الجامعة التكنولوجية – قسم علوم الحاسوب
فرع امنية الحاسوب والامن السيبراني
مادة الشفرات الخبيثة - الفصل الاول
المرحلة الثالثة -٢٠٢٤/٢٠٢٥

أ.د. اخلاص خلف

Syllabus of malicious software/Third class-1st course

- Definition of malware
- Types of Malicious software
  - Backdoor
  - Logic Bomb
  - Trojan Horses
  - Mobile code
  - Multiple-Threat Malware
- Viruses
  - The Nature of Viruses
  - Initial Infection
  - Viruses Classification
    - A virus classification by Target
      - ✓ Boot sector infector
      - ✓ File infector
      - ✓ Macro virus
    - A virus classification by Concealment strategy
      - ✓ No Concealment
      - ✓ Encryption
      - ✓ Stealth
      - ✓ Oligomorphism
      - ✓ Polymorphism
      - ✓ Metamorphism
      - ✓ Strong Encryption
  - Virus Kits
  - Macro Viruses
  - E-Mail Viruses
- VIRUS COUNTERMEASURES
- Antivirus Approaches
  1. Generations of antivirus software (Static methods)
     - ✓ First generation: simple scanners
     - ✓ Second generation: heuristic scanners

- ✓ Third generation: activity traps
- ✓ Fourth generation: full-featured protection
- ✓ Fifth generation: Integrity Checkers

2. Advanced Antivirus Techniques (Dynamic Methods)
   - ✓ 1 .Generic Decryption.
   - ✓ 2 .Digital Immune System.
   - ✓ 3. Behavior Blocking Software

- Comparison of Anti-Virus Detection Techniques:
- Verification, Quarantine, and Disinfection:
- Virus Databases and Virus Description Languages
- Worms
  - ✓ The Morris Worm
  - ✓ Worm Propagation Model
  - ✓ Recent Worm Attacks:
  - ✓ State of Worm Technology
  - ✓ Mobile Phone Worms:
  - ✓ Worm Countermeasures
  - ✓ NETWORK-BASED WORM DEFENSE
- Distributed Denial of Service Attacks:
  - ✓ DDoS Attack Description
  - ✓ Constructing the Attack Network
  - ✓ Types of scanning strategies
  - ✓ DDoS Countermeasures

References

1- Cryptography And Network Security Principles And Practice Fifth Edition William Stallings

2- Computer Viruses and Malware by John Ay cock University of Calgary Canada, 2006 Springer

Definition: -

Malware – short for malicious software – is software used or programmed by attackers to disrupt computer operation, gather sensitive information or gain unauthorized access to computers.

Malicious software or Malware is a software that designed for harm, exploit, or compromise the functionality, security of computer system, networks or devices some examples of malware are backdoor, trojan horse, worms , e - mail viruses, spyware,

TYPES OF MALICIOUS SOFTWARE

Malicious software can be divided into two categories: those that need a host program, and those that are independent. The former, referred to as **parasitic**, are essentially fragments of programs that cannot exist independently of some actual application program, utility, or system program. Viruses, logic bombs, and backdoors are examples. **Independent malware** is a self-contained program that can be scheduled and run by the operating system.Worms and bot programs are examples.

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. Examples are 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.Viruses and worms are examples.

Backdoor

A **backdoor**, also known as a **trapdoor**, 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. Programmers have used backdoors legitimately for many years to debug and test programs; such a backdoor is called a **maintenance hook**. This usually is done when the programmer is developing an application that has an authentication procedure, or a long setup, requiring the user to enter many different values to run the application. To debug the

program, the developer may wish to gain special privileges or to avoid all the necessary setup and authentication. The programmer may also want to ensure that there is a method of activating the program should something be wrong with the authentication procedure that is being built into the application. 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.

- A backdoor is a means to access a computer system or encrypted data that bypasses the system's customary security mechanisms. Web server backdoors are used for a number of malicious activities, including:
  - Data theft
  - Website defacing
  - Server hijacking
  - Infecting website visitors

How does a backdoor work?

Backdoors are used by hackers to gain access to a device by circumventing security mechanisms. Often time's developers install backdoors as a means of troubleshooting their program, but this also leaves a gap for hackers to exploit. The term is often used to describe vulnerabilities put in place on purpose, for example, to allow government surveillance groups to access citizens' smart phones and computers.

Logic Bomb

One of the oldest types of program threat, predating viruses and worms, is the 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.

Logic bombs are often used with **viruses**, worms, and Trojan horses to time them to do maximum damage before being noticed. For example, a programmer may hide a piece of code that starts deleting files.

Trojan Horses

A Trojan horse is a useful, or apparently 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. For example, to gain access to the files of another user on a shared system, a user could create a Trojan horse program that, when executed, changes the invoking user's file permissions so that the files are readable by any user. The author could then induce users to run the program by placing it in a common directory and naming it such that it appears to be a useful utility program or application. An example is a program that ostensibly produces a listing of the user's files in a desirable format. After another user has run the program, the author of the program can then access the information in the user's files. An example of a Trojan horse program that would be difficult to detect is a compiler that has been modified to insert additional code into certain programs as they are compiled, such as a system login program . The code creates a backdoor in the login program that permits the author to log on to the system using a special password.This Trojan horse can never be discovered by reading the source code of the login program.

Trojan horses 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.

Mobile Code

Mobile code often acts as a mechanism for a virus, worm, or Trojan horse to be transmitted to the user's workstation. In other cases, mobile code takes advantage of vulnerabilities to perform its own exploits, such as unauthorized data access.

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.

Another definition - Malicious mobile code (MMC) is any software program designed to move from computer to computer and network to network, in order to intentionally modify computer systems without the consent of the owner or operator. MMC includes viruses, Trojan horses, and worms.

Mobile code is any program or application capable of movement while embedded in an email, document or website. Mobile code uses network or storage media, such as a Universal Serial Bus (USB) flash drive. The term is often used in a malicious context; mobile code creates varying degrees of computer and system damage. Mobile code is usually downloaded via the body of an HTML email or email attachment. Mobile code is also known as executable content, remote code and active capsules.

Multiple-Threat Malware

Viruses and other malware may operate in multiple ways. The terminology is far from uniform; this subsection gives a brief introduction to several related concepts that could be considered multiple-threat malware.

A **multipartite** virus infects in multiple ways.Typically, the multipartite virus is capable of infecting multiple types of files, so that 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.An example of a blended attack is the Nimda attack, erroneously referred to as simply a worm.

Nimda uses four distribution methods:

- ✓ **E-mail**: A user on a vulnerable host opens an infected e-mail attachment; Nimda looks for e-mail addresses on the host and then sends copies of itself to those addresses.
- ✓ **Windows shares:** Nimda scans hosts for unsecured Windows file shares; it can then use NetBIOS86 as a transport mechanism to infect files on that host.
- ✓ **Web servers**: Nimda scans Web servers, looking for known vulnerabilities in Microsoft IIS. If it finds a vulnerable server, it attempts to transfer a copy of,itself to the server and infect it and its files.
- ✓ **Web clients**: If a vulnerable Web client visits a Web server that has been infected by Nimda, the client's workstation will become infected**.**

Viruses

The Nature of Viruses

A computer virus is a piece of software that can "infect" other programs by modifying them; the modification includes injecting the original program with a routine to make copies of the virus program, which can then go on to infect other programs. Computer viruses first appeared in the early 1980s,

Biological viruses are tiny scraps of genetic code—DNA or RNA—that can take over the machinery of a living cell and trick it into making thousands of flawless replicas of the original virus. Like its biological counterpart, a computer virus carries in its instructional code the recipe for making perfect copies of itself. The typical virus becomes embedded in a program on a computer. Then, whenever the infected computer comes into contact with an uninfected piece of software, a fresh copy of the virus passes into the new program. Thus, the infection can be spread from computer to computer by unsuspecting users who either swap disks or send programs to one another over a network. In a network environment, the

ability to access applications and system services on other computers provides a perfect culture for the spread of a virus.

A virus can do anything that other programs do.The difference is that a virus attaches itself to another program and executes secretly when the host program is run. 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.

A computer virus has three parts :

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

Typical virus goes through the following four phases During its lifetime:

1. **Dormant phase:** The virus is idle. The virus will eventually be activated b 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.
2. **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 propagation 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.
3. **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.
4. **Execution phase:** The function is performed. The function may be harmless such as a message on the screen, or damaging, such as the destruction o programs and data files.

Initial Infection:

Once a virus has gained entry to a system by infecting a single program, it is in a position to potentially infect some or all other executable files on that system when the infected program executes. Thus, viral infection can be completely prevented by preventing the virus from gaining entry in the first place. Unfortunately, prevention is extraordinarily difficult because a virus can be part of any program outside a system. In general, 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. In contrast, while it is easy enough to write a machine code virus for UNIX systems, they were almost never seen in practice because the existence of access controls on these systems prevented effective propagation of the virus. Traditional machine code based viruses are now less prevalent, because modern PC OSs do have more effective access controls.

Viruses Classification

There has been a continuous arms race between virus writers and writers of antivirus software since viruses first appeared. As effective countermeasures are developed for existing types of viruses.

Viruses are classified into two orthogonal axes:

1. the type of **Target** the virus tries to infect .
2. by Concealment **strategy** which is a method the virus uses to conceal itself from detection by users and antivirus software.

1- A virus classification by Target includes the following categories:

1. **Boot sector infector:** Infects a master boot record or boot record and spreads when a system is booted from the disk containing the virus.
2. **File infector:** Infects files that the operating system or shell consider to be executable.
3. **Macro virus:** Infects files with macro code that is interpreted by an application.

2- A virus classification by Concealment strategy includes the following categories:

1. **No Concealment:** Not hiding at all is one concealment strategy which is remarkably easy to implement in a computer virus. It goes without saying, however, that it's not very effective - once the presence of a virus is known, it's trivial to detect and analyze.

2. **Encryption:** A typical approach is as follows. A portion of the virus creates a rando 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.
3. **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. One example of a **stealth virus** was discussed earlier: a virus that uses compression so that the infected program is exactly the same length as an uninfected version. Far more sophisticated techniques are possible. For example, a virus can place intercept logic in disk I/O routines, so that when there is an attempt to read suspected portions of the disk using these routines, the virus will present back the original, uninfected program.Thus, *stealth* is not a term that applies to a virus as such but, rather, refers to a technique used by a virus to evade detection.

4. **Oligomorphism**: Assuming an encrypted virus' key is randomly changed with each new infection, the only unchanging part of the virus is the code in the decryptor loop. Anti-virus software will exploit this fact for detection, so the next logical development is to change the decryptor loop's code with each infection.

   An *oligomorphic* virus, or *semi-polymorphic* virus, is an encrypted virus which has a small, finite number of different decryptor loops at its disposal. The virus selects a new decryptor loop from this pool for each new infection. For example, Whale had 30 different decryptor variants, and Memorial had 96 decryptors. In terms of detection, oligomorphism only makes a virus marginally harder to spot. Instead of looking for one decryptor loop for the virus, anti-virus software can simply have all of the virus' possible decryptor loops enumerated, and look for them all.

5. **Polymorphic virus**: A virus that mutates with every infection, making detection by the "signature" of the virus impossible and harder. A **polymorphic virus** creates copies during

replication that are functionally equivalent but have distinctly different bit patterns. As with a stealth virus, the purpose is to defeat programs that scan for viruses. In this case, the "signature" of the virus will vary with each copy. To achieve this variation, the virus may randomly insert superfluous instructions or interchange the order of independent instructions. A more effective approach is to use encryption.The strategy of the encryption virus is followed. The portion of the virus that is responsible for generating keys and performing encryption/decryption is referred to as the *mutation engine*. The mutation engine itself is altered with each use.

6. **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.
7. **Strong Encryption:** The encryption methods discussed so far result in viruses that, once captured, are susceptible to analysis. The major problem is not the encryption method, because that can always be strengthened; the major problem is that viruses carry their decryption keys with them.

   This might seem a necessary weakness, because if a virus doesn't have its key, it can't decrypt and run its code. There are, **however, two other possibilities.**

1. The key comes from outside an infected system:

- A virus can retrieve the key from a web site, but that would mean that the virus would then have to carry the web site's address with it, which could be blocked as a countermeasure. To avoid knowing a specific web site's name, a virus could use a web search engine to get the key instead.
- A binary virus is one where the virus is in two parts, and doesn't become virulent until both pieces are present on a system. There have only been a few binary viruses, such as Dichotomy and RMNS.

  One manifestation of binary viruses would be where virus V1 has strongly encrypted code, and virus V2 has its key. But this scheme is unlikely to work well in practice. If V1 and V2

travel together, then both will bear the same risk of capture and analysis, defeating the purpose of separating the encryption key. If V1 and V2 spread separately (e.g., V2 is released a month after V1, and uses a different infection vector) then their spread would be independent.

2. The key comes from inside an infected system. Using environmental key generation, the decryption key is constructed of elements already present in the target's environment, like:

- the machine's domain name
- the time or date
- some data in the system (e.g., file contents)
- the current user name
- the interface's language setting (e.g., Chinese, Hebrew).

This makes it very easy to target viruses to particular individuals or groups. A target doesn't even know that they possess the key.

Combined with strong encryption, environmental key generation would render a virus unanalyzable even if captured. To fully analyze an encrypted virus, it has to be decrypted, and while the elements comprising the key may be discovered, the exact value of the key will not. In this case, the only real hope of decryption lies in a poor choice of key. A poorly-chosen key with a relatively small range of possible values (e.g., the language setting) would be susceptible to a brute-force attack. method is to catch exceptions that invalid code may cause, then try to run the decrypted "code" and see if it works.

Virus Kits

Another weapon in the virus writers' armory is the virus-creation toolkit. Such a toolkit enables a relative novice to quickly create a number of different viruses. Although viruses created with toolkits tend to be less sophisticated than viruses 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 take advantage of a feature found in Word and other office applications such as Microsoft Excel, namely the macro. In essence, a macro is an executable program embedded in a word processing document or other type of file. Typically, users employ macros to automate repetitive tasks and thereby save keystrokes. The macro language is usually some form of the Basic programming language.A user might define a sequence of keystrokes in a macro and set it up so that the macro is invoked when a function key or special short combination of keys is input.

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. As in other types o viruses, the arms race continues in the field of macro viruses, but they no longer are the predominant virus threat.

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

- ✓ The e-mail virus sends itself to everyone on the mailing list in the user's e-mail package.
- ✓ The virus does local damage on the user's system.

In 1999, a more powerful version of the e-mail virus appeared. This newer version can be activated merely by opening an e-mail that contains the virus rather than opening an attachment. The virus uses the Visual Basic scripting language supported by the e-mail package.

Thus, we see a new generation of malware that arrives via e-mail and uses e-mail software features to replicate itself across the Internet. 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.This makes it very difficult for antivirus software to respond before much damage is done.
Ultimately, a greater degree of security must be built into Internet utility and application software on PCs to counter the growing threat.

Melissa is a fast-spreading macro virus that is distributed as an e-mail attachment that, when opened, it targeted Microsoft Word system, and if the user has the Microsoft Outlook e-mail program, causes the **virus** to be resent to the first 50 people in each of the user's address books.

VIRUS COUNTERMEASURES

1- **Antivirus Approaches**

The ideal solution to the threat of viruses is prevention: Do not allow a virus to get into the system in the first place, or block the ability of a virus to modify any files containing executable code or macros.This goal is, in general, impossible to achieve,

although prevention can reduce the number of successful viral attacks.The next best approach is to be able to do the following:

- **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 virus cannot spread further.

If detection succeeds but either identification or removal is not possible, then the alternative is to discard the infected file and reload a clean backup version. Advances in virus and antivirus technology go hand in hand. Early viruses were relatively simple code fragments and could be identified and purged with relatively simple antivirus software packages. As the virus arms race has evolved, both viruses and, necessarily, antivirus software have grown more complex and sophisticated.

Generations of antivirus software(Static methods)

- First generation: **simple scanners**
- Second generation**: heuristic scanners**
- Third generation: **activity traps**

- Fourth generation: **full-featured protection**
- Fifth generation: Integrity Checkers

1- First generation: Simple Scanners

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. Another type of first-generation scanner maintains a record of the length of programs and looks for changes in length.

2- Second generation: heuristic scanners

A **second-generation** scanner does not rely on a specific signature. Rather, the scanner uses heuristic rules to search for probable virus infection. One class of such scanners looks for fragments of code that are often associated with viruses. 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 virus to identify it, then remove the infection and return the program to service.

3- Third generation: activity traps

Third-generation programs are memory-resident programs that identify a virus by its actions rather than its structure in an infected program. Such programs have the advantage that it is not necessary to develop signatures and heuristics for a wide array of viruses. Rather, it is necessary only to identify the small set of actions that indicate an infection is being attempted and then to intervene.

4- Fourth generation: full-featured protection

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.

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.

5- Integrity checkers:

Integrity Checkers: Viruses operate by changing files. An integrity checker exploits this behavior to find viruses, by watching for unauthorized changes to files. Integrity checkers must start with a perfectly clean, 100% virus-free system. The integrity checker initially computes and stores a checksum for each file in the system it's watching. Later, a file's checksum is recomputed and compared against the original, stored checksum. If the checksums are different, then a change to the file occurred.

Note: A checksum is a small-sized block of data derived from another block of digital data for the purpose of detecting errors that may have been introduced during its transmission or storage. By themselves, checksums are often used to verify data integrity but are not relied upon to verify data authenticity.

There are three types of integrity checker:

a) Offline. Checksums are only verified periodically, e.g., once a week.
b) Self-checking. Executable files are modified to check themselves when run.
c) Integrity shells. An executable file's checksum is verified immediately prior to execution.

2- Advanced Antivirus Techniques (Dynamic Methods)

More sophisticated antivirus approaches and products continue to appear. In this subsection, we highlight some of the most important.

1. Generic Decryption.

2. Digital Immune System.

3. Behavior Blocking Software

1- GENERIC DECRYPTION

Generic decryption (GD) technology enables the antivirus program to easily detect even the most complex polymorphic viruses while maintaining fast scanning speeds . Recall that when a file containing a polymorphic virus is executed, the virus must decrypt itself to activate. In

order to detect such a structure, executable files are run through a GD scanner, which contains the following elements:

• **CPU emulator:** A software-based virtual computer. Instructions in an executable file are interpreted by the emulator rather than executed on the underlying processor. The emulator includes software versions of all registers and other processor hardware, so that the underlying processor is unaffected by programs interpreted on the emulator.

• **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. At the start of each simulation, the emulator begins interpreting instructions in the target code, one at a time.Thus, if the code includes a decryption routine that decrypts and hence exposes the virus, that code is interpreted. In effect, the virus does the work for the antivirus program by exposing the virus. Periodically, the control module interrupts interpretation to scan the target code for virus signatures. During interpretation, the target code can cause no damage to the actual personal

computer environment, because it is being interpreted in a completely controlled environment. The most difficult design issue with a GD scanner is to determine how long to run each interpretation.Typically, virus elements are activated soon after a program begins executing, but this need not be the case. The longer the scanner emulates particular program, the more likely it is to catch any hidden viruses. However, the antivirus program can take up only a limited amount of time and resources before users complain of degraded system performance.

2- DIGITAL IMMUNE SYSTEM

The digital immune system is a comprehensive approach to virus protection developed by IBM and subsequently refined by Symantec The motivation for this development has been the rising threat of Internet-based virus propagation.. Traditionally, the virus threat was characterized by the relatively slow spread of new viruses and new mutations. Antivirus software was typically updated on a monthly basis, and this was sufficient to control the

problem. Also traditionally, the Internet played a comparatively small role in the spread of viruses.

Two major trends in Internet technology have had an increasing impact on the rate of virus propagation in recent years:-

1. **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.
2. **Mobile-program systems:** Capabilities such as Java and ActiveX allow programs to move on their own from one system to another. In response to the threat posed by these Internet-based capabilities, IBM has developed a prototype digital immune system. This system expands on the use of program emulation discussed in the preceding subsection and provides a general purpose emulation and virus-detection system. The objective of this system is to provide rapid response time so that viruses can be stamped out almost as soon as they are introduced. When a new virus enters an organization, the immune system automatically captures it, analyzes it, adds detection and shielding for it, removes it, and passes information about that virus to systems running IBM AntiVirus so that it can be detected before it is allowed to run elsewhere.

The following steps explain the typical steps in digital immune system operation:

1. A monitoring program on each PC uses a variety of heuristics based on system behavior, suspicious changes to programs, or family signature to infer that a virus may be present.The monitoring program forwards a copy of any program thought to be infected to an administrative machine within the organization.

2. The administrative machine encrypts the sample 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.Techniques used for this purpose include emulation, or the creation of a protected environment within which the suspect program can be executed and monitored. 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 around the world receive regular antivirus updates that protect them from the new virus.

Figure(1) illustrates the typical steps in digital immune system operation:

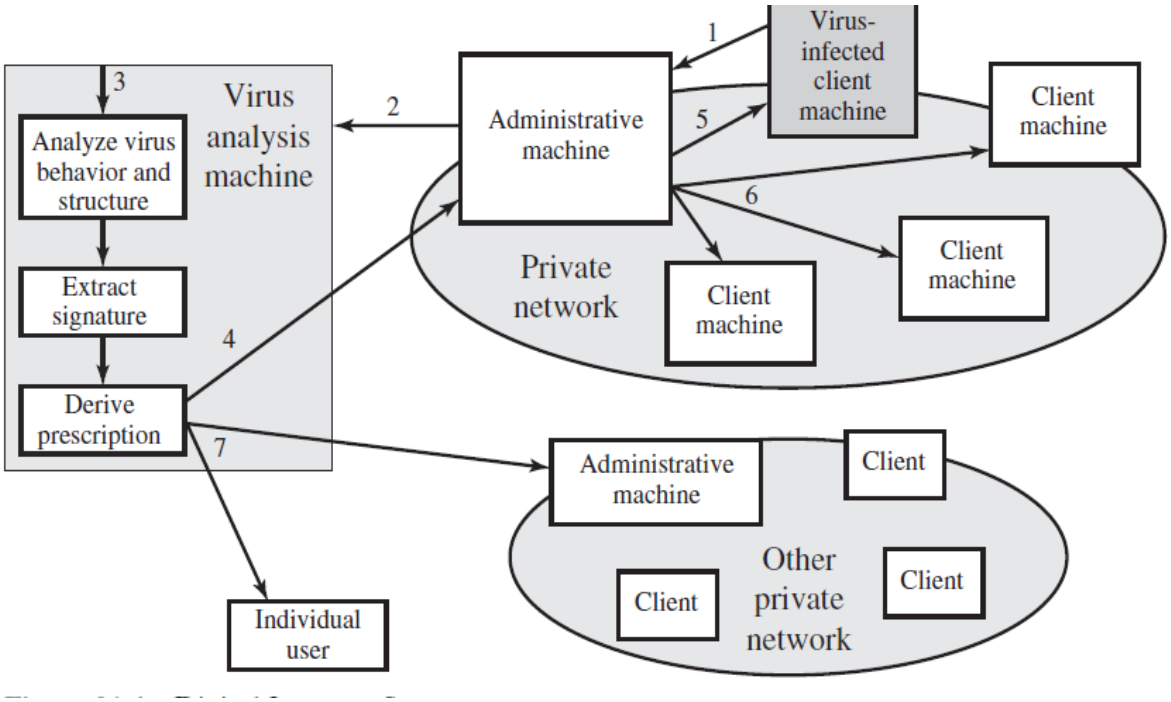


Figure 1 Digital Immune System

The success of the digital immune system depends on the ability of the virus analysis machine to detect new and innovative virus strains. By constantly analyzing and monitoring the viruses found in the wild, it should be possible to continually update the digital immune software to keep up with the threat.

3- BEHAVIOR-BLOCKING SOFTWARE

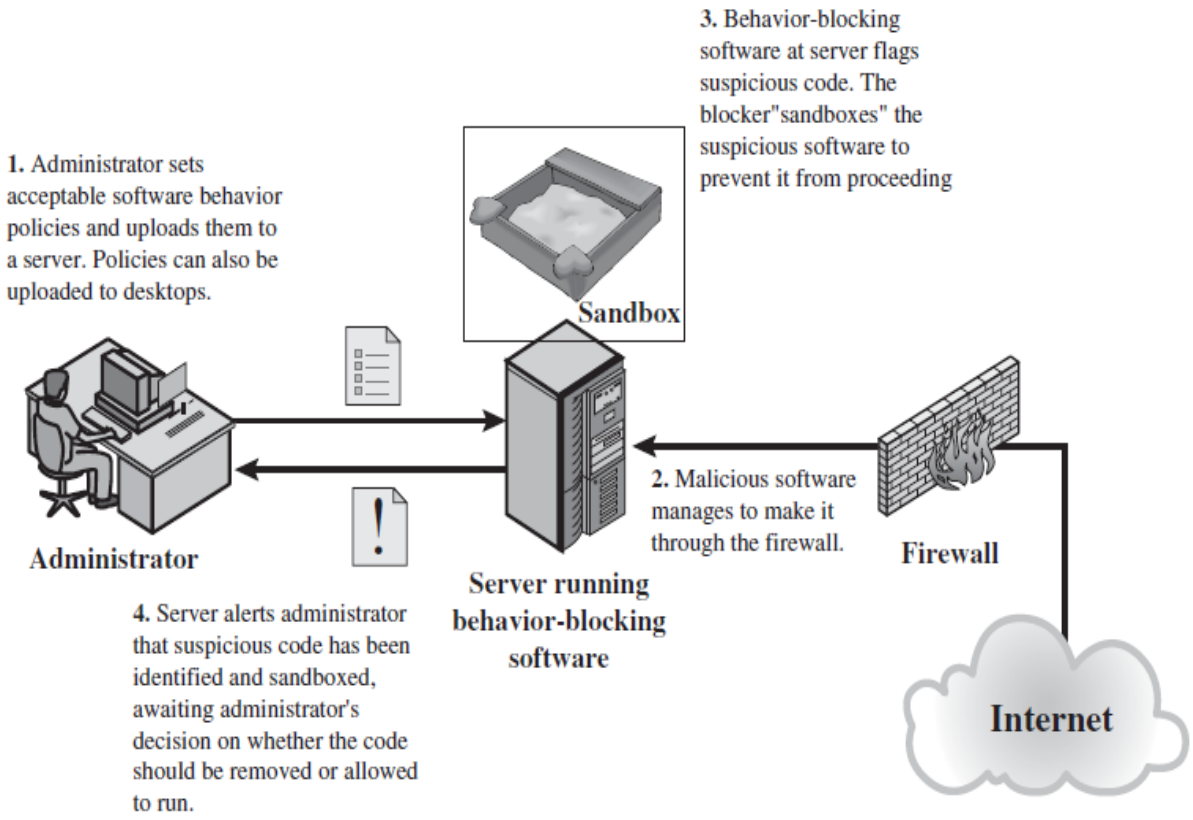
Unlike heuristics or fingerprint-based scanners, behavior-blocking software integrates with the operating system of a host computer and monitors program behavior in real-time for malicious actions .The behavior blocking software then blocks potentially malicious actions before they have a chance to affect 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.

(Figure 2) illustrates the operation of a behavior blocker. 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 a behavior blocker can block suspicious software in real-time, it has an advantage over such established antivirus detection techniques a fingerprinting or heuristics. While there are literally trillions of different ways to obfuscate and rearrange the instructions of a virus or worm, many of which will evade detection by a fingerprint scanner or heuristic, eventually malicious code must make a well-defined request to the operating system. Given that the behavior blocker can intercept all such requests, it can identify and block malicious actions regardless of how obfuscated the program logic appears to be.

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. For example, a new virus might shuffle a number of seemingly unimportant files around the hard drive before infecting a single file and being blocked. Even though the actual infection was blocked, the use may be unable to locate his or her files, causing a loss to productivity or possibly worse.



(Figure 2) **BEHAVIOR-BLOCKING SOFTWARE**

Comparison of Anti-Virus Detection Techniques:

Scanning: Gives precise identification of any viruses that are found. This characteristic makes scanning useful by itself, as well as in conjunction with other anti-virus techniques. But requires an up-to-date database of virus signatures for scanning to be effective. Even assuming that users update their virus databases right away, which isn't the case, there is a delay between the time when a new threat is discovered and when an anti-virus company has a signature update ready.

Static heuristics: Static heuristic analysis detects both known and unknown viruses. But: False positives are a major problem, and a detected virus is neither identified, nor disinfectible except by using generic methods.

Note: In terms of the accuracy of an IDS, there are four possible states for each activity observed. A true positive state is when the IDS identifies an activity as an attack and the

activity is actually an attack. A true positive is activity is actually an attack. That is, a false negative is when the IDS fails to catch an attack.

Integrity checkers: Integrity checkers boast high operating speeds and low resource requirements. They detect known and unknown viruses. But: Detection only occurs after a virus has infected the computer, and the source of the infection can't necessarily be pinpointed. An integrity checker can't detect viruses in newly-created files, or ones modified legitimately,

such as through a software update. Ultimately, the user will be called upon to assess whether a change to a file was made legitimately or not. Finally, found viruses can't be identified or disinfected.

Behavior blockers: Known and unknown viruses are detected. But: While a behavior blocker knows which executable is the problem, unlike an integrity checker, it again cannot identify or disinfect the virus.

Emulation: Any viruses found are running in a safe environment. Known and unknown viruses are detected, even new polymorphic viruses. But emulation is slow. The emulator may stop before the virus reveals itself, and even so, precise emulation is very hard to get correct.

Verification, Quarantine, and Disinfection:

The tasks for anti-virus software that lie beyond detection are verification, quarantine, and disinfection. Compared to detection, these three tasks are performed rarely, and can be much slower and more resource-intensive if necessary.

Verification: After the initial detection of a virus occurs, Anti-virus software will often perform a secondary verification. It is performed for two reasons. First, it is used to reduce false positives that might happen by coincidence. Second, verification is used to positively identify the virus.

Quarantine: When a virus is detected in a file, anti-virus software may need to quarantine the infected file, isolating it from the rest of the system. Quarantine is only a temporary measure, and may only be done until the user decides how to handle the file.

Disinfection: It does not mean that an infected system has been restored to its original state, even if the disinfection was successful, there are different ways to do disinfection: Restore infected files from backups. Because everyone meticulously keeps backups of their files, the affected files can be restored to their backed-up state.

Virus Databases and Virus Description Languages

A virus database and virus description languages are crucial components in the field of cybersecurity, particularly in antivirus and anti-malware systems. Here's an overview of both.

Virus Database

- Purpose: A virus database is a comprehensive repository of known malware signatures, behaviors, and other characteristics. Antivirus software uses this database to detect and remove malware on a system.

- Content:

- Signatures: Unique patterns or sequences in the code of known viruses. These can be binary patterns, file hashes, or specific code sequences.
- **Behavioral Patterns:** Descriptions of how a virus operates, such as file access patterns, network traffic behavior, or changes to system settings.
- **Metadata:** Information about each virus, including its name, type, origin, known payloads, and potential impacts.

- Update Mechanism: The database needs regular updates to include new threats as they are discovered. Most antivirus programs frequently update their virus databases to stay effective against emerging malware.

Virus Description Languages

- Purpose: Virus description languages are used to create formal descriptions of malware. These languages help security tools analyze and categorize viruses based on their behaviors, signatures, and effects on systems,e.g (YARA:, OpenIOC (Indicator of Compromise, Snort Rules)

Key Features:

- Pattern Matching: These languages allow for complex pattern matching against files, processes, or network traffic, helping to identify known or unknown malware.
- Flexibility: The languages are typically flexible enough to describe both known viruses and emerging threats, allowing for the creation of signatures that can evolve with the threat landscape.
- Automation: Descriptions written in these languages can be automatically used by security tools to scan for malware without manual intervention.

How They Work Together

1. Detection: When a file or behavior on a system matches a signature or description in the virus database (often defined using a virus description language like YARA), the antivirus software flags it as malicious.
2. Analysis: The descriptions in the virus database help cybersecurity professionals understand the nature of the detected malware, its potential impact, and how to remove or mitigate it.
3. Updates: As new viruses are discovered, their characteristics are described using virus description languages, and these descriptions are added to the virus database.

In summary, the virus database contains the data necessary for identifying malware, while virus description languages provide the tools to describe and detect that malware in a structured, automated manner.

Worms

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. An e-mail virus has some of the characteristics of a worm because it propagates

itself from system to system. we can still classify it as a virus because it uses a document modified to contain viral macro content and requires human action. 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.

The first known worm implementation was done in in the early 1980s. Network worm programs use network connections to spread from system to system. Once active within a system, a network worm can behave as a computer virus or bacteria, or it could implant Trojan horse programs or perform any number of disruptive or destructive actions.

To replicate itself, a network worm uses some sort of network vehicle.

Examples include the following:

1. 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.
2. 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.

Note: Remote code execution is the ability an attacker has to access someone else's computing device and make changes, no matter where the device is geographically located.

3. Remote login capability: A worm logs onto a remote system as a user and then use commands to copy itself from one system to the other, where it then executes.

The new copy of the worm program is then run on the remote system where, in addition to any functions that it performs at that system, it continues to spread in the same fashion.

A network worm exhibits the same characteristics as a computer virus: (a dormant phase, a propagation phase, a triggering phase, 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.

Note: Remote address is the IP Address/host name of the remote computer to which the connection is connected.

2. Establish a connection with a remote system.

3. Copy itself to the remote system and cause the copy to be run.

Note: A remote computer is a computer to which a user does not have physical access, but which he or she can access or manipulate via some kind of computer network. Remote desktop software allows a person to control a remote computer from another computer.

The Morris Worm

Until the current generation of worms, the best known was the worm released onto the Internet by Robert Morris in 1988 .The Morris worm was 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.The worm performed this task by examining a variety of lists and tables, including system tables that declared which other machines were trusted by this host, users' mail forwarding files, tables by which users gave themselves permission for access to remote accounts, and from a program that reported the status of network connections.

For each discovered host, the worm tried a number of methods for gaining access:

1. It attempted to log on to a remote host as a legitimate user. In this method, the worm first attempted to crack the local password file and then used the discovered passwords and corresponding user IDs. The assumption was that many users would use the same password on different systems. To obtain the passwords, the worm ran a password-cracking program that tried

- **a)** Each user's account name and simple permutations of it
- **b)** A list of 432 built-in passwords that Morris thought to be likely candidates2
- **c)** All the words in the local system dictionary

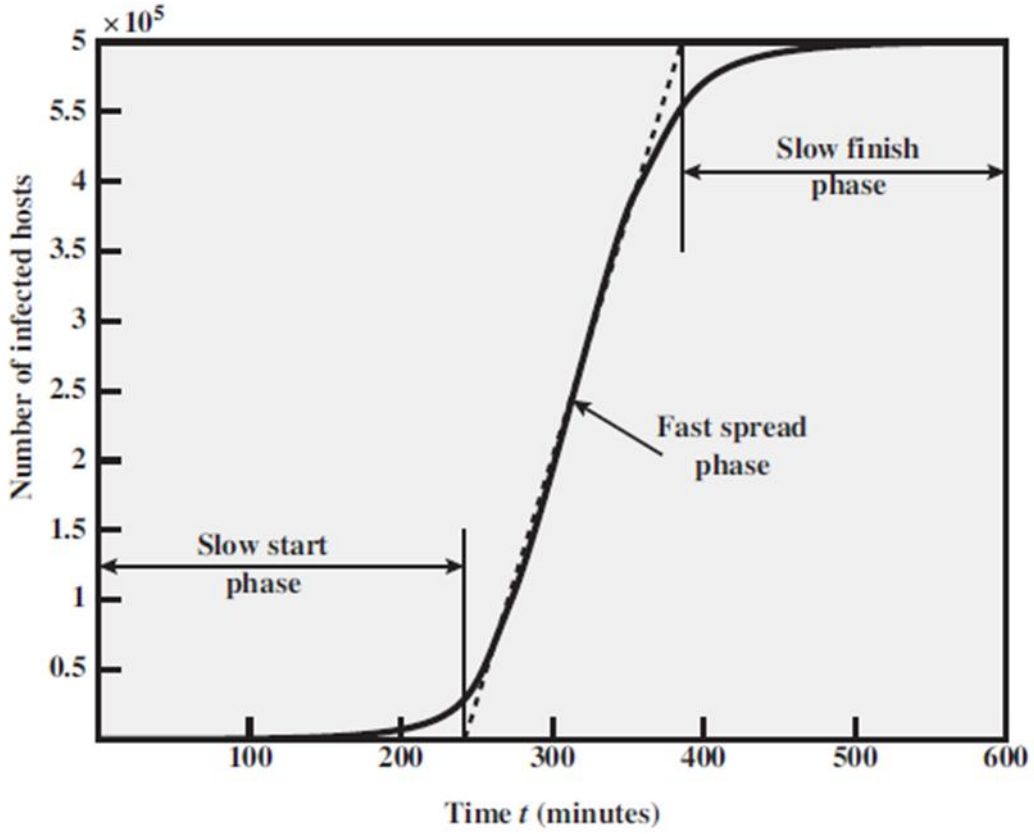
2. It exploited a bug in the UNIX finger protocol, which reports the whereabouts of a remote user.

3. It exploited a trapdoor in the debug option of the remote process that receives and sends mail.

If any of these attacks succeeded, the worm achieved communication with the operating system command interpreter. It then sent this interpreter a short bootstrap program, issued a command to execute that program, and then logged off. The bootstrap program then called back the parent program and downloaded the remainder of the worm. The new worm was then executed.

Worm Propagation Model

The speed of propagation and the total number of hosts infected depend on a number of factors, including **the mode of propagation, the vulnerability or vulnerabilities exploited, and the degree of similarity to preceding attacks**. For the latter factor, an attack that is a variation of a recent previous attack may be countered more effectively than a more novel attack. Figure 3 shows the dynamics for one typical set of parameters. Propagation proceeds through three phases. In the initial phase, the number of hosts increases exponentially. To see that this is so, 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. 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.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.



(Figure 3) Worm Propagation Model

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.

Recent Worm Attacks:

The contemporary era of worm threats began with the release of the Code Red worm in July of 2001. Code Red exploits a security hole in the Microsoft Internet Information Serve (IIS) to penetrate and spread. It also disables the system file checker in Windows.

During a certain period of time, it only spreads. It then initiates a denial-ofservice 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 to the havoc it caused at the targeted server, Code Red consumed enormous amounts of Internet capacity, disruptin service.

Code Red II is a variant that targets Microsoft IISs. In addition, this newer worm installs a backdoor, allowing a hacker to remotely execute commands on victim computers.
Mass-mailing e-mail worm that appeared in 2004. It followed a growing trend of installing a backdoor in infected computers, thereby enabling hackers to gain remote access to data such as passwords and credit card numbers. This worm replicated up to 1000 times per minute and reportedly flooded the Internet with 100 million infected messages in 36 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 networkbased 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.
- **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.

Mobile Phone Worms:

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 mobile phone.

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.

An example of a mobile phone worm is CommWarrior, which was launched in 2005. This worm replicates by means of Bluetooth to other phones in the receiving area. It also sends itself as an MMS file to numbers in the phone's address book and in automatic replies to incoming text messages and MMS messages. In addition, it copies itself to the removable memory card and inserts itself into the program installation files on the phone.

Worm Countermeasures

There is considerable overlap in techniques for dealing with viruses and worms. Once a worm is resident on a machine, antivirus software can be used to detect it.

let us consider the requirements for an effective worm countermeasure scheme:

- **Generality:** The approach taken should be able to handle a wide variety of worm attacks, including polymorphic worms.
- **Timeliness:** The approach should respond quickly so as to limit the number of infected systems and the number of generated transmissions from infected systems.
- **Resiliency:** The approach should be resistant to evasion techniques employed by attackers to evade worm countermeasures.
- **Minimal denial-of-service costs:** The approach should result in minimal reduction in capacity or service due to the actions of the countermeasure software. That is, in an attempt to contain worm propagation, the countermeasure should not significantly disrupt normal operation.
- **Transparency:** The countermeasure software and devices should not require modification to existing (legacy) OSs, application software, and hardware.

- **Global and local coverage:** The approach should be able to deal with attack sources both from outside and inside the enterprise network.

Countermeasure Approaches

A. Signature-based worm scan filtering: This type of approach generates a worm signature, which is then used to prevent worm scans from entering a network / host. 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.

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. This approach can be quite effective but requires efficient detection algorithms and rapid alert dissemination.

NETWORK-BASED WORM DEFENSE

The key element of a network-based worm defense 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:**

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 worm traffic.

Note: In computer terminology, a **honeypot** is a computer security mechanism set to detect, or, in some manner, counteract attempts at unauthorized use of information systems. A honeypot is a computer system for trapping hackers or tracking unconventional or new hacking methods. Honeypots are designed to purposely deceive hackers and identify malicious activities performed over the Internet.

- **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.

(Figure 4) shows an example of a worm countermeasure architecture .

The system works as follows (numbers in figure refer to numbers in the following list):

1. Sensors deployed at various network locations detect a potential worm. The sensor logic can also be incorporated in IDS sensors.

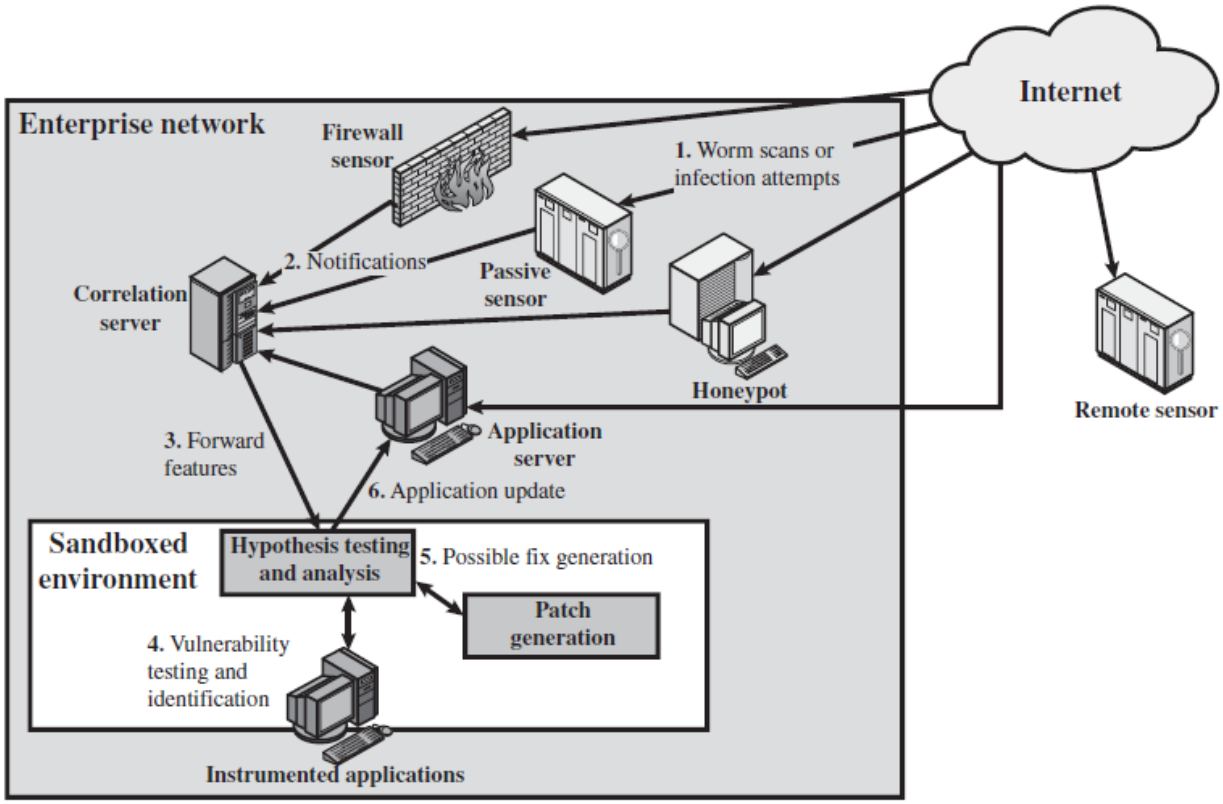
2. The sensors send alerts to a central server that correlates and analyzes the incoming alerts.The correlation server determines the likelihood that a worm attack is being observed and the key characteristics of the attack.

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 4) Placement of Worm Monitors

Distributed Denial of Service Attacks:

Distributed denial of service (DDoS) attacks present a significant security threat to corporations and the threat appears to be growing. In one study, covering a three-week period in 2001, investigators observed more than 12,000 attacks against more than 5000 distinct targets, ranging from well-known ecommerce companies such as Amazon and Hotmail to small foreign ISPs and dial-up connections. DDoS attacks make computer systems inaccessible to servers, networks, with useless traffic so that legitimate users can no longer gain access to those resources.

A denial of service (DoS) attack is an attempt to prevent legitimate users of a service from using that service. When this attack comes from a single host or network node, then it is simply referred to as a DoS attack. A more serious threat is posed by a DDoS attack.

In a DDoS attack, 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.

DDoS Attack Description

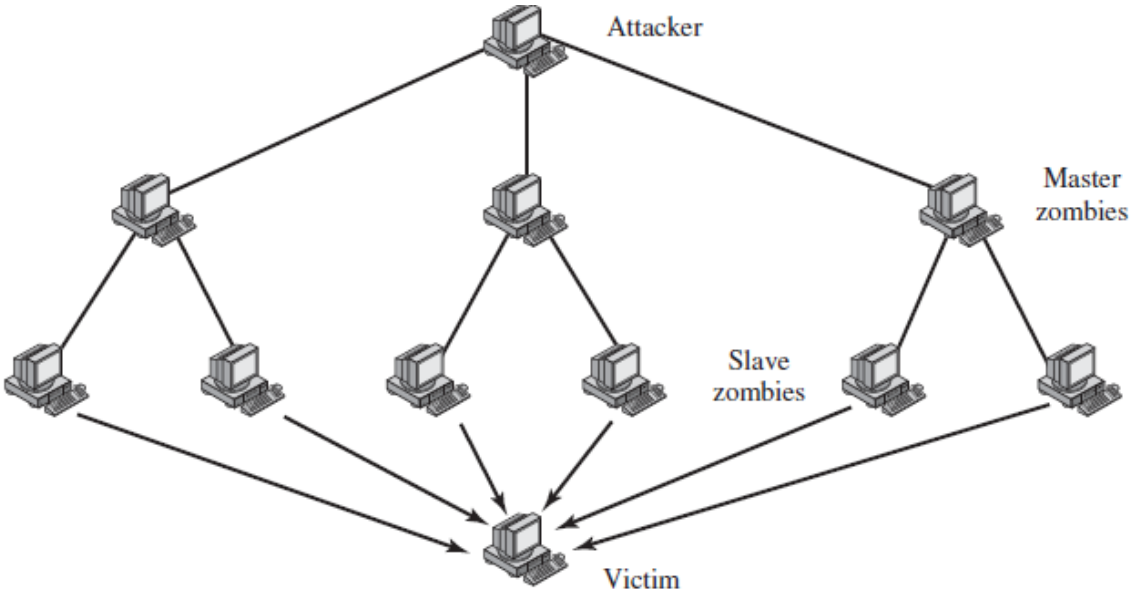
A DDoS attack attempts to consume the target's resources so that it cannot provide service. One way to classify DDoS attacks is in terms of the type of resource that is consumed. Broadly speaking, the resource consumed is either an internal host resource on the target system or data transmission capacity in the local network to which the target is attacked.

The following examples are one of the popular internal resources for the TCP data structure:

1. In many systems, a limited number of data structures are available to hold process information (process identifiers, process table entries, process slots, etc.). An intruder may be able to consume these data structures by writing a simple program that does nothing but repeatedly create copies of itself.
2. An intruder may also attempt to consume disk space in other ways, including:
   - generating excessive numbers of mail messages
   - intentionally generating errors that must be logged99999-0
   - placing files in anonymous areas or network-shared areas

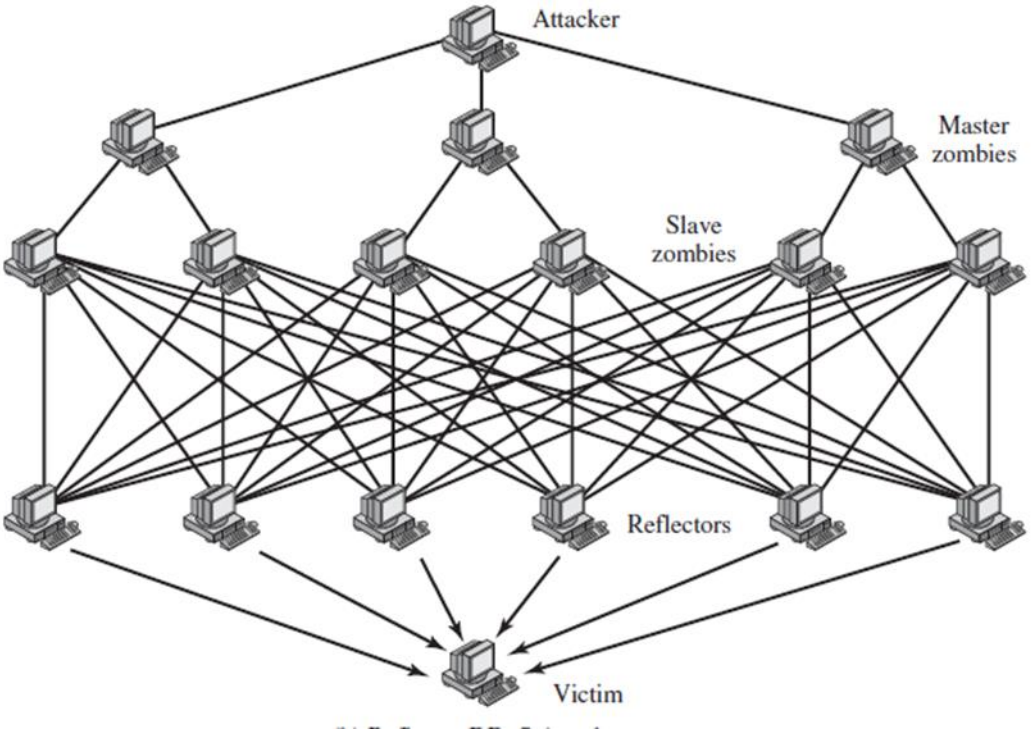
Another way to classify DDoS attacks is as either direct or reflector DDoS attacks.

In a **direct DDoS** attack (Figure 5-a), the attacker is able to implant zombie software on a number of sites distributed throughout the Internet. Often, the DDoS attack involves two levels of zombie machines: master zombies and slave zombies. The hosts of both machines have been infected with malicious code.The attacker coordinates and triggers the master zombies, which in turn coordinate and trigger the slave zombies.The use of two levels of zombies makes it more difficult to trace the attack back to its source and provides for a more resilient network of attackers..



(Figure 5-a) Direct DDoS Attack

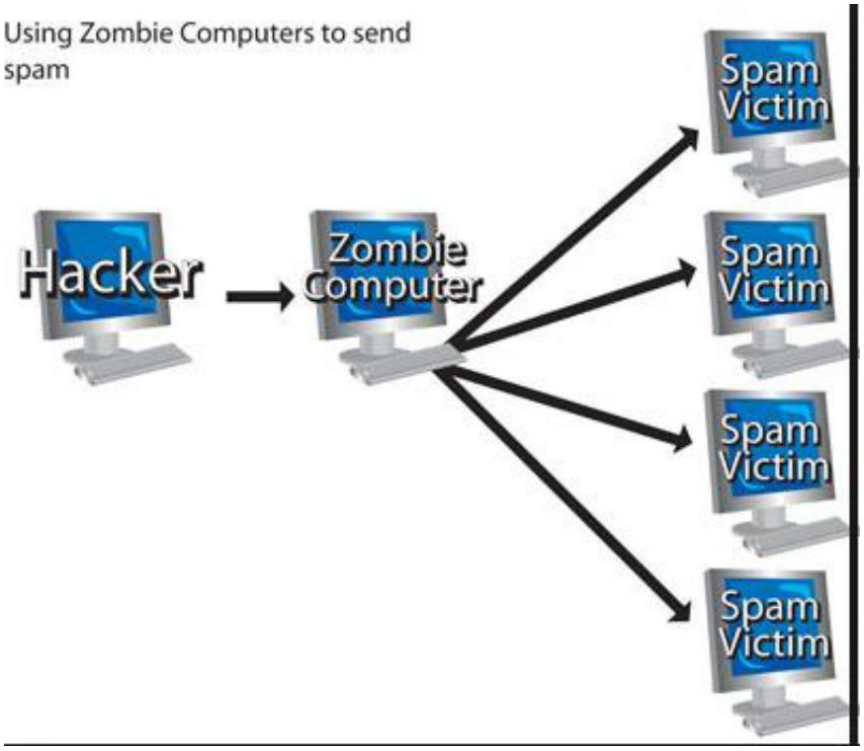
A **reflector DDoS** attack adds another layer of machines (Figure 5.b). In this type of attack, the slave zombies construct packets requiring a response that contains the target's IP address as the source IP address in the packet's IP header. These packets are sent to uninfected machines known as reflectors. The uninfected machines respond with packets directed at the target machine. A reflector DDoS attack can easily involve more machines and more traffic than a direct DDoS attack and hence be more damaging. Further, tracing back the attack or filtering out the attack packets is more difficult because the attack comes from widely dispersed uninfected machines.



(Figure 5-b) Reflector DDoS Attack

Note: In computing, a zombie is a computer connected to the Internet that has been compromised by a hacker, computer virus or Trojan horse program and can be used to perform malicious tasks under remote direction. A zombie (also known as a bot) is a computer that a remote attacker has accessed and set up to forward transmissions (including spam and viruses) to other computers on the Internet.(figure 6)

The zombie computer sends an enormous amount of packets of useless information to a targeted Web site in order to clog the site's routers and keep legitimate users from gaining access to the site. The traffic sent to the Web site is confusing and therefore the computer receiving the data spends time and resources trying to understand the influx of data that has been transmitted by the zombies. Compared to programs such as viruses or worms that can eradicate or steal information, zombies are relatively benign as they temporarily cripple Web sites by flooding them with information and do not compromise the site's data.



(Figure 6) Zombie connection in internet

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.

The essential ingredients in this phase of the attack are the following:

1. Software that can carry out the DDoS attack.The software must be able to run on a large number of machines, must be able to conceal its existence, must be able to communicate with the attacker or have some sort of time-triggered mechanism, and must be able to launch the intended attack toward the target.
2. A vulnerability in a large number of systems.The attacker must become aware of a vulnerability that many system administrators and individual users have failed to patch and that enables the attacker to install the zombie software.
3. A strategy for locating vulnerable machines, a process known as scanning.

In the scanning process, the attacker first seeks out a number of vulnerable machines and infects them. Then, typically, the zombie software that is installed in the infected machines repeats the same scanning process, until a large distributed network of infected machines is created.

Types of scanning strategies:

- **Random:** Each compromised host probes random IP address. This technique produces a high volume of Internet traffic, which may cause generalized disruption even before the actual attack is launched.
- **Hit-List:** The attacker first compiles a long list of potential vulnerable machines. This can be a slow process done over a long period to avoid detection p-0that an attack is underway. 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 strategy results in a very short scanning period, which may make it difficult to detect that infection is taking place.
- **Topological:** This method uses information contained on an infected victim machine to find more hosts to scan.

DDoS Countermeasures:

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

1. **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.

2. **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.

3. **Attack source trace back 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.