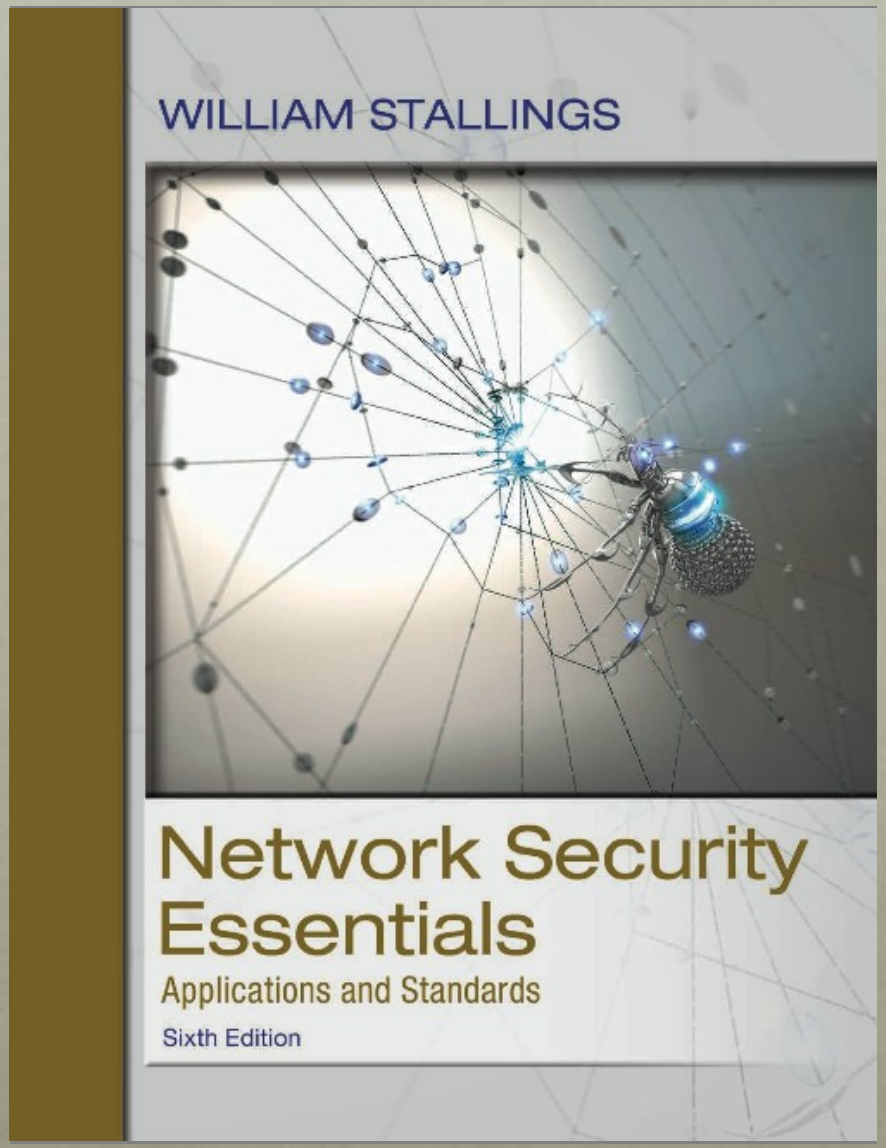
Network Security Essentials

Sixth Edition

by William Stallings



Chapter 10

Malicious Software

Table 10.1

Terminology for Malicious Software

(This table can be found on page 323 in the textbook.)

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 mechanisms 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.
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.

A Broad classification of malware

- Can be classified into two broad categories
  - How it spreads
  - What it does after infection

Propagation mechanisms:

- Include infection of existing executable or interpreted content by viruses that is subsequently spread to other system
- Exploit of software vulnerabilities either locally or over a network by worms or drive-by-downloads to allow the malware to replicate
- Social engineering attacks that convince users to bypass security mechanisms to install trojans or to respond to phishing attacks

Broad classification (continued)

- Earlier approaches to malware classification distinguished between:
  - Those that need a host program, being parasitic code such as viruses
  - Those that are independent, self-contained programs run on the system such as worms, trojans, and bots
- Another distinction used was:
  - Malware that does not replicate, such as trojans and spam e-mail
  - Malware that does, including viruses and worms
- Payload actions performed by malware once it reaches a target system can include:
  - Corruption of system or data files
  - Theft of service in order to make the system a zombie agent of attack as part of a botnet
  - Theft of information from the system, especially of logins, passwords, or other personal details by keylogging or spyware programs
  - Stealthing where the malware hides its presence on the system from attempts to detect and block it
- Blended attack
  - Uses multiple methods of infection or propagation to maximize the speed of contagion and the severity of the attack

Attack kits

- Initially the development and deployment of malware required considerable technical skill by software authors
- This changed with the development of virus-creation toolkits in the early 1990s and more general attack kits in the 2000s
  - These toolkits are often known as *crimeware*
  - Include a variety of propagation mechanisms and payload modules that even novices can combine, select, and deploy
  - Can easily be customised with the latest discovered vulnerabilities in order to exploit the window of opportunity between the publication of a weakness and the deployment of patches to close it
  - These kits greatly enlarged the population of attackers able to deploy malware

Attack sources

- Another significant malware development over the last couple of decades is the change from attackers being individuals to more organized and dangerous attack sources
  - These include politically motivated attackers, criminals, organized crime, organizations that sell their services to companies and nations, and national government agencies
- This has significantly changed the resources available and motivation behind the rise of malware leading to development of a large underground economy involving the sale of attack kits, access to compromised hosts, and to stolen information

Advanced persistent threat (apt)

- A well-resourced, persistent application of a wide variety of intrusion technologies and malware to selected targets, usually business or political
- APTs differ from other types of attack by their careful target selection, and persistent, often stealthy, intrusion efforts over extended periods
  - Aurora, RSA, APT1, and Stuxnet are often cited as examples
- Named as a result of these characteristics:
  - Advanced
    - The individual components may not necessarily be technically advanced, but are carefully selected to suit the chosen
  - Persistent
    - Determined application of the attacks over an extended period against the chosen target in order to maximize the chance of success
  - Threats
    - Threats to the selected targets as a result of the organized, capable, and well-funded attackers intent to compromise the specifically chosen targets

Viruses

- Parasitic software fragments that attach themselves to some existing executable content
- Can “infect” other programs or any type of executable content and modify them
- The modification includes injecting the original code with a routine to make copies of the virus code, which can then go on to infect other content
- One reason viruses dominated the malware scene in earlier years was the lack of user authentication and access controls on personal computer systems

Virus Structure

A computer virus and many contemporary types of malware includes one or more variants of each of these components:

- A computer virus, and more generally many contemporary types of malware, includes one or more variants of each of these components:

• Infection mechanism: The means by which a virus spreads or propagates, 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, sometimes known as a logic bomb .

• Payload: What the virus does, besides spreading. The payload may involve damage or benign but noticeable activity.

Virus phases

During its lifetime, a typical virus goes through the following four phases:

• 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 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.

• 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.

Most viruses that infect executable program files carry out their work in a manner that is specific to a particular operating system and, in some cases, specific to a particular hardware platform. Thus, they are designed to take advantage of the details and weaknesses of particular systems. Macro viruses, though, target specific document types, which are often supported on a variety of systems.

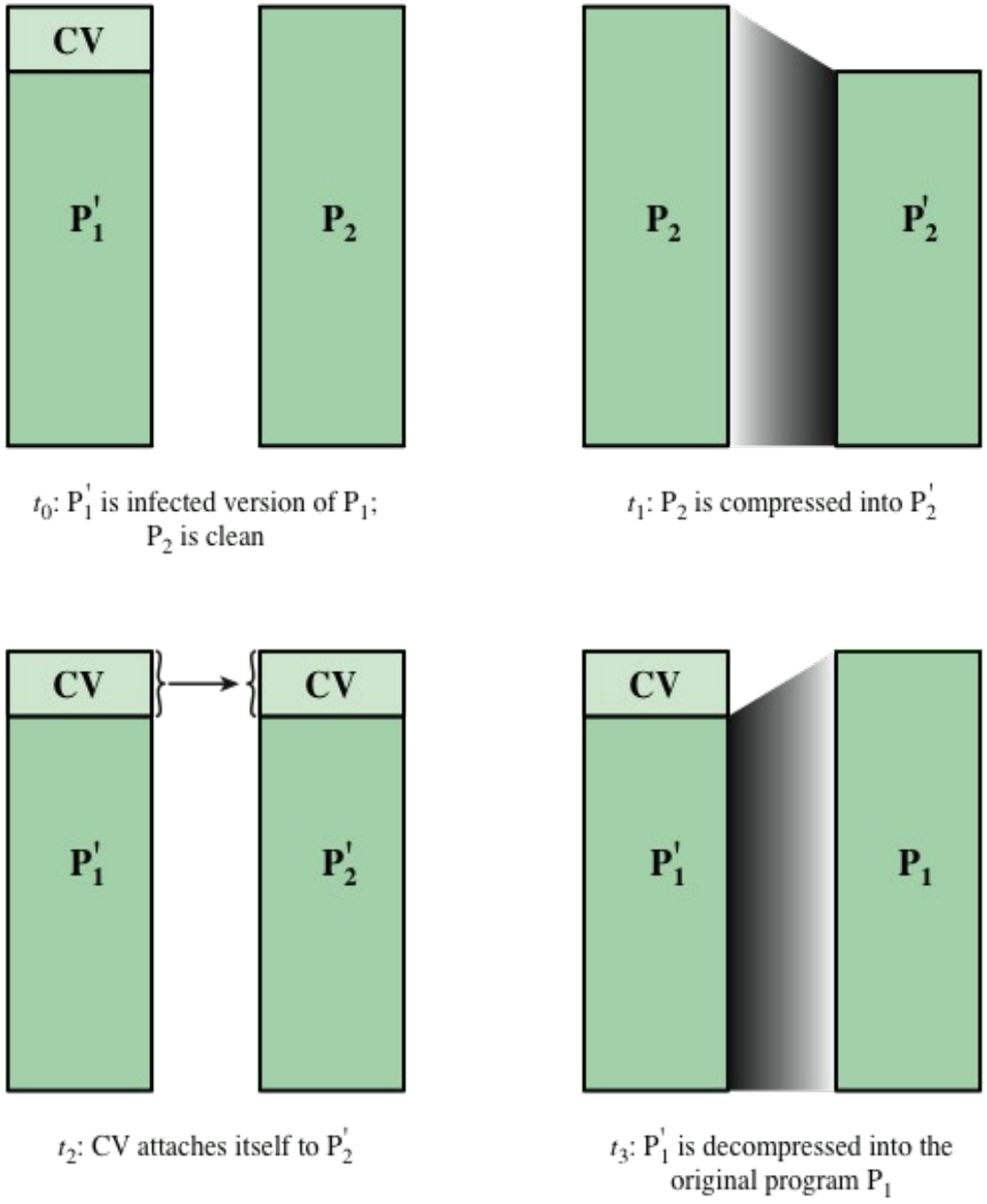
```
   program V :=

{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:

}
```

(a) A simple 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;}
        }
```

(b) A compression virus

Figure 10.1 Example Virus Logic



t_0: P_1' is infected version of P_1; P_2 is clean

t_1: P_2 is compressed into P_2'

t_2: CV attaches itself to P_2'

t_3: P_1' is decompressed into the original program P_1

Figure 10.2 A Compression Virus

Virus Classification by target

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 or scripting code that is interpreted by an application.

• Process attachment: Infect the process of an otherwise safe application. Remains in memory.

Virus classification by concealment strategy

- Includes the following categories:
  - Encrypted virus
    - Portion of the virus creates a random encryption key 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
    - Because the bulk of the virus is encrypted with a different key for each instance, there is no constant bit pattern to observe
  - Stealth virus
    - A form of virus explicitly designed to hide itself from detection by antivirus software
    - 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
    - Mutates with every infection
    - Rewrites itself completely at each iteration, increasing the difficulty of detection
    - May change their behavior as well as their appearance

Macro and scripting viruses

- Macro viruses infect scripting code used to support active content in a variety of user document types
- Threatening for a number of reasons:
  - A macro virus is platform independent
  - Macro viruses infect documents, not executable portions of code
  - Macro viruses are easily spread, as the documents they exploit are shared in normal use
  - Because macro viruses infect user documents rather than system programs, traditional file system access controls are of limited use in preventing their spread

worms

- A program that actively seeks out more machines to infect
  - Upon activation, the worm may replicate and propagate again
- To replicate itself, a worm uses some means to access remote systems:
  - Electronic mail or instant messenger facility
  - File sharing
  - Remote execution capability
  - Remote file access or transfer capability
  - Remote login capability

Worm phases

- A worm typically uses the same phases as a computer virus:
  - Dormant
  - Propagation
  - Triggering
  - Execution
- The propagation phase generally performs the following functions:
  - Search for appropriate access mechanisms to other systems to infect by examining host tables, address books, buddy lists, trusted peers, and other similar repositories of remote system access details
  - Use the access mechanisms found to transfer a copy of itself to the remote system and cause the copy to be run

Target discovery

- Scanning/fingerprinting
  - The function in the propagation phase for a network worm to search for other systems to infect
- Worm network scanning strategies:
  - Random
    - Each compromised host probes random addresses in the IP address space, using a different seed
    - Produces a high volume of Internet traffic, which may cause generalized disruption even before the actual attack is lunched
  - Hit list
    - The attacker first compiles a long list of potential vulnerable machines
    - Once the list is compiled, the attacker begins infecting machines on the list
    - Each infected machine is provided with a portion of the list to scan
    - This results in a very short scanning period, which may make it difficult to detect that infection is taking place
  - Topological
    - Uses information contained on an infected victim machine to find more hosts to scan
  - Local subnet
    - If a host is infected behind a firewall, that host then looks for targets in its own local network
    - The host uses the subnet address structure to find other hosts that would otherwise be protected by the firewall

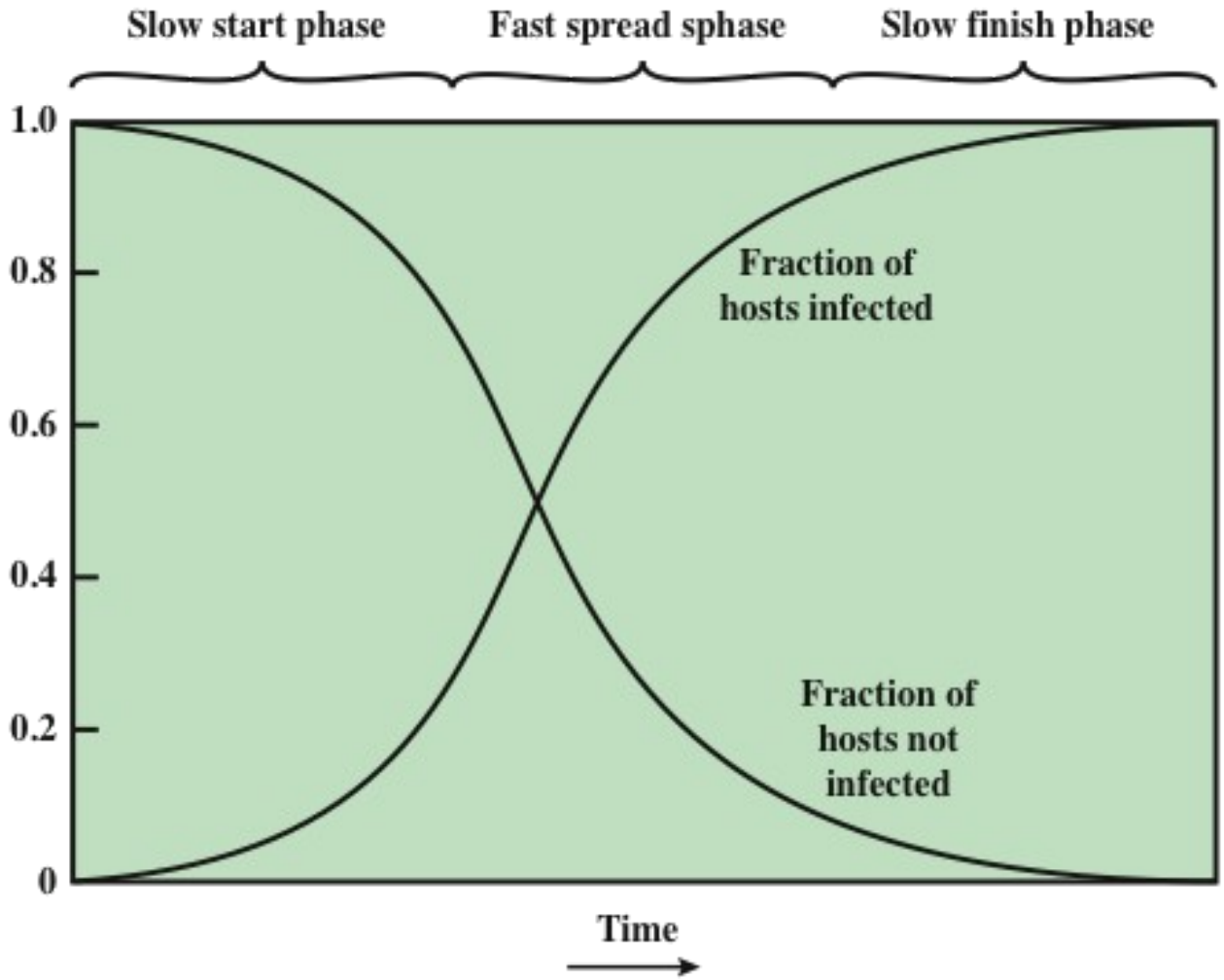


Figure 10.3 Worm Propagation Model

The morris worm

- Released onto the Internet by Robert Morris in 1988
- Designed to spread on UNIX systems and used a number of different techniques for propagation
- When a copy began execution its first task was to discover other hosts known to this host that would allow entry from this host
- For each discovered host, the worm tried a number of methods for gaining access:
  - It attempted to log on to a remote host as a legitimate user
  - It exploited a bug in the UNIX finger protocol, which reports the whereabouts of a remote user
  - It exploited a trapdoor in the debug option of the remote process that receives and sends mail

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, or exploit macro or scripting languages supported in popular document types.

• Multiexploit: New worms penetrate systems in a variety of ways, using exploits against Web servers, browsers, e-mail, file sharing, and other network-based applications, or via shared media.

• Ultrafast spreading: Exploit various techniques to optimize the rate of spread of a worm to maximize its likelihood of locating as many vulnerable machines as possible in a short time period.

• 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.

• 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 a wide variety of malicious payloads, such as distributed denial-of-service bots, rootkits, spam e-mail generators, and spyware.

• 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.

Mobile code

- Refers to programs that can be shipped unchanged to a heterogeneous collection of platforms and execute with identical semantics
- Transmitted from a remote system to a local system and then executed on the local system without the user's explicit instruction
- Often acts as a mechanism for a virus, worm, or Trojan horse to be transmitted to the user's workstation
- 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
  - Downloads from untrusted sites or of untrusted software

client-side vulnerabilities and Drive-by-downloads

- Drive-by-download
  - Exploits browser vulnerabilities so that when the user views a Web page controlled by the attacker, it contains code that exploits the browser bug to download and install malware on the system without the user's knowledge or consent
  - Does not actively propagate as a worm does, but rather waits for unsuspecting users to visit the malicious Web page in order to spread to their systems
- Watering-hole attacks are a variant of this used in highly targeted attacks
  - The attacker researches their intended victims to identify Web sites they are likely to visit and then scans theses sites to identify those with vulnerabilities that allow their compromise with a drive-by-download attack
- Malvertising is another technique used to place malware on Web sites without actually compromising them
  - The attacker pays for advertisements that are highly likely to be placed on their intended target Web sites, and which incorporate malware in them

clickjacking

- Also known as a *user-interface (UI) redress attack*
  - Is a vulnerability used by an attacker to collect an infected user's clicks
  - The attacker can force the user to do a variety of things from adjusting the user's computer settings to unwittingly sending the user to Web sites that might have malicious code
  - Also, by taking advantage of Adobe Flash or JavaScript, an attacker could even place a button under or over a legitimate button, making it difficult for users to detect
  - A typical attack uses multiple transparent or opaque layers to trick a user into clicking on a button or link on another page when they were intending to click on the top level page
- Using a similar technique, keystrokes can also be hijacked
  - With a carefully crafted combination of stylesheets, iframes, and text boxes, a user can be led to believe they are typing in the password to their e-mail or bank account but are instead typing into an invisible frame controlled by the attacker

spam

- Unsolicited bulk e-mail
- Imposes significant costs on both the network infrastructure needed to relay this traffic and on users who need to filter their legitimate e-mails
- Most recent spam is sent by botnets using compromised user systems
- Is a significant carrier of malware
- May be used in a phishing attack
- Although a significant security concern, in many cases it requires the user's active choice to view the e-mail and any attached document or to permit the installation of some program, in order for the compromise to occur

Trojan horses

- Is a useful, or apparently useful, program or utility containing hidden code that, when invoked, performs some unwanted or harmful function
- Can be used to accomplish functions indirectly that the attacker could not accomplish directly
- Fit into one of three models:

. Continuing to perform the function of the original program and additionally performing a separate malicious activity

• Continuing to perform the function of the original program but modifying the function to perform malicious activity (e.g., a Trojan horse version of a login program that collects passwords) or to disguise other malicious activity (e.g., a Trojan horse version of a process-listing program that does not display certain processes that are malicious)

• Performing a malicious function that completely replaces the function of the original program

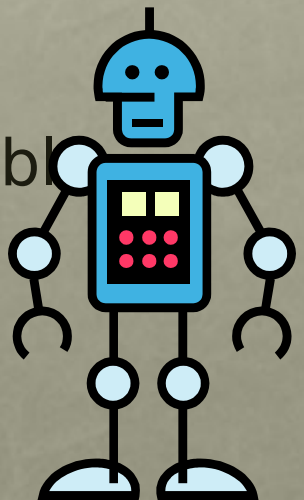
Some trojans avoid the requirement for user assistance by exploiting some software vulnerability to enable their automatic installation and execution. In this they share some features of a worm, but unlike it, they do not replicate. A prominent example of such an attack was the Hydraq Trojan used in Operation Aurora in 2009 and early 2010. This exploited a vulnerability in Internet Explorer to install itself and targeted several high-profile companies [SYMA13]. It was typically distributed either by spam e-mail or via a compromised Web site using a “drive-by-download.”

Payload – system corruption

- Once malware is active on the target system, the next concern is what actions it will take on this system
- Examples:
  - Data destruction on the infected system when certain trigger conditions were met
  - Display unwanted messages or content on the user's system when triggered
  - Encrypt the user's data and demand payment in order to access the key needed to recover this information (ransomware)
  - Inflict real-world damage on the system
    - Attempt to rewrite the BIOS code used to initially boot the computer
    - Target specific industrial control system software
  - Logic bomb
    - Code embedded in the malware that is set to "explode" when certain conditions are met

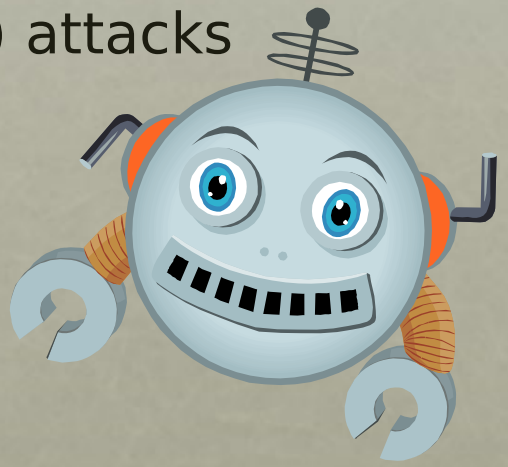
Payload – attack agent

- Malware subverts the computational and network resources of the infected system for use by the attacker
  - Bot (robot), zombie, drone
  - Secretly takes over another Internet-attached computer and then uses that computer to launch or manage attacks that are difficult to trace to the bot's creator
- A *botnet* is a collection of bots often capable of acting in a coordinated manner



Uses of bots

- Distributed denial-of-service (DDoS) attacks
- Spamming
- Sniffing traffic
- Keylogging
- Spreading new malware
- Installing advertisement add-ons and browser helper objects (BHOs)
- Attacking Internet Relay Chat (IRC) networks
- Manipulating online polls/games



Remote control facility

- Distinguishes a bot from a worm
  - A worm propagates itself and activates itself, whereas a bot is controlled from some central facility
- Typical means of implementing is on an IRC server
- More recent botnets use covert communication channels via protocols such as HTTP
- Distributed control mechanisms, using peer-to-peer protocols, are also used, to avoid a single point of failure
- Once a communications path is established between a control module and the bots, the control module can activate the bots
  - Can also issue update commands that instruct the bots to download a file from some Internet location and execute it

Payload – information theft

Typically, users send their login and password credentials to banking, gaming, and related sites over encrypted communication channels (e.g., HTTPS or POP3S), which protect them from capture by monitoring network packets. To bypass this, an attacker can install a keylogger , which captures keystrokes on the infected machine to allow an attacker to monitor this sensitive information. Since this would result in the attacker receiving a copy of all text entered on the compromised machine, keyloggers typically implement some form of filtering mechanism that only returns information close to desired keywords (e.g., “login” or “password” or “paypal.com”).

In response to the use of keyloggers, some banking and other sites switched to using a graphical applet to enter critical information, such as passwords. Since these do not use text entered via the keyboard, traditional keyloggers do not capture this information. In response, attackers developed more general spyware payloads, which subvert the compromised machine to allow monitoring of a wide range of activity on the system. This may include monitoring the history and content of browsing activity, redirecting certain Web page requests to fake sites controlled by the attacker, dynamically modifying data exchanged between the browser and certain Web sites of interest. All of which can result in significant compromise of the user’s personal information.

Payload – information theft

Another approach used to capture a user's login and password credentials is to include a URL in a spam e-mail that links to a fake Web site controlled by the attacker, but which mimics the login page of some banking, gaming, or similar site. This is normally included in some message suggesting that urgent action is required by the user to authenticate his or her account, to prevent it being locked. If the user is careless, and doesn't realize that he or she is being conned, then following the link and supplying the requested details will certainly result in the attackers exploiting the user's account using the captured credentials.

More generally, such a spam e-mail may direct a user to a fake Web site controlled by the attacker or to complete some enclosed form and return to an e-mail accessible to the attacker, which is used to gather a range of private, personal, information on the user. Given sufficient details, the attacker can then "assume" the user's identity for the purpose of obtaining credit or sensitive access to other resources. This is known as a phishing attack, which exploits social engineering to leverage user's trust by masquerading as communications from a trusted source [GOLD10].

Such general spam e-mails are typically widely distributed to very large numbers of users, often via a botnet. While the content will not match appropriate trusted sources for a significant fraction of the recipients, the attackers rely on it reaching sufficient users of the named trusted source, a gullible portion of whom will respond, for it to be profitable.

A more dangerous variant of this is the spear-phishing attack. This again is an e-mail claiming to be from a trusted source. However, the recipients are carefully researched by the attacker, and each e-mail is carefully crafted to suit its recipient specifically, often quoting a range of information to convince him or her of its authenticity. This greatly increases the likelihood of the recipient responding as desired by the attacker.

Payload – stealthing

- Backdoor
  - Is a secret entry point into a program that allows someone who is aware of the backdoor to gain access without going through the usual security access procedures
  - 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
  - Usually implemented as a network service listening on some nonstandard port that the attacker can connect to and issue commands through to be run on the compromised system

Payload – stealthing

- Rootkit
  - A set of programs installed on a system to maintain covert access to that system with administrator (or root) privileges, while hiding evidence of its presence to the greatest extent possible
  - Alters the host's standard functionality in a malicious and stealthy way
  - An attacker has complete control of the system and can add or change programs and files, monitor processes, send and receive network traffic, and get backdoor access on demand
  - Hides by subverting the mechanisms that monitor and report on the processes, files, and registries on a computer

rootkits

A rootkit can be classified using the following characteristics:

• Persistent: Activates each time the system boots. The rootkit must store code in a persistent store, such as the registry or file system, and configure a method by which the code executes without user intervention. This means it is easier to detect, as the copy in persistent storage can potentially be scanned.

• Memory based: Has no persistent code and therefore cannot survive a reboot. However, because it is only in memory, it can be harder to detect.

• User mode: Intercepts calls to APIs (application program interfaces) and modifies returned results. For example, when an application performs a directory listing, the return results don't include entries identifying the files associated with the rootkit.

• Kernel mode: Can intercept calls to native APIs in kernel mode. The rootkit can also hide the presence of a malware process by removing it from the kernel's list of active processes.

• Virtual machine based: This type of rootkit installs a lightweight virtual machine monitor and then runs the operating system in a virtual machine above it. The rootkit can then transparently intercept and modify states and events occurring in the virtualized system.

• External mode: The malware is located outside the normal operation mode of the targeted system, in BIOS or system management mode, where it can directly access hardware.

This classification shows a continuing arms race between rootkit authors, who exploit ever more stealthy mechanisms to hide their code, and those who develop mechanisms to harden systems against such subversion or to detect when it has occurred.

countermeasures

- One of the first countermeasures that should be employed is to ensure all systems are as current as possible, with all patches applied, in order to reduce the number of vulnerabilities that might be exploited on the system
- The next is to set appropriate access controls on the applications and data stored on the system, to reduce the number of files that any user can access, and hence potentially infect or corrupt, as a result of them executing some malware code
- The third common propagation mechanism, which targets users in a social engineering attack, can be countered using appropriate user awareness and training

Malware countermeasure approaches

- If prevention fails, then technical mechanisms can be used to support the following threat mitigation options:
  - Detection
  - Identification
  - Removal
- Requirements for effective malware countermeasures:
  - Generality
  - Timeliness
  - Resiliency
  - Minimal denial-of-service costs
  - Transparency
  - Global and local coverage

Host-based scanners

Four generations of antivirus software:

- **First generation: Simple scanners**
- **Second generation: Heuristic scanners**
- **Third generation: Activity traps**
- **Fourth generation: Full-featured protection**

A first-generation scanner requires a malware signature to identify the malware. The signature may contain "wildcards" but matches essentially the same structure and bit pattern in all copies of the malware. Such signature-specific scanners are limited to the detection of known malware. Another type of first-generation scanner maintains a record of the length of programs and looks for changes in length as a result of virus infection.

A second-generation scanner does not rely on a specific signature. Rather, the scanner uses heuristic rules to search for probable malware instances. One class of such scanners looks for fragments of code that are often associated with malware. For example, a scanner may look for the beginning of an encryption loop used in a polymorphic virus and discover the encryption key. Once the key is discovered, the scanner can decrypt the malware to identify it, and then remove the infection and return the program to service.

Another second-generation approach is integrity checking. A checksum can be appended to each program. If malware alters or replaces some program without changing the checksum, then an integrity check will catch this change. To counter malware that is sophisticated enough to change the checksum when it alters a program, an encrypted hash function can be used. The encryption key is stored separately from the program so that the malware cannot generate a new hash code and encrypt that. By using a hash function rather than a simpler checksum, the malware is prevented from adjusting the program to produce the same hash code as before. If a protected list of programs in trusted locations is kept, this approach can also detect attempts to replace or install rogue code or programs in these locations.

Third-generation programs are memory-resident programs that identify malware by its actions rather than its structure in an infected program. Such programs the advantage that it is not necessary to develop signatures and heuristics for a wide array of malware. Rather, it is necessary only to identify the small set of actions that indicate that malicious activity is being attempted and then to 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 malware to penetrate a system and then limits the ability of a malware to update files in order to propagate.

The arms race continues. With fourth-generation packages, a more comprehensive defense strategy is employed, broadening the scope of defense to more general-purpose computer security measures. These include more sophisticated antivirus approaches. We now highlight two of the most important.

Host-based behavior-blocking software

- Integrates with the operating system of a host computer and monitors program behavior in real time for malicious actions
- The software then blocks potentially malicious actions before they have a chance to affect the system
- Can block suspicious software in real time so it has an advantage over antivirus detection techniques such as fingerprinting or heuristics
- 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

Perimeter scanning approaches

- Antivirus software is used on an organization's firewall and IDS
  - Typically included in e-mail and Web proxy services running on these systems
  - May also be included in the traffic analysis component of an IDS

Two types of monitoring software may be used:

- Ingress monitors: These are 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 malware traffic. An example of a detection technique for an ingress monitor is to look for incoming traffic to unused local IP addresses.

- 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. In the former case, the egress monitor can be part of the egress-filtering software of a LAN router or switch. As with ingress monitors, the external firewall or a honeypot can house the monitoring software. Indeed, the two types of monitors can be collocated. The egress monitor is designed to catch the source of a malware attack by monitoring outgoing traffic for signs of scanning or other suspicious behavior

Perimeter worm countermeasures

A. Signature-based worm scan filtering: This type of approach generates a worm signature, which is then used to prevent worm scans from entering/leaving a network/host. Typically, this approach involves identifying suspicious flows and generating a worm signature. This approach is vulnerable to the use of polymorphic worms: 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. [NEWS05] is an example of this approach.

B. Filter-based worm containment: This approach is similar to class A but focuses on worm content rather than a scan signature. The filter checks a message to determine if it contains worm code. An example is Vigilante [COST05], which relies on collaborative worm detection at end hosts. This approach can be quite effective but requires efficient detection algorithms and rapid alert dissemination.

C. Payload-classification-based worm containment: These network-based techniques 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. An example of this approach, which looks for exploit code in network flows, is reported in [CHIN05]. This approach does not generate signatures based on byte patterns but rather looks for control and data flow structures that suggest an exploit.

Continued . . .

Perimeter worm countermeasures

D. Threshold random walk (TRW) scan detection: TRW exploits randomness in picking destinations to connect to as a way of detecting if a scanner is in operation [JUNG04]. TRW is suitable for deployment in high-speed, lowcost network devices. It is effective against the common behavior seen in worm scans.

E. Rate limiting: This class limits the rate of scanlike traffic from an infected host. Various strategies can be used, including limiting the number of new machines a host can connect to in a window of time, detecting a high connection failure rate, and limiting the number of unique IP addresses a host can scan in a window of time. [CHEN04] is an example. 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: This approach immediately blocks outgoing traffic when a threshold is exceeded either in outgoing connection rate or in diversity of connection attempts [JHI07]. The approach must include measures to quickly unblock mistakenly blocked hosts in a transparent way. 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. Rate halting appears to offer a very effective countermeasure. As with rate limiting, rate-halting techniques are not suitable for slow, stealthy worms.

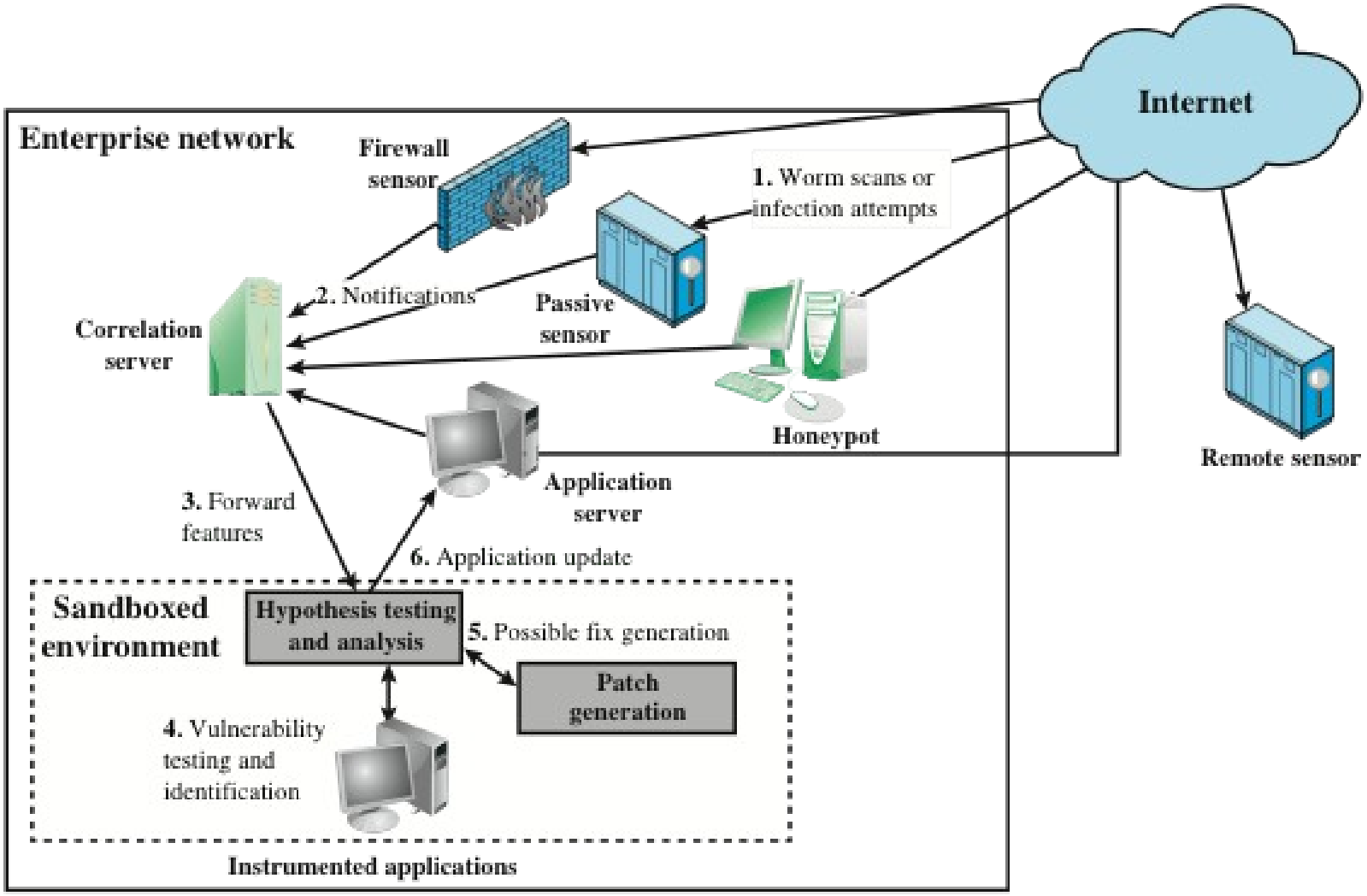
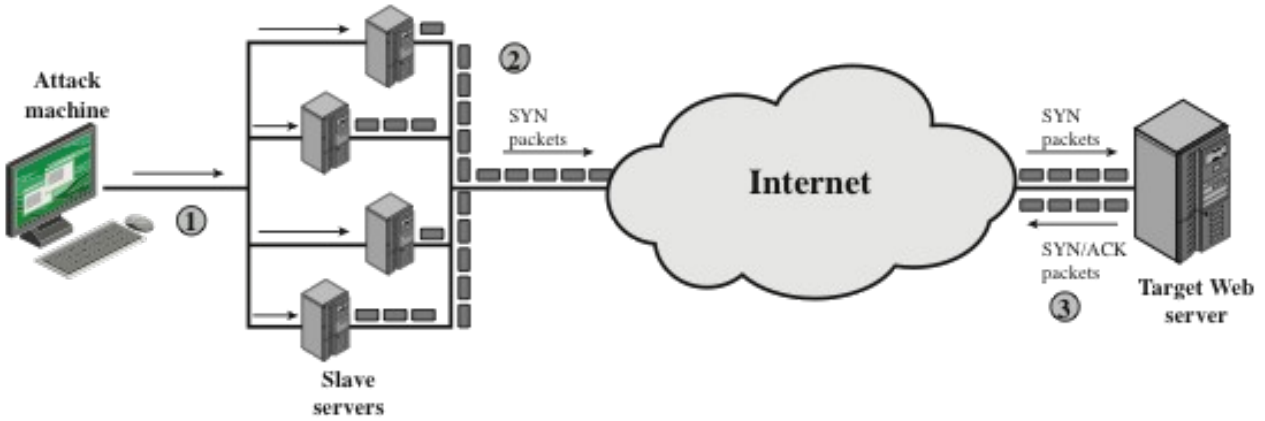


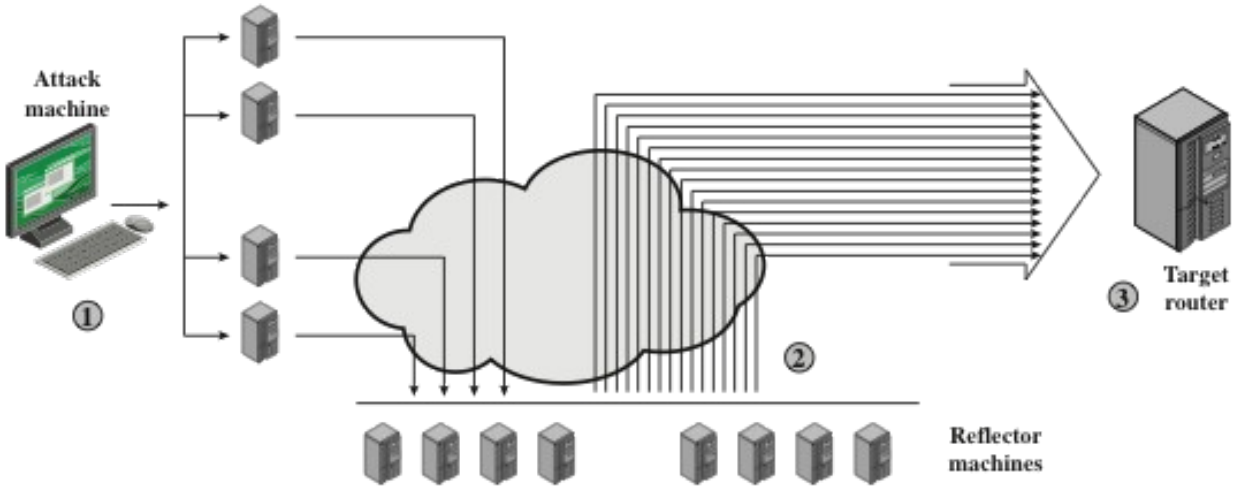
Figure 10.4 Placement of Worm Monitors

Distributed Denial of Service Attacks (DDOS)

- Attacks that make computer systems inaccessible by flooding servers, networks, or even end-user systems with useless traffic so that legitimate users can no longer gain access to those resources
- One way to classify DDoS attacks is in terms of the type of resources that is consumed
- The resource consumed is either an internal host resource on the target system or data transmission capacity in the local network to which the target is attacked

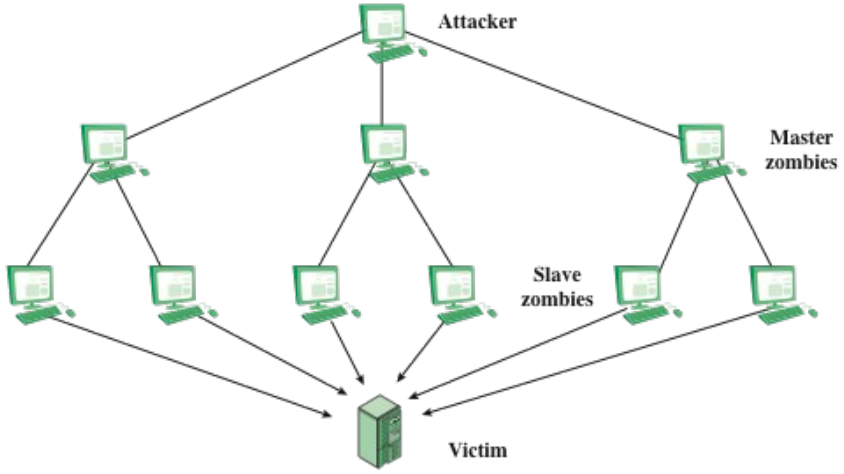


(a) Distributed SYN flood attack

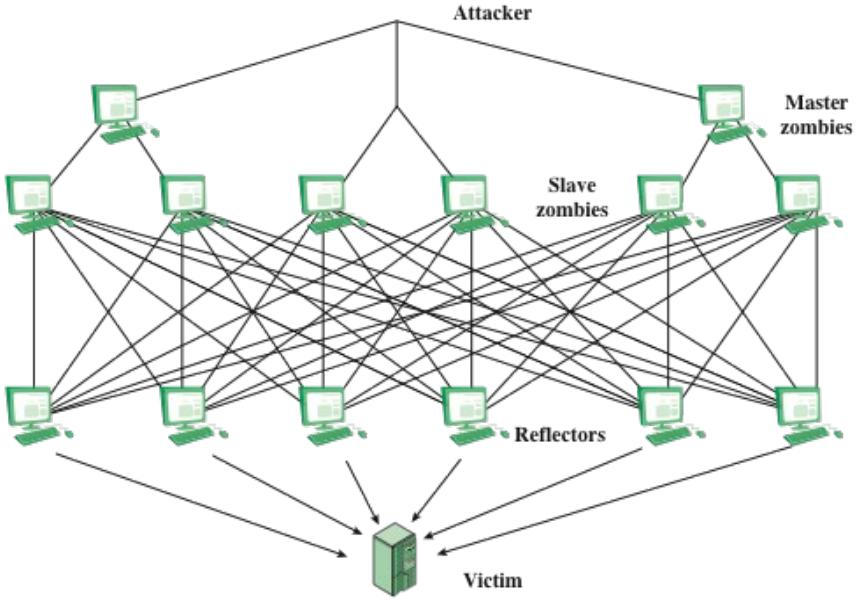


(a) Distributed ICMP attack

Figure 10.5 Examples of Simple DDoS Attacks



(a) Direct DDoS Attack



(b) Reflector DDoS Attack

Figure 10.6 Types of Flooding-Based DDoS Attacks

Constructing the Attack Network

- The first step in a DDoS attack is for the attacker to infect a number of machines with zombie software that will ultimately be used to carry out the attack
- Essential ingredients:
  - Software that can carry out the DDoS attack
  - A vulnerability in a large number of systems
  - A strategy for locating vulnerable machines (*scanning)*
- Scanning strategies:
  - Random
    - Each compromised host probes random addresses in the IP address space, using a different seed
  - Hit list
    - The attacker first compiles a long list of potential vulnerable machines
  - Topological
    - This method uses information contained on an infected victim machine to find more hosts to scan
  - Local subnet
    - If a host is infected behind a firewall, that host then looks for targets in its own local network

DDoS Countermeasures

In general, there are three lines of defense against DDoS attacks [CHAN02]:

• Attack prevention and preemption (before the attack): These mechanisms 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.

• Attack detection and filtering (during the attack): These mechanisms attempt to detect the attack as it begins and respond immediately. This 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): This is 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.

The challenge in coping with DDoS attacks is the sheer number of ways in which they can operate. Thus, DDoS countermeasures must evolve with the threat.

Summary

- Types of malicious software (malware)
- Advanced persistent threats
- Propagation:
  - Infected content – viruses
  - Vulnerability exploit – worms
  - Social engineering – spam e-mail, trojans
- Payload:
  - Attack agent – zombie, bots
  - Information theft – keyloggers, phishing, spyware
  - Stealthing – backdoors, rootkits
- Countermeasures
- DDoS attacks