Virus Protection

Pavan Verma
EECS Department, University of Michigan
pverma@eecs.umich.edu

Computer viruses are today the most well-known and widespread threat to the security of computer systems. Infact, some viruses such as Chernobyl, Melissa and The Love Bug spread so rampantly that they became common household names. Till now most of the viruses that were widespread were written either as an experiment to gauge the extent of their proliferation or for amusement. They were not written to cause direct damage (such as deleting files on the infected host). In spite of this, these viruses were a menace because of the enormous network bandwidth they consumed. However, the real threat is that someday a virus would get written that spreads to a millions of computers and is armed with a deadly payload. It is unimaginable as well as almost impossible to correctly estimate the disastrous effect such a virus would have, especially on today's computer-driven economy.

What is a virus?

The first thing in a study of viruses is obviously knowing exactly what one is. Unfortunately, it is not easy to find a satisfactory definition because of several reasons. Over the years, the term virus has become a wide ranging and over-exploited term. Infact many computer security problems are wrongly attributed to viruses by the media and the layman public at large. Often a virus gets confused with a worm or a Trojan horse. Admittedly, this is sometimes unavoidable because a "good" virus (one that spreads rapidly, avoids detection and causes lot of damage) often has several characteristics of a worm and Trojan horse, and vice-versa. There is also a wide variety of programs that have been legitimately called a virus, making convergence to a definition even harder.

First of all, a virus is just another computer program, written in a similar way as other normal programs. Infact, anybody with even the most modest programming knowledge can write one [Coh86, Adl88, Duf89]. The most distinguishing property of a virus is that *the virus program copies itself to other programs or documents so that the virus code is executed whenever the program is run or the document is opened* . Programs to which the virus copies or attaches itself are said to be *infected* with the virus. Most commonly, whenever a virus runs on the local host it searches for uninfected files and tries to infect them too.

One of the things that is presumed about viruses is that they are malicious in nature. Although this is true for most viruses, it is not strictly correct because non-malicious viruses are certainly possible although not common. In this entry, we limit our discussion to malicious viruses. The malicious action of viruses includes but is not restricted to: deleting or zeroing the files, trashing the BIOS, leaving backdoors, spying private information, using the infected machine to mount DoS attacks, etc. Even if the virus does not perform any such destructive activity, it might impede the normal working of computer systems by causing too much network (email) traffic or CPU load. If it doesn't do any of these too, a virus is just annoying to have on your machine simply because it is potentially a malicious, unreliable, unwanted and unsafe piece of foreign code.

Often, the virus' malicious action is triggered by a time bomb or logic bomb. These are pieces of code that get activated when a certain date or time is reached (time bomb) or when some given logic condition becomes true (logic bomb). For example, the famous `CIH` or `Chernobyl` virus was triggered to destroy files on the infected machine based on a time bomb that would go off on exactly the 26^{th} of April, the date of the Chernobyl nuclear disaster.

Infection of files

This section briefly describes how a virus infects a file. Traditionally, most of the viruses have infected executables. This is because the goal of a virus is to run on the local host and the way to achieve this is through an executable. Recently however, a new category of viruses called macro-viruses have developed that attach themselves to document files and are able to run whenever the document is opened. Being able to cause the same effect by infecting data files rather than program files makes macro-viruses much more threatening. In this section, we discuss the two types of viruses in turn.

Traditional executable virus

The traditional virus targets executables and either overwrites the entire file or attaches itself to it so that the virus also runs whenever the executable is run. Viruses normally attach themselves in such a way that the virus is run first and then the program proceeds normally. Such a strategy ensures that the virus runs even in cases when the infected program is a daemon-like process that never halts. Moreover, it makes detection harder because the program seems to perform as expected.

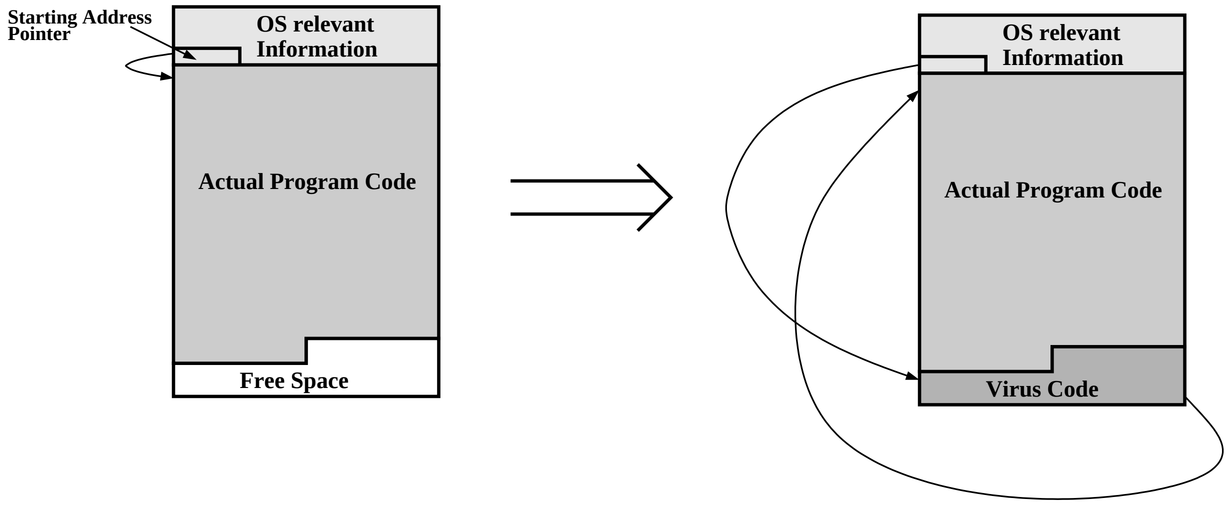

Figure 1: How a Virus Attaches itself to a binary

Attaching a virus to text-based executables (such as shell scripts) is trivial – just put the virus code in the beginning – but suffers from the obvious disadvantage of being easily detectable if anybody happens to view the code. Attaching a virus to a binary executable is more complicated but has the advantage that the virus is better hidden and this technique is also more widely applicable. In this entry, we present a very general overview of how a virus attaches itself to a binary. All operating systems have a minimum unit of hard disk access called block (it is often 512 or 1024 bytes). Files on the hard disk occupy an integer number of blocks, thus their size is a multiple of the block size. This forces each file to have some amount of free space (zeros) in it. For example, if the block size is

512 bytes, and if a program actually needs only 998 bytes, it would occupy 2 blocks on disk, of which 998 bytes are the program itself and 26 bytes are unused. Viruses fit themselves within this unused space and thus do not require any additional blocks. Figure 1 gives a graphical picture of how this is done. It shows the structure of a binary before and after a virus attaches itself to it. The virus is able to modify the binary so as to fit in the available unused space, run before the program and then let the program itself run. Note that the binary contains a location called the *starting address pointer* which points to the first instruction of to be executed and is needed by the operating system to load the binary. The virus, after fitting itself into the unused space modifies the starting pointer to point to the virus' first instruction. At the end of its own code it includes a jump to the first instruction of the program, allowing the program to run after it is done.

Macro-virus

Some software packages allow their data files to contain script like code that is executed when the file is opened. Viruses exploit this feature by attaching themselves to data files in the form of a script. The most common software vulnerable to such viruses are Microsoft Word and Microsoft Excel. Word and Excel files can contain macros (VBScript code) that is executed when the document is opened. As Word and Excel are widely used throughout the world, this forms a very attractive way for viruses to spread.

Virus propagation and the first run

In the previous section, we described how a virus infects a file. However, we have not yet discussed how it initiates the infection. In this section, we describe this process which requires two things: firstly, the virus needs to reach the host (propagation) and secondly, it needs to run on the host at least once in order to initiate the infection.

For propagation, viruses can potentially use any communication medium used to connect computer systems. Since the widespread deployment of computer networks, they have become the de facto medium. More specifically, some of the common ways viruses have spread are:

1. Email has been the most popular transport for viruses in the last few years. Melissa was the first virus to spread through email. Since then, Happy 99, Worm.ExploreZip, BubbleBoy, The Love Bug and others have also used it. The Love Bug sent out emails from the infected host to addresses it obtained from Microsoft Outlook's address-book. The message had the subject "I Love You" and asked the recipient to open the accompanying attachment which of course was the virus executable. People got pulled by the love message into opening the attachment which let the virus loose on their machine. BubbleBoy exploited a feature in Microsoft Outlook that allowed it to execute code on the local host when the email was shown on the preview pane. Another common feature viruses have exploited is that some operating systems (such as all versions of Windows) decide what action to perform on a file (whether to execute it, open it with Microsoft Word, etc.) based on just the file's extension, and they take this step automatically without requiring user intervention. Thus, an intelligent combination of users' unsuspecting actions, social engineering, software bugs/idiosyncrasies along with email as the underlying transport has been very successful for viruses.

2. A very common way to get viruses on your machine is by downloading infected files from the Internet or bulletin boards. When the infected file is opened, it infects the local machine. Infact, this strategy is often employed by virus writers use to launch their viruses: they post their

infected files on the Internet or some newsgroup as supposedly useful programs or documents. Of course, when the file is downloaded and the opened, the local machine gets infected with the virus.

3. Before networks became widespread, floppies were the most common medium through which viruses moved from one machine to another. There are two ways in which this is done. The first way is to use floppies to move infected files between machines. For example, an infected program file `format.exe` is copied from the infected machine, `M1` to a floppy and from there to a clean machine `M2`. When this `format.exe` is run on `M2`, it also gets infected. The second way is to infect a floppy's boot sector so that whenever the floppy is used in a machine (for any purpose) the virus gets transferred to the machine.

4. A common technique viruses employ to ensure that they get activated on an infected host is to install themselves in the boot sector or partition sector of the host's disk drive. This activates the virus every time the system boots up. These viruses are significantly more difficult to remove with surety and the only failsafe method seems to be to rewrite the disk's partition and boot sectors.

Guidelines to prevent virus infection

As described above, very often viruses spread simply because of unsafe usage practices. Curbing some of these would greatly reduce the risk of virus infection. Some guidelines for safe computer usage from a perspective of virus infection are:

1. Do not carelessly open executables or macro-supported documents downloaded from the Internet or received as email attachments. If there is any way to verify the authenticity and integrity of such files using digital signatures or cryptographic checksums, it should be done. If such techniques are not available, the least that should be done is to download files only from reputed websites, check with the person who sent the email and pass the file through an anti-virus software.

2. If possible, support for macros or similar scripting ability in documents should be disabled. In particular, macro support in Microsoft Office software such as Microsoft Word and Microsoft Excel should be turned off.

3. Do not allow operating systems to hide file extensions from the user or make security critical decisions (such as opening a file received as an email attachment) on its own.

4. Be extremely careful when booting systems from floppies. Firstly, floppies should not be carelessly left in drives because many systems have their BIOS configured to first try to boot from a floppy. If it is necessary to use a floppy to boot a system, it should be thoroughly checked to be clean of viruses.

5. Most viruses are operating system specific. Thus, having a heterogeneous computing environment greatly helps in ensuring that not all machines get infected or compromised at the same time.

Anti-Virus software

Anti-virus software has become more and more important over the last few years and has become a necessity on the more vulnerable operating systems. Even though users can adopt safe usage practices such as those mentioned above, viruses still get through. Over fifty thousand viruses exist today and new ones get written everyday. Keeping track of all these viruses and protecting a machine from them is certainly very difficult, if not completely impossible, without anti-virus software. Even though it has been theoretically proved that it is impossible to build software that is always able to correctly determine whether a file is virus-infected or not [Coh86], anti-virus software is definitely a potent and effective weapon against all known viruses and to some extent against new viruses too.

Many anti-virus software are available today in the market and all of them employ a combination of heuristics to detect viruses. The heuristics can be put into two categories: *static* heuristics and *dynamic* heuristics. Static heuristics go through a file, analyzing its structure and looking for malicious patterns, and use this information to decide whether the file contains a virus. On the other hand, dynamic heuristics set up a controlled virtual environment in which they run the program (or open a document) and observe its behavior to see if there is any malicious activity. Based on these observations, it is decided whether the file is infected. As both the static and dynamic schemes are heuristics, they are not always correct. The exact details of the scheme determine the trade-offs between the false positive and false negative rates as well as efficiency. However, in general, static schemes have the advantage of being fast whereas dynamic schemes provide a lower false positive rate. Dynamic schemes though are often susceptible to the logic and whims of viruses (because they involve actual execution of the virus) affecting their false negative rate.

The basic strategy to use heuristics is to maintain a database of "signatures" of known viruses and checking files against this database to check if it is infected. For static heuristics, signatures are code segments (or some function over them) of already known viruses and other patterns of malicious virus-like code. For dynamic heuristics, signatures are malicious requests to the operating system or patterns of malicious activity. The technique of maintaining a virus signature database and then checking all files against it works pretty well for known viruses but fairs poorly against new ones simply because chances are that they would not match any of the known patterns. Thus, it is safe to say that an anti-virus software is only as good as its signature database. Therefore, anti-virus software companies need to continuously update their database with new signatures (as new viruses are discovered) and include update mechanisms in their software to download the updated database to the customer's site. Apart from signature matching, some other heuristics that are used are based on detecting changes in files based on either cryptographic checksums or file size.

The battle between virus writers and anti-virus software companies is analogous to an arms race with each side getting an upper hand sometime but the other side soon being able to catch up. For example, if an anti-virus software came up with some really effective technology to detect yet unknown viruses, it can be safely assumed that virus writers will find some way of beguiling it.

The latest in viruses

Unfortunately for the security community, viruses are an evolving technology so that new, harder to detect viruses are being written everyday. In particular, two of the latest types of viruses are the polymorphic and encrypted viruses aimed at making detection harder. Polymorphic viruses change themselves (i.e. change the code) every now and then, allowing little time for detecting any single manifestation. Encrypted viruses encrypt the virus code so that it does not match any regular patterns. The encryption key can also be changed, and this results in a polymorphic encrypted virus.

Not only are viruses getting harder to detect, they are also getting faster in spreading and [SPW02] discusses some techniques that can be used for this.

Where to learn more about viruses

Viruses are not only a vast area of study but also involve a great amount of relevant detail which is impossible to provide in a short encyclopedia entry. Thus, the goal here was to give only an introduction to viruses and if possible initiate the interested reader into further exploration of the field. Therefore, it is important to provide references to some more in-depth material for future study.

Some of earliest ground-laying academic work is presented in [Coh86, Coh89, Adl88, Isr87]. [Duf89] presents a simple virus on the UNIX operating system, and greatly helps in demystifying viruses in general and the process of writing them in particular.

Most security books include a chapter or at least a few sections on viruses. In particular, [Den90] is a good reference.

The CERT Coordination Center [CER] is an excellent reference for practical details about viruses. It keeps an up to date list of viruses describing their symptoms, operating systems affected, safeguards, etc.

Some good online sources for reference are `viruslist.com` [Vir] and the websites of two major anti-virus software providers: McAfee [McA, NAI] and Symantec [Sym]. [Vxh] is a site for hackers and among other things, it contains virus code and material on how to write a virus.

References

[Adl88] L. Adleman. An Abstract Theory of Computer Viruses. In *Advances in Cryptology – CRYPTO '88 Proceedings*, pages 354–374, New York, NY, August 1988.

[CER] The CERT Coordination Center. `http://www.cert.org`.

[Coh86] F. Cohen. *Computer Viruses.* PhD thesis, University of Southern California, January 1986.

[Coh89] F. Cohen. Practical Defenses Against Computer Viruses. *Computers and Security*, 8(2):149–160, April 1989.

[Den90] P. Denning. *Computers Under Attack: Intruders, Worms and Viruses.* Addison Wesley, 1990.

[Duf89] T. Duff. Experience with Viruses on UNIX systems. *Computing Systems*, 2(2), 1989.

[Isr87] H. Israel. Computer Viruses: Myth or Reality? In *Tenth National Computer Security Conference Proceedings*, pages 238–244, September 1987.

[McA] McAfee Security. `http://www.mcafee.com`.

[NAI] Network Associates Technology, Inc. `http://www.nai.com`.

[SPW02] Stuart Staniford, Vern Paxson, and Nicholas Weaver. How to Own the Internet in Your Spare Time. In *Proceedings of the* 11*th* *USENIX Security Symposium*, August 2002.

[Sym] Symantec Corporation. `http://www.symantec.com`.

[Vir] Viruslist.com. `http://www.viruslist.com/eng/`.

[Vxh] VX Havens. *http://vx.netlux.org/index.shtml*.