Making Sense of Anti-Malware Comparative Testing

David Harley BA CISSP FBCS CITP*

Director of Malware Intelligence,

ESET, 610 West Ash Street, Suite 1900, San Diego, CA 92101 USA

Keywords:

anti-malware comparative testing; detection testing; dynamic testing; static testing

Abstract:

If there's a single problem illustrating the gulf between the anti-malware industry and the rest of the online world, it revolves around the difficulties and misunderstandings that plague product testing and evaluation. This article considers these issues and the initiatives taken by the anti-malware and testing sectors to resolve some of them.

1 Introduction

The testing world is, from a certain point of view, divided into a number of groups.

- There are those who consider that malware evaluation is just about detection testing, and that it's easy for someone with a quantity of samples to test the ability of a range of products to detect those samples.
- There are those who have a vested interest in disproving the bona fides of either the entire anti-malware industry or of products that compete with those in which they have an interest.
- There is a huge audience of consumers influenced by bold statements about product competence ranking that may indeed reflect comparative competence, but may also (or alternatively) reflect the testing practices and prejudices of the tester. (We are, of course, talking here of corporate evaluation and procurement teams, not just individual users.) Those who go beyond reading (or performing, or commissioning) a single review to basing their procurement decisions on multiple reviews and tests face a daunting game of "pin the tail on the donkey": either hoping to find the One True Review, or trying to synthesize an adequate evaluation from a jumble of conflicting reviews of varying conviction and competence.
- There are the providers of security products and services who are perpetually on the defensive, as their marketability fluctuates according to their perceived importance in highly visible, highly variable tests.

- And there are the mainstream testers. In general, these are to some degree allied to the security industry, because it’s very difficult to test security software without those alliances to enable the sharing of samples and information. However, they also (usually) stand to some extent outside the vendor community, due to the need to maintain integrity and independence.

For many years, the vendor community has been open to the accusation that while it was eager to protest at what it perceived as incompetent or intentionally biased testing, it was far less ready to provide help or guidance on what it considers to be “good” practice.

This paper will, therefore, examine the issues and problems that make testing malware far more difficult than is normally assumed, and evaluate some of the ongoing initiatives aimed at addressing those problems, including the initial work of the Anti-Malware Testing Standards Organization (AMTSO).

2 Testing and Evaluation

In educationalist circles, it’s sometimes said that “examinations are intended to find out what you know, not what you don’t know.” Many anti-malware product tests (in the broadest sense) derive from the opposite viewpoint: they are intended to “trick” the tested products into failing. It’s sometimes argued that this is essentially invalid [Andrew Lee& David Harley, “Antimalware Evaluation and Testing”, in “The AVIEN Malware Defense Guide for the Enterprise”, Edited by David Harley, Syngress 2007]: this may be an overstatement, but such an approach can be seriously misleading if it doesn’t reflect current (or likely future) real-world malware technology and practice. However, the problems are deeper than this.

We talk about testing in the anti-malware field as if it were much simpler than is actually the case. Firstly, it isn’t only about detection testing. Secondly, detection testing is far more complex than most people think it is, including many people who regularly publish test reports (whether it’s to the world at large, or to subscribers to a specific publication.

We can’t possibly cover all aspects of this complex topic in detail here, so I’ll restrict myself to a general overview of the topic as a whole, and then focus on detection testing, which is probably what most readers are really interested in, rightly or wrongly.

2.1 Evaluation

Detection is important, of course: it is, after all, one of the primary functions of a malware-specific product or service, which can be categorized as:

- Malware detection
- Prevention of infection or compromise by malware
- Remediation in the even t of a compromise

However, stellar detection performance is of little use if the product doesn’t meet the needs of the customer in other ways, such as:

- Usability (both at the systems administration level and at the end-user level)
- Configurability
- Adaptability, especially when there's a drastic change in the threat landscape (such as a significant new threat vector)
- Responsiveness to changes in the organizational environment or infrastructure (network changes, hardware and software upgrades, changes in policy or strategy framework)
- Responsiveness or adaptability to business needs: for instance, the impact of security software on host hardware and other applications, and therefore on day-to-day business processes.

Some reviewers try to take some of these factors into account. However, organizations that try to take an approach to procurement that balances technical, operational and business requirements rarely find a comparative review that tries to include the same areas of interest, yet is neither too subjective nor too focused on the assumption of a one-size-fits-all view that applies to all types and size of business.

As there is little in the way of formal objective testing that addresses these issues, it's not surprising that reviewers and their audiences tend to fall back on detection testing as the main criterion for comparative evaluation. It is, after all, a core function, and offers a deceptively simple, apparently objective metric.

2.2 Objective Performance Data

Before we consider how objective detection testing really is, let's consider other possible ways to harvest "hard" objective data. Perhaps the most obvious possible alternative metrics centre round the impact of a tested product on system performance.

There are, in fact, a number of basic functionalities that can be tested fairly objectively. For instance:

- On-demand scanning speed on a clean machine. (That is, passive scanning of a folder, system, or individual files without actually opening files for execution.) This has the advantage that it doesn't even require a malware collection. However, it has to be done properly, which needs reasonable understanding of the technology. For instance, if product X scans all files by default and product Y scans only files with selected file extensions, it is misleading to present a test that doesn't take this difference into an account as an objective speed test. This is because it doesn't take into account the likely differences in detection performance between the two products. Even though this isn't a detection test, detection is an important consideration, since product X may be penalized for scanning speed performance, without reference to the fact that in some scenarios product Y may miss malware that X won't.

  On-access scanning (scanning each file as it's opened for reading or execution) needs a little more setting up, not only to measure a product's performance as it goes through a significant volume of test files, but also to ensure a level playing field between products.

While it’s generally assumed that on-access scanners scan files as they are accessed (as the term suggests), some products actually try to maintain a speed advantage by scanning only if the file has a filename extension recognized by the scanner as denoting an executable *unless* the file is accessed for execution – that is, just reading the file won’t trigger a scan. While it’s sometimes argued that it isn’t necessary to scan a file until it’s executed, this clearly isn’t an approach taken by all vendors or the approach expected by all customers.

- Scanning speed on an infected machine is even more of a can of worms (not to mention Trojans, viruses, bots, adware…) Many of the problems described below that apply to out-and-out detection testing might also apply here (for instance, proper validation and classification of samples), but the issue is complicated further in that the sample set must consist of samples that are known to be detected by all products (otherwise it’s a speed-and-detection test, not a speed test). In addition, considerable care has to be taken to ensure that the configuration of all scanners is equivalent, so as to lessen the risk of bias. Commonly, tests are carried out using “out-of-the-box” (default) configuration. However, anti-malware scanning can be a trade-off, not only between speed and security, but also between speed and other factors such as thoroughness of removal. For example, not so many years ago, there was a product which “disinfected” macro viruses by simply turning off the macro bit, leaving the actual macros intact, and thus generating a potential security hole somewhat similar to disinfecting a folder full of malware by changing the filename extension to .TXT. While we won’t consider the issue of latent malware further here, this is an issue that arises in many contexts in present-day testing. [Execution Context in Anti-Malware Testing, David Harley, EICAR 2009 Conference Proceedings]

- False positive (FP) testing is a useful idea in principle, but a significant FP test set is not a trivial undertaking – just for starters, it needs to be large and “clean” (each test sample must be validated as “innocent”, meaning that all traces of malware infection have been filtered out). This is a problem actually closely analogous to the need to implement and maintain a database of clean applications for white-listing applications, where the idea is to block all unapproved applications rather than to detect all known malicious applications (blacklisting).
- Impact on system resources such as processor load, impact on disk capacity, and memory usage. A particularly useful metric, when properly measured, is commit charge, which may be defined as the “maximum potential pagefile usage (http://en.wikipedia.org/wiki/Commit_charge), or, more simplistically, as the amount of physical and virtual memory allocated to applications (including the antimalware program under test) and the operating system (http://www.webopedia.com/TERM/commit_charge.html). An apparently straightforward approach to measuring this is to measure commit charge before the program runs and then measure the difference during a scan. However, this is not as simple as it sounds: to be useful, it requires the tester to monitor total memory used, the combined limit of memory used, and the peak system memory usage during a scan, and then find a meaningful way to present some kind of mean performance data.

- Loading time at startup, or for specific applications, or for opening, processing and/or executing documents, archives and so on.

Even assuming good faith, there are many approaches to measuring issues like these (passive scanning on idle system, on-access scanning on undemanding system tasks like copying files, or opening and closing them one by one, or processor-intensive tasks (using application scripting, for example). Unfortunately, comparative performance on one type of load test may tell you nothing about performance on a different test.

2.3 Other Test Targets

There are "softer", more subjective data that would be useful to someone evaluating anti-malware products, or has at least been *presumed* to be useful by some reviewers. For example:

- Quality of helpline support
- Usability – now there's a concept that needs some expansion...
- Quality of Documentation
- The quality (or even the attractiveness) of the Interface – it could be argued that if an attractive interface is important, so is the colour of the box...

Unfortunately, some of the reviews that have attempted to address these issues have been so subjective as to be all but useless. In fact, some of them can be approached them from an ergonomic viewpoint, or with regard to HCI criteria (Human/Computer Interaction) with a degree of standardization and objectivity. However, to test these functionalities properly would actually be at least as daunting a task as detection testing is (or should be, if it's done properly). In real life, though, reviews are more about "I like it" or "I don't like it" than informed objective analysis. Furthermore, a comparative test can't be judged by the same criteria as a scientific experiment: it may be reproducible and objective and yet inaccurate, even with no intention to fiddle the figures.

3.0 Detection Testing

What detection functionality *can* be tested? There are, in fact, quite a few ways of testing anti-malware detection, depending on both the type of malware and the type of detection that interests the tester. Of course, to get a useful picture of whole-product capabilities, the customer needs to synthesise the results of many tests (and types of test). However, the resources needed to execute more than the most basic test mean that inevitably, a tester will focus either on doing one type of test at a time in some depth, or on doing a number of tests but at a very superficial level. I'm assuming here that the tester thinks out of the 1990s "on-demand scan of a collection of infected or malicious files" box: however, experience indicates that this is not a safe assumption.

3.1 Detection by malware type

This should not be seen as an all-inclusive taxonomy of malware of course:

- Replicative malware (viruses and worms, primarily)
- Non-replicative malware, falling into the general class of Trojan Horses, for example:
    - Bots
    - Rootkits

    - Spyware
    - Banking Trojans
    - Fake security programs
- "Greyware" such as some categories of adware and "possibly unwanted applications"

Clearly, the days when mainstream programs in this space primarily detected replicative malware are long gone, since the proportion of replicative malware to non-replicative malware has dwindled dramatically. As a result, what we used to call anti-virus (AV) software (and often still do, from sheer force of habit) is now better referred to as anti-malware software.

However, some classes of security software are stronger in some areas than others. Mainstream anti-malware products tend to address the widest range of threats, but programs that specialize in one area such as spyware or adware may be particularly successful in that area. Testing detection of a particular malware class may mislead the test's audience if it doesn't take these differences into account: if a test focuses on rootkits, a program that has exceptional performance in rootkit detection but is less capable in other areas is likely to be of less use to the average consumer than a product that does less well in rootkit detection, but has good detection rates across the entire spectrum of malware.

Greyware poses particular challenges. Security vendors have learned the hard way that software that can't be unequivocally described as malware can lead to protracted disputes, and has led to the use of such terms as "Possibly Unwanted Program" (PUP), "Potentially Unwanted Application" (PUA) and even "Possibly Unwanted Software" (PUS). This problem goes further than nomenclature: more often than not, detection of such applications is not a default setting, as it may be "safer" for the vendor to leave the decision to enable it to the user. [Adware, Spyware and Possibly Unwanted Applications, David Harley, 2008: http://www.eset.com/threat-center/blog/?p=138] If a test doesn't take this into account (or the tester intentionally leaves all packages under test at default settings), this may result in a whole class of sample being ignored by some packages and detected by others. The problem here is that the test has thus become a test of design philosophy, not of detection.

Unfortunately, this is not the only instance in testing where testing with default configurations results in anomalies. Take the common testing scenario where detection is tested by aiming an on-demand scanner at a directory containing multiple samples of malware to see how many samples are detected. This deceptively simple methodology has many potential pitfalls, one of which is encountered when two different products are capable of detecting all the samples in the test set, but can be set to different levels of paranoia. A product that by default scans all files may detect all samples, irrespective of such factors as file extension or file archiving, where the other product by default scans only files with names it recognizes as executable files. Depending on the exact test set and methodology, this can result in serious disadvantage to the second product.

It can work the other way, too. In a speed test, irrespective of whether clean or infected files are used, the second product may appear to run much faster, because it isn't scanning every file in the target directory. The problem isn't restricted to on-demand scanners, either. At least one product exists which gains a speed advantage in tests by selecting its scan targets by filename: in other words, some malware will only be scanned and recognized on execution. Other forms of access (opening for file copy, for instance) will not result in the scanning and recognition of malware. The trade-off between speed and security may be acceptable in many scenarios and to some customers, but if the tester is unaware of (or ignores) it, scores based on raw speed will be misleading.

3.2 WildList Testing

Despite the shrinking proportion of viruses to non-replicative malware, a significant number of tests and product certifications are still largely based on virus detection: to be precise, based on the WildCore collection of viruses reflecting the current WildList. The WildList Organization International (WLO: see http://www.wildlist.org), though now part of the Verizon empire, is a voluntary, collaborative venture originally initiated by researcher Joe Wells. The WildList itself is compiled from samples contributed by reporters who may represent vendor research labs or corporate organizations, or may be independent researchers.

The advantages of using WildCore for testing purposes are that (1) it consists of samples that have been validated both by the reporter who sent them in and by WLO, reducing the risk of false positives, and (2) it consists of malware known to have been “In the Wild” (ItW) recently, though it can certainly be argued that the term has lost much of its significance in an age of short-life, non-replicative malware. Furthermore, it’s a sample set of a manageable size, and because it’s a collection to which nearly all mainstream vendors have access, thus reducing the risk of certain kinds of bias (regional bias, for example).

The disadvantages are that (1) it’s not particularly representative of the totality of current malware, most of which is not replicative (2) there’s usually a fairly long processing time between the acquisition of the sample, forwarding to WLO, and appearance on the next published WildList. Nonetheless, it continues to drive several influential certifications such as ICSA Labs Anti-Virus Certification (https://www.icsalabs.com/icsa/topic.php?tid=dfgdf$gdhkkjk-kkkk) and VB100 (http://www.virusbtn.com/vb100/index).

3.3 Testing by Detection Type

Testing by malware type is, in truth, often of very limited use. Not only does it focus sharply on just one aspect of a single function (important though that function is), it can actually be seriously misleading because it doesn’t take into account the fact that a single product’s detection rate may be different according to a number of factors, including the type of scanner, which module is under test, and execution context. To understand why these factors are important, we need to consider (briefly) the most common types of detection technology. For a more comprehensive view, Peter Szor’s book, despite its age, is probably still the best available resource. [Peter Szor, The Art of Computer Virus Research and Defense, Addison-Wesley, 2005]

- Malware–specific (signature) detection (one signature per instantiation) is the identification of a known malicious program by name. There was a time when one instance of a specific variant – polymorphic malware and corrupted samples apart – looked just like another, if you could discriminate between the infective code and the infected object. Nowadays, however, one name can be applied to a million items of malware that still have the same core code but are continually re-packed or re-obfuscated, using run-time packers and obfuscators. [A Dose by Any Other Name, David Harley & Pierre-Marc Bureau, Virus Bulletin 2008 Conference Proceedings]. Nowadays, many "signatures" are actually generic signatures, where there may be many instantiations per signature. "Many" may range from a few major variants detected with a single signature, to thousands (or more) of closely related variants and sub-variants. An effective generic signature may catch unknown as well as known variants and sub-variants. The term heuristic is sometimes applied to this form of detection: while this is a fairly loose use of the term, it's defensible: even strict signature detection is heuristic to a limited extent [Heuristic Analysis – Detecting Unknown Viruses, David Harley & Andrew Lee, http://www.eset.com/download/whitepapers/HeurAnalysis(Mar2007)Online.pdf]

- Basic or passive heuristics (proactive detection by static analysis) usually involves looking at the code without executing it. This is sometimes considered to be the opposite of behaviour analysis, though in fact, code analysis can and often does analyse the predicted behaviour of a presumed malicious application. [Behavioral detection of malware: from a survey towards an established taxonomy; Gregoire Jacob, Herve Debar, Eric Filiol; Journal of Computer Virology (2008) 4:251-256; Springer]
- Advanced heuristics is often seen as synonymous with behaviour analysis. In fact, passive heuristic analysis varies enormously from one product to another, and passive code analysis may be very sophisticated. Active heuristics is a term sometimes applied to proactive detection by some form of dynamic analysis, where the program under investigation is allowed to execute in an isolated environment to see what it does. Clearly, if the program is not executed, there is no actual (as opposed to predicted) behaviour to analyse.

There are at least two other aspects of detection functionality that we should consider: the execution context of the scanner (as opposed to the execution context of the scanned object). [David Harley, Execution Context in Anti-Malware Testing, EICAR 2009 Conference Proceedings] Mainstream anti-malware products (at least, those with an anti-virus provenance) usually have two main functional modes, usually categorized as on-demand scanning and on-access scanning (sometimes referred to as real-time scanning). [Differences between On-Demand and Whole Product Evaluation: guidelines document by working group of the Anti-Malware Testing Standards Organization, in draft.]

- On-demand scanning (whether of an object or of a whole system, possibly including available network resources such as remote drives and folders) requires the system, the system administrator, or the system user to initiate the scan. Initiation may be carried out either directly as a one-off act action (for instance, to clean an infected system, or to scan a newly-noticed suspicious object), or as a regular, scheduled action. In the latter case, initiation of the scan is by using functionality built into the scanner, or another program such as a shell script using a system utility such as Windows Scheduled Tasks or an external scheduling utility.
- On-access scanning describes the checking an object for infection or malicious content when it is accessed. Access is a term with many shades of meaning, in this context. Not all potentially malicious objects are program files: for example, documents containing inline macros or embedded executables are, nonetheless, primarily data files, so access does not

necessarily entail execution. For example, a Word document read into Wordpad cannot execute any macros it may contain. Even in the case of pure program files, access does not necessarily entail execution. It may be opened for reading only, as when it is read into another application for some form of analysis, or it may be opened for reading and writing, as when a file is moved or copied from one location to another, without being executed.

Individual anti-malware products may include other forms of functionality such as change detection and other forms of white-listing, HIPS (Host Intrusion Prevention System), firewalling and other forms of traffic and file filtering: these functionalities are not, for reasons of space, further addressed in this article, however.

3.4 Dynamic versus Static Testing

We now move from the execution context of a scanner under test, to that of the sample used to test it. As previously indicated, much informal testing is carried out using a very basic passive scanning technique: load the sample collection onto a system and scan it, using the on-demand component. This is a specific instance of static testing. Dynamic testing is defined by the Anti-Malware Testing Standards Organization [“Best Practices for Dynamic Testing”, at http://www.amtso.org/documents/doc_download/7-amtso-best-practices-for-dynamic-testing.html) as testing “where a PC is exposed to a live threat (for example, by attempting to execute the malware] as part of the test.”

This is regarded as a more realistic, accurate test of product efficacy than static testing, and is seen nowadays as the only way to test some anti-malware technologies fairly and accurately, but it’s much more complex and resource-intensive to test dynamically than it is statically. It’s simply impractical for some aspirant testers to carry out useful dynamic testing, even if they are aware of the desirability of doing so.

Since on-access scanning is not only initiated by the execution of programmatic content, testing on-access scanning is not the same as dynamic testing: the scanning may be strictly passive (or static). Similarly, on-demand scanning is not invariably passive: a scanner may check a program while it’s executing, and there may even be a conflict when the on-demand and on-access scanners hit on the same object at the same time, though this is rarely seen nowadays.

Aspirant testers should be aware that on-demand scanning does not necessarily make full use of the scanner’s detection ability: some scanners only execute full dynamic analysis of suspect code on an on-demand scan. For example, while a Windows on-demand scanner may possess full emulation or sandboxing capability to enable full-strength dynamic and/or behaviour analysis, often in the form of active heuristics, but a Linux server version of the same scanner may not have the same virtualization capabilities.

This is not a problem where “like-to-like” comparative testing is carried out, but where scanners are tested irrespective of their hosting environment, for example, the testing process may become seriously biased.

Platform, defensive topology, and operating environment may have a major impact on detection mechanisms used. Server-hosted products such as mail filters may use aggressive heuristics and be more tolerant of false positives, or use class-based detections triggered by use of runtime packers where a desktop product would be more discriminating. Some server products are command-line driven rather than GUI-driven: this is convenient for testers because shell-scripting can facilitate automation, but adds complexity and unpredictability in terms of dynamic detection.

3.5 Sample Validation

Valid testing requires a valid set of samples.

Validation is a more complex concept than it might at first appear: for replicative malware (viruses and worms) validation requires confirmation that the sample actually replicates (if it doesn't, it's an "intended" virus, or a corrupted virus that doesn't have full functionality, or exhibits some other validation problem. For all forms of malware, validity entails some form of confirmation that the sample is intended maliciously and is capable of executing that malicious intent. There is a grey area here, in that there is no consensus within or outside the testing and vendor industries as to how a variety of objects (intendeds, corruptions, objects that are not themselves malicious but are used as part of the process of compromise) should be treated by testers. There is also a corpus of objects (zero-byte files, text files, miscellaneous test files, garbage files) that still turn up in test collections, and that are only there because the sample library isn't properly maintained.

3.6 Testing Heuristic Detection

Testing signature detection is (conceptually, at least) comparatively easy. All it requires is a collection of valid samples of known malware. Testing heuristics and other proactive technologies for the detection of unknown malware is another matter. Commonly, testers attempt to address this by creating new malware or by modifying known malware to create unknown variants. The anti-malware industry, however, detests this approach. One reason for this is that we don't believe that testing should put the public at risk, and there is significant historical precedent for newly-created test samples somehow finding their way into the "real" world.

However, there are practical reasons for avoiding this approach: for instance, many custom samples are simply not valid or fit for purpose: for instance, because they are not capable of executing, or because they are used in testing in a form in which they couldn't possibly exist in the real world.

There is a way in which the proactive detection capabilities (heuristics, behaviour analysis and so on) can be tested by preventing the product from updating its signatures for a set period, then testing its performance against malware that appears during the period during which signatures are not being updated. While there are some technical issues with this approach, it's usually a much more reliable approach to testing than the creation or unnecessary modification of malware.

4.0 The Anti-Malware Testing Standards Organization (AMTSO)

The company for which the author works is a member of AMTSO, and he is active in that organization as a representative of his employer, but does not speak for AMTSO (http://www.amtso.org). However, we do believe that AMTSO represents a major force in the efforts of security vendors, security testers, reviewers and other stakeholders to improve the quality of comparative testing and reviewing across the board. We interpret its aims as being the promotion of:

- Objectivity/Impartiality – no hidden agenda
- Quality/Sound Practice: recognition of the fact that (a) testing in general is a discipline that in itself requires technical knowledge and experience (b) testing of malware requires knowledge of malware and anti-malware technology.
- Relevance/Consistency with avowed testing aims: comparing apples to apples, not apples to oranges (or, even worse, grapes to melons).

Its mission is defined (see http://www.amtso.org/members/2-amtso-charter.html) as to "Improve the objectivity, quality and relevance of anti-malware technology testing," and its charter is defined as to:

- Provide a forum for discussions related to the testing of anti-malware and related products
- Develop and publish objective standards and best practices for testing of anti-malware and related products
- Promote education and awareness of issues related to the testing of anti-malware and related products, and provide tools and resources to aid standards-based testing methodologies
- Provide analysis and review of current and future testing of anti-malware and related products

The organization's first deliverables are already available on the AMTSO web site, and other forthcoming deliverables include a glossary, a document on the issues around creating malware for testing, and another on testing on-demand capability versus testing complete detection capability. There is now also a group which can analyse and evaluate tests – a review of reviews – on request.

4.1 AMTSO Guidelines

The following nine points are the principles embodied in the "Fundamental Principles of Testing" document recently approved by the testers, vendors and other interested parties who make up the Anti-Malware Testing Standards Organization (AMTSO), and available at http://www.amtso.org/documents/cat_view/13-amtso-principles-and-guidelines.html.

While membership of the organization is not cheap, anyone with a commercial interest in fair and accurate detection testing is advised to consider joining the organization and participating in the discussions on AMTSO lists and at its frequent meetings, but anyone with an interest in the field (not just testers and vendors, but customers!) is likely to find it useful to track some of the documentation now being published by the organization.

The Guidelines document does not offer an aspiring tester everything they need to know to start testing, but it's a major step forward to understanding what sound testing techniques entail.

1. Testing must not endanger the public.
2. Testing must be unbiased.

3. Testing should be reasonably open and transparent.
4. The effectiveness and performance of anti-malware products must be measured in a balanced way.
5. Testers must take reasonable care to validate whether test samples or test cases have been accurately classified as malicious, innocent or invalid.
6. Testing methodology must be consistent with the testing purpose.
7. The conclusions of a test must be based on the test results.
8. Test results should be statistically valid.
9. Vendors, testers and publishers must have an active contact point for testing related correspondence.

These guidelines are, of course, expanded upon and interpreted in the original document: the summaries that follow, and should not be seen as in any way authoritative, or representing the views of the Anti-Malware Testing Standards Organization in any official sense. They are simply my own interpretation of the guidelines, though obviously that interpretation is influenced by twenty years in the anti-malware research community.

4.2 Principles Unfurled

1. *Testing must not endanger the public.*

This principle addresses the frequently-cited concern in the anti-malware research community that care should be taken so that samples of malicious software aren't accidentally released "into the wild", increasing the danger to the general public. Since it's almost impossible to carry out some kinds of detection testing accurately using some modern malware in a "closed" environment, this is not a minor issue.

The organization (and the industry as a whole) is also anxious to discourage testers from attempting to create new malware (including completely new malware, new variants, and kit-generated malware). Opinion varies on whether the concealment or obfuscation of known variants using run-time packers and other tools sometimes used by criminals to evade detection of malware is acceptable for research purposes, but is likely to pose problems if used for testing purposes, especially if re-packing or obfuscation results in the creation of samples which not only do not but could not exist in the real world.

The anti-malware research community (by which we don't just mean the vendors) has always opposed the creation of malware for testing purposes on the following grounds:

- The belief that it's unethical to create new malware at all: this view isn't universally held, but a significant number of eminent researchers will not create malware for any purpose. This is not a purely subjective ethical position, however, but has practical implications. Situations have arisen where the tightly controlled generation of malware for the purposes of forestalling likely future attacks backfired dramatically, highlighting additional problems in regard to the creation of upconverted samples for testing. [The problems of wordmacro virus upconversion, Vesselin Bontchev, http://www.sciencedirect.com/science?_ob=ArticleURL&_udi=B6V8G-3WSVS8C-7&_user=10&_rdoc=1&_fmt=&_orig=search&_sort=d&view=c&_acct=C000050221&_version=1&_urlVersion=0&_userid=10&md5=1ab26d9319f7c3c8cff6de244bed88f9, 1999.]

- The belief that it's unnecessary to create totally new malicious artifacts when tens of thousands of new malicious programs are seen every day.
- The fear that malware created for testing purposes (or other purposes such as teaching) may find its way into the wild and increase the risk to the general population. While many people outside the anti-malware community consider this risk overstated [Future Threats, John Aycock & Alana Maurushat, Virus Bulletin Conference 2007 Proceedings; Teaching Viruses and Worms, Bruce Schneier, June 12 2007, http://www.schneier.com/blog/archives/2007/06/teaching_viruse.html], the fact remains that neither dynamic testing nor dynamic analysis are always feasible in a closed environment, even a simulated "Internet in a Box", depending on the type of samples used. While the effects of errors in such testing may not have the same lethal potential as errors in the "real" world with real "biological malware" [http://www.newscientist.com/article/mg20126983.400-bird-flu-mixup-could-have-spelled-disaster.htm] it would be naive to assume that they could not seriously affect many people.
- The sternly practical view that many of the objects created for testing purposes should not be used for that purpose because they don't meet the technical definition of a viral or malicious object. When the ability to detect corrupted files (for example) becomes a measure of malware detection, a test's relevance and accuracy becomes severely compromised. This matters because if a program is recommended or attacked, based on its performance against inappropriate criteria, that is potentially very damaging to the consumer. It is sometimes argued that in some instances, it's not unreasonable to expect a product to detect objects that are intended to have a malicious effect but fail to do so because of corruption, programming errors and so on. This is defensible only if:
    - It's clear from the published terms of reference that the test makes this assumption, even though it's based on a viewpoint that is by no means universally held.
    - Clear distinctions are made between objects that have an indisputable link to malicious action and intent, objects that are not in themselves malicious in intent, and objects that would not and could not actually exist in the real world.

2. *Testing must be unbiased.*

There are many tests where the results clearly reflected a hidden or overt agenda such as the promotion of a favoured product. However, there are also many occasions where bias is introduced unintentionally. For instance, where a favoured product or products is or are used to validate samples, or where no validation is performed at all, so the product that flags the most detections is assumed to be the "best", irrespective of the risk of false positives.

3. *Testing should be reasonably open and transparent.*

Clearly, some publications are not always comfortable with disclosing their methodology. However, a moderately transparent methodology is necessary for the reader to evaluate the reliability of the test. For instance: product versions, patches and updates; source of samples, and how they were validated; product configuration settings; test environment.

4. *The effectiveness and performance of anti-malware products must be measured in a balanced way.*

This is about effective interpretation of raw data. For instance, not prioritizing detection of "suspicious" files over the risk of false positives; not comparing apples to oranges (comparing different products based on a feature set that is more appropriate to product set A than product set B); testing in ways that don't reflect real world user experience. It's often misleading to attempt to summarize product efficacy using a single measurement. For example, a product that flags a very high volume of malicious objects but also misdiagnoses many innocent objects as malicious or at least suspicious is not necessarily a "better" solution than a scanner with a slightly lower detection rate but a much lower false positive rate.

5. *Testers must take reasonable care to validate whether test samples or test cases have been accurately classified as malicious, innocent or invalid.*

The most common form of problematic methodology is insufficient or incompetent validation of samples.

Irrespective of the size of a sample set, the proportion of samples unrecognized as unfit for purpose may invalidate the test results, and, in particular, product ranking in a comparative test. It is incumbent upon the tester to take care to categorize test samples appropriate, and to revalidate samples that may have generated false negative or false positive results. Corrupted and other non-viable samples should be flagged and excluded unless the test specifically addresses these issues. In such a case, it's essential that the tester make it clear that non-viable samples are included in the set, as his audience needs to be aware that he's assuming functionality and design features that neither the vendor nor the customer might consider appropriate.

6. *Testing methodology must be consistent with the testing purpose.*

We sometimes find that there are incompatibilities between the apparent test objective, the methodology used, or the conclusions drawn. Test results may be severely compromised because the test methods aren't the best way to generate accurate data.

7. *The conclusions of a test must be based on the test results.*

This principle directly addresses an issue where the conclusions drawn by the tester are not supported by the actual data from which the conclusions were alleged to have been drawn. This can happen, for instance, where the conclusion owes more to the preconceptions of the tester than to scrupulous analysis of the test data.

8. *Test results should be statistically valid*.

The accuracy of a test's results is often directly related to the size of the sample set proportional to the whole population of known potential samples within the class of malware that the test set represents. For instance, one hundred samples of miscellaneous Windows malware is laughably small, given the very high volumes of potential samples, whereas one hundred unique samples of

Macintosh malware might represent close to the entire population of known malware for that platform. Small sample sets rarely provide good test data, though the quality of validation may improve that scenario.. However, the nature of the malware problem is such that there is a degree of unreliability in almost any test that would not be acceptable in other types of testing.

Essentially, the test set should be both large enough to be significant and properly representative of the population of malware it represents.

9. *Vendors, testers and publishers must have an active contact point for testing-related correspondence.*

Much of the acrimony and distrust generated around testing issues would be dissipated if there was better communication between testers and vendors. The vendor community appreciates information about tests and results, and it can benefit publishers and testers to exchange product information.

5. Conclusion

What is testing for? Presumably, it's intended to guide people, so that they can make the best possible choice (and what people often want is to have the choice made *for* them, not to have enough information in front of them to allow them to reach their own decision). But what do we mean by "best"? Testers and reviewers are not invariably honest and impartial, or even competent. Magazine reviews are there to give the customer something they want, and do that irrespective of how knowledgeable they are about the topic, sometimes to the extreme displeasure of the industry.

Blogs don't always have a commercial agenda, of course, but they usually aim to attract subscribers, and a blogger, like any journalist, may know a lot or virtually nothing about the topic. The tester or reviewer may be influenced by factors that militate against the objectivity of a test, consciously or not.

Leaving aside the question of deliberate bias and malpractice, such factors may include preconceptions about how a product type should work, so that the interpretation of the results is biased towards products that meet those preconceptions.

In principle, anyone can undertake a comparative test: there is no certification scheme for testers at the moment. Furthermore, while there are ISO and other standards to which a good commercial testing group will attempt to conform, they're not mandatory. Even these can be misused: certification of a lab's Quality Assurance processes is not the same as validating its testing methodology.

Testers and reviewers do want (ideally) to give their audience the information they want, though they may have secondary aims or hidden agendas. We would argue that:

- Testers and testing groups are accountable to their audience. That doesn't mean that they're accountable to the anti-malware industry, of course, but that industry does know more about malware and anti-malware technology than virtually anyone else, and deserves closer attention to its technical and ethical concerns.

- Software testing in general is a discipline requiring skill and experience
- In-depth anti-malware software testing requires extensive knowledge of malware and anti-malware technology

It's for these reasons that it's sometimes asserted that there should be a way to certify testers (that is, those who certify products as effective, or who assess or compare product performance – especially detection – across a range of products :

- To qualify for sample sharing, for example to gain access to a standard test like WildCore.
- To prove competence knowledge, experience and ethical fitness to test.
- To prove conformance with testing standards such as the AMTSO guidelines and other relevant standards such as:
    - ISO 17024 – assessing and certifying personnel
    - ISO 9001 – quality management
    - ISO 17025 – requirements for competence of testing and calibration laboratories

However, there is no formal certifying body currently. Among the bodies that could contribute to a certification framework are academia, security certification agencies, standards bodies, other industry stakeholders like the WildList Organization, researchers and customers like those represented in AVIEN (Anti-Virus Information Exchange Network).

Here, as they say, is the bottom line:

- You can't take impartiality or competence for granted, just because you read it in a newsgroup, on a web site, or a mailing list. On the Internet, no-one knows that you're a dog, let alone a wolf in sheep's clothing. In fact, you should mistrust anyone who asserts that anyone can run a test.
- You can't trust a test methodology you don't have enough information on to evaluate it properly. Even if the information is there, you can't evaluate it if you don't understand the methodology and the technology it's designed to test.
- We recommend that you look for (and support!) the evolution of objective, reasonably standardized testing procedures. Not that standardization should stifle innovation, but anarchy has little or no place in the world of objective testing.

The anti-malware industry has often been accused of using ethical objections as a cloak for its own malpractice and incompetence, among other things: as an industry, we are probably slightly less popular than traffic wardens and tax officials. ["I'm OK, You're Not OK": David Harley, Virus Bulletin, November, 2006.] Nonetheless, we would contend that to offer misleading advice to review readers looking for guidance is, nonetheless, as serious an ethical breach as you're likely to find in this field. Review audiences and potential customers of reviewed products are entitled to expect that a tester should uphold reasonable standards of truth, accuracy and ethical behaviour. Testers and reviewers, however, are equally entitled to expect cooperation and, where appropriate, guidance from the anti-malware industry. AMTSO seems a very positive step towards meeting the expectations of all parties.

References

Andrew Lee& David Harley, “Antimalware Evaluation and Testing”, in “The AVIEN Malware Defense Guide for the Enterprise”, Edited by David Harley, Syngress 2007

Adware, Spyware and Possibly Unwanted Applications, David Harley, 2008: http://www.eset.com/threat-center/blog/?p=138

Peter Szor, The Art of Computer Virus Research and Defense, Addison-Wesley, 2005

A Dose by Any Other Name, David Harley & Pierre-Marc Bureau, Virus Bulletin 2008 Conference Proceedings

Behavioral detection of malware: from a survey towards an established taxonomy; Gregoire Jacob, Herve Debar, Eric Filiol; Journal of Computer Virology (2008) 4:251-256; Springer

David Harley, Execution Context in Anti-Malware Testing, EICAR 2009 Conference Proceedings

Differences between On-Demand and Whole Product Evaluation: guidelines document by working group of the Anti-Malware Testing Standards Organization, in draft

“Best Practices for Dynamic Testing”, at http://www.amtso.org/documents/doc_download/7-amtso-best-practices-for-dynamic-testing.htm

The problems of wordmacro virus upconversion, Vesselin Bontchev, http://www.sciencedirect.com/science?_ob=ArticleURL&_udi=B6V8G-3WSVS8C-7&_user=10&_rdoc=1&_fmt=&_orig=search&_sort=d&view=c&_acct=C000050221&_version=1&_urlVersion=0&_userid=10&md5=1ab26d9319f7c3c8cff6de244bed88f9, 1999.

Future Threats, John Aycock & Alana Maurushat, Virus Bulletin Conference 2007 Proceedings;

Teaching Viruses and Worms, Bruce Schneier, June 12 2007, http://www.schneier.com/blog/archives/2007/06/teaching_viruse.html

http://www.newscientist.com/article/mg20126983.400-bird-flu-mixup-could-have-spelled-disaster.htm

“I’m OK, You’re Not OK”: David Harley, Virus Bulletin, November, 2006.