

Behavior Blocking: The Next Step in Anti-Virus Protection

Carey Nachenberg 2002-03-19

Behavior Blocking: The Next Step in Anti-Virus Protection

by Carey Nachenberg

last updated March 19, 2002

Introduction

Before the arrival of the fast-spreading worm/blended threat, the staple technology of anti-virus software – fingerprinting - arguably provided both *preventative* and *proactive* protection against the average computer virus. That is, in the past, vendors were able to ship new fingerprints for most viruses before they could achieve widespread distribution. This is because traditional viruses spread slowly - only when humans exchange infected files - on the order of days or weeks. Consequently, in the majority of cases, anti-virus software blocked initial infection, preventing corporate machines from being compromised and precluding the need for costly manual cleanup and downtime.

In contrast, given the prolific speed at which worms and blended threats spread today, the fastest spreading infections sometimes sneak past traditional anti-virus software and entrench themselves in desktop and server systems before anti-virus vendors can post an appropriate fingerprint. Once these machines are infected, the role of anti-virus software fundamentally shifts from a proactive/protective shield to that of a clean-up utility.

Clearly traditional anti-virus software is less effective against the fastest spreading threats. The question is: is there a technology that could transform anti-virus solutions from their current role as clean-up tools to their original role as a protective solution? I believe that the answer is “yes” and that the technology that will make this possible is behavior blocking. This article will provide a high-level look at behavior blocking technology and explore how this technique may help save corporations from the next generation of fast spreading worms and blended threats.

Fingerprinting and Heuristics – Still Effective?

Traditional fingerprint-based anti-virus software detects malicious code by searching for tens of thousands of digital fingerprints in all scanned files, disks and network transmissions. Each fingerprint is a short sequence of bytes extracted from the body of a specific virus strain. If a

given fingerprint is found, the content is reported as infected; however, since anti-virus fingerprints are based on known sequences of bytes from known infections, this technique often fails to detect new strains.

In contrast to fingerprinting, heuristic anti-virus technology detects infections by scrutinizing a program's overall structure, its computer instructions and other data contained in the file. The heuristic scanner then makes an assessment of the likelihood that the program is malicious based on the logic's apparent intent. Such a scheme can detect unknown infections since it searches for generally suspicious logic rather than looking for specific fingerprints.

To cope with the most complex infections, modern fingerprinting and heuristics engines often employ CPU emulation or "sand-boxing" techniques in conjunction with simpler bit-and-byte scanning. These products work by performing limited emulation of a program within a virtual machine to reveal otherwise obscured logic. This emulation is extremely limited (often fewer than 1000 instructions are emulated in the typical program) and the program under scrutiny never actually runs on the real CPU or poses a risk to the system.

A big plus for both fingerprinting and heuristics is their ability to detect infections in files before these threats have a chance to run and infect computers. This is because these techniques can detect infections merely by examining the bits and bytes of each file (or performing a very limited, virtualized emulation session). However, since these schemes don't actually observe full execution of the scanned software, they often fail to detect new infections; there are simply too many ways to obfuscate malicious code, and often the only way to know something is malicious is to watch it run on real silicon and attempt harm. This is where behavior blocking comes in.

Behavior Blocking

Unlike heuristics or fingerprint-based scanners, behavior blocking software integrates with the operating system of a *host* computer and monitors program *behavior* in real-time for malicious actions. The behavior blocking software then blocks potentially malicious actions before they have a chance to affect the system. Monitored behaviors can include:

1. Attempts to open, view, delete, and/or modify files;
2. Attempts to format disk drives and other unrecoverable disk operations;
3. Modifications to the logic of executable files, scripts or macros;
4. Modification of critical system settings, such as start-up settings;

5. Scripting of e-mail and instant messaging clients to send executable content; and,
6. Initiation of network communications.

If the behavior blocker detects that a program is initiating would-be malicious behaviors as it runs, it can block these behaviors in real-time and/or terminate the offending software. This gives it a fundamental advantage over such established anti-virus detection techniques such as fingerprinting or heuristics. While there are literally trillions of different ways to obfuscate and rearrange the instructions of a virus or worm, many of which will evade detection by a fingerprint scanner or heuristic, eventually malicious code must make a well-defined request to the operating system. Given that the behavior blocker can intercept all such requests, it can identify and block malicious actions regardless of how obfuscated the program logic appears to be.

The ability to watch software as it runs in real-time clearly confers a huge benefit to the behavior blocker; however, it also has drawbacks. Since the malicious code must actually run on the target machine before all its behaviors can be identified, it can cause a great deal of harm to the system before it has been detected and blocked by the behavior blocking system. For instance, a new virus might shuffle a number of seemingly unimportant files around the hard drive before infecting a single file and being blocked. Even though the actual infection was blocked, the user may be unable to locate their files, causing a loss to productivity or possibly worse. This is why it is always preferable to detect and prevent infections using the tried-and-true scanning schemes when possible (and why fingerprinting will never go away).

Policy and Expert-based Systems

Existing behavior blocking systems can be split into two categories: policy-based blocking systems and expert-based blocking systems.

Policy-based systems allow the administrator to specify an explicit blocking policy stating which behaviors are allowed and which are blocked. Each time a program makes a request to the operating system, the behavior blocker intercepts the request, consults its policy database and either allows the request to proceed, or blocks the request entirely. For instance, a policy-based behavior blocking system for Java might provide the following options:

Apply this policy to all applets:

Operation description	Block Request?
Allows applets to open files:	Yes
Allows applets to delete files:	Yes
Allows applets to initiate network connections:	Yes
Allow applets to access files in the system directory:	No

Such systems are often appealing to administrators because their logic is transparent and easy to understand. However, these systems are also most prone to false positives and have the largest impact on employee productivity because they block activities of both malicious and legitimate programs with equal vigor; no attempt is made to identify whether the behaviors are malicious or not. Also, few administrators understand the implications of specific blocking policies such as "block all programs from accessing system files." How is the administrator supposed to know how many legitimate programs require such operations, or how effectively such a policy selection will protect against viruses and worms?

In contrast to policy-based systems, expert-based systems employ a more opaque method of operation. In these systems, human experts have analyzed entire classes of malicious code and then designed their behavior blocking systems to recognize and block suspicious behaviors. Under some circumstances a would-be dangerous behavior is allowed, and under others, it is blocked. For example, a behavior blocking expert may know that 80% of malicious code first attempts to modify the startup area of the registry before accessing system files. So he can design his behavior blocking system to only block a program's access to system files after first seeing it modify the startup area of the registry. Such a rule is less likely to block legitimate programs and generate false alarms, yet still blocks a high percentage of threats. While a policy-based system might offer an option to "block access to system files," the expert-based system would offer the option to "block virus-like behavior." Clearly, with such a system, the administrator must take a leap of faith that the experts that built the system have chosen their blocking criteria wisely.

Targets of Behavior Blocking

When building an expert-based behavior blocker, engineers need to consider different blocking rules for each type of malicious code. This section will give some insight into some of the operations that might be blocked to thwart each type of malicious code.

Blocking Parasitic Viruses

Parasitic viruses are self-replicating programs that attach themselves to other programs. When an infected program is launched, the virus gains control and then inserts its logic into other executable files. Behavior blockers can protect against this type of threat by observing modifications of one executable file by another that are characteristic of viral infection. Such modifications include changes to the file headers and modification of code sections in executable files, among others. Behavior blockers can use a range of techniques here: block all programs from modifying other programs, block modifications to certain header fields that allow for infection, etc.

Blocking Worms and Blended Threats

Worms and blended threats spread over networks via e-mail, drive sharing or by exploiting other vulnerabilities. To block these types of threats, the behavior blocker must inject itself between programs and their vectors of propagation. Possible approaches include blocking suspicious use of e-mail APIs to send executable code, preventing unknown programs from communicating over the network, and blocking programs from using drive sharing to copy executable content to other computers.

Blocking Trojan Horses

Trojan horses are perhaps the most difficult to block without false positives since these threats don't have an easily characterized set of behaviors like viruses and worms. Here, behavior blockers have to check for a variety of possibly suspicious behaviors, including modification of system files or the registry, anomalous attempt to access documents or other data ("why is this cute graphics program accessing my spreadsheets?"), blocking unknown/unauthorized programs from communicating over the network, etc. The problem is that many legitimate programs do all of the above activities, and it's often impossible to discern between innocuous and malicious intent.

Behavior Blocking: Room for Improvement

Behavior blocking systems arguably hold great promise as an additional layer of protection against the latest malicious code threats. In fact, there are already a number of small companies offering these solutions. Why have these products had only limited success in the enterprise? I believe that there are four reasons for this lack of success:

1. False positives. Behavior blocking, like intrusion detection software, has a horrible reputation for false alarms. Existing blocking systems have their share of problems and users are unwilling or afraid to deploy them on servers or desktops because of the potential repercussions. False-positives also cost money in terms of man-hours spent responding to them.
2. Administration headaches. Current behavior blocking systems do not have the level of manageability required by the enterprise. They are difficult to roll out, require extensive configuration, and their management consoles (if any) fail to scale well. Finally, some systems require users to change the way they work – a difficult prospect at best - in order to provide protection (e.g. place all of your critical files in one directory and configure the product to block access to this directory).
3. System overhead. Behavior blocking systems must integrate with the operating system and intercept system calls in order to provide protection. Such integration comes at a cost: reduced performance. Given the existing real-time protection already running on machines, a second driver could mean trouble.
4. Some vendors are selling solutions that fail to address the real threats facing corporations: A number of companies offer Java and ActiveX behavior blocking systems. As an engineer and malicious code researcher here at Symantec for the past 11 years, I can count all of the Wild Java and ActiveX threats on the fingers of one hand. I believe customers recognize that some commercial behavior blocking systems offer little protection against real-world viruses, worms and Trojan horses, and have invested their dollars elsewhere.

These are obviously my own observations in speaking with customers, systems engineers and security professionals. I invite those readers who have used these systems to contact me with your experiences.

Next Steps

Over the next few years, I believe we will see a great deal of movement in this area. Large security companies have already spent a fair amount of time researching these systems, and smaller companies are starting to show up on customers' and bigger competitors' radar screens.

There are still a number of significant issues that must be worked out with existing behavior

blocking systems. As with intrusion detection systems (a very close cousin to behavior blockers), false positives are still a concern. This is a "hard" problem in computer science and one for which there is no known good solution. Organizations such as DARPA, universities and private companies have been trying to solve these false positive issues for years, with few commercially viable results. Eventually, I expect that customers will learn to live with a certain level of false positives and obtrusiveness in exchange for protection against the nastiest worms and blended threats, but this remains to be seen.

Whether or not behavior blocking does take off is anyone's guess, but one thing is clear: existing anti-virus solutions are not providing the proactive protection that they did in the past; these solutions will need to evolve along with the threats if they are to survive in the marketplace. Either that or we need to hire more IT staff.

[Privacy Statement](#)

Copyright 2006, SecurityFocus