

CHAPTER - 10

Malicious Software



Key Points

- Malicious software is software that is intentionally included or inserted in a system for a harmful purpose.
- A virus is a piece of software that can “infect” other programs by modifying them; the modification includes a copy of the virus program, which can then go on to infect other programs.
- A worm is a program that can replicate itself and send copies from computer to computer across network connections.
- Upon arrival, the worm may be activated to replicate and propagate again.
- In addition to propagation, the worm usually performs some unwanted function.
- A denial of service (DoS) attack is an attempt to prevent legitimate users of a service from using that service.
- A distributed denial of service attack is launched from multiple coordinated sources.



Introduction

- The most sophisticated types of threats to computer systems are presented by programs that exploit vulnerabilities in computing systems.
- Such threats are referred to as **malicious software**, or **malware**.



10.1 Types of Malicious Software

- Malicious software can be divided into two categories:
- First one, **Parasitic** that need a host program, and cannot exist independently of some actual application program, utility, or system program. E.g. Viruses, logic bombs, and backdoors
- Second one, **Independent malware** is a self-contained program that can be scheduled and run by the operating system. E.g. Worms and bot programs.
- We can also differentiate between those software threats that do not replicate and those that do.
- The former are programs or fragments of programs that are activated by a trigger. E.g. logic bombs, backdoors, and bot programs.
- The latter consist of either a program fragment or an independent program that, when executed, may produce one or more copies of itself to be activated later on the same system or some other system. E.g. Viruses and worm.



Table 10.1 Terminology of Malicious Programs

Name	Description
Virus	Malware that, when executed, tries to replicate itself into other executable code; when it succeeds the code is said to be infected. When the infected code is executed, the virus also executes.
Worm	A computer program that can run independently and can propagate a complete working version of itself onto other hosts on a network.
Logic bomb	A program inserted into software by an intruder. A logic bomb lies dormant until a predefined condition is met; the program then triggers an unauthorized act.
Trojan horse	A computer program that appears to have a useful function, but also has a hidden and potentially malicious function that evades security mechanisms, sometimes by exploiting legitimate authorizations of a system entity that invokes the Trojan horse program.
Backdoor (trapdoor)	Any mechanism that bypasses a normal security check; it may allow unauthorized access to functionality.
Mobile code	Software (e.g., script, macro, or other portable instruction) that can be shipped unchanged to a heterogeneous collection of platforms and execute with identical semantics.
Exploits	Code specific to a single vulnerability or set of vulnerabilities.
Downloaders	Program that installs other items on a machine that is under attack. Usually, a downloader is sent in an e-mail.
Auto-rooter	Malicious hacker tools used to break into new machines remotely.
Kit (virus generator)	Set of tools for generating new viruses automatically.

Table 10.1 Terminology of Malicious Programs

Name	Description
Spammer programs	Used to send large volumes of unwanted e-mail.
Flooders	Used to attack networked computer systems with a large volume of traffic to carry out a denial-of-service (DoS) attack.
Keyloggers	Captures keystrokes on a compromised system.
Rootkit	Set of hacker tools used after attacker has broken into a computer system and gained root-level access.
Zombie, bot	Program activated on an infected machine that is activated to launch attacks on other machines.
Spyware	Software that collects information from a computer and transmits it to another system.
Adware	Advertising that is integrated into software. It can result in pop-up ads or redirection of a browser to a commercial site.



Backdoor

- A **backdoor**, also known as a **trapdoor**, is a secret entry point into a program that allows someone to gain access without going through the usual security access procedures.
- Backdoors have been used legitimately for many years to debug and test programs; such a backdoor is called a **maintenance hook**.
- The **backdoor** is code that recognizes some special sequence of input or is triggered by being run from a certain user ID or by an unlikely sequence of events.
- Backdoors become threats when unscrupulous programmers use them to gain unauthorized access. E.g. idea behind the movie *War Games*, during the development of Multics
- It is difficult to implement operating system controls for backdoors. Security measures must focus on the program development and software update activities.




Logic Bomb

- The logic bomb is code embedded in some legitimate program that is set to “explode” when certain conditions are met.
- Examples of conditions that can be used as triggers for a logic bomb are the presence or absence of certain files, a particular day of the week or date, or a particular user running the application.
- Once triggered, a bomb may alter or delete data or entire files, cause a machine halt, or do some other damage.



Trojan Horses

- A Trojan horse is a useful program or command procedure containing hidden code that, when invoked, performs some unwanted or harmful function.
 - Trojan horse programs can be used to accomplish functions indirectly that an unauthorized user could not accomplish directly.
 - Another common motivation for the Trojan horse is data destruction. The program appears to be performing a useful function (e.g., a calculator program), but it may also be quietly deleting the user's files.
 - Trojan horses fit into one of three models:
 - Continuing to perform the function of the original program along with a separate malicious activity.
 - Continuing to perform the function of the original program but modifying the function to perform malicious or to disguise other malicious activity.
 - Performing a malicious function that completely replaces the function of the original program.
- 

Mobile Code

- Mobile code refers to programs (e.g., script, macro, or other portable instruction) that can be shipped unchanged to a heterogeneous collection of platforms and execute with identical semantics.
- Mobile code is transmitted from a remote system to a local system and then executed on the local system without the user's explicit instruction.
- The most common ways of using mobile code for malicious operations on local system are cross-site scripting, interactive and dynamic Web sites, e-mail attachments, and downloads from untrusted sites or of untrusted software.



Multiple-Threat Malware

- A **multipartite virus** infects in multiple ways infecting multiple types of files, so virus eradication must deal with all of the possible sites of infection.
- A **blended attack** uses multiple methods of infection or transmission, to maximize the speed of contagion and the severity of the attack.
- Some writers characterize a blended attack as a package that includes multiple types of malware. E.g. Nimda attack which uses four distribution methods: Email, Windows shares, Web servers, and Web clients.
- Nimda has worm, virus, and mobile code characteristics. Blended attacks may also spread through other services, such as instant messaging and peer-to-peer file sharing.



10.2 VIRUSES

○ **The Nature of Viruses**

- A computer virus is a piece of software that can “infect” other programs by modifying them; E.g. injecting the original program with a routine to make copies of the virus program, which in turn infects other programs.
- Computer viruses first appeared in the early 1980s.
- Like biological viruses, a computer virus carries in its instructional code the recipe for making perfect copies of itself.
- A virus attaches itself to another program and executes secretly when the host program runs.
- Once a virus is executing, it can perform any function, such as erasing files and programs that is allowed by the privileges of the current user.



VIRUSES

- A computer virus has three parts:
- **Infection mechanism:** The means by which a virus spreads, enabling it to replicate. The mechanism is also referred to as the infection vector.
- **Trigger:** The event or condition that determines when the payload is activated or delivered.
- **Payload:** What the virus does, besides spreading. The payload may involve damage or may involve benign but noticeable activity.



VIRUSES

- During its lifetime, a typical virus goes through the following four phases:
- **Dormant phase:** The virus is idle and 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 a copy of itself into other programs or into certain system areas on the disk.
- The copy may not be identical to the propagating version; viruses often morph to evade detection.
- Each infected program will now contain a clone of the virus, which will itself enter a propagation phase.



VIRUSES

- **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.
- The function may be harmless, such as a message on the screen, or damaging, such as the destruction of programs and data files.
- N.B.
- Most viruses carry out their work specific to a particular operating system and a particular hardware platform.
- Thus, they are designed to take advantage of the details and weaknesses of particular systems.



VIRUS STRUCTURE

- A virus can be prepended or postpended to an executable program, or it can be embedded in some other fashion.
- The key to its operation is that the infected program, when invoked, will first execute the virus code and then execute the original code of the program.



Example

- The virus code, V here, is prepended to infected programs, as the entry point to the program.
- The first line of code is a jump to the main virus program.
- The second line is a special marker that is used by the virus to determine whether or not a potential victim program has already been infected with this virus.
- On invocation, control is immediately transferred to the main virus program.
- The virus program may find uninfected executable files, infect them and may perform some detrimental action.
- This action could be performed every time the program is

```
program V :=  
  
{goto main;  
 1234567;  
  
  subroutine infect-executable :=  
    {loop:  
      file := get-random-executable-file;  
      if (first-line-of-file = 1234567)  
        then goto loop  
        else prepend V to file; }  
  
  subroutine do-damage :=  
    {whatever damage is to be done}  
  
  subroutine trigger-pulled :=  
    {return true if some condition holds}  
  
main:  main-program :=  
       {infect-executable;  
       if trigger-pulled then do-damage;  
       goto next;}  
next:  
  
}
```

Figure 10.1 A Simple Virus

Example

- If the infection phase of the program is reasonably rapid, a user is unlikely to notice any difference between the execution of an infected and an uninfected program.
- Such viruses can be detected easily because an infected version of a program is longer than the corresponding uninfected one.
- If compression is applied, both infected and uninfected files may look identical in length.
- Figure 10.2 shows in general terms the logic required.
- We assume that program P1 is infected with the virus CV. When this program is invoked, control passes to its virus.

```
program CV :=  
  
  {goto main;  
   01234567;  
  
   subroutine infect-executable :=  
     {loop:  
       file := get-random-executable-file;  
       if (first-line-of-file = 01234567) then goto loop;  
       (1) compress file;  
       (2) prepend CV to file;  
     }  
  
  main: main-program :=  
    {if ask-permission then infect-executable;  
     (3) uncompress rest-of-file;  
     (4) run uncompressed file;  
    }
```

Figure 10.2 Logic for a Compression Virus

Example

- 1. For each uninfected file P2 that is found, the virus first compresses that file to produce P2', which is shorter than the original program by the size of the virus.
- 2. A copy of the virus is prepended to the compressed program.
- 3. The compressed version of the original infected program, P1', is uncompressed.
- 4. The uncompressed original program is executed.
- Here, the virus does nothing other than propagate. As previously mentioned, the virus may include a logic bomb.



INITIAL INFECTION

- Once virus enters into a system by infecting a single program, it can potentially infect some or all other executable files when the infected program executes.
- Preventing the virus from gaining entry in the system can be a best solution to virus prevention which is extremely difficult.
- Many forms of infection can also be blocked by denying normal users the right to modify programs on the system.
- The lack of access controls on early PCs is a key reason why traditional machine code based viruses spread rapidly on these systems.
- Traditional machine code based viruses are now less prevalent, because modern PC OSs do have more effective access controls.
- However, virus creators have found other avenues, such as macro and e-mail viruses.



Viruses Classification

- There is no simple or universally agreed upon classification scheme for viruses.
- We follow and classify viruses along two orthogonal axes: the type of target the virus tries to infect and the method the virus uses to conceal itself from detection by users and antivirus software.
- A virus **classification by target** includes the following categories:
- **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 the operating system or shell consider to be executable.
- **Macro virus:** Infects files with macro code that is interpreted by an application.



Viruses Classification

- A virus classification by concealment strategy includes the following categories:
- **Encrypted virus:** A portion of the virus creates a random encryption key and encrypts the remainder of the virus.
- The key is stored with 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.
- Because the bulk of the virus is encrypted with a different key for each instance, there is no constant bit pattern to observe.



Viruses Classification

- **Stealth virus:** A form of virus explicitly designed to hide itself from detection by antivirus software.
- **Polymorphic virus:** A virus that mutates with every infection, making detection by the “signature” of the virus impossible.
- **Metamorphic virus:** A metamorphic virus also mutates with every infection. The difference is that it rewrites itself completely at each iteration, increasing the difficulty of detection.
- Metamorphic viruses may change their behavior as well as their appearance.



Virus Kits

- Virus-creation toolkit enables a relative novice to quickly create a number of different viruses.
- Although toolkit created viruses tend to be less sophisticated than ones designed from scratch, the sheer number of new viruses that can be generated using a toolkit creates a problem for antivirus schemes.



Macro Viruses

- In the mid-1990s, macro viruses became by far the most prevalent type of virus. Macro viruses are particularly threatening for a number of reasons:
- 1. A macro virus is platform independent. Many macro viruses infect Microsoft Word documents or other Microsoft Office documents.
- Any hardware platform and operating system that supports these applications can be infected.
- 2. Macro viruses infect documents, not executable portions of code. Most of the information introduced onto a computer system is in the form of a document rather than a program.
- 3. Macro viruses are easily spread. A very common method is by electronic mail.
- 4. Because macro viruses infect user documents rather than system programs, traditional file system access controls are of limited use in preventing their spread.



Macro Viruses

- Macro viruses take advantage of a feature found in Word and other office applications such as Microsoft Excel, namely the macro.
- Successive releases of MS Office products provide increased protection against macro viruses.
- For example, Microsoft offers an optional Macro Virus Protection tool that detects suspicious Word files and alerts the customer to the potential risk of opening a file with macros.
- Various antivirus product vendors have also developed tools to detect and correct macro viruses.



E-Mail Viruses

- A more recent development in malicious software is the e-mail virus.
- The first rapidly spreading e-mail viruses, such as Melissa, made use of a Microsoft Word macro embedded in an attachment.
- If the recipient opens the e-mail attachment, the Word macro is activated. Then
 - 1. The e-mail virus sends itself to everyone on the mailing list in the user's e-mail package.
 - 2. The virus does local damage on the user's system.



E-Mail Viruses

- The virus propagates itself as soon as it is activated (either by opening an e-mail attachment or by opening the e-mail) to all of the e-mail addresses known to the infected host.
- As a result, whereas viruses used to take months or years to propagate, they now do so in hours.
- A greater degree of security must be built into Internet utility and application software on PCs to counter the growing threat.



10.3 VIRUS COUNTERMEASURES

Antivirus Approaches

- The ideal solution to the threat of viruses is prevention
- or block the ability of a virus to modify any files containing executable code or macros.
- This goal is impossible to achieve, but prevention can reduce the number of successful viral attacks.




10.3 VIRUS COUNTERMEASURES

- The next best approach is to be able to do the following:
- **Detection:** Determine that infection has occurred and locate the virus.
- **Identification:** After detection, identify the specific virus that has infected a program.
- **Removal:** Once 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 virus cannot spread further.
- If identification or removal is not possible, discard the infected file and reload a clean backup version.
- As the virus arms race has evolved, both viruses and, necessarily, antivirus software have grown more complex and sophisticated.



VIRUS COUNTERMEASURES

- Four generations of antivirus software:
 - First generation: simple scanners, requires a virus signature to identify a virus. The virus has same structure and bit pattern in all copies.
 - Such signature-specific scanners are limited to the detection of known viruses. Another type of scanner maintains a record of the length of programs and looks for changes in length.
 - Second generation: heuristic scanners instead of signature, uses heuristic rules to search for probable virus infection.
 - Another second-generation approach is integrity checking. A checksum can be appended to each program.
 - If a virus infects the program without changing the checksum, then an integrity check will catch the change.
 - If virus changes the checksum too, an encrypted hash function can be used along with the key is stored separately so that the virus cannot generate a new hash code and encrypt that.
- 

VIRUS COUNTERMEASURES

- Third generation: activity traps. Memory-resident programs that identify a virus by its actions rather than its structure.
- This approach doesn't require to develop signatures and heuristics for a wide array of viruses but small set of actions that indicate an infection is being attempted and then to intervene.
- Fourth generation: full-featured protection, products are packages consisting of a variety of antivirus techniques including scanning and activity trap components.
- With access control facility it can limit ability of viruses to penetrate a system and to update files in order to pass on the infection.



Advanced Antivirus Techniques

- **Generic decryption (GD)** technology enables the antivirus program to easily detect even the most complex polymorphic viruses while maintaining fast scanning speeds.
- Executable files are run through a GD scanner, which contains the following elements:
- **CPU emulator:** A s/w based virtual computer. Instructions in an executable file are interpreted by the emulator rather than executed on the underlying processor.
- **Virus signature scanner:** A module that scans the target code looking for known virus signatures.
- **Emulation control module:** Controls the execution of the target code.



Digital Immune System

- A comprehensive approach to virus protection developed by and subsequently refined by Symantec.
- Two major trends in Internet technology have had an increasing impact on the rate of virus propagation in recent years:
- **Integrated mail systems:** Systems such as Lotus Notes and Microsoft Outlook make it very simple to send anything to anyone and to work with objects that are received.
- **Mobile-program systems:** Capabilities such as Java and ActiveX allow programs to move on their own from one system to another.
- The success of the digital immune system depends on the ability of the virus analysis machine to detect new and innovative virus strains.



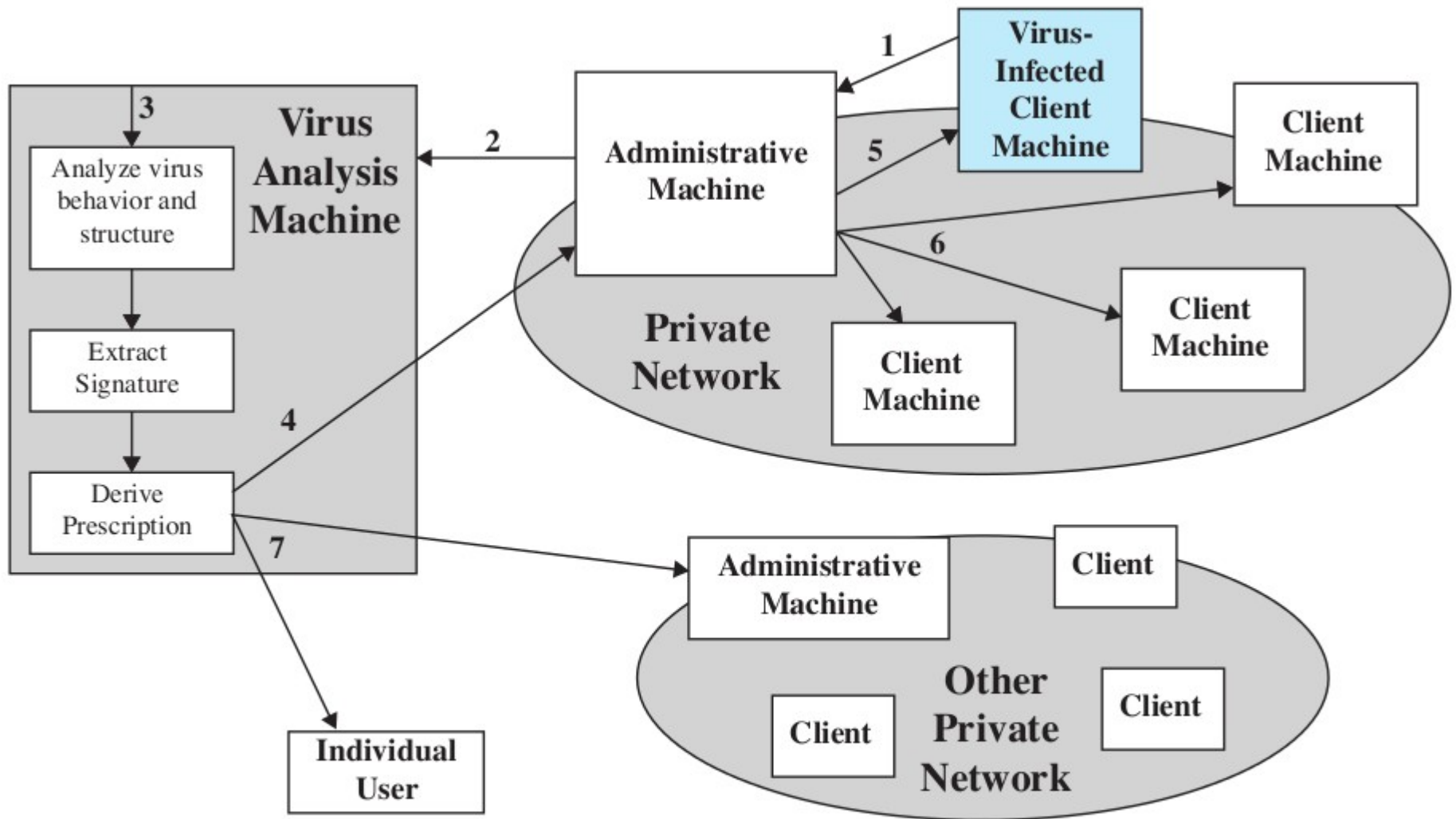


Figure 10.4 Digital Immune System

Behavior-Blocking S/W

- Integrates with the operating system of a host computer and monitors program behavior in real-time for malicious actions.
- It blocks potentially malicious actions before they have a chance to 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.



1. Administrator sets acceptable software behavior policies and uploads them to a server. Policies can also be uploaded to desktops.

3. Behavior-blocking software at server flags suspicious code. The blocker "sandboxes" the suspicious software to prevent it from proceeding

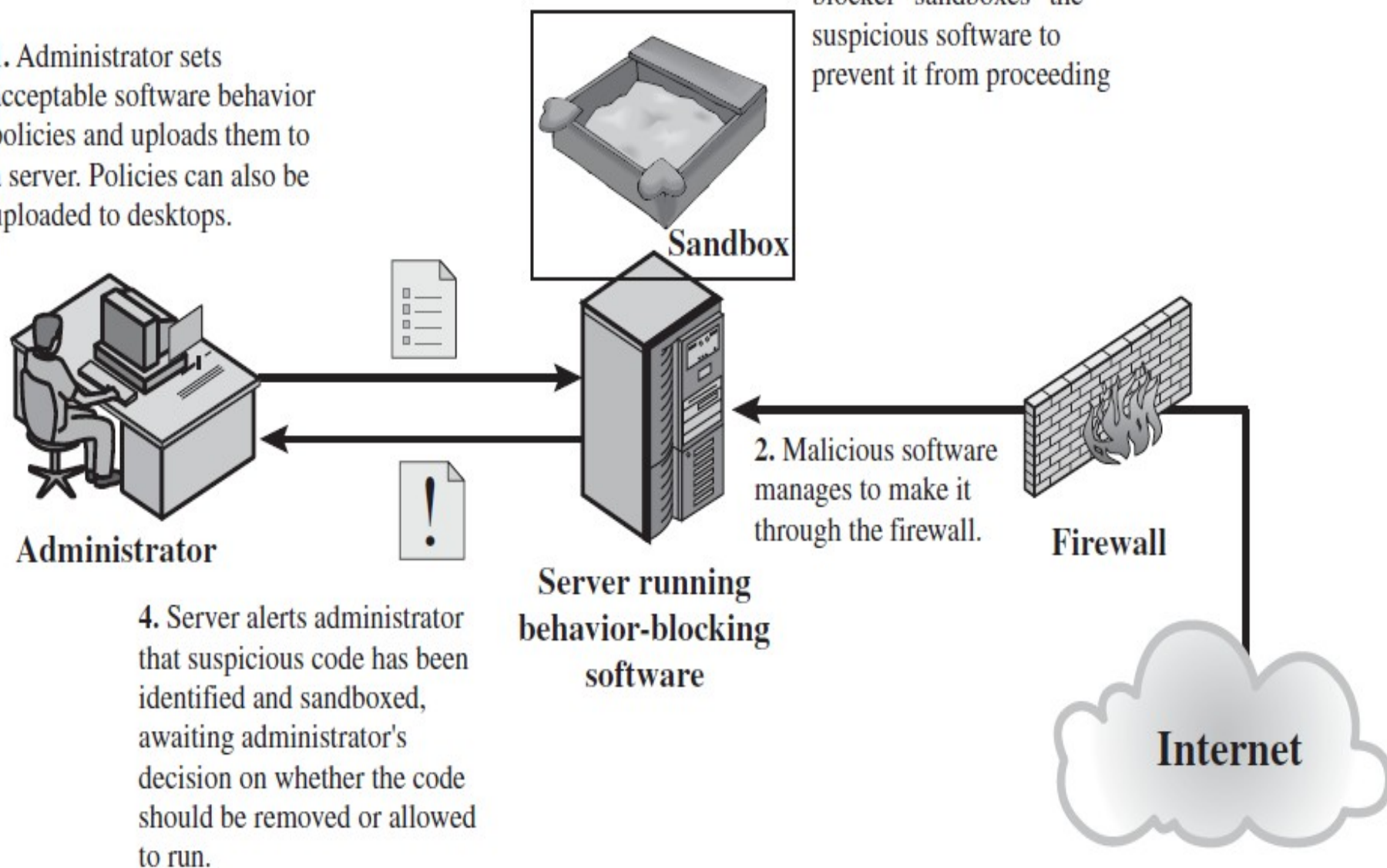


Figure 10.5 Behavior-Blocking Software Operation

Behavior-Blocking S/W

- 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.



10.4 Worms

- A worm is a program that can replicate itself and send copies from computer to computer across network connections.
- Along with propagation, the worm usually performs some unwanted function.
- A worm actively seeks out more machines to infect and each machine that is infected serves as an automated launching pad for attacks on other machines.



Worms

- To replicate itself, a network worm uses some sort of network vehicle.
- Examples include the following:
- **Electronic mail facility:** A worm mails a copy of itself to other systems, so that its code is run when the e-mail or an attachment is received or viewed.
- **Remote execution capability:** A worm executes a copy of itself on another system, either using an explicit remote execution facility or by exploiting a program flaw in a network service to subvert its operations.
- **Remote login capability:** A worm logs onto a remote system as a user and then uses commands to copy itself from one system to the other, where it then executes.



Worms

- A network worm exhibits the same characteristics as a computer virus: **dormant, propagation, triggering, and an execution phase.**
- The propagation phase generally performs the following functions:
 - 1. Search for other systems to infect by examining host tables or similar repositories of remote system addresses.
 - 2. Establish a connection with a remote system.
 - 3. Copy itself to the remote system and cause the copy to be run.
- The n/w worm may also attempt to determine whether a system has previously been infected.
- In a multi-programming system, it may also disguise its presence by naming itself as a system process or using some other name that may not be noticed by a system operator.



Worm Propagation Model

- The speed of propagation and the total number of hosts infected depend on a number of factors like:
 - Mode of propagation,
 - The vulnerability or vulnerabilities exploited, and
 - The degree of similarity to preceding attacks.



Recent Worm Attacks

- Code Red worm in July of 2001.
- Code Red exploits a security hole in the Microsoft Internet Information Server (IIS) to penetrate and spread.
- It also disables the system file checker in Windows. The worm probes random IP addresses to spread to other hosts.
- Initially, it only spreads and then initiates a denial-of-service attack against a government Web site by flooding the site with packets from numerous hosts.
- The worm then suspends activities and reactivates periodically.
- In the second wave of attack, Code Red infected nearly 360,000 servers in 14 hours.
- In addition Code Red also consumed enormous amounts of Internet capacity, disrupting service.



Recent Worm Attacks

- In early 2003, the SQL Slammer worm appeared. This worm exploited a buffer overflow vulnerability in Microsoft SQL server.
- The Slammer was extremely compact and spread rapidly, infecting 90% of vulnerable hosts within 10 minutes.
- In late 2003 Sobig.f worm arrived, which exploited open proxy servers to turn infected machines into spam engines.
- At its peak, it accounted for one in every 17 messages and produced more than one million copies of itself within the first 24 hours.



State of Worm Technology

- The state of the art in worm technology 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.



State of Worm Technology

- **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:** Worms are ideal for spreading other distributed attack tools, such as distributed denial of service bots.
- **Zero-day exploit:** a worm should exploit an unknown vulnerability that is only discovered by the general network community when the worm is launched.




State of Worm Technology

- Worms first appeared on mobile phones in 2004.
- These worms communicate through Bluetooth wireless connections or via the multimedia messaging service (MMS).
- The target is the smartphone/mobile phone that permits users to install software applications from sources other than the cellular network operator.
- Mobile phone malware can completely disable the phone, delete data on the phone, or force the device to send costly messages to premium-priced numbers.
- E.g. CommWarrior, launched in 2005, replicates by means of Bluetooth to other phones in the receiving area.
- It also sends itself as an MMS file to address book numbers and in automatic replies to incoming text messages and MMS messages.
- It also copies itself to the removable memory card and inserts itself into the program installation files on the phone.



Worm Countermeasures

- Once a worm is resident on a machine, antivirus software can be used to detect it.
 - Because worm propagation generates considerable network activity, network activity and usage monitoring can also form the basis of a worm defense.
 - The requirements for an effective worm countermeasure scheme:
 - **Generality** (to handle a wide variety of worm attacks)
 - **Timeliness** (respond quickly to limit the number of infected systems)
 - **Resiliency** (resistant to evasion techniques employed by attackers)
 - **Minimal denial-of-service cost** (the countermeasure should not significantly disrupt normal operation)
 - **Transparency** (should not require modification to existing (legacy) OSs, application software, and hardware)
 - **Global and Local coverage** (to deal with attack sources both from outside and inside the enterprise network)
- 

Countermeasure Approaches

- **A. Signature-based worm scan filtering:** this approach involves identifying suspicious flows and generating a worm signature.
- Either the detection software misses the worm or, if it is sufficiently sophisticated to deal with polymorphic worms, the scheme may take a long time to react.
- **B. Filter-based worm containment:** this approach is similar to the previous one but focuses on worm content. The filter checks a message to determine if it contains worm code.
- **C. Payload-classification-based worm containment:** examine packets to see if they contain a worm. Various anomaly detection techniques can be used, but care is needed to avoid high levels of false positives or negatives.
- **D. Threshold random walk (TRW) scan detection:** a way of detecting if a scanner is in operation. TRW is suitable for deployment in high-speed, low-cost network devices.

Countermeasure Approaches

- **E. Rate limiting:** This class limits the rate of scanlike traffic from an infected host. This class of countermeasures may introduce longer delays for normal traffic.
- This class is also not suited for slow, stealthy worms that spread slowly to avoid detection based on activity level.
- **F. 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

- PWC is designed to address the threat of worms that spread rapidly.
- The software on a host looks for surges in the rate of frequency of outgoing connection attempts and the diversity of connections to remote hosts.
- When such a surge is detected, the software immediately blocks its host from further connection attempts.
- A deployed PWC system consists of a PWC manager and PWC agents in hosts.
- In following example, the security manager, signature extractor, and PWC manager are implemented in a single network device.
- In practice, these three modules could be implemented as two or three separate devices.



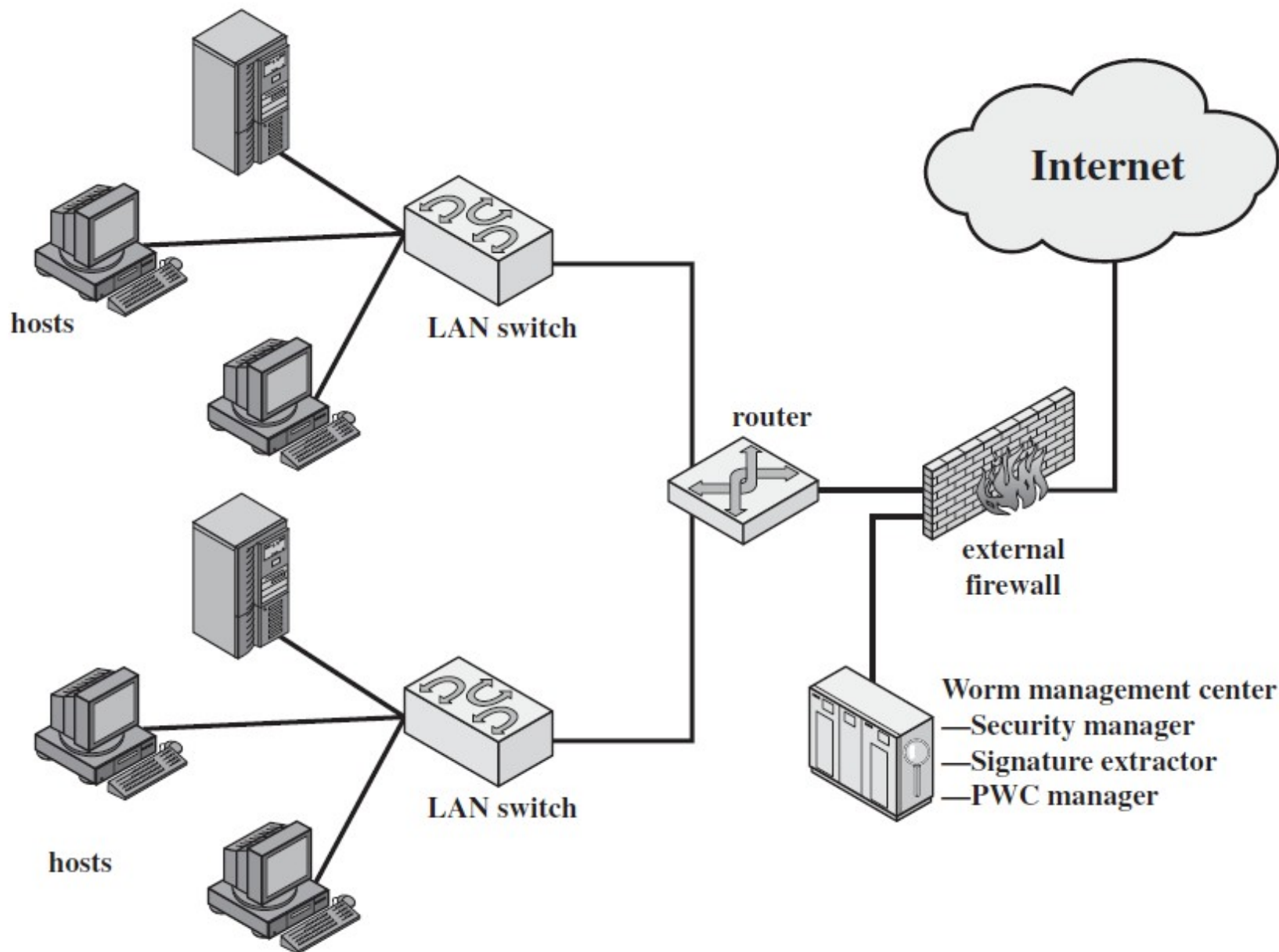


Figure 10.7 Example PWC Deployment

Network-Based Worm Defense

- The key element here is worm monitoring software. Consider an enterprise network at a site, consisting of one or an interconnected set of LANs.
- Two types of monitoring software are needed:
- **Ingress monitors:** located at the border between the enterprise network and the Internet.
- They can be part of the ingress filtering software of a border router or external firewall or a separate passive monitor.
- A honeypot can also capture incoming worm traffic.
- **Egress monitors:** These can be located at the egress point of individual LANs on the enterprise network as well as at the border between the enterprise network and the Internet.
- The egress monitor is designed to catch the source of a worm attack by monitoring outgoing traffic for signs of scanning or other suspicious behavior.



Worm Countermeasure Architecture

- Figure 10.8 shows an example of a worm countermeasure architecture.
- 1. Sensors deployed at various network locations detect a potential worm. The sensor logic can also be incorporated in IDS sensors.
- 2. The sensors send alerts to a central server that correlates and analyzes the incoming alerts about the likelihood that a worm attack is being observed and the key characteristics of the attack.
- 3. The server forwards its information to a protected environment, where the potential worm may be sandboxed for analysis and testing.
- 4. The protected system tests the suspicious software against an appropriately instrumented version of the targeted application to identify the vulnerability.
- 5. The protected system generates one or more software patches and tests these.
- 6. If the patch is not susceptible to the infection and does not compromise the application's functionality, the system sends the patch to the application host to update the targeted application.

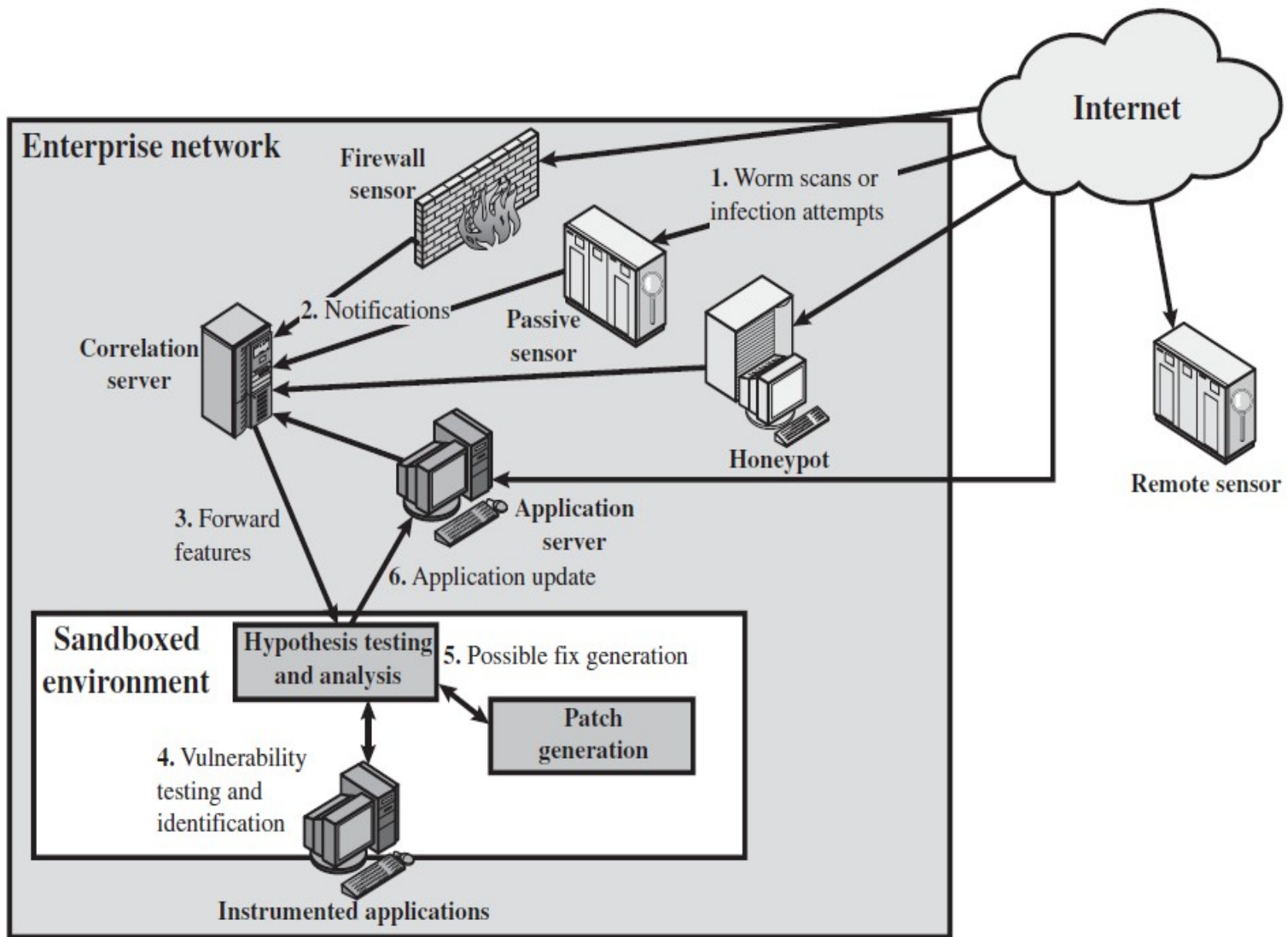


Figure 10.8 Placement of Worm Monitors

10.5 Distributed Denial of Service Attacks

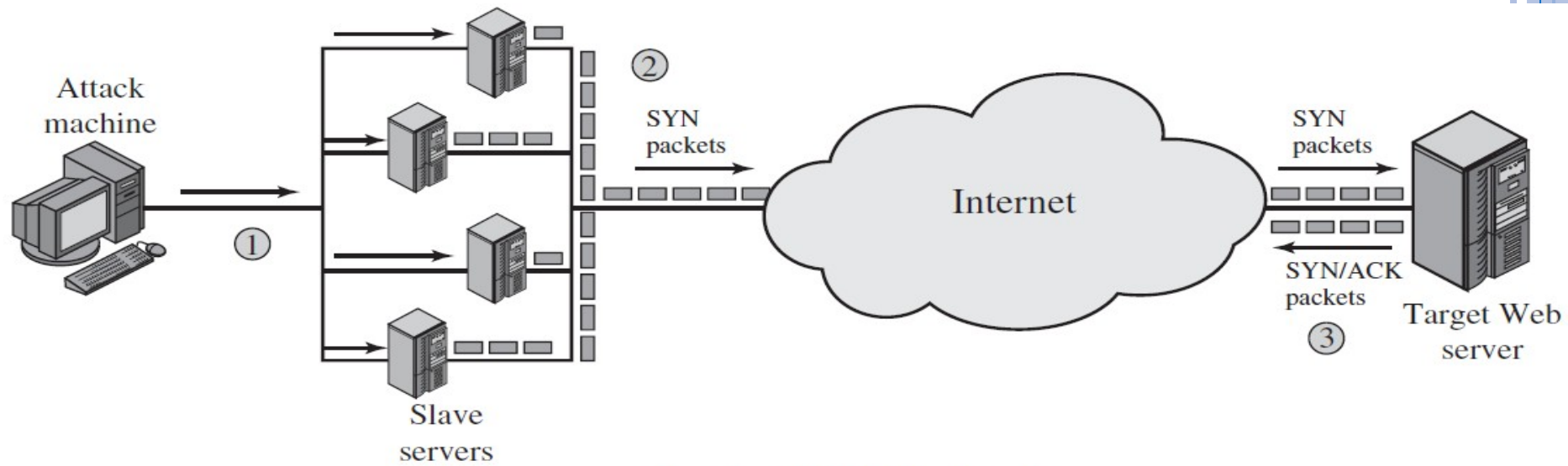
- A significant security threat to corporations, and the threat appears to be growing.
- A denial of service (DoS) attack is an attempt to prevent legitimate users of a service from using that service.
- An attacker is able to recruit a number of hosts throughout the Internet to simultaneously or in a coordinated fashion launch an attack upon the target.
- A DDoS attack attempts to consume the target's resources so that it cannot provide service. One way to classify DDoS attacks is in terms of the type of resource that is consumed.



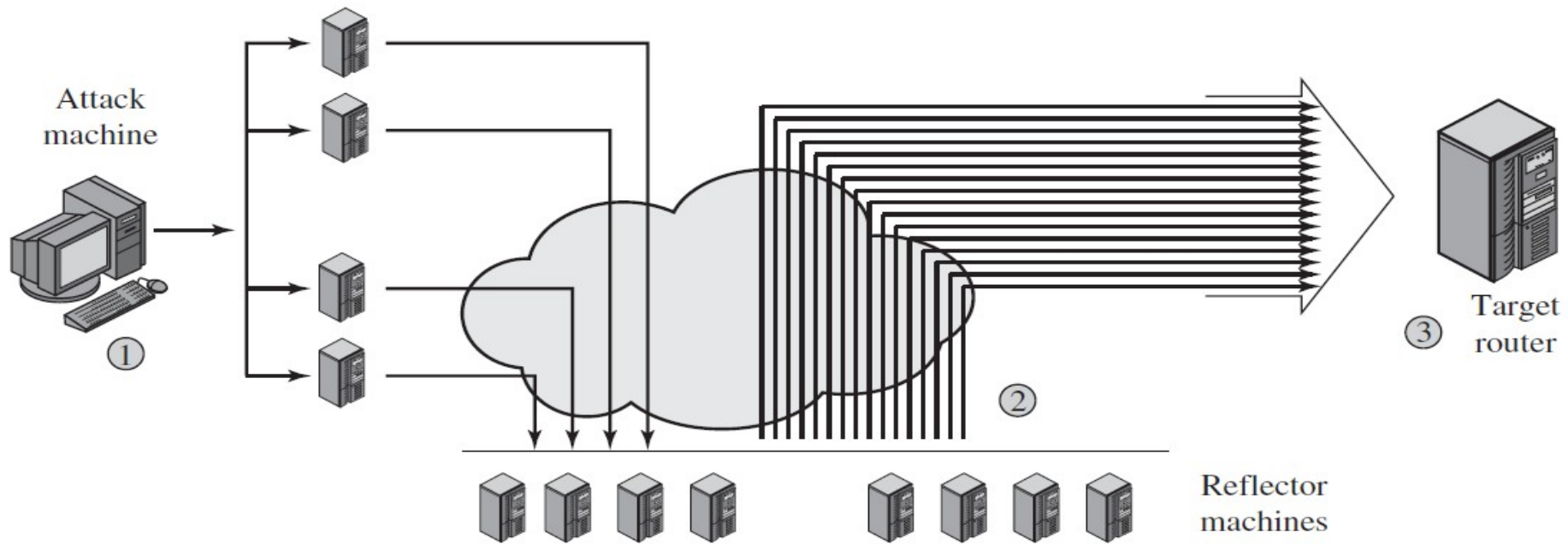
DDoS Attack Description

- The resource consumed by DDoS is either an internal host resource on the target system or data transmission capacity in the local network to which the target is attacked.
- A simple example of an **internal resource attack** is the SYN flood attack as shown in Figure 10.9a.
- Figure 10.9b illustrates an example of **an attack that consumes data transmission resources**.





(a) Distributed SYN flood attack

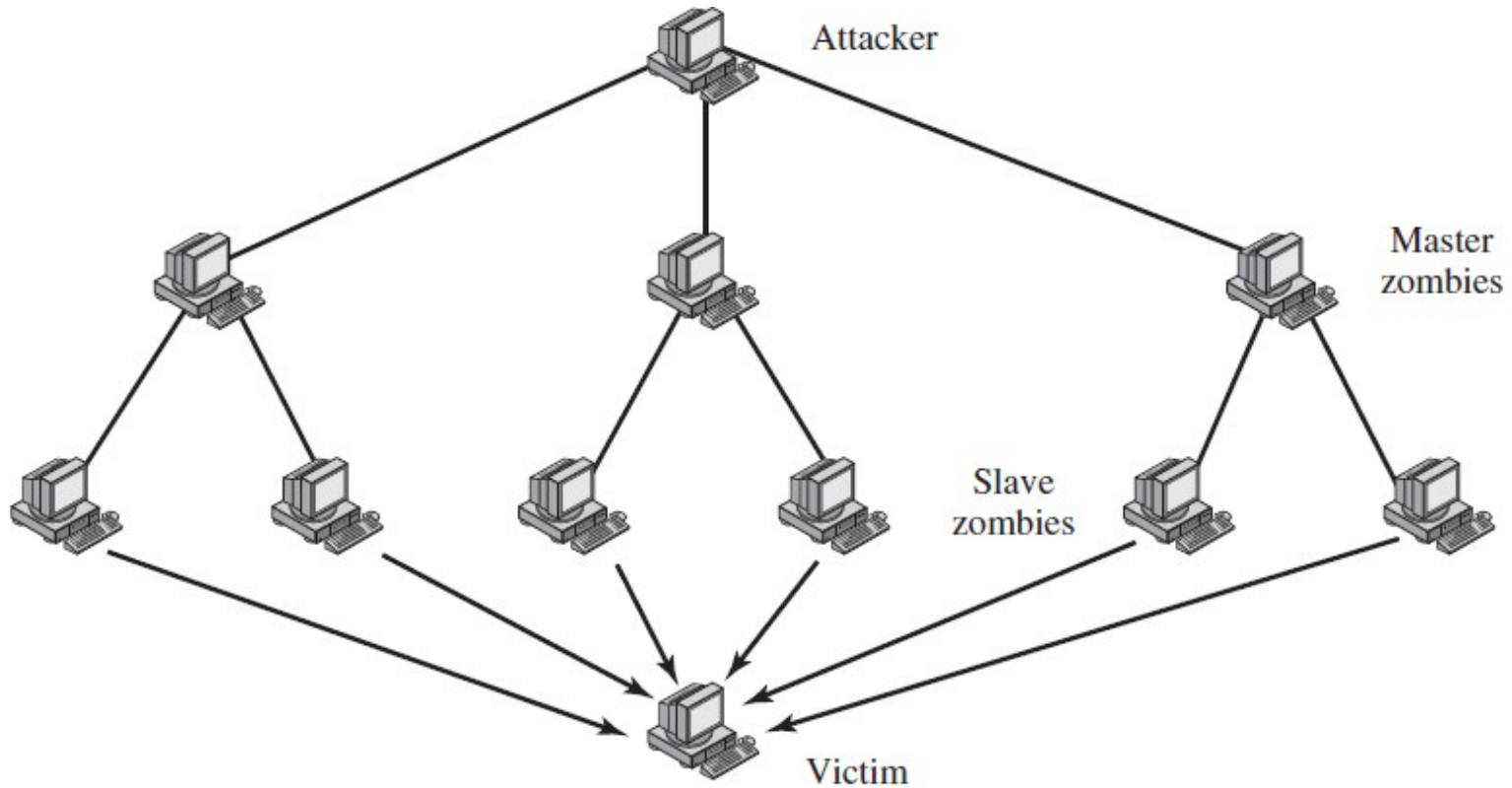


(a) Distributed ICMP attack

Figure 10.9 Examples of Simple DDoS Attacks



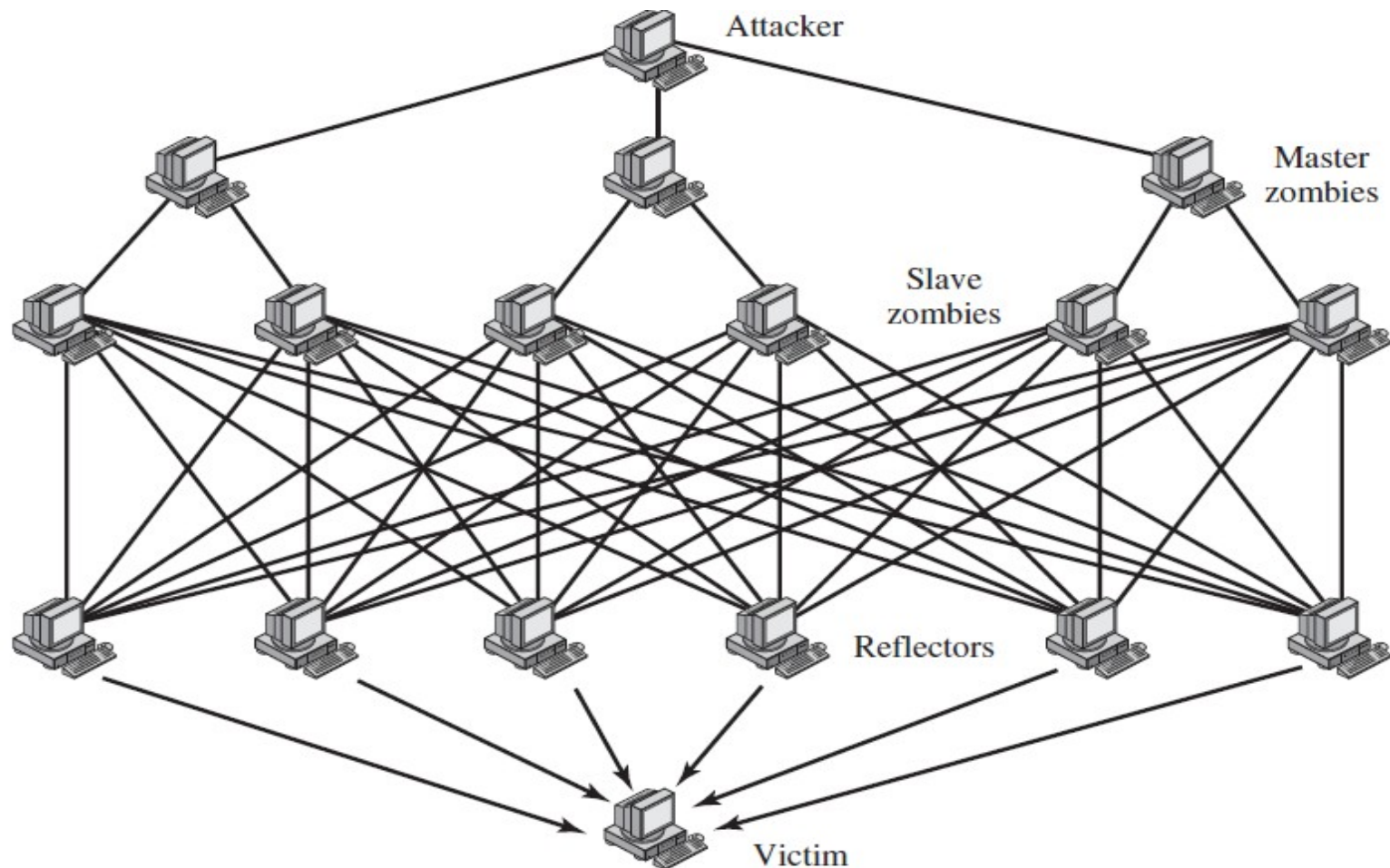
Direct DDoS attack



(a) Direct DDoS Attack



Reflector DDoS attack



(b) Reflector DDoS Attack

Figure 10.10 Types of Flooding-Based DDoS Attacks



DDoS Countermeasures

- **Attack prevention and preemption (before the attack):** enable the victim to endure attack attempts without denying service to legitimate clients.
- Techniques include enforcing policies for resource consumption and providing backup resources available on demand.
- In addition, prevention mechanisms modify systems and protocols on the Internet to reduce the possibility of DDoS attacks.



DDoS Countermeasures

- **Attack detection and filtering (during the attack):** attempt to detect the attack as it begins and respond immediately and minimizes the impact of the attack on the target.
- Detection involves looking for suspicious patterns of behavior. Response involves filtering out packets likely to be part of the attack.
- **Attack source traceback and identification (during and after the attack):** an attempt to identify the source of the attack as a first step in preventing future attacks.
- However, this method typically does not yield results fast enough, if at all, to mitigate an ongoing attack.

