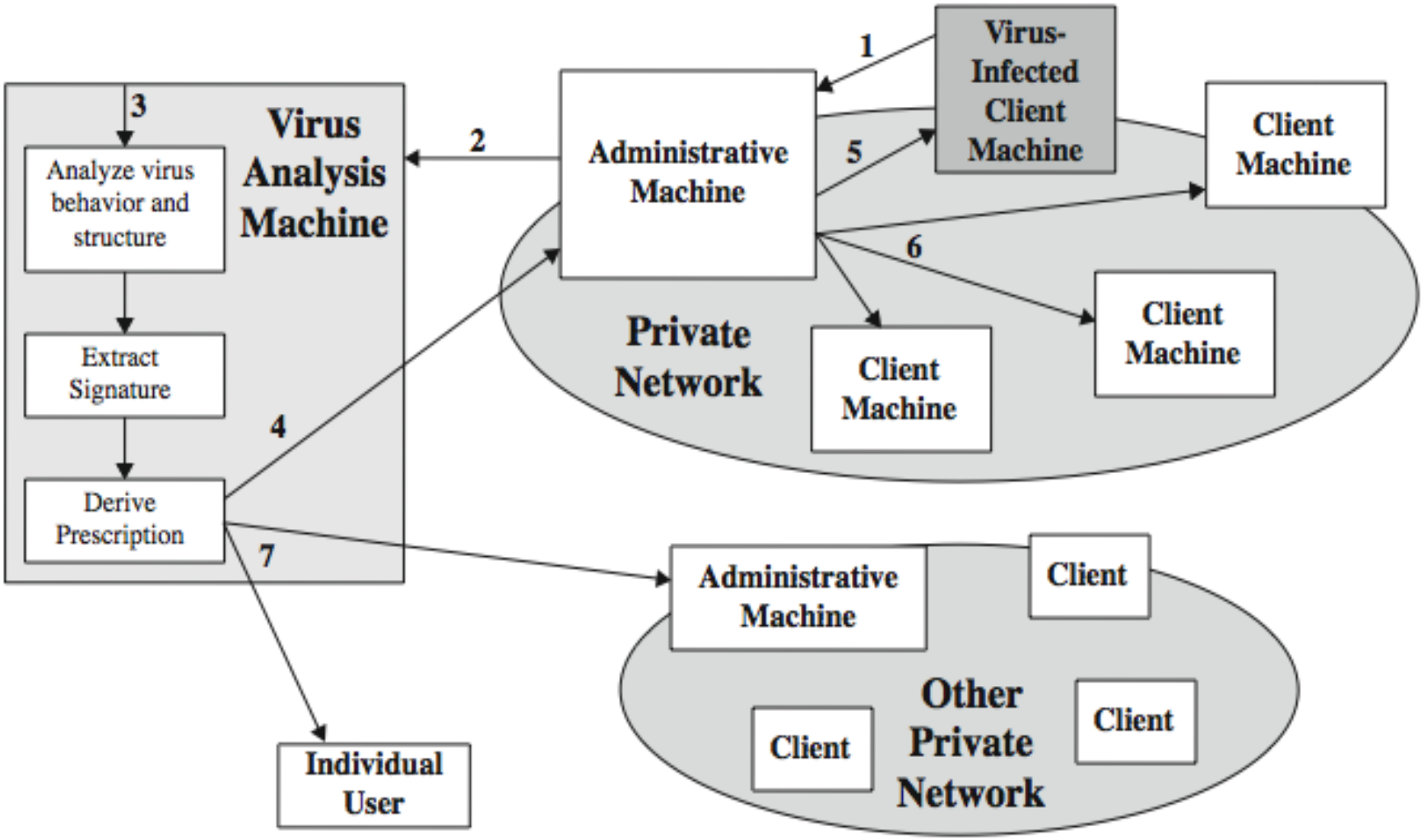
Countermeasures

- The ideal solution to the threat of viruses is prevention:
  - Do not allow a virus to get into the system in the first place. This goal is, in general, impossible to achieve
- Realistically need
  - **Detection:** Once the infection has occurred, determine that it has occurred and locate the virus.
  - **Identification:** Once detection has been achieved, identify the specific virus that has infected a program.
  - **Removal:** Once the specific virus has been identified, remove all traces of the virus from the infected program and restore it to its original state. Remove the virus from all infected systems so that the disease cannot spread further.
- If detect but can't identify or remove, must discard and replace infected program

Anti-Virus Evolution Path

- Virus & antivirus tech have both evolved early viruses simple code, easily removed as become more complex, so must the countermeasures
- A **first-generation** scanner requires a virus signature to identify a virus. The virus may contain "wildcards" but has essentially the same structure and bit pattern in all copies. Such signature-specific scanners are limited to the detection of known viruses.
- A **second-generation** scanner uses heuristic rules to search for probable virus infection, e.g., to look for fragments of code that are often associated with viruses. Another second-generation approach is integrity checking, using a hash function rather than a simpler checksum.
- **Third-generation** programs are memory-resident programs that identify a virus by its actions rather than structure in an infected program. These have the advantage that it is not necessary to develop signatures / heuristics, but only to identify the small set of actions indicating an infection is attempted and then intervene.
- **Fourth-generation** products are packages consisting of a variety of antivirus techniques used in conjunction. These include scanning and activity trap components. In addition, such a package includes access control capability, which limits the ability of viruses to penetrate a system and then limits the ability of a virus to update files in order to pass on the infection.

Example: Digital Immune System



Digital Immune System

- The digital immune system is a comprehensive approach to virus protection developed by IBM and subsequently refined by **Symantec**.

- **Detailed Steps:**

  1. A monitoring program on each PC uses a variety of heuristics to infer that a virus may be present, and forwards a copy to an administrative machine.

  2. Admin machine encrypts this and sends it to a central virus analysis machine.

  3. This machine creates an environment in which the infected program can be safely run for analysis. The virus analysis machine then produces a prescription for identifying and removing the virus.

  4. The resulting prescription is sent back to the administrative machine.

  5. The administrative machine forwards the prescription to the infected client.

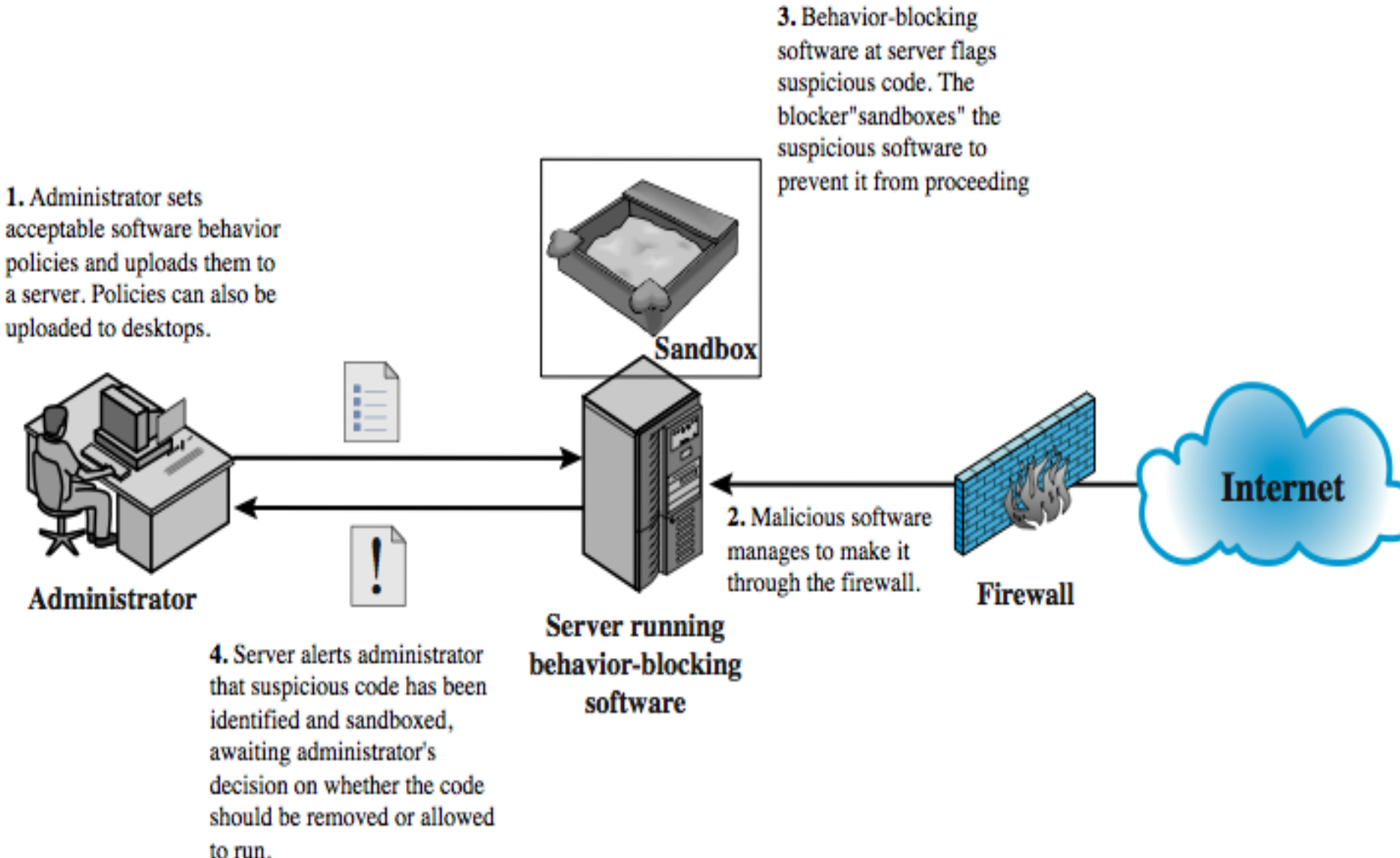
  6. The prescription is also forwarded to other clients in the organization.

  7. Subscribers worldwide get regular antivirus updates to protect from new virus

Behavior Blocking Software

- Unlike heuristics or fingerprint-based scanners, behavior-blocking software integrates with the operating system of a host computer and monitors program behavior in real-time for malicious actions.
- The behavior blocking software then blocks potentially malicious actions before they can affect the system.
- Monitored behaviors can include
  - Attempts to open, view, delete, and/or modify files;
  - Attempts to format disk drives and other unrecoverable disk operations;
  - Modifications to the logic of executable files or macros;
  - Modification of critical system settings, such as start-up settings;
  - Scripting of e-mail and instant messaging clients to send executable content; and
  - Initiation of network communications.

Behavior-Blocking Software



Behavior Blocking Software

- Behavior-blocking software runs on server and desktop computers and is instructed through policies set by the network administrator to let benign actions take place but to intercede when unauthorized or suspicious actions occur.
- The module blocks any suspicious software from executing. A blocker isolates the code in a sandbox, which restricts the code's access to various OS resources and applications.
- The blocker then sends an alert. Because behavior blocker can block suspicious software in real-time, it has an advantage over such established antivirus detection techniques as fingerprinting or heuristics.
- Behavior blocking alone has limitations. Because the malicious code must run on the target machine before all its behaviors can be identified, it can cause harm before it has been detected and blocked.

Worm

- A computer worm is a program that copies itself from one computer to another.
- Replicating program that propagates over computer network
  - Using email, remote exec, remote login
- Has phases like a virus:
  - Dormant, propagation, triggering, execution
  - Propagation phase:
    - Searches for other systems to infect by examining host tables or similar repositories of remote system addresses
    - Establishes a connection with a remote system; and copies itself to the remote system and cause the copy to be run.
- May disguise itself as a system process

Worms (Cont.)

- Different from viruses
  - Viruses depend on other programs
  - Worms are usually standalone applications
  - Viruses usually trick people into propagating them
  - Worms can hack into vulnerable systems and spread without depending on others

A few known examples

Morris Worm

- One of best know worms
- Released by Robert Morris in 1988
- The Morris worm was designed to spread on UNIX systems and used a number of different techniques for propagation.
- Various attacks on UNIX systems
  - Cracking password file to use login/password to logon to other systems
  - Exploiting a bug in the finger protocol
  - Exploiting a bug in sendmail
- If succeed have remote shell access
  - Sent bootstrap program to copy worm over

The Slammer Worm

- Facts about Slammer
  - Happened slightly before 5:30 UTC on Saturday, January 25, 2003.
  - The fastest worm in history
  - Doubled in size every 8.5 seconds at the beginning
  - Infected more than 90% of vulnerable hosts within 10 minutes

The Slammer Worm (Cont.)

- How does it find vulnerable computers?
  - Random scanning
    - Select IP addresses at random to infect
- How does it get into vulnerable computers?
  - Exploit a buffer overflow vulnerability in MS SQL Server or MSDE 2000
    - Vulnerability discovered in July 2002
- Why was it so fast?
  - Small: 376 bytes; a 404 byte UDP packet
  - Based on UDP

Other Known Worms

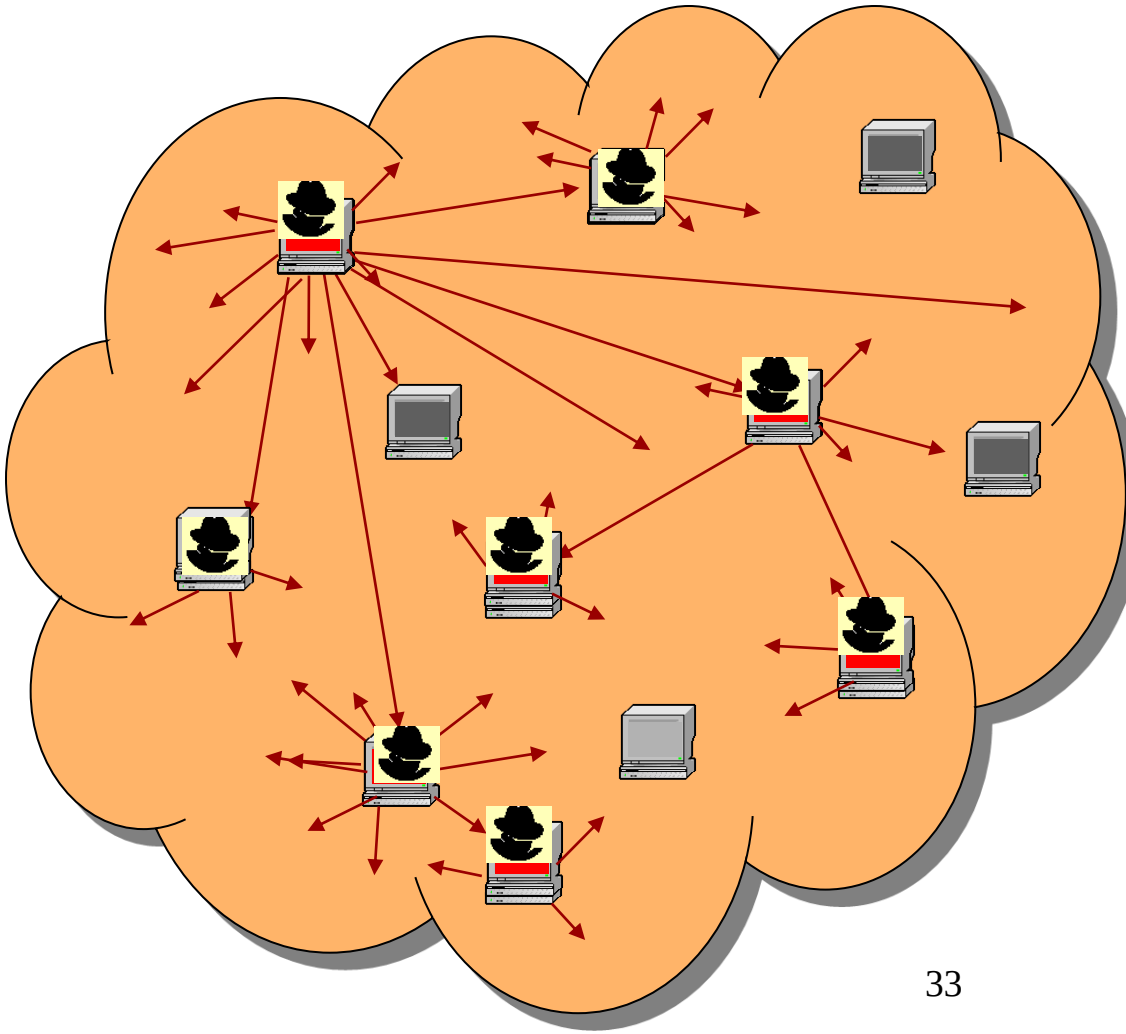
- Code Red
  - July 2001 exploiting MS IIS bug
  - probes random IP address, does DDoS attack
- Code Red II variant includes backdoor
- SQL Slammer
  - Early 2003, attacks MS SQL Server
- Mydoom
  - Mass-mailing e-mail worm that appeared in 2004
  - Installed remote access backdoor in infected systems
- Warezov family of worms
  - Scan for e-mail addresses, send in attachment

Consequences of Worms

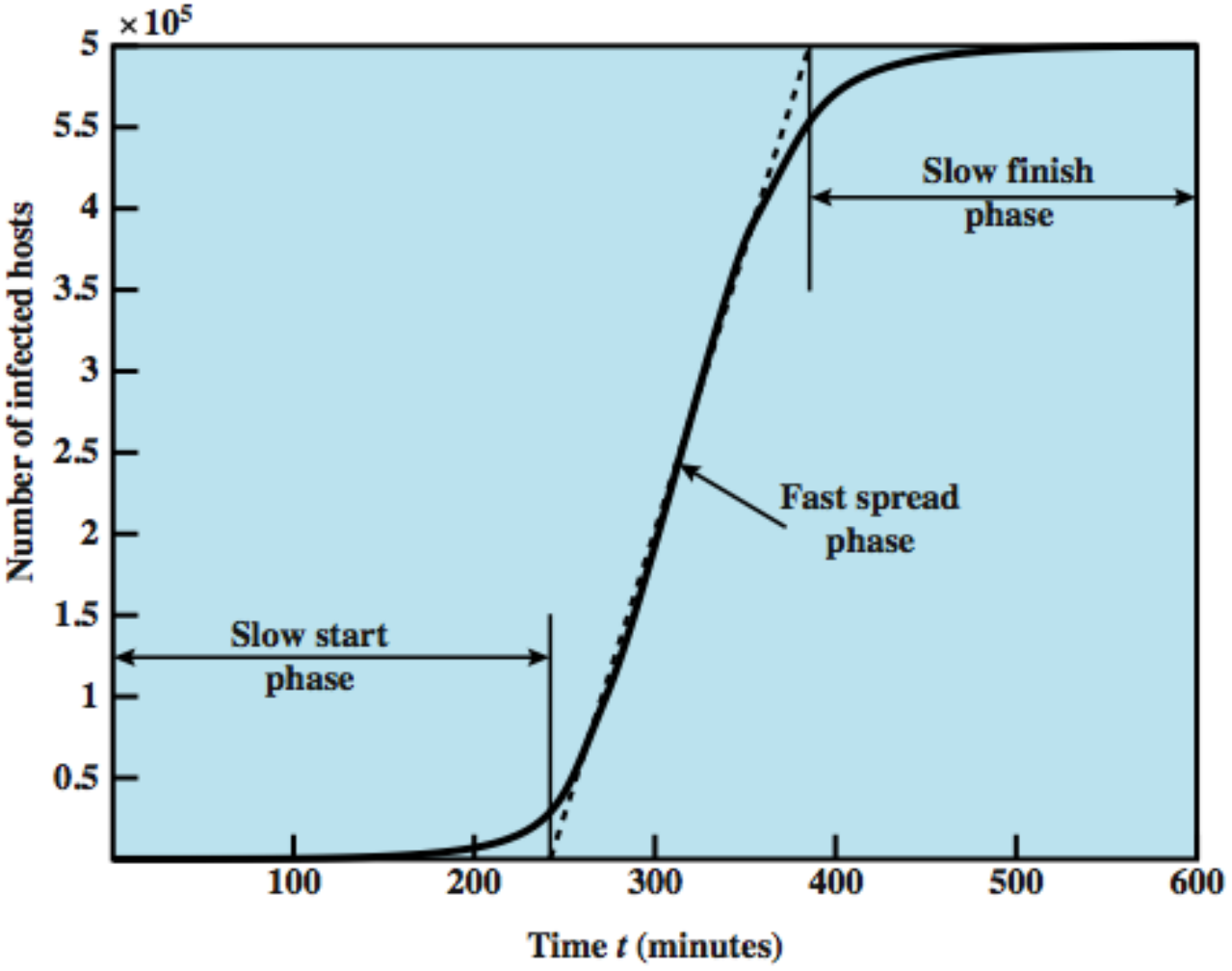
- Worm attacks have recently posed major threats to the Internet
  - Code-Red I, Code Red II (TCP 80, Windows Internet Information Service)
    - Infected more than 350,000 computers in July, 2001, within 14 hours
  - SQL Slammer (UDP 1434, SQL server)
    - Infected more than 75,000 computers in January, 2003, in 10 minutes
- Used as a utility to launch subsequent attacks
  - Denial-of-service attack, confidential information access, email spam, and malware spreading

Propagation Traffic Generated by Worm Attack

- Worm propagates itself to other computers
- Generate propagation traffic
  - Messages that intend to identify vulnerable computers



Worm Propagation Model

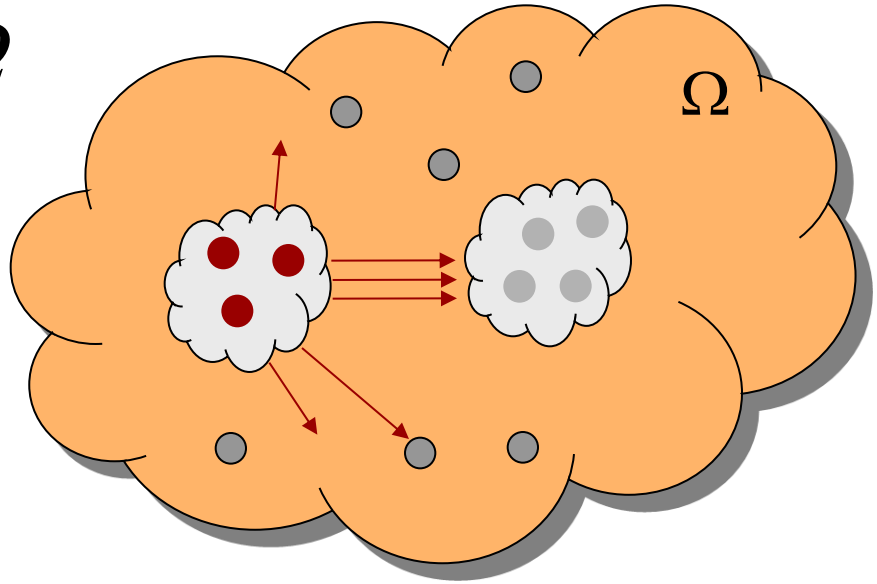


Worm Propagation Model

- **Slow start phase**
  - The number of hosts increases exponentially
  - Consider a simplified case in which a worm is launched from a single host and infects two nearby hosts. Each of these hosts infects two more hosts, and so on. This results in exponential growth.
- **Fast spread phase**
  - After a time, infecting hosts waste some time attacking already-infected hosts, which reduces the rate of infection
  - During this middle phase, growth is approximately linear, but the rate of infection is rapid
- **Slow finish phase**
  - When most vulnerable computers have been infected, the attack enters a slow finish phase as the worm seeks out those remaining hosts that are difficult to identify.
  - Clearly, the objective in countering a worm is to catch the worm in its slow start phase, at a time when few hosts have been infected.

Worm Propagation Model (continuous form)

- Address space, size Ω
- N : total vulnerable
- I_t : infected by time t
  - N-I_t vulnerable at time t
- scan rate (per host), η



$$\propto \eta \cdot I_t \cdot \underbrace{\frac{N - I_t}{\Omega}}$$

Prob. of a scan hitting vulnerable

$$\underbrace{\frac{dI_t}{dt}} = \frac{\eta}{\Omega} I_t (N - I_t)$$

# of increased infected in a unit time

Worm Propagation Model (Discrete Form)

Parameters	Notation	Explanation
# of vulnerable machines	N	the number of vulnerable machines
Size of hitlist	h	the number of infected machines at the beginning of the spread of active worms
Scanning rate	s	the average number of machines scanned by an infected machine per unit time
Death rate	d	the rate at which an infection is detected on a machine and eliminated without patching
Patching rate	p	the rate at which an infected or vulnerable machine becomes invulnerable

Although the Internet's address space isn't completely connected, active worms always scan 2^{32} entry addresses. Therefore, for random scanning, the probability that any computer is hit by one scan is $\frac{1}{2^{32}}$. Let m_i and n_i denote the total number of vulnerable machines (including the infected ones) and the number of infected machines at time tick i ($i \geq 0$) respectively. Before the active worms spread ($i = 0$), $m_0 = N$ and $n_0 = h$.

Theorem 1: If there are m_i vulnerable machines (including the infected ones), and n_i infected computers, then on average, the next time tick will have $(m_i - n_i)[1 - (1 - \frac{1}{2^{32}})^{sn_i}]$ newly infected machines, where s is the scanning rate.

Chen et al, "Modeling the Spread of Active Worms", in Proceedings of INFOCOM 2002
http://www.ieee-infocom.org/2003/papers/46_03.pdf

Worm Propagation Model (Discrete Form)

PROOF: Let e_i denote the number of newly infected machines at time tick i $(i \geq 0)$. n_i infected machines can generate sn_i scans in an attempt to infect other machines. So if we can prove that $E\{e_{i+1}/k\} = (m_i - n_i)[1-(1-\frac{1}{2^{32}})^k]$ for any k $(k > 0)$ scans, then the equation also holds when $k = sn_i$.

We prove the above equation by induction on k. When $k = 1$, since there are $(m_i - n_i)$ vulnerable machines that have not yet been infected, the probability that one scan can add a newly infected machine is $\frac{m_i - n_i}{2^{32}}$, which is equivalent to $(m_i - n_i)[1-(1-\frac{1}{2^{32}})^1]$. Suppose that the theorem is true for $k = j$, i.e., $E\{e_{i+1}/k = j\} = (m_i - n_i)[1-(1-\frac{1}{2^{32}})^j]$. Then, when $k = j+1$, we divide $j+1$ scans into two parts: the first j scans and the last scan. There are two possibilities for the last scan: adding a newly infected machine or not. Let the variable $Y = 1$ if the last scan hits a vulnerable machine that has not yet been infected and let $Y = 0$ otherwise. Then,

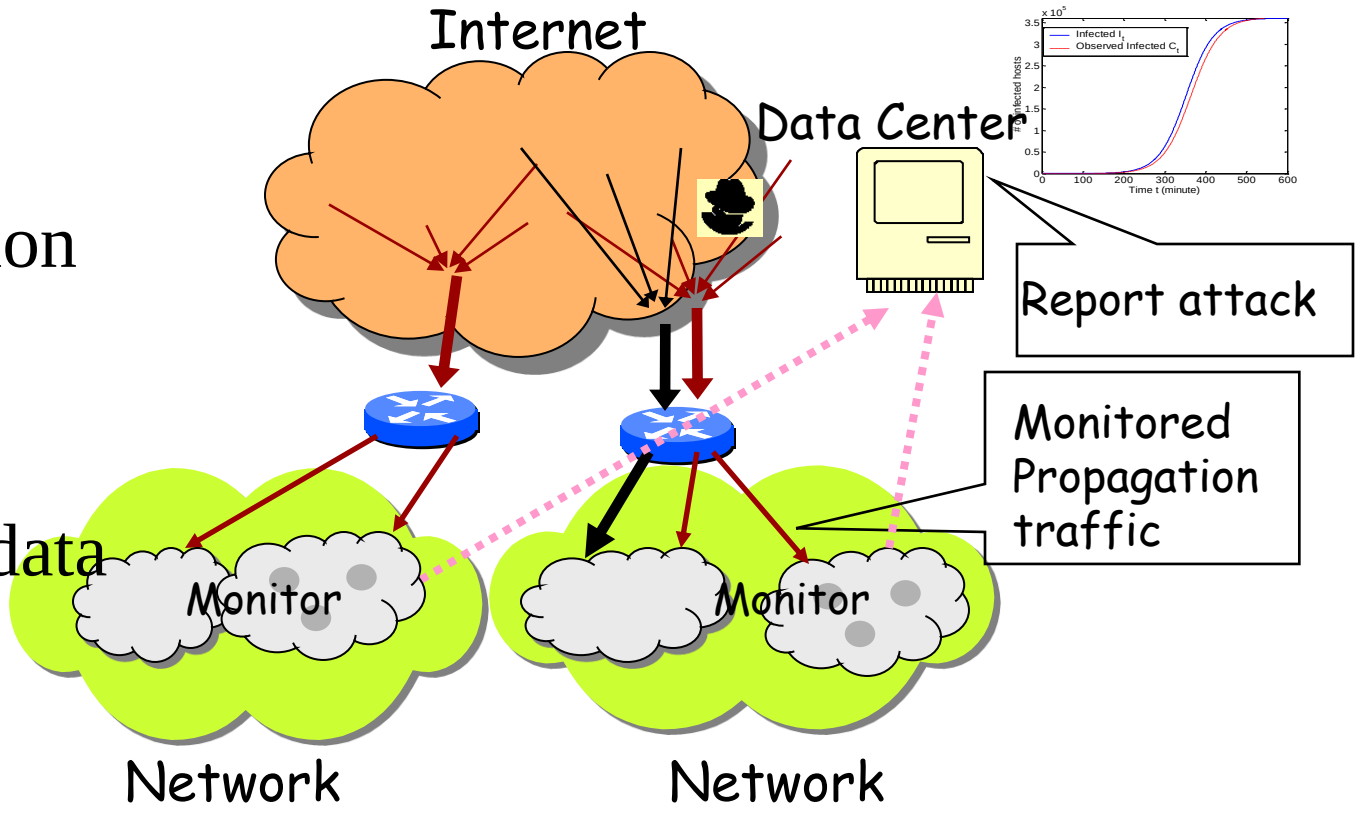
$$
\begin{aligned}
& E\{e_{i+1}/k = j+1\} \\
= \; & (E\{e_{i+1}/k = j\} + 1)P(Y = 1) + E\{e_{i+1}/k = j\} \cdot \\
& P(Y = 0) \\
= \; & (E\{e_{i+1}/k = j\} + 1)\frac{m_i - n_i - E\{e_{i+1}/k = j\}}{2^{32}} + \\
& E\{e_{i+1}/k = j\}(1 - \frac{m_i - n_i - E\{e_{i+1}/k = j\}}{2^{32}}) \\
= \; & \frac{m_i - n_i}{2^{32}} + (1 - \frac{1}{2^{32}})E\{e_{i+1}/k = j\} \\
= \; & (m_i - n_i)[1-(1-\frac{1}{2^{32}})^{j+1}]
\end{aligned}
$$

which means that when $k = j+1$, it is also true. Therefore, when $k = sn_i$, $E\{e_{i+1}/k = sn_i\} = (m_i - n_i)[1-(1-\frac{1}{2^{32}})^{sn_i}]$. That is, on the next time tick there will be $(m_i - n_i)[1-(1-\frac{1}{2^{32}})^{sn_i}]$ expected newly infected machines. ■

Chen et al, "Modeling the Spread of Active Worms", in Proceedings of INFOCOM 2002
http://www.ieee-infocom.org/2003/papers/46_03.pdf

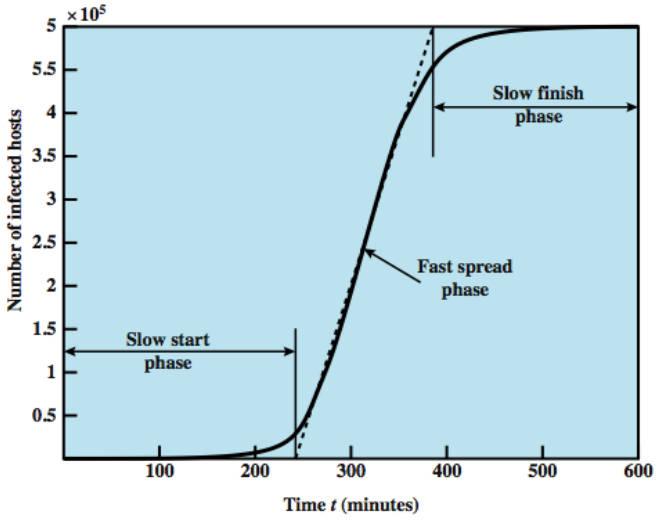
Monitoring Worm Propagation

- Monitors:
  - Worm propagation traffic
- Data center:
  - Collects data
  - Reports attack



Worm Propagation Model vs. Detection

$$\frac{dI_t}{dt} = \frac{\eta}{\Omega} I_t(N - I_t)$$



- Signature-based vs. Anomaly-based
  - Examples???

Worm Detection Strategies

- **Signature-based detection**
  - Exponential growth trend in slow start phase
    - Cliff C. Zou, Weibo Gong, Don Towsley, and Lixin Gao. The Monitoring and Early Detection of Internet Worms," IEEE/ACM Transactions on Networking, 13(5), p.961-974, October 2005.
- **Anomaly-based detection**
  - Large variance associated with propagation traffic
    - J. Wu, S. Vangala, L. Gao, and K. Kwiat, An Effective Architecture and Algorithm for Detecting Worms with Various Scan Techniques, in Proceedings of NDSS 2004
  - Propagation traffic rate volume (rate is high)
    - S. Venkataraman, D. Song, P. Gibbons, and A. Blum, “New Streaming Algorithms for SuperSpreader Detection,” Proceeding of NDSS 2005.
  - Source/ Destination Address Distribution
    - A. Lakhina, M. Crovella, and C. Diot, “Mining Anomalies Using Traffic Feature Distribution,” Proc. ACM SIGCOMM, Aug. 2005.

Propagation Model Plus Defense

$$\frac{dI_t}{dt} = \frac{\eta}{\Omega} I_t (N - I_t) \underbrace{- r * I_t}_{\text{Defense impact}}$$

Question 1: when we increase *r*, what is *It*

Worm Strategies

- The state of the art in worm strategies includes the following:
  - Multiplatform: Newer worms are not limited to Windows machines but can attack a variety of platforms, especially the popular varieties of UNIX.
  - Multi-exploit: New worms penetrate systems in a variety of ways, using exploits against Web servers, browsers, e-mail, file sharing, and other network-based applications.
  - Ultrafast spreading: One technique to accelerate the spread of a worm is to conduct a prior Internet scan to accumulate Internet addresses of vulnerable machines.
  - Polymorphic: To evade detection, skip past filters, and foil real-time analysis, worms adopt the virus polymorphic technique. Each copy of the worm has new code generated on the fly using functionally equivalent instructions and encryption techniques.

Worm Strategies

- The state of the art in worm strategies includes the following:
  - ….
  - Metamorphic: In addition to changing their appearance, metamorphic worms have a repertoire of behavior patterns that are unleashed at different stages of propagation.
  - Transport vehicles: Because worms can rapidly compromise a large number of systems, they are ideal for spreading other distributed attack tools, such as distributed denial of service bots.
  - Zero-day exploit: To achieve maximum surprise and distribution, a worm should exploit an unknown vulnerability that is only discovered by the general network community when the worm is launched.

Research on Worm Progagation

- **See Stuart Staniford, et al., "How to Own the Internet in Your Spare Time"**

Mobile Phone Worms

- First appeared on mobile phones in 2004
  - Communicate through Bluetooth wireless connections or via the multimedia messaging service (MMS).
  - Target smartphone which can install s/w
- They communicate via Bluetooth or MMS to disable phone, delete data on phone, or send premium-priced messages
- CommWarrior, launched in 2005
  - Replicates using Bluetooth to nearby phones and via MMS using address-book numbers
  - It copies itself to the removable memory card and inserts itself into the program installation files on the phone.
- **Today's smart phone**
  - **Becoming a new active research area!**

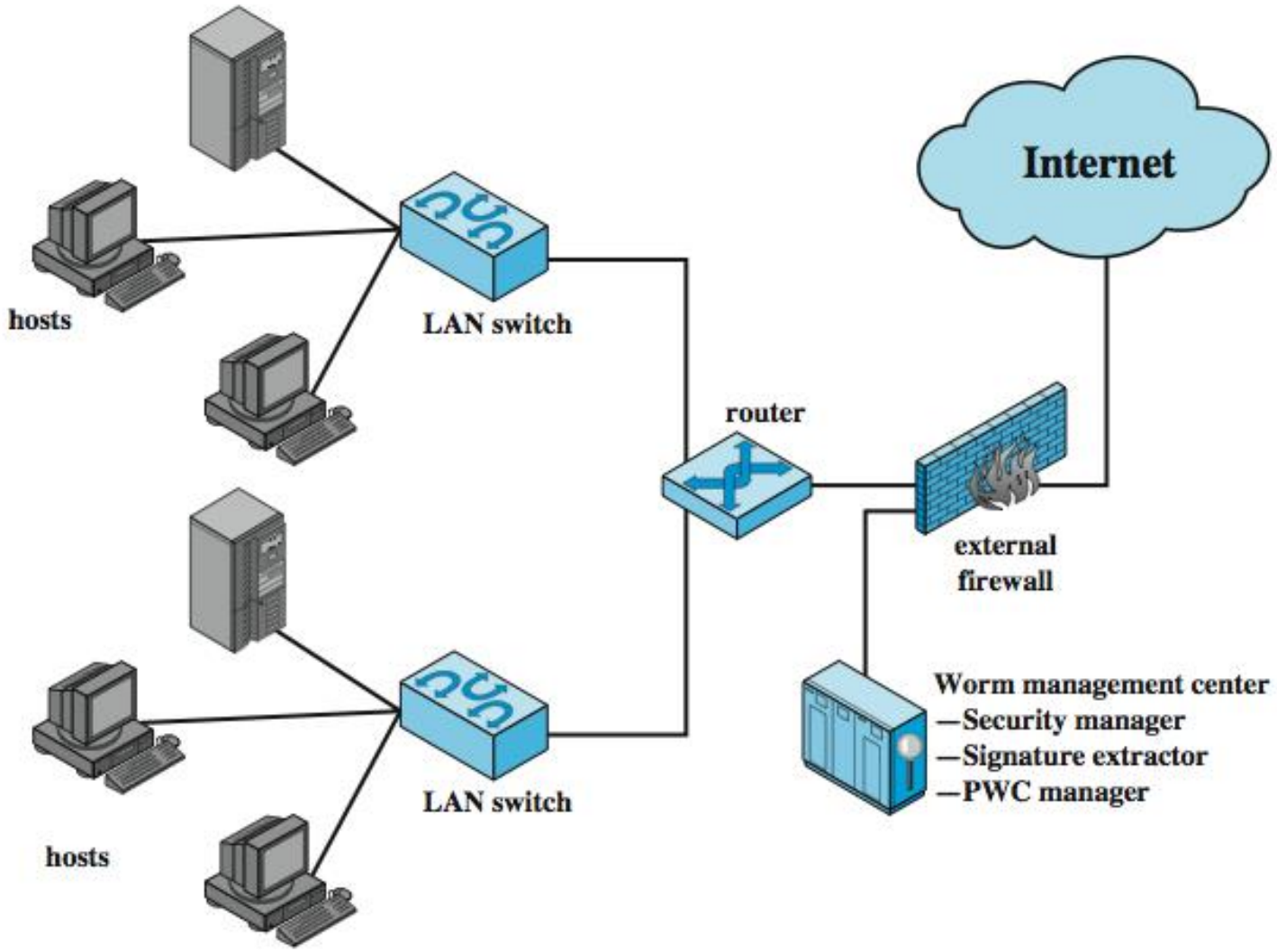
Worm Countermeasures

- Worm defense approaches include:
  - Signature-based worm scan filtering: Generates a worm signature, which is then used to prevent worm scans from entering/leaving a network/host.
  - Filter-based worm containment: Focuses on worm content rather than a scan signature. The filter checks a message to determine if it contains worm code.
  - Payload-classification-based worm containment: Examine packets to see if they contain a worm using anomaly detection techniques
  - Threshold random walk (TRW) scan detection: Exploits randomness in picking destinations to connect to as a way of detecting if a scanner is in operation

Worm Countermeasures

- Worm defense approaches include:
  - Rate limiting: Limits the rate of scan like traffic from an infected host.
  - Rate halting: Immediately blocks outgoing traffic when a threshold is exceeded either in outgoing connection rate or diversity of connection attempts.
    - Rate halting can integrate with a signature- or filter-based approach so that once a signature or filter is generated, every blocked host can be unblocked; as with rate limiting, rate halting techniques are not suitable for slow, stealthy worms.

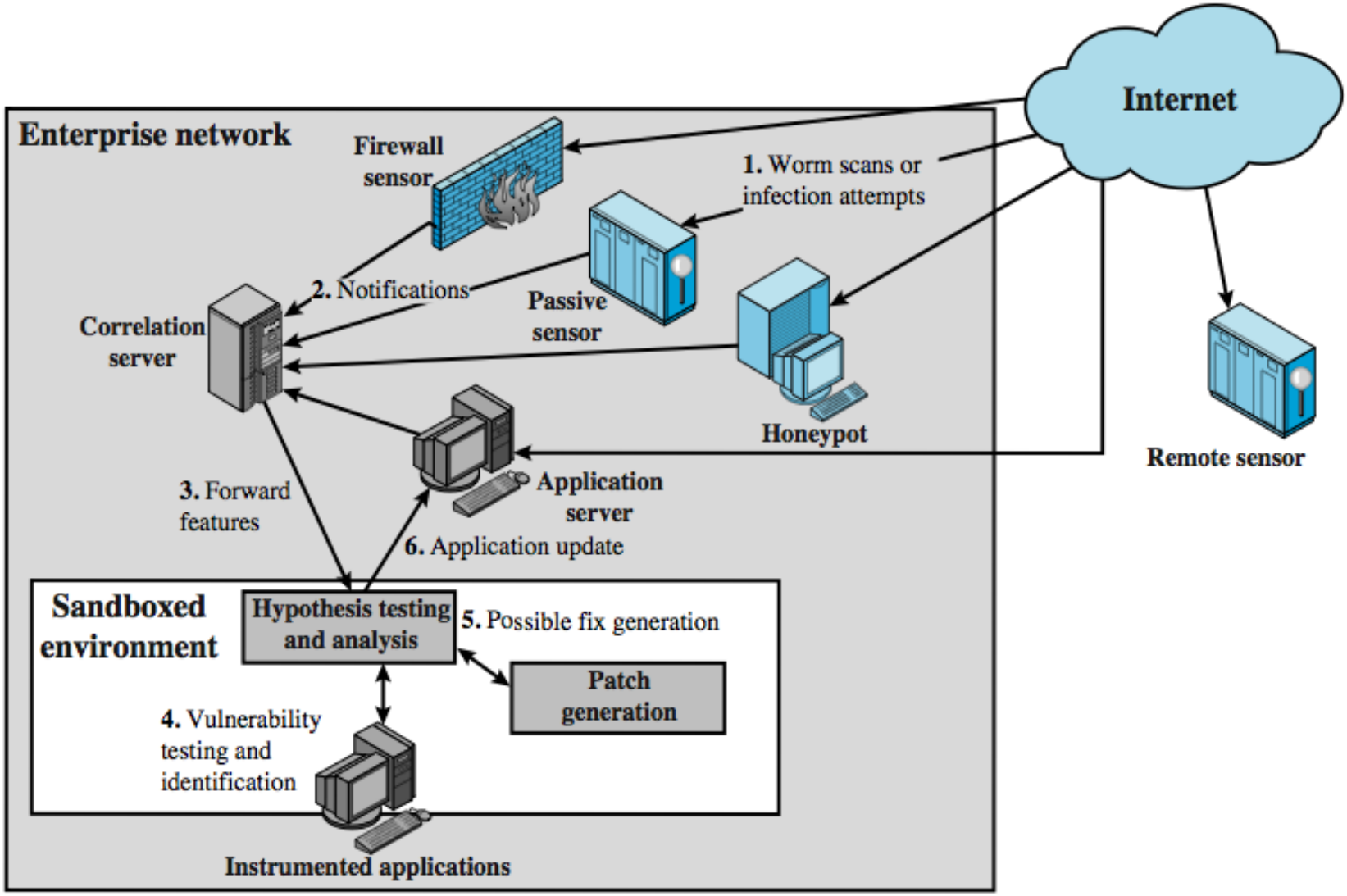
Proactive Worm Containment



Proactive Worm Containment

- PWC agent monitors outgoing traffic for scan activity, determined by a surge in UDP / TCP connection attempts to remote hosts. If a surge is detected, the agent: 1) issues an alert to local system; 2) blocks all outgoing connection attempts; 3) transmits the alert to the PWC manager; and 4) starts a relaxation analysis.
- A PWC manager receives an alert, and propagates the alert to all other agents.
- The host receives an alert, and must decide whether to ignore the alert. If agent assumes that it might be infected and performs the following actions: (1) blocks all outgoing connection attempts from the specific alerting port; and (2) starts a relaxation analysis.
- Relaxation analysis. An agent monitors outgoing activity for a fixed window of time to see if outgoing connections exceed a threshold. If so, blockage is continued and relaxation analysis is repeated until the outgoing connection rate drops below the threshold, at which time the agent removes the block. If the threshold continues to be exceeded over a sufficient number of relaxation windows, the agent isolates the host and reports to the PWC manager.

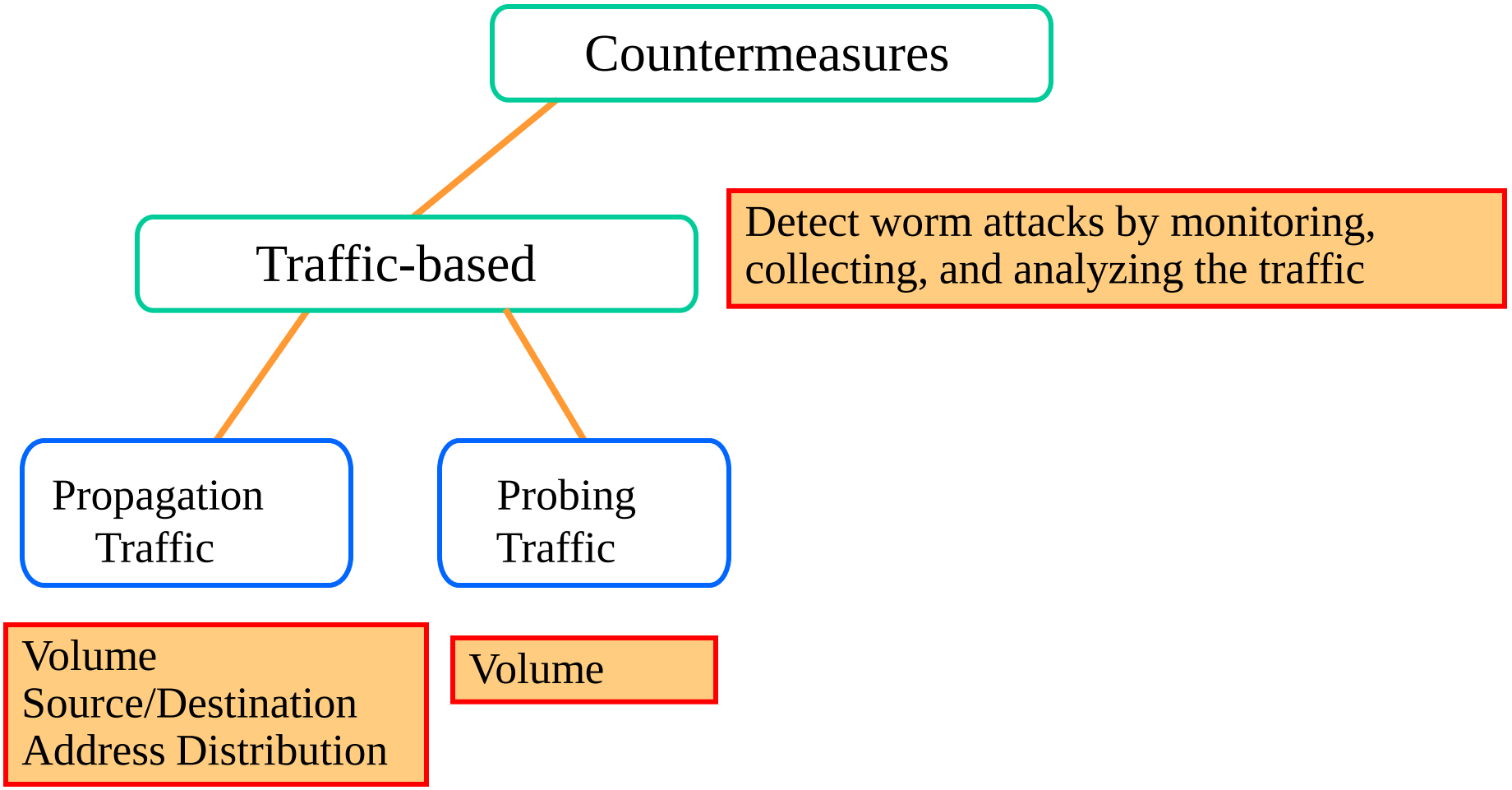
Network Based Worm Defense



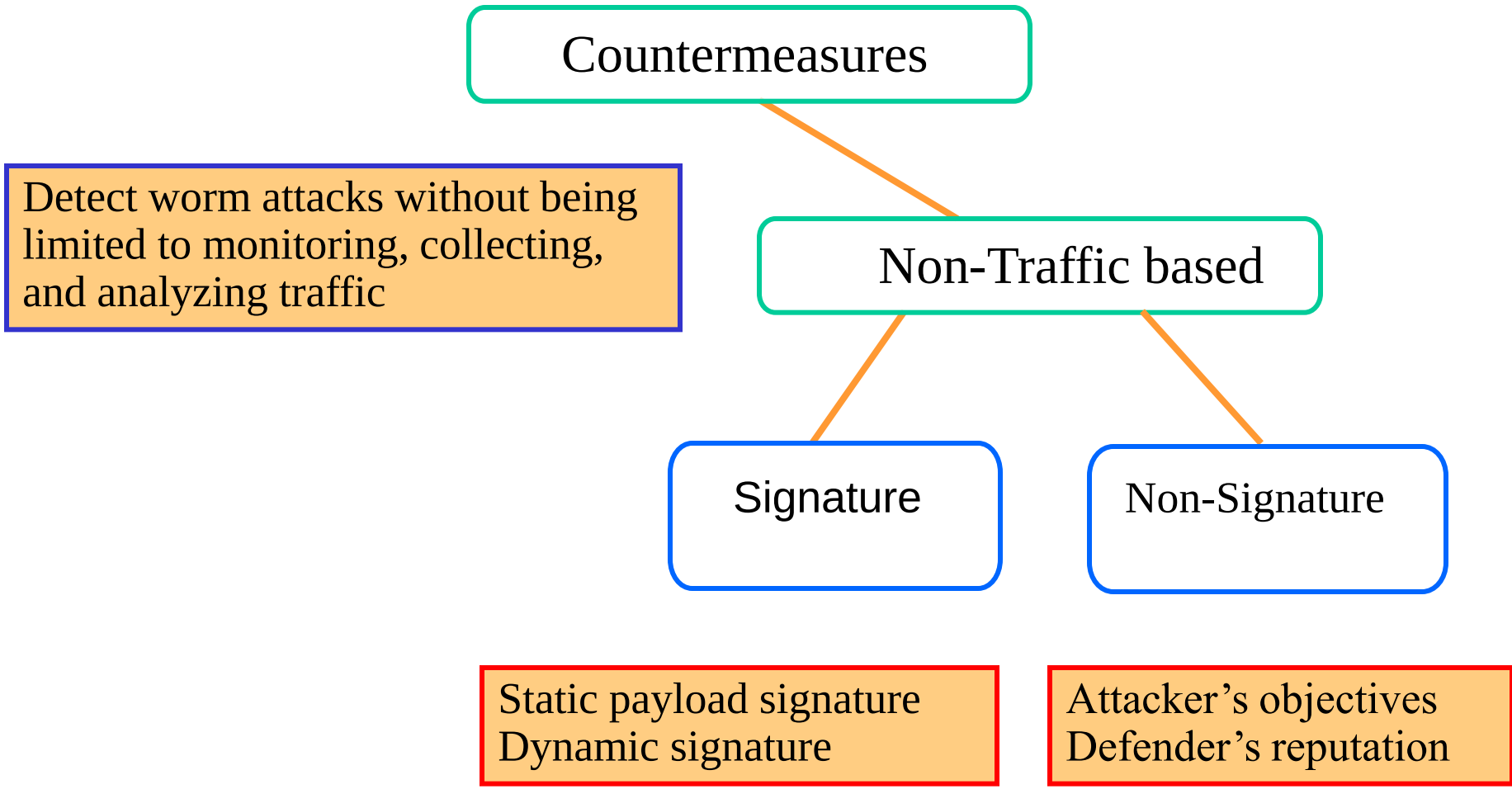
Network Based Worm Defense

- Ingress monitors: located at the border between the enterprise network and the Internet, in a border router, external firewall, separate passive monitor, or honeypot.
- Egress monitors: located at the egress point of individual LANs on the enterprise network as well as at the external border, in a LAN router or switch, external firewall or honeypot.
- The two types of monitors can be collocated. It is designed to catch the source of a worm attack by monitoring outgoing traffic for signs of scanning etc.
- Basic Workflow of Worm Defense
  1. Sensors deployed at various network locations detect a potential worm and send alerts to a central server that correlates / analyzes incoming alerts.
  2. forwards info to a protected environment, where worm is sandboxed for analysis
  4. protected system tests the suspicious software against an appropriately instrumented version of the targeted application to identify the vulnerability.
  5. protected system generates one or more software patches and tests these.
  6. system sends the patch to the application host to update the targeted application.

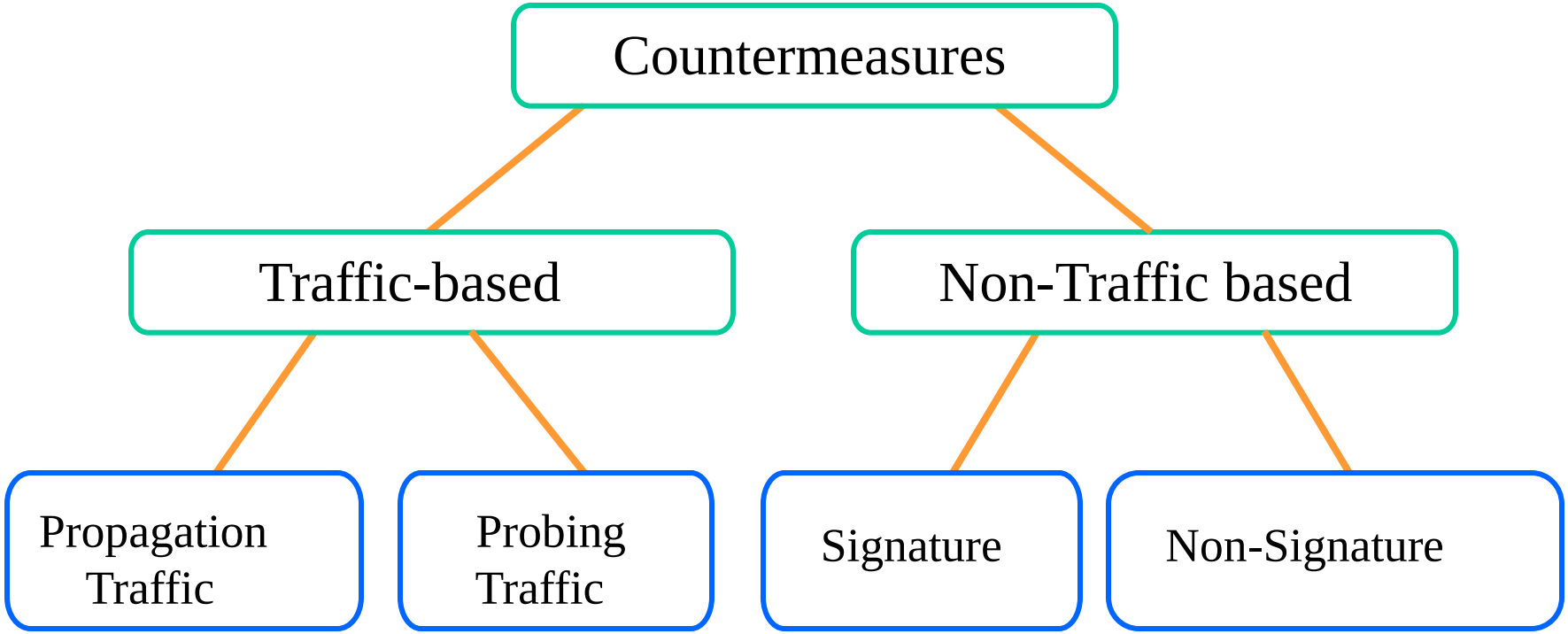
Classification of Worm Detection



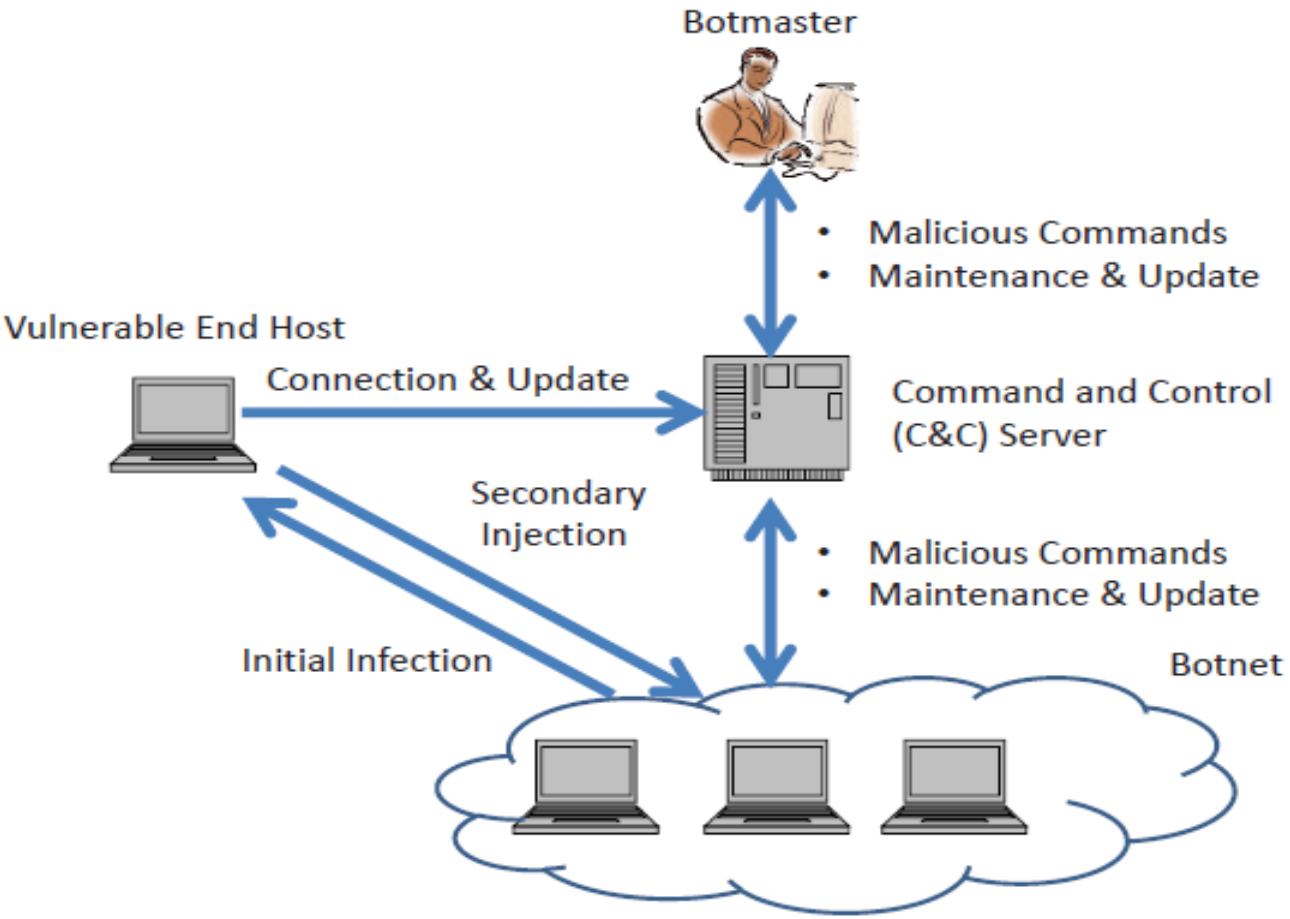
Classification of Countermeasures (cont.)



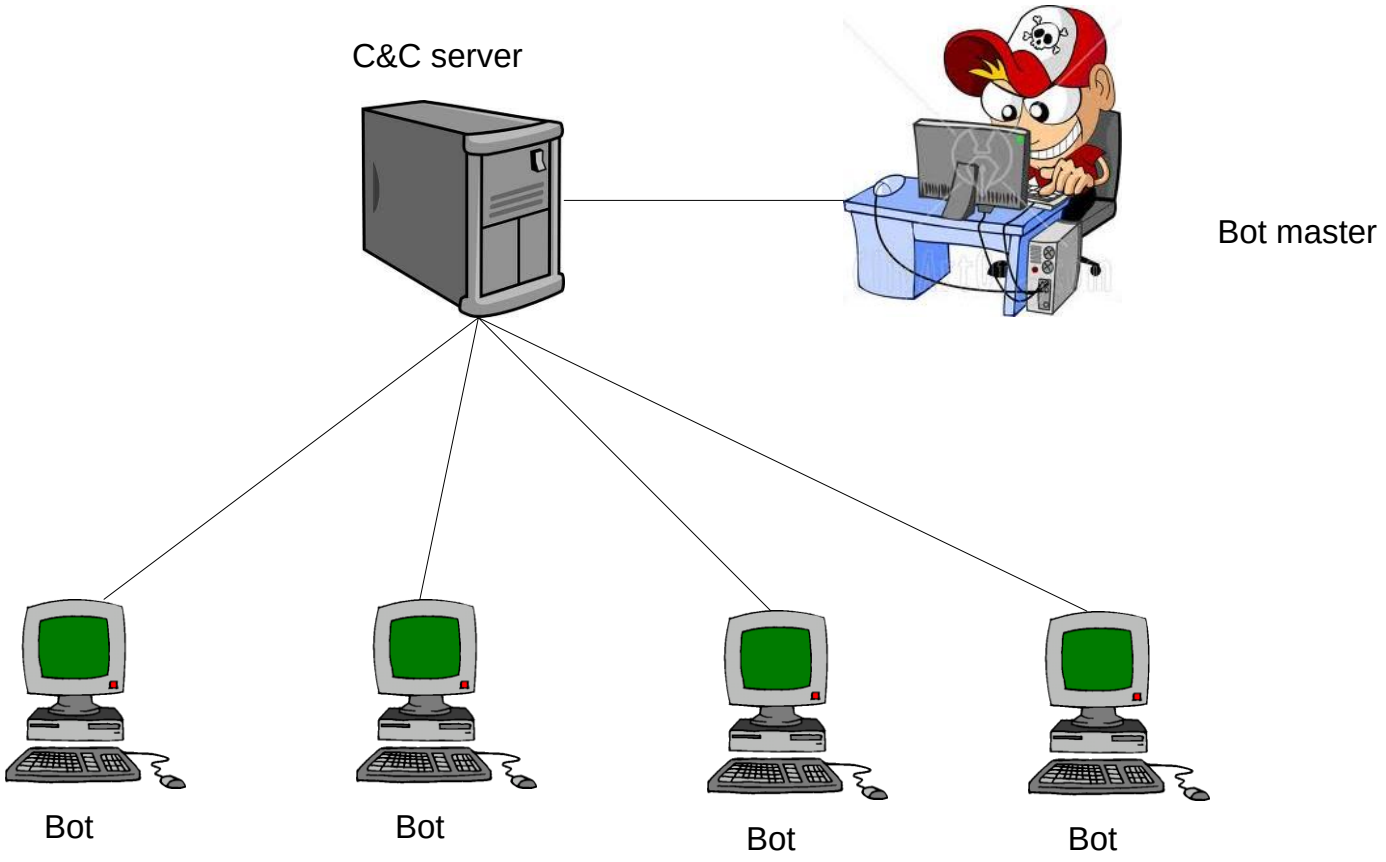
Classification of Worm Detection (cont.)



New Generation of Worm - Botnet

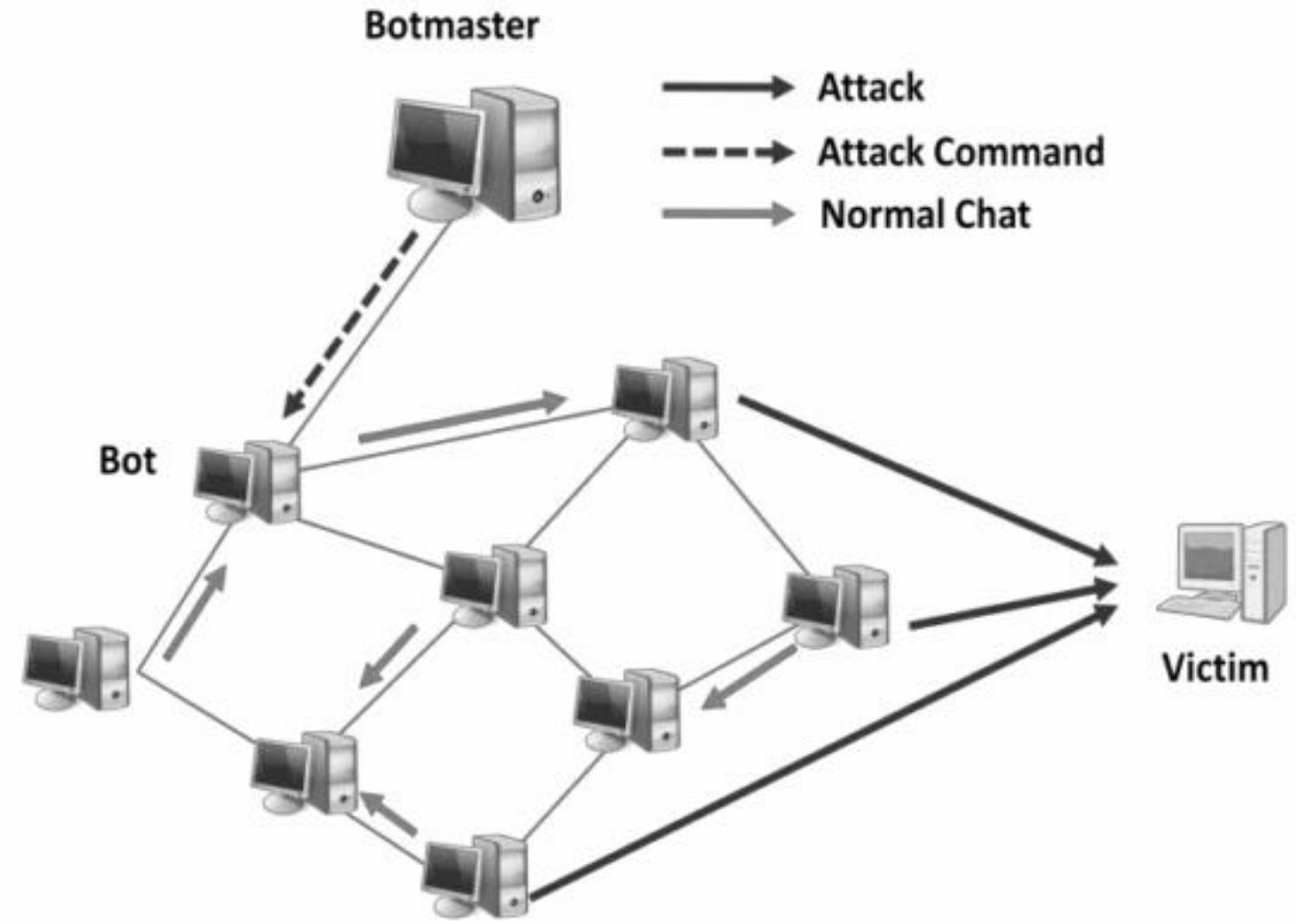


New Generation of Worm - Botnet



Centralized Architecture

New Generation of Worm - Botnet

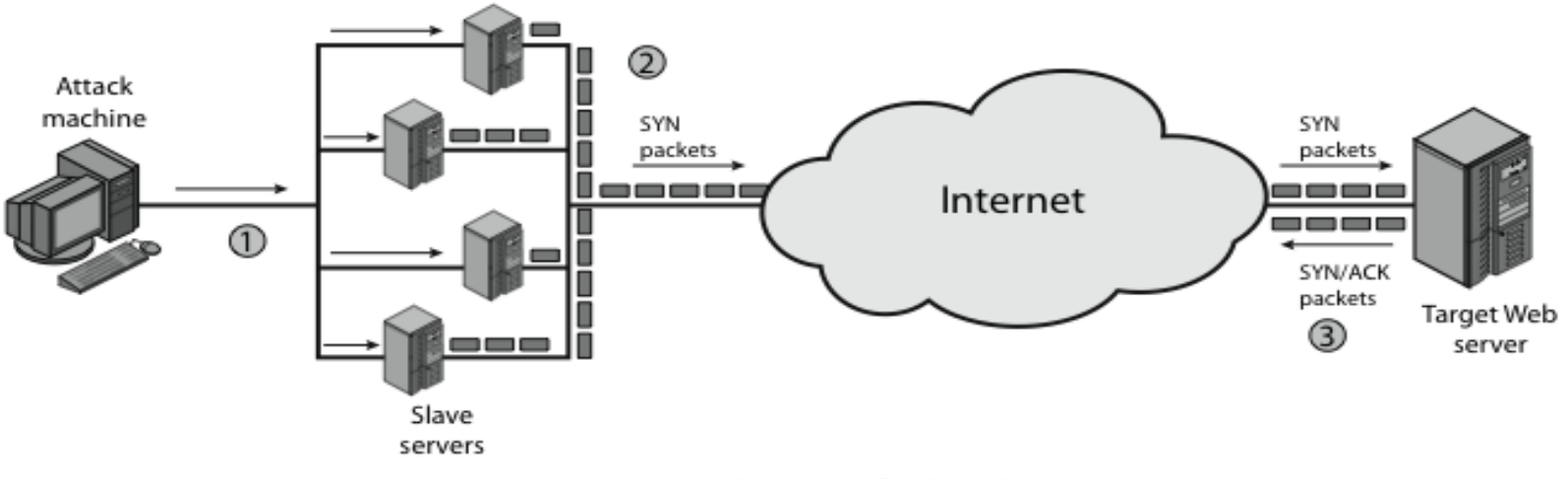


Distributed Architecture

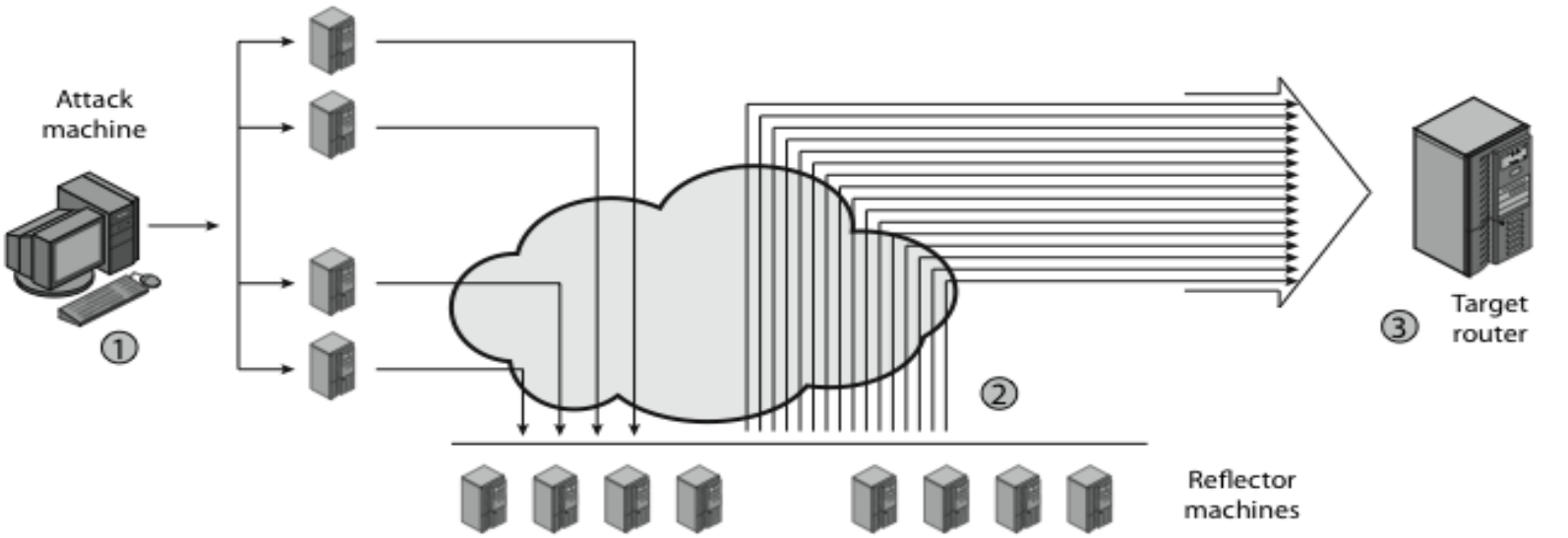
Distributed Denial of Service Attacks (DDoS)

- Distributed Denial of Service (DDoS) attacks form a significant security threat
- Making networked systems unavailable by flooding with useless traffic
- Using large numbers of “zombies”
- Growing sophistication of attacks
- Defense technologies struggling to cope

Distributed Denial of Service Attacks (DDoS)



(a) Distributed SYN flood attack



(a) Distributed ICMP attack

Workflow of DDoS

- Previous slide figure (a) shows an example of an internal resource attack - the SYN flood attack.
  - The attacker takes control of multiple hosts over the Internet
  - The slave hosts begin sending TCP/IP SYN (synchronize/initialization) packets, with erroneous return IP address information, to the target
  - For each such packet, the Web server responds with a SYN/ACK (synchronize/acknowledge) packet. The Web server maintains a data structure for each SYN request waiting for a response back and becomes bogged down as more traffic floods in.
- Previous slide figure (b) illustrates an example of an attack that consumes data transmission resources.
  - The attacker takes control of multiple hosts over the Internet, instructing them to send ICMP ECHO packets with the target's spoofed IP address to a group of hosts that act as reflectors
  - Nodes at the bounce site receive multiple spoofed requests and respond by sending echo reply packets to the target site.
  - The target's router is flooded with packets from the bounce site, leaving no data transmission capacity for legitimate traffic.

Constructing an Attack Network

- Must infect large number of zombies and needs:
  - Software to implement the DDoS attack
  - An unpatched vulnerability on many systems
  - Scanning strategy to find vulnerable systems
    - Random, hit-list, topological, local subnet

DDoS Countermeasures

- Three broad lines of defense:
  - Attack prevention & preemption (before): to enable victim to endure attack attempts without denying service to legitimate clients
  - Attack detection & filtering (during): to attempt to detect attack as it begins and respond immediately, minimizing impact of attack on the target
  - Attack source traceback & ident (after): to identify source of attack to prevent future attacks.
- Huge range of attack possibilities
- Hence evolving countermeasures

Summary

- Have considered:
  - Various malicious programs
  - Trapdoor, logic bomb, trojan horse, zombie
  - Viruses
  - Worms
  - Distributed denial of service attacks