Realtime publishers
"Leading the Conversation"

The Essentials Series: Modern Malware Threats and Countermeasures

Tools and Techniques for Eliminating Modern Malware

sponsored by

Sunbelt Software

by Greg Shields

Copyright Statement

Tools and Techniques for Eliminating Modern Malware

The first article in this series talked about the classifications of malware seen in today's modern landscape. There, we discussed the economics of malware and how those financial forces are driving the underground malware industry towards more efficient and effective use of malware for dollar gain. Following on, the second article discussed the advanced behaviors seen in those sophisticated malware packages, focusing on a few high-impact techniques that malware authors use today to hide the presence of their wares while they accomplish their mission.

In this, the final part of this series, the focus is on getting rid of these ever-evolving little buggers all across the business IT environment. With extortion and financial gain a primary motivator for malware creators, you need to keep malware away from your IT environment more than ever before.

Signature Limitations

There is a problem with the traditional model for locating malware on a candidate computer. This model has historically relied on a signature-based approach for locating the breadcrumbs of malware's presence on an infected system. Signature-based solutions have been moderately successful in the past due to their fast ability to compare known malware characteristics—files, registry keys, or code snippets patched into files. But in the war between the malware developers and those on the anti-malware side, a number of significant software architecture improvements have been developed by the bad guys that make signature-based detections less effective than before.

As discussed in the second article in this series, a new software architecture found in many sophisticated malware packages is the addition of randomization to their compiling, installation, and sometimes even their regular processing. These randomization features change the way the malware "looks" over time. Much like a biological virus adapts to the attacks brought on by its host, the process of morphing malware's core code changes the characteristics that are used to categorize and identify it. When malware no longer "looks" like what a signature says it should, the signature no longer works for identification.

This failure associated with the signature-based approach illustrates a critical weakness in its core workflow. In order for a signature to work, a developer needs to find a copy of the new malware. They then need to reverse-engineer that software code to find the pieces that can be uniquely identified. Once uniqueness elements are indentified, the developer then needs to codify the results into a signature that is later distributed to servers and clients.

The weakness in this process has to do with the effort and timing required to get from initial detection through reverse engineering to signature distribution. This signature-based identification is highly work-intensive for an anti-malware industry that is exceptionally time-dependent. In an environment in which malware authors are constantly changing their tactics and code is morphing into new and unrecognizable forms, anti-malware companies find themselves with more work and less time.

Behavioral-Based Detection

What's interesting about all forms of malware—no matter their vector of infection, payload, or signature—is that virtually all forms of malware tend to aim towards achieving one of a limited set of goals. Financial gain is the primary goal of today's malware; additional goals tend to be one or more of the following:

- *Data destruction*—The wholesale removal of data on a system
- *Data disclosure*—This can include personal/financial data, username/password data, or configuration data for espionage purposes
- *Redirection*—Changing the behavior of a system or application to perform some other function, such as switching a user to an alternative Web site
- *Surveillance*—Spying on the activities of a user, usually to reach one of the previously mentioned goals

Thus, because the mechanisms for malware installation and processing are many while the goals are few, a different architecture for malware identification may be superior. Behavioral-based detection is that alternative architecture.

Consider the anti-malware clients that may already be installed onto servers and desktops in your environment. They are currently configured to repeatedly scan the system and running processes for the presence of software that looks like known malware. Signature updates are a daily—and sometimes hourly—occurrence to keep up-to-date. Now consider a reconfiguration of that software to instead look for any processing whatsoever on systems where that processing attempts to accomplish one of the behaviors identified previously.

In this situation, it can be significantly easier to code an anti-malware client that is always looking for certain types of behaviors. No matter how often or how much malware morphs in an attempt to evade detection, any time it attempts to accomplish its mission, that nefarious activity will be sensed by the client and prevented. It is similarly possible for clients to track the source of the inappropriate activity and begin remediation activities such as removal. Because the offending process can be more easily identified, removal can be more quickly completed. Should the correct removal procedures not be present on the system to initiate the removal, the computer remains partially protected while the bad behavior remains inhibited by the client.

Multiple Approaches Are Necessary

Not stated to this point is the necessity of multiple approaches towards resolving identification and removal requirements. Although the behavior-based approach may be superior for identifying and preventing bad behavior from occurring on the client, the signature-based approach may be better for actually identifying and removing the specific malware class and instance. Anti-malware products that incorporate multiple approaches will by default have more "vision" into the inner workings of servers and desktops than those with single approaches alone.

A few additional technologies that tack on to both of these approaches are similarly necessary for the environment that wants to get the most "bang" out of their anti-malware client dollar. Consider the following additional new methodologies that can take the identification and removal processes even further.

Kernel-Level Protection

From a software-layer perspective, the closer that anti-malware products can get to the kernel, the more likely they will have the ability to identify malware activity as it occurs on the system. When malware (rootkits being a perfect example) manages to shim itself between any anti-malware engine and the kernel, it is difficult or impossible for the anti-malware scanning engine to locate that bad code. Conversely, when anti-malware software operates at a layer directly atop the kernel, it retains the ability to see all inputs and outputs as they pass. Obviously, with the changes to the kernel with the release of Windows Vista and Windows Server 2008, this level of driver development must occur with the inclusion of Microsoft itself.

Surgical Remediation

If a malware removal tool you've attempted to use has ever resulted in the crash of the infected system, you're familiar with the need for highly tailored removal capabilities once malware has been found. When the removal process goes too far in what it eliminates from the system, to the point where the system is no longer stable, the removal system or the scripts used to instruct it are ineffective. Surgical remediation allows an anti-malware removal system to remove not only the files and registry keys where malware code has infiltrated but also the specific patches to system files. The result here is an IT environment that can easily survive an infection incident with little risk to desktop and server operations.

Pre-Boot Scanning

Rootkits are particularly difficult due to the way they infiltrate themselves into the file system and subsequently cloak their presence. One resolution with finding installed rootkits on systems when all other options fail is to look at that file system from two different perspectives. The first perspective is from the file system itself. The second is from a dismounted instance of the file system. When the file system is dismounted, the mechanisms described in article two of this series cannot function to enable the cloaking effect. By looking at the differences in results from each of these two scans, any difference found must be a set of code that has attempted to cloak itself. Using pre-boot scanning on what is effectively a dismounted file system enables the second of these two needed scans.

Executable-Layer Firewalls

Lastly, the Windows OS by default has no logic to determine what processes should and should not be executed on the system. Thus, any process that attempts to gain processor attention for execution will be run. Needed in many environments is a type of executable-based firewall on the system itself. This firewall enables administrators to identify the processes that should be run on systems. Processes that don't belong in the environment are forbidden from running. This on-system "firewall" helps prevent certain types of malware from executing on system when they aren't part of the white list of accepted programs. It also serves the secondary purpose of preventing legitimate but inappropriate and potentially risky applications from being run on company hardware such as file swapping applications, games, or other applications that can lead to a down-the-road infection.

Today's Anti-Malware Tools Must Be Sophisticated

The reason for this need of sophistication has been stated over and over in this article series: Malware itself is growing ever more sophisticated every day. For IT environments that have had success in the past using traditional troubleshooting tools, the naked eye, and the "fix it after it breaks" approach, new tools must be brought into place that prevent problems before they happen.

The anti-malware tools of yesterday, installed and run only after an event occurs, are no longer the best practice for proactive IT environments. Necessary are always-on alternatives that leverage multiple mechanisms for finding malware in all its categories and behaviors for the protection of the IT environment as a whole.