TIP

Five command line tools to detect Windows hacks

Learn about five of the most useful Windows command-line tools for machine analysis and how they can tell if a machine has been hacked in this tip.

By **Ed Skoudis,** SANS Technology Institute

Published: **10 Mar 2008**

Let's face it, Windows machines get hacked, and in some environments it happens a lot. Fortunately, Microsoft has built numerous tools into Windows so administrators and power users can analyse a machine to determine whether it's been compromised. In this tip, which is the first of a two-part series, I'll cover five useful command-line tools built into Windows for such analysis.

Click here for *five more* command line tools to detect Windows hacks.

1) WMIC: A world of adventure awaits

Windows Management Instrumentation Command-line (WMIC) is not merely a command; it's a world unto itself. Offering a command-line interface to the ultra-powerful Windows Management Instrumentation API within Windows, WMIC lets administrative users access all kinds of detailed information about a Windows machine, including detailed attributes of thousands of settings and objects. WMIC is built into Windows XP Professional, Windows 2003 and Windows Vista.

To use WMIC, users must invoke it by running the WMIC command followed by the area of the machine the user is interested in (often referred to as an alias within the system). For example, to learn more about the processes running on a machine, a user could run:

```
C:\> wmic process
```

Output of that command will likely look pretty ugly because an output format wasn't specified. With WMIC, output can be formatted in several different ways, but two of the most useful for analysing a system for compromise are the "list full" option, which shows a huge amount of detail for each area of the machine a user is interested in, and the "list brief" output, which provides one line of output per report item in the list of entities, such as running processes, autostart programs and available shares.

For example, we can look at a summary of every running process on a machine by running:

C:\> wmic process list brief

That command will show the name, process ID and priority of each running process, as well as other less-interesting attributes. To get even more detail, run:

C:\> wmic process list full

This command shows all kinds of details, including the full path of the executable associated with the process and its command-line invocation. When investigating a machine for infection, an administrator should look at each process to determine whether it has a legitimate use on the machine, researching unexpected or unknown processes using a search engine.

Beyond the process alias, users could substitute startup to get a list of all auto-start programs on a machine, including programs that start when the system boots up or a user logs on, which could be defined by an auto-start registry key or folder:

C:\> wmic startup list full

A lot of malware automatically runs on a machine by adding an auto-start entry alongside the legitimate ones which may belong to antivirus tools and various system tray programs. Users can look at other settings on a machine with WMIC by replacing "startup" with "QFE" (an abbreviation which stands for Quick Fix Engineering) to see the patch level of a system, with "share" to see a list of Windows file shares made available on the machine and with "useraccount" to see detailed user account settings.

A handy option within WMIC is the ability to run an information-gathering command on a repeated basis by using the syntax "/every:[N]" after the rest of the WMIC command. The [N] here is an integer, indicating that WMIC should run the given command every [N] seconds. That way, users can look for changes in the settings of the system over time, allowing careful scrutiny of the output. Using this function to pull a process summary every 5 seconds, users could run:

C:\> wmic process list brief /every:1

Hitting CTRL+C will stop the cycle.

2) The net command: An oldie but a goodie

While WMIC is a relatively new command, let's not lose site of some useful older commands. One of my favourites is the venerable "net" command. Administrators can use this to display all kinds of useful information.

For example, the "net user" command shows all user accounts defined locally on the machine. The "net localgroup" command shows groups, "net localgroup administrators" shows membership of the administrators group and the "net start" command shows running services.

Attackers frequently add users to a system or put their own accounts in the administrators groups, so it's always a good idea to check the output of these commands to see if an attacker has manipulated the accounts on a machine. Also, some attackers create their own evil services on a machine, so users should be on the lookout for them.

3) Openfiles: Deep scrutiny

Many Windows administrators are unfamiliar with the powerful openfiles command built into Windows. As its name implies, this command shows all files that are opened on the box, indicating the process name interacting with each file. It's built into modern versions of Windows, from XP Pro to Vista. Like the popular lsof command for Linux and Unix, it'll show administrators all open files on the machine, giving the process name and full path for each file. Unlike lsof, however, it doesn't provide many more details, such as process ID number, user number and other information.

Considering the volume of information it gathers, it's no surprise that the openfiles command is a performance hog. Thus, the accounting associated with openfiles is off by default, meaning users can't pull any data from this command until it is turned on. This function can be activated by running:

C:\> openfiles /local on

Users will need to reboot, and when the system comes back, they will be able to run the openfiles command as follows:

C:\> openfiles /query /v

This command will show verbose output, which includes the user account that each process with an open file is running under. To get an idea of what malware has been installed, or what an attacker may be doing on a machine, users should look for unusual or unexpected files, especially those associated with unexpected local users on the machine.

When finished with the openfiles command, its accounting functionality can be shut off and the system returned to normal performance by running the following command and rebooting:

C:\> openfiles /local off

4) Netstat: Show me the network

The Windows netstat command shows network activity, focusing on TCP and UDP by default. Because malware often communicates across the network, users can look for unusual and unexpected connections in the output of netstat, run as follows:

C:\> netstat -nao

The -n option tells netstat to display numbers in its output, not the names of machines and protocols, and instead shows IP addresses and TCP or UDP port numbers. The -a indicates to display all connections and listening ports. The -o option tells netstat to show the processID number of each program interacting with a TCP or UDP port. If, instead of TCP and UDP, you are interested in ICMP, netstat can be run as follows:

C:\> netstat -s -p icmp

This indicates that the command will return statistics (-s) of the ICMP protocol. Although not as detailed as the TCP and UDP output, users can see if a machine is sending frequent and unexpected ICMP traffic on the network. Some backdoors and other malware communicate using the payload of ICMP Echo messages, the familiar and innocuous-looking ping packets seen on most networks periodically.

Like WMIC, the netstat command also lets us run it every N seconds. But, instead of using the WMIC syntax of "/every:[N]", users simply follow their netstat invocation with a space and an integer. Thus, to list the TCP and UDP ports in use on a machine every 2 seconds, users can run:

C:\> netstat -na 2

5) Find: Searching output for useful stuff

Most of the commands I have discussed so far spew a lot of output on the screen, which could be hard for a human to look through to find a specific item of interest. But, Windows comes to the rescue. Users can search through the output of a command using the built-in find and findstr commands in Windows. The find command looks for simple strings, while findstr supports regular expressions, a more complex way to specify search patterns. Because the regular expressions supported by findstr go beyond the scope of this tip article, let's focus on the find command. By default, find is case sensitive - use the /i option to make it case insensitive.

The find command also has the ability to count. Invoked with the /c command, it'll count the number of lines of its output that include a given string. Users often want to count the number of lines in the output of a command to determine how many processes are running, how many startup items are present, or a variety of other interesting tidbits on a machine. To count the lines of output, users could simply pipe their output through find /c /v "". This command will count (/c) the number of lines that do not have (/v) a blank line ("") in them. By counting the number of non-blank lines, the command is, in effect, counting the number of lines.

Now, with the find command, users can look through the output of each of the commands I've discussed so far to find interesting tidbits. For example, to look at information every second about cmd.exe processes running on a machine, type:

```
C:\> wmic process list brief /every:1 | find "cmd.exe"
```

Or, to see which autostart programs are associated with the registry hive HKLM, run:

```
C:\> wmic startup list brief | find /i "hklm"
```

To count the number of files open on a machine on which openfiles accounting is activated, type:

```
C:\> openfiles /query /v | find /c /v ""
```

Whenever counting items in this way, remember to subtract the number of lines associated with column headers. And, as a final example, to see with one-second accuracy when TCP port 2222 starts being used on a machine, along with the process ID using the port, run:

```
C:\> netstat -nao 1 | find "2222"
```

Researching output

With these five tools, users can get a great deal of information about the configuration and security state of a Windows machine. To use each command in identifying a compromise, however, a user needs to compare the current settings of the machine under analysis to a "normal," uninfected machine.

There are three options for establishing a baseline comparison. First, if the user is an experienced malware hunter, he or she may have a sense of what is right and what is wrong with a given kind of machine, identifying evil or unusual stuff based on experience. Alternatively, this comparison can be performed against a clean, uninfected machine, if there is one handy. If there isn't, a user may need to rely on a third option -- researching specific files, process n