Realtime publishers
"Leading the Conversation"

The Essentials Series

Modern Malware Threats and Countermeasures

sponsored by
Sunbelt Software

by Greg Shields

Copyright Statement

Article 1: Understanding the Modern Malware Landscape

The word “spyware” is often used incorrectly to describe all types of unwelcome software that makes its way onto computers. Spyware, by definition, is only a small portion of the possible types of malware found in the wild. This terminology misuse highlights one of the central problems with truly understanding the landscape of bad code that could potentially infect unprotected servers and desktops: Each class of bad software has its own mechanisms for infection. Each arrives with different payloads, some attempting to do data destruction, others for financial gain. Virtually all arrive with a bent towards replicating themselves anywhere possible. As an IT professional, it is your responsibility to keep bad code away from systems, all the while rooting it out of those that become infected.

Malware, short for “malicious software”—and the correct term used to identify all classes of unwanted and potentially unwanted software—is a major problem in IT environments. Like all types of software, it evolves over time with new versions exploiting newly found vulnerabilities in operating systems (OSs). But malware as a class of software is changing as well. In the old days, malware was often written by disgruntled or bored code developers as a way to prove their mettle or enact revenge upon some segment of the online world. The malware of the early days often resulted in total destruction of computers and their data.

These days, malware development is big business. No longer relegated to computer hackers writing code in their mothers’ basements, malware is used most often these days as a tool for extorting money out of its victims. Fraud and scare tactics are a major priority of current malware creation. Malicious software in the form of rogue security programs are used to convince uneducated computer users to purchase removal software from the very people who wrote the malware itself. Other types collect proprietary information, such as credit card and Social Security numbers left in browser cache locations, and send it off for inappropriate use. Even more nefarious types work together in swarms for large-scale, massively parallel computing activities such as spam mail transmission, Denial of Service (DoS) attacks, and other activities.

The issue with all these types of unwanted software is the wide spread of their mechanisms for attacking systems and their payloads once an infection has occurred. The next article of this series will discuss in-depth those technologies, behaviors, and practices, and how the level of sophistication with this software has increased substantially with its monetization. In this article, let's take a look through a current list of known malware classes commonly seen in the wild. Understanding the types of malware you're up against will help you improve your ability to keep it out of your environment.

Adware

This class of malware describes software that monitors Internet use for known e-commerce sites. When a user attempts to reach a site, adware can pop up an alternate suggested site, which may or may not be legitimate. Not long ago, adware was a substantial component of all malware infections, with legitimate companies crafting specially-worded End User Licensing Agreements (EULAs) that made many infections deceptively legal. Recent litigation along with user education has resulted in the closure of many of these early organizations or their reconfiguration from legitimate to illegitimate entities. The financial gain associated with adware has lessened somewhat in recent years, with a resulting reduction in instances of this type.

Porn Dialers

Another aged malware class is the dialer or porn dialer. This software was used heavily in the days when modems were a primary mechanism for connecting to the Internet. This class of malware could silently disconnect a modem from its service provider and redial to another premium-rate telephone number. The resulting phone number charges, usually to far-removed countries, would be found by the user on their next telephone bill. Dialers have gone out of vogue as more Internet connections are broadband-based, and as telephone companies update their policies to find and eliminate the businesses that use such practices.

Rogue Security Programs

A much more modern construct, rogue security programs are a common occurrence in today's malware landscape. These software bundles download and install one or more obvious malware packages onto a targeted machine, while simultaneously installing code that alerts the user to the infection. The "rogue" in rogue security programs is named so because users are then shown how to purchase specialized software that will remove the malware. In essentially all cases, utilizing the suggested software does not actually remove the initial infector itself, keeping the system open for continued use by the attacker.

Backdoors

Backdoors are software tools that are installed to bypass existing security mechanisms present in either an OS or an application. Backdoors can be used to get around OS authentication systems, identify and replicate themselves around local networks, or send spam email from the infected host. Backdoors are often a component of other malware packages, enabling a mechanism for later download of additional malicious code.

Exploits

An exploit is a generic term used to describe any software code specifically designed to take advantage of a known weakness in OS or application code. When desired software on a computer system has not been coded properly and vulnerabilities exist on that system, exploits can be created to grant the attacker administrative privileges, complete some task, or destroy or disclose proprietary data. A common exploit in recent years has been the buffer overflow attack. This type of exploit software attempts to "overflow" existing pre-established areas of memory, pushing its malicious data into the next area of memory where it is later executed by the system. This type of attack is usually done in an attempt to acquire administrator privileges on the system, which are then used to process additional malware.

Keyloggers

Keyloggers are the original "spyware." Although, as discussed earlier, the term "spyware" is used loosely by many to describe all forms of bad software, spyware as a class of malware is actually used to "spy" on the user of a system. One way to accomplish this goal is to log every keystroke typed into that system. When all keystrokes are logged, it is possible to data mine the results to find credit card and Social Security numbers, password information, bank account information, and other personal information. Keyloggers need not necessarily be malware, as they are sometimes used in high-security corporate environments to monitor the activities of users on corporate networks.

Remote Control/Remote Access Tools

Another example of software that can be used for both legitimate and illegitimate means, remote control and remote access tools enable an individual to access areas of an infected computer remotely over the Internet. The level of remote access with this class of malware depends on the sophistication of the software itself. Some uses grant the user command-line access, while others can involve complete control of the desktop itself. Although used for illegitimate purposes, not all remote control or remote access tools are malware. Tools exist on the market today for the legitimate remote access of systems across the Internet.

Trojans

Like the fabled wooden horse in Greek mythology, a Trojan in computing relates to a piece of software that illegitimately performs some action that is different than its stated purpose. With Trojan software, the software may appear to be a legitimate software package that accomplishes a task desired by the user. However, in installing the software, it also performs some illegitimate task at the same time, destroying or exposing personal data, or any of the other payloads discussed to this point.

Trojan Downloaders

One class of Trojans commonly seen on the Internet is the Trojan downloader. A type of Trojan, the subversive task completed by this software once installed is to silently download additional malware packages to the machine. Because the initial application is installed to the computer through legitimate means by a privileged user, the Trojan itself gains the administrative rights necessary to download, install, and execute its follow-on packages. Often, the Trojan downloader itself does not perform any further function, masking its nefarious alternative purpose and leaving the further infection to the later downloaded code.

Rootkits

A particularly nasty class of malware, today's rootkits are created to hide the presence of software code on an infected system. That hidden code can be the rootkit itself as well as additional malware that is brought down to the infected system along with the rootkit. What makes rootkits challenging to locate and eliminate is the mechanism by which they hide themselves. Digging deep into the files and file system of the infected computer, rootkits install themselves by "patching" onboard system files for the OS itself. The result of the patching enables the rootkit to intercept user requests to view files on the system. It then replaces the results provided by the system with results of its own, hiding the presence of files and folders on the system.

Worms

Worms are a class of malware with a programmed drive to replicate. Although virtually all malware is written and distributed to spread itself across computers, worms include the code necessary to replicate themselves directly from the infected computer. The difference between malware that exhibits worm-like behaviors and other types of malicious code is the ability of the infected machine itself to spread the malware further. Worms can arrive on-system with a payload to accomplish some task like those discussed to this point, or can be written with no other reason than to simply replicate themselves.

Viruses

Along with worms are viruses. These common forms of malware are different than worms in that they actively attempt to infect files on-board a system. The infection can involve the wholesale replacement of a file, or more likely, the injection of malware code into the file itself. This injection of code into existing files allows the virus to run on the infected computer without being obviously seen through common systems monitoring tools. Because of this, virus scanning utilities are required to locate and remove instances of the injected code.

Summary

The economics associated with malware have grown to make malware production big business for the underground IT industry. Although many of the early mass-infection events were highly publicized and very obvious events to the user, modern malware is usually designed to be much more subtle in its behaviors. Similar to how the Ebola virus is significantly more virulent and deadly to its victims than the common cold, it also has a tendency to burn itself out rather quickly while the common cold remains prevalent today. The same holds true with today's malware landscape. Some are designed to create big results, while most quietly infect computers with the intention of keeping their presence hidden as long as possible for maximum gain.

These days, the war between good software and bad no longer pits the individual IT person against an individual malware developer. Entire underground industries have been built that write and test code against unsuspecting systems with an eye towards profit. As such, effective tools that incorporate fast responses against all these categories are critical to ensuring the security posture of today's IT environment.

Article 2: Uncovering Modern Malware's Technologies, Behaviors, and Practices

It is one thing to understanding what malware is. Understanding how it infiltrates into your computer systems during an infection is quite another. The discussion on types of malware in the first article of this series is intended to help enlighten your understanding that malware arrives in multiple forms, each with its own mechanism of infection, ways of replicating, and payloads for accomplishing its mission. The list in the first article of this series provides insight into the types of malware you've likely found on systems or heard of through the media.

The intent of this second article is to expand beyond mere classifications of malware and focus specifically on the technologies, behaviors, and practices used by each to infect computers, hide themselves, and remain resident on computers as long as possible. As with the evolution of the malware industry, new versions of its software have grown to become significantly more intelligent in recent years. Understanding how malware behaviors have changed over time will help you realize the scope of today's modern malware landscape.

Early Attempts

In the beginning, Windows-based malware attempts could be considered relatively easy to spot with the naked eye. In comparison with the malware of today, early infectors often utilized their own individualized processes for running the activities desired by the malware creator. Having their own individual process made early malware attempts easy to detect through casually browsing the contents of the Windows Task Manager. A skilled IT professional could search through Task Manager's list of running processes on an infected computer and often find processes that seemed out of place:

- Perhaps the process was not on the organization's list of those associated with approved applications
- Processes were sometimes masked with filenames similar to those already on the system, leaving duplicate entries in the list
- Occasionally, processes were given similar though not exact names, also making them stand out to the trained eye

In any of these cases, the process of determining that a computer was actually infected was an easier process than it is today. Although the removal of the malware components could be complex, its identification could be done through traditional IT troubleshooting techniques. Applications that assisted with the identification process could be run on an as-needed basis once the infection was confirmed to remove the offending software.

Modern Trickery

Today's malware, however, has reached a level of sophistication at which its identification can no longer be completed with the naked eye. In most cases, neither can it be identified using the common troubleshooting applications traditionally kept in the IT technician's toolkit. Though the different categories of malware utilize different mechanisms of trickery to evade detection, some current common behaviors make specialized applications necessary to find and get rid of the bad software. The following sections take a look at a set of these behaviors.

> ✐ This list includes only a subset of known behaviors. With malware development in a constantly evolving state, used mechanisms are always being modified to evade detection. Effective anti-malware tools incorporate always-on and real-time or near-real time updates to ensure that clients are always scanning using up-to-date methods.

System File Patching and Process Infection

When the typical IT professional thinks of the term "patching," this usually describes the monthly process done to update operating systems (OSs) and applications with newly released vendor code. But the concept of patching is much more general than this definition. Any file—whether it be an OS file, one for an application, or a data file itself—can be "patched" or updated with new code.

In the malware environment, this patching process involves reverse engineering known-good system files and injecting customized code into the result that is specific to the malware software. This activity commonly occurs with system files such as executables or library files used for processing of certain activities. In any of these cases, consider the result. Executables and library files within the Windows OS are collections of functions that enable an activity to be processed on the system. By extending these functions to include desired malware behaviors, the "patched" system file can now operate both for its original use plus the functions needed by the malware software (see Figure 1).

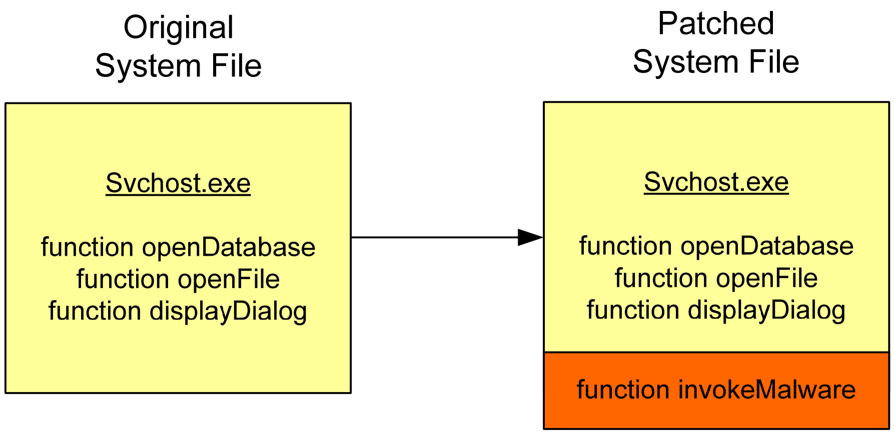

Figure 1: Once patched, the original system file can now perform malware functions.

The problem with identification in this case is that effectively nothing "new" has been added to the file system. The infected computer's list of running processes shows only expected processes, effectively hiding the malware's presence from the naked eye and common troubleshooting tools. The only way to eliminate the malware is to specifically target and remove the offending code snippets out of the file.

Code Resuscitation

Making this problem even more insidious is when multiple malware software packages are installed during a single infection event. When integrated during their code development, these packages can be augmented with the ability to monitor for the presence of each other. The resulting malware package can then monitor for the elimination of other malware packages resident on-system, similar to what Figure 2 shows. Should any of those packages be removed—typically as part of a removal activity—any remaining malware components still resident can restart or even reinstall the missing software from pieces left on-disk or downloaded from the Internet.

This assurance that malware remains resident on-system even during removal activities complicates the removal process. Standard removal tools such as manual file or registry deletions, process removal, or many of the other traditional troubleshooting tools available to IT technicians are insufficient to completely remove every piece simultaneously. Thus, missing even one component or not removing them all within a short enough time span results in the machine being re-infected within a short period of time.

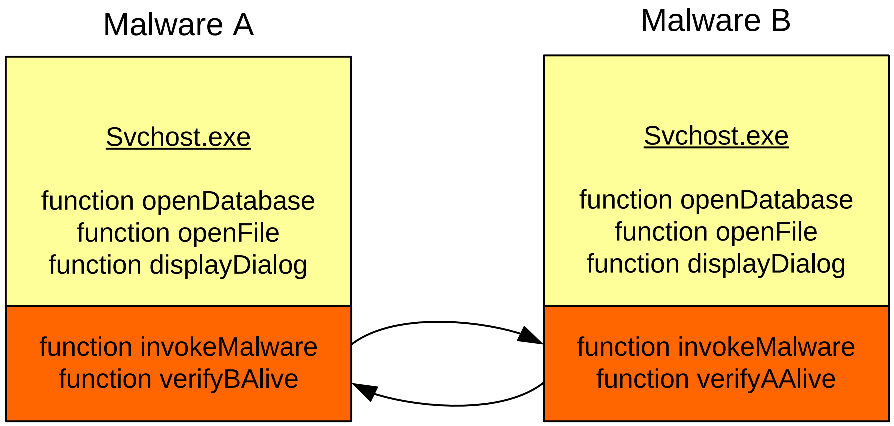

Figure 2: Malware instance A includes code that checks for the presence of Malware instance B. If not present on-system, instance A attempts to install or restart instance B.

Code Randomization

One mechanism for finding malware is to use code "signatures." These tools heuristically look for the signs of malware code resident on an infected system. When the signature of a piece of code either in its own file or attached to a system file is found, the anti-malware tool can recognize its presence and attempt a removal. Signature-based detection is a common tool used with many anti-malware toolkits. However, the exclusive use of this method of detection is not without its intrinsic limitations.

Complicating the use of signature-based identification are randomization elements built-in to the self-compiling of the malware code as it installs itself. These features enable the malware to install itself onto a candidate computer using more than one possible configuration. Due to these features, the way the malware software "looks" to the detector actually changes over time. Figure 3 shows a simplistic example of how a function name can be slightly different in malware instance A than in its original form.

This level of randomization obviously requires enough similarity between instances so that its intended mission is accomplished. Yet, the randomization effects mean that multiple signatures must be created for a single malware instance as it replicates itself across the Internet. These code randomization features now seen in some of the most sophisticated of malware packages are a source for the dramatic increase in the number of perceived malware instances in today's landscape. Only through equally sophisticated reverse engineering and behavior modeling on the part of anti-malware software and software companies can many of these malware "isomers" be identified.

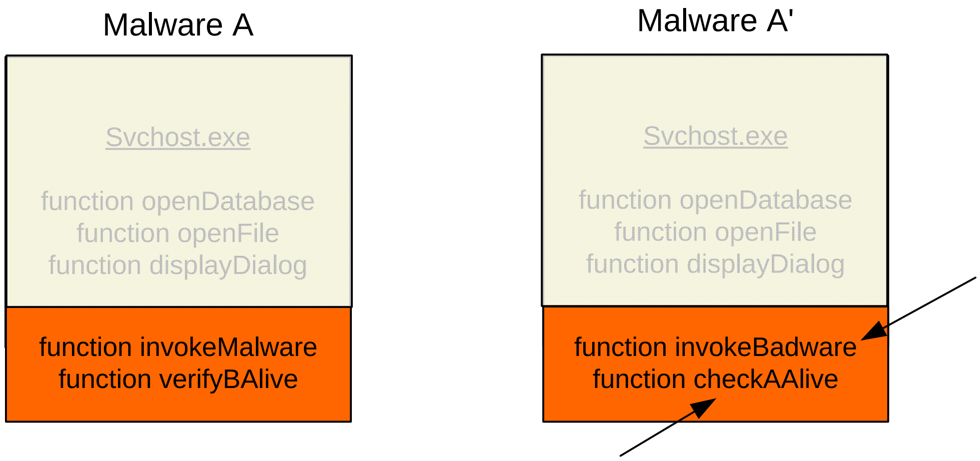

Figure 3: Subtle and randomized changes in the "look" of malware help keep it hidden from detection.

Rootkit and Cloaking Behavior

The last classification of modern behavior that is particularly dangerous due to its abilities to hide from even the native OS are rootkits. Although substantially different than the UNIX-based rootkits of yesteryear, this class of Windows malware focuses specifically on cloaking activities.

Rookits leverage some of the same file patching functionality described earlier but for a different purpose. Software patching in "regular" malware typically installs the types of functions desired for the processing of the malware's payload; rootkits are instead used to redirect system function calls through a process called "system call hooking."

Simply put, each system file is equipped with a set of addresses where it can expect to find other functions in system RAM. When it requires the use of another function, it seeks out that function at its listed address (see the top portion of Figure 4). In the case of a computer that has been infected by a rootkit, the rootkit modifies the target location for the other function to its own location in memory. This allows the rootkit to shim itself between layers of the system (see the bottom portion of Figure 4). There, it can identify what requests are being made by the system, and then alter the results of those requests as they are returned back to the function that initiated the request. Typically, the alteration is done to prevent the system from displaying the presence of files related to the rootkit itself or other malware on-system.

The end result of a rootkit infection is that the rootkit is effectively "invisible" to the system as well as to its user. Attempting to view the contents of a folder through either the GUI or command line is intercepted by the rootkit, and any trace of malware file presence is removed before the result is displayed to the user.

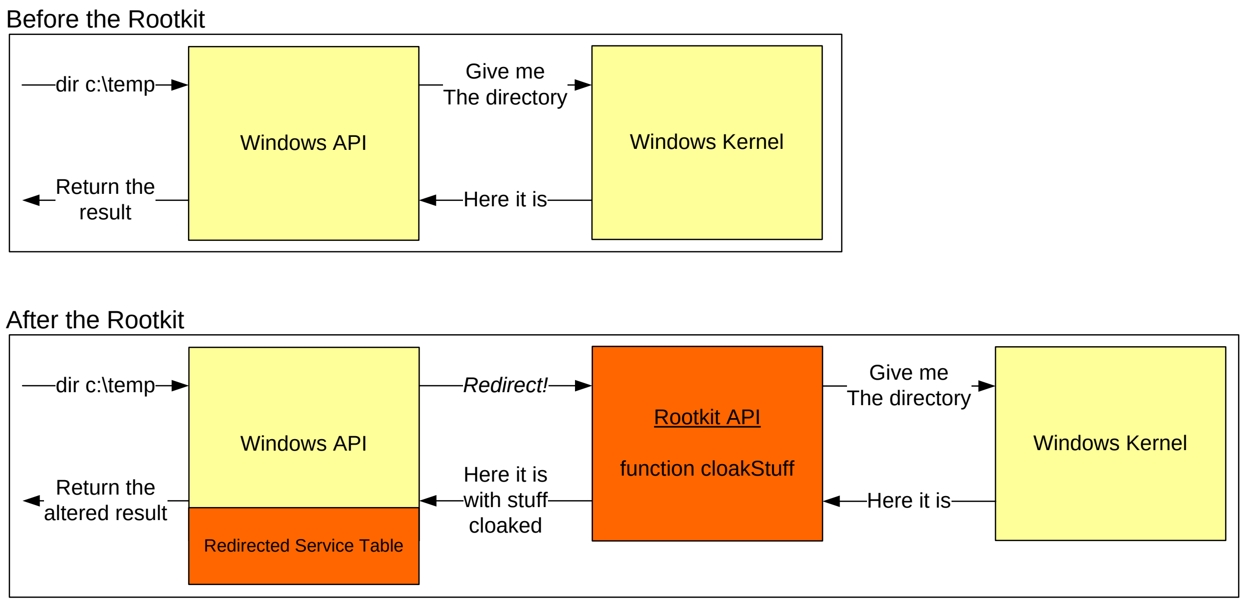

Figure 4: Upon installation, the rootkit shims itself between layers of the OS. There, it can view requests and alter associated responses without system recognition. The typical behavior is to cloak the presence of itself and other malware from the system.

Malware Is Getting "Badder"

When considering the economics associated with malware infections and the potential transfer of money as part of its development, it is easy to see why the level of sophistication is increasing over time. As the companies that write anti-malware software change their tactics, malware authors update their infection and replication mechanisms to suit. Only through effective software and diligent work in identifying these behaviors will IT organizations be able to maintain the health of their critical business systems.

Article 3: 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.

About the Sponsor: Sunbelt Software

Sunbelt Software was founded in 1994 and is a leading provider of Windows security and management software with product solutions in the areas of antispam and antivirus, antispyware, and vulnerability assessment. Leading products include the CounterSpy and VIPRE product lines, Ninja Email Security, Sunbelt Exchange Archiver, and endpoint firewall technologies. The company, located in Tampa Bay, Florida is part of the French-based Sunbelt International Group.

For more information about Sunbelt Software, please visit the company's website at: www.sunbeltsoftware.com. To learn more about current activities, products, and ideas at Sunbelt Software, please visit Sunbelt's corporate blog at www.sunbeltblog.com.

About VIPRE Enterprise™

VIPRE Enterprise was built for the administrator that is tired of earlier generation antivirus programs that have bolted on module after module and now are peddling bloatware as a security solution. Instead, VIPRE Enterprise is designed to optimize overall performance by melding anti-virus and spyware protection into a single, powerful engine. This combination of technologies offers high-performance anti-malware software that does not slow down users' PCs, consumes few system resources, and makes it easy for administrators to protect their networks. With its next-generation technology, VIPRE Enterprise means powerful virus and spyware protection against today's highly complex malware threats. It eliminates the system slowness and resource headaches of older anti-virus products. For more information on VIPRE Enterprise, please visit www.vipreenteprise.com.