Defeating Windows Personal Firewalls:
Filtering Methodologies, Attacks, and Defenses

Chris Ries
Security Research Analyst

VigilantMinds Inc.
4736 Penn Avenue
Pittsburgh, PA 15224

info@vigilantminds.com

1. Introduction

Microsoft Windows provides a variety of methods by which security software can perform network traffic filtering and other security-related tasks. However, these same capabilities can be used by malicious software, also known as malware, to tap into the operating system's network architecture in order to circumvent security software, open backdoors, and steal information. A number of articles have been published that discuss and compare the features of different software firewalls, but there are few resources that explore the filtering techniques that these firewalls use. Understanding these filtering techniques is not only useful for choosing a software firewall and troubleshooting problems with it, but it also helps to understand, detect, and prevent the malware threats that exploit inherent weaknesses in them.

This paper provides a better understanding of how network traffic is processed by Microsoft Windows operating systems, and how various security tools such as personal firewalls and host intrusion prevention systems monitor network activity to protect a system. It also explores how malware can attack the networking architecture of Windows to disable or circumvent some of these security tools and steal information. The paper focuses on the processing path of TCP/IP network traffic, although most other protocols follow a similar path.

2. Windows Network Architecture

From the time network traffic reaches a host's Network Interface Card (NIC) until the time it reaches the application, it is subject to processing at many different layers. Traffic headed to the network from an application follows a very similar processing path. Before discussing the security capabilities provided by each layer, we will first explore the layers themselves. The complete processing path of network traffic is very complex, so only the basic layers are covered. For more detailed information on the processing path see *Microsoft Windows Internals, Fourth Edition* [1]. Note that these processing layers do not have a one-to-one correspondence with the OSI model layers.

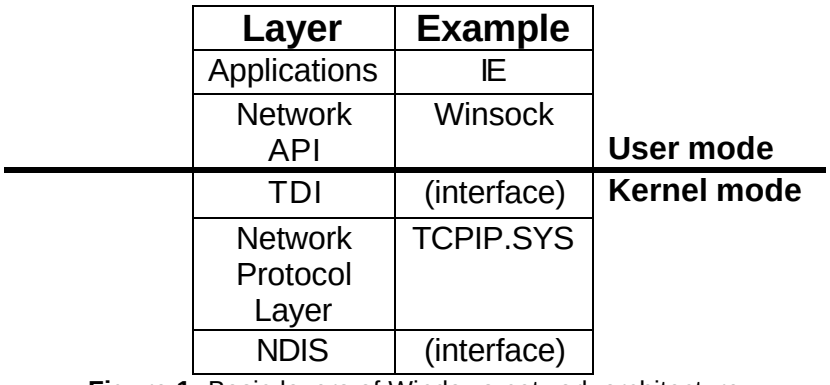

Figure 1: Basic layers of Windows network architecture.

NDIS

The Network Driver Interface Specification (NDIS) layer is the first layer that incoming network traffic passes through. The specification's purpose is to define a standard interface that higher level protocols can use to communicate with NICs. The interface provides a library of wrapper functions for communicating with the NIC card. The NDIS layer acts as an interface between the datalink layer (layer 2) and the network layer (layer 3) of the OSI model.

Network Protocol Layer

The Network Protocol Layer is the level at which network and transport layer protocols (layers 3 and 4) are implemented. For example, TCPIP.SYS, which implements the TCP/IP stack in Windows, sits at this layer. Below it, the network protocol layer communicates with NDIS. Higher layers can communicate with the Network Protocol Layer via the TDI.

TDI

The Transport Data Interface (TDI) exists in the upper edge of the kernel portion of the transport protocol stack and provides a set of functions by which clients, such as Winsock, can communicate with lower-level transport providers, such as the TCP/IP protocol driver.

Network API

The Network Application Programming Interface (API) layer operates in user mode and is the layer used by most applications for network communication. The most commonly used Network API in Windows is the Winsock API. It communicates with applications above it, and with transport providers via the TDI below it.

3. Security Capabilities of Layers

Every layer of processing provides its own filtering capabilities, each of which has its own advantages and disadvantages. If processing is done at a lower layer, more packet information, such as header flags, is available for inspection. However, at a lower layer traffic is seen before actions such as reassembly have occurred. As a result, these actions need to be duplicated, which has a performance impact. At higher layers these processing actions do not need to be duplicated, and associations can be more easily made between traffic and applications. However, higher layers do not protect the kernel code that processes packets, and can also be bypassed more easily by malware.

This section explores the security capabilities provided by each processing layer, as well as their advantages and disadvantages.

Layer	Example	Filtering	
Applications	IE		
Network API	Winsock	Winsock LSP	**User mode**
TDI	(interface)	TDI Filter driver	**Kernel mode**
Network Protocol Layer	TCPIP.SYS	Filter-hook driver Firewall-hook driver	
NDIS	(interface)	NDIS-hooking driver Intermediate driver	

Figure 2: Filtering available for Windows network layers.

NDIS

Since NDIS filtering occurs at such a low level, filtering can be performed according to data from layer 3 and above. This includes IP headers, TCP/UDP/ICMP headers, as well as any application layer data. However, since none of this data has been parsed or dissected, the filter will have to process the packet itself.

Filtering can be performed at the NDIS layer by either loading an Intermediate Driver or an NDIS-hooking filter driver. The Intermediate Driver is installed into the NDIS between the bottom layer which communicates with the NIC and the top layer which provides the NDIS interface to protocol drivers. NDIS-hooking filter drivers accomplish the same task, but they go about it a little differently. These drivers hook some of the NDIS wrapper functions so that they can intercept every call to them. For more information on NDIS-hooking drivers, see *Firewall For Windows* [2].

The advantage of filtering at the NDIS layer is that it occurs at the lowest possible layer. This makes it very difficult to bypass, and exposes less of the operating system's TCP/IP stack to unwanted traffic. This layer can be used to filter both incoming and outgoing packets, and can inspect packets moving to and from protocol drivers other than the one for the TCP/IP stack.

Because they operate at such a low level, NDIS filters inspect traffic before actions such as packet reassembly have occurred. As a result NDIS filters must perform these actions before filtering can occur. The NDIS layer also does not provide any stateful filtering, and since traffic is inspected multiple layers below the application, it is more difficult to associate traffic with

processes. Despite the drawbacks of filtering at the NDIS layer, it is the filtering method recommended for third-party software firewalls by Microsoft because it provides the ability to protect the entire TCP/IP stack.

Network Protocol Layer

There are a variety of different ways in which traffic can be filtered at the TCP/IP network protocol layer.

Filter-hook driver. Microsoft has supported the ability for a filter-hook driver to be installed since the release of the Windows 2000 operating system. The driver implements a function, which is used by the system-supplied IP filter to determine which incoming and outgoing packets to allow and which to drop. Even though the filter-hook driver is simpler to implement than NDIS layer filtering options, only one can be used at a time. It also exposes more of the operating system to potentially malicious traffic than NDIS filtering does. In addition, there is no built-in support for stateful filtering. For more information on filter-hook drivers, see [3] and [4].

RRAS Packet Filtering API. Microsoft also added the ability in Windows 2000 for user mode applications to apply traffic filtering rules using an API available with the Routing and Remote Access Service. This API has the advantage of being easy to use and accessible from user mode. However, filtering can only be performed according to limited layer 3 and 4 information such as source and destination IP address and port, as well as limited flags. More information on the Packet Filtering API can be found at [5].

Firewall-hook drivers and the Windows XP Firewall. Microsoft added support for firewall-hook drivers with the release of Windows 2000. These drivers work similarly to filter-hook drivers, the main difference being that more than one filter can be applied at a time.

With the release of Windows XP, Microsoft added a built-in firewall with stateful filtering capabilities. This firewall was designed to be an easy to use lightweight firewall, and was implemented as a firewall-hook driver called IPNat.sys. As the name suggests, the firewall is also capable of performing Network Address Translation (NAT). To configure and manage this firewall from user mode, Microsoft added the Internet Connection Firewall (ICF) API, which was replaced by the Windows Firewall API in Service Pack 2.

The Windows XP Firewall can perform filtering based on limited layer 3 and 4 information, such as IP addresses and ports. Rules can also be created to allow or deny certain network activity by specific applications. It provides incoming filtering of TCP, UDP, and ICMP traffic, with the ability to open ports by adding applications that use them to an “exceptions” list, or by creating port mappings to be used by NAT. It is important to note that the only outbound filtering that is offered through the Windows Firewall API is for the ICMP protocol.

The Windows XP Firewall was in no way intended to be a heavy-duty firewall, and therefore it is lacking in a number of areas, the most obvious of which is the ability to filter outbound traffic. The firewall also provides an API for applications to silently open ports and add programs to the exception list, as well as the capability to disable the firewall altogether. The Windows Firewall API allows filtering rules to be based on a very limited amount of information, and since it filters at a higher level it exposes more of the operating system’s TCP/IP stack to potentially malicious traffic. For Microsoft’s documentation on the Windows Firewall API, see [6] and [7].

TDI

Filtering can be performed at the TDI layer by using a TDI Filter Driver, which is installed between a TDI client, such as Winsock, and a TDI Provider, such as the TCP/IP Protocol Driver. Since this filter works above the NDIS layer and protocol driver, it is unable to protect the underlying TCP/IP stack. TDI filtering can also only be performed according to limited layer 3 and 4 data. However, it does perform filtering closer to the applications, making it easier to associate traffic with a specific process. For more information on TDI Filter Drivers, see [8].

Network API

Network API filtering can be performed based on limited network and transport layer data and on higher layer protocols. There are a variety of ways to apply filtering at the Network API level, each of which is specific to a particular Network API. Since it is the most common Network API used by applications, some filtering options for Winsock are discussed here.

Filtering can be accomplished in Winsock by hooking its functions or modifying its DLLs, but the built-in Service Provider Interface (SPI) provides a much easier method. Winsock's SPI allows developers to either create base protocols that work at the lower end of Winsock, or to create Layered Service Providers (LSP) that are plugged in between a base protocol and the Winsock API that applications communicate with. These LSPs, along with their base provider, make up a protocol chain.

Filtering can be accomplished in Winsock by creating an LSP and chaining it above the TCP/IP base protocol. The LSP can filter any incoming or outgoing connection attempts made by a Winsock application according to the local and remote IP addresses and ports. Additionally, data flowing to and from the application can be inspected and filtered.

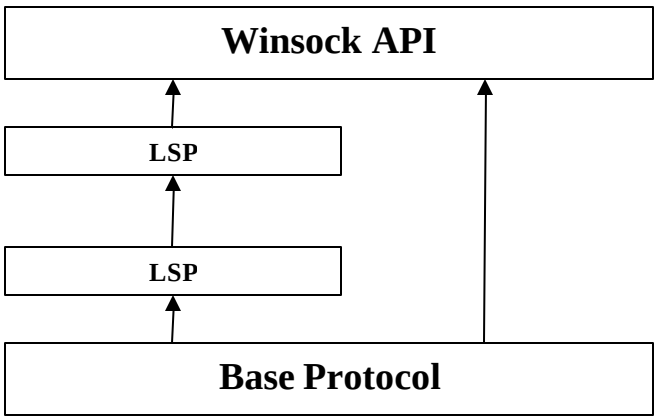

Figure 3: Winsock's SPI architecture.

Because it works at such a high level, LSPs are very limited in what they can see. They only see data flowing to and from applications that use the Winsock API, and they operate at a layer too high to inspect any layer 3 or 4 header fields from packets. LSPs can still be very effective at filtering network traffic that is more closely related to applications, such as web filtering, parental controls, or e-mail filtering. One advantage that LSP filtering has over filters operating at lower layers is the ability to perform content filtering on data before it is encrypted or after it is decrypted. Depending on which encryption scheme is in use, lower layer filters may not have access to data that LSPs can inspect. Additional information on LSPs can be found at [9] and [10].

LSPs can even be used for security purposes other than filtering, such as adding layers of encryption to protocols. For example, Zone Lab's IMSecure PRO product uses LSPs to implement encryption for secure instant messaging [11].

4. Attacks Against Filtering Capabilities

Even though these built-in filtering capabilities were intended to be used by security software to protect a system, they are not impenetrable and can still be susceptible to a variety of attacks. Additionally, they can be used by attackers and malware to hide and protect themselves from security tools and users.

For example, if security software can filter malicious traffic, then malicious software can filter benign traffic such as virus definition updates. If security software can intercept traffic to filter out malicious attacks, then malicious software can intercept traffic to steal information. In this section, we will take a look at how an attacker or malicious program can bypass some of these security capabilities, and how they can be turned against the systems that they are designed to protect. More importantly, we will also explore how security software can defend itself against such attacks.

Many of these attacks require administrative privileges to be carried out, which may make them seem to be less of a risk. It is not uncommon, however, for malware to be run with administrative privileges. Worms often spread by exploiting vulnerabilities, which in many cases provide remote administrative privileges to a system. For example, the LSASS vulnerability that Sasser exploited and the RPC DCOM vulnerability exploited by Blaster both provided administrative access to hosts. More recently, the Zotob worm exploited a vulnerability in Microsoft's implementation of Plug and Play (PNP) to gain remote administrative privileges to Windows 2000 machines. Even mass-mailing worms and other malware that use social engineering to infect machines are often run as admin, since it is often a home user running with administrative privileges that executes the malicious code.

If malware is run with administrative privileges, it may seem pointless to attempt to circumvent it when it can just be disabled. However, it may be advantageous for malicious individuals or software to circumvent filtering instead of disabling it, because this makes its presence on the machine less apparent. Additionally, it is not always possible for malware to simply disable filtering from user land. If the filtering is performed by a kernel driver that cannot be easily unloaded, then malware may not be able to disable the filtering without executing code with ring 0 (kernel) privileges. Depending on what other protection mechanisms the security software has built into it, executing code to disable filtering may be very complex, or even impossible. As a result, it may be much easier for the malware to simply bypass the filtering instead of disabling it.

Circumventing Filtering

Malware developers may want to circumvent filters on a system for a variety of reasons. If there are outbound filters in place, then a worm would need to bypass these filters in order to spread to other machines. If malware tries to open a backdoor either by connecting to an IRC channel or binding a shell to a port, then filters will need to be bypassed as well. One of the easiest ways that malware can accomplish this is by disabling the filtering altogether. However, as we previously mentioned, this is not very stealthy, and is not always easy to accomplish. Therefore, it is worth exploring the circumvention of filtering without disabling it. Methods that malware can use to disable filtering altogether are explored later in this section.

If malware is running with administrative privileges, then the simplest way that it could circumvent filtering is to open up a hole in the firewall. Most software firewalls do not provide a documented API for applications to do this. However, with the firewall built into Windows XP, an application can silently open a port. This is only possible with the Windows Firewall because it was meant to be lightweight and provide limited notifications so as not to bother the user. Most firewalls do not allow applications to silently open up ports, because it would provide malware with an easy way around them.

Another popular method used to circumvent all but NDIS-level firewalls is to simply bypass the firewall and communicate directly with lower layers of the protocol stack. For example, if a firewall is implemented as a TDI filter, then the malware can simply implement its own lightweight protocol and speak directly to the NDIS interface. This "parallel stack" method is nothing new, and has been explored in-depth in the past [15][16]. The best way to combat this category of circumvention is to either implement filtering at the lowest possible interface (NDIS), or to monitor any protocol drivers that are loaded or unloaded by hooking kernel functions. Both of these methods involve providing kernel layer protection, which a number of heavy-duty firewalls and Host Intrusion Prevention Systems (HIPS) already utilize.

Many of the filtering methods provided by the Windows protocol stack can also be used to implement backdoors. For example, a rootkit could install a filter-hook driver that drops packets matching particular criteria, but also extracts commands from those packets to execute on the machine. This approach was implemented in the "Thorny Path" kernel backdoor, which is discussed in article [17]. A similar backdoor could be implemented as an NDIS-hooking driver or NDIS intermediate driver. Such a backdoor would be difficult to detect, because the traffic it sent and received would not be visible to layers above NDIS. Running a packet sniffer such as ethereal [18] on the compromised host would not show the traffic, since Winpcap's protocol driver works above the NDIS at the network protocol layer.

All of the circumvention methods discussed so far require the malware to run with administrative privileges. If malware is not run with administrative privileges, it cannot open up holes in the firewall, install its own protocol driver, or install any NDIS drivers. It may still be possible, however, for malware to sneak through the firewall if it is not running with administrative privileges. A well-known method that can be used to accomplish this is for the malware to inject itself into a trusted process [19]. For example, if the firewall already has a rule to allow Internet Explorer to make outbound connections, then the malware can inject itself into an Internet Explorer process to gain the process's privileges. There are a large number of trojans, spyware programs, and worms that inject themselves into processes that are usually trusted such as Explorer.exe. Aside from bypassing filtering, this injection technique also makes detection of the malware more difficult, because it is not visible in the process list. Some variants of the Lovgate and Korgo worms use this technique [20][21]. This attack has been around for a long time, and many software firewalls have already addressed it [22]. To combat this attack, security tools can monitor API calls that are used for process injection. However, this attack is still effective against a number of popular personal firewalls.

A second method that malware can use to gain the privileges of a trusted application is to use the application itself. If a trojan needs to download files to install on an infected system, it can just use the system's trusted web browser to download these files. If it needs to transmit stolen information to a remote attacker, it can simply send the information as part of an HTTP request. This type of attack can be countered by monitoring API calls that are used to create processes, and to check the parent process of trusted applications to make sure that it is a trusted process.

There are variations of this attack, however, that can bypass these countermeasures. For example, a malicious process can first create a process such as cmd.exe, and then have the cmd.exe process create the process for the trusted application. To detect this variation of the attack, a firewall needs to not only check the trusted application's parent process, but also check every parent process above the trusted application. Another variation of the attack is for a malicious process to launch a trusted application and then terminate itself. If the firewall attempts to check the trusted application's parent process, it will not be able to find it. Some firewalls will respond to this situation by allowing the trusted application to execute instead of notifying the user that the parent process does not exist and that this is a potential attack. These attack variations have been implemented in a number of tools, and many firewall vendors have

responded by adding protection against them [23].

Disabling Filtering

Most malware today does not need to be sophisticated in order to be effective. As a result, most malware simply tries to disable any security measures on a system instead of circumventing them. After all, if a user is naïve enough to execute an email attachment named 'important-details.pif', they will probably not notice that their firewall or anti-virus is turned off, and even if they do they will not interpret this to mean that their machine is infected.

Most malware applications attempt to disable security software by terminating its processes. Although this may work for some security products, firewalls that work at lower layers such as TDI and NDIS filters require more than this to be disabled. For example, disabling the drivers themselves by unloading and uninstalling them would be more effective. An RRAS packet-filtering driver can also be unloaded in order to disable it. In order to fend off these attacks, the drivers should make themselves difficult to unload and the kernel functions used to unload the drivers should be monitored.

Disabling the Windows XP Service Pack 2 Firewall is easy to accomplish. Just as Microsoft provides API functions for programs to open up holes in it, they also provide functions for disabling the firewall altogether. This is not a flaw as much as it is the result of the design goals of an easy to use firewall. This firewall can also be disabled by unloading and uninstalling the ipnat.sys driver. If this is done, it is not possible to re-enable the firewall until the driver has been installed and loaded again.

Adding Filters to Block Access to Anti-virus Sites

In order to gain a foothold on infected systems, most malware attempts to disable anti-virus scanners. This is usually accomplished by either terminating processes with names associated with well known security tools, or by adding entries to the hosts file that point security-related domain names at 127.0.0.1. The hosts file is the first place that Windows checks when trying to find an IP address associated with a hostname. Therefore, by adding these entries to the hosts file, tools such as anti-virus scanners become unable to download updated signatures, because the hostname of the signature download site points to the loopback IP address. Without new signatures, the scanner cannot detect the new malware, which allows it to safely remain on the system. As effective as they may be, these methods of disabling anti-virus tools are easy to detect. A user or scanner simply needs to check the hosts file and running process list. A variety of tools, such as Hijackthis [12], make discovering and removing such malware easy.

An alternative approach that malware could take is to add filters that block access to the security sites. This is just as effective as modifying the hosts file, except that it is much more difficult to detect, especially considering that filtering can be applied at so many different layers.

The Fantibag family of trojans uses this approach to block access to anti-virus sites [13][14]. It uses the RRAS packet filtering API to accomplish this task, probably because the API is easy to use and can be reached from user mode. The trojan also uses the API to block access to Microsoft sites, preventing infected computers from downloading the latest patches. In order to detect and prevent this attack, security software should monitor the APIs that are used to apply filtering. It should also ensure that no attempts are made to load filtering drivers at any layer.

Stealing Information

Given the dramatic increase in spyware and adware over the past few years, it is obvious that one of the primary goals of malware developers today is to gather and steal information from compromised systems. Usually this is accomplished by searching the registry and files, and by monitoring keystrokes. However, there are a number of spyware, adware, and trojan programs in the wild that insert themselves into the Windows networking stack to intercept data as it passes through. There are rootkit and bot variants, for example, that install Winpcap on infected machines, which allows them to run packet capturing utilities. Malicious code can also install TDI or NDIS hooks to intercept traffic as it passes through.

One method used by a wide variety of malware for nefarious purposes is Winsock LSPs. Various adware applications, such as SpywareNuker, install themselves as LSPs to monitor a user's web surfing habits [24]. Since they are installed below the application layer, they can monitor users despite what browser is being used, as long as the browser uses Winsock. Some trojans, such as Kika [25], monitor traffic passing to and from Winsock applications for login names and passwords by searching for keywords that are used in protocols right before logins. There are even commercial spyware products that install themselves as LSPs in order to monitor user's activities. For example, Tropsoft sells a product called Winvestigator that uses LSPs to monitor users' network activities [26]. LSPs can even be used to set up a backdoor to a system, which is discussed in article [27]. Since firewalls are primarily responsible for filtering incoming and outgoing data, most of them do not monitor the installation of LSPs, so they will not detect or prevent these types of threats.

One problem that often arises with malware that uses LSPs is that removal can break the LSP chain, and therefore disrupt network access for any applications using Winsock. In these cases tools such as LSPFix [28] can be used to repair the problem. Such tools can also be used to detect any malware that has inserted itself as an LSP.

For a more comprehensive list of adware, spyware, and trojans that use LSPs see [29].

5. Protection From Filtering Attacks

Many of the attacks discussed in the previous section pose a serious risk. They could allow an infected system to propagate a worm to other hosts on a network, or facilitate an attacker in gaining a foothold on a system and keeping it secret. Therefore, it is necessary that software firewalls implement protection against these types of attacks. Protection not only involves filtering network traffic, but also includes monitoring processes' behavior, protecting the registry, drivers, and system files, and protecting the security software itself.

In this section, the countermeasures mentioned briefly in the previous section are explored in greater detail. The technical details of some methods that firewalls can use to prevent these attacks are examined. Additionally, countermeasures that system administrators and users can take are discussed.

NDIS Layer Filtering

In order to mitigate the risk of bypassing attacks, filtering should be performed at the NDIS layer. If filtering is performed at a higher layer, it is usually a trivial task to bypass it by using the parallel stack technique. Higher layer filtering also exposes at least part of the protocol stack to potentially malicious traffic. NDIS filtering is not perfect, but many of its weaknesses can be remedied by performing additional filtering at higher layers. Despite its weaknesses, NDIS layer filtering is still the most effective single approach to packet filtering in Windows, which is why Microsoft recommends it over any other filtering method.

The level of security provided by the NDIS layer comes with the price of a more complex implementation, because tasks such as reassembly that occur higher in the processing path need to be duplicated by the filtering code. The result of this complexity is greater performance overhead when it is implemented properly. In order to simplify the implementation and reduce overhead, some filtering can be performed at higher layers in addition to the NDIS layer. Additional higher layer filtering will eliminate the need to duplicate processing actions, and could also provide access to data that may not be visible at the NDIS layer due to encryption.

Driver Protection

Many of Window's packet-filtering techniques are implemented as filtering drivers. As a result, it is necessary to protect these drivers from being unloaded and uninstalled by malware in an attempt to disable the filtering. Many of the attacks discussed in the previous section are also accomplished by installing drivers. For example, attackers can install a protocol driver to bypass higher layer filtering, or can install an NDIS driver to act as a backdoor. Since a driver's code is executed with kernel privileges, rootkits often install drivers in order to attack various parts of the kernel, such as the Interrupt Descriptor Table [IDT], System Service Descriptor Table [SSDT], and various device drivers [40].

To combat these types of attacks it is necessary to monitor the loading and unloading of drivers. Drivers can be loaded and unloaded using a number of different methods, and all of these methods should be closely watched. For example, various functions can be used to install, load, unload, and uninstall drivers, such as CreateService(), StartService(), ControlService(), DeleteService(), NtLoadDriver() and NtUnloadDriver(). These functions, as well as the registry keys that they store information in, should be monitored to protect security-related drivers and to prevent malicious drivers from sneaking into ring 0.

To help prevent malicious code from unloading them, filtering drivers can also incorporate protective measures into their unloading routines, or they can simply not implement an unload routine. This will not prevent a filtering driver from being unloaded, but it will make it more difficult

because the driver cannot be removed from memory without a reboot. To prevent the filtering driver from being unloaded, additional measures, such as registry and API call monitoring, can be performed.

API Call Monitoring and Kernel Hooking

A number of the attacks discussed earlier in the paper are performed using Win32 and Native API calls. For example, an LSP is usually installed by either using functions provided by the Winsock API, or by manipulating the registry using Win32 API functions. Process launching and injection techniques used to hijack trusted applications are also usually carried out using a number of different Win32 API calls. For example, one method of performing DLL injection is to use the CreateRemoteThread() API call to load a DLL into a process' memory and begin execution of its code. Some of the filtering methods available in Windows also provide an API for adding filtering rules, and malware can use these APIs to block access to anti-virus sites.

Security software can detect and prevent many of these attacks by intercepting calls to the API functions used to carry them out. For example, a software firewall can monitor Winsock's API functions associated with installing LSPs, and prompt the user every time these functions are called. The user can then determine if they trust the application to perform the given action, and can allow or deny the action accordingly.

API calls are usually intercepted by hooking the API functions either in the binary file or at run-time in memory. It does not make sense for security software to modify every binary on the system, so performing hooking at run-time is a better approach. Hooking should also be done at the lowest possible level, so that it cannot be bypassed by malware. For example, there are a variety of Win32 API calls that can be used to create a process, such as WinExec(), CreateProcessA(), CreateProcessW(), and CreateProcessAsUser(). However, all of these different functions use the same Native API calls, such as NTCreateProcess() and NtCreateThread(). If security software were to only hook the Win32 API calls, then malware could bypass the hooks by calling the underlying Native API calls, which are accessible in user land via NTDLL.DLL.

The lower the interception is performed the harder it is to bypass. For the best level of security, interception should take place at the lowest possible level in the kernel. This can be accomplished by hooking the SSDT, IDT, or by hooking various routines in device drivers. For more information, see [40].

Call hooking is not only useful for preventing filtering attacks, it can also be used to protect the security software itself. It can be used to prevent firewall-related processes from being terminated, and registry keys associated with the firewall from being modified or deleted. It can also be used to protect security-related files such as the policy file. Of course, if call hooking is being used to protect the host and security software itself, then measures must be taken to prevent the removal of these hooks. How this is accomplished varies depending on the hooking method that is being used.

Administrator and User Countermeasures

Good security practices are important to help prevent intrusion and compromise in the first place. Strong passwords, up to date patches, and up to date anti-virus signatures are all proactive measures that can mitigate the risk of the attacks discussed in this paper. Regularly scanning a computer with anti-virus and anti-spyware tools can also help to detect and stop these attacks. All security software installed on a system should be configured for the highest level of security. For example, a number of the tested software firewalls provided protection against some of the attacks, but the protection was disabled by default. Configuration and preferences should be

checked to ensure that the most secure options are enabled.

Additionally, administrators and users can choose security software that protects against these attacks, or they can use supplementary tools to detect and prevent the specific attacks that affect their software firewall. A tool such as Processguard [30] can be used to protect against many of the launching and injection attacks used by various malware applications. Additionally, tools such as HijackThis [12] and LSPFix [28] can be used to analyze installed LSPs for possible malware. It is much more effective, however, to use software that protects against these attacks than to try to make combinations of different tools to cover all of the bases. If a particular vendor does not provide adequate protection against these attacks in their personal firewall, they may offer another product that does, such as a HIPS.

6. Conclusion

Microsoft Windows provides a variety of different methods for monitoring and filtering network traffic, each of which has its advantages and disadvantages. Although there is no perfect method, it usually makes sense to deploy filtering at multiple layers. Filtering at lower layers makes it more difficult for malware to circumvent the firewall, while higher layer filtering usually allows easier associations between network traffic and processes.

Although these methods were designed to be used for well intentioned purposes, malicious code can also use these same capabilities for nefarious purposes. They can be used to bypass or disable security software, implant backdoors into the system, and steal information.

Most personal firewalls are very effective at preventing external attackers from successfully compromising a system. However, detecting and preventing local attackers and malware from circumventing outbound filtering is an issue that is less frequently addressed. To combat these types of threats, security software must extend its responsibilities beyond traffic filtering. It needs to audit process' behavior, monitor certain parts of the kernel, and implement self-defense mechanisms. While some personal firewalls do provide this level of security, these protection mechanisms are more common in HIPS products, and many of the vendors that do not incorporate protection into their personal firewall products do so in their HIPS.

In their next generation of Windows Operating Systems, called Vista (codenamed Longhorn), Microsoft has added the Windows Filtering Platform, which includes improved access to the packet processing path [31]. This architecture has been designed to allow third party firewall, anti-virus and intrusion detection software to be installed into the protocol stack much more easily. Like the current architecture, it also allows filtering to be performed at multiple layers.

7. Firewall Tests

Attack	Category	Description	Tool(s)
Process Injection	Bypass	This attack is carried out by injecting code (usually by injecting a DLL) into a trusted application's process in order to take advantage of its privileges. If an application such as Internet Explorer has been granted network access by the firewall, then malware could inject a DLL into IE and bypass the firewall. This attack can be prevented by monitoring API calls used to inject code into processes, such as OpenProcess() and CreateRemoteThread().	PCAudit2, Firehole, Thermite, all available at http://www.firewallleaktester.com.
Launching	Bypass	An application can perform this attack by launching a trusted application in order to use its privileges. For example, if Internet Explorer has been granted network access by the firewall, then a trojan could use IE to download files from the Internet, or to report back to a remote host. This attack can be prevented by monitoring API calls that are used to create processes. If a process attempts to launch a trusted application, then the firewall can notify the user of this activity.	WallBreaker, available at http://www.firewallleaktester.com.
Launching - Indirect	Bypass	This is a variation of the launching attack, where malware uses cmd.exe or explorer.exe to indirectly launch an application that is trusted by the firewall.	Wallbreaker, available at http://www.firewallleaktester.com.
Launching - Timing	Bypass	This is a variation of the launching attack, where a malicious process launches a trusted application, then terminates its own process. When the firewall checks the process that has attempted to launch a trusted application, it will not be able to find the process since it has been terminated. As a result, some firewalls will not notify the user of the launch.	Ghost, available at http://www.firewallleaktester.com.
Parallel Stack	Bypass	This attack involves attempting to bypass filtering that is performed at higher layers by communicating directly with the NDIS interface. If the firewall performs filtering at a layer higher than NDIS, then it will not be able to see this communication. The attack works by using its own Network protocol layer driver, so it could be prevented by either monitoring the loading of protocol drivers or performing filtering at the NDIS layer.	Winpcap and Nemesis, available at http://www.winpcap.org and http://www.packetfactory.net.
Unloading Drivers	Disable	This attack is performed by attempting to disable any drivers that are associated with the firewall using a simple driver loading/unloading utility. Note that there are more complex methods of unloading drivers that could be more successful than the simple method used for this test.	Drvloader, available at http://www.toolcrypt.org .
LSP	Stealing Information	This attack attempts to install an LSP, which could be used for a variety of malicious purposes such as information theft. This attack could be detected or prevented by monitoring the installation of LSPs, which can be accomplished by hooking API calls or watching the portion of the registry that stores LSP information.	Komodia LSP, available at http://www.komodia.com.

Table 1: Resources for Attacks that can be performed against Windows software firewalls.

Firewall	Version	Process Injection	Launching			Parallel Stack	Unloading Drivers	LSP Insertion
McAfee Personal Firewall Plus	6.1.6144	Fail	Fail	Fail	Fail	Fail	Pass	Fail
Norton Personal Firewall	8.0.5.14	Fail	Fail	Fail	Fail	Fail	Pass	Fail
Sygate Personal Firewall	5.6.2808.0	Pass*	Pass	Fail	Fail	Pass	Pass	Fail
Tiny Desktop Personal Firewall	6.5.92	Pass	Pass	Pass	Pass	Pass*	Pass	Pass
ZoneAlarm Pro	6.0.631.2	Pass	Pass	Pass	Pass	Pass	Pass	Fail
Windows Firewall	XP SP2	Fail	Fail	Fail	Fail	Fail	Fail	Fail

Table 2: Results of performing attacks listed in Table 1 against various software firewalls.

A 'Pass' rating implies that the firewall correctly identified and protected against the particular attack. 'Fail' implies that it did not. For tests that used multiple tools, such as 'Process Injection', the firewall needed to protect against all tested tools to pass. Results marked a (*) imply that in order for the firewall to pass the test, it was necessary to enable options that are not turned on by default.

Vendor Responses

All vendors were notified of their product's test results in advance of the release. A summary of each vendor's response is provided below.

Since Microsoft does not aim to provide a level of security in their Windows Firewall that would stop these attacks, they were not concerned with their results.

Sygate acknowledged that their firewall was susceptible to some of the attacks, and stated that they offer a HIPS product that provides a higher level of security and addresses these issues.

Tiny Software acknowledged that their Raw Packet protection provides defense against the parallel stack attack.

Zone Labs stated that they are planning to add LSP protection in future releases of ZoneAlarm.

While Symantec (Norton) and McAfee did not comment on the results of their products, they do offer Host IDS and IPS products that may provide protection against the attacks.

References

[1] Russinovich, M.E., Solomon, D.A. *Microsoft Windows Internals, Fourth Edition: Microsoft Windows Server(TM) 2003, Windows XP, and Windows 2000.* Microsoft Press; 4th edition. December 8, 2004.
[2] http://www.ntkernel.com/w&p.php?id=14
[3] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/network/hh/network/fltrhook_65cd5716-ffa7-4fc5-b054-c3adf97344d8.xml.asp
[4] http://www.codeproject.com/internet/FwHookDrv.asp
[5] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/rras/rras/packet_filtering_reference.asp
[6] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/ics/ics/windows_firewall_start_page.asp
[7] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/network/hh/network/firewall_5932d4d9-b61d-472d-8f27-c6a1d26ddc36.xml.asp
[8] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/network/hh/network/303tdi_ffb2fd6d-d03a-4dec-95af-fb9116e151aa.xml.asp
[9] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/winsock/winsock/transport_service_providers_2.asp
[10] http://www.microsoft.com/msj/0599/LayeredService/LayeredService.aspx
[11]http://www.zonelabs.com/store/content/catalog/products/sku_list_imsp.jsp?lid=imlink
[12] http://www.spywareinfo.com/~merijn/downloads.html
[13] http://www.f-secure.com/v-descs/fantibag_b.shtml,
[14] http://securityresponse.symantec.com/avcenter/venc/data/trojan.fantibag.a.html
[15] http://www.dslreports.com/forum/remark,7321041~root=security,1~mode=flat
[16] http://www.dslreports.com/forum/remark,1003666~mode=flat~days=9999
[17] http://www.phrack.org/phrack/62/p62-0x06_Kernel_Mode_Backdoors_for_Windows_NT.txt
[18] http://www.ethereal.com
[19] http://www.phrack.org/phrack/62/p62-0x0d_Bypassing_Windows_personal_fw_with_process_infection.txt
[20]http://securityresponse.symantec.com/avcenter/venc/data/w32.hllw.lovgate.g@mm.html
[21] http://securityresponse.symantec.com/avcenter/venc/data/w32.korgo.i.html
[22] http://www.sygate.com/alerts/Outbound-Blocking.htm
[23] http://www.firewallleaktester.com
[24] http://securityresponse.symantec.com/avcenter/venc/data/adware.spywarenuker.html
[25] http://securityresponse.symantec.com/avcenter/venc/data/backdoor.kika.a.html
[26] http://www.tropsoft.com/winvestigator/
[27] http://www.securityfocus.com/infocus/1840/2
[28] http://www.cexx.org/lspfix.htm
[29] http://castlecops.com/LSPs.html
[30] http://www.diamondcs.com.au/processguard/
[31] http://www.microsoft.com/whdc/device/network/WFP.mspx
[32] http://www.microsoft.com/technet/community/columns/cableguy/cg0605.mspx
[33] http://www.ndis.com/papers/winpktfilter.htm
[34] http://www.securityfocus.com/infocus/1839/2
[35] http://www.informit.com/articles/article.asp?p=130716
[36] http://www.eeye.com/~data/publish/whitepapers/research/OT20050205.FILE.pdf
[37] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/randz/protocol/windows_tcp_ip_stack.asp
[38] http://www.rootkit.com/newsread.php?newsid=219
[39] http://www.rootkit.com/newsread.php?newsid=253
[40] Hoglund, G., Butler, J. *Rootkits : Subverting the Windows Kernel.* Addison-Wesley Professional. July 22, 2005.