PsExec and PsExec v2.11 by Mark Russinovich

Execute processes on a remote system and redirect output to the local system

PsExec is a command-line tool that lets you execute processes on remote systems and redirect console applications' output to the local system so that these applications appear to be running locally. You can download PsExec for free from the Sysinternals website. Here are some advanced tips and tricks to help you leverage the full potential of PsExec as a systems management utility.

The PsTools Suite

PsExec is a member of Sysinternals' PsTools suite, which contains 11 tools. To be in the suite, tools must conform to a set of specifications that includes supporting Windows NT 4.0 and later, being a console application, and having the ability to work on the local system as well as on a remote one. PsTools utilities require no manual installation of software on the remote system, and they let you specify alternative credentials to access the remote system.

Incidentally, the reason that the suite is named PsTools and that all the member tools have Ps as a prefix to their name is that the first tool I developed that satisfied the listed criteria was PsList, a program that lists running processes. I named the tool after the ps utility that performs the same function on UNIX systems.

As with many of the tools in the PsTools suite, PsExec's ability to run processes remotely requires that both the local and remote computers have file and print sharing (i.e., the Workstation and Server services) enabled and that the default Admin$ share (a hidden share that maps to the \windows directory) is defined on the remote system. The reasons for these requirements will become clear later when I describe how PsExec works.

PsExec

PsExec's ability to run processes remotely with no manual installation of software on the remote system makes deployment easy. However, if PsExec were only able to launch a program on a remote system, its usefulness would be limited. PsExec's ability to redirect the input and output of console applications is what makes the tool a versatile systems management utility. Figure 1 shows PsExec's command-line options and gives a hint as to its capabilities. Many Windows administrative console tools can run only on a local machine. PsExec lets you remote-enable any of them. For example, PsExec lets Ipconfig, the Windows utility that displays the TCP/IP configuration for a system's network adapters, show a remote system's configuration. A sample command for that use is

```
psexec \\remote ipconfig
```

where remote is the name or IP address of the system you want to query. You'll see Ipconfig's output as if you had run Ipconfig on the local machine.

If you don't specify the path of the program you want to execute, PsExec looks in the \windows\system32 directory of the remote system. If you know that the program isn't in that directory, enter its full path on the remote system; if it's an executable on the local system that you want to execute on the remote system, specify the -c switch and the file's local path. The -c switch directs PsExec to copy the specified executable to the remote system for execution and delete the executable from the remote system when the program has finished running.

An even more powerful use of PsExec's console-redirection capability is to run a command prompt on a remote system as if the command prompt were running locally. This use of PsExec is similar to running a Telnet client on the local machine and connecting to a Telnet service on the remote machine, but you don't need to have the Telnet service, or any other special service, running on the remote system. Simply execute the command:

```
psexec \\remote cmd
```

If you want to execute one console command on the remote system, pass the command prompt the /c switch followed by the command you want to execute. For example, the command

```
psexec \\remote cmd /c ver
```

displays the Windows version number of the remote system on the local machine's console.

Another popular use of PsExec is to deploy hotfixes or other patches that support a noninteractive interface across your network. To make this task even easier, PsExec takes multiple computer names, the name of a text file containing a list of computer names, or the special name of * that results in an enumeration of all the computers in the current domain. For instance, to execute the Microsoft MyDoom removal tool on computers named Remote and Remote1 and log the exit status of the cleanup to a file, you could use the command

```
psexec \\remote,remote1
        -c doomcln.exe
        -s 2> results.log
```

Upon exit, a process specifies an integer that the process's parent process can read. Programs often use the exit code to report the success or failure of their execution. Whenever a process executed with PsExec is completed, PsExec displays the process's exit code and returns the exit code as its own exit code. You should test a program's behavior or check its documentation to determine what that program's specific error codes mean, but an exit code of 0 typically means success. The -s switch specifies that PsExec should execute the command under the System account. I'll discuss this option more in a moment.

PsExec Security

You should be aware of several ways in which PsExec interfaces with Windows security. By default, the process you execute on the remote system impersonates the account from which you run PsExec on the local system.

Impersonation is somewhat restricted from the perspective of security—the remote process doesn't have access to any network resources, even those that your account typically would be able to access. If the account in which you're running doesn't have local administrative privileges on the remote system, the process you want to run requires access to network resources, or you want to run a process in a different account, then use PsExec's -u switch to provide an alternative account name. For example, you could enter the command

```
psexec \\remote
-u remote\administrator
-p adminpass ipconfig
```

to run Ipconfig under the Administrator account on the remote machine. Use the -p switch to enter the password for the account you specified with the -u switch. If you omit the -p switch, PsExec prompts you to enter the password (for security reasons, PsExec doesn't echo the password you enter to the screen).

If you specify alternative credentials, the remote process runs with those credentials and will have access to network resources that the alternative account can access. To run in a different account, PsExec must use that account to log on to the remote system. PsExec therefore requires the password on the remote system and sends the password in clear text from the local system. You need to be aware of this fact if unauthorized network sniffers could intercept traffic between the local and remote system.

You can also run the remote process in the System account, under which Windows services and core Windows processes, such as Winlogon and the Local Security Authority Subsystem Service (LSASS) are executed. The System account has powerful privileges. Some file-system and registry resources have default security settings that permit access only from the System account—examples are the HKEY_LOCAL_MACHINE\SAM registry subkey and the \System Volume Information directory that's present on each volume of all Windows 2000 or later systems.

For example, if you've ever been curious about the contents of the SAM subkey, which appears empty in regedit because regedit can navigate the subkey only under the System account, you can use PsExec similarly to the way you use the Runas command (which is available on Win2K and later) to run regedit under the System account. The command is this:

```
psexec -s -i

c:\windows\regedit.exe
```

Note that the command doesn't include a remote computer name and does include the -i (interactive) switch. When you enter the command, regedit will appear on your desktop running in the System account, and you'll be able to look inside the HKEY_LOCAL_MACHINE\SAM and HKEY_LOCAL_MACHINE\SYSTEM subkeys. The -i switch is what causes regedit to appear on the console desktop, and it's typically useful only when you want to run a GUI application on the local system where you can interact with it.

A last security note relates to viruses. Several viruses use PsExec to propagate within a network, and as a result, several major antivirus products flag PsExec as a Trojan horse program or a worm. Remember that PsExec works on remote systems only if it runs within an account that has administrator group membership on the remote system. In other words, unless the account from which you run it has administrative access to a remote system, PsExec won't be able to execute a process on the remote system. In addition, PsExec's functionality can be achieved in other ways; thus, PsExec is only a convenience for virus writers, who could otherwise easily implement the functionality that PsExec provides.

Inside PsExec

PsExec starts an executable on a remote system and controls the input and output streams of the executable's process so that you can interact with the executable from the local system. PsExec does so by extracting from its executable image an embedded Windows service named Psexesvc and copying it to the Admin$ share of the remote system. PsExec then uses the Windows Service Control Manager API, which has a remote interface, to start the Psexesvc service on the remote system.

PsExec v2.11

Utilities like Telnet and remote control programs like Symantec's PC Anywhere let you execute programs on remote systems, but they can be a pain to set up and require that you install client software on the remote systems that you wish to access. PsExec is a light-weight telnet-replacement that lets you execute processes on other systems, complete with full interactivity for console applications, without having to manually install client software. PsExec's most powerful uses include launching interactive command-prompts on remote systems and remote-enabling tools like IpConfig that otherwise do not have the ability to show information about remote systems.

Note: some anti-virus scanners report that one or more of the tools are infected with a "remote admin" virus. None of the PsTools contain viruses, but they have been used by viruses, which is why they trigger virus notifications.

Installation

Just copy PsExec onto your executable path. Typing "psexec" displays its usage syntax.

Using PsExec

-a	Separate processors on which the application can run with commas where 1 is the lowest numbered CPU. For example, to run the application on CPU 2 and CPU 4, enter: "-a 2,4"
-c	Copy the specified program to the remote system for execution. If you omit this option the application must be in the system path on the remote system.
-d	Don't wait for process to terminate (non-interactive).
-e	Does not load the specified account's profile.
-f	Copy the specified program even if the file already exists on the remote system.
-i	Run the program so that it interacts with the desktop of the specified session on the remote system. If no session is specified the process runs in the console session.
-h	If the target system is Vista or higher, has the process run with the account's elevated token, if available.
-l	Run process as limited user (strips the Administrators group and allows only privileges assigned to the Users group). On Windows Vista the process runs with Low Integrity.
-n	Specifies timeout in seconds connecting to remote computers.
-p	Specifies optional password for user name. If you omit this you will be prompted to enter a hidden password.
-r	Specifies the name of the remote service to create or interact with.
-s	Run the remote process in the System account.
-u	Specifies optional user name for login to remote computer.
-v	Copy the specified file only if it has a higher version number or is newer on than the one on the remote system.
-w	Set the working directory of the process (relative to remote computer).
-x	Display the UI on the Winlogon secure desktop (local system only).
-priority	Specifies -low, -belownormal, -abovenormal, -high or -realtime to run the process at a different priority. Use -background to run at low memory and I/O priority on Vista.
computer	Direct PsExec to run the application on the remote computer or computers specified. If you omit the computer name, PsExec runs the application on the local system, and if you specify a wildcard (*), PsExec runs the command on all computers in the current domain.
@file	PsExec will execute the command on each of the computers listed in the file.

cmd	Name of application to execute.
arugments	Arguments to pass (note that file paths must be absolute paths on the target system).
-accepteula	This flag suppresses the display of the license dialog.

You can enclose applications that have spaces in their name with quotation marks e.g.

```
psexec \\marklap"c:\long name app.exe"
```

Input is only passed to the remote system when you press the Enter key. Typing Ctrl-C terminates the remote process.

If you omit a user name, the process will run in the context of your account on the remote system, but will not have access to network resources (because it is impersonating). Specify a valid user name in the Domain\User syntax if the remote process requires access to network resources or to run in a different account. Note that the password and command are encrypted in transit to the remote system.

Error codes returned by PsExec are specific to the applications you execute, not PsExec.

Recommended Internet Sites

- PsExec by Mark Russinovich: http://windowsitpro.com/systems-management/psexec
- PsExec v2.11 by Mark Russinovich: https://web.archive.org/web/20161114153514/https://technet.microsoft.com/en-us/sysinternals/pxexec.aspx

Please contact the Course Coordinators if you are unable to access any of the Recommended Internet Sites.