University of Southern Denmark

DM830 - Network Security

Malicous Software

Author 1:
Mikkel Secher Jespersen
101086

Author 2:
Felix Palludan Hargreaves
310587

May 7, 2012

Contents

1 Introduction

There is a war going on in the world of network security.
Programmers are trying to make more and more advanced viruses, and antivirus programmers are trying to find new ways to defend against the growing threat of malicious software.
Malicious software generally fit into three categories:

- Virus - parasitic behaviour, infects and spreads.
- Worm - self-replicating and propagating (but does it as a stand-alone program)
- DoS/DDoS - Denial of Service - overloads systems crippling them, and thereby blocking regular use.

Other security threats include back doors which are intentional openings in system security, put in by programmers (usually for debugging purposes). If a programmers needs to run a test on a secure system with intricate login and authentication procedures, a back door makes it easy to circumvent these by using f.x. a special login and password which grants administrator privileges. This becomes a threat if a hacker learns of this back door, or the programmer decides to use it for sinister reasons.
Logic bombs are included in most viruses in one form or another. They are characterized by being activated by a certain event. These could be a certain date and time, the user opening a specific program, the absence of certain files and much more.
Another threat is Trojan horses which are apparently useful programs, that include dangerous code that is run when the program is opened. These programs can be hidden so the user does not know that bad stuff has happened. The useful program could have the full functionality expected of it, but in the background change the rights of files on the system, so they can be read from the outside. They could also open up a back door in the system and let in a hacker.
Mobile code uses virtual runtime environments like the Java Virtual machine, ActiveX, VBScript and Java Script, to makes malicious software that can be run on several platforms, like operating systems or browser.

2 Viruses

Viruses work much like they do in nature. A biological virus is a scrap of genetic code that can jump into a cell and change it, so that it starts producing perfect copies of the virus to infect other cells.
Computer viruses are scraps of machine code that is able to inject its code to other files and do harmful things.

2.1 Properties

A virus can do anything that other programs can, but instead of being stand-alone programs, they attach or inject themselves into other programs, and their code is executed secretly while the program runs. Viruses generally consist of three parts:

Infection Mechanism

This is how the virus infects other programs i.e. how it replicates.
An infection mechanism could be adding a jump in the beginning of the file, to the virus code and then jump back and continue to run the program as usual. If this is implemented fast enough, the user may never know that the virus code was run.

Trigger

This is an event or a condition that triggers the virus payload. For example a specific input to the host program, the presence of a specific file or a certain date and time.

Payload

This is the actual harmful code of the virus.

2.2 Lifetime of a virus

Typical the lifetime of a virus can be split into four phases.

Dormant phase (optional)

The virus can be completely idle for a time, where it does not do anything. This phase is not in all viruses, but a reason for it to be there, could be to hide from antivirus software scanning for typical virus behaviour, or to make it harder to detect the virus when it is run in a sandbox environment.

Propagation phase

This is the phase where the virus spreads to other programs or systems. This is not necessarily an exact copy of itself, it could be infection of certain areas of the operation system, where it can them infect other files from. It could also be morphing itself to avoid detection by antivirus software.

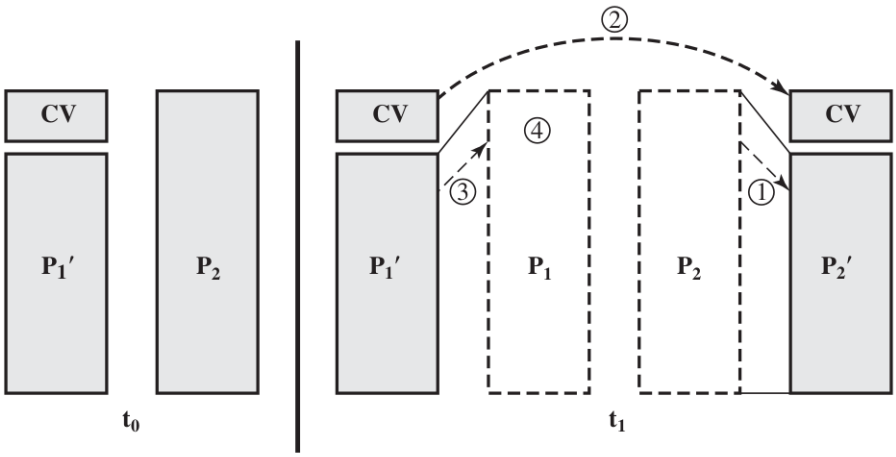
Triggering phase

The virus is activated to perform the harmful activity it was meant to. It could be waiting until it have propagated to a certain number of files, or it could be triggered by a specific system event, or an action by the user.

Execution phase

The function of the virus is executed. It could be something relatively harmless as posting a message on the screen but it could also be something more sinister like corrupting or erasing files or stealing data from the system.

2.3 Structure



1. For each uninfected file P_2 that is found, the virus first compresses that file to produce P_2', which is short than the original program by the size of the virus.
2. A copy of the virus i prepended to the compressed program.
3. The compressed version of the original infected program, P_1', is uncompressed.
4. The uncompressed original program is executed.

Figure 1: A classical compression scheme for a computer virus (From the course book p.348)

Figure 1 shows a typical compression scheme used by viruses. The main reason for the use of compression is to avoid detection by simple virus scanners. If the original file would change in size, the infection would be easily detected by the simplest of virus scanners.
Some stealth use extremely sophisticated methods of keeping hidden like in-

tercepting logic in disk I/O routines, when antivirus software is scanning for it.

2.4 Classification

Although there are no standard for classifying viruses, the producers of antivirus software have found it helpful to classify them by two categories independent of each other.

Classification by target

- Boot sector infector - Infects the master boot record of a system and spreads only when the system is booted from a disk containing the virus. These viruses are relatively rare now.
- File infector - By far the most common type of virus now. It works by infecting regular executable files.
- Macro virus - Can infect filetypes that allow for the execution of macro code when run. This could for example be MS Office files, that allow for including macro code that is interpreted by the Office software, and thereby has priveledges to infect other files of the same type.

Classification by concealment strategy

- Encrypted virus - These viruses generates a key to encrypt most of the virus, in order to avoid having a specific signature that can be searched for by antivirus software. The key will constantly be changed to further confuse the antivirus software.
- Stealth virus - A virus that is designed explicitly to completely hide itself from detection by anti virus software, so everything is hidden so the the user will never know that it was there. This type could for example be used to gather login information to a bank and send it back to the virus programmer, and then erase itself, to make sure the user never knew it was there.
- Polymorphic virus - These viruses rewrites their code with every infection, but keeps their functionality. This makes it harder to find a signature for the virus.
- Metamorphic viruses - These are much like polymorphic viruses, but they can also change their behaviour.

2.5 Virus kits

Virus kits are typically configurable implementations or source code intended for the creation of new generic malware. These can range from simple *point and click* GUI tools to more sophisticated frameworks for the swift development of advanced viruses and other malware. Virus kits present a threat mainly due to the mere addition of viruses that could potentially be created by arbitrary

people with simple computer knowledge. In the malware community (and more generally on the Internet) people using virus kits have been dubbed *script kiddies* as a way of mocking them for not being capable of writing original and sophisticated malware on their own.

2.6 Macro Viruses

Macro viruses are usually a case of *mobile code viruses*. This is mainly due to the fact that most modern high level languages run as virtual instructions. Since high level languages are by far preferable to low level ones for scripting/macro purposes, this leads to macro languages being developed with virtual platforms as the target. This has 2 distinct consequences:

1. *Macro/Mobile viruses* can target a plethora of platforms with a single implementation.
2. *Macro/Mobile viruses* are naturally limited due to lack of knowledge about the *actual target platform* and due to the (often) secure nature of a virtual machine platform. Mostly, the security on a virtual platform is achieved by not allowing the user control over memory management *(Java vs. C)* preventing such attacks as *buffer overflow* and the like.

One reason for the vast amount of macro viruses is the popularity of *macro virus hosts* like certain parts of *Microsoft Office*. Since *MS Office* products are installed on a vast amount of computers worldwide, macro viruses can be easily spread through e-mails. Earlier versions of macro capable software allowed macro code to be executed without the users consent immediately after opening a document which of course led to a big increase in macro viruses.

2.7 E-mail Viruses

With the rise of e-mail popularity and the capability to send files over the Internet, e-mail became a great medium to spread viruses. Programs like MS Outlook, which could be accessed from other MS programs like Word opened up for Macrovirus infected files to send out e-mails to everyone in the Outlook address book. The first e-mail viruses relied on trusting users to open attached files in e-mails, allowing the virus to be run. Later the e-mail clients had the capability to run code directly in the e-mail, allowing for viruses to infect just by opening an e-mail. These types of viruses has two stages. First they send themselves to everyone in the users address book, and then they execute the harmfull code.
One of the best known examples of e-mail viruses is the ILOVEYOU virus that infected millions of computers in the year 2000. The subject line of the e-mail recieved was ILOVEYOU, and the attached file was named LOVE-LETTER-FOR-YOU.txt, but this was not really a .txt file, it was a .vbs file, (visual basic scipt) which at that point was a file extension hidden by default in Windows and was run automatically by MS outlook by default. Its payload sent itself to the first 50 adresses in the adressbook, and then it changed imagefiles on the system to be copies of itself, again hiding the file extension .vbs.

Today e-mail clients are very well protected against these types of virus, so they are becoming more and more rare.

3 Virus Countermeasures

Virus countermeasures and virus development is of course a kind of *arms race*. Complete protection against viruses is basically impossible, but the Internet can be used for both the distribution of virus-protection as well as virus-infection. Virus protection is basically the responsibility of dedicated software as well as inherent security measures in the operating system.

3.1 Approaches

There are basically three aspects to consider when dealing with virus protection:

- Detection
- Identification
- Removal

Since it is nearly impossible to prevent viruses from entering the system, the viable approach is to *detect* them before they can cause harm to the system. Once a virus has been detected, it must be *identified* in order to select the correct method of treatment. Once an identification has been made, the virus can be *removed* using the corresponding remedy.

3.2 Antivirus Software

Advances in virus and antivirus software go hand in hand, because the antivirus developers usually have to react to developments in virus software. Anti virus can generally be divided into four generations

- 1st gen: The antivirus program looked for permanent signatures in viruses.
- 2nd gen: Uses a heuristic approach to look for patterns of probable infection. This could be fragments of code usually associated with viruses, for example an encryption loop, to look for the key, and decrypt the virus and identify it. This could also be checksum checking, to make sure that a file is not tampered with. The sum would be hidden somewhere where the virus could not find it, and replicate it.
- 3rd. gen: Looking for and blocking behaviour of viruses instead of actual structure of a program. This allows for just a set of known actions of viruses so new viruses can be found before a signature is known.
- 4th gen: Uses many different kinds of detection, and also access control so that it limits viruses privileges, such that the viruses cannot update its files to further avoid detection.

3.3 Generic Decryption

Generic Decryption is a method which deals with *encryption viruses*. Finding and decrypting viruses is a losing battle, and thus a novel approach is needed. Since encrypted code cannot be executed, one idea is to let the virus decrypt itself, effectively revealing its true body and making it vulnerable to signature scanners and other antivirus software. For this method 3 elements are needed:

- Virtual machine
- Virus signature scanner
- Emulation control module

The approach is then to simply run a program in a protected/sandboxes virtual environment and periodically pause the program for virus detection. At some point, the virus must reveal itself in order to execute the payload at which point it can be effectively detected.

3.4 Digital Immune System

This is how we know antivirus software today. Back in the day, antivirus programs could be updated maybe once a month and be quite well protecting, because of the slow spreading of individual viruses. Now with the Internet and the capability to send files either by users, or by programs themselves viruses can spread incredibly quickly. Now software protecting against viruses need to be updated daily or even several times a day.
The system works like this.

- A new virus appears.
- It is captured and sent to a central machine that creates a sandbox for analysis.
- Finds a way to detect and remove the virus
- Creates a prescription from this
- prescription is then sent to every machine running the antivirus program.

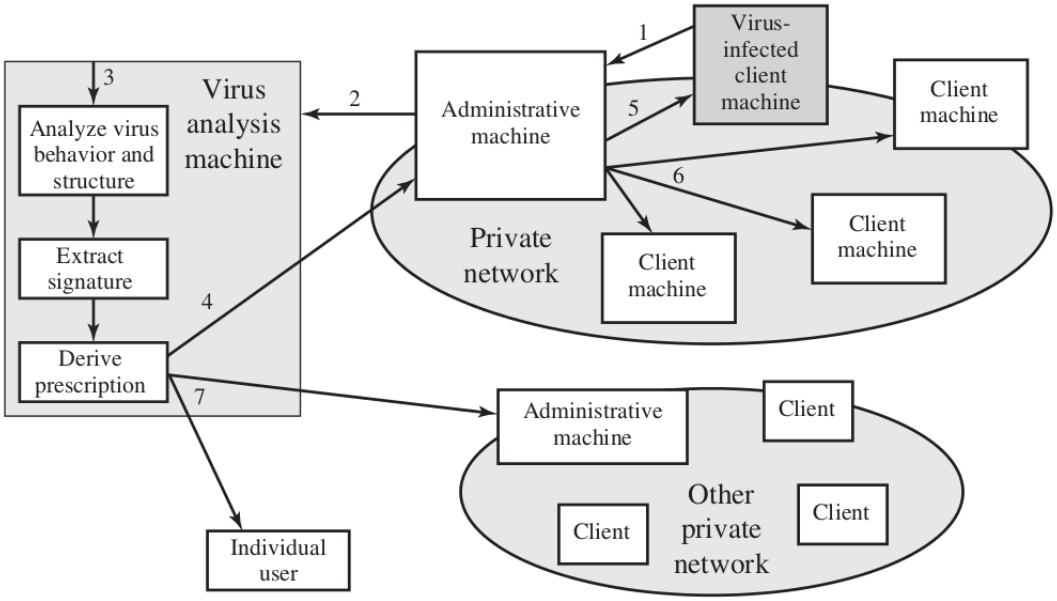


Figure 2: A depiction of a digital immune system

3.5 Behaviour Blocking Software

Behaviour blocking is a more *manual approach* in which a *system administrator* needs to set up rules for correct behaviour. Once a program enters the local system, it is run in a *sandbox environment* for a while before being accepted. In this way suspicious behaviour can be detected before allowing the program to enter the local system fully.

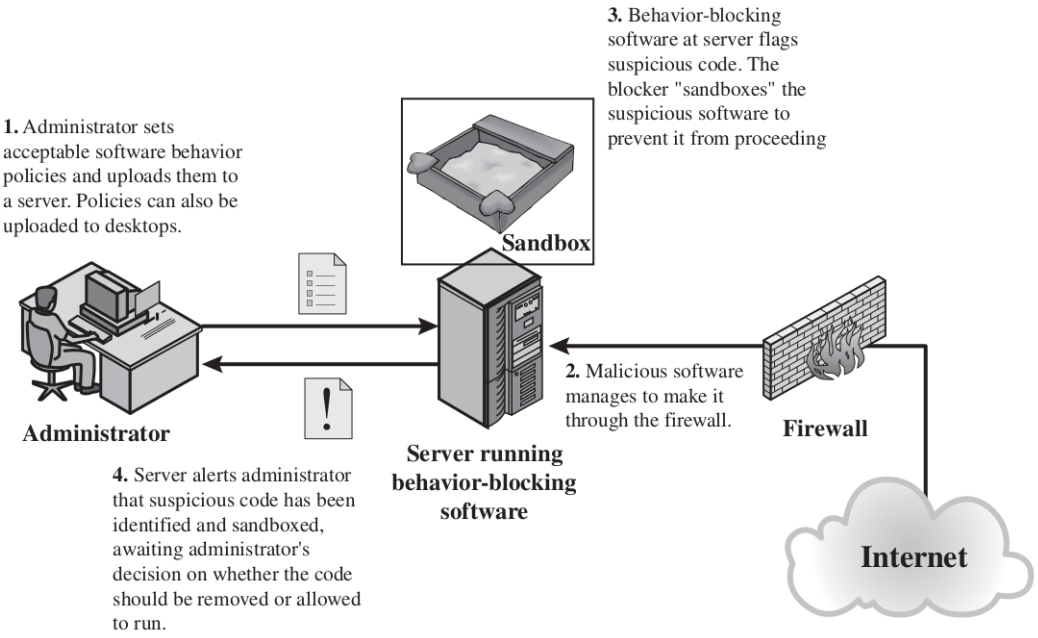


Figure 3: A depiction of a behaviour blocking scheme

This approach has the same problem as *Generic Decryption* in that a virus can choose to wait an arbitrary amount of time *(e.g. a logic bomb)* before revealing its payload. A virus of this type would go undetected by both methods.

4 Conclusion

A big variety of viruses exist in the world today and protecting a computer system against them is a difficult task. The use of dedicated protection software combined with inherently secure operating systems and user care can add a lot to the combined safety of a computer system.