Optimization of Antivirus Software

Catalin BOJA, Bucharest , Romania
Adrian VIŞOIU, Bucharest , Romania

The paper describes the main techniques used in development of computer antivirus software applications. For this particular category of software, are identified and defined optimum criteria that helps determine which solution is better and what are the objectives of the optimization process. From the general viewpoint of software optimization are presented methods and techniques that are applied at code development level. Regarding the particularities of antivirus software, the paper analyzes some of the optimization concepts applied to this category of applications.
***Key words**: optimization, software, antivirus, optimum, criteria.*

Anti-Virus Techniques

The scope of an antivirus software program is to identify and to stop other applications that might in a way or another affect the integrity of user data without its acceptance or agreement. In other words, the antivirus represents one of components of the system defense mechanism It protects system and user data. The techniques that are used to check for software viruses in a software environment define the strategies that the application implement in order to warn the user about malicious activities that takes place on his machine. These techniques include methods for preventing, detecting, and repairing viruses in files or file system areas and are based on:

- scanning; the core routine of the antivirus application scans the file system on regular basis or on user demand and searches inside each file for a virus signature; the signature of a virus represents a sequence of bytes that best describes the virus and its versions; in many cases this string if defined by the machine internal format of assembly instructions found in the virus source code, instructions that are not likely to be located in the clear code of usual applications; for example one of the most successful boot sector virus was named *Stoned* and it was capable to infect boot sector of both floppy disks and hard disks; despite its many versions, it signature was defined by the starting sequence which it uses to infects the floppy in the A: drive:

BE 0400 B801 020E 07BB 0002 33C9 8BD1 419C

this sequence represents the machine internal format for the assembler code:

```
mov SI,4          xor cx,cx
mov ax, 201h      mov dx,cx
push cs           inc cx,
pop es            pushf
mov bx,200h
mov SI,4          xor cx,cx
mov ax, 201h      mov dx,cx
push cs           inc cx,
pop es            pushf
mov bx,200h
```

and is part of the virus routine that it uses to write itself to cylinder 0, header 0, sector 1 on the floppy disk; based on the scanning technique the antivirus will do a string search in target files, mainly executables; if it finds the sequence or part of it, it will announce the user or it will try disinfect the file; despite its history and its known drawbacks regarding polymorphic and stealth viruses, this method has an important advantage over other approaches in that it catch the virus before it is executed;

- behavior checking; the antivirus software contains a monitor component which is a routine that is executed in the background by the system; this routine is a *terminate and stay resident* program which hooks important BIOS interrupts, like 13h or 21h; every time an application calls these interrupts, the anti-

virus routine checks the function parameters and it request user to accept or to deny the execution;

- integrity checking; this method is based on the fact that an infected file looks different than the original one; not taking into consideration the name of the file, as in case of companion viruses, the file data represents the main objective; as results, the infected file will have a different size, date/time modified stamp or it will generate a different checksum, CRC; once it is installed the antivirus will generate and it will manage a database that records all these information about files; the scan routine compares, for target files, the recorded information with the actual data and it alerts users about situations in which files have been modified since last check; because most of the executable files does not change at runtime their content, this represents a valid approach.

Despite the evolution of virus types and the methods used to deliver them, the antivirus software contains a mixture of techniques and particular methods in order to maximize their efficiency and to minimize resources and their impact.

2. Optimum criteria

The software application used as defense against computer viruses is defined in the category of daily used software as email clients, messaging and Internet browsers. Based on this fact, this software must provide a high security level and in the same time it must have a low profile. The degree in which it affects user actions must be at the lowest level because if it generates discomfort to users by affecting other applications response time, it might get uninstalled or turn off.

In order to develop an antivirus application it is important to define the characteristics considered important by producers and users. By setting target levels for these criteria, there are described the premises for a more clear development and qualitative analysis processes.

Space minimization refers to the dimension of the memory zone that the application requires on the hard disk or in RAM. Every antivirus application uses for the scanning routine a database of strings or virus signatures in order to correctly identify malicious code. Because the strings are small sized, as in most cases a 16 or 32 byte sequence is enough to conduct a search, the database size depends mainly on the number of known viruses. If it has an integrity checking routine, then the application must generate a log database in which it records the parameters of all the files in the protected zone. For a large file system this may requires more space on the hard disk. If it is taken into consideration an activity monitor, then the objective is defined by the amount of virtual memory it requires to be active in the system and to scan real time network traffic or accessed files.

Speed maximization is a criterion that is directly correlated with the effort, measured in time, of the antivirus software to scan memory or a drive for infected files. This characteristic is important for scanners and integrity checkers because these components must analyze each file in the protected zone. The effort consists in string search inside file data or in computing different types of checksums for the file content.

Low profile describes the impact of the antivirus application on user activities. During the scanning or the monitor process, the user must be affected in a limited way that will allow continuing his work without any discomfort. The antivirus activities must have a low profile, making them unobservable by users. The worst case scenario is the situation in which the system crashes, is not responding any more to command, or current applications loose data.

Minimize used resources refers to the maximum level of needed resources. In order to be an efficient application, the antivirus must run on different hardware and software architectures with minimum requirements. Also the level of needed resources affects the characteristics measured levels during the execution. Being part of a required system configuration next to operating system and a office suite, the antivirus software must use in an efficient manner existing resources without imposing a minimum configuration that will limit its implementation.

3. Software optimization

The evolution of software and hardware technologies permits for complex software applications in the present, but also with great requirements for processing speed and memory usage. Software applications included in this category are operating systems, entertainment applications and multimedia applications.

With all this waste of resources, transparent to the user, the developer gives particular importance to the optimization process, looking to maximize the performance of the final product.

Another category of software products is constrained from the start to be efficient with respect to the system resources used. In this category are included antivirus applications, drivers, viruses, applications implemented in microcontrollers or smartcards, function libraries, applications for mobile devices.

The objective of software optimization is to obtain a new product or a new version of an existing product, which presents a higher quality level. This grade is worked out based on the levels obtained from the set of software characteristics or the established optimum criteria. By direct comparison to the base levels or by determining aggregated values based on the way multi-criteria models are composed, the level of improvement is obtained.

The optimization process implements techniques and methods used in:

- *problem analysis*; this implies that a lot of the problems in software optimization are generated in stages before the development of the source code; the implementation of an inefficient analysis leads to defining a solution that isn't characterized by a required quality level;
- *source code*; if this is based on a bad implementation of an algorithm, it will lead to obtaining inefficient results in most situations, even if the complexity of the source code may be reduced, or if it is of high quality; the main cause of problems at source code level lies in a low level of its experience, and last but not least in the mistakes it makes; the primary methods, [10], used at this level are based on the elimination of repeating sub-expressions, instructions without any meaning, sequences in which instructions with opposite effect appear, invariations, by substituting complex reference expressions with simpler ones and by regrouping control structures;
- *compiler*; this component is responsible for transforming source code in the form associated with the high level language into machine code; as this form is directly processed by the microprocessor, it greatly influences the way in which resources are used and especially the total processing time; using a good complier that contains a lot of techniques for optimizing memory usage and processing speed leads to optimizing the application without the need for any other additional effort; the second solution for getting optimized machine code is to write the source code in assembler languages; analyzing the efficiency of using an optimization routine for the compiler or writing the application in assembler languages leads to defining two approaches; in the case of routines with a high level of importance for application performance, the programmer can generate , in particular situations, machine code more efficiently than the compiler; despite this, the solution of developing the whole application in machine code is not viable because the effort and time resources are too great;

4. Specific optimization methods

This category of software product has unique characteristics that required a particular approach when defining optimization methods that will increase the quality level.

In order to reduce the influence on other applications or on users' sessions, the antivirus software must be activated when the system is in idle state or when the level of available resources is enough to scan the system without affecting the conditions in which users interact with others applications. If the main objective is to scan the system and this task does not have a time limit, the antivirus application must schedule its scanning and integrity checking routines depending on the user current activities.

Between the components of an antivirus software application, the routine that scans the file system for infected files continues to be reliable and to represent the major component. It also has the greatest impact on the application behavior and consumed resources because of the effort to search string sequences in a large number of files. From this point of view, particular optimization methods target to define a faster and a more reliable string searching mechanism:

- *wildcards* allows scanners to ignore some characters or sequences from the target string; this will speed up the search operation because it is not required a one to one identification;
- *generic degree* is strictly related to different versions of a virus; the developer identifies a signature that is common to all the versions of a virus; in this way , the generic signature helps reduce the number of search strings;
- *mismatches* is a method introduced first in the IBM antivirus and it allows a finite number of bytes in the string to be any value, regardless of their position;
- *Top-and-Tail scanning*; because most of the viruses place their code at the end or the beginning of the infected file, the scanners will search only in this zones; this will reduce the scanning effort but may affect the precision;
- *Entry-Point and Fixed-Point scanning* defines a method in which the search begins at a certain point in the file, the entry point; in order to manage the execution of a file, the virus must take control from the start and to pass it to the infected file after it terminates its routine; this is achieved by redefining the entry point of the executable by placing viral code at the beginning of the file or by placing a jump instruction to the virus sequence; based on that, the antivirus will begin the search from the entry point of the file by following jump or call instructions;
- *hashing* is used as a data storage structure in order to have fast access to elements based on nonnumeric or large value keys; for antivirus scanners is used to reduce the number of strings that are searched inside the file; the methods uses 16-bit or 32-bit words of the scan strings to generate a hash value that have the role of the index in the hash table;
- *smart scanning* represents a countermeasure to viruses that try to hide their code inside sequences of no value instructions such as do-nothing NOP instructions;
- *skeleton detection* is a method defined by Eugene Kaspersky and it reduce the searching zone inside a target file; the idea is to eliminate all the instructions that are not likely to be part of the virus code before starting the scanning routine.

5. Conclusions

In today information society, the information represents the most important good. With the growing number of data communication services, channels and available software applications, data are processed in large quantities in a more efficient manner. This will also increase the vulnerable spots of the computer system and in order to protect data, an antivirus system becomes a must have integrated component of the system. That is not enough, because to provide valid protection against known and future viruses, the application must be continuous updated with better versions of the database but also of the scanning and monitoring routines.

References

[BOJA05] *Cătălin BOJA* – Software Multicriterial Optimization*, The Proceedings of the Seventh International Conference of Informatics in Economy, May 2005, Academy of Economic Studies, Bucharest, Romania, Inforec Printing House, pp. 1068 – 1074, ISBN 973-8360-04-8.*

[IBM00] IBM Corporation – Antivirus Research Project http://www.research.ibm.com/antivirus/index.htm*, 2000.*

[IVAN05a] *Ion IVAN, Cătălin BOJA* – Empirical Software Optimization*, Revista Informatică Economică, vol. IX, nr. 2/2005, Editura Inforec, Bucureşti, 2005, ISSN 1453 – 1305, pp 43 – 50.*

[IVAN07] Ion IVAN, Cătălin BOJA – Practica optimizării aplicaţiilor informatice, Editura ASE, 2007, în curs de apariţie.

[LUDW95] Mark LUDWIG – The giant black book of computer viruses, American Eagle Publications INC, Arizona, USA, 1995.

[SZOR05] Peter Szor – The art of computer virus research and defense, Addison Wesley Professional, ISBN 0-321-30454-3, 2005.