جامـــــعة المــــستـقـبـل
AL MUSTAQBAL UNIVERSITY

قـــســـــــم الامــــــــــــــــــــن الـــــــــــــــــــــــــــــــسيبرانـــــــــــــــــــــــــي

Department of Cyber Security

Subject:

Malicious codes

Class:

THIRD

Lecturer:

Dr. Suha Alhussieny

Lecture: (5)

E-Mail Viruses , Virus Countermeasures

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:

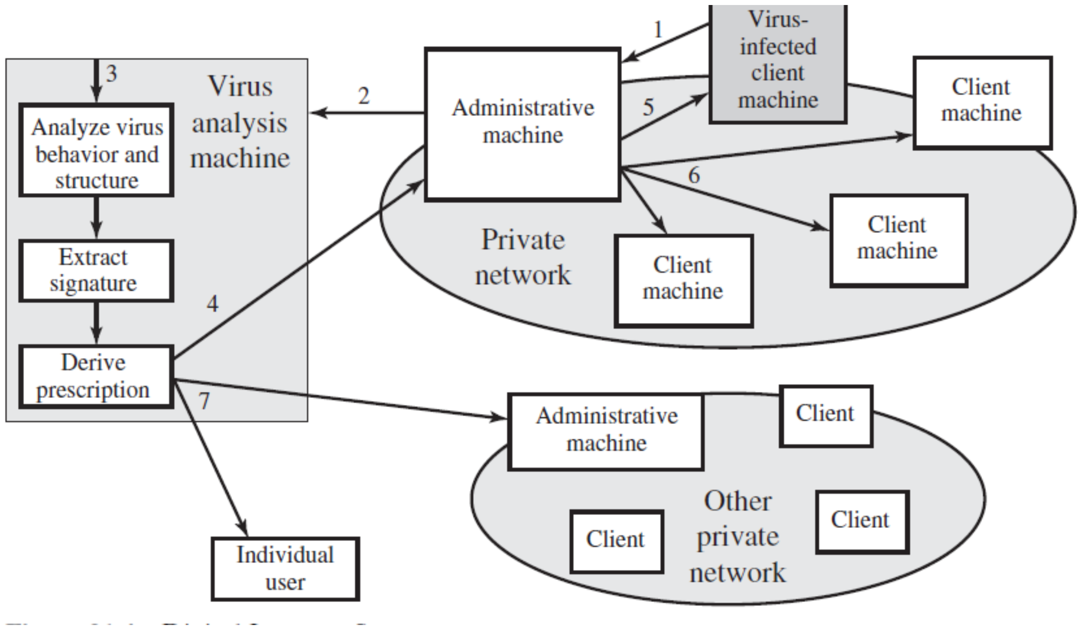


Figure 1 Digital Immune System

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

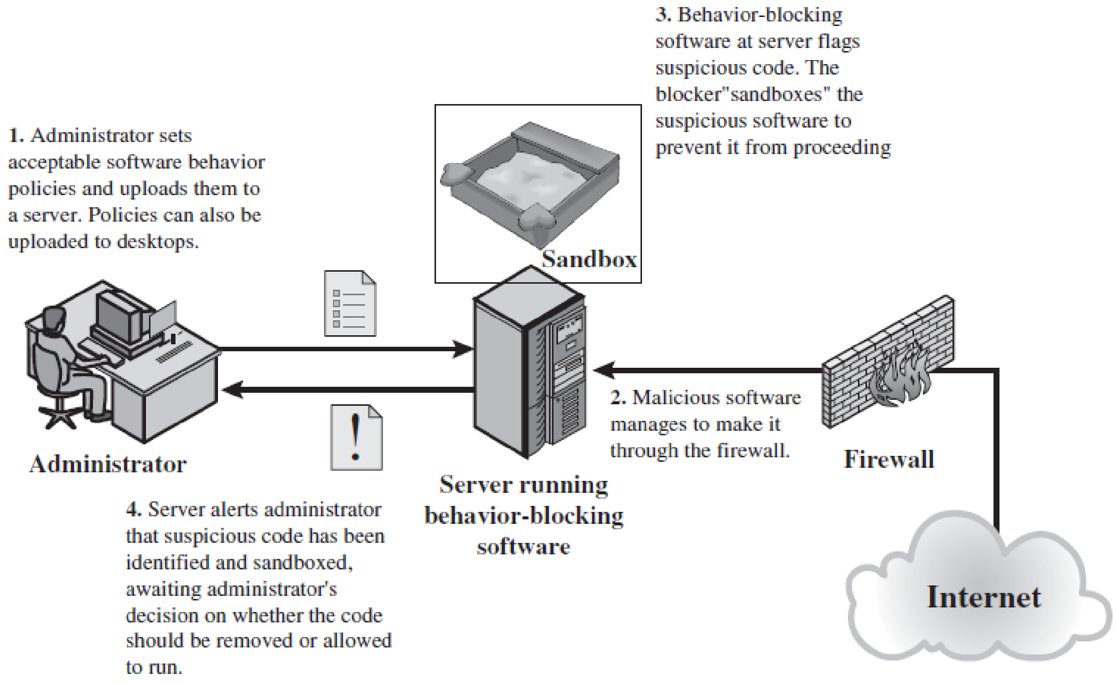
Unlike heuristics or fingerprint-based scanners, behavior-blocking software integrates with the operating system of a host computer and monitors program behavior in real-time for malicious actions .The behavior blocking software then blocks potentially malicious actions before they have a chance to affect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


3. Behavior-blocking software at server flags suspicious code. The blocker"sandboxes" the suspicious software to prevent it from proceeding
1. Administrator sets acceptable software behavior policies and uploads them to a server. Policies can also be uploaded to desktops.
Sandbox
2. Malicious software manages to make it through the firewall.
Firewall
Administrator
Server running behavior-blocking software
4. Server alerts administrator that suspicious code has been identified and sandboxed, awaiting administrator's decision on whether the code should be removed or allowed to run.
Internet

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

H.W/

1. What is an e-mail virus?
2. The first rapidly spreading e-mail virus was known as:
3. Melissa virus primarily targeted which application?
4. The Melissa virus was distributed as:
5. What does the term 'virus countermeasure' refer to?
6. What is the main goal of antivirus approaches?
7. Which of the following is NOT part of antivirus process steps?
8. What do first-generation scanners rely on?
9. What do second-generation heuristic scanners search for?
10. Third-generation antivirus software identifies a virus by:
11. Fourth-generation antivirus software includes:
12. Integrity checkers work by:
13. What must integrity checkers start with?
14. What is a checksum used for?
15. Generic decryption (GD) technology detects:
16. The main challenge in GD scanning is:
17. The digital immune system was developed by:
18. The purpose of the digital immune system is:
19. The digital immune system forwards suspected infected files to:
20. In behavior-blocking software, suspicious code is isolated in a:
21. Which of the following behaviors may trigger blocking?
22. The advantage of behavior blocking is that it:
23. A limitation of behavior blocking is:

24. Which generation introduced activity traps?
25. Integrity shells verify checksums: