Heuristic Analysis – Detecting Unknown Viruses

Anti-virus does much more than reactively detect known viruses; it proactively scans for unknown ones too.

So, how do scanners really work?

David Harley
Security Author and Consultant

and

Andrew Lee
Chief Research Officer
ESET LLC

SMALL BLUE-GREEN WORLD
8 Clay Hill House, Wey Hill, Haslemere, SURREY GU27 1DA
Telephone: +44 7813 346129 Fax: 44 8717 143451
http://smallblue-greenworld.co.uk

www.eset.com

NOD 32
antivirus system

About the Authors

David Harley

David Harley has been researching and writing about malicious software and other security issues since the end of the 1980s. From 2001 to 2006 he worked in the UK's National Health Service as a National Infrastructure Security Manager, where he specialized in the management of malicious software and all forms of email abuse, as well as running the Threat Assessment Center. Since April 2006 he has been working as an independent author and technology security consultant.

David's first major book was "Viruses Revealed" (Harley, Slade, Gattiker), Osborne's comprehensive guide to computer virus protection. He has contributed chapters and editing to many other books on security and education, as well as a multitude of articles and conference papers. His latest writing project is as technical editor and major contributor to "The AVIEN Guide to Malware Defense in the Enterprise", to be published by Syngress in 2007.

Andrew Lee

Andrew Lee, CISSP, is Chief Research Officer at ESET LLC. He was a founding member of the Anti-Virus Information Exchange Network (AVIEN) and its sister group AVIEWS (AVIEN Information & Early Warning System). He is a member of AVAR (The Association of Anti-virus Asia Researchers) and a reporter for the WildList International Organization, a group that maintains a list of computer viruses which are still "in the wild." Before joining ESET, he spent his time managing malware defenses as a senior security administrator for a large government organization in the UK.

Andrew is the author of numerous articles on malware issues, and is a frequent speaker at conferences and events including AVAR, EICAR, and Virus Bulletin.

Table of Contents

Introduction

"It ain't what you don't know that kills you, it's what you know that just ain't so."

Some of the most persistent myths in computing relate to virus and anti-virus (AV) technology. The widely-held belief that AV software can only detect specific, known viruses has been around since the early days of AV research. It wasn't altogether true then; some of the first AV programs weren't intended to detect specific viruses, but rather to detect or block virus-like behavior, or suspicious changes in files. And, it's definitely not true now.

Commercial AV systems supplement signature scanning with a variety of more generic approaches, which are often grouped together under the banner of heuristic analysis. Furthermore, most modern AV products are capable of detecting a wide range of malicious software (malware is a contraction of the words "malicious" and "software"), not just viruses. These may be combined with other security technologies such as the detection of spam and phishing messages.

The aim of this paper is to reduce some of the confusion around the workings of AV technology, and to clarify what is realistic to expect from AV protection, particularly heuristic analysis.

The specifics of heuristic scanning are discussed in some detail. For the moment we'll simply describe heuristic analysis as a method of estimating the probability that a program that hasn't been identified as known malware is, nevertheless, viral or malicious.

Watching the Detectives

What does an AV program detect? Quite a lot as it happens, including some items that aren't technically viruses. Most of what we see referred to as viruses might be better described as malware. The irony is that many specialist detection products (i.e. for detecting spyware or Trojans) are marketed as being necessary because AV only detects viruses.

Most of what we see referred to as viruses might better be described as malware

In fact, commercial AV catches a far wider range of malware than most of these specialist services. A specialist program may detect more threats within its own specialty, but this depends not only on the program's ability to catch specific threats and threat types, but also on other factors such as:

- The program's generic detection capabilities
- The criteria used to differentiate between malware variants
- The sample sharing mechanisms between vendors (AV vendors have particularly effective and well-established ways of doing this, compared to vendors in other areas of malware detection.)

The following sections consider three major types of malware. A complete taxonomy of all malware would be out of scope for this paper.

Viruses

It's certainly reasonable to expect AV software to detect viruses, and it is partly because AV has been so successful at detection over the years, that its capacity for detecting other types of malware has been underestimated.

While there are many definitions of virus, a definition accepted by most malware researchers is "a computer program that can infect other computer programs by modifying them in such a way as to include a (possibly evolved) copy of itself" [1, 2].

This definition covers many types of virus, including:

- Boot sector and/or partition sector infectors
- File infectors (parasitic viruses)
- Multipartite viruses
- Macro and script viruses

It is partly because AV has been pretty successful over the years, that its capacity for detecting other types of malware has been underestimated.

While some of these virus types are rarely seen today (for example boot sector and partition sector infectors), AV programs generally detect all known viruses for the platform on which they are found (and sometimes for other platforms). In general, they're also pretty good at detecting new and unknown "true" viruses heuristically.

Worms

The AV industry has never quite reached consensus on whether worms are, as Cohen stated, a "special case of virus" [1], but whatever the case, AV software normally detects them anyway.

There are at least as many definitions of worm as there are of virus, but most AV researchers define a worm as a program that replicates non-parasitically, i.e. without attaching itself to a host file. Mass mailers could be described as a special type of worm. Most AV companies describe this type of email-borne malware as a worm, but some mailers and mass mailers have the characteristics of a "pure" virus (Melissa, for example, was actually a pure virus, a macro virus that spread like a worm, while W32/Magistr was a file infector).

Here too, vendors have a pretty good handle on the detection of new variants. New mass mailers, for example are usually flagged by messaging security providers and systems almost as soon as they appear.

Non-replicative Malware

It follows from the above definitions that if a malicious program doesn't replicate, it can't be a virus or worm. But that doesn't mean AV software can't detect it, or that it isn't damaging.

Keep in mind that even when vendors used to protest at the detection of non-replicative objects because they weren't viruses, some non-replicative objects (some of them not even executable programs, let alone malicious) were still detected and flagged. [3] For example:

> Many non-replicative objects have circulated for years in poorly maintained virus collections.

- Intendeds (viruses that fail to replicate) and corruptions
- Garbage files
- Virus-related but non-viral programs such as germs, droppers, and virus generators
- Legitimate test programs such as an EICAR test file [4]

Many non-replicative objects have circulated for years in poorly maintained virus collections that have been used by some reviewers to test AV software. Most vendors gave up protesting long ago and added definitions (signatures) for these objects to their databases, in the hope of avoiding being penalized for not detecting them. Unfortunately, the increasing sophistication of heuristic scanners has barely kept pace with the ability of AV testers to find new and not always appropriate ways of testing. Later in this paper we will briefly consider technically acceptable ways of testing a product's heuristic capabilities.

The best-known non-replicative malware is the Trojan Horse (or Trojan for short). A Trojan is "a program that claims to perform some desirable or necessary function, and might even do so, but also performs some function or functions that the individual who runs the program would not expect and would not want." [5] This covers a range of specialized malware, including:

- Droppers
- Keyloggers
- Destructive Trojans
- Downloaders
- Spyware
- Adware
- Rootkits and stealthkits
- Joke programs (some)
- Zombies (bots, Remote Access Trojans, DDoS agents, and so forth)

Replicative malware such as viruses can also sometimes be described as Trojans (or as Trojanized or Trojaned, implying that a formerly legitimate program has been subverted, altered or replaced to make it in some way damaging), though most people are likely to find that use more confusing than helpful. Detection of all versions of non-replicative malware is even less attainable than the detection of all forms of viruses, since a far wider range of functions has to be tested for than the mere ability to replicate.

Much of the debate on what is or is not a Trojan (or malicious) rests not on function, but rather on intent. For example, a keylogger is not a Trojan if it has been legitimately or consensually installed, and yet the function is identical. This leads to detection problems, because computers are less able than humans to determine intent.

Spyware and adware – perhaps due to the heightened media interest, and the products available exclusively for their detection – have recently been separated into their own subclasses of malware. The distinction here, though, is mostly unnecessary, although it could be (and often is) argued that adware in particular is not always malware. However, the same argument can be made for almost all of the items in this list, in that it's not what the program does that makes it malicious; it's the gap between the bad intentions of the programmer and the expectation of the program user.

What Does Heuristic Really Mean?

"Heuristic" refers to the act or process of finding or discovering. The Oxford English Dictionary defines heuristic as "enabling a person to discover or learn something for themselves" or (in the computing context) "proceeding to a solution by trial and error or by rules that are only loosely defined". [6] The Merriam-Webster Dictionary defines it as "an aid to learning, discovery, or problem-solving by experimental and especially trial-and-error methods" or (again, in the context of computing) "relating to exploratory problem-solving techniques that utilize self-educating techniques (as the evaluation of feedback) to improve performance." [7]

Heuristic programming is usually regarded as an application of artificial intelligence, and as a tool for problem solving. Heuristic programming, as used in expert systems, builds on rules drawn from experience, and the answers generated by such a system get better as the system "learns" by further experience, and augments its knowledge base.

As it is used in the management of malware (and indeed spam and related nuisances), heuristic analysis, though closely related to these elements of trial-and-error and learning by experience, also has a more restricted meaning. Heuristic analysis uses a rule-based approach to diagnosing a potentially-offending file (or message, in the case of spam analysis). As the analyzer engine works through its rule-base, checking the message against criteria that indicate possible malware, it assigns score points when it locates a match. If the score meets or exceeds a threshold score [8], the file is flagged as suspicious (or potentially malicious or spammy) and processed accordingly.

In a sense, heuristic anti-malware attempts to apply the processes of human analysis to an object. In the same way that a human malware analyst would try to determine the process of a given program and its actions, heuristic analysis performs the same intelligent decision-making process, effectively acting as a virtual malware researcher. As the human malware analyst learns more from and about emerging threats he or she can apply that knowledge to the heuristic analyzer through programming, and improve future detection rates.

Heuristic programming has a dual role in AV performance: speed and detection. In fact, the term heuristic is applied in other areas of science [9] in a very similar sense; aiming to improve performance (especially speed of throughput) through a "good enough" result rather than the most exact result. As the total number of known viruses has increased, so has the need to improve detection speed. Otherwise the increased time needed to scan for an ever-increasing number of malicious programs would make the system effectively unusable.

Heuristic analysis uses a rule-based approach to diagnosing a potentially-offending file (or message, in the case of spam analysis).

Despite the much-improved performance of some contemporary heuristic engines, there is a danger that the impact of heuristic (and even non-heuristic) scanning may be seen as outweighing the advantages of improved detection. There is a common belief that heuristic scanners are generally slower than static scanners, but at a certain point of sophistication this ceases to be true.

As the total number of known viruses has increased, so has the need to improve detection speed.

Even early heuristic scanners using simple pattern detection benefited from optimization techniques that searched only the parts of an object where a given virus could be expected to be found. (A simple example - there's no point in scanning an entire file for a virus signature, if that virus always stores its core code at the beginning or end of an infected file.) This reduces scanning overhead and lessens the risk of a false positive.

The inappropriate detection of a viral signature in a place where the virus would never be found in normal circumstances is not only a side effect of poor detection methodology, but a symptom of poorly designed detection testing. For instance, some testers have attempted to test the capabilities of an AV program by inserting virus code randomly into a file or other infectible object. Similarly, a particular kind of object such as a file or boot sector can be selectively scanned for only those types of malware that can realistically be expected to be found in that object, a process sometimes described as "filtering". After all, there's no reason to look for macro virus code in a boot sector.

However, correct identification of a file type is not concrete proof of an uncontaminated file. For example, Microsoft Word document files containing embedded malicious executables have long been a major attack vector for information theft and industrial espionage. Similarly, malware authors are constantly in search of attacks where an object not normally capable of executing code can be made to do so for example, by modifying the runtime environment. W32/Perrun, for example, appended itself to .JPG and .TXT files, but could not actually run unless specific changes were made in the operating environment to allow the Perrun code to be extracted and run.

Signature Scanning

Signature scanning refers to fairly straightforward pattern matching algorithms, searching for a sequence of bytes (a string), characteristic of each virus or variant in the scanner's definitions database, but one that isn't likely to occur by accident in an uninfected file. Some AV researchers have tried to discourage [2] the use of the signature scanning description in favor of "search string" or "scan string", but that seems pointless when even AV companies routinely use the expression.

In fact, many viruses cannot be identified by searching for a static string.

An objection to the term is that it perpetuates an antiquated notion of the workings of scanners, though the same argument could also be applied to the alternative terms.

The real difficulties with the use of the term “signature scanning” are that it:

- Perpetuates the myth that it is the only kind of detection performed by AV scanners. In fact, many viruses cannot be identified by searching just for a static string.
- Suggests that there is a single sequence of bytes in every virus that is used by all scanners to identify it. In fact, different scanners may use very different search strings (and algorithms) to detect the same virus.

Some sources [10] have confused the issue further by giving the impression that scanners look for simple text strings rather than byte sequences. Such a method is generally unreliable, totally ineffective with many types of malware, and programmatically inefficient. It is also easily exploitable by a virus writer – or, in fact, anyone capable of editing a file – and dangerous in its potential for generating numerous false positives.

The advent of complex polymorphic viruses actually killed off some scanners that were unable to move to more advanced detection techniques.

Wildcards and UNIX-like regular expressions allow more flexibility in string searching. Instead of looking for a static string (a fixed sequence of bytes), the scanner recognizes a virus-associated string even when other bytes or byte sequences (noise bytes) are interpolated between string elements. A simple example of a noise byte is the insertion of a NOP (No Operation) instruction, which performs no function except to take up processing time without performing an actual operation.

These enhancements to basic string-scanning enable detection of some encrypted and polymorphic viruses [8]. However, even with this kind of enhancement, string scanning isn’t particularly efficient when it comes to scanning for multiple viruses, and the advent of complex polymorphic viruses actually killed off some scanners that were unable to move to more advanced detection techniques [8, 11].

Algorithmic virus-specific scanning in current AV technology is often based on interpreted code run inside a virtual machine. Virtualization and emulation may, for example, be used to remove incidental or intended obfuscation such as packing, compression or encryption. Once the file is de-obfuscated, it can then be analyzed algorithmically – or heuristically – by an AV scanning process.

Virtual machines also play a major part in the implementation of heuristic analysis, and can be very successful, despite the many problems associated with emulating an environment as complex as a modern Windows™ environment [12]. (However, it needs to be understood that emulation cannot be perfect, and the latency penalty (the increased processing time) can be considerable, and varies according to the particular file being tested.)

The Opposite of Heuristics

The myth that commercial AV can only detect known instances, variants and sub-variants of known malware is, perhaps, less widespread than it was. However, it has been partly supplanted by the lesser myth that virus-specific scanners and heuristic scanners are two completely different types of scanners. In fact, heuristic analysis as we know it has been in use for over a decade, but heuristic techniques to optimize virus control have been used for much longer in "known virus" scanners. They have also had a place in related countermeasures such as behavior blockers and monitors, and integrity checkers.

In a sense, the opposite of heuristic analysis in AV is not signature scanning but algorithmic scanning, of which signature scanning is a special case.

Algorithmic scanning, like other forms of algorithmic coding, is based on mathematically provable procedures [13]. What is referred to in the industry as algorithmic scanning is normally understood to be based on an algorithm (other than simply searching for a static string – a fixed sequence of bytes) which is specific to the virus it is intended to detect.

In real life of course, heuristic analysis as described above is also considered algorithmic in the more general sense. However, the use of the term algorithmic in the specialized, virus-specific (and therefore somewhat misleading) sense has become too widely used within the industry [12] to ignore. Heuristics are normally characterized as using a specific scoring algorithm that determines the likelihood of the scanned object being malicious, rather than by the unequivocal identification of a specific malicious program.

> In a sense, the opposite of heuristic analysis in AV is not signature scanning, but algorithmic scanning, of which signature scanning is a special case.

Generic Anti-Virus

Heuristic analysis is often considered a generic AV detection mechanism, not a virus-specific detection mechanism. What is not always considered is that the converse is also true; generic solutions use heuristic rule-sets as part of the diagnostic process.

For instance:

- Mail gateway filters use rules to specify what file types and file names are permitted as attachments. Such filters are very good at countering obvious threats such as files with extensions like .LNK or .JPG, and .EXE, but can be rather inflexible in their rejection of whole classes of executable files. [1] Some filters use more advanced techniques, such as checking that the headers of the file scanned match the filename extension. This can significantly reduce the risk of false positives (and false negatives).

[1] Why are these obvious threats? In the first case, because the .LNK suffix denotes a program shortcut, which doesn't usually make sense as an email attachment because there is no direct link between the shortcut and the program to which it should be linked: however, a shortcut file in an email attachment is often simply a Windows executable file, renamed to evade filters intended to block executable attachments. In the second case, the double extension suggests an attempt to pass off an executable file as a non-executable (graphics) file, a common virus writer's trick.

Generic solutions use heuristic rule-sets as part of the diagnostic process.

- Change detectors use the rule that if an object's characteristics have changed, it should be treated as suspicious. Since there are many contexts in which a binary can legitimately change its checksum (as in self-modifying code, recompiled code, reconfiguration, run-time compression, a patched or updated program), such a crude change detection criterion (i.e., the file has changed, so it must be infected) can exhibit a high false positive rate. However, change detection can work well in conjunction with virus-specific scanning. A well-proven technique is to compare an object to its checksum and run a full scan on it only if a previously calculated checksum has changed, reducing the time it takes to process a file that hasn't changed. This is why an initial scan of a system may take longer than subsequent scans with some AV programs.
- Behavior monitors and blockers, which evaluate and act upon the way applications behave, were among the earliest forms of AV software. This approach dovetails nicely with heuristics, which can enhance behavior blocking performance and actually reduce false positives. Classic AV behavior monitoring tends to check for two types of code behavior: replication and potential damage.
  - — Replicative code, by definition, strongly suggests the presence of a virus (or worm, depending on the type of code and the definition you favor). This approach has an advantage in that system calls suggesting replicative code are comparatively easy to programmatically identify, especially where the code isn't significantly obfuscated. It is, however, easier to identify a virus that replicates by writing a straight copy of itself rather than an evolved copy (i.e. a non-polymorphic virus).
  - — Potentially damaging code reflects the likelihood of a malicious payload. This approach is ineffective where there is no payload, or where the payload is not obviously damaging. Some forms of damage, such as file deletion, are easier to programmatically detect than others, such as the unwanted and potentially embarrassing display of offensive messages or images. On the other hand, successful detection by payload has an advantage when it comes to detecting non-replicative malware (such as Trojans and other non-viral programs). There is a need for caution though. For instance, deleting a file is by itself an unreliable indicator of malice, since many programs routinely and legitimately delete or overwrite files such as obsolete configuration or data files.

I'm Absolutely Positive

Virus identification is a balance between two imperatives: the avoidance of false negatives (failure to detect an infection where one exists) and false positives (detection of a virus where none exists). As demonstrated by a cluster of false positive problems in several major scanners in the first few months of 2006, advances in the optimization of scanner technology have not eliminated the risk of false positives.

> Virus identification is a balance between two imperatives: the avoidance of false negatives (the scanner fails to detect an infection) and false positives (the scanner detects a virus where none exists).

Elimination of false positives is not always possible using heuristics, which by definition entail a degree of trial and error. As discussed earlier, the aim of heuristic programming is less to produce the "perfect" result than to produce a consistently "good enough" result. So what is the problem?

The "safest" way to identify a known virus is to check for the presence of every byte of virus code that should be present in an infected object, by generating a checksum of every constant bit in the virus body. This process is often referred to as "exact identification."

Identification is a measurement of the ability of AV software to detect and recognize a virus sample as a specific virus or variant. Exact identification therefore denotes a level of precision whereby every constant byte of virus code is taken into account. While it sounds desirable for this precision to be applied to every virus scan, this is rarely done in the real world, because of the potential impact on scanning time and system resources, and because this level of detail is not often necessary.

The term "almost exact identification" is applied if the identification "is only good enough to ensure that an attempt to remove the virus will not result in damage to the host object by the use of an inappropriate disinfection method." [2] Detection and removal do not always pose the same problems. Some AV companies have long advocated that infected program binaries should be replaced rather than cleaned, preferring to concentrate on detection. There are also scenarios (rootkits and stealthkits are good examples) where the substitution of a Trojanized program for a legitimate program means that security software can only delete, not clean. In such a case it's usually necessary for the administrator or user to restore the legitimate program; automatic restoration may not be an option, or even safe.

Malware movement over past several years has been away from the classic parasitic infection of files, to the manipulation of the operating environment (for instance, modification of the registry). This can make it much harder to remove all traces of malware once it has taken hold. Incomplete (or incorrect) removal can leave the system damaged or even unusable, sometimes requiring radical measures, such as the reinstallation of the operating system and application software, and restoration of data from backups.

However, where malware is detected proactively (i.e. before it has the opportunity to install on the target system) by heuristic or generic methods, this problem does not generally arise, unless the malicious (viral or Trojanized) object is needed in a non-infectious form (as, for example, when the object contains data).

"Generic detection" is a term applied when the scanner looks for a number of known variants, using a search string that can detect all of the variants. While it may detect a currently unknown variant in which the same search string can be found, it's only a heuristic detection if it involves the use of a scoring mechanism. Otherwise it's really a special case of virus-specific detection. Some systems use a hybrid approach, where a scoring system is added to the generic detection capabilities to give a probability of the variance or family membership with differing degrees of certainty. For instance, if the similarity is close enough, the scanner may report "a variant of x," or if less sure, it may report "probably a variant of x".

> "Generic detection" is a term applied when the scanner looks for a number of known variants, using a search string that can be used to detect all those variants.

Sensitivity and Misdiagnosis

Accuracy in heuristic analysis depends on how aggressively the scoring criteria are set. If the target malware is new to the scanner, the accuracy of the analyzer output is not dependent on a simple binary decision (either "yes, it's a known virus called XXX" or "no, it's not known malware"). Rather, the forcefulness of its response lies on a threshold continuum from high (keeping the number of false positives as low as possible) to low (detecting as many new viruses as possible). An aggressive response prioritizes detection of possible viruses over the risk of false positives, where a non-aggressive response is more appropriate where the adverse impact of false alarms is considered unacceptable.

It's not unusual for a product to offer a stark choice between a default setting (heuristics off), or a setting with heuristics. (Since we have already pointed out that all scanners are to some extent heuristic, perhaps it would be more accurate to refer to the default setting as having basic heuristics enabled.) Some vendors also distinguish between passive and active heuristics. In both cases, code is scanned for suspicious characteristics, but in active mode, the scanner uses an emulator environment to execute and trace the code. In passive mode, it simply statically inspects the code.

> The forcefulness of its response lies on a threshold continuum from high (keeping the number of false positives as low as possible) to low (detecting as many new viruses as possible).

One way of looking at how scanner technology maps to the threshold continuum might be along the following lines:

Threshold Level	Corresponding Level of Heuristic
Highest	Exact (or near-exact) identification only; Heuristics are not used, or kept to a minimum.
Normal	Known virus detection using algorithmic scanning and emulation as appropriate, as well as exact (or near exact) identification where needed. Probably with generic signatures to identify fairly close variants.
Heuristic mode	Medium heuristic level, enhanced detection; fairly low risk from false positives, use of passive analysis rather than emulator-based heuristics.
Lowest	Highest (advanced or most sensitive) heuristics, including some form of emulation. High proportion of new malware detected, but risk of false positives is increased.

Not all scanners have all these levels of sensitivity, nor do they allow thresholds to be set or reconfigured manually, and those that do support levels of sensitivity may not document them. It should also be emphasized that some form of emulation could be in use anywhere on the above continuum.

Vendors who disable their advanced heuristics by default may not only be trying to reduce the risk of false positives, they may be actually trying to improve the product's perceived speed. All levels of heuristic analysis add processing overhead to scanning time, and for some products the slower performance can be all too obvious. However, as we mentioned earlier, even as the number of known malicious objects increases, with well implemented coding routines on today's powerful computers, the impact can be reduced to a manageable level. In fact, there is a great degree of variability in terms of speed performance degradation between scanners from different vendors. A properly implemented heuristic engine should only have a minimal impact on system performance.

All levels of heuristic analysis add processing overhead to scanning time, and for some products, the slower performance can be all too obvious.

Heuristic sensitivity is not just a technical issue related to the accuracy of diagnosing the presence of a previously unknown virus. It's also a psychosocial issue; how should we flag a possible virus to the end-user, and what should we advise them to do?

The way a possible virus is flagged tells the customer a great deal about the AV vendor. Some products are cautious, using messages that say, in effect, that it could be a variant of Virus X, but they're not completely sure. This eliminates the vendor's risk of false positives, by leaving the final diagnosis and choice of action to the customer.

In reality, most customers would prefer that the diagnosis by made by the scanner. Users may feel uncomfortable with the possibility that the software could be wrong, which could suggest that the technology is less reliable than it really is.

Others vendors display a more impressively detailed message that says something like "XXXX malware detected and blocked", or "W32/nastybackdoortrojan detected and removed". That sounds great, and the customer may be duly grateful that malware has been identified and neutralized, but may not know initially that these names are simply generic names that indicate a heuristic detection of possible malware, and not indicative of a specific virus.

Unfortunately, there are no reliable statistics to indicate how many legitimate programs, emails, etc. have been maligned because of an overconfident scanner.

Some vendors advise that advanced heuristics should only be enabled in contexts where the presence of new malware is suspected or most likely to be found, on email gateway scanners, for instance. This reduces the confusion caused by the risk of false positives at the desktop, but increases the risk of false negatives where perimeter scanning fails.

Testing Issues

Testing virus scanners for detection performance has always been a contentious issue [14], and only a very few testers and testing bodies are recognized as competent in this area by other members of the AV research community.

The testing organizations generally considered to be competent in this area include:

- AV Comparatives (http://www.av-comparatives.org/)
- AV-Test.org (http://www.av-test.org/)
- ICSA Labs (http://www.icsalabs.com/)
- SC Magazine/West Coast Labs (http://www.westcoastlabs.org/)
- Virus Bulletin (http://www.virusbtn.com/)
- Virus Research Unit, University of Tampere (http://www.uta.fi/laitokset/virus)
- Virus Test Center, University of Hamburg (http://agn-www.informatik.uni-hamburg.de/vtc/naveng.htm.)

(Note that the last two organizations have not been very active in testing recently.)

Unlike testers with no links to the AV research community, these organizations are generally

> Only a very few testers and testing bodies are recognized as highly-proficient in this area by other members of the AV research community.

trusted by that community – though not necessarily by all members of that community – to test competently, safely, and ethically, while remaining independent. This trusted status means they often have access to authenticated virus samples such as those collected, tested and authenticated by the WildList International Organization (http://www.wildlist.org/), a group of collaborating researchers representing most of the major AV vendors and a number of large corporations and educational institutions.

The AV community argues that most other tests performed by those outside the group of industry-sanctioned testers are potentially invalid or otherwise inappropriate because:

- Tester competence cannot be assumed, and therefore, neither can:
  - The appropriateness of the testing methodology
  - Adherence to safe practice, industry ethical codes and standards

Because of these issues, members of the AV research community cannot ethically share samples with untrusted testers. Therefore, the provenance and authenticity of the samples against which the products are tested cannot be assumed. Often, testers who are unable to access AV community sample pools try to substitute samples taken from virus exchange web sites and other (potentially dubious) resources which may contain all sorts of non-viral samples (garbage files, intendeds, corrupted samples and so forth). Some of these issues can be overcome if the reviewing organization outsources the testing to an accepted organization. (For instance, AV-Test performs several types of testing for magazine reviews.)

Curiously enough, these difficulties have contributed to (but not caused) a situation where testers, concerned about the effectiveness of a certain scanner against unknown viruses, were testing via heuristics even before the technology acquired the heuristic label and its 21st century capabilities, by generating variants. Unfortunately, this typically involved the use of unreliable virus generators, irrelevant virus simulators, random placement or masking of virus code and text strings, and so on [15].

Of course, testing the heuristic capabilities of a scanner is a perfectly valid objective (especially now that scanners have heuristic capabilities). However, it is as important for such a test to be carried out competently and safely as it is for testing known virus detection. In the absence of a competently administered baseline test set, there is no guarantee that scanners are being tested against valid, working viruses. Testers whose competence is already questionable because of lack of direct interface with the AV research community, create further difficulties for themselves,

> In the absence of a competently administered baseline test set, there is no guarantee that scanners are being tested against valid, working viruses.

and for those who rely on their testing, if they don't publish information on their testing methodology, especially sample validation.

By validation we mean whether the code under test is actually malicious – i.e. a virus must have the ability to replicate, worms must be able to spread correctly and so on. Often, when testers perform tests without such validation it is later discovered that many of the pieces of code were not malicious, but rather broken or legitimate files that were mistakenly used.

In a recent example [16], it was obliquely suggested that the group commissioned to perform the testing used virus generators. This immediately caused AV researchers to doubt the testers' competence, as virus generation kits are notoriously unreliable when it comes to producing viable viruses. Since they didn't describe their testing methodology in useful detail, it wasn't known how or if they verified their testing samples.

The possibility that some or all samples were non-viral invalidates tests of AV scanners, if it is assumed the samples were viral. If this is the case, the highest detection rate does not necessarily equal the best performance, since it could include a large number of false positives [15], even supposing that all scanners tested were consistently configured.

The AV industry is reluctant to condone creation of new malware or viral code, even if just for testing. There are many reasons for this stance: the adherence of most researchers to a stringent code of ethics, concern about safety issues when new viruses are handled by inexperienced testers, validation difficulties, and so on. That said, it isn't actually necessary for anyone to create viruses to test heuristics.

"Retrospective testing" involves testing a scanner that hasn't been updated for a period of time (three months is a period commonly chosen), with validated malware that has appeared since the last update applied to the test-bed scanner. This provides reasonable assurance that heuristic capability is being tested, not the detection of known viruses by virus-specific algorithms. Such a test by no means lessens the need for competent testing, but it avoids the ethical and practical difficulties associated with the creation of new viruses for testing purposes. However, it doesn't eliminate the need to validate samples, or to carefully construct meaningful tests.

Almost all of the major AV vendors provide daily (or more frequent) detection updates, so testing a scanner when it's three months out of date doesn't say very much about its current detection capabilities. A more valid approach might be to test the capabilities at different points, or to test with a specific virus to determine the first point at which detection occurs. Clearly it's worth noting if a scanner was capable of detecting malware before it was known to exist.

> "Retrospective testing" involves testing a scanner that hasn't been updated for a period of time, with validated malware that has appeared since the last update that was applied to the test-bed scanner.

Conclusion: An Heuristic Paradox

Interestingly, even though heuristic technology is more sophisticated now than it was in the 1990s, overall detection rates have fallen dramatically, though detection rates for "old school" malware (macro viruses, mass mailers, and so on) remains impressively high.

While it's sometimes suggested that this overall decline is due to the ineffectiveness of the AV industry, or its desire to cling to a virus-specific detection model, this isn't so. A major contributing factor is the increased sophistication of malware authors, who have developed a wide range of approaches to minimizing the susceptibility of their product to heuristic detection, and who test the effectiveness of these approaches against suitably updated and configured scanners. The problem is more troublesome now than it was a few years ago, when it was considered (by the vendors, at least) something of a bonus if an AV product detected anything other than viruses.

> Malware authors have developed a wide range of approaches to minimizing the susceptibility of their product to heuristic detection.

Nowadays, viruses (i.e. programs with an identifiable replicative functionality) constitute a far smaller proportion of all malicious programs [17]. In a sense this makes the job of the heuristic scanner far harder; it's conceptually simple to detect a virus heuristically if you can untangle the code enough to determine that it's intended to replicate, though it isn't always technically possible to detect a replicative program. Determining automatically that a program is a bot, or a Trojan of some sort, or simply that it's malicious in intent, is a much greater challenge. [5]

Take a classic example: a program that reformats a disk is not malicious by definition – indeed, that might be its overt and only function. However, if it's executed because the computer user has been duped into believing that it will play a movie or improve Internet access, it's reasonable to regard it as malicious. The real problem in such a case lies in establishing an algorithm which will discriminate on the basis of the user's understanding of the program's purpose and the programmer's intent, rather than on a programmatic characteristic.

If we can't establish a reliable heuristic for malice or intent, though, we can apply other heuristics and assign a score to a program accordingly. A close programmatic resemblance to known malware is a likely high scorer. There are many other behaviors that can ring alarm bells, according to context, say for instance, opening an SMTP or IRC channel, or a file transfer mechanism. Analysis of executable files can flag many coding oddities, such as suspicious patches and flag combinations, inconsistent header characteristics, indications of size mismatches, and so on. The wider context in which a possibly malicious program is found can also provide valuable clues as to its nature. Message analysis may indicate similarities to a known mass mailer or email-borne Trojan, and may even contain useful information such as the password for an encrypted archive file.

Although some scanners have this capability, it might be optimistic to expect a heuristic scanner to automatically scan for a passphrase, especially in a message with a high percentage of graphic content. The chances of detecting such a passphrase may be higher in a message that resembles other malicious messages. Messages carrying malicious programs or URLs may also resemble other types of malicious messaging traffic such as phishes and spams – virus writers and spammers have been borrowing techniques from each other for many years now; evidence shows a growing confluence of interest between these previously disparate groups. Email scanners are often expected to detect these and other forms of email abuse, as well as pure malware. Traffic analysis may show patterns associated with malicious activity, such as mass mailers, botnet-generated spam and scams, and so forth. For these reasons, gateway scanning for spam (heuristic and otherwise) can also add considerably to the effectiveness of malware detection.

However, it's by no means certain that we will see the same high percentages of proactive detections in the foreseeable future that we did in the early days of heuristic scanning, welcome though that would be, to users and vendors alike. Malware authors have different priorities. Rather than a scattergun approach (maximum spread of a single variant), now their focus is frequent but short runs of a given instance of malware, which may be targeted to specific individuals or groups. Even simple changes like a quick-fire series of modified run-time packers to change the program's footprint can reduce detection (heuristic and non-heuristic), and stretch resources at even the biggest anti-malware laboratories. Forms of malware that make frequent use of botnet technology to self-update and self-modify once installed on a compromised machine can be very hard to detect.

There's no cause for panic, though — we've been living with these problems for several years. And common sense computer hygiene, good patching practices, and frequent anti-malware updates continue to provide pretty good protection. Not only that, but increasingly sophisticated virtualization and emulation techniques, coupled with heuristic analysis, remains a strong and continually improving component of the security vendor's armory. However, neither the AV vendors nor the proponents of "flavor-of–the-month" alternative technologies can realistically claim to be able to detect all future threats proactively.

The trick is to keep your expectations realistic.

References

[1] "A Short Course on Computer Viruses 2nd Edition", pp 2, 49 (Dr Frederick B Cohen): Wiley, 1994.

[2] "VIRUS-L/comp.viru Frequently Asked Questions (FAQ) v2.00" (N. FitzGerald et al., 1995): http://www.faqs.org/faqs/computer-virus/faq/ (Date of access 12th January 2007)

[3] "Analysis and Maintenance of a Clean Virus Library" (Dr. V. Bontchev): http://www.people.frisk-software.com/~bontchev/papers/virlib.html (Date of access 12th January 2007)

[4] "The Anti-Virus or Anti-Malware Test File": http://www.eicar.org/anti_virus_test_file.htm

[5] "Trojans" (Harley), in "Maximum Security 4th Edition" (ed. Anonymous): SAMS, 2003

[6] Oxford Compact English Dictionary, Oxford University Press: http://www.askoxford.com/ (Date of access 12th January 2007)

[7] Merriam-Webster Online: http://www.m-w.com/ (Date of access 12th January 2007)

[8] "Viruses Revealed" (Harley, Slade, Gattiker) pp158-159: Osborne 2001

[9] "Evolution Discussion Group Fall 1996 Phylogenies and Evolution, Useful Terms" - University of British Columbia Zoology Department: www.bcu.ubc.ca/~otto/EvolDisc/Glossary.html (Date of access 12th January 2007)

[10] "Virus Proof" (P. Schmauder), page 187: Prima Tech (2000)

[11] Dr. Solomon's Virus Encyclopaedia (Solomon, Gryaznov), pp30-31: S&S International (1995).

[12] The Art of Computer Virus Research and Defense (Szor), page 441, pp451-466: Addison-Wesley (2005).

[13] "Heuristic Programming": http://www.webopedia.com/TERM/h/heuristic_programming.html (Date of access 12th January 2007)

[14] Anti-virus programs: testing and evaluation (Lee): in "The AVIEN Guide to Malware Defense in the Enterprise" (Ed. Harley): Syngress (2007, in preparation).

[15] "AV Testing SANS Virus Creation" (Harley): Virus Bulletin pp6-7, October 2006

[16] "Consumer Reports Creating Viruses?" (Sullivan): http://redtape.msnbc.com/2006/08/consumer_report.html (Date of access 12th January 2006).

[17] "Email Threats and Vulnerabilities" (Harley). In "The Handbook of Computer Networks" (Ed. Bidgoli): Wiley (2007 – in press).

Glossary

Adware	Program that performs some action (such as showing a popup screen or sending a browser to a web site) that brings an advertiser/product to the attention of the computer user. Often considered a Trojan if installed without the knowledge or permission of the user.
Almost Exact Identification	Recognition of a virus where the identification is only good enough to ensure an attempt to remove the virus will not result in damage to the host by using an inappropriate disinfection method. Every section of the non-modifiable parts of the virus body is not uniquely identified.
Checksum	In this context a checksum is a computed value that is dependent upon the content of a specific file. If the content of that file changes, the checksum will change. (Some checksumming methods are prone to collisions – i.e. a file may be produced that has the same checksum as another, but in the majority of cases applied to a single file, a change in that file will affect the calculated checksum – this is enough for most purposes of integrity/change checking.)
Corruption	Damage causing altered or impaired function, or non-viability (in this context specifically to a virus).
DDoS	Distributed Denial of Service attack. Characteristically, a remote attacker uses zombie or agent software maliciously installed on a network of machines to attack other systems in such a way that their functionality is impaired.
Destructive Trojan	Trojan that causes (usually deliberate) direct damage, as opposed to something less damaging, such as stealing passwords or other data.
Dropper	Program (usually non-viral) that installs another malicious program such as a worm or virus.
EICAR test file	Uniquely formatted program file, which most AV programs recognize as a test program, and respond to in a very similar way to that in which they respond to viruses. The EICAR file is not a virus and presents no malicious threat: if executed, it simply displays a screen identifying itself as a test file.
Exact Identification	Recognition of a virus when every section of the non-modifiable parts of the virus body is uniquely identified.
False Negative	Describes the scenario where an anti-malware scanner fails to detect actual malware.
False Positive	Describes the scenario where an anti-malware scanner incorrectly detects malware where there is none.
Garbage files	In AV research this file is not a malicious program, but included in badly maintained malware collections as if it were.

Generic	Describes security programs that don’t recognize specific threats, but defend using a method that blocks a whole class (or classes) of threats. A generic signature is a special case of this; a whole set of variants are detected and processed by a single signature rather than by individual signatures for each variant. Antonym of “virus specific.”
Germ	A “generation zero” virus that hasn’t yet infected anything (e.g. a file that consists only of virus code, rather than an infected binary).
Heuristic detection/ scanning	Recognition of an object that has enough viral or malicious characteristics to suggest that it is probably a virus or other malware.
Intendeds	Viruses (or, less often, other malicious programs) that don’t work for one reason or another, often because of insufficient testing by the author.
Joke program	Program that performs some unexpected act which may be annoying, but isn’t actually destructive. The line between a joke and a Trojan can be very tenuous.
Keyloggers	A program that monitors keystrokes, often installed for malicious or criminal purposes such as password theft.
Known Virus Scanning, Virus-Specific Scanning	Scanning for known viruses resulting in the identification by name of a virus found in the scanned environment.
Negative heuristic	A rule or criterion, which if met lessens the likelihood that the object being analyzed is not viral or malicious.
Passphrase	As opposed to a password which is usually a single ‘word’ or string, a passphrase is usually a longer group of words which is used as a more secure form of password.
Positive heuristic	A rule or criterion, which if met increases the likelihood that the program being analyzed is viral or malicious.
Retrospective testing	A technique for testing the heuristic capabilities of a scanner or scanners by not updating it for a set period of time, then using it to scan malware that has appeared subsequent to the last update.
Rootkit	A program or suite of programs installed covertly in order to allow unauthorized, privileged access to a system. Sometimes the term stealthkit is used, though this can denote unauthorized but unprivileged access. [See “The Root of All Evil? Rootkits Revealed” by David Harley & Andrew Lee - http://www.eset.com/download/whitepapers.php]
Scan String, Search String	A sequence of bytes found in a known virus that shouldn’t be found in a legitimate program. The term is not restricted to static search strings, and may include wildcards and regular expressions, or the use of another virus-specific detection algorithm. Also sometimes known as “scan signature”.

Self launching	Term used to describe malicious software that doesn't require any action on the part of the victim to spread or trigger, or both.
Signature	Synonym for "scan string". May be applied to a static search string, but best avoided altogether, particularly as it often misleads people into thinking there is a single byte sequence used by all virus scanners to recognize each virus or variant.
Spyware	Program that covertly gathers information about the computer user and passes it on to an interested party. Includes some forms of adware.
Virus generator program	Program that is not itself a virus, but generates viruses. May also be referred to as a "virus kit".
Virus-specific detection	Detection of known viruses using search strings specific to those viruses or variants.
Wildcard	Character that can be used to represent another character or sequence of bytes, or indicates the use of a specialized form of regular expression.
Zombie	Backdoor program on a compromised PC that waits for and acts upon instructions from a remote machine, or the compromised PC itself.