13

A DETECTION SCHEME FOR THE SK VIRUS

Eng. Dalia Salah[1], Dr. Heba K. Aslan[2], and Dr. Mahmoud T. El-Hadidi[3]

[1] *System Engineer, Khalda Petroleum Company.*
E.mail: Dalia_nour@yahoo.com
[2]*Informatics Dept., Electronics Research Institute, Dokki, Giza, Cairo, Egypt.*
E-mail: haslan@eri.sci.eg
[3]*Professor of Computer Networks, Dept. of Electronics & Elect. Comm., Faculty of Engineering, Cairo University, Giza, Cairo, Egypt.*
E-mail: hadidi@mailer.scu.eun.eg

Abstract: Computer viruses pose an increasing risk to computer data integrity. They cause loss of valuable data and require an enormous effort in restoration/duplication of lost and damaged data. Each month many new viruses are reported. As the problem of viruses increases, we need to detect them and to eradicate them. This paper provides a brief introduction to computer viruses and points to the emergence of more intelligent and targeted viruses. Existing methods of virus detection are discussed. A method for the detection and removal of a macro virus called " SK virus " is described. This is achieved through the development of a scanner written in Visual Basic.

1. INTRODUCTION

In the information era, various software attacks to sensitive information could take place. The damage produced could range from degrading system efficiency to losing valuable data. The virus history began in 1986 when it was realized that the code in the boot sector of a floppy could be replaced with a program that could reside in memory and then copy itself onto any accessed floppy. The authors called their program a virus: all it did was to put a volume label on infected diskettes. In the same year, the first file virus named Virdem, was written as a demonstration that it was possible to do it. Then in 1987, the first file viruses and the first EXE infectors were written in Tel Aviv, including the famous Jerusalem. In 1988, the first anti-virus

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35586-3_46 programs appeared, although some people still claimed viruses were an urban legend. Also in the same year, Robert Morris produced the first Internet worm, Morrison worm. In 1989, some nasty new viruses appeared, and IBM gave their customers copies of their own anti-virus software. At the end of the year the famous Trojan, the AIDS diskette, was distributed and caused widespread damage. 1990 saw the first polymorphic virus, and the first virus exchange Bulletin Board System (BBS) in Bulgaria, where many were written that year. At the end of the year there were about 150 known viruses. By the end of 1991 larger companies were getting into the anti-virus market and there were about 1000 known viruses. Virus exchange BBS's run all over the world, and polymorphic viruses were widespread (in particular, Tequila and Maltese Amoeba). In 1992, Dark Avenger released the Self Mutating Engine, that could make any virus polymorphic, but the anti-virus researchers quickly learnt how to detect viruses using it. The famous Michelangelo virus appeared, and virus authoring packages came out. In 1993, more encryption engines appeared, the Tremor virus became famous after German TV broadcast an infected PKUNZIP.EXE, and some of the anti-virus companies gave up. A Generic Decryption Engine was produced to decrypt suspect files, ensuring that false alarms were not generated. Then, the macro viruses have been invented to attack word processor documents [1].

In the present paper, the development of a new detection and removal tool of a macro virus concerning the Arabic word documents (word 97/2000), known as SK virus, is considered. Methods of detection and removal are detailed. The paper is organized as follows: Section 2 details the various types of software attacks against computer. In section 3, discussion of the basic classification of viruses is given. In section 4, several methods of virus detection techniques are described. In section 5, the SK virus characteristics are presented, and the newly proposed approach for its detection, and removal is given. The paper concludes with a number of remarks in section 6.

2. TYPES OF SOFTWARE ATTACKS AGAINST COMPUTER

Software attacks against computer are namely: time/logic bombs, Trojans, worms, and viruses. Among these types, the virus has the most destructive consequences [2].

Logic/Time Bombs

It is a hidden program but just like a real bomb, a logic/time bomb will lie dormant until triggered by some event/time. The trigger can be a specific data, the number of times executed by another program, a random number, a specific date, or even a specific event such as deletion of an employee's payroll record.

When the logic/time bomb is triggered, it will usually do something unpleasant. This can range from changing a random byte of data somewhere on your disk to making the entire disk unreadable [2].

Trojans

The Trojan program appears to be a useful program, but when a certain event occurs, it will attack the PC in some way [3].

Worms

A worm is a self reproducing program which does not infect other programs as virus does; instead it creates copies of itself, which create even more copies and try to spread to other machines until bringing the network to its knees. Worms use up computer resources such as memory and network bandwidth, slowing down both PCs and servers. In addition, worms sometimes delete data and spread rapidly via e-mail [2].

Viruses

A computer virus is simply a list of instructions that tell computers what actions to execute and precisely how to execute them. Viruses load and run without users requesting them to run; they hide inside normal programs and run when the hosts are run [2]. The virus reproduces its own code by attaching itself to other programs in such a way that the virus code is executed when the infected program is executed.

Besides replication, some computer viruses have something else in common: a damage routine that can deliver the virus payload. While some payloads may only display messages or images, others can also destroy files, reformat hard drives, or cause other kinds of damage. If the virus does not contain a damage routine, it can still cause trouble by taking up storage space and memory, and downgrading the overall performance of a computer.

Several years ago, most viruses spread primarily via floppy disks, but the Internet has introduced new virus distribution mechanisms. With email now used as an important business communication tool, viruses are spreading faster than ever. Viruses attached to email messages can infect an entire enterprise in a matter of minutes, costing companies millions of dollars annually in productivity loss and clean-up expenses.

According to the International Computer Security Association, more than 10,000 viruses have been identified, and 200 new ones are created every month [4]. With numbers like these, it is safe to say that most organizations will deal regularly with virus outbreaks. No one who uses computers is immune from viruses. In the next section, basic classification of viruses are discussed.

3. BASIC CLASSIFICATION OF VIRUSES

Computer viruses could be classified according to the affected location in a computer. Following are the main virus categories.

Boot Sector Viruses

Boot sector viruses (BSV) affect the boot sector. A computer is infected after accessing a disk containing the virus, which checks to see if it is already installed; if not, it installs itself in memory. It replaces the disk boot sector with its own code, and moves the original boot sector to somewhere else on the disk. Thus, it can be loaded after the virus has finished. The BSV runs at every boot, loads itself into memory, and causes a damage, which could lead to the loss of all data on the hard drive. An example of BSV is the Monkey virus [2].

File Viruses

File viruses are divided into two types: Direct Action File Viruses (DAFV) and Indirect Action File Viruses (IAFV). For DAFV, it affects the .COM, .EXE, and .SYS files. Once the virus is loaded into the memory, it searches for an uninfected file and infects it. The virus appends itself to the end of the file, patch the jump to the start so that it jumps to the virus code, and the last instruction of the virus code is a jump to the original program, so that it runs in a normal way. An example of DAFV is the Major Tom virus [5]. On the other hand, IAFV only affects .COM and .EXE files. IAFV installs itself into memory, usually by replacing interrupt 21h or other interrupts that are called regularly. Once the virus is installed in memory, every time DOS functions are executed, the virus has control and can infect lots of other files. The damage produced includes the loss of all data in the hard drive.

Multi-Purpose Viruses

Multi- Purpose Viruses (MPV) combine some or all of the infections attributes of boot sector and file viruses. They affect boot sector, .COM, and .EXE files. By adapting two infectious techniques, MPV achieve a higher level of survivability and have fewer problems reproducing themselves than do viruses possessing a single infectious technique. When MPVs. gain control, they test for viral–markers (V–markers), the coded bytes that identify infected files and can be taken as virus signature. When V–markers are not found, MPVs proceed with the infection of the uninfected file. An example of MPVs is the one half virus [2].

Macro viruses

Macro viruses affect both Word and Excel files when an infected document is opened. The virus modifies the Normal template (Normal.dot). Once a computer is infected, Normal.dot will infect any file when opened. An example of macro viruses is the Melissa virus [6].

Companion viruses

Companion viruses affect .EXE files. A companion virus is a virus that tries not to modify the files themselves but change their execution path instead. It can locate a file with an .EXE extension and put the virus name but with a .COM extension. This type could be easily detected by the presence of the extra COM files [7]. An example for this type is the Win32.Melting virus [8].

Polymorphic Viruses

The polymorphic viruses can affect any file. A polymorphic virus is a virus that has the capability of changing its own code allowing the virus to have hundreds and sometimes thousands of different variants making it much more difficult to detect [9].

A polymorphic virus includes three main components: the first is a scrambled virus body, which is encrypted. The second is decryption routine that first gains control of the computer, and then decrypts the virus body. The last component is a mutation engine that generates randomized decryption routines that change each time the virus infects a new program. The mutation engine is also encrypted. When a user runs a program infected with a polymorphic virus, the decryption routine first gains control of the computer, then decrypts both the virus body and the mutation engine. Next, the decryption routine transfers control of the computer to the virus body, which locates a new program to infect. At this point, the virus makes a copy of both itself and the mutation engine in random access memory (RAM). The virus then invokes the mutation engine, which randomly generates a new decryption routine that is capable of decrypting the virus, yet bears little or no resemblance to any prior decryption routine. Next, the virus encrypts this new copy of the virus body and mutation engine. Finally, the virus appends this new decryption routine, along with the newly encrypted virus and mutation engine, onto a new program [9].

4. VIRUS DETECTION TECHNIQUES

In this section we review current methods of virus detection techniques.

Signature Scanning

It is a static analysis detection tool [10], where the detection is performed by pattern matching. An executable is searched for selected binary code sequences, called a virus signature, which are unique to a particular virus. The virus signatures are generated by examining samples of the virus. The advantages of signature scanning are: firstly, scanners identify both the infected executable and the virus that has infected it. Secondly, up-to-date scanners could be used to detect more than 95 percent of all virus infection

[10]. Due to the fact that detection is achieved by scanning specific patterns, scanners are limited to the detection of known viruses. Thus, the signature algorithm must be updated frequently. Another disadvantage of the signature scanning is that it suffers from the possibility of occurrence of both false positives and false negatives. While, false positive occurs when an uninfected executable includes a byte string matching a virus signature in the scanner's database, false negative occurs when an infected executable is scanned but no pattern match is detected. An example of signature scanning method is *Dr Solomon's Anti-Virus Toolkit* which offers facilities for scanning and memory- resident checking [11].

Integrity Checker

An integrity checker (also known as a checksummer) is a program that determines whether another program has been altered or changed [10, 12]. An integrity checker can only flag a change as suspicious; it cannot determine whether it is a virus infection or other normal changes. The system must be virus-free when the checksums are calculated; resident viruses may fool the integrity checker. Integrity checker employs two basic mathematical techniques:

a. Cyclic Redundancy Checks (CRC): the calculation of CRC is based on a known set of algorithms. Therefore, it can easily be broken. To enhance the security of the system, the checksums must be stored encrypted as done in cryptographic checksums technique.

b. Cryptographic checksums: In this technique, a hash function is applied to the file which the checksum is calculated for. Then, a cryptographic algorithm is applied to calculate the cryptographic checksum. The advantages of the integrity checker are: firstly, it can detect every virus, i.e. no false negatives. Secondly, it does not need updating. However, this technique suffers from the possibility of a high number of false positives. Another disadvantage is that it cannot find viruses, only changes. Finally, the efficiency of this method relies on the fact that the system must be virus-free when the checksums are calculated. Examples for integrity checker techniques are: *ASP Integrity Toolkit, Integrity Master [11], and Samhain file system integrity checker* [1].

General Purpose Monitors

The designer of such a system begins with a model of malicious behaviour, then builds modules which intercept and halt attempts to perform those actions. These modules which operate as a part of the operating system are intended to sound an alarm every time a software package performs some suspicious action considered to be virus like behaviour[12]. It helps in real-time detection of viruses and Trojan horses. The main advantage of the general purpose monitors is that it can detect infections before its occurrence. However, it suffers from several disadvantages such as: new viruses may utilize new methods that may fall outside the model of suspicious behaviour in the monitoring program. Such a virus would not be detected. Another disadvantage is that there are some viruses that turn off the monitoring program and fool the monitor. Examples for general purpose monitors are: *Secure*, and *Gatekeeper* [11].

Access Control Shell

Access control shell function as a part of the operating system. The shell attempts to enforce an access control policy for the system such that it will sound an alarm every time a user attempts to access or modify a file with an unauthorized software package [12]. Access control shells perform real-time detection of viruses and Trojan horses. The access control policy must be rebuilt each time software or hardware is added to a system, job descriptions are altered, or security policies are modified. It can detect infections before its occurrence as in the case of general purpose monitors. Also, the access control shell suffers from the possibility of occurrence of false negatives.

The Norton Utilities package includes a facility for protecting the disk's system areas and/or its files: attempts to write are intercepted and your approval is requested before the write is allowed.

Polymorhic Detection

Two types of polymorphic detection exist: the generic decryption technique and the heuristic- based generic decryption technique [9]. In the following, a brief discussion of these two types is given.

- Generic Decryption

Generic decryption assumes:

- The body of a polymorphic virus is encrypted to avoid detection.
- A polymorphic virus must be decrypted before it can execute normally.
- Once an infected program begins to execute, a polymorphic virus must immediately usurp control of the computer to decrypt the virus body, and then yield control of the computer to the decrypted virus.

A scanner that uses generic decryption relies on this behaviour to detect polymorphics. Each time it scans a new program file, it loads this file into a self-contained virtual computer created from RAM. Inside this virtual computer, program files execute as if running on a real computer. The scanner monitors and controls the program file as it executes inside the virtual computer. A polymorphic virus running inside the virtual computer can do no damage because it is isolated from the real computer. When a scanner loads a file infected by a polymorphic virus into this virtual computer, the virus decryption routine executes and decrypts the encrypted virus body. This exposes the virus body to the scanner, which can then search for signatures in the virus body that precisely identify the virus strain.

- Heuristic-Based Generic Decryption

Generic decryption employs "heuristics," a generic set of rules that helps differentiate non-virus from virus behaviour. Heuristic-based generic decryption looks for such inconsistent behaviour. An inconsistency increases the likelihood of infection and prompts a scanner that relies on heuristic-based rules to extend the length of time a suspect file executes inside the virtual computer, giving a potentially infected file enough time to decrypt itself and expose a lurking virus.

Agent Based Virus Detection

Mobile agents are agents that can physically travel across a network, and perform tasks on machines that provide agent hosting capability. This allows processes to migrate from computer to computer, for processes to split into multiple instances that execute on different machines, and to return to their point of origin.

A research programme at the University of Southampton is currently considering the application of Mobile Intelligent Agents in the detection of previously unseen, intelligent computer viruses [13]. The decision to investigate the application of mobile intelligent agents in this domain was based on the following factors:

- mobile agents are less vulnerable to attack.
- mobile agents are able to penetrate all areas of a network and therefore fully exploit system redundancy.

The research programme comprises three stages:

Stage 1 : Replication Detection aimed at defining and developing an agent capable of detecting the replication of a virus through an Information System.

Stage 2 : Similarity Assessment aimed at investigating and developing a method for the identification of similarity between code sentences, where either evolution or encryption is being used to mask the existence of a computer virus

Stage 3 : Intention Discrimination aimed at investigating and developing a method of discriminating between beneficial and malicious code sentences.

The development of an effective defense against previously unseen viruses requires the identification of an invariant feature which must exist during every virus attack. The extraction of viral signatures is clearly an ineffective method of protection as the features identified are not invariant. By definition there is only one invariant feature exhibited during every viral attack; that feature is the replication of the viral code sentence, or an evolution of it, across the Information System. This raises the question of whether it is possible to identify the existence of this feature by exploiting the redundancy inherent in a modern Information System. Clearly the number of agents deployed to detect the virus will have a significant bearing on the defense system's effectiveness.

In the next section, description of a detection technique for a macro virus concerning the word document called the SK virus is detailed.

5. AN APPROACH FOR DETECTING AND REMOVING THE SK VIRUS

The SK Virus first appeared two years ago during the election of the Egyptian President Mohamed Hosni Mubarak for his third term of office. The SK virus cannot be detected by signature scanning because the scan string approach is not very adequate for detecting Visual Basic Application (VBA) viruses in the first place [14]. The internal representation of a VBA (VBA3 or VBA5) program consists of two–byte opcodes, each followed by one or more (zero– to eight–byte) operands. The operands are very often pointers to different complex data structures and are, therefore, variable. That is, one and the same VBA program can be represented by one and the same sequences of opcodes which sequences, however, have different contents of their operands—simply because the data structures they point to reside in different places. In practice, the above means that a VBA module consists of constant areas which can be as small as two bytes and which are separated by zero– to eight–byte areas with variable contents. Therefore, a scan string for a VBA virus must often be rather long and contains lots of wildcards. Moreover, it is possible to have two different VBA programs which consist of one and the same opcodes—and which can be differentiated from each other only if the pointers in the operands of these opcodes are resolved to the data each of them points to. The consequence is that even a carefully chosen scan string of a VBA virus can match some other, legitimate program and therefore cause false positives. That is why we strongly discourage the usage of scan strings for VBA virus detection and recommend exact identification instead.

Virus Description

The SK virus is a virus for Word97/2000 documents and templates. It will lower the macro warning option in Word. In addition, it will remove the check on ' Macro virus protection ', which is found in Tools/ Options so that when we open an infected file, it will not ask to disable or enable macros. The virus body is contained in a macro named Mxfile.

Depending on the system month/date, it will insert some text in an Arabic language into the document. If the Days Mod Months = 0, then the text "لكرابم عيابن انلك " is inserted in the header and footer.

Once the system is infected, the Mxfile runs in case of: Starting MS Word program, Creating a new document, Opening a document, Closing a document, or Quitting Word.

Indications of Infection

Macro warning when opening infected documents on non-infected system is displayed. When opening *Tools / Macros,* the macros will not be displayed.

The Mxfile module will be found when opening *Tools / Macros/Visual Basic Editor,* then the next screen will appear:

```
PROJECT - ANORMAL
    ANORMAL
      MICROSOFT WORD OBJECT
           MODULES
             MXFILE
```

Method of Infection

The first step in the virus attack of an uninfected system is to infect the normal template (Normal.dot). Then, it disables the macro warning option in *Tools / Options*. Therefore, in case of opening an infected file, it will not ask for enabling or disabling macros. Thus, any operation on any file (new, open, close, or exit) cause the transfer of the virus to this file by copying the Mxfile – virus body – to this file.

Virus Detection and Removal

Our scanner uses the Organizer dialog box to check for the virus macro (Mxfile) attached to the file tested for infection. The Organizer can open a document in the background (without running any attached macros) and lists the macros attached to it. Also the Organizer could be used to delete macros from a document. There are two ways to open the Organizer:

1. Use the *Tools/ Macro* command and press the Organizer button;
2. Use the *Tools/ Templates* and add- INS command and press the Organizer button.

To remove infection, our scanner executes the following steps:

Step 1: Clear the normal template (Anormal.dot) from the normal path.

Step 2: Restart the computer to load the clean normal template.

Step 3: In the organizer dialog box, the file or directory to be tested is chosen. Then, the Scanner searches for the Mxfile and if it finds it, it will delete it. It then deletes the inserted Arabic text from header and footer.

The scanner is written using the Visual Basic programming language, and the detailed flow chart of the above mentioned steps is illustrated in Figure 1.

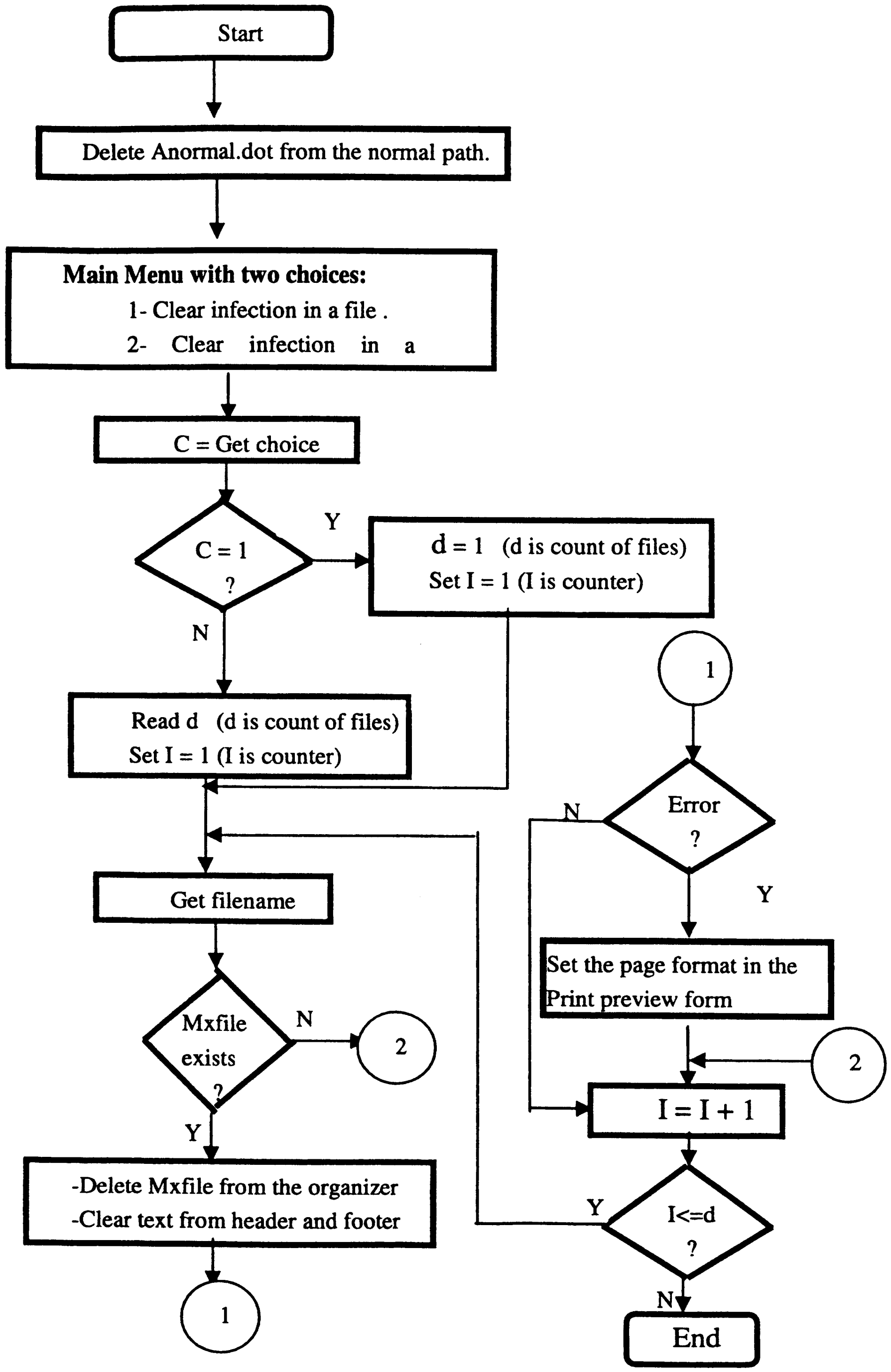
Start
Delete Anormal.dot from the normal path.
Main Menu with two choices:
1- Clear infection in a file .
2- Clear infection in a
C = Get choice
C = 1 ?
Y
N
d = 1 (d is count of files)
Set I = 1 (I is counter)
Read d (d is count of files)
Set I = 1 (I is counter)
Get filename
Mxfile exists ?
N
Y
2
-Delete Mxfile from the organizer
-Clear text from header and footer
1
Error ?
N
Y
Set the page format in the Print preview form
2
I = I + 1
I<=d ?
Y
N
End

Figure 1 Flow chart representing the operations done by the new scanner during detection and removal of the SK virus

6. CONCLUSION

Computer viruses pose an enormous threat to different computer resources. One of the most popular viruses are viruses written in the macro programming language of the office applications like Microsoft Word. This type of viruses known as macro viruses becomes extremely widespread. In the present paper, we have discussed the development of a new technique for eradicating a special macro virus concerning Word documents named the SK virus. The new technique efficiently cleans the SK virus and removes it from infected files. Among proposed improvements to the new technique is the enhancement of the algorithm to detect the SK virus before infection and the elimination of the possibility of file corruption. Other improvements could be done through the study of several macro viruses' behavior and generalizing the new technique to detect these macro viruses. This will lead to a more versatile technique, which could be used to detect different macro viruses other than the SK virus.

REFERENCES

[1] " A Short Course on Computer Viruses" , www.man.ac.uk/virus/

[2] Richard B. Leuin, The Computer Virus Hand Book, Osborne, McGraw. Hill, 1990.

[3] "Cknow.com", http://www.cknow.com/vtutor/vttrojan.htm

[4] "PC Tune - up USA" , http://www.pctusa.com/virus.html

[5] Kevin Powis, "Ground Control to Major Tom", http://www.virusbtn.com/virusinformation/major1644.html

[6] Lan Whalley, " Melissa the little virus that could...", http://www.virusbtn.com/virusinformation/melissa..html

[7] Vesselin Bontchev, " Possible virus attacks against Integrity Programs and how to prevent them" , http://www.complex.is/~bonchev/papers/attacks.html

[8] "sophos virus information", http://www.sophos.com/virusinfo/analyses/w32melting.html

[9] Symantec, "Understanding and managing Polymorphic viruses" , The Symantec Enterprise Papers. http://www.symantec.com/avcenter/reference/striker.pdf.

[10] Sandeep Kumar and Eugene H.Spafford, "A Generic Virus Scanner in C++", -+Computer Security Applications Conference, Los Alamitos, CA, U.S.A., pp. 210 – 219, December 1992.

[11] "All you want to know about computer viruses", http://www.linkks.com/tutorials/viruses2.htm

[12] W. T. Polk and L. E. Bassham "A Guide to the Selection of Anti-Virus Tools and Techniques" , National Institute of Standards and Technology, Computer Security Division. http://csrc.ncsl.nist.gov/publications/nistpubs/800-5/select/table of contents3.1.html.

[13] J. Luke, and C J Harris, "Application of CMAC Based Intelligent Agent in the Detection of Previously Unseen Computer Viruses", International Confrence on Information Intelligence and Systems Proccedings, pp. 662-666, 1999.

[14] Vesselin Bonchev," The Problem of WordMacro Virus Upconversion", Computer & Security Vol.18, No.3, pp.241-255, 1999.