# COMPUTER VIROLOGY: TOWARD DESIGNING AN IDEAL ANTI-VIRUS SYSTEM

**Hamid M. A. Abdul-Hussain**
Computer Engineering Department
College of Engineering, University of Baghdad

## ABSTRACT

A new systematic approach is introduced and used to define and study the basic concepts of the anti-virus system. These basic concepts include definitions, design and analysis criteria, and classifications. Also, the characteristics of the ideal anti-virus system are defined. To understand how the proposed design and analysis criteria can be used in practice, the currently available anti-virus products are analyzed using the proposed criteria to see whether they meet the requirements of the ideal anti-virus system or not.

## الخلاصة

طريقة منهجية جديدة تم استحداثها واستخدامها في تعريف ودراسة المبادئ الأساسية للنظام المضاد لفيروسات الحاسبات الإلكترونية (Computer Anti-Virus System). المبادئ الأساسية للنظام المضاد لفيروسات الحاسبات الإلكترونية تضمنت: تعريف، معيار للتصميم والتحليل، وتصنيف مضادات فيروسات الحاسبات. كذلك تم تحديد وتعريف الخواص ومتطلبات النظام المثالي المضاد لفيروسات الحاسبات. ولكي نفهم كيفية استخدام المعيار والتحليل المقترح في الواقع التطبيقي. تم تحليل أغلب المضادات لفيروسات الحاسبات، المتوفرة حاليا، باستخدام المعيار الجديد وذلك لغرض معرفة فيما إذا كانت تطابق متطلبات النظام المثالي المضاد لفيروسات الحاسبات.

**KEYWORDS**
Computer viruses, virus execution, virus detection, virus infection, virus eradication, virus spreading

## .INTRODUCTION

The anti-virus industry is the bright side of the computer virus history. For every virus program there is a utility to get rid of the virus and fix the damage it causes. Once the computer virus appears, a battle begins between the bad guys "i.e. the virus designers" and the good guys "i.e. the anti-virus designers". The effort to combat the computer virus threat has spawned an entire industry. Over the years, the anti-virus industry has had to keep pace, as virus writers have become more sophisticated [Garber 1998].

Even though the anti-virus technology receives increased importance, the great burden of the published literature about the anti-virus takes the form of a story: "Some virus (V) appear somewhere in the world, it can cause the damage (D) to your system, hence, you must purchase the

anti-virus (A) to protect your computer from (V)". In those literatures you will find information about how the computer user can purchase, install, and use the anti-virus. Also, how fast the anti-virus is, how many viruses it can detect, and how it can be updated to cope with new viruses. Even though these stories are more than enough to alert the computer users about the threat and motivate them to purchase and install the anti-virus programs, it is worthless for the academic researchers, technicians, security professionals, or 'in general' those people that are interested with designing the anti-virus systems that can cope with viruses using systematic methods.
There is a risk that the incredible growth of the information society will be brought to a halt by the spread of diseased programs [Fites and Johanson 1989]. Therefore, the efforts to combat the computer virus must continue until an ideal, universal anti-virus is designed. Designing the ideal anti-virus needs a deep understanding of the computer virus and anti-virus concepts. Therefore, a standard scientific approach to the study of the computer virus problem is needed. The researchers must understand the computer virus definition, characteristics, principles of operation, methods of penetration, methods of attacks, and deception tricks so that they can define the characteristics and requirements of the ideal anti-virus.
In part this research has been a follow-up to the paper titled "Computer Virology: Formal Analysis of Computer Viruses" [Abdul-Hussain and Shabib 2002], but in the main it is a complement to that paper. After defining **"**what we are going to fight**"**, that is, the computer virus (see [Abdul-Hussain and Shabib 2002]), this paper explains **"**how to fight it**"**, that is, the anti-virus. The main objective of this work is to introduce a formal systematic approach in the analysis and design of the anti-virus systems. Before we advance to the proposed design and analysis criteria of the anti-virus system, the following are definitions and basic concepts that will be used along this work.
The Anti-Virus is: "A utility of a predefined 'Type', designed specifically to protect every target in a predefined 'T-group' (Target-group) against a predefined 'Viral-Attack' of every virus in a predefined 'V-group' (Virus-group)".
The T-group is: "The group of target sites and/or target cells that must be protected by the anti-virus. And TGS (Target Group Size) is the number of targets in the T-group".
The V-group is: "The group of viruses that the anti-virus can cope with. And VGS (Virus Group Size) is the number of viruses in the V-group".
The Viral Attack is: "Any form of infection, change, or destruction caused by the virus to the targets of the T-group".
The anti-virus utilities can be classified into three Types:
1- Pure-Hardware Anti-Virus**:** In this type the only way to cope with viruses is to use hardware. The writ protection tab on the floppy diskette and the hard disk lock are typical examples of the Pure-Hardware anti-virus.
2- Pure-Software Anti-Virus: In this type, the only way to cope with viruses is to use software.
3- Hardware-Based Anti-Virus**:** In this type, the software anti-virus programs are supported by special hardware anti-virus utilities.
The Anti-Virus System is: "A group of cooperative anti-virus utilities each one of them is designed to perform a predefined protection task".

## PROPOSED DESIGN AND ANALYSIS CRITERIA OF THE ANTI-VIRUS SYSTEM

**Fig.(1)** shows how the ant-virus should adjust the virus variables in order to minimize Virus Detection Age (VDA) see [Abdul-Hussain and Shabib 2002]. The anti-virus can be classified according to detection, prevention, eradication, damage control, and false alarms:
1- Detection: The anti-virus must be capable of detecting any virus that belongs to the V-group and resides in any one of the T-group executables.
2- Prevention: The anti-virus must be capable of preventing any virus belonging to the V-group from infecting any one of the T-group executable.

3- Eradication: The anti-virus must be capable of eradicating any virus belonging to the V-group attached to any one of the T-group executables and, at the same time, repair any change caused due to the virus infection.
4- Damage Control: The anti-virus must be capable of preventing any virus belonging to the V-group from destroying any one of the T-group target cells.
5- Minimizing False-Alarms: The anti-virus makes a False-Alarm (or false positive), if it reports an executable to be infected while it is not, or reports a given piece of code to be a viral-code while it is not.

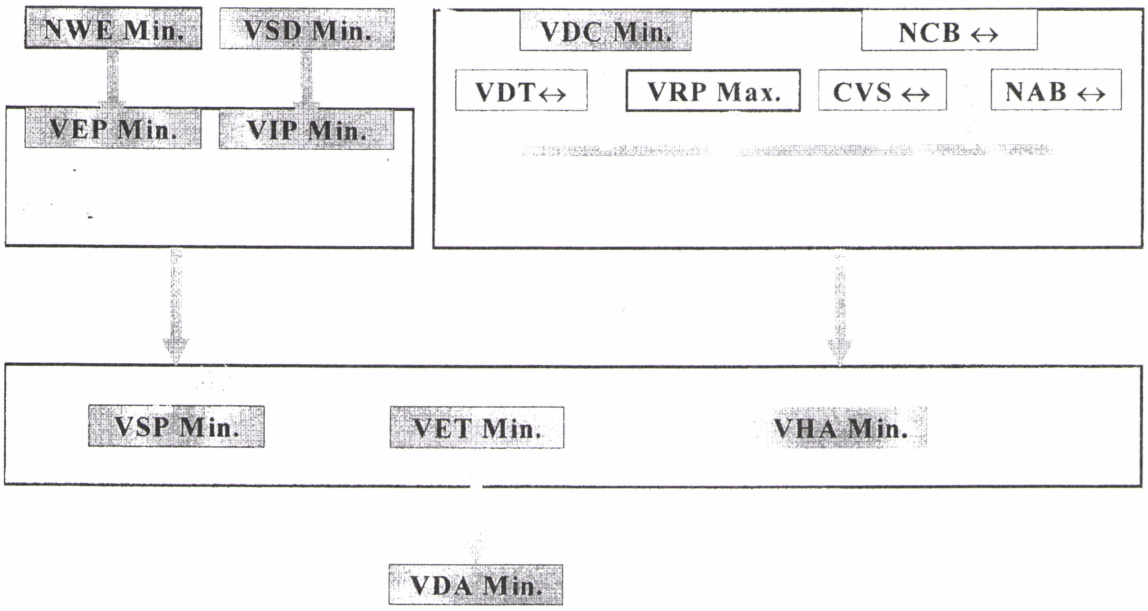

Fig. (1): General trends of maximizing or minimizing the virus variables from the Anti-virus designer point of view.

**Detection (Terminating VHA)**

"The task of discovering the virus presence to terminate its Virus Hidden Age (VHA)". Detection represents the bottleneck in the anti-virus design. Because without discovering the virus presence, there is no way to prevent it from infecting new target sites or eradicating it from the infected ones. In order to design an anti-virus that can cope with viruses using reliable and systematic method, a general and trusted rule must be formalized to be used to detect the computer virus. The detection techniques can be classified into: Target-Based Detection, Code-Based Detection, and Behavior-Based Detection.

1- Target-Based Detection : "The technique of detecting the computer virus by distinguishing the clean executable from the infected executable". By definition, viruses infect other executables by attaching themselves to the infected executable. This attachment will change the executable size by Number of Added Bytes (NAB-bytes), contents by Number of Changed Bytes ( NCB), and execution time (by VDT). This fact can be used to derive the following detection rule:

RULE1: "IF the size, contents, and/or execution time of a given executable is changed, THEN, the executable is infected by a computer virus".

RULE1 is trusted to be satisfied , absolutely, by any computer virus regardless of its generation date and Virus Deception Capability (VDC). However, the effectiveness of the above rule according to the number of false alarms must be considered. The number of false alarms associated with RULE1 is minimum because of the fact: "Unlike data files, executables, such as executable files and BPS code, normally don't change unless you re-install them [Fites and

Johanson 1989]". To be precise, unintended change to executables due to physical or system errors may cause false alarms if RULE1 is used. In fact, detecting and fixing unintended changes is the responsibility of the OS. Therefore, for example, CRC "Cyclical Redundancy Check" is used by DOS to determine whether the data was transferred properly during disk read and write operation. As general rule, DOS consider any intended change (including viral attacks) as a legal change, and consider any unintended change as illegal change. Because, unintended executable change can be detected and fixed by the OS or one of the traditional error detection and recovery utilities, it will be excluded from the discussion. The following Target-Based detection methods are based on RULE1.

A-File Comparison: File comparison programs compare two copies of the same program to see if there are any differences. If a difference is discovered, there is a possibility that a virus has caused the change [Endrijonas 1993]. This method has the following drawbacks:

- Since there is two copies of each file, one half of the storage will be redundant.
- It assumes that the original program disk you used to copy the program to your computer was virus-free. If your original is infected, chances are that the virus has also infected the copy on your disk. As a result, the two will be identical when compared; leading you to believe that no virus is present [Endrijonas 1993].
- The backups must be updated in case of new executables are added. Updating backups is considered a vulnerable spot because it may give viruses the opportunity to attack the backups.

B- Encryption: Encryption can be used to detect the infection by encrypting the executables using conventional or public-key cryptography, or by recording a cryptographic checksum, so that any modification can be detected prior to execution. **Fig.(2)** depicts the proposed detection mechanism. The executable, E, is encrypted by the Cryptosystem to produce E'. The run-time environment passes E' to the Cryptosystem where the decryption is performed. The deciphered executable is passed back to the run-time environment which will attempt to run the result. If this attempt fails, the executable has been modified since it was encrypted and the proper authorities are modified. Thus the run-time environment detects any modification, whether unintended or due to a viral attack [Pozzo and Gray 1987]. The run-time environment described above represent the anti-virus. The following drawbacks are inherent to this method:

- If a virus exists in an executable prior to its encryption, its presence will not be detected by this mechanism [Pozzo and Gray 1987].
- This detection method cannot be used with another detection method based on RULE2 (see "Code-Based Detection") at the same time. Because, if this method applied, the executables will be stored in an encrypted form and hence cannot be analyzed or searched for viral-activities.

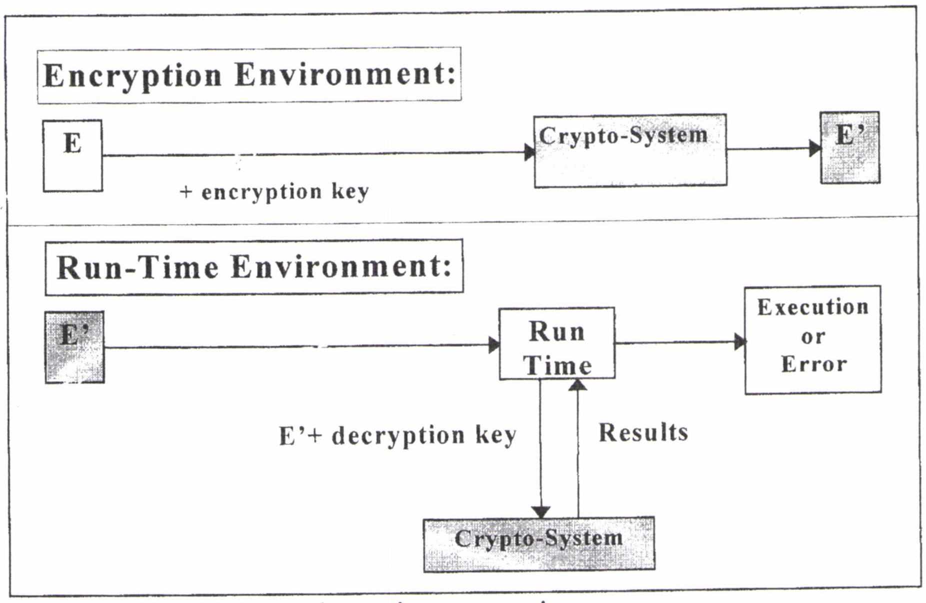

Fig. (2) Detection using encryption.

C- Checksum: BCC (Block Checking Character) or CRC (cyclic redundancy check) comparison programs can be used to find changes in other programs [Endrijonas 1993]. BCC is often used to detect multiple bit (byte) errors. The BCC may be computed as 2's complement sum of all the preceding bytes in the data block. Unlike the BCC, the CRC method is not byte oriented. Instead, the data block is thought of as a "stream" of serial data bits [Uffenbeck 1987]. For more information about the BCC and CRC see [Uffenbeck 1987].

Checksum and CRC techniques are used in basically the same way. The anti-virus must calculate a BCC/CRC for the executable and attach this checksum number into the executable. Or calculate a BCC/CRC for each executable in the system and store the executable name with its corresponding BCC/CRC in a data file. The anti-virus then calculates the BCC/CRC of the executables either periodically or on request. If the calculated BCC/CRC doesn't equal to zero, then the executable was changed and might be infected by a virus. The following drawback is inherent to this method:

- If the BCC/CRC number calculated for an infected program, it will be based on the attached virus, therefor the detection mechanism cannot detect the existence of the virus by checking the corresponding values of BCC or CRC.

2- Code-Based Detection: "The technique of detecting the computer virus by distinguishing the viral code from the legitimate code". All of the code-based detection methods are based on the following rule:

RULE2: "IF a questionable piece of software includes a segment that matches a known viral-activity, THEN, the software is infected".

Any thing the virus can do during its execution is called a viral-activity. However, not all viral-activities can be used in detection. In general, to maximize the detection capability and, at the same time, minimize the number of false alarms, the selected viral-activity must satisfy the following requirements:

a) Common to all viruses of the V-group.

b) Virtually guaranteed not to exist in any one of the T-group executables.

Satisfying the requirements in (a) and (b) depends on the selected V-group and T-group. For example, assuming that the V-group includes, absolutely, any computer virus (VGS= Max.), and the T-group includes, absolutely, any target site/cell (TGS= Max.), then the following can be concluded:

- Some of the viral-activities don't meet the requirements in (a) and (b), the damage routine falls into this group. The damage activities do not meet the requirement in a) because it is not essential to all viruses. The virus may not have a damage routine at all, but still performs its basic task of infection and replication. Even for those viruses that have a damage routine, there is no common nasty task for these damage routines, for example, some viruses destroy the BPS, while the others just display a message on the screen. The damage routine doesn't meet the requirement in b) because what is called a damage routine in the virus may exist as a useful routine in a legitimate program. For example, formatting the hard disk is one of the most common virus nasty tasks, on the other hand, many legitimate programs, such as NORTON UTILITIES, includes special routines that are specifically designed to perform this task. Another example is overwriting FAT or RD, which is also considered one of the virus nasty tasks, but DOS itself overwrite the FAT and RD to add, insert, and delete new file clusters and directory entries.
- Some viral-activities meet the requirements in (a) but not in (b), the searching activities fall into this group. The searching activities meet the requirement in (a) because they are common and essential for all viruses. However, the searching activities do not meet the requirement in (b) because many legitimate programs share the same characteristics with the virus searching routine, for example, DOS DIR command search for files the same way as a program-to-program searching type virus. Also there is many legitimate programs that search for existing executable, for example, some executable files search for their overlies.
- Some viral-activities meet the requirement in (b) but not in (a).This group include those viral-activities that are common in only one specific type of viruses but not common to all viruses, for example:
  # Decreasing the amount of URAM, common between all resident-searching type BPS viruses.
  # Code division into VLC and VMC, common between all BPS viruses.
  # Nonstandard TSR installation, common between all resident-searching type viruses and ADWA stealth type viruses.
  # Self encryption/decryption, common between all encrypted and polymorphic viruses.
  # Overwriting the executable code in memory, common between all BPS and COM file viruses.
- Some viral-activities meet the requirements in (a) and (b), the infection activities fall into this group. The infection activities meet the requirement in (a) because the infection routine must exist in any computer virus by definition. Because any program that contain an infection routine is considered a virus, the infection routines will never exist in legitimate programs, therefore, the infection activities meet the requirement in (b). Regardless of how their infection routines are designed and which target site they attack, the sequence of instructions used by these routines are, relatively, similar. These instruction sequences are used to attach a limited number of bytes (NAB) to the executable and/or modify a limited number of bytes (NCB) in the existing executable code or header. This process is usually called "Suspicious File Access [TBAV 1989-96]" or more general **"**Suspicious Executable Access".

  There is only one way to exploit RULE2, the executable code must be disassembled to see what it is intended to do.

3- Behavior-Based Detection: "The technique of detecting the computer virus by monitoring the computer system for a suspicious executable behavior**".** The Behavior-Based detection methods are based on the following rule:

RULE3: "IF any active executable 'E' tries to change any thing in an already existing executable 'T', THEN, 'E' is infected by a virus".

RULE3 is trusted to be satisfied by any computer virus regardless of its generation date and VDC. The fact that only program development applications such as "LINK.EXE" write to existing executables, minimizes the number of false alarms associated with RULE3.
Behavior-based detection methods detect viruses while the virus is active. Terminate-and-Stay Resident (TSR) programs are used as Behavior-Based virus detectors. A virus-detecting TSR monitors the behavior of the other software on the system. It might detect a virus in the act of attempting to spread. This will be explained later.

**Prevention (Minimizing VSP)**
"The task of preventing the virus from replicating and spreading itself to minimize its VSP". The prevention techniques can be classified into: VEP-Based Prevention, and VIP-Based Prevention.

1- VEP-Based Prevention: "The technique of minimizing VSP by avoiding the execution of the virus and adjusting its VEP=0". Before executing any one of the T-group executables, a VEP-based prevention anti-virus must be capable of deciding whether the executable is clean to execute it, or it is infected to avoid executing it. Clearly, in order to decide whether the executable is clean or infected, the anti-virus must use the detection rules. It must be clear that the prevention capabilities of the anti-virus system depend on its detection capabilities. Therefore, VEP-Based prevention is considered the integral part of the Target-Based detection and Code-Based detection.
2- VIP-Based Prevention: "The technique of minimizing VSP by preventing the virus from infecting new target sites and adjusting its VIP=0". If any active program tries to write something into an existing executable, this program may be a virus, and the VIP-Based prevention anti-virus must suspend its execution so that it cannot infect the executable. Therefore, VIP-Based prevention is considered the integral part of the Behavior-Based detection.

**Eradication (Terminating VDA)**
"The task of eradicating the virus from the infected executables to terminate its VDA". The anti-virus eradication capability depend on: the efficiency of the eradication method (algorithm), VET of the virus being eradicated, and TGS. There are two possible ways to eradicate a detected virus. The first is trivial, that is, delete or overwrite (i.e. kill) both the virus and its host. The second way is to wipe out the virus and repair the host so that it can execute like if it was not infected at all. Clearly, the second way is the only way to be considered when designing the anti-virus. The eradication techniques can be classified into: Backup-Based Eradication, and Analysis-Based Eradication.

1- Backup-Based Eradication: "The technique of eradicating the virus by overwriting the infected executable by a clean backup copy". In this method there must be two copies for each executable. The first is trusted to be clean, and will be called the "Backup Copy". The second copy is vulnerable for viral attacks, and will be called the "Vulnerable Copy". If the vulnerable copy is infected by a virus, the virus can be eradicated and the vulnerable copy can be repaired by overwriting the vulnerable copy by the backup copy. The availability of the backup copy is very important and critical requirement in this method.
2- Analysis-Based Eradication: "The technique of eradicating the virus and repairing its host by analyzing the host to separate the virus code/data from the executable code/data". Only one copy is needed in this approach. However, the required analysis is difficult. If you have an infected executable, then, you must decide which parts of this executable belong to the virus to wipe them out, which parts of this executable belong to the host to leave them unchanged, and which parts of the host was changed by the virus to repair them. Since the virus changes the executable by modifying a limited number of bytes (i.e. NCB) and/or adding a limited number

of bytes (i.e. NAB), an analysis-based eradication anti-virus must be capable of obtaining the following information:

a- The added bytes (NAB) and their positions
b- The changed bytes (NCB) and their positions
c- The original value of any byte changed by the virus during the infection and its position.

Once the above information obtained, the virus can be eradicated as follows:

a- Load the original values of the changed bytes and store them in their correct positions according to NCB.
b- Wipe out any added byte(s) according NAB.

**Damage Control**

"The task of protecting the target cell against destruction by viral attacks". Damage makes viruses a serious security threat, therefore, the anti-virus must control this damage. The damage control techniques can be classified into: Backup-Based, Cell-Based, and Virus-Based Damage Control:

1- Backup-Based Damage Control: "The technique of repairing the damage caused by the virus to the target cell by using a clean backup copy of the cell". Backup-based damage control is similar to backup-based eradication. This method needs a backup copy of each target cell stored in a safe place. The backup copy can be used to replace the vulnerable copy of the target cell after the viral attack. The availability of the backup copy is very important and critical requirement in this method

2- Cell-Based Damage Control: "The technique of preventing the virus from destroying the target cell by preventing it from writing to this cell". Cell-Based damage control is similar to VIP-Based Prevention. In this technique, the anti-virus must prevent the virus from destroying the target cell, by preventing the virus from writing into or changing this cell.

3- Virus-Based Damage Control: "The technique of preventing the virus from destroying the target cell by avoiding the execution of the virus". Virus-Based damage control is similar to VEP-Based prevention. The anti-virus must prevent viruses from destroying the target cells by avoiding the execution of the virus.

## CHARACTERISTICS OF THE IDEAL ANTI-VIRUS SYSTEM ACCORDING TO THE PROPOSED CRITERIA

After defining the basic concepts and structure of the anti-virus system, the characteristics of the Ideal Anti-Virus System (IAVS) must be defined and summarized.

**V-group and T-group**

In general, the anti-virus can be designed as a special purpose anti-virus to protect a predefined group of targets. This group usually includes secret or important data or programs. Also, the anti-virus can be designed as a general purpose anti-virus to protect all targets within its environments. Therefore: "For the IAVS, the T-group can include only one target (TGS=1), a group of targets (TGS>1), or all targets (TGS= Max.) within the anti-virus environments". However, maximizing TGS is desirable. Once the T-group defined, the IAVS must be capable of protecting it against any viral attack at any time. Therefore: "The V-group of the IAVS must include all computer viruses (VGS= Max.) regardless of their generation date and VDC"

**Performance**

With VGS= Max. and a predefined TGS, the IAVS must be ideal with respect to detection, prevention, eradication, damage control, and false alarms. Therefore, the IAVS system must be capable of performing all of the following tasks:

1- Detection: The IAVS must be capable of detecting any virus residing in any one of the T-group executables.
2- Prevention: The IAVS must be capable of preventing any virus from infecting any one of the T-group executable.
3- Eradication: The IAVS must be capable of eradicating any virus attached to any one of the T-group executables and, at the same time, repair any change due to the virus infection.
"NOTE: To terminate VDA (see [Abdul-Hussain and Shabib 2002]), any subsequent copy of the detected virus must be eradicated from any disk anywhere in the world. To do this, the anti-virus must be used in, absolutely, every computer in the world. It is clear that this is an impractical requirement. Therefore, the anti-virus system is considered ideal if it can eradicate, absolutely, any virus detected within its T-group. It is clear that if every anti-virus in the world is ideal, VDA will be terminated in short interval of time".
4- Damage Control: The IAVS must be capable of preventing any virus from destroying any one of the T-group target cells.
5- Number of False-Alarms=0:

**Type**

Satisfying the requirements of ideal detection, prevention, eradication, damage control, and false alarms depend on the anti-virus type:
1- Pure-Hardware Anti-Virus: The Pure-Hardware anti-virus can satisfy the requirements of ideal prevention and damage control, because viruses cannot deceive or bypass a hardware protection layer. However, the Pure-Hardware anti-virus has the following disadvantages:
- Cannot satisfy the requirement of ideal detection because it cannot detect viruses that reside in the T-group. For example, the virus may reside in a write-protected floppy diskette.
- Cannot satisfy the requirement of ideal eradication because it cannot eradicate viruses that reside in the T-group.
- Cannot minimize the number of false alarms, because it cannot distinguish between legitimate activities and viral activities. Therefore, for example, it prevents legitimate programs from writing to the protected disk.
2- Pure-Software Anti-Virus: The Pure-Software anti-virus can satisfy the requirements of ideal detection, eradication, and minimum false alarms. However, it can satisfy the requirement of ideal prevention and damage control, only if: "The virus is detected and eradicated without getting any opportunity to execute". Because, if the virus gets an opportunity to execute in a Pure-Software anti-virus system, then, it may deceive the protection system or bypass it by using the direct hardware access method. Therefore, a Pure-Software anti-virus can satisfy all of the requirements of the IAVS system only by using:
- Target-Based or Code-Based detection.
- VEP-Based prevention.
- Virus-Based damage control.
3- Hardware-Based Anti-Virus: The Hardware-Based anti-virus combine the good characteristics of Pure- Hardware and Pure-Software types, therefore, it can satisfy the requirements of ideal detection, prevention, eradication, damage control, and minimum false alarms regardless of whether the virus get the opportunity to executed or not. Consequently, all of the techniques of detection, prevention, eradication, and damage control can be used to implement an ideal Hardware-Based anti-virus system.

## CLASSIFICATION OF ANTI-VIRUS SYSTEMS USING THE PROPOSED CRITERIA

As a case study, some of the currently available anti-virus systems will be studied and their efficiency will be analyzed with respect to DOS machines. There is no known Hardware-Based anti-

virus system for DOS machines, because of the lack of hardware protection levels in Real-Mode PCs.
Pure-Hardware anti-virus product exist in practice: "To reduce the risk of viruses entering a PC in the first place, Corporate Management Group of Austin, Texas, sells a series of floppy-disk-drive locks that the company characterizes as 'chastity belts' for computers. Thunder Byte PC Immunizer, from Glynn International of Brookline, Mass., is a circuit board that plugs into an IBM-compatible PC and attaches via a cable to system's hard-disk controller board. This connection physically prevents viruses from modifying program files or start up areas of a hard disk. Virus Trap, a board from JAS Technology of Warrenton, Va., offers similar protection [O'mally 1993]".
Pure-Software anti-virus systems are very widely used in practice. Examples of these systems are Dr. Solomon's Anti-Virus, F-Prot Professional, McAfee VirusScan, PC-cillin, TBAV, Norton Anti-Virus, ViruSweep, and IBM's Immune System for Cyberspace ([Garber 1998], [Endrijonas 1993]). To protect your PC against viruses you must purchase one of these systems and install it in your PC. For simplicity, these Pure-Software anti-virus systems will be referred to as the "Dependent-Defence System (DDS)". This name used because: "The executables in the T-group 'Depend' on an external anti-virus to protect them against the attacks of the V-group viruses". The operation and performance of these systems will be explained in what follow.

## Scanners

Scanners are the generic name of the virus detection programs used in DDS. This name is used because the anti-virus scans the T-group searching for viruses (see [Gralla 1997]). Almost all scanners are designed to detect the virus in any target site, therefore, they are designed with TGS= Max. In general, to satisfy the requirement of ideal detection, scanners must solve a seriously difficult problem, that is, the **"Lack of Knowledge"** problem that can be explained as follows: the scanner must detect the presence of the virus 'V' in the executable 'E' without any previous knowledge about the contents of 'E' or 'V'. Therefore, deciding whether 'E' is infected by 'V' or not, as well as, distinguishing the code of 'E' from the code of 'V', will be a very difficult task.

1- Target-Based Detection Scanners: Target-based detection scanners tries to solve the "lack of knowledge" problem by storing some information (a backup copy, encrypted image, or checksum number) about the protected executable once installed in the system. Some scanners store a backup copy of each file and then detect viruses using the 'File Comparison' method. Some scanners store the protected executables in an encrypted form and detect viruses using the 'Encryption' method. Other scanners calculate and store a checksum number (BCC or CRC) for each protected executable and detect viruses using the 'Checksum' method. In general, scanners used by the DDS cannot satisfy the requirement of ideal Target-Based detection because of the following:

- The scanner must ensure that the executable is clean before storing the backup copy, preparing the encrypted image, or calculating the checksum number. Because otherwise, no one of the three methods can detect the virus presence. Clearly, there is no way to do this because of the lack of knowledge problem described above. Therefore: "Target-Based detection scanners cannot detect viruses that reside in the T-group prior to their installation".
- Standard Protection Problem: In general, the scanner must use a standard method to store the backups, encrypt/decrypt executables, or calculate the checksum numbers. Using a different method for each executable increase the code and execution time of the scanner.
  Viruses can deceive or escape detection by the scanner if they know the standard protection method used by the scanner. This problem considered as a vulnerable spot from the virus designer point of view. Knowing the standard method used by the scanner is not a difficult task in practice. Because the scanner can be purchased and used by any computer user. The virus designer can obtain a copy of this scanner and analyze its code to see how it works. For

example, if the scanner store the checksum in a special data file, then, the virus can modify the entry (in the data file) of the executable after infecting it so that every thing seems to be correct. If the scanner embeds the checksum number inside the executable, the virus can avoid detection easily by knowing which method (BCC or CRC) is used. If the 2's complement BCC method is used, the summation of all bytes must be zero. The virus can, after infection, calculate a 2's complement BCC and embeds it inside the executable. In this case, the summation of all bytes will be zero, even though the executable is infected. This explains why the virus strains designed after the releasing of the scanner can escape detection by this scanner, therefore, the scanner must be updated periodically.

As a summary, the scanner detection capability depends on the virus generation date and VDC. This means that scanners cannot maximize VGS, therefore: "Scanners cannot satisfy the requirement of ideal Target-Based detection".

2- Code-Based Detection Scanners: The well Known Code-Based Detection Scanners are Heuristic Scan and neural Network Scanners:

A- Heuristic Scan: Heuristic scanners search for "certain inevitable code" that all viruses exhibit in their attempts to infect files. The developers of this method have formulated a list of basic rules that all viruses, regardless of the strain, follow when attempting to spread infection (See [Endrijonas 1993]). In other word, they search for evidence of a viral activity rather than relying on any signature (V-Marker). However, the viral-activities used by these scanners vary from one to the other. In general, all the viral-activities explained in the context of RULE2 above are applied. The traditional method used by such scanners is to disassemble the executable program and then find out what this program is intended to do, during the search a special flag is set for each viral activity found. Depending on the flags state at the end of the search the scanner can decide whether the current program is capable of infecting another program or not. Even though the heuristic scan technique is considered a sophisticated enhancement in the design of the detection methods it has the following disadvantages:

- It has increased number of false alarms.
- Its detection capability depend on the virus generation date and VDC, because professional virus designers can design new strains of viruses that can escape detection by the heuristic scanners using special tricks. This means that heuristic scanners cannot maximize VGS. Therefore: "Heuristic scanners cannot satisfy the requirement of ideal Code-Based detection".

To understand how heuristic scanners can be deceived, some of the heuristic flags used by TBAV "Thunder Byte Anti-Virus" will be explained as a practical example:

B: Back to entry flag: The program seems to execute some code, and after that jumps back to the entry-point of the program. Normally this results in an endless loop, except when the program also modifies some of its instructions. This is quite common behavior for computer viruses. In combination with any other flag, TBAV reports a virus [TBAV 1989-96].

For example, the standard entry point of COM program is (PSP-SEG:0100H), where PSP-SEG is the segment address of the PSP. TBAV will set the heuristic flag "B" if it finds within a COM file a JMP instruction that transfer control to (PSP-SEG:0100H). For example, if TBAV find the instruction "JMP BX" within a COM file then it will set the "B" flag if BX contain the value 100H and CS= PSP-SEG.

Trick-1
```
;BX= Offset of the first byte after the relative jump instruction
            MOV BX,OFFSET BACK
;BP= Virus Base Address
            ADD BX,BP
```

```
        ;Subtract 100H-bytes (PSP-Size)
                    SUB BX,100H
        ;BX= 2's complement of BX
                    NEG BX
        ;Store the 2's complement displacement of the relative jump instruction
                    MOV CS:[BP+RJP],BX
                    DB 0E9H         ;Relative JMP instruction Op-Code
         RJP        DW (0) ;Relative JMP instruction displacement
         BACK       DB 90H          ;The first byte after the JMP instruction
```

Viruses can avoid this flag by using the piece of code shown in "Trick-1", in which the JMP instruction is not stored in its legal form in the virus code, instead, the virus store the instruction Op-Code (E9H), and generate the required displacement (OFFSET) during its execution. When TBAV scan this piece of code, it will find the instruction (JMP 0000), and therefore does not set the heuristic flag "B".

O: Code Overwrite flag: This flag appears if TBAV detects that the program overwrites some of its instructions. However, it does not seem to have a complete (de)cryptor routine[TBAV 1989-96].

Most shell type COM file viruses set this flag, because, they overwrite the first three bytes "at (PSP-SEG:0100H)" of the COM program before executing it by using, for example, the following two instructions "assuming that DS= PSP-SEG":

```
MOV WORD PTR DS:[0100H], XX      ;XX= The first two bytes
MOV BYTE PTR DS:[0102H],YY              ;YY= The last byte
```

If TBAV encounter these two instructions, or even one of them, it will set the heuristic flag "O". However, viruses can use Trick-2 to avoid this. The piece of code in Trick-2 will overwrite the three bytes started at PSP-SEG:[0100H] using the fact: "(PSP-SEG:0100H) Is Equivalent To (PSP-SEG + 10H:0000H)**".** When TBAV scan this piece of code it will find that the program writes something to segment whose address stored in DS, TBAV will check the content of DS to find that it is not equal to PSP-SEG, and therefore, do not set "O".

```
                              Trick-2
;Set DS= PSP SEG + 10H
        MOV BX,DS
        ADD BX,10H
        MOV DS,BX
;Store 3-bytes at DS:[0000H], DS:[0001H], and DS:[0002H]
        MOV BX,ES:BUFW[BP] ;BX= first word
        MOV DS:[0000H],BX   ;Store BX at DS:[0000H]
        MOV BL,ES:BUFB[BP]  ;BL= Last byte
        MOV DS:[002H],BL            ;Store BL at DS:[0002H]
```

S: Search for executables flag: The program searches for *.COM or *.EXE files. This by itself does not indicate a virus, but it is an ingredient of most viruses, since they have to search for suitable files to spread themselves. If accompanied by other flags, TBAV assumes the file is infected by a virus [TBAV 1989-96].

TBAV set the flag 'S' if the string "*.COM" or "*.EXE" exist in any executable, to avoid this, viruses do not store this string in its legal form in their permanent parts, instead, they create this string during their execution. In fact a voiding this flag is similar to avoiding the flag "B" above.

F: Suspicious file access: TBAV has found instruction sequences common to infection schemes that viruses use. This flag appears with those programs that are able to create or modify existing file [TBAV 1989-96].

The DOS system interrupt (INT 21H) is used to access files. If TBAV find the instruction (INT 21H), it will check its input arguments in the CPU registers, and then it can decide whether the accesses is a suspicious file access or not. Viruses can avoid this flag, if they avoid using the instruction INT 21H. To do this, viruses can use Trick-3. In Trick-3, vector 68H of the user interrupt "INT 68H" is redirected to point to the same interrupt service routine "i.e. DOS-API" as vector 21H, in this case, the instruction INT 68H can be used to access DOS-API. When TBAV scan the virus code, it will never find the instruction INT 21H, therefore, it will think that this executable "the virus" does not access any file at all.

```
                              Trick-3
;Set DS=0 "Segment address of the interrupt vector table"
        MOV BX,0
        MOV DS,BX
;DI= Offset address of the vector 68H
        MOV DI,01A0H
;BX= Offset address of the vector 21H
        MOV BX,0084H
;Set the vector 68H equal to the vector 21H, by coping the double word at DS:BX ;to ;the double
word at DS:DI
        MOV DX,[BX]
        MOV [DI],DX
        MOV DX,[BX+2]
        MOV [DI+2],DX
```

All of the above tricks are used successfully in practice. Even though heuristic scanners can be modified to avoid these tricks, the virus designers may use many other tricks, and then heuristic scanners should be modified again and so on.

B- Neural Network Scanners: IBM now uses neural networks to detect boot sector viruses. The so-called neural network computer programs trained to detect patterns to spot newly hatched viruses that have not been seen before. Neural networks are a form of artificial intelligence in which a computer simulates the way in which human brains process information. A neural network learns pretty much the way a human being does. Neural network designed to detect a new virus that share subtle features with known viruses but does not necessarily match them line for line. The neural network starts with no rules built in. The rules evolve, you show the net a series of key measurements from an infected piece of software then the measurements for a piece of clean software, now here's another infected piece of code, now another clean one, on its own the neural network will pick up a pattern. The problem in this method is that the number of training examples has to be far greater than the number of measurements being assessed. In 1995 Virus Bulletin Conference in Boston, a researcher presents a paper on why neural network could not be used to detect computer viruses "There weren't enough samples to train them". In 1988, Gerald Tesauro, found a way to use neural nets on viruses nonetheless, there aren't enough samples to train the net?. His solution: "Don't feed entire viruses to the net, feed virus particles". Tesauro digested computer code into sequences of three bytes and extracted those likely to be presented in viruses but not in legitimate programs. These triplets then became the bases of a neural network. There turned out to be just enough examples to train a neural network using this approach. The net told Tesauro how many demerits to give for each of the different suspect triplets found on a questionable boot sector. Enough demerits and you put the questioned piece of code in quarantine. The IBM neural network have only three false positives, and all of these were security programs whose codes share many similarities with viruses [Rao 1996].

As a summary this method, just like heuristic scan method, is based on RULE2. Therefore, this method can be tricked using the tricks explained in heuristic scanners or some other tricks. Note that, by using these tricks, viruses perform their infection tasks without using the traditional straightforward techniques on which the neural network was trained. The neural network cannot detect the tricky virus because it is not trained to detect this type of viruses, the tricky virus should be detected using another technique and then train the neural network to detect it. This means that, neural network detection depends on the virus generation date and VDC, hence, it cannot maximize VGS. Therefore: "Neural Networks cannot satisfy the requirement of ideal Code-Based detection**"**.

3- Behavior-Based Detection Scanners: The well known Behavior-Based Detection Scanners are TSR Monitors and CPU-Emulated Generic Decryption:

A- TSR Monitors**:** TSR programs are used for virus detection in DOS machines. Once installed, the TSR monitor must redirect some interrupt vectors to point to its own code so that it can intercept any call to this interrupt. Typical interrupts that are usually redirected by TSR monitors are the BIOS disk interrupt (INT 13H), and DOS interrupt (INT 21H). For example, the TSR monitor can detect viruses that infect their target sites using DOS services by redirecting vector 21H to point to its own handler and, then, monitoring calls to INT 21H. The future for the effectiveness of the TSR as a virus detector is further limited by the fact that viruses can (and probably will) be written to evade the TSRs detection methods [Endrijonas 1993]. In fact viruses can bypass the TSR monitor in the following three situations:

- A TSR Virus Installed Before the TSR Monitor**:** If a TSR virus and TSR monitor installed in the system, and both redirect the same interrupt vector, then: "The TSR monitors can detect the viral activity of a TSR virus only if it was installed before the TSR virus because any virus access to the vector will pass through the TSR monitor. But if the TSR virus installed before the TSR monitor then the TSR monitor can see only normal activities passing through the vector. The virus resides in the vector list after the TSR monitor, so the TSR monitor will never see any activity generated by the virus. The TSR monitor thinks that things are progressing well, while, in reality, the virus is either spreading or doing damage behind the TSR monitor's back [R.,G. 1989]".

  For this reason TSR monitors cannot be used to detect BPS viruses, since these viruses will run and install themselves before loading the OS, and hence before the TSR monitor (see [R.,G. 1989]). To solve this problem most TSR monitors modify the BPS code by another one, the last one will install the TSR monitor during the booting sequence. In fact, even if they have done this they may fail to be the first installed TSR if, for example, the partition sector code of the hard disk was designed to install the TSR monitor and the system booted from a floppy diskette which is infected by a TSR type BPS virus. Clearly, the TSR virus will be the first TSR installed. Further more, if the virus in the floppy diskette infect the partition code of the hard disk, the TSR virus will run and install itself before the TSR monitor in the next booting from the hard disk.
- A Virus Uses (CALL FAR) Instructions**:** Viruses can directly transfer control to the interrupt handler using a far call instruction (CALL FAR) if the start address of the interrupt handler in memory is known. Far call instruction will transfer control directly to the original interrupt handler, any installed TSR handler will be bypassed and hence the virus can accomplish its task without any interception by the TSR monitor. For example, some viruses check things like ROM version number, and know the absolute addresses in the ROMs of the functions they want. By using those addresses, they can bypass

subsequent links (i.e. TSRs) in the vector list, and still do their dirty work. They can also refuse to install themselves if the addresses or version numbers do not correspond to the environment they want [R.,G. 1989].

- A Virus Uses Direct Hardware Access: Just like any other Pure-Software anti-virus, the software TSR monitors cannot prevent the virus from accessing the disks by using the direct hardware access method.

As a summary, some viruses can bypass or deceive the TSR monitor. This means that the effectiveness of the TSR monitor depends on the virus generation date and VDC, hence, it cannot maximize VGS. Therefor: "Pure-Software TSR monitors cannot satisfy the requirement of ideal Behavior-Based detection".

B- CPU-Emulated Generic Decryption: This technique is used to detect the polymorphic viruses, by executing the programs in a "virtual computer" in which the viruses can be tricked to deliver their payloads, they are then detected, analyzed, and zapped [Rao 1996]. The CPU-Emulated Generic Decryption scanner loads any new executable file into a "virtual computer" it creates, and runs the program. If the file is infected the virus executes its decryption routine and is detected by the CPU-Emulated Generic Decryption scanner without any harm because the program is running in a virtual and contained environment. Detected viruses are analyzed and added to scanner database. But some viruses can avoid detection by a CPU-Emulated Generic Decryption scanner and virus writers are likely to find more ways to subvert CPU-Emulated Generic Decryption weaknesses in the future [Whalley 1996].

As a summary, the detection capability of the CPU-Emulated Generic Decryption Scanners depend on the virus generation date and VDC, hence, it cannot maximize VGS. Therefore: "CPU-Emulated Generic Decryption Scanners cannot satisfy the requirement of ideal Behavior-Based detection".

## Inoculators

Inoculators are the generic name of all the anti-virus programs that are designed specifically to prevent the infection and damage. Inoculators must be active (i.e. resident) in the system at any instance of time so that it can monitor and prevent viruses [Gralla 1997]. TSR programs are used as inoculators in DOS machines. TSR monitors can be used in prevention and damage control as follows:

1- VEP-Based Prevention & Virus-Based Damage Control: TSR monitors can function as VEP-Based prevention and Virus-Based damage control anti-virus. Before executing any executable, the TSR monitor can check if it is infected to avoid executing it. The TSR monitor must use a scanner, either as integral part of the TSR monitor or as independent program, to scan the executable for viruses. As shown above, there is no ideal scanner, consequently: "TSR monitors cannot satisfy the requirement of ideal VEP-Based prevention and Virus-Based damage control".

2- VIP-Based Prevention & Cell-Based Damage Control: TSR monitors can function as VIP-Based prevention anti-virus. If the TSR monitor, for example, intercept system calls to INT 13H then it can prevent viruses from infecting the BPS, and if it intercepts INT 21H it can prevent viruses from infecting executable files. Some TSR monitors prevent any other program from installing itself as a TSR program in the system, or at least alert the user about that. This may increase VRP of any TSR type virus.

TSR monitors can function as a Cell-Based damage control anti-virus by preventing viruses from destroying a predefined target cell such as FAT, RD, or BPS. To prevent a virus from destroying the FAT, the anti-virus must test any write-to-FAT activity to see whether it is legal to pass it on, or it is a viral-activity to prevent it. Distinguishing between legal and illegal write-

to-FAT activities is a difficult or impossible task. Because, DOS itself write to the FAT many times during its operation to add or delete file clusters.
As mentioned "Behavior-Based Detection Scanners" , the effectiveness of the TSR monitor depend on the virus generation date and VDC, hence, it cannot maximize VGS. Therefor: "Pure-Software TSR monitors cannot satisfy the requirement of ideal VIP-Based prevention and Cell-Based damage control".

3- Backup-Based Damage Control: Some inoculators keep backups of the BPS, FAT, RD, and CMOS in special backup disk usually called the "Rescue Disk". Since the FAT and RD changed when files added or deleted, the rescue disk must be updated. Updating the rescue disk is a vulnerable spot because there is a possibility that the virus will get the opportunity to access the rescue disk. Therefore: "Inoculators cannot satisfy the requirement of ideal Backup-Based damage control".

## Eradication Programs

In general, backup-based eradication can be used only if a clean backup copy of the infected executable exist. Therefore, almost all of the eradication programs use the Analysis-Based eradication. Even though, determining NAB, NCB, and the required information about their positions is adequate to eradicate the virus from one specific executable. The fact that the eradication program must eradicate any subsequent copy of the virus from any executable in the T-group, suggests that additional information about the detected virus must be obtained. The eradication programs obtain the following information about the detected virus.

- Unique Virus Identifier (VID): "A special string of characters or piece of code which is guaranteed to exist in any copy of the virus, therefore, it can be used to find any subsequent copy of the virus".
- The NAB with a standard rule to determine their positions in any subsequent infected executable.
- The NCB with a standard rule to determine their positions in any subsequent infected executable.
- The original value of any byte changed by the virus during the infection with a standard rule to determine its position in any subsequent infected executable.

The process of obtaining the above information is called "Identification". Identification can be defined as: "The process of obtaining the necessary information needed to eradicate any subsequent copy of one specific virus from any infected executable within the T-group; where (TGS>1)". The identified virus is commonly known as a "known virus". The eradication programs are designed to find any subsequent copy of the known virus by using the unique VID, and then eradicate the virus by using the identification information. Note that the eradication programs uses a special detection method, usually called "Signature Scan method", in which the following general detection rule is used: IF (The VID of the known virus "V" exist in the executable "E") THEN ( "E" is infected by "V").

Obtaining the constant values (NAB, NCB, and VID) is relatively simple. Such information added as constants into the eradication program database. However, formalizing a standard rule to determine the "position" of VID, NAB, and NCB relative to any executable within the T-group is a difficult task. This type of information is variable and depends, in general, on the size and type of the infected executable and the method of infection used. Therefore, this type of information is added to the eradication program as rules rather than constants. The difficulty of deriving these rules is a consequence of the lack of previous knowledge about how the detected virus infects the target site. There are two general methods to obtain the identification information:

1- Backup-Based Identification: Like Backup-based eradication, the identification information can be obtained by comparing a clean backup copy with the infected vulnerable copy. Unlike backup-based eradication, the backup copy is not necessarily available for every executable in the T-group. Instead, some systems use special purpose decoy programs. For example, the

IBM's Immune System for Cyberspace, which is linked to customers computer networks via the Internet. In this system, once an infection has been identified, a copy of the infected program is sent to a central IBM computer. The computer's software provokes a virus into action on a decoy program so that it can be analyzed. The system then compares infected and uninfected decoy programs, using pattern-matching algorithms, to determine the virus structure and the way it causes infections [Garber 1998]. Determining the constants is trivial by using this method. But, formalizing the required rules for variables may need to repeat the comparison on more than one infected executable. Also, this method cannot be used to completely analyze a DCV, because, only the VLC will be attached to the decoy program.

2- Analysis-Based Identification: In this method one of the traditional program debugging utilities, such as DOS DEBUG utility or Microsoft Code View (CV.EXE) Debugger, can be used to trace the virus execution in the single step mode to see what this virus is designed to do. This method need the infected executable disk copy only, and can be used to formalize a trusted rules for the variables during a single analysis. Also, it can be used to completely analyze, absolutely, any computer virus. However, this method is very lengthy, tedious, and complex to use. Also, determining the constants is difficult using this method.

As a summary, the eradication programs used in the DDS can eradicate only 'Known Viruses', therefore: "The eradication programs used in the DDS cannot satisfy the requirement of ideal eradication because they cannot maximize VGS".

## CONCLUSIONS AND DISCUSSION

This paper introduced the basic concepts of the anti-virus systems that include:

1- Definition: A clear definition of the anti-virus and the anti-virus system.

2- Design and Analysis Criteria: A new design and analysis criteria is proposed. This criteria is represented by a group of precisely defined constants, variables, rules, and requirements. The proposed criteria can be used to analyze the efficiency of a given anti-virus system or to compare two anti-virus systems.

3- Characteristics of the ideal anti-virus system: The characteristics of the ideal anti-virus system "according to the proposed criteria" are introduced and explained. Such information is greatly helpful for the anti-virus designers to know what they should do to arrive at the level of the ideal anti-virus.

4- Classification: Almost all of the possible classifications of the anti-virus are introduced. Each class of anti-virus is analyzed by using the proposed criteria, to identify the characteristics of the most efficient anti-virus.

The analysis presented in this paper suggests that DDS anti-virus systems cannot satisfy the requirements of the IAVS. In general, DDS suffer from the following problems:

1- Lack of Knowledge: Explained in sub-heading "Scanners".

2- Standard Protection: Explained in sub-heading "Scanners".

3- Intended Vulnerability: Some peoples claims that the anti-virus developers are themselves the developers of the computer virus. This seems to be believable in case of DDS by knowing that the computer user must pay to the anti-virus developers monthly to update the anti-virus. The anti-virus developers may leave secret vulnerable spots in their anti-virus systems. Later, they can design new strains of viruses that are undetectable by the anti-virus system. Consequently, they can sell new products.

4- User Responsibility: In case of DDS the computer user, with its limited experience, must be responsible for selecting the most suitable anti-virus. The computer user must read a book, a paper, or any other document to know what is the computer virus, how a given anti-virus work, how to use a given anti-virus, which anti-virus is better than others, ..., etc. Other than that many users cannot even understand what is the computer virus or how it is work. Reading about the computer virus seems to be a difficult, tedious, or worthless for many computer users. It is difficult because the user will find complicated, strange, and new terms related to computer security. It is tedious because new viruses appear in, almost, every day. Most of the new viruses cannot be detected by the old or the current anti-virus. The user must know about the new viruses and how he can upgrade the anti-virus or replace it with new ones that can cope with the new strains of viruses. Finally, it is worthless because many of the computer users have specified goals when using the computer. Some of them need to know how to use a specific word processor, others need to know how to use a graphic tool, others need to know how to play a game, and so one. Clearly, understanding the computer virus is worthless for such peoples. In fact, the study of the computer virus must be the task of the security professionals or, in the worst case, the professional application program developers.

The present study developed a new concept to overcome most of the DDS problems. This concept is based on building a "Self-Defence System" in which every target site 'executable' must be capable of protecting itself against viral attacks. In this system each executable will be vaccinated by a special self-defence anti-virus so that it can protect itself against viral attacks.

## REFERNCES

Endrijonas,J. (1993), "Rx PC: THE ANTI-VIRUS HANDBOCK", WINDCREST/McGRAW-HILL.

Fites, P., Johnston,P. , Kratz,M. (1989), "The Computer Virus Crises", Van Nourtrand

Reinhold.Garber,L. (1998), "Antivirus Technology Offers New Cures", IEEE Computer Magazine. February.

Gralla,P. (1997), "How the Internet Works", Macmillan Computer Publishing USA.

Abdul-Hussain H. M. A. and Shabib H. M.(2002) , "Computer Virology: Formal Analysis of Computer Viruses". Journal of Engineering, College of Engineering, University of Baghdad, No.1, Vol 8, March, PP. 103-121.

Kaspersky,E. (1992-1997), "AVP Virus Encyclopedia", Version 1.3.

O'mally,C.(1993), "STALKING STEALTH VIRUSES", Popular Science, January.

Pozzo,M.,M. and Gray,T.,E.(1987), "An Approach to Containing Computer Viruses", Computer & Security, June.

Rao,S.,S. (1996), "THE HOT ZONE", Forbes, Vol. 158 Issue 12, p252, November.

R.,G. (1989), "Virus 101- Chapters 1,2,3,4", Woodside@ttidca.TTI.COM .

"TBAV 1989-1996 (Thunder Byte Anti-Virus) User Manual".

Tischer,M. (1992), "PCINTERN SYSTEM PROGRAMING", Abacus,.

Uffenbeck,J. (1987), " THE 8088/8086 FAMILY: DESIGN, PROGRAMING, AND INTERFACING", Prentice-Hall, Inc..

Whalley,I. (1996), "VIRUS DEFENCE FOR THE FUTURE", Security Management, Vol. 40 Issue 11, p60, November.

| List of Abbreviations | |
|---|---|
| ADWA= Avoid Detection While Active | URAM= Usable RAM |
| ADWI= Avoid Detection While Inactive | VDA= Virus Detected Age |
| BPS= Boot (or) Partition Sector | VDC= Virus Deception Capability |
| CVS= Computer Virus Size | VDT= Virus Delay Time |
| DCV= Divisible Computer Virus | VEP = Virus Execution Probability |
| DDS= Dependent Defence System | VET= Virus Eradication Time |
| IAVS= Ideal Anti-Virus System | VHA= Virus Hidden Age |
| NAB= Number of Added Bytes | VIP= Virus Infection Probability |
| NCB= Number of Changed Bytes | VLC= Virus Loader Code |
| NWE= Number of Wake up Events | VMC= Virus Main Code |
| OS= Operating System | VRP= Virus eRror Probability |
| PWE= Primary Wake up Event | VSD= Virus Searching Domain |
| SWE= Secondary Wake up Event | VSP= Virus Spreading Probability |

| List of Symbols | |
|---|---|
| Max. = Maximum | Min. = Minimum |
| $\leftrightarrow$= Independent ,or Not Affected | |