KASPERSKY lab

White Paper

Malicious Code Detection Technologies

By Alisa Shevchenko
Virus Analyst, Kaspersky Lab

Preface

Just like every other type of technology, malicious code has grown increasingly sophisticated and complex. The antivirus industry must try to stay one step ahead, especially since it is often easier to produce malicious code than it is to detect it. This white paper provides an overview of the evolving combat tactics used in the antivirus battle, giving both simplified explanations of technological approaches as well as a broad chronological perspective.

Many of the technologies and principles discussed in the paper are still current today, not only in the antivirus world, but also in the wider context of computer security systems. The early malicious code detection technology was based on signatures – segments of code that act as unique identifiers for individual malicious programs. Using signatures is a relatively primitive and repetitive technology which requires little explanation and is widely understood.

As viruses have evolved, the defense technologies also had to evolve. Now they involve the use of more advanced approaches, such as heuristics and behavior analyzers, that we collectively refer to as "nonsignature" detection methods. This paper focuses primarily on these nonsignature technologies. It will define terms such as "heuristic," "proactive detection," "behavioral detection," and "HIPS"; it will explain how they are related; and identify some of the advantages and disadvantages of each. Some of the technologies currently used by the antivirus industry – such as unpacking packed programs and streaming signature detection – were intentionally not included in this paper to allow for a more in-depth discussion of nonsignature detection methods.

This paper was developed for readers who have a very basic understanding of antivirus technologies, but who are not experts in the field. Its aim is to systematically and objectively examine issues surrounding the use of malicious programs and the defense techniques that are essential for protection from them.

Table of Contents

Malicious Program Defense Systems – A Model

Let's begin by defining a model for discussing malicious program detection technologies that will help simplify and clarify some of the explanations. This model operates on the basic premise that any defense technology can be separated into two components – a technical component and an analytical component. In reality, these components may not be clearly separable at the module or algorithm level within every malicious program. However, in terms of function, their differences are significant and important.

Accept this premise: Any defense technology can be separated into two components – a technical component and an analytical component.

The technical component is a collection of program functions and algorithms that selects the data that will be analyzed by the analytical component. This data may be anything – from text strings within a file, to a specific action the program performs, to a full sequence of actions that the program performs, and more.

The analytical component serves as the decision-making system. It assesses the data provided by the technical component using one or more algorithms and then issues a verdict about the data. The security program will then use the verdict to take action on the malicious program according to the security policy that has been set in the security program. For example, a few of the possible actions that could occur based upon the verdict might be –

- Notifying the user
- Requesting further instructions from the user
- Placing a file in quarantine
- Blocking unauthorized program actions

Here's how the model applies to one of the simplest security program techniques – signature detection. The technical component collects information about the file system, files, and file contents; it then passes that information on to the analytical component. The analytical component compares byte sequences in the data provided against byte sequences known to be suspicious or malicious, and issues a verdict accordingly.

Most of today's security programs are exceedingly complex, but it helps to "break them apart" conceptually into these two types of components to understand how they work. Separating the components also helps us explain how components relate to one another and the pluses and minuses of each. For example, we'll explore later in this paper how the heuristics method is only one type of analytical component, rather than an independent technology itself. Similarly, HIPS (Host Intrusion Prevention System) is just a type of technical component (a way to collect data) and not an independent technology itself. This lets it become more apparent that heuristics and HIPS are neither contradictory nor mutually exclusive technologies, because their

basic functions within a security program are fundamentally different. The beauty of the model used in this white paper is that it allows us to discuss heuristics without specifying exactly what data is being analyzed, and we can talk about a HIPS system and the data it collects without knowing anything about the principles that cause certain verdicts to be issued.

Figure 1, below, identifies many of the key concepts that will be discussed in this white paper. The horizontal axis positions technical components along a continuum and shows how they overlap. The vertical axis helps to suggest the level of sophistication of analytical components – from simple comparisons to detailed analysis. You'll understand and be able to use Figure 1 more easily once you better understand some of the technical components.

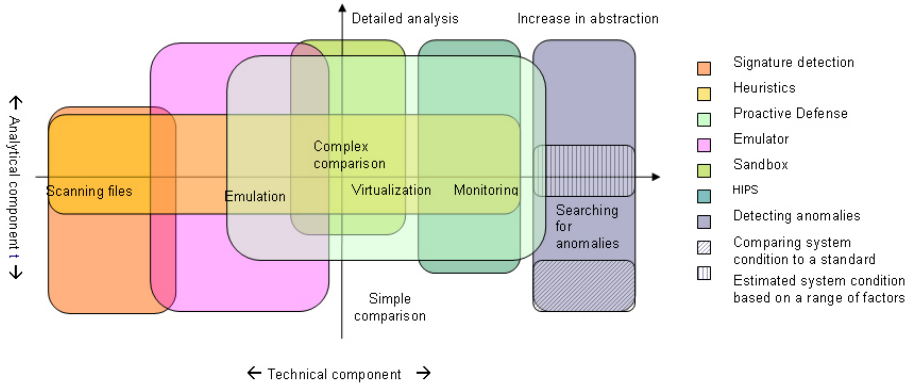

Figure 1 – A Model for Assessing Methods of Detecting Malicious Code

The Technical Component

As explained earlier, the technical component of a malicious program detection system is the data collection system that provides the data that needs to be analyzed. Before talking about data collection, though, it is important to understand that there are different and important "views" of a malicious program. It can be evaluated or viewed as a long string of data, as a series of instructions, or by the effect it has on the operating system. These different views help to explain why there are so many different approaches and possibilities for data collection, even before any analysis begins.

These are some of the most common methods used for collecting the data that will be used to identify malicious programs –

- Treat the file as a mass of bytes.

- Emulate the program code. (Emulation means placing the program in a different environment and "tricking" it into behaving as if it's in its intended environment so that the results can be preserved.)
- Launch the program in a sandbox. (Provide a safe environment and launch the program to determine if it "plays nicely with others.")
- Monitor system events.
- Scan the system for anomalies.

These methods are listed in terms of increased levels of abstraction. It is simple and straightforward to think of a malicious program as a collection of bytes, a bit more abstract to think of it as a sequence of actions (behaviors), and still more abstract to think of it as a collection of effects that it has within an operating system.

To combat malicious programs today, greater and greater levels of abstraction are required. For that reason, our list of methods also provides a simple chronology of the emergence of malicious program detection techniques. It should be noted that the methods listed above are not so much completely separate technical approaches as they are points on a continuum of technology that can be used to collect data for analysis in detecting malicious programs. Technical approaches to malicious code detection gradually evolve and intersect with one another. This will become more apparent as we examine each of these methods in greater detail.

Technical approaches to malicious code detection gradually evolve and intersect with one another.

Scanning Files

The very first antivirus programs analyzed file code as simple byte sequences. This "analysis" was a simple comparison of byte sequences in the file against known signatures (specific byte sequences that are representative of each known virus). However, we are currently focusing not on the analysis, but on the technical component which provides the data. Scanning merely refers to extracting data from files, structuring that mass of bytes in a specific way, and then transmitting those structured bytes to the analytical component.

While this data collection method is relatively old and does not take into account any of the behavior of the program, it is still used by all modern antivirus software. It is no longer the sole, or even the main, method used today; it is used as a complement to other technologies.

Emulation

The emulation approach is a step between treating a program as a collection of bytes (scanning) and processing a program as a particular sequence of actions.

An emulator breaks down the program's byte code into commands, and then launches each command in a virtual environment, which emulates the computer environment. The use of this virtual environment allows security solutions to observe program behavior without posing a threat to the real operating system or user. Think of emulation as a nanny putting a child into a big plastic bubble to isolate it from the real world, and then observing to make sure the child (the virus) doesn't do anything that might cause harm in the real world.

While an emulator still works with a file, its primary focus is on events rather than inanimate bytes of data per se. Emulators are used in many, and possibly all, major antivirus products. They may be used as a basic, core-level protection engine, or as "insurance" for a more abstract and sophisticated engine, such as a sandbox.

While an emulator still works with a file, its primary focus is on events rather than inanimate bytes of data per se.

Virtualization – The Sandbox

Virtualization is a logical extension of emulation, and a sandbox is one form of virtualization. To continue the nanny metaphor, the plastic bubble is gone, and the sandbox is part of the real world. However, before the child is allowed to play in the sandbox, many rules have been established and will be enforced by the nanny with respect to the way the child is permitted to behave in the sandbox. In the context of information security, the operating system is the world, and the malicious program is the rambunctious child, and the rules are the restrictions on interactions with the operating system. One such rule might be a ban on modifying the system directory. If a program tries to modify the system directory, it may be fed a virtual copy of the system directory so that it can continue to operate without impacting the operating system.

The line between emulation and virtualization may be a fine one, but it is a clear one. Emulation occurs in a fully contained, controlled, and separate environment – the plastic bubble. Virtualization occurs in the real world (the operating system), but under careful rules and guidance. The child plays in the real world, but may be handed a plastic cup (a virtual copy of a system resource), rather than a glass cup (a real, breakable system resource) when he requests refreshment.

Sandboxing, like emulation, isn't used extensively in antivirus products, mainly because it requires a large amount of resources. It's easy to tell when an antivirus program uses a sandbox, because there will always be a time delay between when the program is launched and when it actually starts to run. (Or, if a malicious program is detected, there will be a delay between the program's launch and the virus detection notification.) At the moment, sandbox engines are used in only a handful of antivirus products because of the performance

issues. However, a great deal of research is underway on hardware virtualization, which could remove that performance issue in the near future.

Monitoring System Events

While an emulator or sandbox observes each program separately, monitoring system events is the next level of abstraction. It involves the simultaneous observation of all programs to understand their impact on the operating system. Data is collected by intercepting operating system functions. By intercepting calls to various system functions, information can be obtained about exactly which program is doing something to the system. Over time, the monitor collects statistics on these actions and transfers them to the analytical component for analysis.

This technology approach is currently the most rapidly evolving one. Monitoring of system events is used as the technology component in several major antivirus products and as the main component in individual system monitoring utilities such as Prevx, ThreatFire (formerly CyberHawk) and a number of others. However, given that it is possible to defeat any form of protection, this detection method carries special risks because the programs are always being launched in a real environment, allowing damage to potentially occur before the detection does. To some extent, this might be likened to sending several children to play in the sandbox without any rules or training, while the nanny waits on the park bench to observe problems as they arise – a limited overhead approach, but a high-risk one.

Monitoring system events might be likened to sending several children to play in the sandbox without any rules or training, while the nanny waits on the park bench.

Scanning for System Anomalies

Included here as the final, logical extension of other technology approaches, scanning for system anomalies is the most abstract method used to collect data about a potentially infected system. This method relies on three basic principles –

- An operating system, together with the programs running within that system, is an integrated system.
- The operating system has an intrinsic “system status.”
- If malicious code is run in the environment, then the system will have an “unhealthy" status. This differs from a system with a "healthy" status, in which there is no malicious code.

These principles are used to help determine a system's status, and the approach requires analysis that compares the status to a standard and/or investigates all system parameters as a single, composite entity.

To detect malicious code effectively using the system anomalies method, a relatively complex analytical system, such as an expert system or neural network, is required. The obvious challenges imposed by this approach include defining what a "healthy" status is, determining which discrete parameters need to be tracked, and deciding how they should be analyzed. In keeping with our metaphor, this might be likened to a mother sending her children to the sandbox to play, without the protection of a nanny, but instead relying on her own intuition about a problem arising. A mother's intuition can be likened to the neural network or expert system – difficult to explain exactly how it works, but also proven to be quite effective at times.

Due to its complexity, the system anomalies technology is still classified as an emerging technology.

Due to its complexity, the system anomalies technology is still classified as an emerging technology. For the most part, it has emerged in the form of anti-rootkit utilities. (Certain Trojans, for example, are known for going undetected, because they gain "root" access to the computer. That means that they run at the most basic level of the machine and have unusual powers, such as the ability to hide files.)

The Analytical Component

Now that we've explored the technical component, we turn to the analytical component. As Figure 1 indicates, the degree of sophistication of decision-making algorithms varies as you traverse the vertical axis. Generally speaking, decision-making algorithms can be divided into three different categories, although these analytical categories merely represent three different points on a continuum of sophistication.

Simple Comparison

Technologies that fall into this category issue a verdict based on the comparison of a single object to an available sample. The result of the comparison is binary – a clear "yes" or "no." An example is the identification of malicious code by locating a specific byte sequence. Another higher level example is identifying a suspicious program through its use of a single action that it takes, such as creating a record in a critical section of the system registry or in a folder that would cause it to automatically run.

Complex Comparison

In a complex comparison, a verdict is rendered based on the comparison of one or multiple objects with corresponding samples. The templates for these comparisons can be flexible and the results will be probability based. An example of this is identifying malicious code by using several byte signatures, each of which is non-rigid; that is, the individual bytes are not determined.

This analytical approach could be likened to the identifying a criminal using imprecise but fairly detailed witness descriptions (female, dark hair, light eyes, a mole on the left side of her face, slight limp). The probability of it actually being the criminal rises as more factors are positively matched during the analysis. Another higher level example is identifying malicious programs based on the calls it makes to other programs (not necessarily in sequential order), as well as the parameters that it passes to those other programs when the calls are made. To further the criminal analogy, this would be similar to scouring a suspect's phone records for calls to certain people, and then listening to a wiretap of how each conversation began to determine the likelihood that the suspect committed a crime.

Expert Systems

Expert systems issue a verdict only after a sophisticated analysis of data. An expert system may include elements of artificial intelligence. One example of an expert system is identifying malicious code not by a strict set of parameters, but by the results of a multifaceted assessment of all of its parameters at once, taking into account the "potentially malicious" weighting of each parameter and calculating the overall result.

One example of an expert system is identifying malicious code not by a strict set of parameters, but by the results of a multifaceted assessment of all of its parameters at once.

Using our criminal suspect identification analogy, an expert system might be able to identify a criminal taking a composite view of the various types of evidence surrounding a case. Each type of evidence considered alone may suggest different suspects committed the crime. Evidence might include conflicting eye-witness reports, each suspect's last known location prior to the crime, each suspect's prior criminal behavior, whether the victim knew the suspect from school or some other venue, and somewhat confusing polygraph results. In this case, an expert system (one that assigned probabilities to all the different evidence types and considered all the evidence as a unified set of data) might issue a verdict that confirmed with relatively high certainty that one suspect was guilty, while also being able to eliminate the "false positives" represented by the other suspects.

Real Technologies at Work

Now that we've fully explored the simplified model for discussing malicious code detection technologies, we will look at some of the actual technologies available.

Typically, security software producers market the new technologies they provide under names intended to build confidence, but which offer no indication of the actual technologies being applied. Common examples include Proactive Protection in Kaspersky Anti-Virus, TruPrevent from Panda, and DeepGuard from F-Secure. One advantage of this approach is that the technologies aren't automatically pigeon-holed in narrow technical

categories. Nevertheless, product descriptions that are aimed at purchasing decision-makers are typically laden with general and commonly recognized terms, such as "heuristic," "emulation," "sandbox," and "behavior blocker."

This is where the tangled web of terminology begins. These terms, which are ones that some deem to be more user-friendly, are used liberally in marketing literature and reviews, do not all have precise meanings. Ideally, there would be one clear definition for each term, so that one person would not interpret a term in a completely different way from someone else. Furthermore, the definitions used by those who have authored the descriptions of the user-friendly terms used to describe malicious code detection are often very different from the definitions used by the experts in the antivirus industry. This helps to explain why the descriptions of technologies on developer websites may be heavily laden with technical terminology while still missing the mark in providing information about how the technology works that would facilitate an unbiased assessment of the technology being sold.

The definitions used by those who have authored the descriptions of user-friendly terms used to describe malicious code detection are often very different from the definitions used by the experts in the antivirus industry.

For example, some antivirus software manufacturers say their products are equipped with a host intrusion protection system (HIPS), proactive technology, or nonsignature technology. A user's understanding of HIPS, based on a user-friendly definition that describes HIPS as "a monitor that analyzes system events for malicious code" would be very imprecise, and the description is one that could mean almost anything in the security world, such as an emulator engine that is equipped with a heuristic analysis system. "Heuristic" is another term which, when used alone, provides inadequate detail to understand what type of technology is actually being used.

This is not to say that developers are trying to deceive prospective customers. Remember, even the technical and analytical approaches discussed earlier in this paper are not crisp categories, but rather points on a continuum that help to describe variations in the approach. So it's easy to see how those who write about technologies, both inside and outside the companies, can get the terminology confused, especially when they try to adapt the descriptions to use the "user-friendly" terminology. For this reason, those who make purchase decisions for security software solutions must be wary of some of the descriptions they read in the literature.

Figure 1 positions the more commonly used terms against the terms we used in the simplified, two-dimensional model we explained in the earlier sections of this paper. Their placement with respect to the axes is also indicative of their relative level of sophistication. We will now explore each of the common terms, whether precise or imprecise, in the context of the earlier definitions of components that exist in our model.

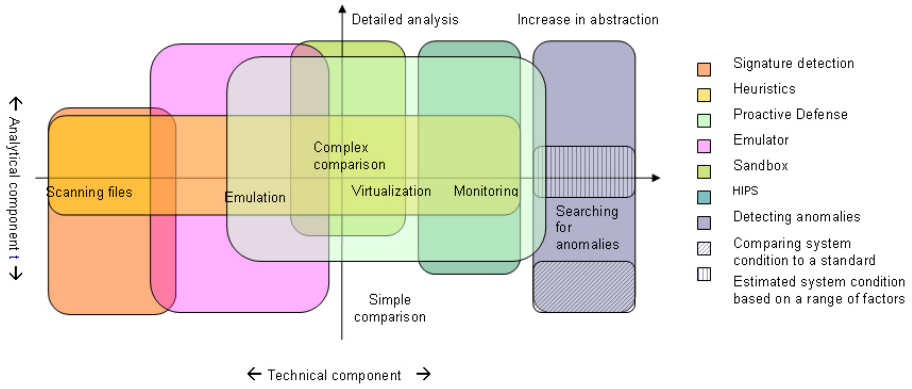

Figure 1 – A Model for Assessing Methods of Detecting Malicious Code

Signature Detection, Emulators, and Sandboxes

Fortunately, there are a few commonly used terms that cause little or no confusion. We'll begin with those. First of all, there are few variations in the meaning of the term *signature detection*. From a technical perspective, it means working with file byte code, and from an analytical point of view, it is a primitive means of processing data, usually by using simple comparison. As mentioned earlier, while it is old technology, it is also highly reliable. That's why antivirus software companies continue to incur the considerable costs associated with keeping signature databases up to date.

While signature detection is old technology, it is also highly reliable. That's why antivirus software companies continue to incur the considerable costs associated with keeping signature databases up to date.

There aren't many possible interpretations of the terms *emulator* or *sandbox*, either. When those terms are used, people are relatively consistent with the definitions of those technical approaches as described in an earlier section of this paper. The analytical component used in conjunction with this type of technology can be an algorithm of any complexity, ranging from simple comparison to expert systems.

Heuristics

The term *heuristics* is less transparent. According to Ozhegova-Shvedovaya, the definitive Russian dictionary, "heuristics is a combination of research methods capable of detecting what was previously unknown." Heuristics are first and foremost a type of analytical component in protection software, but not a clearly defined technology. Outside a specific context, in terms of problem-solving, it closely resembles an "unclear" method used to resolve an unclear task.

When antivirus technologies, as well as the term heuristic itself, first began to emerge, the term referred to a distinct technology – one that would identify a virus by using several flexibly assigned byte templates. That is, it was a system

with a technical component (working with files) and an analytical component (using complex comparison). Today the term heuristic is usually used in a much broader sense to refer to technology being used to search for unknown malicious programs. In other words, when speaking about heuristic detection, developers are usually referring to a protection system with an analytical component that uses a fuzzy search to find a solution. This is the equivalent of saying that the analytical component involved uses either complex analysis or an expert system. And the technical component (the part that collects the data for this analysis) can range from simply working with files up to working with events or the status of the operating system.

Behavioral Detection, Proactive Detection, and HIPS

Behavioral detection and *proactive detection* are terms that are far from being clearly defined. They can refer to a wide variety of technologies, ranging from heuristics to system event monitoring.

The term *HIPS* is also frequently used in descriptions of antivirus technologies, but not always appropriately. In spite of the fact that the acronym stands for Host Intrusion Prevention System, those words do not reflect the essential nature of the technology in terms of antivirus protection. In one context, the technology can be very clearly defined as a type of protection with a technical component based on the monitoring of system events. The analytical component of the protection software may be of any type, ranging from coinciding separate suspicious events to complex analysis of a sequence of program actions.

One must note, however, that in practice, the term HIPS is often used to describe a wide variety of things. It has been used to refer to primitive protection for a few registry keys, to a system that provides notification of attempts to access certain directories, to a more complex system that analyzes program behavior, and even to another type of technology that relies on system event monitoring.

One must note, however, that in practice, the term HIPS is often used to describe a wide variety of things, ranging from protection of a few registry keys…to technology that relies on system event monitoring.

The Pros and Cons of Different Detection Methods

If we use the model introduced in this paper to examine malicious code detection technologies as a group rather than individually, an interesting picture of the tradeoffs that have to be made in both developing and selecting a malicious code detection system begins to emerge.

Practical Facets of the Technical Component

The technical component controls three important facets of the malicious code defense system that affect its desirability for a particular user or environment –

- **Resource consumption** is the share of processor time and RAM required either continually or periodically to ensure protection. If the technical approach being used requires a lot of resources, it may slow down system performance. Emulators run slowly; regardless of implementation, each emulated instruction will create several instructions in the artificial (plastic bubble) environment. The same is true for virtualization. System event monitors also slow system performance, but the extent to which they do so depends on the implementation. Similarly, with file detection or system anomaly detection, the load on the system is entirely dependent on the implementation.
- **Security** is the level of risk that the operating system and user data will be subjected to during the process of identifying malicious code. This risk is always present when malicious code is run in an operating system. The architecture of system event monitors means that malicious code has to be run before it can be detected, whereas emulators and file scanners may detect malicious code before it is executed in the real system environment.
- **Protection** is the extent to which a technology may be vulnerable, or how easy it may be for a malicious program to avoid detection. Packing files, polymorphism, and rootkit technologies are a few approaches that virus writers use to combat file detection. It's a little tougher to circumvent emulators, but it is still possible. Because emulators may react to certain commands a little differently from the actual processor, virus writers can sometimes detect and circumvent an emulator. On the other hand, it's very difficult for malicious programs to hide from a system event monitor, because it's nearly impossible to mask a behavior.

It's very difficult for malicious programs to hide from a system event monitor, because it's nearly impossible to mask a behavior.

In summary, the implications of these three facets of the technical component are: 1) the less abstract the form of protection, the more secure it will be, 2) the less abstract the form of protection, the easier it will be for malicious programs to circumvent it, and 3) resource consumption must always be factored into the equation.

Practical Facets of the Analytical Component

The analytical component of a technology also has three important facets that must be taken into consideration in evaluating a solution –

- **Proactivity** - refers to a technology's ability to detect new, not-previously-identified malicious programs. For example, the simplest type of analysis (simple comparison) represents the least proactive technologies. That's why signature detection is only an approach to detecting known malicious programs. The more complex an analytical

system is, the more proactive it is. Proactivity is directly linked to how frequently updates need to occur. For example, signature databases have to be updated frequently; more complex heuristic systems remain effective for longer periods of time; and expert analytical systems can function successfully for months without an update.

- The **false positive rate** is also directly related to the complexity of a technology's analytical component. If malicious code is detected using a precisely defined signature or sequence of actions, as long as the signature (be it byte, behavioral, or other) is sufficiently long, identification will be absolute. The signature will only detect a specific malicious program, and not others. The more programs an analytical component attempts to identify, the less definitive it becomes. The less definitive it becomes, the more likely it is to indict a non-malicious program – an occurrence which the industry refers to as a false positive.
- The **level of user involvement** is the extent to which a user needs to participate in defining protection policies – creating rules, exceptions, blacklists, and white lists. It also reflects the extent to which the user participates in the process of issuing verdicts by confirming or rejecting the suspicions of the analytical system. The level of user involvement depends on the implementation. However, as a general rule, the more complex the analysis, the more false positives there will be to review and correct. Correcting false positives always requires user input.

As a general rule, the more complex the analysis, the more false positives there will be to review and correct. Correcting false positives always requires user input.

Summarizing the impact of these three facets of the analytical component, we can conclude that 1) the more complex the analytical system, the more powerful the antivirus protection is and 2) increased complexity means an increased number of false positives, which in turn places a dependency on user involvement.

Hopefully, understanding this model and the facets of each component makes it easier to evaluate the pros and cons of any technology. Consider, for example, an emulator with a complex analytical component. This form of protection is very secure because it does not require the file to be launched in the real environment. However, a certain percentage of malicious programs will go undetected, either due to anti-emulator tactics used by the malicious code or due to deficiencies in the emulator itself. However, this type of protection has great potential; if carefully implemented it will detect a high percentage of unknown malicious programs, albeit at the expense of system resources.

How to Choose Nonsignature Protection

Currently, most security solutions combine several different technologies. Classic antivirus programs often use signature detection in combination with some form of system event monitoring, an emulator, and a sandbox. So what should buyers look for to find protection that best suits their specific needs?

The best protection solution must take into account the specific but variable demands and conditions that the business environment and/or individual users place on the system.

First of all, let's dispel the myth that there is a universal solution or a "best" solution. Each technology has advantages and drawbacks. For example, monitoring system events continually consumes a great deal of processor time, even though it's one of the toughest approaches for malicious program writers to crack. Malicious code can circumvent an emulator by using certain commands in its code, but if those commands are used, the malicious code will be detected preemptively, leaving the system untouched. Simple decision-making rules require a lot of user input (sometimes, too many burdensome questions), while more complex decision-making rules require little user input but can yield many false positives.

Selecting the appropriate technologies is a bit like finding the golden mean. That is, the best protection solution must take into account the specific but variable demands and conditions that the business environment and/or individual users place on the system. For example, if end-users are responsible for installing patches on their own systems (and probably won't find the time) and are allowed to run whatever browser plug-ins and scripts that they wish, they are highly vulnerable. The right tradeoff to make for that user might be a solution that provides a sandbox-type system with a quality analytical component. This type of system offers maximum security, but will also consume enough RAM and processor time that it could slow the operating system beyond acceptable levels on certain machines and for users for whom fast response is critical.

On the other hand, expert users who want to control all critical system events and protect themselves from unknown malicious programs will do well with a real-time system monitor. This kind of system works steadily, but with relatively low overhead on the operating system. However, it does require user input to create rules and address exceptions.

Finally, a user who either has limited resources or does not want the system overhead associated with constant monitoring, and who prefers not to be bothered with creating rules, may be best served by simple heuristics.

Ultimately, it's not a single component that ensures quality detection of unknown malicious programs, but the security solution as a whole. A sophisticated analytical component can help compensate for using simpler technical components.

In choosing a new product, the best advice is to understand user characteristics, and then rely on your own personal evaluation and independent test results.

Kaspersky Lab, Inc. • 500 Unicorn Park • Woburn, MA 01801
phone: (781) 503-1800 • fax: (781) 503-1818
www.kaspersky.com

About Us

Kaspersky Lab delivers the world's most immediate protection against IT security threats, including viruses, spyware, crimeware, hackers, phishing and spam. Kaspersky Lab products provide superior detection rates and the industry's fastest outbreak response time for large enterprises, SMBs, home users and the mobile computing environment. Kaspersky® technology is also used worldwide inside the products and services of more than 100 of the industry's leading IT security solution providers.

For the latest on antivirus, anti-spyware, anti-spam and other IT security issues and trends, visit www.viruslist.com.

Learn more at www.kaspersky.com