realtimepublishers.com®

The Definitive Guide™ To

Securing Windows in the Enterprise

SCRIPTLOGIC

Don Jones

Copyright Statement

Chapter 8: Securing the Network

Of course, the network—the backbone upon which the enterprise functions—is a major source of potential security problems in a Windows—or any other—environment. Securing the network is a critical requirement in order for the overall enterprise to be secure, so this final chapter will focus on often-overlooked network security problems and solutions.

Security Through Architecture

Many techniques exist for securing the network. Some of the common techniques are probably in use on almost every corporate network in the world:

- Use firewalls to protect the network from Internet-based attacks
- Use common wireless security mechanisms, such as Wi-Fi Protected Access (WPA), to secure wireless connections
- Use authentication and authorization to protect network-attached resources, such as file servers, from unauthorized access

Unfortunately, one area that can provide excellent security but is commonly overlooked is securing network architecture. The reason more secure architecture techniques are often overlooked is that the intranet is often seen as homogenous when it comes to trust, access, and security. In other words, once you're in the intranet, you can do whatever you want. Of course, nothing could (or at least should) be further from the truth; *most* security problems come from *within* the intranet, completely unaffected by the firewalls and other technologies meant to protect the network from attack. It's these "inside jobs" that can be prevented by better security architecture.

Resource Clustering

A good practice is to start an architectural examination of the network by clustering resources, at least logically if not physically. As Figure 8.1 shows, this process involves grouping resources that have common, or at least similar, security and communications requirements. For example, file servers rarely need to communicate with one another but always need to communicate with client computers. They have common security needs—perhaps authenticating users to an Active Directory (AD) domain—and are often administered in the same way. Extranet Web servers need to be accessible to select Internet users (usually company business partners of some kind), and may also need to be accessible to intranet users. Figure 8.1 shows these resources being grouped into common network segments or subnets—a first step toward a more secure architecture.

✎ A network *segment* is usually defined as a group of network hosts connected by a common medium or device, such as all devices connected to a particular hub. However, with the prevalent use of switches and VLANs, a *subnet*—a contiguous IP address space—is more useful for breaking the network apart. For example, all file servers in the network might be attached to a single subnet and share a contiguous address space, even though they might be located in separate sections of a building or (in particularly creative VLAN architecture) across WAN links.

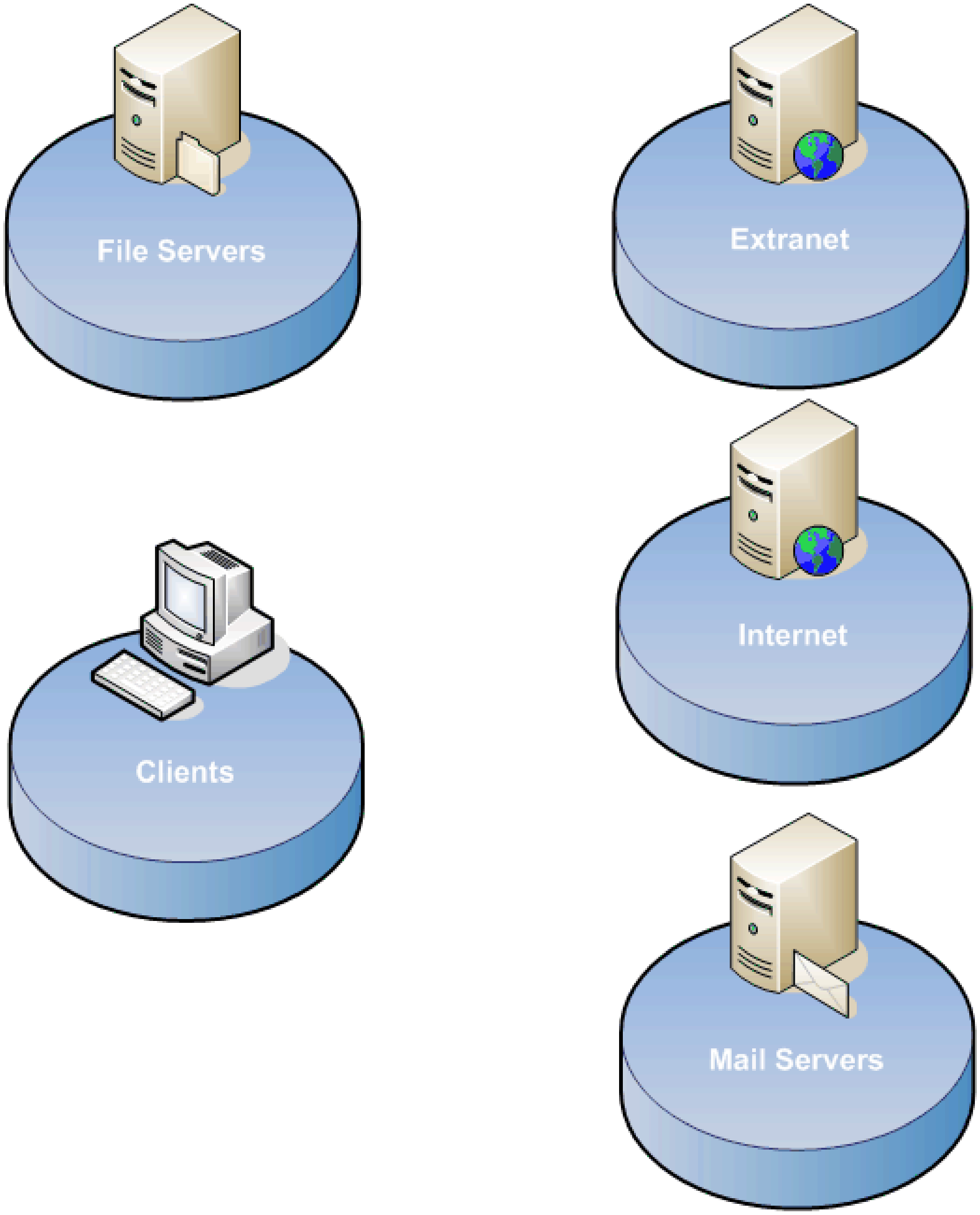


Figure 8.1: Grouping common resources together.

The primary reason behind this resource grouping is to get resources with common security and communication requirements into common, independent address spaces. The reason to do so is that network security is relatively easy to manage on a per-subnet basis, through segmentation, which we'll explore in the next section. Figure 8.2 shows a more detailed, real-world example of resource grouping, in which resources are more likely to be sub-grouped by access requirements.

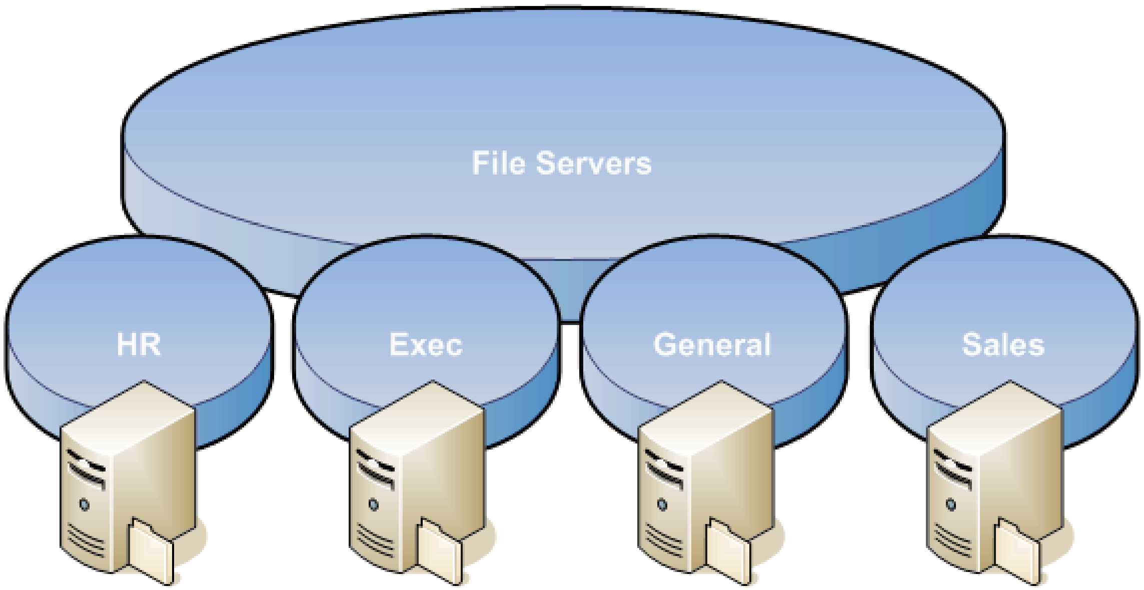


Figure 8.2: Grouping resources by access requirements.

This example only deals with file servers, which is typically a large number of servers and other resources in an organization. Here, the resource grouping of file servers has been further broken down into those accessed by Human Resources (HR), company executives, general company users, and a sales department's users. This breakdown is important because of their unique security requirements and communications needs. For example, sales department employees might have no need to communicate with HR file servers, making the separation between these servers useful.

Does each sub-grouping need a distinct IP address subnet? Perhaps. Certainly, as you'll see in the next section, doing so can provide for more granular security. Besides, as most organizations use private IP addresses on their intranets, there is no shortage of IP addresses to work with, so why not break down the network into smaller, manageable chunks?

Segmentation

Once your resources are grouped into distinct subnets, you can begin the process of *segmentation*, which means restricting access to each group's independent subnet based on the security and communication needs of that group. For example, consider the network diagram in Figure 8.3.

This example shows departmental subnets (which might include file servers and other department-specific servers) connected to a common Clients subnet. Additional subnets provide access to resources such as an extranet and email, which are shared across the organization rather than being specific to a department. The important feature here is that the subnets are connected by firewalls.

> ✎ Of course, you don't *need* to use actual firewalls; any device capable of providing basic firewall-like functionality—such as a router—can get the job done.

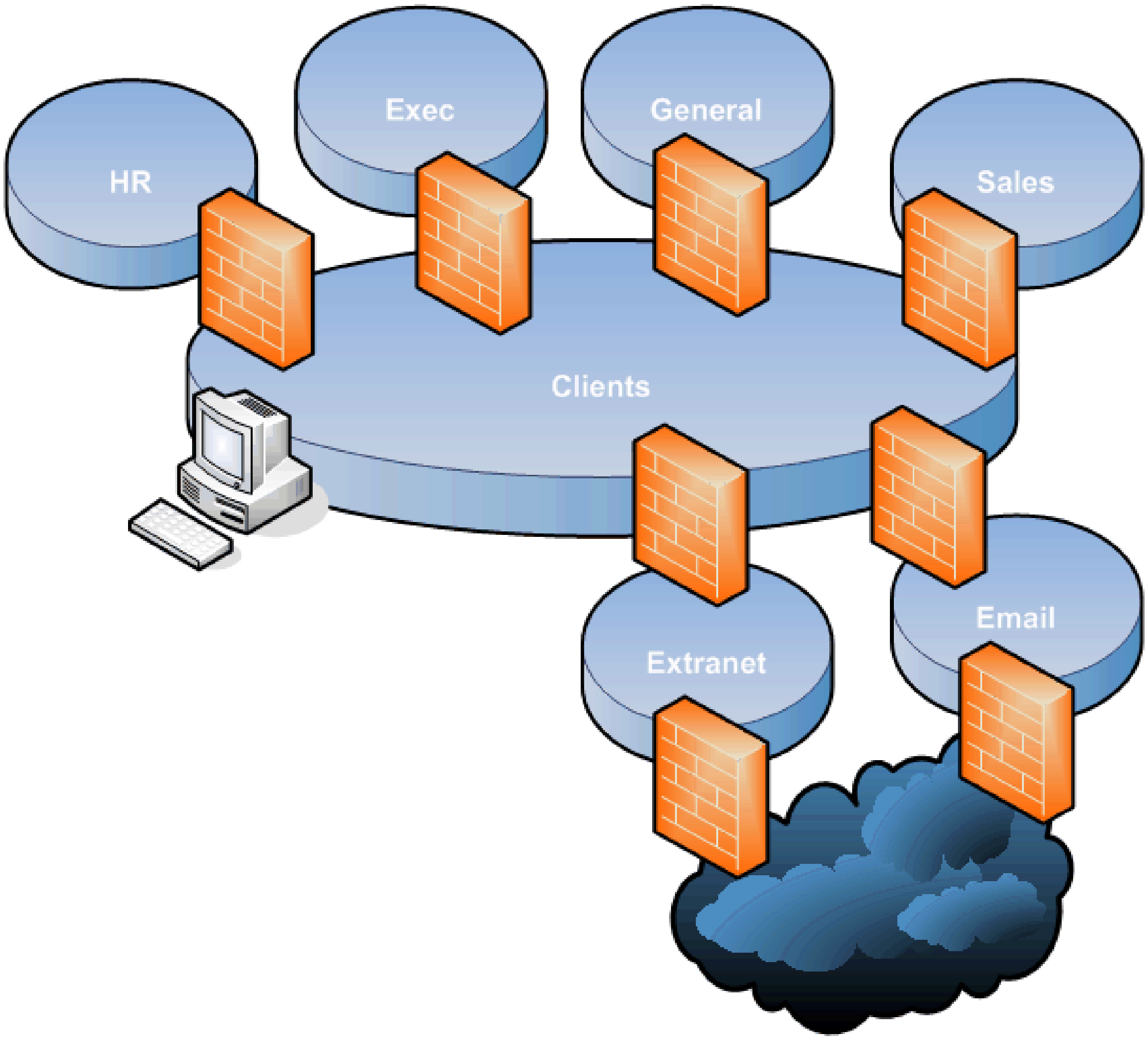


Figure 8.3: Segmenting the network by using firewalls.

The intent in this example is to acknowledge two important facts:

- Not all client computers need to communicate with all server resources
- The fewer servers a client can communicate with, the less damage it can do in the event that it becomes a platform for attack

☞ Although Figure 8.3 shows the Clients segment as one contiguous subnet, in reality, you could use additional firewalls to break down client computers into individual subnets, perhaps grouping a couple of dozen or so clients per subnet.

The firewalls between the subnets would then be used to restrict communications between the subnets. For example, client computers sitting on an HR Clients subnet might be the only ones permitted to communicate with the HR Servers subnet, and even those communications might be restricted to the selection of UDP and TCP ports actually used for authorized communications.

Does this architectural model create additional administrative overhead? Absolutely. Does it serve any purpose in increasing security? Definitely. A problem that the IT industry currently faces is the myth that the intranet can be trusted and therefore allowed to communicate however and whenever it wants with whatever other intranet resources. It is in this manner that viruses spread (and how other internal problems, such as embezzlement, occur). For example, when the Code Red worm was released for IIS 5.0, it quickly spread throughout networks because it was allowed to do so. Certainly, Microsoft's decision to install IIS by default on all Windows 2000 (Win2K) computers contributed to the problem, but had those computers had more restricted communications between one another, the problem would have been stopped cold. For example, do client computers in your company typically need to communicate with one another via HTTP? That is rarely the case, so why allow the communication? After all, HTTP communication was a big part of how Code Red was so successful at replicating itself.

The IT industry has long realized that the Internet is not to be trusted, and has taken steps to secure communications with it. Similarly, you need to realize that your own intranet can't be fully trusted and you must take steps to secure internal communications to help restrict the effect of an internal attack.

Creating Trust Zones

Another way to help secure internal network links is to stop considering the entire intranet to be a uniformly trustworthy area, and instead to define intranet *trust zones*. Figure 8.4 shows an example. Here, trust is based primarily on your ability to secure physical connections. The corporate LAN segment, for example, represents shared resources such as file servers and email servers. Clients connected to the inside office—that is, using network wiring only accessible from within a physically secured area—are granted the most trust because you can exercise more control over who connects to this network. The firewall allowing access to the corporate LAN might be the most permissive.

In contrast, conference room connections—which can be used by non-employees using computers they've brought from who knows where—are inherently less trustworthy. Although some access to the corporate LAN might be permitted, such access might be more restricted—simply providing access to a few less-sensitive file servers, for example, as well as access to the Internet.

Although not illustrated in Figure 8.4, the inside office connections would likely be allowed to connect to the Internet, as well, through an appropriate firewall.

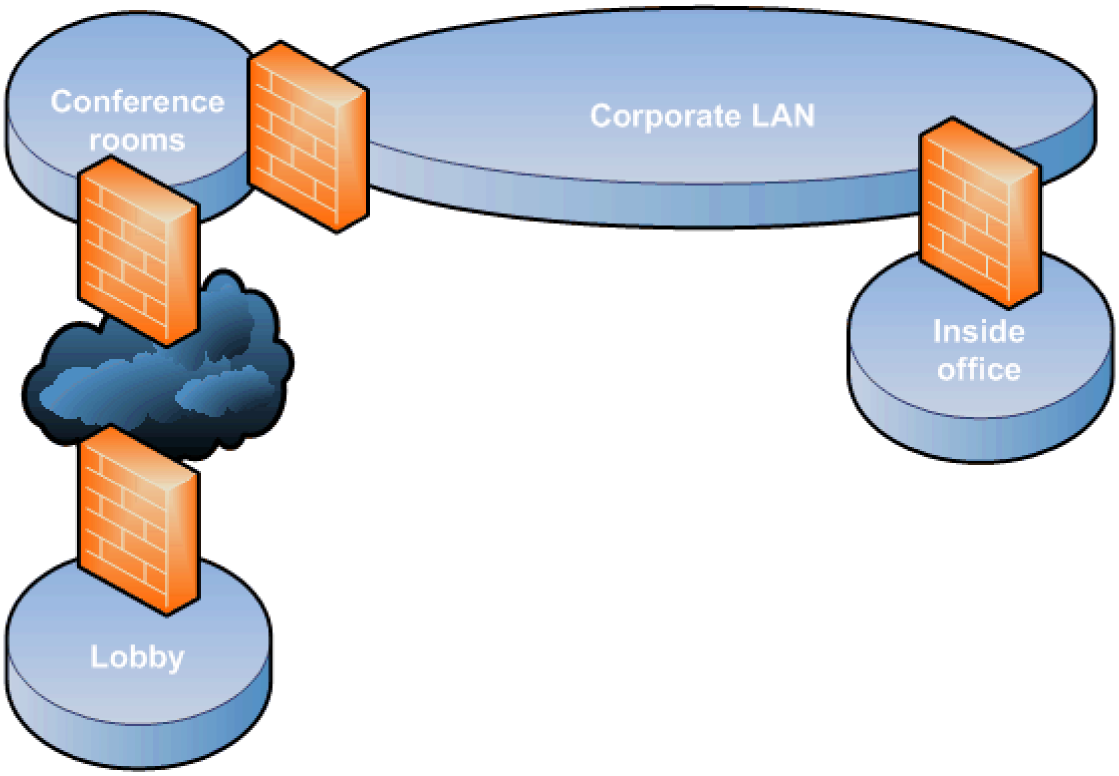


Figure 8.4: Defining intranet trust zones.

Wired connections in the building lobby, which offer the most opportunities for unauthorized individuals, might be provided only with Internet connectivity, allowing visitors to check their email, for example, but providing no connectivity to the resources on the corporate LAN.

This entire trust model is based on your ability to physically control the computers connecting to these various areas. The less physical control you have—to keep patches and antivirus software updated, for example—the less you trust the computer, and the less you trust the zone in which it can connect.

✐ What about wireless connections, which aren't tied to a physical area? Base your trust of wireless connections on the difficulty an attacker would have in connecting. Fully secured wireless connections that use WPA or other reliable security protocols might be considered fully trusted; less-secure or completely open wireless networks (perhaps ones provided for visitors' use) would be untrusted.

The Architectural Big Picture

This concept of trusted zones becomes an overlay on your segmented, resource-grouped security model. Thus, not only do HR clients have exclusive connectivity to HR servers, but those clients must be connecting from a trusted zone; untrusted zones don't provide any connectivity to the IP address ranges used by HR servers. This concept is *very* different than simply allowing free-flowing connectivity and restricting resource access based on authentication to AD. Of course, you're still restricting access based on NTFS (or other appropriate) permissions, but you're *also* preventing unauthorized client computers from even establishing a connection to these resources.

Figure 8.5 shows an example of the big picture. Complicated? It certainly is. But security *is* complicated and you shouldn't try to artificially simplify it. In reality, a complex communications plan such as this—which shows exactly which client segments are allowed to communicate with which server segments—might be implemented as a large set of rules within a single, centralized router or firewall. Once established, relatively little ongoing maintenance would be required. Notice, too, that server segments don't communicate with one another, other than as required.

✐ Not all connections are shown; obviously, subnets would need to communicate with a domain controller subnet, and the Internet connectivity also isn't shown. But hopefully this simplified diagram communicates the overall concept of segmenting communications.

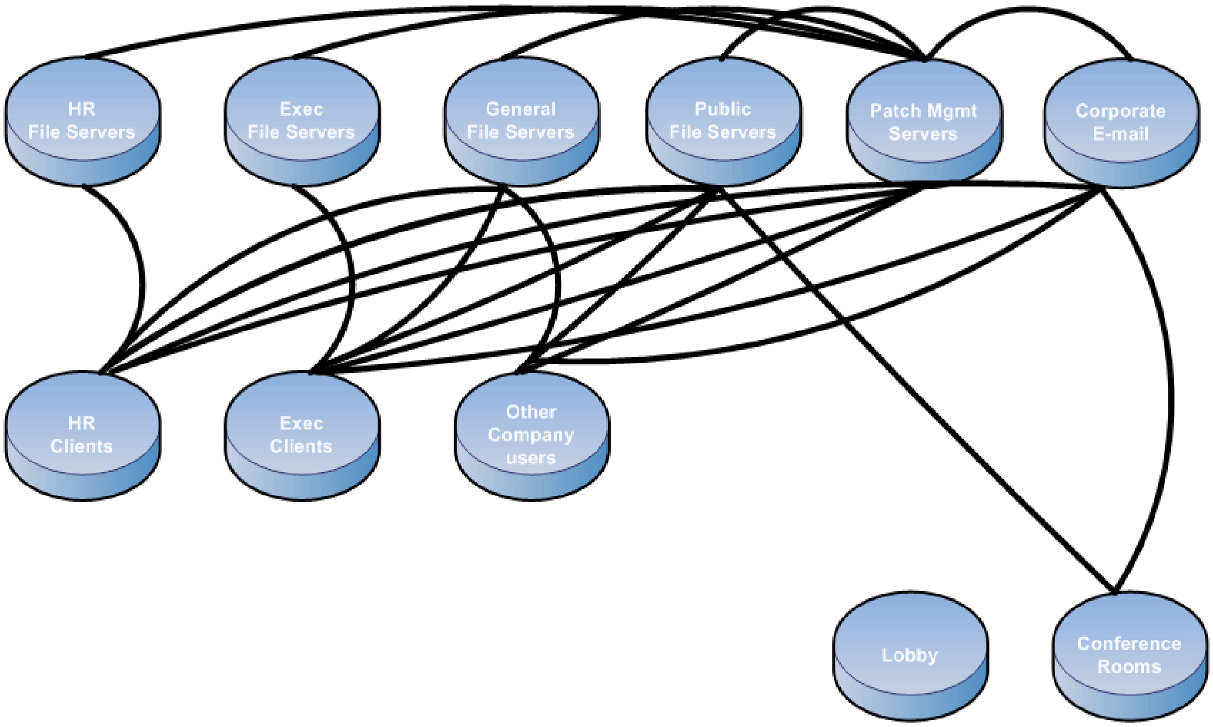


Figure 8.5: Creating a more secure communications plan.

What about the inevitable user who needs to access multiple subnets, such as an HR Director who needs access to both the HR and Exec subnets? You have a few options. You could simply bind two IP addresses to the person's computer, making them part of both the HR Clients and Exec Clients subnet (which are each likely to be a VLAN, anyway). You could enter the correct firewall rules for that one client. You could create a new client subnet that has access to both server subnets (the networking equivalent of creating a new user group to represent the unique access needs). This makes network administration as complex as user group administration—and shouldn't it be?

Remember, the overall intention is to acknowledge that *clients who will never have permissions to access a computer shouldn't be able to connect to that computer.* This philosophy accommodates the fact that certain anonymous connections (such as HTTP) can be leveraged for malicious purposes, and that vulnerabilities in the Windows OS can turn a simple connection into a major exploit. Although patch management, antivirus scanners, and other techniques can help mitigate these potential problems, an additional layer of defense—simply denying connectivity whenever possible—helps seal the deal and make the network that much more secure.

Security Through Monitoring

Monitoring is another critical aspect of network security, as it allows you to ensure that the network you designed and implemented stays the way you intended. It also allows you to detect new situations and circumstances related to security and to respond to them appropriately. Unfortunately, most network administrators do little or no routine monitoring, which creates significant opportunities for attackers.

Quarantine

One rapidly emerging concept in network security is the idea of *quarantine*. Although not yet implemented in a significant fashion within Windows, it's expected to be a major component of the forthcoming Windows Longhorn release.

> Microsoft has named the client version of Longhorn Windows Vista; the server version is expected to carry a name like Windows Server 2006. I'll continue to use the Longhorn code-name in this chapter.

The purpose behind quarantine is actually similar to the purpose behind trust zones, which were described earlier. Connections from your building's lobby, for example, are untrusted because you have no idea what is going to be plugged into there, and you have no control over it. Of course, *location* isn't really a definitive factor; a laptop plugged into your inside office network can be just as untrustworthy, especially if it's been off the network—and outside your control—for a significant period of time. Quarantine seeks to create a more dependable indication of trust.

Essentially, quarantine consists of both a client and server component. The client component is responsible for various scanning and inventory tasks, not unlike a systems management client of the kind installed by Microsoft Systems Management Server (SMS) or similar software. The server component contains network access policies. Those policies dictate certain requirements for accessing various different network resources. As Figure 8.6 shows, if a client can't meet the minimum requirements, the client isn't permitted to access those resources.

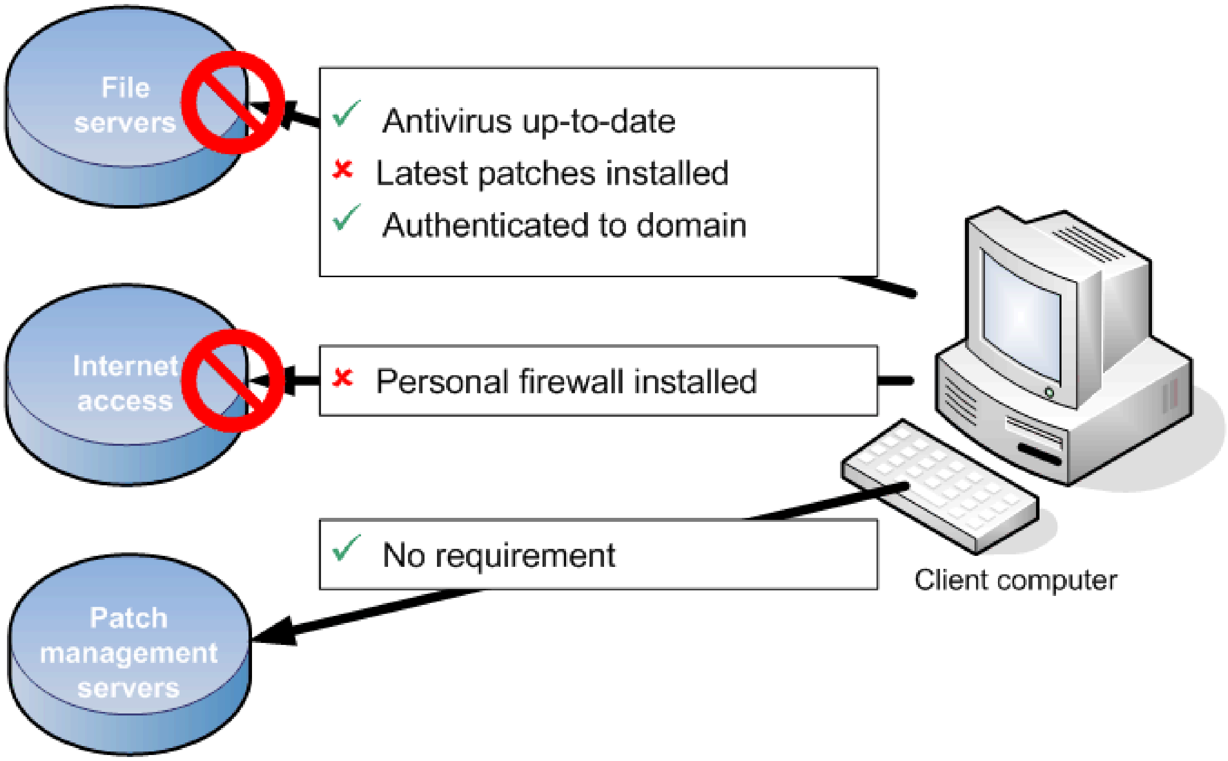


Figure 8.6: Quarantine—or network access policies—in action.

Here's an example: Full access is allowed to patch management servers, without restriction, because clients will need access to those servers in order to meet other requirements (this is a typical quarantine scenario—let clients get enough access to fix whatever problems they have). In order to access the Internet, clients must have a personal firewall installed and enabled (updated antivirus software might be another requirement). In this example, a personal firewall hasn't been detected, so Internet access isn't allowed. Access to corporate resources might require authentication, up-to-date patches, and up-to-date antivirus software; in this example, one of those requirements hasn't been met, so access is denied.

The purpose behind these policies is to guarantee a certain level of end-to-end security within your network. Clients without up-to-date patches might have been the victim of an exploited security vulnerability, and are denied access to resources that might be the target of the exploit's attack. Because infrastructure devices—such as routers and switches—will participate in quarantine (Microsoft has done extensive work with Cisco in creating standards for this purpose), computers that don't meet minimum requirements will have their connectivity denied, almost as if a smart firewall were in place. Thus, you'll be able to define what you consider a secure client to be, and ensure that only secure clients are allowed to connect to sensitive resources.

Network Sniffing

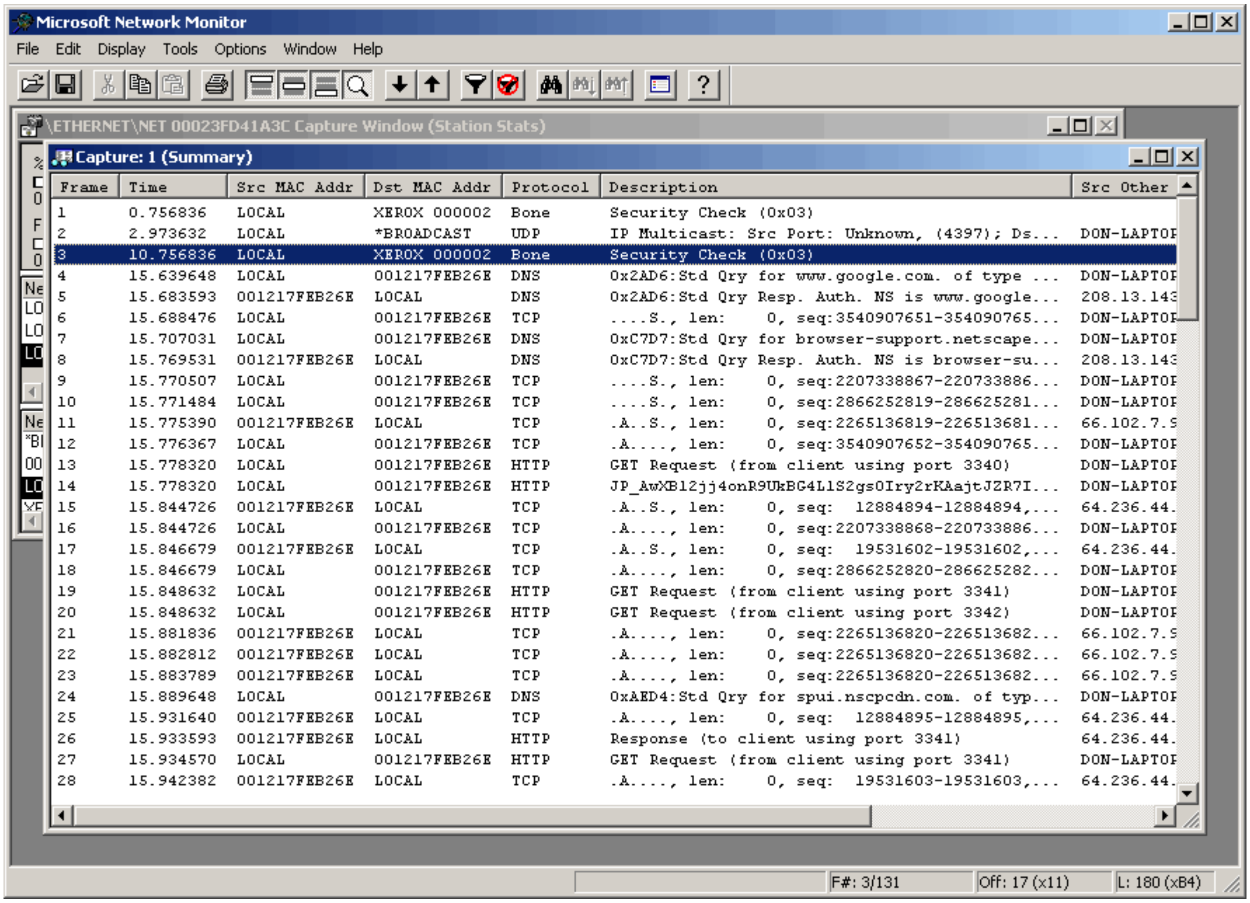
The value of a network sniffer—or packet capture tool, if you prefer the formal term—as a security tool is somewhat dubious. Although it's technically possible for you to watch packets fly by and detect anomalous activity, it's not likely to happen. In fact, network sniffers are more likely to be used as *part* of an attack rather than as a means of detecting one. For this reason, Microsoft's own network sniffing tool—Network Monitor, a version of which is included with Windows and a more flexible version of which is included with SMS—has its own security precaution built into it (see Figure 8.7).

Microsoft realized that Network Monitor (or NetMon, for short) could be used to attempt to capture sensitive information from the network as part of an attack. For this reason, every copy of NetMon periodically transmits a special packet in the Bone protocol. Other copies of NetMon can capture and identify this packet, telling you that a copy of NetMon is in use, and giving you the MAC address of the computer that's running it.

> ✎ Microsoft's internal code-name for NetMon is Bloodhound, which makes the name of the Bone protocol a sort of in-joke.

Periodically running a copy of NetMon and looking for Bone packets from computers other than your own will tell you whether NetMon is in use on your network. If it is, you can take appropriate steps. However, there are some tricks to ensuring you actually see the Bone packets that are transmitted:

- Leave a copy of NetMon running continuously to catch any packets. Have NetMon use a large capture buffer and drop packets other than those you're specifically looking for.
- Because switches can prevent your workstation from seeing all the traffic on the network, consider running NetMon on a standalone workstation that is connected to a *promiscuous* switch port—one that will echo all traffic that passes through the switch. Doing so helps to ensure that any transmitted Bone packets are, in fact, received.


Microsoft Network Monitor
File Edit Display Tools Options Window Help
\ETHERNET\NET 00023FD41A3C Capture Window (Station Stats)
Capture: 1 (Summary)
F#: 3/131
Off: 17 (x11)
L: 180 (xB4)

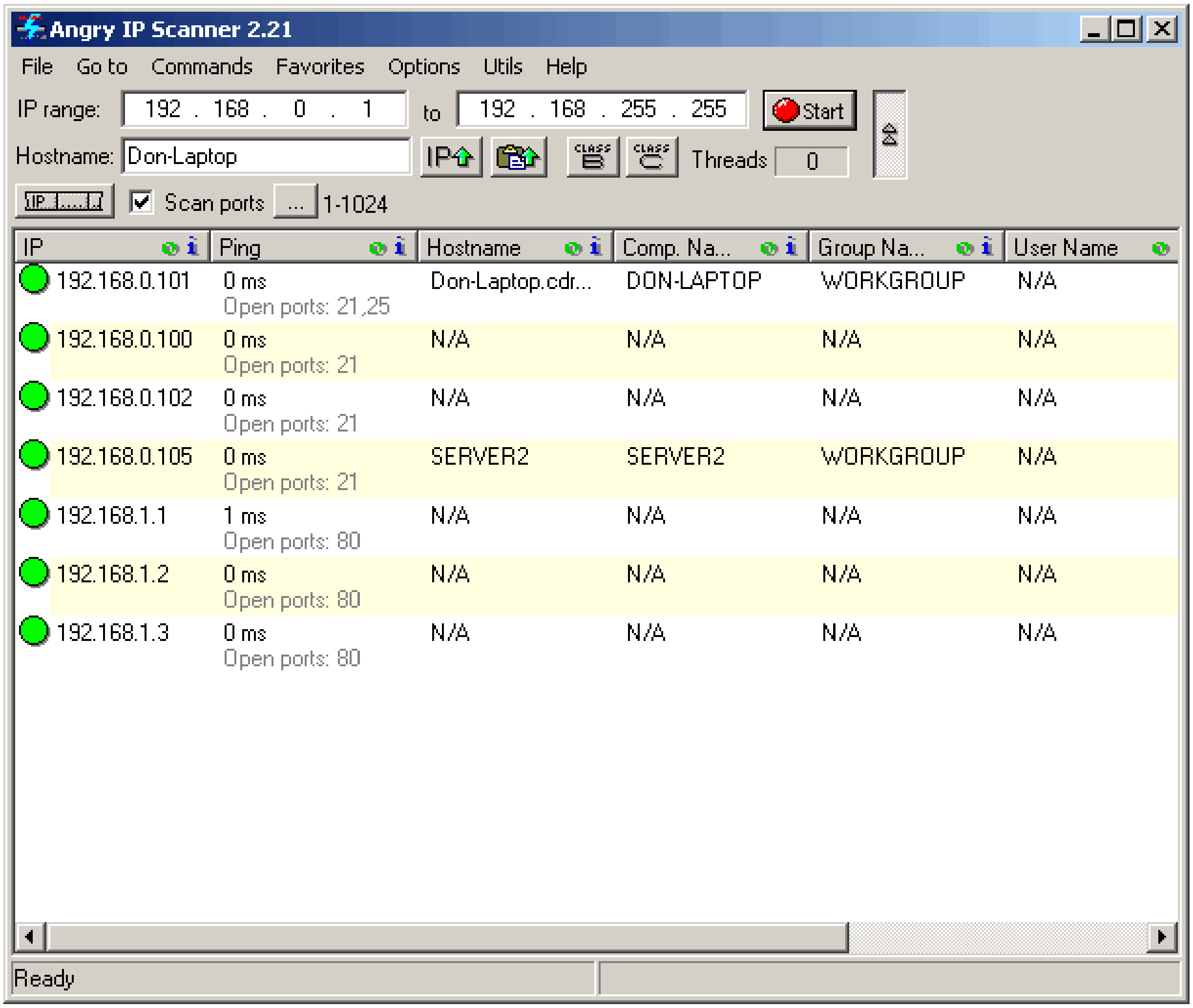
Frame	Time	Src MAC Addr	Dst MAC Addr	Protocol	Description	Src Other
1	0.756836	LOCAL	XEROX 000002	Bone	Security Check (0x03)	
2	2.973632	LOCAL	*BROADCAST	UDP	IP Multicast: Src Port: Unknown, (4397); Ds...	DON-LAPTOP
3	10.756836	LOCAL	XEROX 000002	Bone	Security Check (0x03)	
4	15.639648	LOCAL	001217FEB26E	DNS	0x2AD6:Std Qry for www.google.com. of type ...	DON-LAPTOP
5	15.683593	001217FEB26E	LOCAL	DNS	0x2AD6:Std Qry Resp. Auth. NS is www.google...	208.13.143
6	15.688476	LOCAL	001217FEB26E	TCP	....S., len: 0, seq:3540907651-354090765...	DON-LAPTOP
7	15.707031	LOCAL	001217FEB26E	DNS	0xC7D7:Std Qry for browser-support.netscape...	DON-LAPTOP
8	15.769531	001217FEB26E	LOCAL	DNS	0xC7D7:Std Qry Resp. Auth. NS is browser-su...	208.13.143
9	15.770507	LOCAL	001217FEB26E	TCP	....S., len: 0, seq:2207338867-220733886...	DON-LAPTOP
10	15.771484	LOCAL	001217FEB26E	TCP	....S., len: 0, seq:2866252819-286625281...	DON-LAPTOP
11	15.775390	001217FEB26E	LOCAL	TCP	.A..S., len: 0, seq:2265136819-226513681...	66.102.7.9
12	15.776367	LOCAL	001217FEB26E	TCP	.A...., len: 0, seq:3540907652-354090765...	DON-LAPTOP
13	15.778320	LOCAL	001217FEB26E	HTTP	GET Request (from client using port 3340)	DON-LAPTOP
14	15.778320	LOCAL	001217FEB26E	HTTP	JP_AwXB12jj4onR9UkBG4L1S2gs0Iry2rKAajtJZR7I...	DON-LAPTOP
15	15.844726	001217FEB26E	LOCAL	TCP	.A..S., len: 0, seq: 12884894-12884894,...	64.236.44.
16	15.844726	LOCAL	001217FEB26E	TCP	.A...., len: 0, seq:2207338868-220733886...	DON-LAPTOP
17	15.846679	001217FEB26E	LOCAL	TCP	.A..S., len: 0, seq: 19531602-19531602,...	64.236.44.
18	15.846679	LOCAL	001217FEB26E	TCP	.A...., len: 0, seq:2866252820-286625282...	DON-LAPTOP
19	15.848632	LOCAL	001217FEB26E	HTTP	GET Request (from client using port 3341)	DON-LAPTOP
20	15.848632	LOCAL	001217FEB26E	HTTP	GET Request (from client using port 3342)	DON-LAPTOP
21	15.881836	001217FEB26E	LOCAL	TCP	.A...., len: 0, seq:2265136820-226513682...	66.102.7.9
22	15.882812	001217FEB26E	LOCAL	TCP	.A...., len: 0, seq:2265136820-226513682...	66.102.7.9
23	15.883789	001217FEB26E	LOCAL	TCP	.A...., len: 0, seq:2265136820-226513682...	66.102.7.9
24	15.889648	LOCAL	001217FEB26E	DNS	0xAED4:Std Qry for spui.nscpcdn.com. of typ...	DON-LAPTOP
25	15.931640	001217FEB26E	LOCAL	TCP	.A...., len: 0, seq: 12884895-12884895,...	64.236.44.
26	15.933593	001217FEB26E	LOCAL	HTTP	Response (to client using port 3341)	64.236.44.
27	15.934570	LOCAL	001217FEB26E	HTTP	GET Request (from client using port 3341)	DON-LAPTOP
28	15.942382	001217FEB26E	LOCAL	TCP	.A...., len: 0, seq: 19531603-19531603,...	64.236.44.

Figure 8.7: Using Network Monitor as a security tool.

☞ Of course, NetMon is the only such tool that uses the Bone protocol, so other network sniffers won’t be detected this way. A better technique would be to use Software Restriction Policies to define the software you want to allow on your network, and to prevent other software—including network sniffers—from running. Still, because a non-domain computer wouldn’t be affected by Software Restriction Policies, monitoring for the Bone protocol can still help spot attackers who are using the readily available NetMon tool.

Port Monitoring

Port monitoring is an absolutely essential part of ongoing network security maintenance. It doesn't need to be difficult, either; port monitors (or scanners) are easy to come by. What you should have is a complete list of every computer on your network and the ports that are allowed to be open on each (for client computers, this might be as simple as saying "no ports should be open"). You can then routinely scan computers to ensure that only authorized ports are open. Figure 8.8 shows a simple, free port scanner available from http://www.angryziber.com/ipscan/.



IP range: 192 . 168 . 0 . 1 to 192 . 168 . 255 . 255

Hostname: Don-Laptop

Threads 0

☑ Scan ports ... 1-1024

IP	Ping	Hostname	Comp. Na...	Group Na...	User Name
192.168.0.101	0 ms Open ports: 21,25	Don-Laptop.cdr...	DON-LAPTOP	WORKGROUP	N/A
192.168.0.100	0 ms Open ports: 21	N/A	N/A	N/A	N/A
192.168.0.102	0 ms Open ports: 21	N/A	N/A	N/A	N/A
192.168.0.105	0 ms Open ports: 21	SERVER2	SERVER2	WORKGROUP	N/A
192.168.1.1	1 ms Open ports: 80	N/A	N/A	N/A	N/A
192.168.1.2	0 ms Open ports: 80	N/A	N/A	N/A	N/A
192.168.1.3	0 ms Open ports: 80	N/A	N/A	N/A	N/A

Ready

Figure 8.8: Angry IP Scanner.

As you can see, the scanner can automatically scan your entire network—based on IP address ranges—and report on each host it finds, listing the open ports on each. Simply compare this list with your list—especially on server computers—and take whatever action is necessary if unauthorized ports are found to be open.

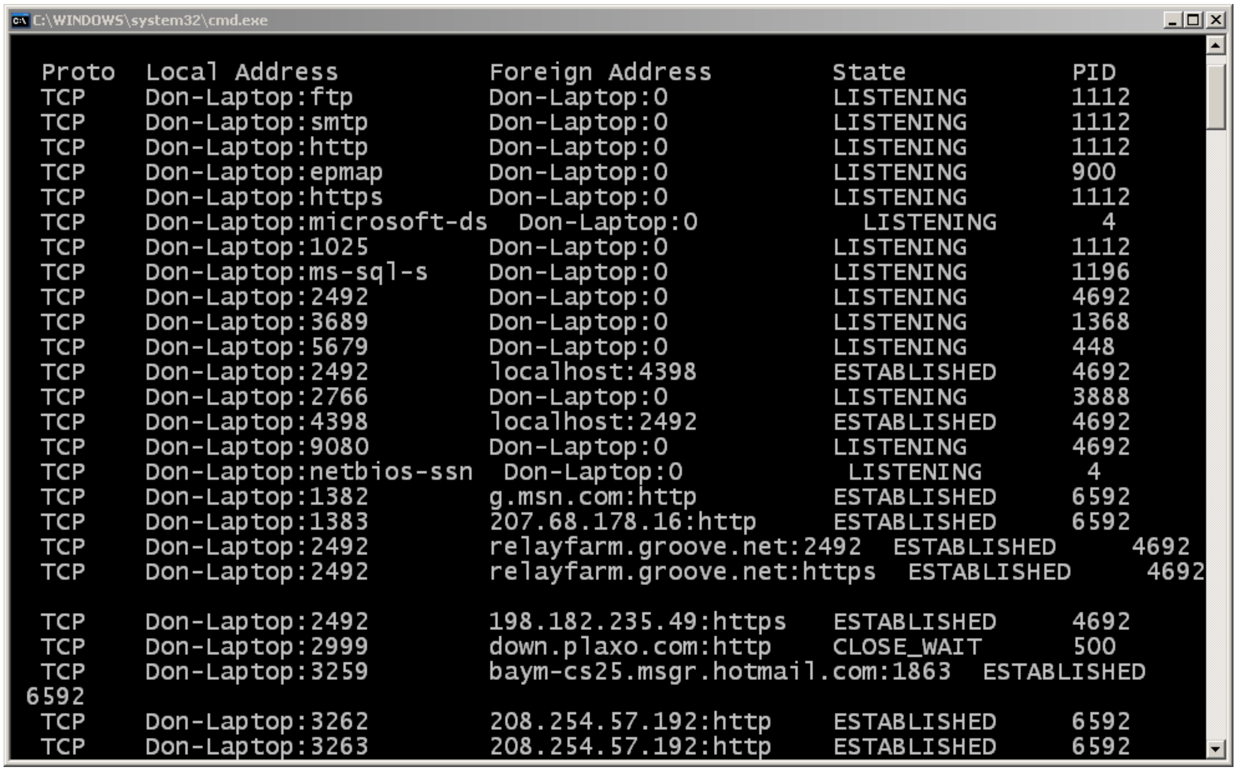
> ☞ As a bonus, this particular IP port scanner has an add-in that can display Windows shared folders, which are another excellent thing to monitor from a security standpoint, and the tool makes it easier to see which shared folders exist on each scanned computer.

Hundreds of other port scanners exist. In fact, Windows can easily display open ports simply by running

```
netstat –a –o
```

from a command-line. As Figure 8.9 shows, you'll not only get a list of ports but also the process ID (PID) of the process that has the port open, making it easier to determine which software opened the port (use Task Manager to look up the PID and see the process name). Although Netstat can't be used to scan ports on a remote computer, it is an effective way to examine the ports opened on a local computer and figure out what software has the port open.

> ☞ You can also run Netstat –a –b to display the executable name associated with each open port.



```
  Proto  Local Address          Foreign Address        State           PID
  TCP    Don-Laptop:ftp         Don-Laptop:0           LISTENING       1112
  TCP    Don-Laptop:smtp        Don-Laptop:0           LISTENING       1112
  TCP    Don-Laptop:http        Don-Laptop:0           LISTENING       1112
  TCP    Don-Laptop:epmap       Don-Laptop:0           LISTENING       900
  TCP    Don-Laptop:https       Don-Laptop:0           LISTENING       1112
  TCP    Don-Laptop:microsoft-ds  Don-Laptop:0           LISTENING       4
  TCP    Don-Laptop:1025        Don-Laptop:0           LISTENING       1112
  TCP    Don-Laptop:ms-sql-s    Don-Laptop:0           LISTENING       1196
  TCP    Don-Laptop:2492        Don-Laptop:0           LISTENING       4692
  TCP    Don-Laptop:3689        Don-Laptop:0           LISTENING       1368
  TCP    Don-Laptop:5679        Don-Laptop:0           LISTENING       448
  TCP    Don-Laptop:2492        localhost:4398         ESTABLISHED     4692
  TCP    Don-Laptop:2766        Don-Laptop:0           LISTENING       3888
  TCP    Don-Laptop:4398        localhost:2492         ESTABLISHED     4692
  TCP    Don-Laptop:9080        Don-Laptop:0           LISTENING       4692
  TCP    Don-Laptop:netbios-ssn  Don-Laptop:0           LISTENING       4
  TCP    Don-Laptop:1382        g.msn.com:http         ESTABLISHED     6592
  TCP    Don-Laptop:1383        207.68.178.16:http     ESTABLISHED     6592
  TCP    Don-Laptop:2492        relayfarm.groove.net:2492  ESTABLISHED     4692
  TCP    Don-Laptop:2492        relayfarm.groove.net:https  ESTABLISHED     4692

  TCP    Don-Laptop:2492        198.182.235.49:https   ESTABLISHED     4692
  TCP    Don-Laptop:2999        down.plaxo.com:http    CLOSE_WAIT      500
  TCP    Don-Laptop:3259        baym-cs25.msgr.hotmail.com:1863  ESTABLISHED
 6592
  TCP    Don-Laptop:3262        208.254.57.192:http    ESTABLISHED     6592
  TCP    Don-Laptop:3263        208.254.57.192:http    ESTABLISHED     6592
```

Figure 8.9: Using Netstat to monitor open ports.

Only *listening* ports are ones you need to be worried about; established ports may be used for *outgoing* communications, which aren't typically a concern.

Lockdown Tools

Locking down ports on Windows computers can be tricky, especially on servers. One wrong move and you can completely disable critical network services. The primary way in which ports are locked down is to disable the software that opened the port. You can also use local firewall software to simply block the port, whether a piece of software wants it open or not.

With Windows Server 2003 (WS2K3) Service Pack 1 (SP1), Microsoft introduced the Security Configuration Wizard, a tool that creates security templates to help achieve server lockdown. This tool includes an XML-formatted database that allows the wizard to understand the complex inter-service dependencies and port requirements in a Windows server, making it possible for administrators to simply indicate which roles a server is fulfilling—domain controller, file server, and so forth—and have the wizard create a security template that, when applied, will lock down the server correctly.

The wizard's templates can even provide proactive security, ensuring that any additional services that might be installed won't be able to start—and open any ports—until you modify the template (by using the wizard) to specifically allow that service.

Some of the commonly used ports you might find on a Windows computer include:

- TCP 20-21—FTP
- TCP 22—SSH
- TCP 23—Telnet
- TCP 25—SMTP
- TCP/UDP 53—DNS
- UDP 67-68—DHCP
- UDB 69—TFTP
- TCP 80—HTTP
- TCP 88—Kerberos
- TCP 110—POP3
- TCP 119—NNTP
- UDB 123—NTP
- TCP 139—NetBIOS Session
- TCP 143—IMAP4
- TCP 389—LDAP
- TCP 443—HTTPS
- TCP 445—Microsoft DS
- UDG 445—Microsoft SMB
- TCP 636—LDAP over SSL
- TCP 993—IMAP4 over SSL

- TCP 995—POP3 over SSL
- TCP 1344, 1434 (also on UDP)—SQL Server
- TCP 1494—Citrix MetaFrame
- TCP 1863—Windows/MSN Messenger
- TCP 3389—Terminal Services/RDP
- TCP 5190—AOL Instant Messenger (AIM)
- TCP 5222, 5223, 5269—Jabber
- TCP 5800, 5900—VNC

> 🕮 You can find more comprehensive lists at http://en.wikipedia.org/wiki/List_of_well-known_ports_(computing) and http://www.iana.org/assignments/port-numbers. The latter address is the official list of registered ports, and it includes port assignments through 49,151 (although not every port is assigned). The maximum port number is 65,535.

Many Microsoft products use endpoint mapping. Exchange Server is an excellent example—clients contact Exchange on a well-known port, connecting to the server's Remote Procedure Call (RPC) service. Exchange then dynamically opens a new port (usually well above 10,000) and assigns the client to that port. This technique allows Exchange to spread clients across ports, which provides better performance but makes it difficult to tell which ports are legitimate and which aren't.

> ✎ Exchange—and other applications—often allow you to configure them so that they use a very narrow range of ports, making it easier to work with them. However, this generally degrades the application's performance as well.

Intrusion Detection/Prevention Systems

Intrusion Detection Systems (IDSs) and Intrusion Prevention Systems (IPSs) act as, to one degree or another, automated monitoring and response systems. Essentially, they continually monitor your network for suspicious activity, then do something about it. An IDS is more likely to alert you upon detecting something amiss, while an IPS may be programmed to take some corrective or defensive action. A Common Intrusion Detection Framework (CIDF) defines four components for IDSs:

- "A" boxes, which analyze network activity
- "C" boxes, which include countermeasure mechanisms or response procedure equipment
- "D" boxes, which are disk storage mechanisms—essentially, logging devices
- "E" boxes, which are event generators, or sensors

Obviously, a single device or software application can encompass one or more of these pieces of functionality; a commercial IDS or IPS generally includes all four (with the "C" portion being what sets an IPS apart from an IDS, because the countermeasure mechanism is generally some sort of active response).

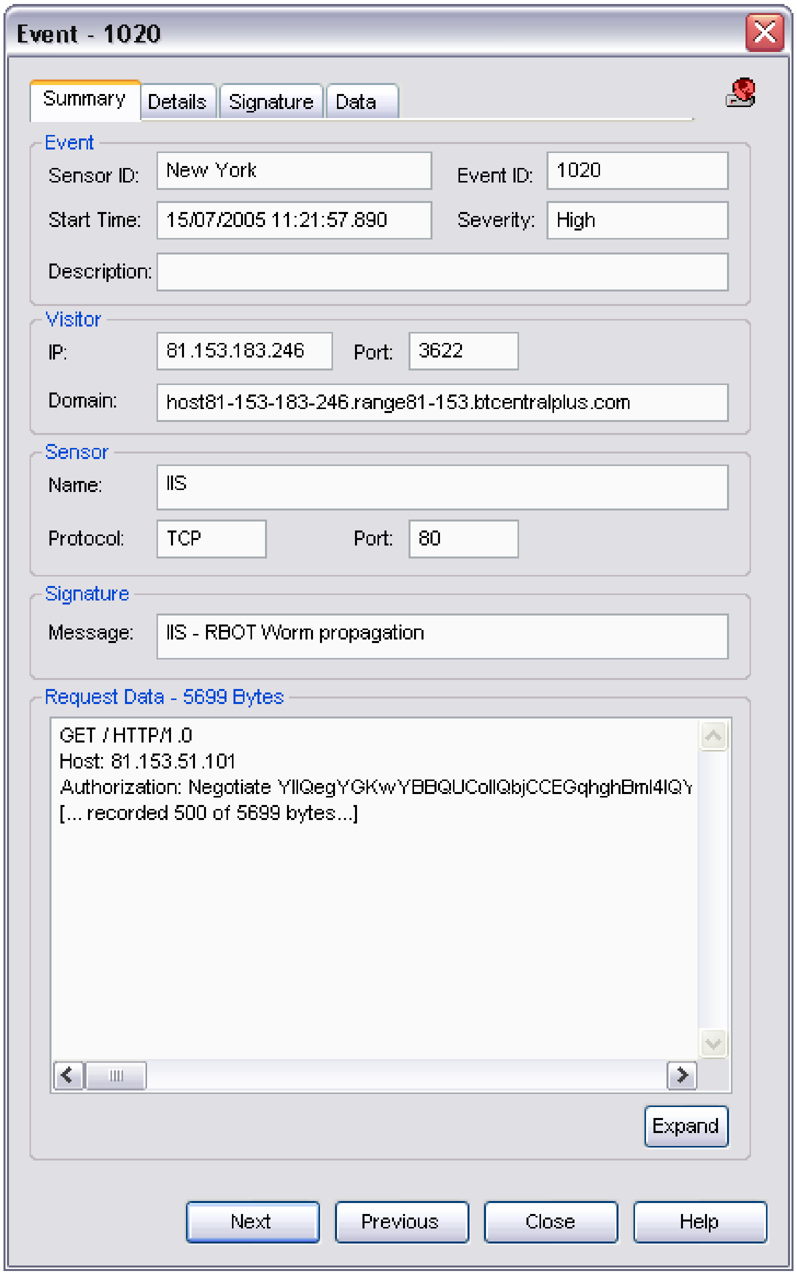
Different systems use different means of detecting potentially unwanted activity:

- *Suspicious detection* is used to flag a particular activity, such as Telnet usage on a network on which Telnet isn't normally used. Port scans are another suspicious activity.
- *Abnormal detection* is used to look for abnormal behavior, such as after-hours access to a particular resource.
- *Signature* or *pattern detection* looks for specific patterns of activity in much the same way an antivirus solution looks for patterns of code. A database usually defines a set of patterns known to be associated with particular types of attacks. This can, for example, be used to spot a worm that is transmitting itself across the network.

Better systems will obviously incorporate two or all of these techniques to provide the best coverage. In terms of form factor, IDSs are common in both hardware and software formats. For example, Figure 8.10 shows a partially disassembled Cisco WS-X6381-IDS unit; a hardware "black box" that is connected to your network and administered remotely.

Figure 8.10: Cisco hardware IDS.

Figure 8.11 shows a software IDS, the Windows-based KFSensor. This application scans for a variety of different potential attacks and can alert you to suspicious activity. As shown, the application has detected activity that appears to be an IIS-based worm attempting to propagate itself.



Event

Sensor ID: New York

Event ID: 1020

Start Time: 15/07/2005 11:21:57.890

Severity: High

Description:

Visitor

IP: 81.153.183.246

Port: 3622

Domain: host81-153-183-246.range81-153.btcentralplus.com

Sensor

Name: IIS

Protocol: TCP

Port: 80

Signature

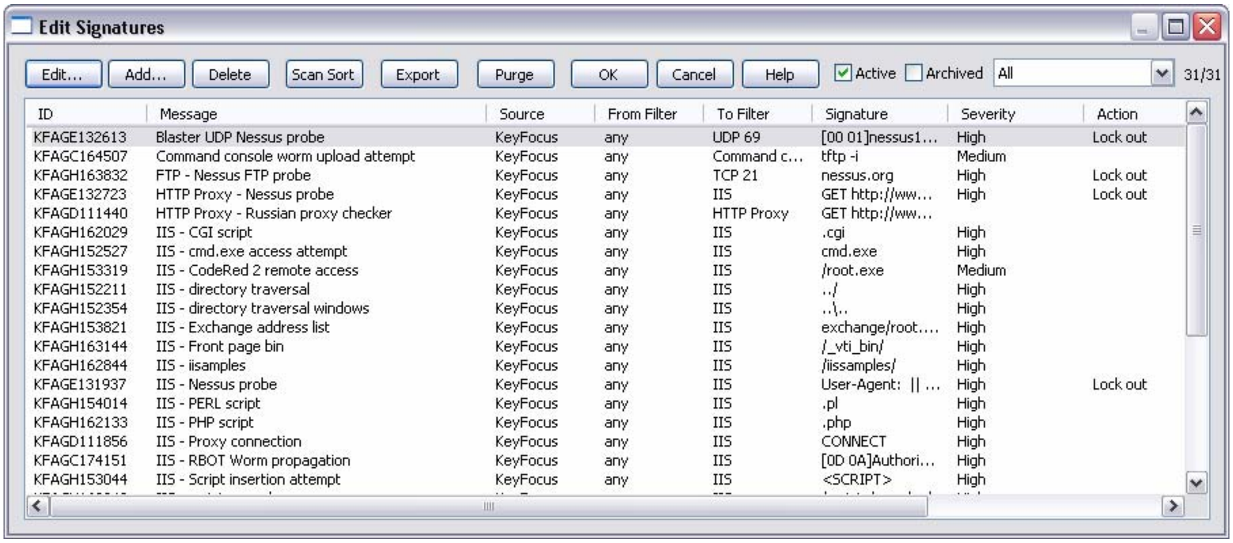
Message: IIS - RBOT Worm propagation

Request Data - 5699 Bytes

```
GET / HTTP/1.0
Host: 81.153.51.101
Authorization: Negotiate YIIQegYGKwYBBQUCoIIQbjCCEGqghBml4IQY
[... recorded 500 of 5699 bytes...]
```

Figure 8.11: Software IDS.

This system uses signature matching to detect the worm. This sort of system can be a valuable last resort in your overall security scheme, because it will let you know that the attack is taking place even if all of your active defenses fail. You can then act immediately to stop the attack, or mitigate its effects, before it gets completely out of control. A system such as KFSensor provides a complete signature database (see Figure 8.12) to detect attacks. Of course, just like an antivirus solution, this database must be continually updated to remain effective against emerging attacks.



ID	Message	Source	From Filter	To Filter	Signature	Severity	Action
KFAGE132613	Blaster UDP Nessus probe	KeyFocus	any	UDP 69	[00 01]nessus1...	High	Lock out
KFAGC164507	Command console worm upload attempt	KeyFocus	any	Command c...	tftp -i	Medium	
KFAGH163832	FTP - Nessus FTP probe	KeyFocus	any	TCP 21	nessus.org	High	Lock out
KFAGE132723	HTTP Proxy - Nessus probe	KeyFocus	any	IIS	GET http://ww...	High	Lock out
KFAGD111440	HTTP Proxy - Russian proxy checker	KeyFocus	any	HTTP Proxy	GET http://ww...		
KFAGH162029	IIS - CGI script	KeyFocus	any	IIS	.cgi	High	
KFAGH152527	IIS - cmd.exe access attempt	KeyFocus	any	IIS	cmd.exe	High	
KFAGH153319	IIS - CodeRed 2 remote access	KeyFocus	any	IIS	/root.exe	Medium	
KFAGH152211	IIS - directory traversal	KeyFocus	any	IIS	../	High	
KFAGH152354	IIS - directory traversal windows	KeyFocus	any	IIS	..\..	High	
KFAGH153821	IIS - Exchange address list	KeyFocus	any	IIS	exchange/root....	High	
KFAGH163144	IIS - Front page bin	KeyFocus	any	IIS	/_vti_bin/	High	
KFAGH162844	IIS - iisamples	KeyFocus	any	IIS	/iissamples/	High	
KFAGE131937	IIS - Nessus probe	KeyFocus	any	IIS	User-Agent: || ...	High	Lock out
KFAGH154014	IIS - PERL script	KeyFocus	any	IIS	.pl	High	
KFAGH162133	IIS - PHP script	KeyFocus	any	IIS	.php	High	
KFAGD111856	IIS - Proxy connection	KeyFocus	any	IIS	CONNECT	High	
KFAGC174151	IIS - RBOT Worm propagation	KeyFocus	any	IIS	[0D 0A]Authori...	High	
KFAGH153044	IIS - Script insertion attempt	KeyFocus	any	IIS	<SCRIPT>	High	

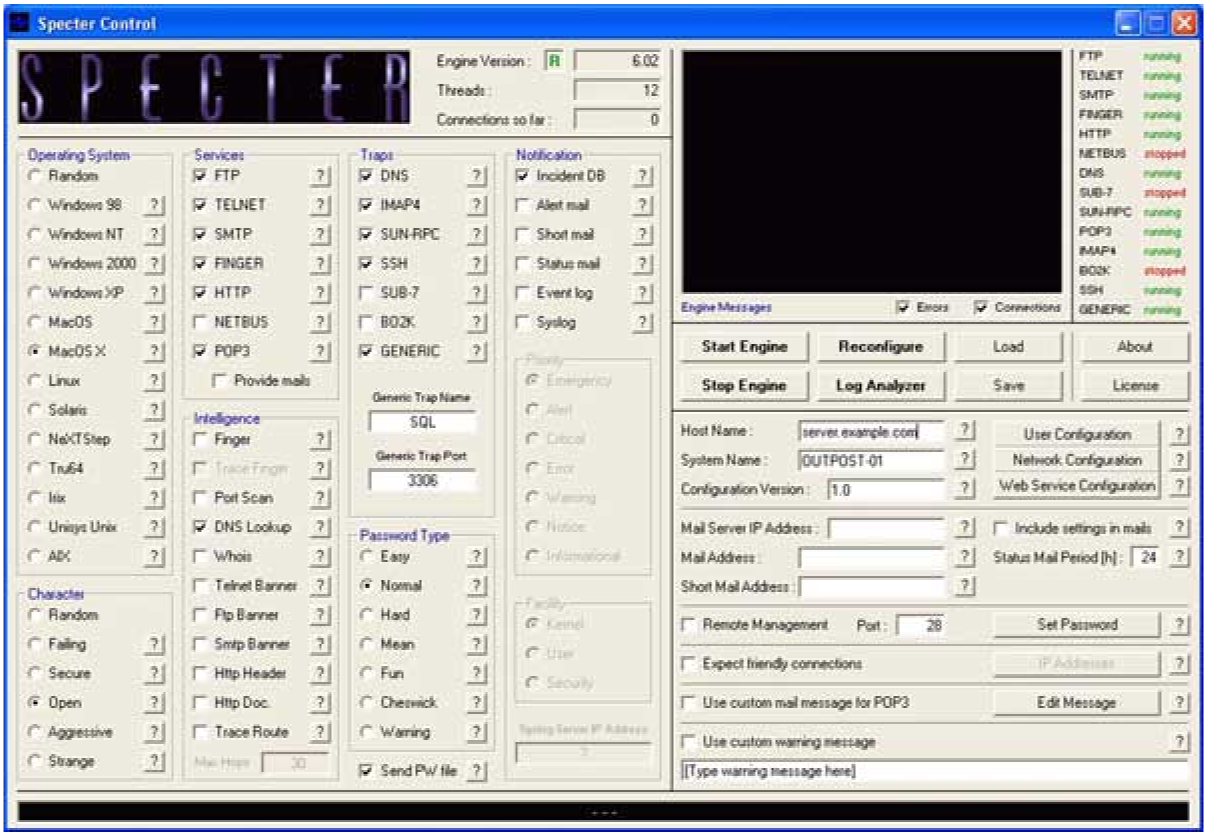
Figure 8.12: Signature database in KFSensor.

> You should be able to define your own signatures in an IDS (KFSensor allows you to), as well as import signatures from the popular SNORT format (KFSensor allows this, too).

One key way that IDSs detect attacks is by acting as traps, or *honeypots*. With this technique, the IDS doesn't continually scan all network traffic—which, depending on the size and configuration of your network, might be impractical or difficult to set up—but rather sets itself up to be attacked. The IDS simulates various services and OSs, essentially appearing to would-be attackers as a wide-open server. By placing various IDS *sensors* throughout your network, you can be sure of attracting attacks and spotting them before they get too far on your network.

For example, Figure 8.13 shows another software IDS, Specter Control, being used to emulate a Mac OS X computer with open FTP, Telnet, SMTP, Finger, HTTP, and POP3 ports. It's providing traps for DNS, IMAP4, SUN-RPC, SSH, and generic traffic. By emulating an "open" system, the IDS will be one of the first attack targets (especially if your other hosts are locked down). Once attacked, the IDS can provide full details about where the attack is coming from, allowing you to stop it.

> A really flexible IDS system will combine passive network scanning—along the lines of network sniffing—with honeypot techniques. This feature allows the IDS to see attacks being made against other systems as well as provide a target for attacks. Because both types of monitoring—sniffing and honeypot—are useful, you might use different IDS systems that each provides one of these techniques.
>
> Keep in mind that a sniffer-style IDS is limited to the traffic you allow it to see. For that reason, they're usually connected to switch ports that have been defined as *promiscuous,* allowing the IDS to "see" all of the traffic passing through the switch.



Specter Control

Engine Version: R 6.02
Threads: 12
Connections so far: 0

Operating System
○ Random
○ Windows 98
○ Windows NT
○ Windows 2000
○ Windows XP
○ MacOS
● MacOS X
○ Linux
○ Solaris
○ NeXTStep
○ Tru64
○ Irix
○ Unisys Unix
○ AIX

Character
○ Random
○ Failing
○ Secure
● Open
○ Aggressive
○ Strange

Services
☑ FTP
☑ TELNET
☑ SMTP
☑ FINGER
☑ HTTP
☐ NETBUS
☑ POP3
☐ Provide mails

Intelligence
☐ Finger
☐ Trace Finger
☐ Port Scan
☑ DNS Lookup
☐ Whois
☐ Telnet Banner
☐ Ftp Banner
☐ Smtp Banner
☐ Http Header
☐ Http Doc
☐ Trace Route
Max Hops: 30

Traps
☑ DNS
☑ IMAP4
☑ SUN-RPC
☑ SSH
☐ SUB-7
☐ BO2K
☑ GENERIC
Generic Trap Name: SQL
Generic Trap Port: 3306

Password Type
○ Easy
● Normal
○ Hard
○ Mean
○ Fun
○ Cheswick
○ Warning
☑ Send PW file

Notification
☑ Incident DB
☐ Alert mail
☐ Short mail
☐ Status mail
☐ Event log
☐ Syslog

Priority
● Emergency
○ Alert
○ Critical
○ Error
○ Warning
○ Notice
○ Informational

Facility
● Kernel
○ User
○ Security

Syslog Server IP Address

Engine Messages ☑ Errors ☑ Connections

FTP running
TELNET running
SMTP running
FINGER running
HTTP running
NETBUS stopped
DNS running
SUB-7 stopped
SUN-RPC running
POP3 running
IMAP4 running
BO2K stopped
SSH running
GENERIC running

Host Name: server.example.com
System Name: OUTPOST-01
Configuration Version: 1.0
Mail Server IP Address:
Mail Address:
Short Mail Address:
☐ Include settings in mails
Status Mail Period [h]: 24
☐ Remote Management Port: 28
☐ Expect friendly connections
☐ Use custom mail message for POP3
☐ Use custom warning message
[Type warning message here]

Figure 8.13: IDSs emulate open systems to attract attackers.

IDS deployment techniques often suggest placing a honeypot IDS in your perimeter network (such as alongside your public Web servers) to attract attackers. This method is a good idea, but it presumes that all attacks will come from outside your network, which isn't the case. Deploy IDSs everywhere an attack might originate or target, including throughout your network and most especially alongside resource servers, internal Web servers, and even on a handful of client segments. Many, many, many modern attacks first target more vulnerable client computers, often through social engineering (sending in a virus disguised as an animated postcard from a family member, for example). Once established on a trusted computer, the virus then launches more subtle attacks against corporate resources, including servers. Always remember that your internal network can never be completely trustworthy; treat it in much the same way you would the Internet by minimizing the exposure corporate resources have even to internally launched attacks.

Security Through Technology

There are a number of technologies that can, in and of themselves, help to increase the security on your network. Some are a bit complicated to deploy and may require careful planning, but they're unique in that, no matter what type of network you're operating or what your security issues are or business needs may be, these can almost always provide an "instant win" by helping increase the level of security you're working with.

802.1X

Although 802.1X is often associated with wireless security (an area I'll touch on in the next section), it is in fact equally applicable to wired networks. 802.1X is an IEEE standard for port-based network access control, requiring devices—rather than users—to authenticate before they're actually connected to the network. 802.1X requires devices—such as routers, client computers, and other connected equipment—to have a client component, while network switches typically provide the access control point. Typically, 802.1X uses the Extensible Authentication Protocol (RFC 2284) to handle authentication.

When fully implemented, 802.1X can be a thing of beauty. No longer do you have to worry about trusted and untrusted zones of your network; only trusted clients—those who can authenticate—can obtain network "dial tone." Unauthorized computers can't even get an IP address because they're not granted network access. Network switches usually rely on RADIUS for authentication, as depicted in Figure 8.14.

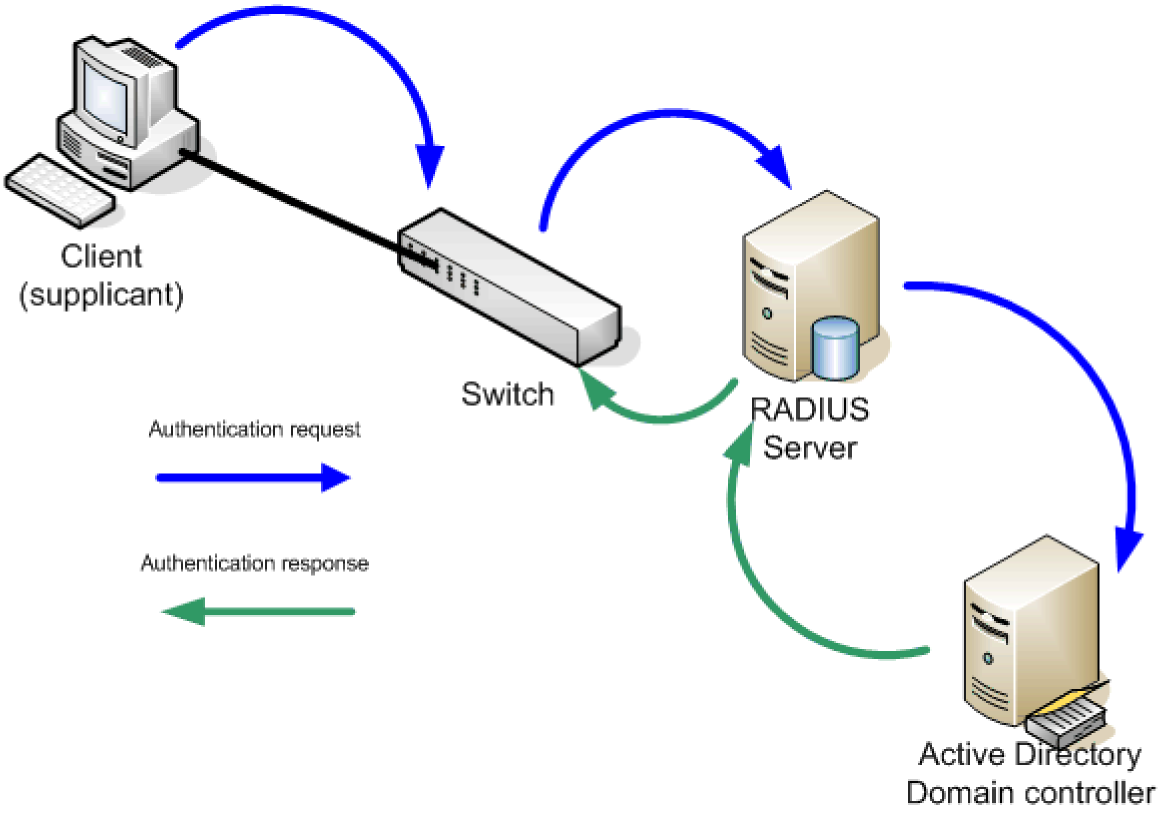


Figure 8.14: Using RADIUS in an 802.1X environment.

Here, the client—known as a *supplicant* in 802.1X terminology—passes authentication credentials to the switch. The switch then passes these to a RADIUS server, which may pass them on to another directory—such as AD—for processing. When the RADIUS server gives the switch a positive response, the switch begins transmitting network traffic for that client. Without the positive authentication, however, the switch port the client is connected to is never fully activated, cutting the client off from the entire network.

Windows has supported 802.1X since Windows XP and WS2K3, although to date, implementations have primarily focused on wireless networks, where it's far easier for unauthorized individuals to "plug in" to the network. However, deploying 802.1X for your wired networks will eliminate the possibility of an attacker compromising your physical network—accessing the network from the lobby or other public locations—and increase the overall level of security on your network.

Wireless Security

In addition to using 802.1X to secure actual connections to your wireless networks, using a wireless security protocol can help protect the contents of wireless transmissions and keep unauthorized users off the network.

Originally, the Wired Equivalent Privacy (WEP) protocol was intended to provide the same level of privacy one could expect with a wired connection (which isn't actually all that much privacy). However, the protocol was quickly cracked due to some inherent vulnerabilities, and the industry as a whole became wary of it. WEP has since been superseded by WPA, which has been codified in IEEE standard 802.11i (called WPA2). WEP continues to provide a minimal level of security, but WPA is considered the minimum level of security for sensibly run wireless networks.

802.11i utilizes the Advanced Encryption Standard (AES) block cipher for encryption (WEP and the original WPA only used a less-secure RC4 stream cipher). 802.11i actually requires 802.1X for port-level authentication. 802.11i is designed to use 802.1X for authentication, or it can use pre-shared (symmetric) keys. Pre-shared keys are easier to implement (because you essentially just configure the wireless access point—AP—and your client devices with a password) but is considered less secure because the password is rarely (if ever) changed and because it's used too much. Windows XP supports WPA2/802.11i in the latest service pack, although you may also need updated drivers for your wireless network adapter in order to take advantage of the service.

Secure Network Adapters

Major network adapter manufacturers, including companies such as Intel and Broadcom, are beginning to incorporate security features into network hardware. Typically, these products are developed with unique embedded digital certificates that uniquely identify the network adapter. Because these certificates cannot be forged (the certificates themselves are often digitally signed using the adapter manufacturer's own private key), they represent a more secure form of identification than MAC addresses (which can easily be forged). Network infrastructure devices—such as DHCP servers, switches, and so forth—can be designed to identify devices by their unique certificate, helping to ensure that only authorized, known devices are allowed to access the network and its services. The embedded certificate can also be used to speed network-level encryption and other services.

IP Security

IP Security (IPSec) is a broad suite of protocols designed to improve network security. From an implementation standpoint, IPSec can be viewed as a set of filters and actions. For example, a filter might define all traffic destined to a particular server, or all traffic sent over a particular protocol. When traffic matches a filter, a companion action is applied to that traffic. Actions may be as simple as dropping the traffic (thus giving IPSec firewall-like capabilities), or it might require the traffic to be encrypted or authenticated. Figure 8.15 illustrates this process.

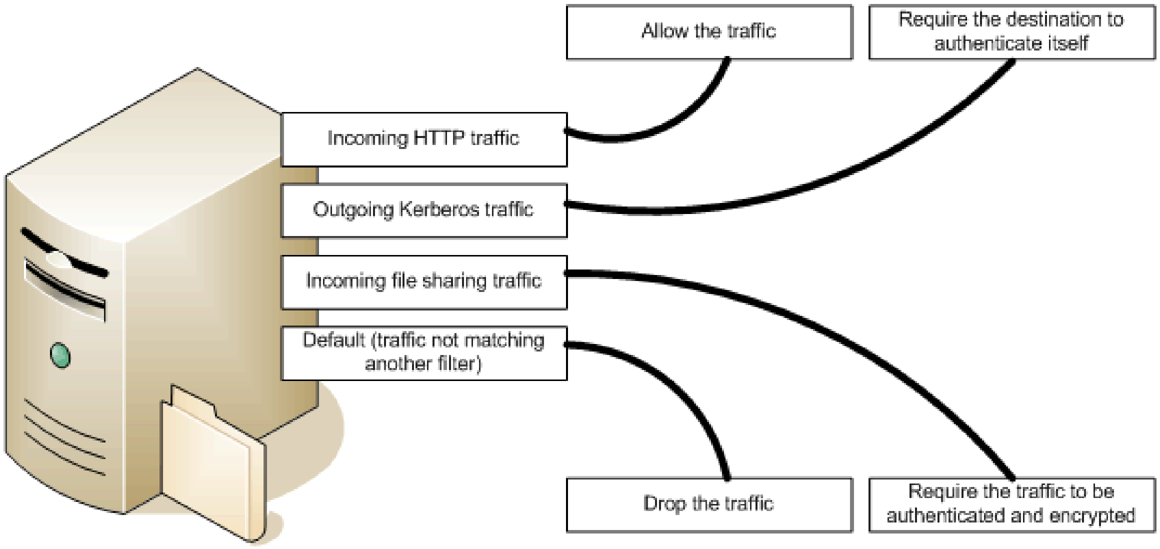
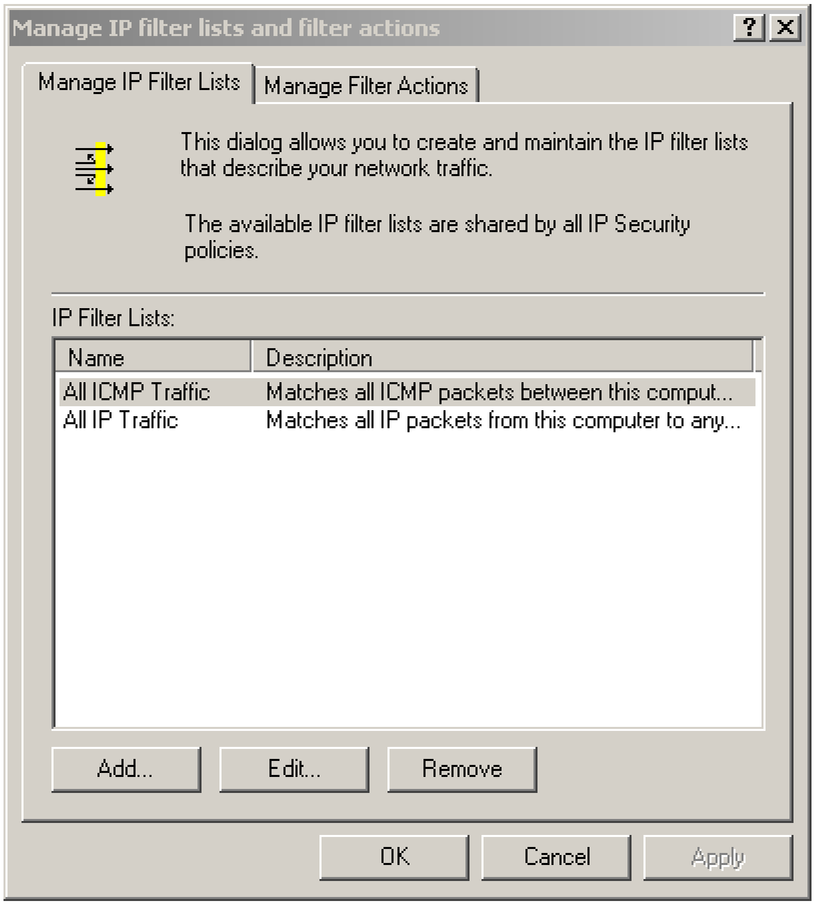

Allow the traffic
Require the destination to authenticate itself
Incoming HTTP traffic
Outgoing Kerberos traffic
Incoming file sharing traffic
Default (traffic not matching another filter)
Drop the traffic
Require the traffic to be authenticated and encrypted

Figure 8.15: IPSec filters and actions.

For example, you might define a filter for all ICMP traffic and another for all IP traffic. Figure 8.16 illustrates how Windows XP might list two such filters.



This dialog allows you to create and maintain the IP filter lists that describe your network traffic.

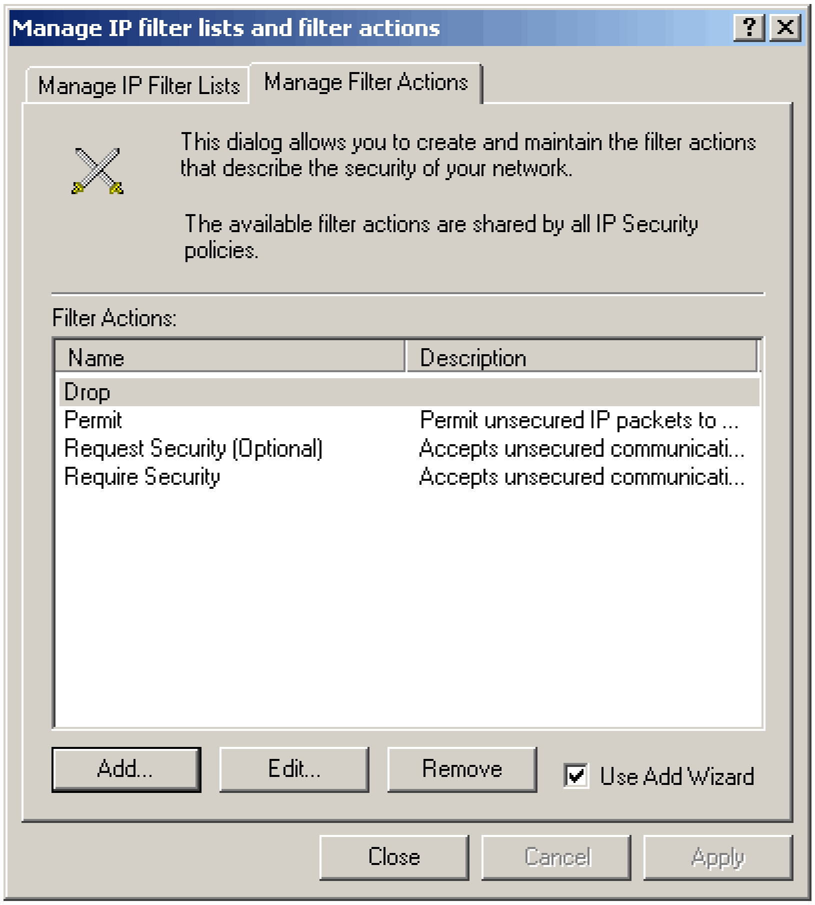
The available IP filter lists are shared by all IP Security policies.

IP Filter Lists:

Name	Description
All ICMP Traffic	Matches all ICMP packets between this comput...
All IP Traffic	Matches all IP packets from this computer to any...

Figure 8.16: IPSec filters in Windows XP.

You might define rules that drop traffic, allow it, request security (but make it optional), or require security. Figure 8.17 lists these rules in the Windows XP IPSec interface.



This dialog allows you to create and maintain the filter actions that describe the security of your network.

The available filter actions are shared by all IP Security policies.

Filter Actions:

Name	Description
Drop	
Permit	Permit unsecured IP packets to ...
Request Security (Optional)	Accepts unsecured communicati...
Require Security	Accepts unsecured communicati...

☑ Use Add Wizard

Figure 8.17: IPSec rules in Windows XP.

An IP Security Policy (or IPSec Policy) joins filters and actions: For example, you might create a policy that drops ICMP traffic or one that requests security for all other IP traffic.

IPSec obviously requires a lot of planning before implementing it across your organization, but it’s yet another way in which your organization’s network security can immediately benefit a great deal. IPSec can be centrally controlled through AD Group Policy, allowing you to create centralized IPSec policies for your entire organization. This setup can help ensure that sensitive data is never transmitted in clear text; when combined with technologies such as 802.1X to prevent unauthorized network connections, IPSec can act as part of a layered defense that makes your network as close to absolutely secure as possible.

Summary

This chapter covers several techniques, technologies, and tools that can be used to help close the loop on Windows enterprise security. By providing a better, more secure network backbone on which Windows can operate, you'll help close security loopholes, defend against common attacks on Windows systems, and provide an overall more secure environment for your users.

Windows security is a matter of details: Paying attention not only to major security issues but also the many commonly overlooked issues this guide has explored will help you maintain a more secure, more reliable Windows network. Security has become the lynchpin for a number of enterprise needs, including compliance, reliability, accountability, and more; by creating a more secure Windows enterprise, you'll be helping your organization meet a number of important business needs. Good luck!