# A Guideline to Anti-Malware-Software testing

*Trend Micro Student Runner Up Award*

*Andreas Marx*[1]
*Student of Business Information Systems at the Otto-von-Guericke-University Magdeburg*

## About the Author

*The author was born 1979 in Magdeburg, Germany. In 1991 he became interested in computer viruses when he got the Yankee_Doodle virus and he started to write his first anti-virus program called CGL. It was based on integrity checking and used heuristic rules against boot and file infectors. In 1996 he won two first prizes and the 5th prize overall at the "Jugend forscht" competition. After this accomplishment, he started to write anti-virus-tests for the "Anti-Virus-News" magazine. Since 1997, he has written virus-based articles and tests for the German "CHIP" computer magazine and has performed large anti-virus comparison tests in 1998 and 1999. He also performed tests for the "WIN" and the "PConline" magazines in 1999. Since October 1999, he has been working for the "PC Welt" magazine, too. After finishing his military service, he started to study at the Otto-von-Guericke-University Magdeburg.*

*Mailing address: Andreas Marx, Foerderstedter Strasse 11, D-39112 Magdeburg, Germany, Phone: +49(0)391 613303; Fax: +49(0)391 6218501; E-Mail: amarx@gegasoft.de*

## Descriptors

*Anti-virus testing, malware testing, virus detection, computer security, malicious software, program testing, black box tests, software evaluation, sorting virus collections, publishing articles*

**Reference** to this paper should be made as follows: Marx, A. (2000) 'A Guideline to Anti-Malware-Software testing', EICAR 2000 Best Paper Proceedings, pp.218-253.

[1] I would like to thank Prof. Dr. Claus Rautenstrauch and all the people who helped me to write and improve this paper.

EICAR Best Paper Proceedings
Edited by Urs E. Gatikker, © EICAR c/o TIM World ApS, Aalborg, Denmark, ISBN: 87-987271-1-7
www.papers.weburb.dk

# A Guideline to Anti-Malware-Software Testing

## Abstract

*Today, most of the popular anti-virus test strategies are obsolete. Index numbers like "best detection score" are unsatisfying, because a program can only find viruses it searches for and the number may be a little bit accelerated by heuristic rules to find new and unknown viruses. This way of testing is possibly interesting for anti-virus researchers, but not the real world, because the customers do not know what to do with all the numbers in the tables.*

*Therefore a closer look at the real needs is required. This includes on-demand, on-access, disinfection and false positive testing of the products. The goal for this type of testing is not a mass test, but a test which is as exact as possible. Of course, a test of anti-virus products should include stability tests, tests of user-friendliness, security, efficiency, compatibility, documentation, support, installation and many other things, too. After a test, usually a review have to be written and the test is going to be published in a magazine, however, there are some problems which can occur and they will be described in this paper, too.*

*It discusses the need for current and future enhancement in anti-virus-testing techniques for IBM-compatible computers, including the foundations and limitations on such testing.*

## Introduction

This paper describes a methodical framework for anti-virus program testing for small and home office use. It is written for data security managers and for professional testers who write for magazines. However, all practical considerations are limited to anti-virus programs running under Windows '9x, due to the large amount of users having installed this system, but most features can be tested under Windows NT with only minor adjustments. Aspects besides testing, like management, education and policies can be found in (Wack & Carnahan 1989). Larger overviews over this topic can be found in (Polk & Bassham 1992). This paper shows often found errors and how to avoid them. There are several certification schemes (Gordon & Ford 1996), like ITSEC, but this paper has been written for an other target group.

Before going into detail on the testing methods, two major ways to protect the user from being infected by computer viruses will be introduced. They can be classified in software- and hardware solutions and follow different philosophies and realizations.

### Software Protection Mechanisms

Software protections against viruses are usually the well-known *anti-virus programs* that can be exerted under different operating systems, like Windows, DOS, Netware, OS/2, Macintosh, Linux or professional Unix systems. Together with special application tools like Lotus Notes or Microsoft Exchange, they are used to keep the system virus-free. An anti-virus program is employed scanning for viruses in files at real-time or periodically. If viruses are found, the files can be disinfected or deleted. Furthermore, information about the virus problem and the infected files can be given to the user for further decisions. Some programs use content filtering to be able to delete spam, junk mails and hoaxes automatically and some use unusual, non-anti-virus related features, like the detection of non-Y2K compliant macros or Excel sheets.

In large companies, the information technology (IT) infrastructure often is very complex using different operating systems, networks, and hardware platforms. However, virus protection in such heterogeneous environments will not be discussed here, since the particular challenges in such environments would fill another paper.

### Hardware Protection Mechanisms

Hardware protection mechanisms like "*Trend Chip Away Virus*" (Jang 1998) or the "*Virus Warning*" feature - both can be found in newer BIOS versions - will not be discussed here in detail because only newer computers (around 1997 for the first and around 1994 for the second solution) have this feature and they are disabled by default. Older computers - systems that very likely only work under DOS - do not have such methods to protect the system against boot viruses.

Some viruses fool this form of protection, because it is easy to deactivate the hurdle before infection and reactivate it afterwards. The reason for this is that it is a simple and unprotected implementation - the configuration data is saved in the CMOS area where everyone has access. Both hardware protection solutions have limited use, often trigger unexpected problems, and only work with boot sector viruses. If a tester replicates viruses or wants to test the detection of boot viruses by using a scanner, those options have to be disabled to get correct results.

## Part I - Prerequisites for Evaluation and Testing

In this chapter, the preconditions for anti-virus testing will be discussed. Though some of the preconditions may seem trivial, they are essential and often overlooked.

### Four Important Points of Interest to Testers

First, the tester must have knowledge about the *theoretical foundations* of viruses and anti-virus testing. The tester must be clear on what to do with viruses, how they work and what kind of damage they are able to cause. The tester should carefully work with the viruses in an isolated environment and the tester should know how to review anti-virus products, which criteria are important, and which should not be tested because they are irrelevant.

The second point is, that the tester has to be *independent* from every anti-virus company and such a company must not sponsor the test. Obviously, even if testers do not cheat, the sponsor's criteria often include items that the sponsor knows their product will meet and "by accident" the sponsor's product may be the one that is evaluated as best. An example of this affect can be seen in (ZDTag 1999). Another problem of sponsoring occurs if the tester exclusively uses the virus database of one company for the test. This is not objective, because the tester only receives viruses the particular sponsor's program is able to find and repair, and not necessarily the viruses that cause the product difficulty.

Third, the tester requires a *project plan* that depicts what to do and in what order. There should be enough time for all of the things to do, like preparing the test, sorting the virus collection, setting up the computers, getting the products to test, evaluating and testing all programs, writing an article or reporting about it and discussing it with the publishers to make a good story. Usually, there are some problems with parts of the tests that require some extra time. For example, if a product is very slow or crashes on huge virus collections, the testing will have to be restarted often. For this, some buffer time should be taken into account.

Fourth, the *acquisition of resources* should be made according to the budget. For example, if one or more supporters and several PCs are needed to test the programs parallelly, those systems must be acquired before testing can be started. Usually, the virus collection used for the test has to be write-protected on a server. The clients scan it and write the report file to a local disc. The server has to be administrated and the clients have to be installed correctly. Of course, the test network has to be completely (hardware-) isolated from the rest of the other computers in the tester's organization. Enough disk space for the programs, the virus test set (especially the large polymorphic test sets) and the report files have to be available, too.

## How to Get Products for Testing?

Obviously, testers have to use current versions of anti-virus programs from the anti-virus companies. Therefore, the companies have to be informed about the deadlines of the test. This includes the feedback deadlines such as responding on whether they want to take part in the test or not (not answering means not), completing a survey with some facts about the products (available OSes or where to buy it) and sending the products to the reviewer including all updates which should be used.

It should be self-evident that if a company does not want to see their program tested, it will not be tested. The reasons for declining to take part could include a desire to wait for release of new versions or major updates soon, or a concern for performance. If they do not want to take part in the test, because they know that their products are not very good, the tester has offer two options. The first is, the tester will still test the program, but he has to buy it in a shop and get the updates via internet or snail mail using the normal update scheme. The second possibility is that the tester will make a notice about this fact in the review only and the program will not be tested at all. It is worth noting that when a company does not take part in any test and they may be forgotten, because users usually buy only programs where they know what they will get for the money.

If there are too many companies who plan to release new major versions in near future, it may be that the test should be postponed, because it will be outdated when published. If an anti-virus company has only sent a beta version of the program, it should be tested separately and without a final conclusion.

An invitation to the test has to be sent out to the companies usually one month before the test starts to offer them enough time to decide whether they want to take part in the test or not, and which versions of their programs should be submitted. Addresses of developers can be found in (Link 1999).

The invitation which can be sent via (e-)mail or fax to the anti-virus company should include at least the following topics:

- The *objective of the test*, i.e., for which magazine (and issue, if known) or which company is the test? This helps the anti-virus company to decide if the test is interesting or not, and if it is the right target group.

- Under *which operating systems* is the test is going to be performed (exact version)

- The *subject of the test*, like finding the most zoo viruses only or the more important ITW ("in-the wild", viruses which are known to be common) ones. Maybe tests about disinfection or the virus guard special functions for small or medium companies like fast update distribution and things like user-friendliness or the functionality should be mentioned here.

- It should be asked, *which options/settings* are recommended for the GUI (or command-line) version of the program in all tests - to scan for viruses and false positives, too. The parameters have to be clearly documented and it should not be a paranoid long list.

Some programs can be switched to (quick and dirty) modes where they can find many more viruses, but this increases the change of false positives dramatically and the scan speed grows by a factor of five. Another solution would be to use default settings only without any changes on the configuration (with one exclusion: the “scan all files” option) - a common scenario.

- The name, e-mail, telephone and/or fax of a *contact person* who can support the tester if there are some questions and further expertise is required.

- The *deadlines* mentioned above for the test that have to be the same for every program. It should be noted clearly that software can only be included into the test, if the deadline for sending the programs and all updates is met.

- What has to be *delivered*? It could be a program CD only including the software and online documentation or the complete package including a printed manual etc. If the test includes comparison of different tools on the CD, the user’s manual, rescue discs etc., the anti-virus companies have to send the complete package as it could be bought in a computer store.

- What about *updates*? Anti-virus programs become obsolete very fast, and therefore, all updates the tester should use have to be sent together with the program. It should be made clear that the responsibility for offered updates and releases is assigned to the anti-virus companies only. It is important that all programs have nearly the same date where they were last updated.

- A *snail-mail address* to which the products should be sent has to be included in this information as well as the e-mail address, telephone and fax of the tester for questions about this procedure.

It is up to the tester if software submissions or updates received via e-mail will be accepted. Usually, the tester will receive several ten megabyte files of data in a very short period of time. It could be a big problem if the mailbox does not have the capacity for this huge amount of data or if the download of the files requires some hours. The tester can also download all the files needed from the Internet, but it could be that this is not the most current version or only a limited shareware or evaluation version.

Normally, companies are open minded about such tests. Unfortunately, sometimes there are companies who “forget” to answer and should be reminded at least two times before they will be removed from the test. There is often a little communication problem between the tester and the tested company. Replies are not answered or other feedback is missing.

About one week before the final deadline, all companies should be reminded about the test. The people working there are often extremely busy and some things might go wrong. After receiving the products, it should be checked to verify that all companies sent the right versions for the test and that the package is complete, including all updates.

## Preparing the Virus Collection

A malware collection should contain viruses, worms, Trojan horses and other malicious code. The objective of sorting and structuring the virus collection is to find the “minimal” set of all relevant viruses which is applicable to meet the requirements of the test. There are large numbers of viruses that exist now, however, most of them are simply unimportant, like more than 14,000 viruses, which were generated using a virus construction kit (Ducklin 1999b). Definitions of it can be found in (Marx & Michl 1999; Marx & Günther 1999; Marx 1999b; Luckhard & Siering 1999) and future aspects of malware are discussed in (Brunnstein & Schmall 1999). Malware is defined here as software which is made intentionally for sabotage of operational software environments. Therefore, jokes are not included in this definition. All sorts of malware have to be tested, because we want to test anti-malware programs and not only anti-virus ones. This has been shown during outbreaks of worms like Melissa or Explore_Zip. Trojan horses like Backdoors can be found more often, too. However, malware, like worms, are not viruses at all but most vendors call their products anti-virus programs and not anti-malware programs. For more aspects on the identification problems see (Whalley 1999).

This section gives a brief overview on the complex and difficult task of preparing good samples for testing and the (still unsolved) problems for sorting and structuring a virus test set. A checklist for non-macro and non-script viruses can be found in (Bontchev 1993).

### Getting “something”, maybe viruses are included.

There are different ways of getting and making a collection to test anti-virus programs. The first idea - which will hopefully never be seen anymore - is to employ a *Virus Simulator* like VS^2 by Boff Consulting or VirSim by Rosenthal Engineering and generate “test viruses”. These “viruses” only contain a simple routine to print out a message or copyright notice (host) and exit to the OS. Attached to these easy programs are some strings from known viruses like Jerusalem or Vienna that can be found near the beginning of the viruses, but they will never be executed in these “test viruses”. Of course, it makes no sense to test the detection of these files, because they are neither viruses, nor malware, nor contain other malicious code and their design patterns are known and very simple, because the host does not change in most cases. In other words, they do not show, if a program is able to detect real viruses. These and other aspects are discussed in (Gordon 1995).

A second idea to discuss - and this will hopefully never be seen anymore, too - would be the *creation of new viruses* using a virus construction kit, for example VCL, PS-MPC, IVP, or to write new viruses or create new variants. It is a bad idea to use these kits to generate new viruses in order to test anti-virus programs, because most of the products know the design patterns and can find all variants easily. Second, still enough viruses exists and there is no need to create more of them.

Third, a tester could ask the *anti-virus companies* to supply viruses. This will fail in most cases, because they will not give samples to unauthorized persons. However, in some cases, for example, if the tester has a high reputation in the anti-virus community, the tester may be able to get parts of the required collections or at least some samples still missing. Of course, to be fair against all producers of such security programs, the tester should only use parts of every collection (the more, the better, because of the mixture of the samples) received, but should include only replicated samples where possible. The reason is that anti-virus companies often only give samples of which they know that their program will find. Sometimes viruses are added "hard" to the database using a CRC sum over the full length of the file and because of this only this file will be found by the scanner, but not replicated samples. Examples for this are viruses like Zhengxi or SSR that are very hard to detect.

As mentioned above more than one or two collections should be used, because the "sponsors" of such collections will usually win the competitions with the best detection score. Normally, such collections are sorted quite well (but always using different schemes and different philosophies) and are usually free from non-malware, intended viruses, and innocent programs. However, it will still contain some files (up to 1 or 2%), which do not belong in such collections, even if all anti-virus programs are able to identify these files. The reason these files make it into the collection is that the companies receive them often enough (because they can be found in every badly sorted virus collection or most other programs will find them), so they simply added detection for these files to satisfy such people and keep them quiet. Another aspect is that the file contains a damaged virus version, but most scanners will still find the virus or a part of it, because of an inexact identification.

An often-used method for testers to get viruses is to *download them* from the Internet, other network systems (like Fido) or from special virus exchange (VX) bulletin board systems (BBSes) or FTP sites. Often, very complete-looking collections can be found there with many unknown viruses and ones that are not detectable by most recent scanners. However, the reason for this non-detection is that most collections contain a lot of non-viral programs (trash). "They contain huge numbers of viruses, non-viruses, Trojan horses, joke programs, intended viruses (programs written with the obvious intent to write a virus, but too buggy to replicate even once), corrupted files, text files, virus creation tools, completely innocent files, and so on" (Bontchev 1993, Introduction). Therefore, if a tester wants to get samples for a virus test set here, viruses should be separated very carefully from the trash (usually more than 10%).

For this task, a *file weeder utility* can be extremely useful, which includes the information about all known non-virus programs found in such collections. An example of such a tool is DustBin written by Network Associates (formerly Dr. Solomon's). Ideally all anti-virus developers should be contributing to such a tool, so that cheating or mistakes caused by inexact virus analyses are nearly impossible. Such weeders should include all TXT, ASM, files with advertisements for BBS'es (BBS addy's), not working viruses in DOC, XLS, EXE, COM and SYS files, etc. This would solve a hard problem for testers who have to decide which files to add or leave away. It goes without saying that such utilities have to be updated on a regular basis (like anti-virus programs). To be as exact as possible and avoiding false positives, it should not use old CRC32 checksums, but more professional MD5 hashes and some more information on the files, like the type or length of segments in the file.

There is still another possibility: The tester does not evaluate the detection rates of the scanners at all using big collections, but using only good-sorted ITW ones. For the large database tests, the tester can use *tests from some valued outside organization*, like some reputable, independent testing centers. However, most of the time, they will use other versions of the program, usually because not all the programs the tester needed are included or the tests are too old. For this reason, the tester could send the complete packages for testing against all or most virus-related categories as desired, even if it will costs a lot of money. The other categories to be tested can be done without using viruses but rather the EICAR test file, which is not a virus, but a harmless program that displays that it is a test file only. This file and additional information can be found in (Ducklin 1995). By now, every scanner will find it. However, such a test file only exists in an DOS/File (COM) virus form at the moment, not in macro virus environments (Abrams 1999), so the testing possibilities are limited.

### Unpacking and pre-sorting the malicious code

The final collection has to be weedered using programs like TBWeeder by Frans Veldman, Duplicate File Locator by William Ataras, Rose's File Weeder (RFW) by Ralph Roth, Linux File Weeder by Gerald Scheidl or DBScheck by Norbert Huth. In this process, duplicates will be removed. Unfortunately, some programs like TBWeeder can only handle a limited number of files and RFW gets very slow with thousands of files in the database. Archives (like ZIP, ARJ, LZH, RAR or ACE) should be completely unpacked before testing. It is also a good idea to keep pre-sorted databases in their original state and not to t copy all files into one directory, because this accelerates the sorting process.

Now the collection should be grouped into different categories like macro viruses, boot images, file viruses and non-viruses. Non-viruses should be kept on the disk in a special directory so that next time they could be deleted automatically using the file weeder to avoid additional work in the future.

Table 1 shows the most important facts on the files types and their recognition. However, there are quite a low number of viruses that use the same headers or extensions to confuse the scanners. A question mark in the detection column represents an undefined letter. For script-based viruses, it is the same if the letters are upper- or lowercased.

It should be noted that some programs are still unable to report O97M viruses correctly; therefore they will use the prefix of the actual infection only (like W97M or X97M). Other programs might use slightly modified conventions like WM97 or XM97 (Sophos 1999) or simply a different one.

This is only an overview of the most important file formats, see also (Scheidl 1999) for a list of more file formats that could be infected and for the suggested naming conventions for these types of malicious code.

| Type | Detection | Comment |
|---|---|---|
| Macro viruses (OLE/COM file format) | Extensions: usually DO?, XL?, PPT, RTF (sometimes) or none; Header: D0 CF 11 E0 A1 B1 1A E1 (hex) | used for WM, W97M, W2000M (Word); XM, X97M, XF, X97F, X2000M, (Excel); PPT, PP97M (Powerpoint); O97M, O2000M (Office), P98M (Project) and others |
| Macro viruses (MS JET-DB fomat) | Extensions: MD?; Header: 01 00 00 00 (hex) or: 00 01 00 00 (hex) and "Standard Jet DB" | used for A2M, AM, A97M (Access) |
| Macro viruses (old) | Extensions: DO?; Header: DB A5 (hex) | used for W2M (Word 2.x) |
| File viruses and malware (DOS, Windows 16- and 32 bit, OS/2) | Extensions: COM (no special header, but often E9 or EB at the beginning), EXE, DLL or VXD ("MZ" or "ZM"), SYS (mostly FF FF FF FF (hex)) | first or last letter of the file extension often changed for security reasons to get non-executable extensions |
| Linux file viruses and malware | Extensions: LNX or none, Header: 7F 45 4C 46 (hex) | ELF binary file format |
| Java viruses, malware and jokes | Extensions: CLA or Class; Header: CA FE BA BE (hex) | OS independent file viruses; mostly jokes, not viruses |
| Boot images (from boot viruses) | Extensions: IMG, BOO or MBR (no Header, often EB, sometimes EA or E9); size often multiply of 512 bytes | no exact recognition possible (for anti-virus programs, too!) |
| Script-based viruses and malware (BAT) | Extensions: BAT; Header: none, but commands like "echo" can be found | some viruses uses the DEBUG program to create files: hexadecimal scripts can be found very often |
| Script-based viruses and worms (VBS / JS alone or in HTML files) | Extensions: JS or VBS (no header, but the instruction "CreateObject" can be found); embedded in HTML files: HTM, HTML or SHTML (Header: "<html>" at the beginning) | in HTML files the script parts start with "<Script Language="Name">", where Name is "VBS", "VBScript" or "JavaScript" |
| mIRC worms | Extensions: INI; Header: "[script]", usually commands like "dcc send $nick script.ini" can be found | usual filename is SCRIPT.INI, but often renamed (duplicate names) |
| Text files and others (non-viral) | Extensions: TXT, ME, DOC, ASM, A86, DEB, MAC, BAS… (no special header), but mostly contains only characters in the range [10,13,20...7f] (hex) | files with descriptions, source code, site addy's etc. |
| Graphic files (non-viral) | Extensions: BMP (Header: "BM"), GIF (Header: "GIF8"), JPG, WMF etc. | screenshots from the virus' payload etc. |
| Sound files (non-viral) | Extensions: WAV (Header: "RIFF"), MID, MOD (no special header), MP3 (FF FB (hex)) | Intros, sound demonstrations or things for the virus payload |

Table 1: Short overview over most common file formats and their identification.

### Sorting and cleaning the whole collection

The files should be renamed to executable extensions or the extensions that these kinds of viruses should have. If this is too dangerous for the tester (one execution of a virus could destroy the work of the complete sorting process), all files that could be executed by accident should be renamed to unusual extensions. The anti-virus programs used to scan and sort the viruses have to be re-configured so that they are able to search for these kinds of viruses, too. Usually, an option like "scan all files" does this work better than adding all of the new extensions used to the program.

Now it is time to sort the pre-grouped files. The collection should be kept in different major directories. The root directories could be named after the type of the viruses, like "Boot", "File", "Macro" or "Script". If malware other than viruses and worms are tested, a directory like "Other Malware" or simply "Malware" should be added and things like Trojan horses and other things should be put here. For polymorphic samples of macro or file viruses, a special root directory like "Poly" is recommended. Now the files could be sorted to the different directories, using the following scheme.

It is hard to identify and sort out boot images and copy them into special directories. It is very important to keep in mind that some anti-virus programs are not designed to find boot viruses in files, and should not be penalized for not finding boot viruses in files, because boot viruses cannot be found in files and multipartite viruses use different parts for files and boot sectors. Often special parameters exist to enable this feature and/or the files must have a special extension (like BOO or IMG). It should be noted, that many of the image files only contain one part of the virus, but the rest (that is usually on an other track or on some special tracks of the disk) are not included here, so the viruses will not run if the images are written back to disk. Therefore, it could be impossible to infect a machine to create samples on different disc. The replication of boot viruses will not be discussed in this paper.

There is only one way to scan for image files and sort them by using the report files from the anti-virus programs or from individual analyses. Surely, some scanners should be used that are known to identify viruses. However, more than one or two scanners are recommended to ensure that only viruses are added. There is usually the problem that three scanners have four different names for the files infected by the same virus. If a virus is only found by one scanner, it could be a new variant, a damaged sample, or maybe a false positive. Such file needs further analysis and a review to determine if they replicate or not. One scanner (e.g. the one that has the most different virus names in the virus list) should be chosen to order the collection by the names the program gives as outputs. This helps to keep one directory for the same virus variant and avoids having the same virus three times with different names in different directories. If this special scanner is not sure or does not find a virus, the second should be chosen and these samples should be sorted in a special directory (in the same database part) so that the collection is still sorted after the first scanner and there a sub-collection that is sorted after the second one. The same can be done with the third, fourth, etc., scanner.

There are also automatic tools to sort viruses this way, for example Virus Collector's Database by David Smith, VirSort by Christian Julius or ZooSort. Gerald Scheidl from

Ikarus Software in Austria developed a sorter tool called Virus Collection Tool (VCT) that uses more scanners to classify and sort viruses automatically. For this, it looks at the detection of most scanners outputs and sorts it to this directory (case-sensitive).

By sorting a collection manually, some things could be done better. For example, if there is a pre-sorted directory with 100 samples of one virus, but most scanners only detect 98 correctly, the other ones report different or unknown variants of it. This could mean that the samples are damaged or somewhat destroyed. Furthermore, it could be the case that it is really the same working variant, but the anti-virus program developers have overlooked a byte that is not static but rather variable and they identify the virus as a new variant. Such things can only be noticed by sorting the files manually. But all-in-all, it is impossible to make a nearly perfect collection, because there are too many variables and anti-virus research is a soft science.

Another point is that there are varying opinions by different researchers to define when a virus is a variant or not. Some say, one bit changed in an unimportant routine is enough; others say that they are different only if there are differences in the run-time behavior like changed infection or trigger conditions. So it is not a good idea to test the programs if they can identify all viruses like the tester wishes. This means, in every separated directory the anti-virus program should find one and the same virus and this virus should be identified nowhere else. It is more important that the program is able to find the viruses and clean them the correct way than if the programs make distinctions in the identification, because of speed reasons or because there is no need to be so exact. However, the results of such a test only show the differences between the tested products against the reference products that have been used to sort and build-up the collection.

To continue the sorting process, a closer look at *macro viruses* will be taken. Products that can identify macro viruses on an exact basis should be chosen here. There are the ones with the best variant detection (the tester has to check this) and mostly they are using the CARO naming convention for this (see Table 1).

These prefixes should be used as directory names to sort the collection. For example, the virus WM/Concept.A should be put in the directory “Macro/WM/Concept/A” because it is a Word ’95 macro virus (see table 2). A little problem could by caused by the suffixes that are used for different language versions of Word if the viruses can only work on these special version. An example is WM/Macaroni.A:De - the problem is that a “:” cannot be used in a directory name, but it can be substituted by an “_” or an other character, so the directory name could be “Macro/WM/Macaroni/A_De”. If there are more than 26 known variants of a virus, a second letter have to be used for this virus variant (or more, if there are much more variants). An example is WM/CAP.GD which is the 26 * ”G” (=7th character of the alphabet) + ”D” (=4th character of the alphabet) = 186th variant of the CAP macro virus.

If there are new or modified variants of macro viruses in rather old files found by recent versions of the anti-virus program, it often means that the files do not contain a working virus or that the files could be damaged. A further look should be made of these files, like trying to replicate the virus. Some programs are able to find such trash files (like a bad cleaned, no longer infected document) and report them correctly so that they can be copied to the non-malware directory.

*File viruses* are usually the largest group of viruses, even if they are not found very often in the wild compared with macro and boot viruses (Wildlist 1999). Sorting them requires the most effort compared to other types of viruses. File viruses can be grouped into DOS (without any prefix), Windows 16-bit (W16 or W31 prefix), Windows 32-bit (W95, W98, W9x, WNT or W32 prefix), OS/2 (OS2 or no prefix), Linux (Linux, Lnx or Lx prefix) and Java viruses (Java or Jv prefix). The prefixes could also be "Win" instead of the "W" only. While the prefixes W16 and W31 have the same meaning, the 32-bit windows viruses do not. W95 stands for virus that infects only the '95 platform and W98 means the same to Windows '98. Win9x means '95 and '98, WNT are viruses that only run under NT and W32 viruses should run under all windows platforms. These prefixes are quite tricky, because other programs use other prefixes, but they have the same meaning. Therefore, it is not a bad idea to group Windows-based viruses only in two groups that have big differences in the file structure and detection - 16 bit and 32 bit.

In each group it is recommended to keep the number of subdirectories as small as possible by sorting the viruses to directories which contain the first letter of the virus only and then the subdirectory contains all viruses that start with this letter. An example is "File/Dos/A/AntiCAD/4096/A" for the dos file virus AntiCAD in the variant with a length of 4096 bytes, type A (see table 2). Some viruses start with a number or with an "_". These samples can be copied to one directory, because they are only a few. A directory named "0" which contains all of these viruses would be a good idea. Some scanners have problems with large directory trees that can be found in virus collections only. Further, it is easier to restart scanners on these directories if they crash or have other problems.

*Script-based malware* can be divided into BAT (prefix BAT), HTML/VBS (prefixes HTML and VBS) viruses and mIRC (prefixes mIRC or IRC) worms. Some anti-virus programs are still unable to find all this kind of malware or parts of it. VBS and HTML viruses are very similar, because HTML viruses only contain VBS parts. HTML itself does not contain any harmful commands. Therefore, they could be sorted to the same group or not - this is a decision of the tester.

BAT viruses require their own group, because there are no similarities to other kinds of viruses. Most BAT viruses are very primitive and do not work correctly. Therefore, the replication of this kind of viruses is not discussed in the following part. Often, only the first generation of these viruses run, but the second one will fail to replicate. Therefore, they are mostly intended viruses or can be grouped into groups like Trojan horses, which is a subset of the "other malware" part.

mIRC worms are a special problem - they cannot be replicated to get new samples, because a copy of a worm is normally the same file. They can only be observed if they spread. For this reason, an IRC server has to be set up - there is no other way to test whether they spread or not. An existing IRC server could be used, but normally all people would be infected when downloading the script that the worm offers. Because of this, it is a better way to sort this kind of malware only by the names of anti-virus programs. But most of them only use a heuristic based upon a small scan string (see the pre-sorting part of this paper) and often the files are damaged a little bit, because the end of the files are missing. Therefore, it is not an easy task to sort them, but very few programs use a CRC over the whole script file to identify the worm exactly.

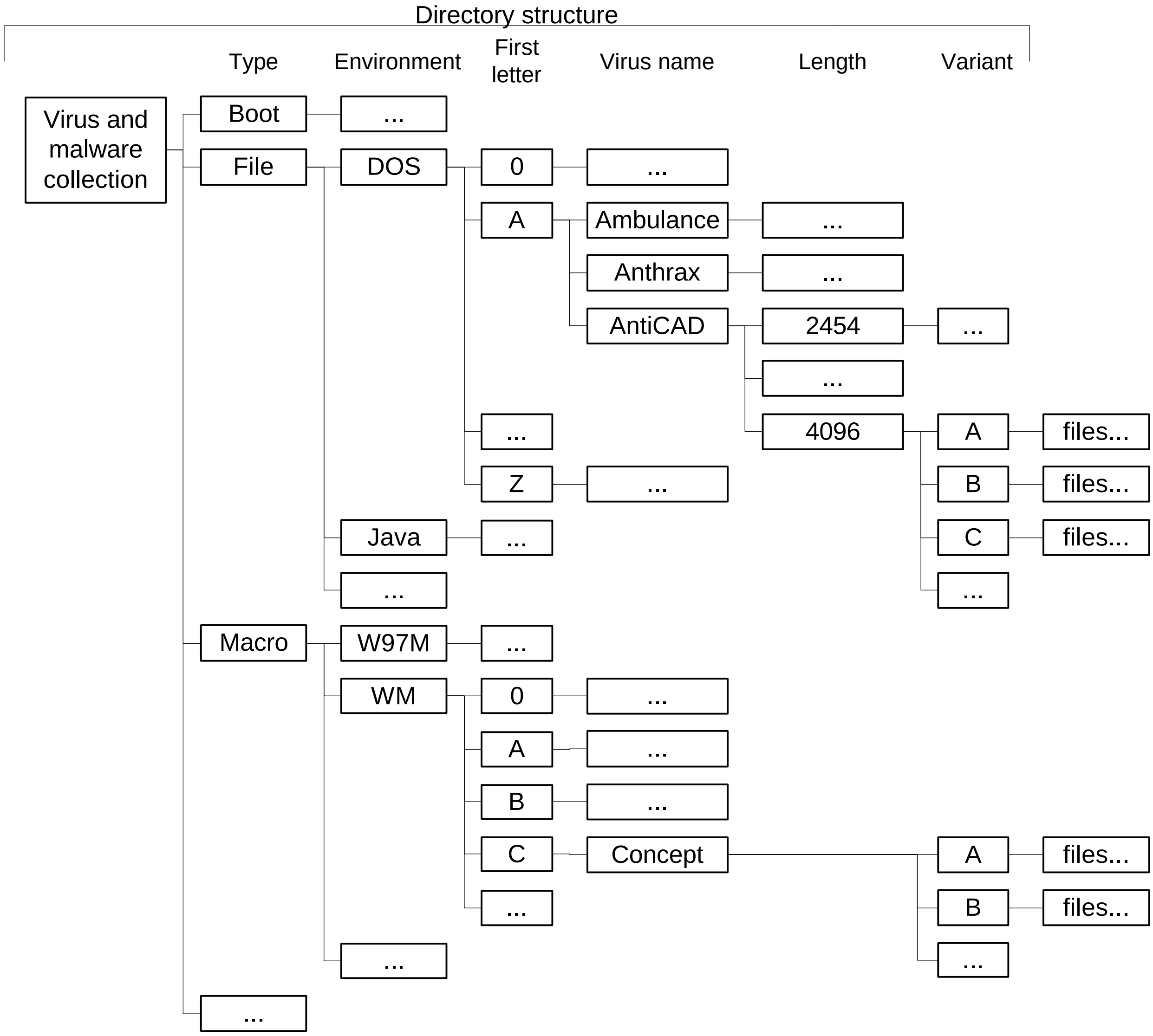


Table 2: Suggestion for sorting the virus collection

The last virus-related category is *Trojan horses, backdoors* and *other malware* that harm the system intentionally. It is difficult to group them in categories exactly, so the name “other malware” should be applied. This includes first generation (germ) files from viruses which do not run as expected, but can destroy data because the trigger and damage function is still active. A system crash is usually also a reason to detect the program. If the germs run and replicate well and the form of the virus is the same (e.g. the same type of infected files and entry point jumps) they should be copied into the respective collection like File/DOS to the other files. Very often germs can be found in a bad collection because the collector does not replicate the files. They are dangerous, because the typical user will only get such files if he downloads “something” and it is hard to clean the system if the germ cannot be found and only the infections can be cleaned each time. Virus droppers should be sorted in the “other malware” category, too. This includes boot virus droppers that write an image of a boot virus to a floppy or hard disc. Some programs have a special option to enable the feature to detect Backdoors like BackOrifice or SubSeven. The reason for this is that some backdoors like NetBus Pro are distributed as commercial or shareware programs and the authors do not like their work to be detected as malicious code (Hansen 1999). There are some companies that use such backdoors as a remote access tool and even Microsoft has a program called SMS that can be used as backdoor (Rötzer 1999).

Last but not least, the false positive collection should be prepared using executable files and documents from the network server of the magazine or a company, together with files that can be found on CD, disks etc. Because on some CD’s of magazines there are viruses, the tester should keep in mind to check this and only add really “good” programs. Joke programs should not be included, but it is an interesting idea to include innocent files found in virus collections to check if the anti-virus researcher works carefully enough. Some anti-virus programs have got false positives in graphic files like BMP or GIF, so files of this type should be in the collection, too.

## Replicating Samples

There are some ways to replicate viruses (semi-) automatically. (Helenius 1998) used a network server, one monitoring PC and one victim PC for replicating viruses fully automatically. For this, the hardware had to be modified. There are other solutions that only require one PC, like an Unix or Windows NT machine with a restricted DOS (emulation) task or a complete system emulator like VMware (VMware 1999). The viruses can spread under these conditions and, if there are enough replicated samples, the task can simply be closed to deactivate the viruses from running.

The basic scheme of a DOS file virus replication process is always the same. The victim PC is initialized by booting from a write-protected disk that contains the memory and network drivers, a RAM disk driver, a hard disc simulator, a goat file program and some BAT jobs to automate this process, including a reboot sequence.

A goat program is used to create small or large, different or similar COM, EXE or SYS files that viruses can infect. Such files usually contain only a hello-world-like routine to display a message and become incarnated. The rest of the file contains only 00 (hex), 90 (hex) bytes or some virus samples could be generated containing random bytes only.

Goat programs are useful to create a large set of files with the required attribute for the special virus. Often viruses do not replicate, because there are no viruses, but an intended or germs variant or Trojan horses. Often they do not run, because they require a special environment, such as a special processor, graphics card, memory size, date or other operating systems or software. For a tester, it is hard to analyze the virus’ special requirements to make the virus replicate. It might be that the description of anti-virus program developers could help to solve this problem. However, not only the goat files should be infected, but normal files like command.com or the network driver should be infected or goat programs itself could be kept for further analysis. Maybe some files are not infected correctly, and therefore, they should carefully be checked to see if they run and if they can infect more files (recursive further infections as a sign that it is really a correct infected file that contains a working virus).

Replication of windows-based viruses is a special problem. Windows cannot be started from a floppy disc or network, so a complete windows installation has to be prepared first on a real hard disc. After this, Drive Image by PowerQuest or Norton Ghost by Symantec should copy a 1:1 image to another drive of this disc or to the network. Next, the virus can be started manually and if the virus becomes memory resident like W95/CIH, it is enough to start some programs to infect them. Usually in the Windows directory, there are enough programs that could be started and infected. In case of W95/CIH, it is sufficient to read all the files once by opening the Explorer to the Windows directory or copy all files to the NUL device, because it infects all files while opening. Some viruses like W95/Marburg are not memory-resident, but they are using fast infector methods that look for some files to infect at each start of an infected file. After some time of doing this, all of the files should be copied to a network directory and the image of the disc should be written back. Only then can the the next virus be tested.

The same mechanism can be used to replicate the different types of macro viruses, but only one Office version should be installed at the same time and only macro viruses for that t version should be tested (e.g., WM viruses and Office 6.0 or ’95; X97M viruses and Office ’97). This prevents the risk of upconverted viruses that some people classify as creating new viruses, for example a Word ‘95 WordBasic virus (WM) that upconverts to a VBA Word ’97 (W97M) virus, which can look different after every upconversion process (Bontchev 1999). However, this view is subjective and this problem can be argued in the way, that upconversions are not “new viruses”; that they happen every day in real user environments. This is a possible danger for them and there could be the need to test the detection of these upconversions. For this, the virus has to be tested to determine, if the upcoverted virus is still able to spread or if it is another type of malicious code that can still harm the user. For the replication process, an infected file should be opened in the Office application the virus has been written for and the tester should create new files, write something into them and then close them. Again and again, the infected file should be opened, saved and closed. Most viruses that use auto-open macros or macros that replace standard functions of the application should have now infected some of the goat files the user has created and saved. After each replication, the image has to be restored to avoid having more than one virus activated at one time where a mix-up of them could happen.

An interesting topic of virus replication is to create a collection of different looking samples of polymorphic file viruses. For this, more than one PC should be used and also the scripts should change some conditions like the date and time of the machine. Some viruses only change a major part of their polymorphic start program after rebooting with a new date. Other viruses check the date before each infection and change the code if a new date has been found; others change their complete code on each infection. Such conditions have to be analyzed before such viruses can be used to create a polymorphic test set. Such test sets should contain at least 1000 samples, but 10,000 or more samples are more useful to get information on the ability of the program. Of course, it is important to check if the generated files still run. Some scanners should be run over the collection and if they all (or nearly all) fail to identify the same files, these ones are likely to contain a damaged variant, which should be tested.

Furthermore, a test set containing only replicated ITW viruses should be generated. Grouped in different categories like "File", "Boot" and "Macro" (or more exactly), it should be prepared using information that can be found in the current (Wildlist 1999). It is not always easy to get the correct virus because different scanners use different names and the Wildlist uses one of the names reported, so log files from other scanners should help to find the right virus from the list and the samples can be put in the collection. It is very important that the tester puts only the exact variant of this virus in the collection. If this variant is not available, this virus has to be left out and no other variant should be put in the collection instead. Additionally, for the ITW collection the same number (e.g. 1 or 10) of files should be used for every virus so that the tester gets comparable results.

## Preparing test files for compressed and archived files

Usually, scanners should also be tested regarding their ability to detect viruses in compressed and archived files. To prepare a test set of runtime compressed files, it should be noted that only file viruses can by tested this way. However, not all types of file viruses can be used for this test, for example, slack space, EXE header or zero hunter viruses, which searches for some free space at the beginning or all over the file, to store the virus code here, cannot be used. Usually, the compression program doesn't care about these blocks and the files will be destroyed and because of this, most anti-virus programs will not find anything. An example for this is Zero_Hunter or W95/CIH. For DOS file viruses, programs like PkLite, Diet, ICE, WWPack, LzExe and UPX can be used. The files will simply be processed by the packers. For Windows file viruses, programs like Petite, WWPack32, and UPX, which simply have to be executed to prepare this part of the test. However, each compressor should be tested on more than one virus infection and for different file types like COM (DOS viruses only) and EXE. It should be noted, that not all virus infected files can be compressed. This includes files which are too short and most infected Win32 files, because of the complex internal structures. However, worms like Explore_Zip or Happy99/Ska and other malicious software, like backdoors can be compressed successfully and used for this test.

If the ability of the products to detect viruses within archives should be tested, all types of viruses can be used, except boot viruses (in image files), of course. It is a good idea to include only one virus in each archive and some clean files around (before and after the infected file) them and to prepare some archives with subdirectories where the infected file can be found. This is, because some scanners only check the first files or are unable to scan subdirectories in the archive. Others do a "dumb" scan over some parts of the archives and too short files like the EICAR test file cannot be compressed and could be found by accident. There are many packers available, like ZIP, CAB, ARJ, LHA, RAR and ACE. It should be tested if the program is able to scan the archives recursively and if the scanner can only process one packer (example: ARJ → ARJ → viral program) at one time or if it can handle more (example: ZIP → CAB → viral program). How deep this process is to go is another issue, but it is unfeasible to test more than 10 iterations or all possible orders because their number grows very fast. Another test could be the detection of run-time compressed files in such archives and if the scanner is able to handle it recursively.

Of course, the detection of self-extracting SFX archives should be tested, too, if such an option is available in the compression program.

### Preparing test files for macro viruses

OLE/COM (Object Linking and Embedding / Compound Object Model) is a powerful end-user tool to keep many objects into one file with an own file system. Many programs, like most programs of the Microsoft Office suite uses it and the abilities of scanners to check such files correctly should be tested. For example, if a scanner is able to find a *linked* or *embedded sub-document* even if linking and embedding is applied over several levels. Furthermore, Word documents can be saved as *native RTF files*. This removes macro viruses if they do not block the transformation. However, embedded files and other objects are still saved in an hex-like ASCII format and some scanners could fail to find the embedded viruses, even if they can still activate -easily.

Another macro-virus related test should check whether or not the *user macros* are removed during a disinfection of a macro virus on different applications. People can get very angry if the programs simply remove the user macros and viral macros, because they can be very important to them. This test can be done for Office ’95 and Office ’97 files, where the user macros can be some simple comments and a message box. Since some macro viruses like WM/CAP remove user macros, it is necessary to check the documents after infecting to see if the user macros still exist and run.

Word and Excel ’95 files can be saved *encrypted*, but the encryption is rather weak, because the programs only use a 16-byte XOR key depending on the user’s password. It is easy to find the key using a known plain-text attack on these files, because there are some strings of the application which can be found every time in encrypted areas. Macros are encrypted completely by this function, too. Some viruses also add a password to saved files as a part of their damaging function. This should be tested as well to check if the program is still able to find the virus inside the password-protected file and clean it by removing the password or not (the best solution would be that the program lets the user decide if he or she wants to remove the password or not). Office ’97 uses a much stronger encryption method (RC4, 40 bit) that cannot be broken easily. Macros are no longer included in the encrypted areas, and therefore, they can be scanned. However, those files cannot be cleaned completely, because a very important part (the 0/1Table) is still encrypted (Marx 1999c) and cannot be changed, so the macros can only be disabled by overwriting them with a string of the same length and not by removing them.

Office 2000 has the capability to allow users to *sign* their *macros digitally* to prevent the run of viral macros (Marx 1999a, b; Ducklin 1999; Chi 1999). Since this feature is rather badly implemented, macro viruses can be signed easily, too, maybe together with user macros. Therefore, it should be tested, if the scanners find the viruses, clean the files correctly leaving the user macros intact, and removes the digital signature so that Office does not report an error message based of an incorrect checksum.

Another feature that should be tested is the detection of viruses in *.MSO files. These files are generated in Office if the current document contains macros and the file has been saved as HTML file in a special subdirectory. In this file, Office will store all macros and some other information in a new, special format in compressed form. The detection and correct disinfection of macro viruses that has been saved in this format should be tested because more and more users use Office 2000.

There are more possibilities to create special test files for the scanners (not only for macro viruses), but this should be enough for this paper. A good tester could find more possibilities what could be tested - but the tests should be relevant for the user.

### Documentation of the collection

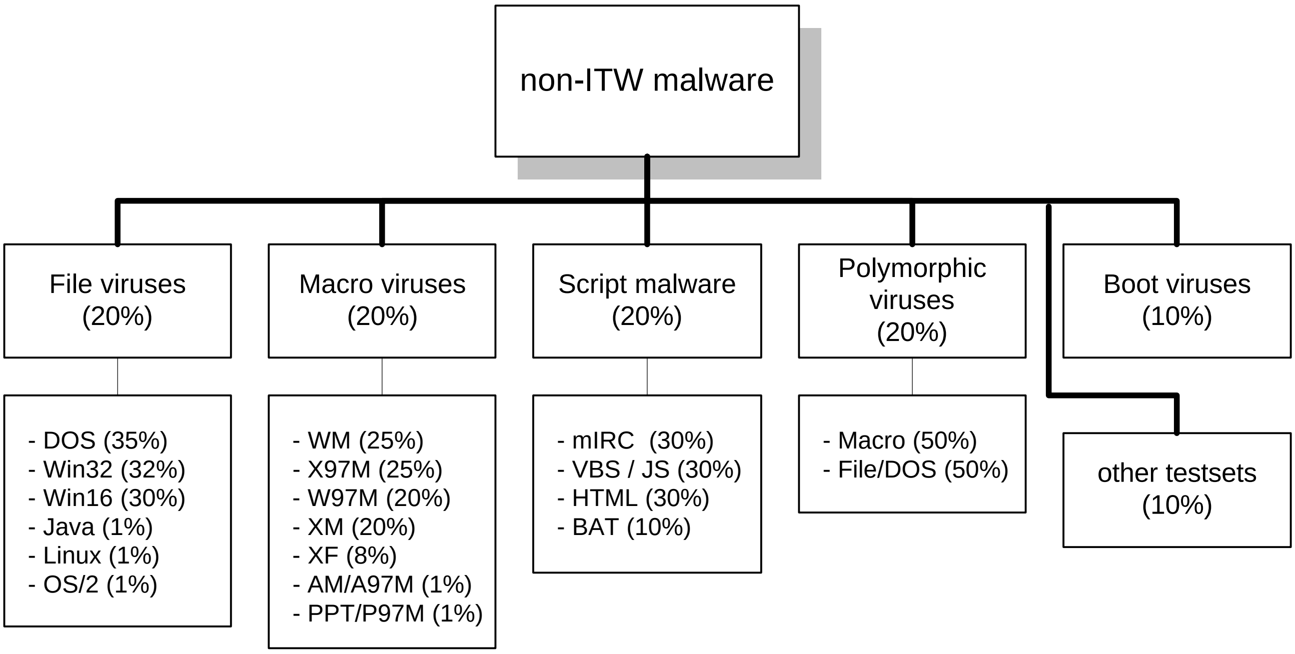
It is very important to make documentation about the viruses in the collection and publish it in a digital form, because it makes no sense to print it in the magazine if it is quite large. This documentation can be done using a script that counts all files from each directory of the sorted virus collection and writes them to a report file with the virus name (which can be get from the directory name).

Report files from all scanners and the options used in the test are part of the documentation of this collection, too, but this list is only available after the test. A report about what and how a test is made should also be given to the participating companies.

Last but not least, the idea should be discussed to give the collection used for the test to all virus companies who ask for it, but dangerous files should not be given to developers of programs that are rather poor in the competition or do not have a high reputation. This makes the test much more transparent, but only a few samples should be given away (e.g. one sample per virus) so the developers cannot cheat for the next test, because they could only include CRC sums to detect viruses they cannot find instead of using the usual scan process.

## Balancing the Weights for Different Categories

It is not easy to decide what is more important - the detection of macro viruses, the cleansing of file viruses or the fast, automatic update processes. For this reason, different types of user groups should be considered, for example, an Internet user, an “offliner” or an administrator in a small company. The first one can update the program as often as he wishes and frequently gets e-mails - maybe with attached word files. The second one usually gets new programs and data on discs and the last one wishes quick, automatic distribution of new updates to all clients etc. Because of this, the tester should use different weights for different types of users. An “onliner” usually gets macro viruses or worms more often than file or boot viruses. For this kind of user, macro viruses should have a higher weight. For non-Internet users, the detection of boot viruses is much more interesting. In this case, the tester should decide that online updates do not help the “offliner” and they are useless for him. The total weight for virus-related categories should not exceed 50%, because there are too many other things that make programs good or bad, like management features or update strategies. Our experience shows that there is no correlation between software quality and virus detection quality. Therefore, the weight of virus-related categories should not be too large (even if a good score is a precondition for choosing a product), because otherwise general software quality features might not be represented adequately. Another idea is that every reader can make his own decision about the weights for the categories, but this can only be done for reviews where the readers have sufficient knowledge about this topic.



non-ITW malware

File viruses (20%)

- DOS (35%)
- Win32 (32%)
- Win16 (30%)
- Java (1%)
- Linux (1%)
- OS/2 (1%)

Macro viruses (20%)

- WM (25%)
- X97M (25%)
- W97M (20%)
- XM (20%)
- XF (8%)
- AM/A97M (1%)
- PPT/P97M (1%)

Script malware (20%)

- mIRC (30%)
- VBS / JS (30%)
- HTML (30%)
- BAT (10%)

Polymorphic viruses (20%)

- Macro (50%)
- File/DOS (50%)

Boot viruses (10%)

other testsets (10%)

Table 3: An example of weights for non-ITW viruses

For the virus-related parts, it should be mentioned that ITW viruses are much more important than all zoo viruses together. The weight should be much higher for ITW viruses than for non-ITW ones, some say 60% should be used (Marx & Günther 1999), other say at least 95% should be the weight, but a weight in this range is sufficient. Furthermore, a detection score difference of 1% between the products on zoo viruses says nothing about the quality of the anti-virus programs, because this kind of viruses is not very interesting, because users will not encounter them. On the other hand, 1% difference in detection of ITW viruses is important.

For the prevalence between the viruses, the (Wildlist 1999) can be used, because the number of files that exists for different categories of viruses is not important. There are more file infectors than macro viruses, although macro viruses are much more important. It contains viruses that are reported to virus experts and corporate reporters from different companies around the world (Coursen 1998). However, this does not mean, that all common viruses will be reported, but most of them. In the ITW test set, three lists are available:

- the "Wildlist" (viruses which were reported by more than just one person),
- the " Supplemental List" (viruses only one person have reported) and
- the "Other List" (other malware besides viruses, like Trojan horses and Backdoors).

All three lists could be used to get correct results in this field. However, an ITW test set according to the Wildlist contains only the first part of it. This is, because viruses on the supplemental list come on and off the list very fast, so there is a definition problem if there are ITW viruses (which can be found for a longer period of time) or not. Second, the "Other List" contains non-viral malware only. There are several definitions of malware available, but there is no strict definition, if the programs listed here are malware or not. For other virus test sets both lists can be used, too, with a lower weight, of course.

While finding viruses increase the program's total score, occurrences of false positives and files damaged during repair should decrease it. For example, every false positive will decrease the total detection score by 0,1 to 1%, with a maximum of 10%.

An example of a weight for the group of online users for the non-ITW viruses according to (Marx & Günther 1999) can be found in table 3.

Even if some categories do not have a heavy weight, they should be noted if they are tested. Some people are interested in those special areas, for example, if a company has problems with XF viruses, it wouldn't use a program which is unable to find such kind of viruses. Other virus types like Perl, ActiveX controls or a special group of polymorphic Win32 and script viruses were not tested this time, but such categories surely become important in future tests.

## Part II - Evaluation and Testing the Anti-Virus Programs

In practise, it is impossible to test all features and aspects of software. Furthermore, only snapshots of the product life cycle can be reviewed and tested. Software is complex and unsteady, it is very easy to change it and minor changes might cause a large effect (Heisel 1998). If only a small section of the product life cycle is tested, errors and bugs can simply be overrated if they only appear in the tested version. However, all programs have to be tested in the same (hardware and software) environment with the same requirements, because otherwise they are not comparable.

If something unexpected happens, e.g., the program cannot be installed, if it is completely unable to scan for viruses, or is extremely slow, the programmers should be contacted. In this case, it is possible that wrong settings were used to test the programs. There are some switches causing huge internal changes in the program, e.g. an activated heuristic. Furthermore, unknown bugs can also cause such problems. Sometimes bugs can be a little bit curious, e.g., if the programmers only use 16-bit-INTEGER variables, the number of scanned files or found viruses switches to zero if it exceeds 65535. Even if the tester is able to get the correct results in spite of such small mistakes, the programmers should be informed about them.

### General Product Evaluation

In the field of software engineering, two common strategies for software testing are known: black box and white box tests (Pomberger 1996, page 151ff.). In a black (exterior) box test the input/output behavior is tested without knowledge about the internal structure of the program works. A white (interior) box test is based on knowledge about the internal structure of the program.

### Black box tests and system test strategies

Since neither source codes nor any other internal information is available about anti-virus programs, only black box tests can be applied. The input of such tests is a file that is infected by a virus. Then, the program has to decide what kind of code it is, the output will be displayed on the screen or in an report file for further analysis and the detection score may be increased.

It is not the objective of an anti-virus comparison test to find as many bugs as possible (this is a task of the software producer who hopefully found and fixed most of them), but at least it should show that the program runs stably and efficiently in most use cases. For this reason, the tester should make sure r the following software quality features are reviewed according to (Wallmüller 1990, pp. 194f.):

- The program package is *complete*, if it contains anti-virus functions for finding viruses on-access, on-demand, in archives and is able to clean them. Tools for global administration of this program or automatic update distribution should be included, too.

- A *volume (mass)* test on many different files should be done (if they contain viruses or not). If files are open (like mail or user databases) or closed, it can be analyzed whether the program is stable on all scanned files even if there are some corrupted files (for example, damaged OLE structures in Office documents) or unexpected structures like (Raiu 1999), because they can get real world problems very fast. Furthermore, the behavior of the program in case of a problem should be tested, e.g., if the program triggers a warning message or a blue screen, because it crashed or halted the system. This includes the ability of the program to scan deep directory trees and the ability to handle large report files without problems.

- The system should be tested under *heavy load* over an extended period. A typical test scenario is a file server where the anti-virus program is running and hundreds of PCs are simultaneously accessing the server or a system where many applications and tasks are running and require a large amount of system resources.

- *User-friendliness* includes that the program gives enough and good feedback in case of errors, interface and status monitor are designed in a way that all-important features can easily be viewed at the same time and an online-help exists.

- The *security* of the program is another aspect. It should be impossible for non-administrators to change settings, delete or manipulate program's data and to uninstall the program. Obviously, the program must not change important security settings of the system automatically and the software may not contain bugs that cause root exploits (the program has usually administrator rights to be able for scanning all files) or other dangerous scenarios.

- The program's *efficiency* is given, if response times are acceptable under usual and higher than usual workload.

- It has to be tested, if the program can be *adapted* to user requirements in different use cases. This includes all tasks to be performed if viruses or other malware were found.

- The anti-virus program has to be *compatible* with other versions of itself, the operating system and application programs. This includes the report file format and the signature data files that have to be supported by all versions, the documentation of updates and information policy. If something is not compatible (like new signatures and an old engine version or old signatures with a new engine), it should clearly be documented and an error message has to be displayed in such a case.

- The *documentation* of the program should be complete, describing all-important facts and be easy to understand. If the documentation is online available online or on CD-ROM, at least the (de-)installation and all emergency situations should be available in a printed version, because the online documentation cannot be read in such cases.

- The software package should include a *service level agreement* including all regulations regarding hotline support, updates, upgrades and online services. If the support shall be tested, it is important to call the hotline anonymous and write e-mail from a special account. Questions could be according special viruses, functions, settings etc., however, the difficulty should be the same for all producers according a special question. Another idea is, to send a fresh replicated sample from a new or unknown virus to the vendors (using quarantine functions or automatic sample sending features) and let it analyze by them. The response time and the quality (Does it function on all samples? Is the disinfection routine correct?) could be tested here.

- Does the *installation* and *deinstallation* routine of the program uses an assistant or other functions to make the life easier, does it require a reboot or can the software be used without it. A deinstallation should remove all shell extensions (the context menu that will be opened by a right-mouse click in the explorer and other programs), all files and added or changed registry keys.

**Bugs in operating systems - the program is not always wrong**

The operating system or device drivers cause some larger problems. In particular, if DOS programs are tested in DOS boxes of Windows environments, timing or file find-/open-/close problems might cause unexpected crashes or other non-deterministic behavior (Brunnstein & others 1999). Because of this, the programs have to be tested only in the environments for which they have been written. Furthermore, minor release changes of the operating system are relevant for programmers. Therefore, programs often include many CASE instructions for all existing different versions. The same is necessary due to files formats. For example, even minor changes from Office '97 to 2000 cause much trouble, because these changes hinder the correct disinfections. Therefore, anti-virus programs have to be changed even if there are no new viruses.

## Virus-Related Tests

Anti-virus programs are available for many different operating systems and hardware platforms. A relevant index number for the comparison of their behavior in different environments is the detection score, even if most parts of the programs are only compiled using other compilers and many instances of #defines's in C/C++. However, this is possible in 32 bit platforms like Windows NT or Linux only. 16 bit platforms like DOS (without an 32 bit extender) usually have a different and/or older engine and other results, but finally they should nearly reach the detection score of their 32 bit companions, including macro viruses. For the end-user review, it should be sufficient to use the 32 bit platform products only.

### Test of on-demand scanners

Today, DOS scanners can only be found on emergency boot disks or as undocumented parts of the program package. They are important in case of emergency scenarios, which will be described later. Their usage is similar to the Windows command line scanners of the program, but the handling is different from the GUI versions, of course, because the GUI versions usually cannot be started with command line parameters. They have to be configured using INI files or options that can be chosen in the GUI. If the tester can only review one version, it should be the Windows version and not the DOS one, because the end-user would usually use the Windows-based engine only.

It should be noted that some scanners are able to scan into all types of files according to their extension. Unfortunately, if the collection contains renamed files as suggested - to hinder execution by accident - some programs do not use heuristic analyses to scan for unknown viruses in such files. For this, the files have to be renamed to executable extensions. After the test, they could be renamed to the original names. Options like scan archived files should be activated - some scanners will need it to scan into PPT files, because of the complex structure.

Only a report file should be used to get the final detection score for the program in different test-sets. It is a problem if the program displays wrong data on the screen, because it has scanned more files than the tester wishes such as repeated scans of the MBR, boot sector, and root directory of drive C. Another example is, if the program has more to display than just the number of infected files, such as the number of possibly infected files or errors. The next example is, if the program crashes on large databases, the tester does not need to summarize the scores all different directories, but an automatic script parsing the log files can do this automatically and more exactly. Such scripts are not easy to write, because all scanners use a different format for writing data to the log files. For example, most scanners use one line for one file, but others use much more lines for exactly the same data. An other group of software uses different schemes of counting viruses. For example, a program that uses two virus-detection engines may count every virus twice if both engines have found a virus or a program may count every virus signature found as a virus even if it has been found it in the same file. Another problem occurs in files that can contain another objects, like packed files (ARJ, ZIP) and OLE/COM files. Some scanners only count the archives, but not the objects or files in the archives, other count only the objects, but not the archived files etc. In all of these cases, the report file is very important to get the correct detection score.

Some programs still cheat to get higher scores in the tests using “features” some companies have documented which others do not have. Usually after some ten or hundred (different) infected files, the scanner will increase the heuristics or adjust the detection by switching the engine in a more paranoid mode. They are described as features to detect more viruses on a heavily infected system - a system that would not be found in the real world, because no system can be infected by so many viruses. More likely it has been established that some testers use the wrong options or settings (Klotz 1997). However, the tester can use a strong weapon against this if all false positives are copied into special directories of the virus collections (Brunnstein & others 1999) or directly to the viral directories using special file names, and automatic scripts processing the report files filter them out. The program has to scan for both types of files and if it cheats, the detection score and the false positive score will be increased together and finally the program will loose some points.

The only category of viruses causing trouble in on-demand scanning is boot viruses. To test anti-virus programs, tools like SimBoot (Gryaznov 1994) - which usually run under plain DOS only - can be used or all image files in the test set can be written back to disc and the scanner can scan one after another. Neither variation is satisfying, because no real viruses are used for this, and the performance measurement of an antivirus product on simulated viruses is not a reliable measurement of the product’s real ability. For example, most anti-virus programs are too intelligent for this, so they do not search for a boot virus on a 3,5” disc that can only infect 5,25” discs, they are unable to use the simulated disc, or they are searching for further information about the infection and not just the boot sector. Anti-virus programmers know about that problem and most of them have implemented algorithms to scan for boot viruses in image files. However, this ability does not reflect the ability of the product to detect real viruses. Some require a special extension; others have to be started with special options. Therefore, the tester has to look carefully at the results and if they are too low the programmers should be contacted about how to proceed.

All other categories can be scanned easily part by part. Mostly, crashes are the only problem. They occur after a special directory size is reached, a large number of files are scanned, the size of the report file grows too large, or a special virus that cannot be analyzed further is scanned, or simply “randomly”. After such a crash, the scanner has to be restarted at the last directory scanned. For this situation, it is important that a new report file is used so the old one cannot be overwritten anymore. After the test, there is the need to check all scanners, if they really scanned all files and if not, they have to be restarted on some directories. Automatic scripts could do much to help solve this problem to get the path and names of the unscanned files, too.

**On-access guard tests**

Windows guards are mostly written as VXD files. Such virtual device drivers can only use two methods to communicate with the user after a virus is found or a problem occurred: they could display a “blue screen” or they could use a 16-bit window. If they use the first method, all other tasks are blocked and they can ask the user in text mode what should be done. Unfortunately some graphic cards have problems with such switching from graphics to text mode and back. The other possibility is to display a 16-bit window that prompts the user for how to proceed but all other tasks are still running and the guard has to block the action (e.g. file access) first. After this, it asks if this was OK and what to do now. Neither method is not satisfying, but no other methods are known that could help solve this user-prompt-for-action problem.

Boot viruses are problematic to test. Only discs with a real infection should be used for this and it is very important that they are write-protected so that they cannot be overwritten by the disinfection routine of the guard. The same write-protection applies to Windows, because it automatically overwrites some bytes during the start of the boot sector every time a new disc is read without write-protection and until now it is not clear why this is done. Some programs have to be re-configured for this test, because they do not scan for boot viruses. The reason is that while reading or writing to the boot sector of the disc, the virus cannot become active. This only happens while booting from an infected disc or by a boot virus dropper. Other programs report the infection without problems; some only scan for viruses on a disc while restarting a PC. Therefore, the tester should ensure that the test is performed the right way and the different philosophies should be noted in the test report.

Other types of viruses are much easier to test with an on-access scanner installed. They can simply be re-configured to delete detected viruses automatically while reading and then the tester reads all files at one time. This can be done, for example, using “copy *.* nul” or “xcopy *.* nul /s” and to answer the appearing question with “file”. After this, only the viruses the program did not find are left over. Another suggestion is to use an option like “report only” while copying the files and to process the report file after it. Some on-access scanners cannot be configured to process the files found automatically, so a user has to answer the questions or an automatic system like in (Helinius 1998) has to be used. Unfortunately, some scanners are not programmed well and they have timing problems between their VXD and their display part. After a number of files or some time they loose their protection - they still prompt the user for action, but they let the virus pass in each case (Marx & Günther 1999). Of course, all these tests have to be prepared first, i.e., all viruses to be tested should be copied from the network drive first to the local hard disc.

After testing to determine if the scanner is able to detect a virus in those files, it should be tested to determine whether it is able to prevent its execution, too. For this testing, it is better to only use files that on-access scanners are able to find in the test copy, because if the scanner is unable to find the virus, it will infect the system and the machine has to be reinstalled using DriveImage or Ghost which will cost some unnecessary time. Each virus category should be tested to determine if they prevent the execution of a DOS- or Windows based virus, on saving an infected attachment, or during a download. For macro viruses, it should also be tested, if the on-access scanner is able to prevent the opening of an infected document, like from the start in the Explorer or by File | Open. In case, the user wishes to prevent the opening, Office programs try to open it up to three times and some scanners warn the user three times instead of ignoring the other attempts and still block them - the user only gets confused by this and a good anti-virus program should be able to display the message only one time.

A particular type of on-access scanners is also used for mail- and file-servers. They are able to scan inside archive files like ZIP, but in the case of on-access scanners for clients, it is an unnecessary feature, because such a virus cannot become active. But, of course, an on-access scanner has to be able to detect viruses in runtime-compressed files, and therefore, this should be tested. The problem is that most programs are unable to do it and the system becomes infected and has to be restored. Such files can be tested using the copy method, too. An additional test could be made for shared network resources - using a machine with infected files and an installed guard and a machine without a guard. Now the machine without the guard should not be able to access the infected files. Unfortunately, some scanners do not hinder this action.

Some anti-virus programs contain an additional behavior blocker that should prevent the execution of malicious code or the damage that could be caused by them (for example, overwrite system sectors or delete important files). Some of them have a teach mode where they learn what usual programs do and what they do not do. After this, they prevent actions that are against the learned rules and prompt the user about what should be done. There are no good test strategies known for this, because most of these projects are not supported anymore or they are simply not behavior blockers by definition. Other companies have integrated the on-access guard with the behavior blocker, so that it cannot be tested separately.

### Test of disinfection

The function to clean infected files was implemented as an emergency solution a long time ago and is still maintained, because users keep asking for this feature, in spite of the fact that the best solution would be to restore the files from backups. Unfortunately, macro viruses have changed this, because the need for a disinfection routine has been multiplied - most of the time only outdated backups of such files exists and they contains user data and a virus at the same time. Some programs still rely on the backup/restore strategy instead of cleaning file viruses, because it is always dangerous to restore a file if the original state of it is unknown. In addition, the virus may have destroyed the victim. However, for emergencies like the W95/CIH, disease killers are available separately, but are not included in the main product.

It is very time consuming and not a good idea to test the disinfection of boot viruses, because many replicated discs (or backup images) are needed for this procedure and after the disinfection, the tester has to examine the floppy disc to determine if it is completely readable. However, real hard drive infections are much harder to test, because the system has to be infected for this, so this will be discussed later.

It is hard to decide whether a file is cleaned correctly or not. Mostly because the file is not in the same state as before the infection, other strategies must be applied according to the type of file. The files that are in the original form can be separated together with those that could not be cleaned.

For file viruses, the cleaned files should be checked to ensure they still run correctly. However, the tester usually only has infected goat files and these run correctly after the test, because they do nothing or only print a text message and exit to DOS. Therefore, it is a better idea to determine the difference between the original, uninfected file and the cleaned file. In some cases, reasons for a program's inability to restore the original state can be deduced. For example, some viruses do not save the complete header of the file or its exact length. But such hypotheses require a lot of knowledge on the part of the tester about the different file structures and viral behavior, so the execution test is easier to apply. It should be noted that a single bit change is enough to prevent execution or the program may still run but it crashes in an unexpected situation. This could be a problem if the cleaned file is a complex program, because there is always the chance, that cleaning mistakes might occur which were not detected.

Another point is, that some ITW viruses do not run under Windows and cannot be cleaned under Windows, this includes DIR-II and other directory viruses, which manipulates FAT directory entries and do not change the host file at all. Most of them runs under MS-DOS 5.0 only. These viruses have to be excluded from a disinfection test, too, even if some scanners are able to repair them. However, it cannot be tested without major problems.

Another idea is to only use some viruses from which it is known that the original state of the file can be restored, like Cascade or No_Frills.1358 (COM); Delwin (EXE); Tremor and One_Half (COM/EXE); W95/CIH and W95/Marburg (PE-EXE). Only those programs that are able to restore infected files *completely* get the points, although some of the above viruses are not easy to disinfect. Examples are One_Half or W95/CIH, which writes parts from their viral code all over the file. However, such cases should not be a problem for a good anti-virus program and it is hard to see why some scanners are still unable to remove such viruses correctly, i.e., leaving no viral parts in the file. Additionally, a test should be made about how the scanner handles runtime-compressed files and double-infected files. A good scanner has to find the double infection while cleaning the file of the first virus and should be able to remove all viruses in the file recursively.

A special situation is the cleaning of an infected Trojan horse. The scanner should be able to clean the virus from the Trojan horse and in the second step, the Trojan horse has to be *deleted*, because such kind of malware cannot be disinfected. It should be noted that there are some backdoors like BackOrifice 2000 (the original distribution) that are not disinfected properly when infected with such things as the W95/CIH virus and after being disinfected, some scanners were unable to find the backdoor anymore.

The tests on disinfecting macro viruses are really hard, because there is no exact solution to decide if a scanner has cleaned the file correctly or not. Office '97 and 2000 files are especially complex to parse and to clean correctly. Therefore, the tester can only have a short look if the file can be opened after cleaning without errors. In the case of user macros, the macro virus warning should be displayed by Word or Excel (if it was not disabled). If there were only viral macros in the file, the macro warning must not appear. The tester should look to see if the file can be modified by inserting pictures and texts and if the file can be saved and printed (to a file only, to save paper). If all of these tests are successful, the tester should look to see whether the file is still able to handle user macros and if those macros run correctly. For this, a short MessageBox function test should be suitable, but some scanners remove the macro part so badly that is impossible to use word macros any longer. Existing user macros should still be complete in all streams and executable. If all of these tests are successful, it is very likely that the files were cleaned correctly, but unfortunately, there is still the chance that the files have some leaks in their internal structure and will crash on later modifications. Such tests should be done not only for Word, but for Excel (including excel formula) and PowerPoint on more than just one file and virus.

### Memory detection tests

It is very time-consuming to test the memory detection of viruses, because the system has to be infected for this, whether it is a boot or file virus. While testing boot viruses, often the emergency disc is needed, because Windows does not run correctly or only starts in “safe mode”. Other viruses cause no problems, but it depends on the computer, the version of Windows and the virus itself. For example, AntiEXE.A or Parity_Boot.B run well under Windows ‘9x, but viruses like Ping_Pong do not, because they only run on old 8086/8088 systems. However, it is quite tricky to test these viruses, because some Windows-based anti-virus programs are not able to disinfect the virus while it is in memory and some are unable to find it, others do not run and/or display the correct message instructing the user to start the system from the emergency disc. This procedure requires significant time, because after each test, the original state has to be recovered again using tools like DriveImage or Ghost.

Some file viruses like W95/CIH are known to infect systems without a problem, but it should be kept in mind that not all viruses go memory resident, like W95/Marburg, which is a direct file infector. In fact, an anti-virus program that is unable to find a memory infection will actually help the virus to spread further by simply scanning the files on the hard disc or network drives. The anti-virus programs vary quite a lot in this point. Some are able to disinfect the virus in memory, but do not display a notice about it. Others only display a message that the program was modified and should be installed again. However, the last suggestion does not help the user, because the files will be infected again and again. Some programs are unable to find such viruses in memory and use a quite interesting trick to disinfect files. If they are unable to detect the virus in memory, they clean the files, a virus like W95/CIH will infect it again, the program would search for a virus in the file and they will clean it again and again - and the program will hang in an endless loop. Because of this, some programs uses a trick, whereby they do not clean the file completely, but they leave the infection marker of the virus intact and so the virus will not infect the file again. After a reboot, the system is clean and virus-free. It should be noted that whenever a memory infection has been found and the program is unable to disinfect this one in memory, a message should be displayed instructing the user to boot from the emergency boot discs and what should be done to get the system disinfected completely.

### Boot-up-scanners and emergency boot discs

The last chance to scan, modify or clean a system without problems - or trigger functions of windows-based viruses like W95/CIH - occurs before Windows ‘9x gets control. For this situation, some companies have written boot-up-scanners, most of them are started in the AUTOEXEC.BAT file. They scan the memory, the system areas like all boot sectors or MBRs, and some files needed in order to start Windows correctly. For example, the root directory including command.com and some files from the Windows directory are scanned before Windows 9X gets control of the system. Unfortunately, most of the products do not find Windows-based viruses, because they do not scan a single PE-EXE file, like calc.exe or sol.exe. But it would be a great chance to detect them at this point, because the virus is not active at this time and the boot-up-scanner could prevent further infections. The program could scan the Office global macro directories to hinder infection by a macro virus, but this is not necessary because it requires too much time at this point (and it is more likely that a user will switch this protection off) and the memory-resident guard will prevent infection by such a virus. It is a good idea for the boot-up-scanner to not only work with the known virus signatures but to creates checksums over the system parts. In emergency situations they could be restored - from the hard or floppy disc. All these things can be tested and should be documented.

Usually one ore more boot discs can be found in the anti-virus package from which the system could be repaired after a virus outbreak. Some of the products want the CD to start programs from there, others only need the floppy discs that have to be inserted one after each other. Some programs can be started directly while booting from the CD, but most of the time, the CD (or the discs) are not up-to-date and the virus that harms the system may not be found or can be found by heuristics only. However, most of the time, there is no problem because most ITW viruses are old, except for macro viruses. Such boot discs should be checked if they exist in the package or the user has to create them. The boot discs need to be checked to determine if they work correctly, for example if they are bootable (some are not) and if they contain a scanner (some are bootable only, no more). The problem is, that every company has to pay license fees for using DOS, so some of them have written their own DOS versions in past, but a Linux boot disc containing a Linux virus scanner could be used, too. In this case, the disc content cannot be viewed under DOS, because of a different file system (ext2). However, they still run correctly and can be tested. Another problem is the data integrity of the disc - some are really old. Using the boot disc, the tester should look to see if it is able to find a sample infection by ITW viruses and if they can be cleaned. However, in order to make the program still fit on a floppy disc, some companies leave parts of their signatures off and/or compress them. As a result, the scanner on the rescue disc is unable to find macro viruses. Of course, such limitations should be noted in an exact test, but most of the time, it is enough to write down which philosophies the boot disc follows and whether this is acceptable or not. It should be noted, that if a user creates his own rescue disc, a virus could have already infected the system and could be in memory and so the rescue disc could also be infected. Such discs would be useless. It is a better solution for the vendor supplies it’s own, write-protected rescue disc set for this reason - at least the floppy disc (or bootable CD-ROM) to start the system. New Windows ’9x CDs have the feature enabled, to start from them, too.

### Test for compressed and archived files

It is rather easy to test the ability of the on-demand scanner to scan inside runtime-compressed and archived files if they are prepared, because they have simply to be scanned. After this, the tester is able to reveal the types of runtime-compressed files the program is able to scan and the types of archives that can be processed. For the recursive scan test, it should be noted, that if a program is, for example, only able to scan inside ZIP files, but not inside ARJ files. It can scan ZIP → ZIP, but not ARJ → ZIP and it can finally process archives recursively, which it has been shown because it was able to scan ZIP files recursively, however it does not know what to do with ARJ archives and the second test has failed. Another test could show if the program is able to clean infected files in their archives or not - maybe recursively, too. Password protected archives cannot be processed in real-time, even if some passwords are easy to restore. Therefore, no scanner would be able to scan inside or clean such files without a list of often used passwords or backdoor-keys.

### Macro virus related tests

It is easy to test some of the abilities a good scanner has while scanning in OLE files. For this type of testing, the created test files (see part I) should be used to demonstrate it's ability to perform scanning in special cases or not - the results can be found in the report file or on screen. This includes the ability for the scanner to scan inside embedded sub-documents and native RTF files and it's ability to clean them correctly (see disinfection tests for macro viruses). A second study should include testing to determine if the program is able to leave the user macros for Office '95 and '97 files intact while cleaning and if the scanner is able to find viruses in encrypted files. It should be able to disinfect Office '95 files with or without removing the password protection. In the case of Office '97, the scanner should not try to disinfect the files - complex documents cannot be opened after such attempts in many cases. A program should inform the user about it's unsuccessful disinfection attempt and substitute the viral macros with a message stating that a complete cleaning was not possible and informing the user that the now harmless macros have to be removed . In the Office 2000 tests, the scanner has to show that it is able to deal with these files (see preparing the test files for macro viruses section).

### Time, hard disc and memory requirement tests

The *performance* of anti-virus programs depends on the accuracy of virus detection and disinfection. If it only determines the possibility of a virus infection much less time is required than for an exact variant detection in most cases. The same applies to archives - some programs scan archives, others skip them because they are unable to unpack the archive and scan inside it. Such files can be found in most Windows installations, because the Java class archives are simply (in old versions renamed) ZIP files. It is not a good idea to test the scanner speed using the Windows and the Program Files directory, because the size of it will be different in every installed anti-virus product. First, some scanners copy VXDs and DLLs to the Windows directory and second, most scanners will be installed in the Program Files directory. In this case, it could be a good idea to use the false positive (e.g. non-viral) testset (that should contain usual Windows programs and DLLs) to test the scan speed and get comparable results - for both normal and archived files in separated tests. This has to be done separately from the other scan speed test. It is useless to test the speed of disinfection time of a program, because some are very accurate and look to determine whether the program is damaged and perform some tests, but others only try to heal them without any knowledge about the consequences. It should be noted that a program that needs more time on scanning or disinfection could be the better one in many cases, but this is not always true. Therefore, this is not an indicator of the quality of the program.

The *hard disc requirements* are easy to test. For this, the free hard disc space before and after the installation should be taken and compared. It is not a good idea, to test only the directory size of the installation directory, because shared DLLs and VXDs from the program will be found in the Windows directory, too (see above). Only the swap file causes a problem - it should be moved to an other drive or deactivated (if possible), because the size changes regularly and will be different after every reboot.

The next problem is to test the *memory requirements* of the virus guard, because Windows is only equipped with inexact tools to determine this. After every reboot, different free memory sizes are shown. Additionally, not only the memory a program requires is important, but also the resources the program actually holds are relevant. Therefore, it makes sense to test using an older computer with Windows '9x running slowly (like an 386 or 486) and low memory (8 or 12 MB RAM). After installing the guard (it has to scan all files at open) and performing a reboot, the tester should look to determine the amount of time that is needed for copying all files on the disc, to a network drive, or simply to NUL, because in this case memory requirements and system resources can be tested together. The actual slow-down caused by the on-access scanner is disclosed and can be compared with other programs. It should be noted, that the Windows swap file can cause a problem, because xcopy will stop if it cannot read a file - like in this case. A solution could be, to create the swap file on an other drive. Deactivation causes only a blue screen, because there is not enough memory available in the test system.

**Other tests**

It should be impossible for an unprivileged user or a malware program to *deactivate* or *uninstall* the anti-virus program without notice to the administrator. Therefore, the tester should delete as many of the files in the installation directory of the anti-virus program and the Windows\System directory, where often anti-virus databases or VXDs can be found. Normally, it should be impossible to do this, such as when the files are opened or locked. If files can be deleted, the program should notice that some parts of them are missing after a reboot and it should display an error message describing the problem and what could be done to correct it. Often, only error messages with confusing numbers or wrong error messages are displayed, which do not help the user in any way. Another test can be done to registry entries, to determine if they are removed and to see what error messages are displayed in these cases. It should be noted that usually many registry settings in different locations can be found for each program.

A good program should perform *a test of itself* and the virus database, so that the tester is able to analyze what happened. The tester can manipulate the program or the virus database by overwriting slack areas in the program or by overwriting more than just some bytes in the database with random values. In both cases, the program has to display an error message and should terminate.

After cleaning macro virus infections, the program should warn the user about the possibility that the *macro virus protection* was switched off by the virus (FitzGerald 1999). The program should be able to restore the original settings if the user want this since he would not notice it until the next infection.

A good anti-virus program has to warn the user if it's getting too old. So the tester should start the program with the PC's date set to a future date, like one year in future and check to see if the program displays a message stating that the program and/or the signature base is too old to run and should be *updated* immediately. It should be noted that those dates could cause interactions with backup programs or the network with switched-on time synchronization.

## Part III - Documenting and Editing the Test Results

After finishing all of the tests and evaluation parts, the tester will have collected a lot of information about the results of the anti-virus programs. Now it is time to compress the information and write concluding reviews about the programs. Furthermore, the reviews have to be edited according to the different kinds of audiences of the review.

### Getting the Final Results

A final result table of every program including the detection scores, successful repair attempts and all of the yes/no criteria should be prepared. Usually it is too big to be published in a magazine in its entirety, but it should be available, maybe on the magazine CD or online in the Intra- or Internet. It is especially important for the anti-virus companies to have access to such tables so they can see what has gone wrong and what should be improved as soon as possible. Usually, the anti-virus researchers cannot wait to get all the report files about the programs and documentation regarding the test.

To develop the final results for the different group of users, weights applicable to the different groups of users should be applied and the products should be sorted after the final result score is tabulated.

### Writing the Final Review

Before writing the final conclusion about a product, it should be reviewed and compared with previous tests to determine if the company has corrected formerly detected bugs and mistakes or made some improvements regarding detection. Additionally, the tester should make some screen shots of the products which gives an overall impression of the program.

The tester himself should write the article for the tester is the one who knows best what has been tested, what was good or bad, how the products vary because of different philosophies and the tester's final impression about the program. If somebody else writes such text only with the information from the final result table and some remarks by the tester, it is very likely that the text will give a bad impression about the products and wrong conclusions are drawn.

It could be a good idea, to send the producers the text and tables before publishing to let them check if all results are correctly, if sentences could be interpreted a wrong way and such things. Another idea is, to ask some people from the anti-virus companies if they want to write one or two sentences about the test or the tested product. This can be a "we are working on it" in case of problems or "we are happy about the results". For this, the companies have to know the results or at least the problems found.

Often, for example, if the page limit given by the publisher is exceeded, the article is re-edited by an editor. In this case, sometimes the “new” version contains bugs after this procedure. Therefore, the tester should proof the galley carefully.

### After Publishing the Review

After the text has been published, contact with the anti-virus programmers should be maintained. Procedures and test strategies should be discussed with them and they often have ideas how to improve some points. However, sometimes it is like in the dark ages, the messengers of bad news are killed with the comment that the tester is the bad one, not the company. Maybe it helps if the tester sends the complete result table and the full and shortened version of the texts to the anti-virus companies.

## Conclusion

Although a lot of features can be tested, the programs can only show their true quality in real world scenarios. The quality of a test depends on the test environment, e.g. the virus collection or experience of the tester. It is similar to a school environment where people learn a lot and have to write exams about it. They will be tested in the real world and only this can show how much they actually learned. Regardless, anti-malware tests should help the professional or home user to determine the best program for his or her special requirements to protect the system against viruses and other malicious software. Together with backup strategies, cryptographic software and intrusion detection systems, the anti-malware products should help to make the life in the networking world easier.

Unfortunately, no program exists that is the best in every category and the subjectivity of the tester really determines which program will win for a special group of users. There are too many things, like virus outbreaks or a big IT environment with some hidden surprises, which can neither be reviewed nor tested as wished. Of course, reviewing such tests as described in this paper should prevent the reader from buying completely the wrong program, the frustration, the loss of time and money experienced as result of a poor buying decision.

## References

Abrams, R. (1999). Giving the EICAR test file some teeth. Proceedings of the Virus Bulletin Conference '99, pages 275-280.

Bontchev, V. (1993). Analysis and Maintenance of a Clean Virus Library. Available at: *ftp://ftp.informatik.uni-hamburg.de/pub/virus/texts/viruses/virlib.zip*

Bontchev, V. (1999). The "pros" and "cons" of WordBasic Virus Upconvertions. Proceedings of the EICAR Conference 1999

Brunnstein, K. & Schmall, M. (1999). Makrokosmos. [Macro universe]. c't magazine 07/1999, pages 146f., Hannover: Verlag Heinz Heise

Brunnstein, K. & others (1999). VTC University of Hamburg Anti-Malware Product Test "1999-03". Available at*: ftp://agn-www.informatik.uni-hamburg.de/ pub/texts/tests/pc-av/1999-03*

Chi, D. (1999). Microsoft Office 2000 and Security Against Macro Viruses, a white Paper. Santa Monica: Symantec Anti-Virus Research Center (SARC)

Coursen, S. (1998). Taming the Wildlist. Proceedings of the Virus Bulletin Conference '98, pages 243-249

Ducklin, P. (1995). EICAR Anti-Virus test file. Available at: *http://www.eicar.com/anti_virus_test_file.htm*

Ducklin, P. (1999). Microsoft Office 2000 and Digital Macro Signatures, a white Paper. Oxford: Sophos Plc

Ducklin, P. (1999b). Counting Viruses. Proceedings of the Virus Bulletin Conference '99, pages 73-85.

FitzGerald, N. (1999). Monkeying with the Wildlist. Proceedings of the Virus Bulletin Conference '99, pages 247-268.

Gordon, S. (1995). Are Good Virus Simulators Still a Bad Idea? Available at: *http://www.commandcom.com/html/virus/res/simulator.html*

Gordon, S. & Ford, R. (1996). Real World Anti-Virus Product Reviews and Evaluations - The current state of Affairs. Proceedings of the 1996 National Information Systems Security Conference. Available at: *http://csrc.nist.gov/nissc/1996/papers/NISSC96/paper019/final.PDF*

Gryaznov, D. (1994). Simboot: A new tool for testing scanners. Proceedings of the EICAR Conference 1994, pages 157-164

Hansen, R. (1999). Elchtest für Windows [Elk test for Windows]. c't magazine 17/1999, page 90, Hannover: Verlag Heinz Heise

Heisel, M. (1998). Einführung in die Algorithmen und Datenstrukturen. [Introduction to algorithms and data structures]. Otto-von-Guericke-University Magdeburg, part 0, page 3

Helenius, M. (1998). Automating Anti-Virus Product Evaluation. Proceedings of the Virus Bulletin Conference '98, pages 251-259

Jang, D. (1998). Trend Chip Away Virus – reviewer's guide. Taiwan: Trend Micro

Klotz, K. (1997). Getäuschte Viren-Tester. [Deceived virus testers]. CHIP 06/1997, page 9, Munich: Vogel Verlag

Link, R. (1999), the list is available online at: *http://rainer.w3.to*

Luckhard, N. & Siering, P. (1999). PC-Parasiten. [PC parasites]. c't magazine 07/1999, pages 140-144, Hannover: Verlag Heinz Heise

Marx, A. & Michl, M. (1999). So schützen Sie Ihren PC vor Viren. [How to protect your PC for viruses]. CHIP 02/1999, pages 142-151, Munich: Vogel Verlag

Marx, A. & Günther, V. (1999). Einsatz der Virenkiller. [Operation of the virus killers]. WIN 02/1999, pages 186-195, Munich: Vogel Verlag

Marx, A. (1999a). Test: Makro-Virenschutz bei Office 2000 ist nicht ausreichend. [Test: The macro virus protection in Office 2000 is not sufficient]. CHIP 08/1999, page 12, Munich: Vogel Verlag

Marx, A. (1999b). Viren-Special: Im Vergleichstest : 13 Antiviren-Programme. [Virus Special: Comparison test of 13 anti-virus programs]. CHIP 11/99, pages 230-237, Munich: Vogel Verlag

Marx, A. (1999c). Gefährliches Office 2000. [Dangerous Office 2000]. *http://www.pcwelt.de/onlinewelt/showonline.asp?dir=o2000*

Polk, W. & Bassham, L. (1992). Guide to the Selection of Anti-Virus Tools and Techniques. Available at: *http://csrc.nist.gov/nistpubs/800-5.txt*

Pomberger, G. & Blaschek, G. (1996). Software-Engineering: prototyping und objektorientierte Software-Entwicklung. [Software-Engineering: prototyping and object-oriented software engineering]. Munich and Vienna: Hanser Verlag, 2nd edition 1996

Raiu, C. (1999). The little fixed variable constant. Virus Bulletin 10/99. Oxford. Available at: *http://homepages.gecad.ro/craiu/papers/*

Rötzer, F. (1999). Cult Of The Dead Cow gegen Microsoft. [Cult Of The Dead Cow against Microsoft*]. http://www.heise.de/tp/deutsch/inhalt/te/5102/1.html*

Scheidl, G. (1999). Virus Naming Convention 1999 (VNC99). Austria: Ikarus Software. Available by mail from: *scheidl.g@chello.at*

Sophos (1999). Low-down on virus names. Sophos News, August 1999 Oxford: Sophos Plc

VMware (1999). *http://www.vmware.com*

Wack, J. & Carnahan, L. (1989). Computer Viruses and Related Threats: A Management Guide. Available at: *http://csrc.nist.gov/nistpubs/sp500166.txt*

Wallmüller, E. (1990). Software-Qualitätssicherung in der Praxis. [Software quality-management in praxis]. Munich and Vienna: Hanser Verlag, 1990

Whalley, I. (1999). Testing Times for Trojans. Proceedings of the Virus Bulletin Conference ’99, pages 55-67.

Wildlist (1999). Wildlist Organization. *http://www.wildlist.org/WildList/wildlist.html*

ZDTag (1999). Virus Security Management Products Comparison and Stress/Scalability Testing. *http://www.zdnet.com/zdtag/reports/navirus.pdf*