Understanding PSEXEC.exe

Used by hackers loved by admins

Lowell Vanderpool & Nathan Vanderpool
TECH SAVVY PRODUCTIONS

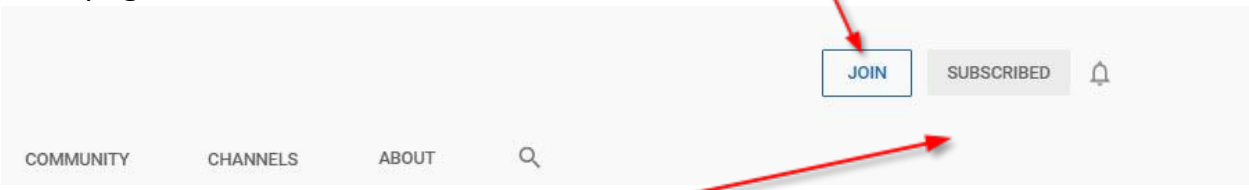
CONTACT US:

mrvanderpool@techsavvyproductions.com

nathan@techsavvyproductions.com

SUPPORT US:

If you would like to support the channel, Join our channel membership, it's $2.99/month (less than a Starbucks coffee); see the "Join" button on our channel homepage.



OR

Subscribe to the channel as it helps our channel perform better on YouTube's algorithm.

SOCIAL MEDIA AND WEBSITE:

Check out our **Website**: https://www.techsavvyproductions.com

Follow us on **Twitter:** @_TechSavvyTeam

Like us on **Facebook:**
https://www.facebook.com/Tech-Savvy-Productions-105287381500897

Mr.V **Linkedin**: https://www.linkedin.com/in/lowell-vanderpool-57970623/

Nathan **Linkedin**: https://www.linkedin.com/in/nathan-vanderpool-50a27822/

Follow on **Instagram**: techsavvyproductions
https://www.instagram.com/techsavvyproductions/

Tech Savvy Productions

We translate subtitles on our videos into the following languages: عربى, БългарскиВ, 简体中文), 中國傳統的）, Nederlands, Suomalainen, français, Deutsche, हिंदी , , bahasa Indonesia, 日本語, 한국어, norsk, Polskie, português, Română, український, eesti keel, Española, lietuvių, latviski , Svenska, čeština, dansk, Ελληνικά, հայերեն, shqiptare, and Tiếng Việt

Social media logos and “Tech Savvy Productions” teaser created by The 11th Hour: https://www.youtube.com/user/The11thH0UR/featured

Quick Access to Links in Document:

PsExec v2.34

PsExec Explainer by Mark Russinovich

PsExec: What It Is and How to Use It

How to use PsExec

How to run PowerShell commands remotely with PsExec

Everything You Wanted to know About Psexec

Lateral movement: A deep look into PsExec

How to: become the LOCAL SYSTEM account with PsExec

Windows's Registry: Understand and Troubleshoot

Cool Command tools to use remotely

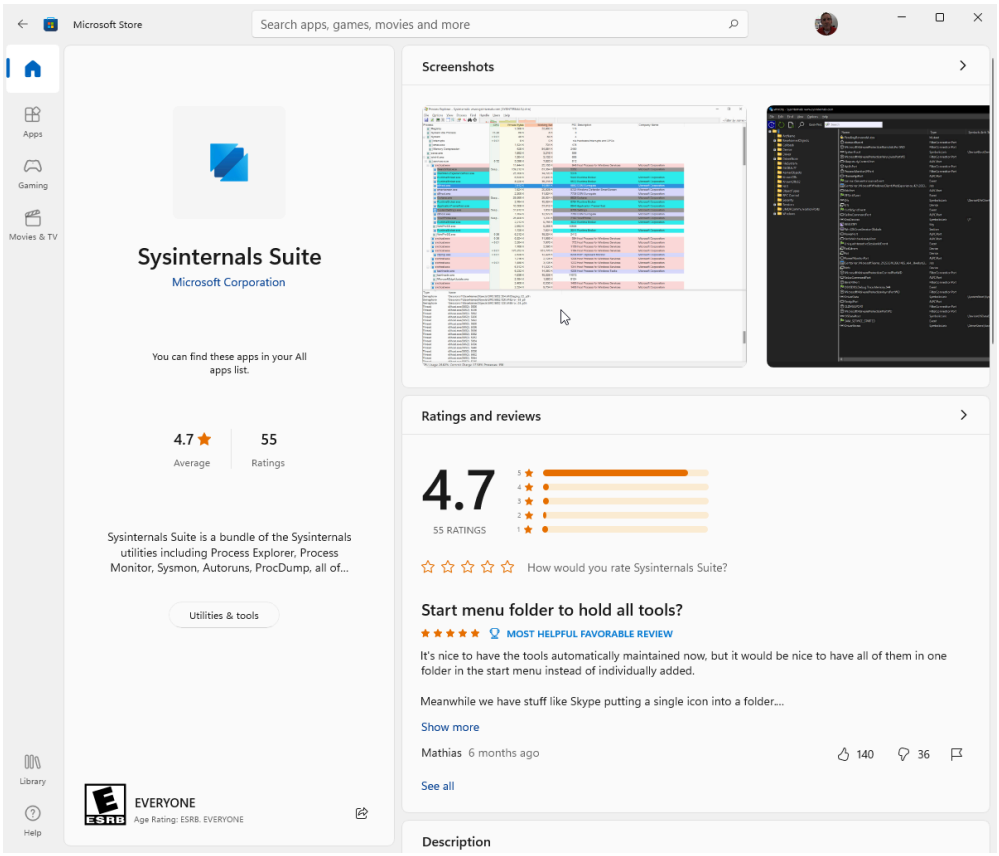
Is someone connected via the network to the local PC?

Remote removal, upgrade and install Firefox across a domain

Use the Microsoft Store to install Sysinternals Suite!

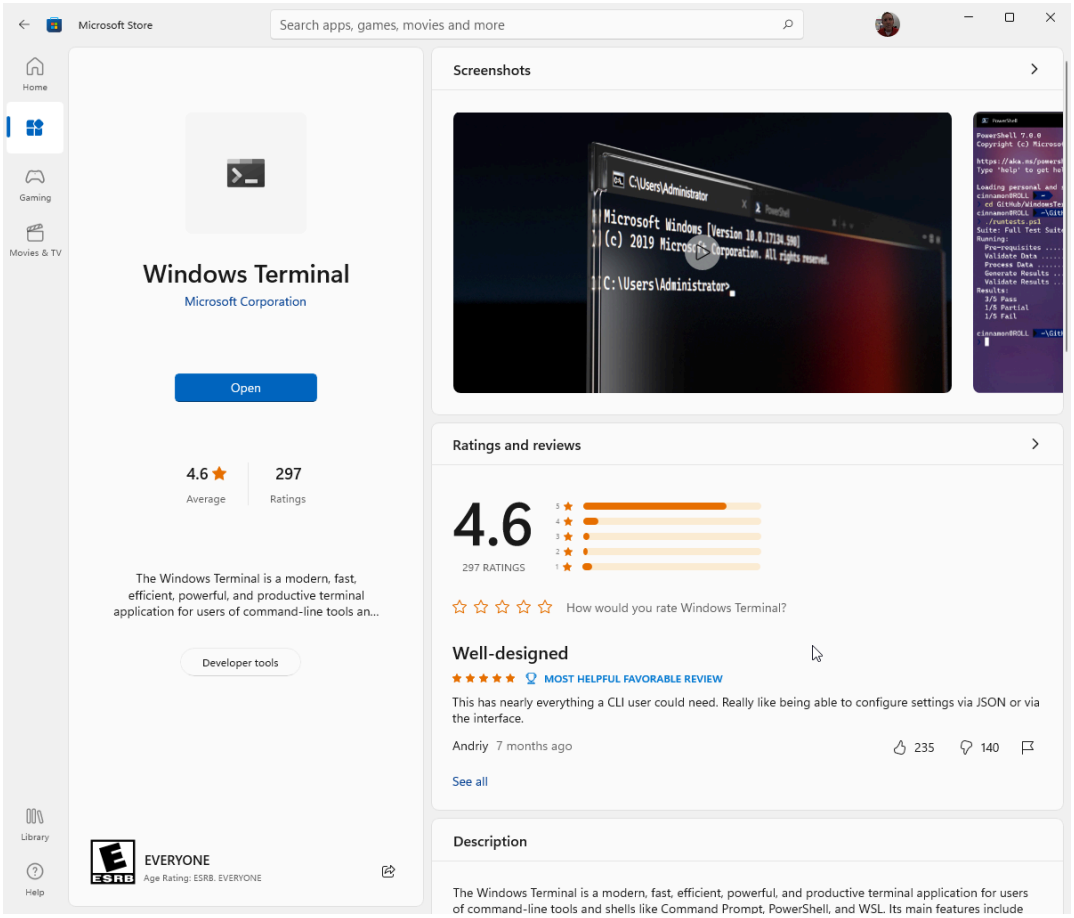
Whether you're an IT Pro or a developer, you'll find Sysinternals utilities to help you manage, troubleshoot and diagnose your Windows systems and applications.

This automatically set the environment variables in the local PC for proper function and populates your app menu for all the GUI tools.



Windows Terminal

Gives you a powerful way of launching and using PowerShell and CMD.



Why are PsTools such a great set of utilities for IT professionals?

Sysinternals PsTools is a suite of 12 Microsoft Windows management utilities with common characteristics:

They are all console utilities.

That is, they are designed to run at a command prompt, PowerShell or from a batch file, and they write to the standard output and standard error streams (which can appear in the console window or be redirected to files).

They can operate on the local computer or on a remote computer.

Single Executable Images: "portable apps"

They do not require software to be installed on the host or remote host. Once the utility completes its function it terminates handles, threads and processes.

Unlike most remote-control programs, the PsTools utilities do not require preinstallation of client software on the remote systems. And of course, like all other Sysinternals utilities, they require no installation on the local computer either.

They provide a standard syntax:

Alternate credentials so that the utilities' tasks can be performed as another user.

They can easily be installed via the Microsoft Store

Both Sysinternals and Windows Terminal are now available via the Microsoft Store.

You can also download any of the Sysinternal tools via the Sysinternals Live web-site https://live.sysinternals.com/

Exit and error codes

When you use PsExec to run a program (or a process), it might display an exit code for that program and returns an error code.

The error code returned is not generated by PsExec; it is generated by the original program that you run using PsExec.

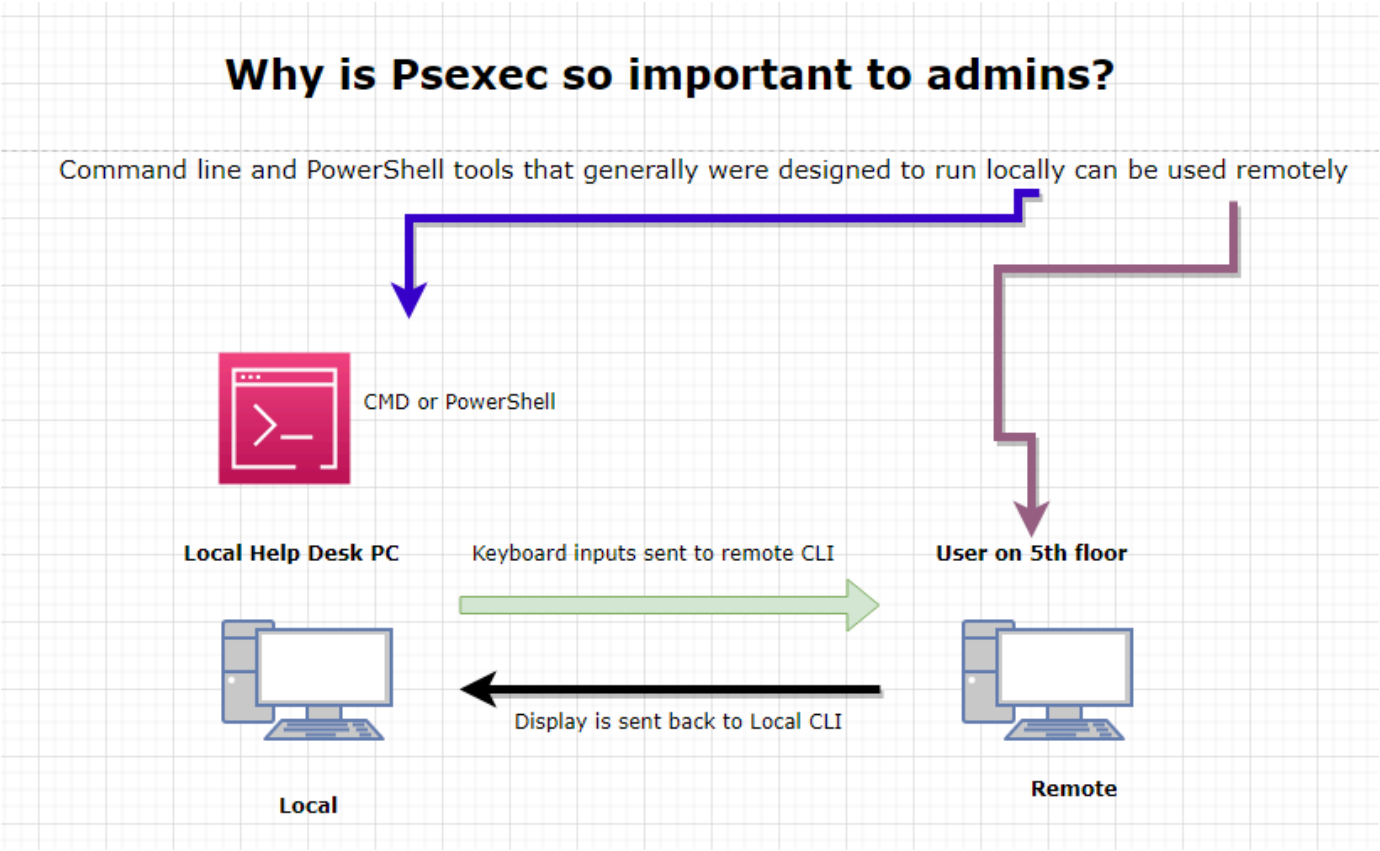
So, if you get any error while running a program, you should consult the program's documentation to determine what that error code means.

An **exit code of 0** typically indicates successful execution.

Most of the Sysinternals utilities that use a kernel-mode driver extract the driver file to %SystemRoot%\System32\Drivers, load the driver, and then delete the file. The driver image remains in memory until the system is shut down. When running a newer version of a utility that has an updated driver, a reboot might be required to load the new driver

The utilities included in the PsTools suite are:

- **PsExec** Executes processes remotely, as a built-in service account such as Local System with redirected output, or both
- **PsFile** Lists or closes files opened remotely
- **PsGetSid** Translates the name of a computer, user, or group to its corresponding Security Identifier (SID), and vice versa
- **PsInfo** Lists information about a system
- **PsKill** Terminates processes by name or by process ID (PID)
- **PsList** Lists information about processes
- **PsLoggedOn** Lists accounts that are logged on locally and through remote connections
- **PsLogList** Dumps event log records
- **PsPasswd** Sets passwords for user accounts
- **PsService** Lists and controls Windows services
- **PsShutdown** Shuts down, logs off, or changes the power state of local and remote systems
- **PsSuspend** Suspends and resumes processes



Psexec allows the IT professional to:

1. Using command-line tools execute them remotely and have the output piped back to their admin host
2. Allows admins to run a tool or tools against multiple hosts
    - Separate computer names by comas
    - Use a text file with a list
3. Allows admins to use both Microsoft and 3rd-party tools to be used with psexec.exe
4. Allows many tools and utilities designed for the local machine to be used remotely
5. Move utilities, batch files or PowerShell files to a remote host and execute
6. Works with PowerShell
7. Can be executed inside a batch file or PowerShell script

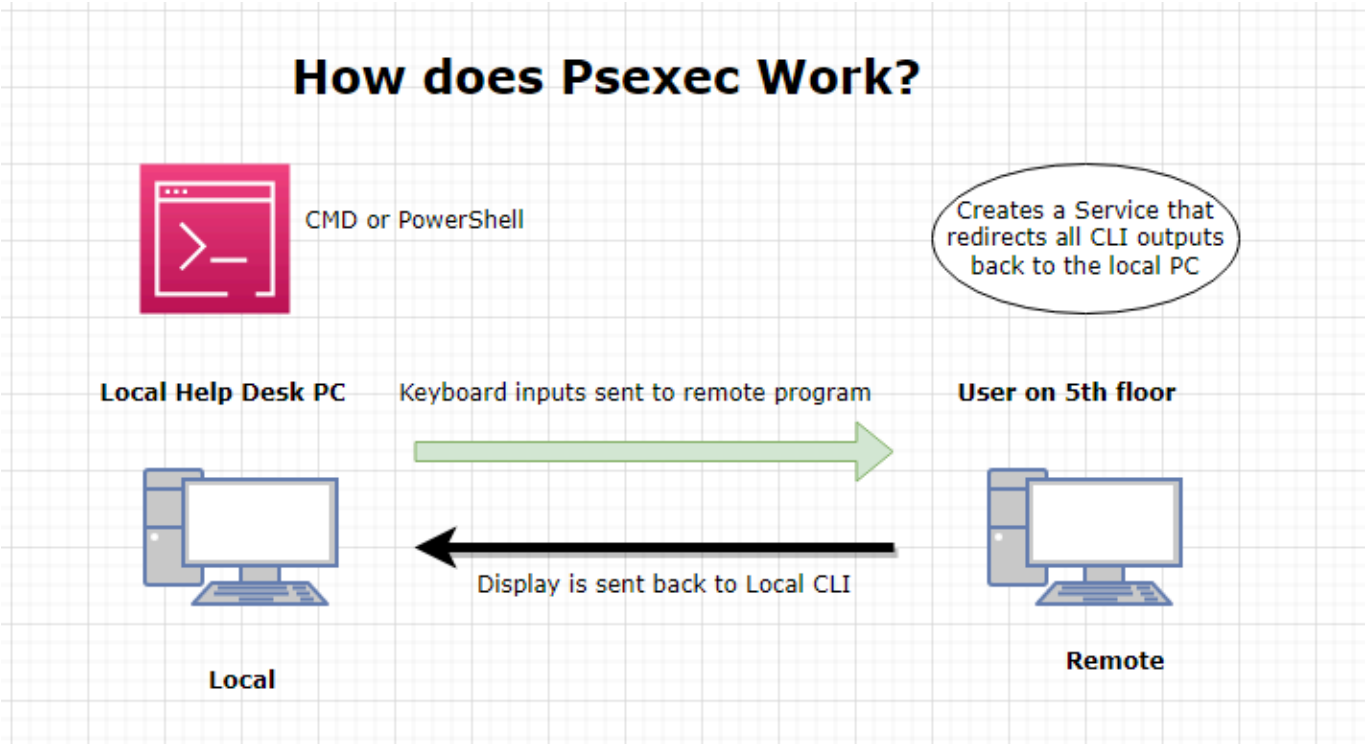
8. Can launch both CLI and GUI programs with **System account** on the Local and remote hosts
9. Can launch GUI programs on different sessions on the remote host (cannot redirect output)
10. Can install software applications remotely.
11. You can control what CPUs are used in executing programs and priority level.
12. You can execute using alternative credentials.
13. They work in domain and non-domain environments

PsExec v2.34

PsExec Executes processes remotely, as a built-in service account such as Local System with redirected output, or both.

PsExec lets you execute arbitrary processes on one or more remote computers. PsExec redirects the input and output streams of console applications so that they appear to be running locally, as though in a Telnet session. In this way, console utilities that normally operate only on the local computer can be remote-enabled. A particularly powerful use of this capability is to run a command prompt on a remote system and interact with it as though it were running on the local computer. Unlike most remote-control utilities, PsExec does not require installation of agents or other client software on the target computer ahead of time. Of course, you do need an account that is authorized for remote administration of the computer

You can also use PsExec to execute programs locally or remotely in the System account, either interactively or noninteractively. PsExec offers many other options that control the way in which the local or remote target process should run, including user account, privilege level, priority level, and CPU assignment.



1 Corinthians 13:4-6

New International Version

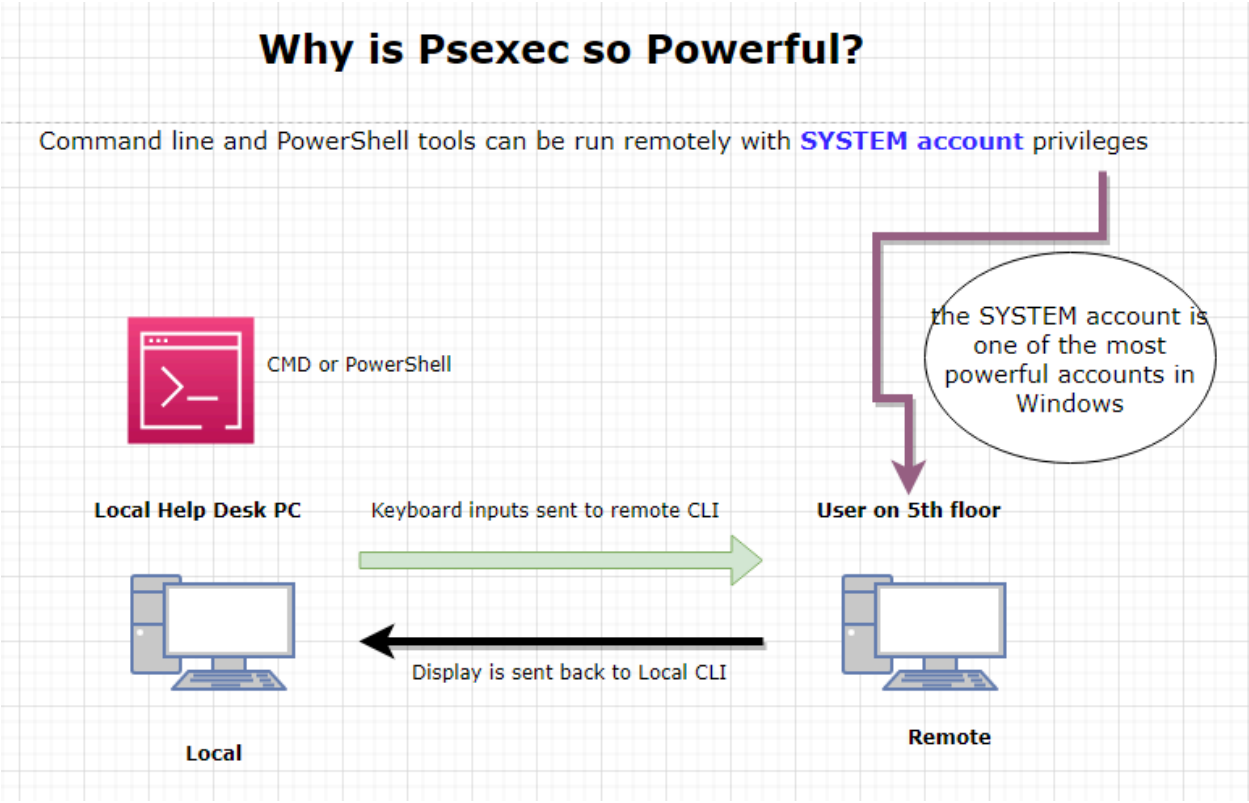
[4] Love is patient, love is kind. It does not envy, it does not boast, it is not proud. [5] It does not dishonor others, it is not self-seeking, it is not easily angered, it keeps no record of wrongs. [6] Love does not delight in evil but rejoices with the truth.

Introduction

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

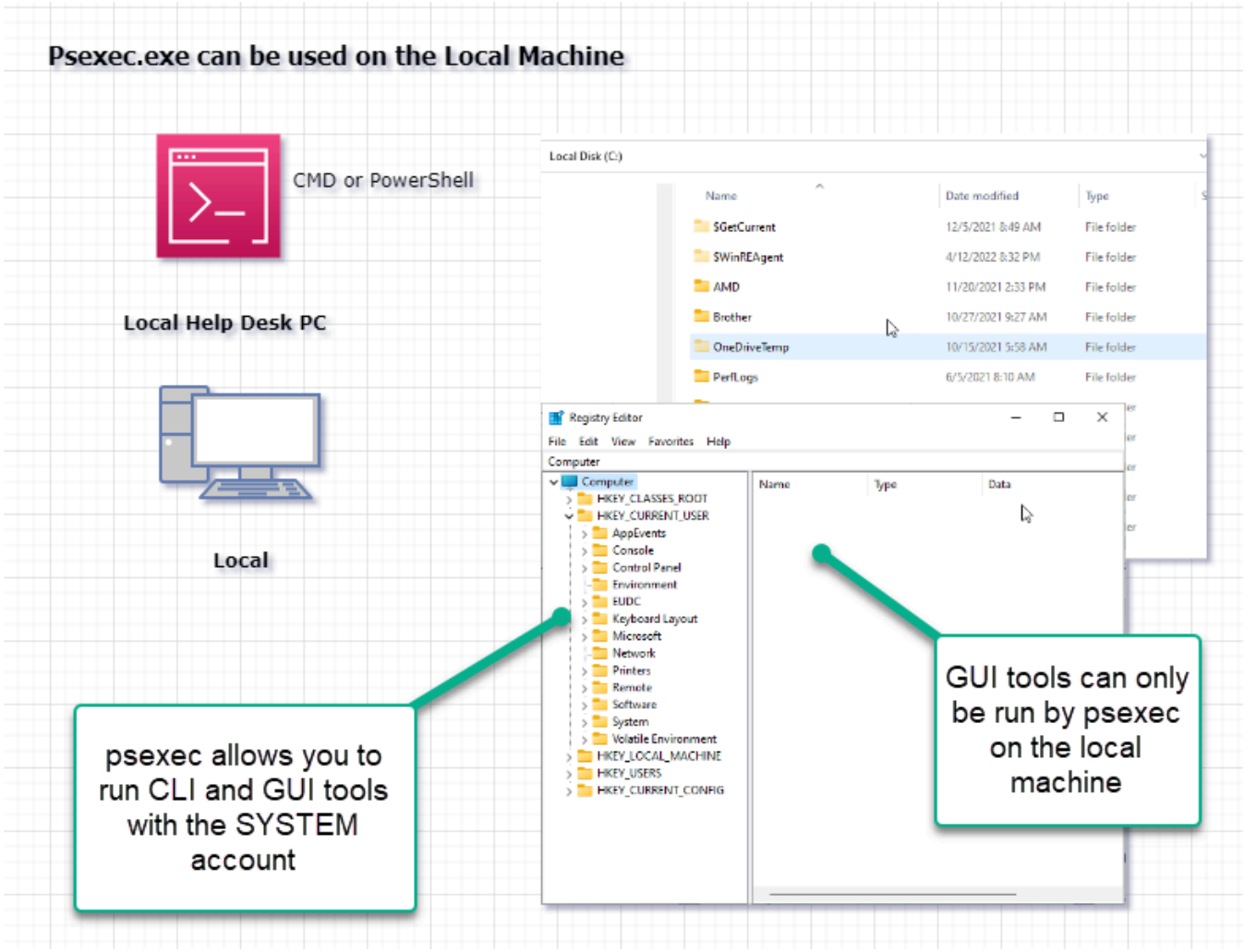
require that you install client software on the remote systems that you wish to access. PsExec is a light-weight telnet-replacement that lets you execute processes on other systems, complete with full interactivity for console applications, without having to manually install client software. PsExec's 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.



Psexec Usage syntax:

```
psexec [\\computer[,computer2[,...] | @file]][-u user
[-p psswd][-n s][-r servicename][-h][-l][-s|-e][-x][-i
[session]][-c executable [-f|-v]][-w
directory][-d][-<priority>][-a n,n,...] cmd [arguments]
```

Command-line syntax key

The following table describes the notation used to indicate command-line syntax.

Notation	Description
Text without brackets or braces	Items you must type as shown.
`<Text inside angle brackets>`	Placeholder for which you must supply a value.
`[Text inside square brackets]`	Optional items.
`{Text inside braces}`	Set of required items. You must choose one.
Vertical bar (|)	Separator for mutually exclusive items. You must choose one.
Ellipsis (...)	Items that can be repeated and used multiple times.

PSEXEC [OPTIONS] PROGRAM [ARGUMENTS]

If the "program" part of the command line contains spaces, you must put quotation marks around the program path.

If parts of the remote command line include special characters such as the pipe or redirection characters, use the command shell's escape character—caret (^) for Cmd.exe, and backtick (`) for PowerShell—to prevent their being treated as special characters by the local command shell.

The following Command Prompt example runs **ipconfig /all** on server1 and redirects its standard output to **C:\ipconfig.out** on server1:

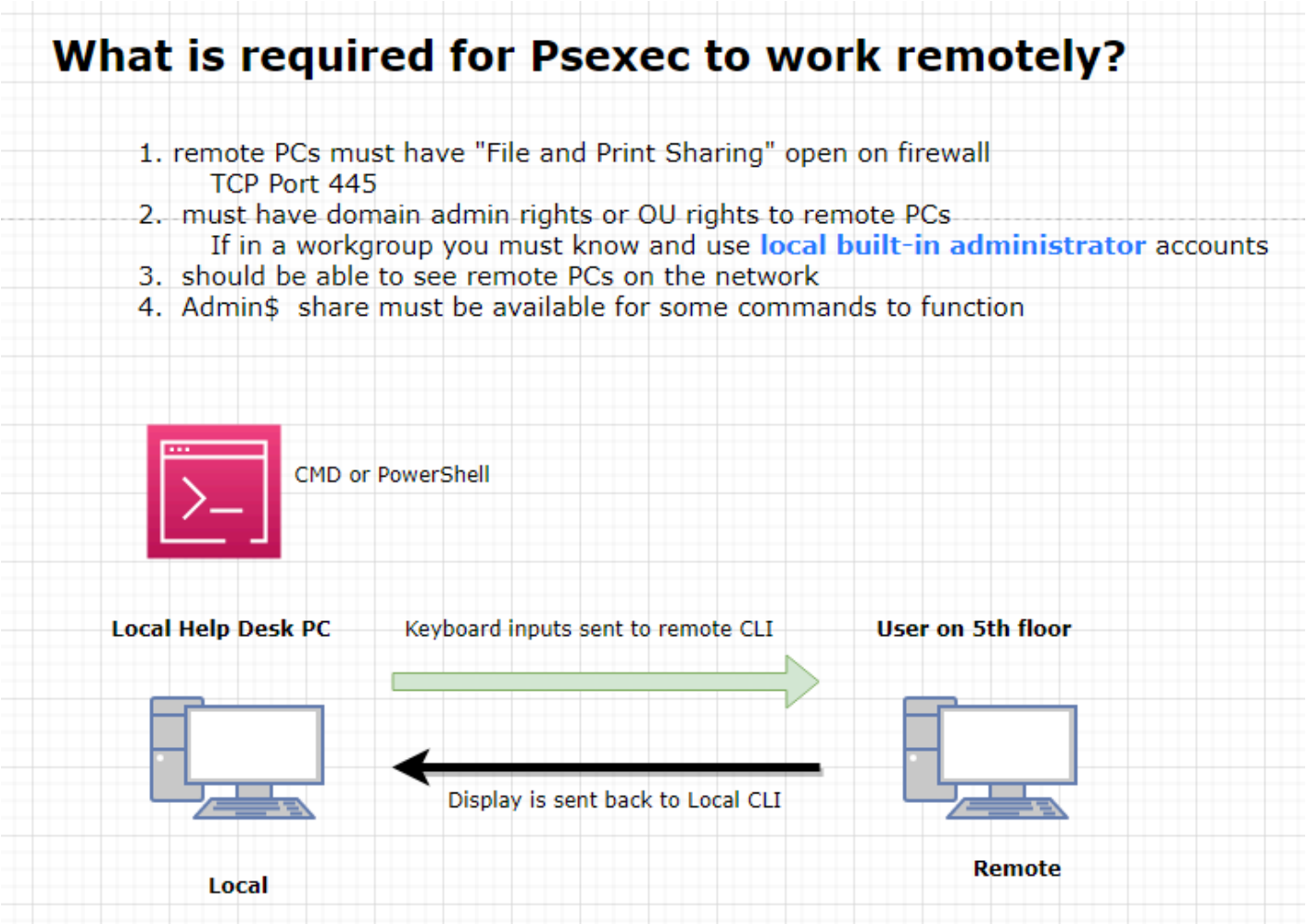
PSEXEC \\SERVER1 CMD.EXE /C IPCONFIG /ALL ^> C:\IPCONFIG.OUT

Without the escape character (^), the standard output of the PsExec command (including the redirected console output of *ipconfig*) would be written to **c:\ipconfig.out** on the local computer.

(PsExec's diagnostic output is written to its standard error stream rather than to its

standard output so that local redirection captures only the output of the remote process.)

If the “program” part of the PsExec command line specifies only a file name, it must be found in the Path on the remote system.



Option	Description
–d	Doesn't wait for the process to terminate. (This is described earlier in the "Remote process exit" section.)
Process performance options	
–background –low –belownormal –abovenormal –high –realtime	Runs the process at a different priority.
–a *n,n...*	Specifies the CPUs on which the process can run.
Remote connectivity options	
–c [–f|–v]	Copies the specified program from the local system to the remote system. If you omit this option, the application must be in the system path on the remote system. Adding **–f** forces the copy to occur even if it already exists on the remote system and is marked read-only; **–v** performs a version or timestamp check and copies only if the source is newer.
–n *seconds*	Specifies the timeout in seconds when connecting to remote computers.
Runtime environment options	
–s	Runs the process in the System account.
–i [*session*]	Runs the program on an interactive desktop.
–x	Runs the process on the Winlogon secure desktop.
–r *servicename*	Specifies the name of the remote PsExec service and executable.
–w *directory*	Sets the working directory of the process.
–e	Does not load the specified account's profile.
–h	Uses the account's elevated context, if available.
–l	Runs the process as a limited user.

Parameter	Description
-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 executable 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. This flag is **required** when attempting to run console applications interactively (with redirected standard IO).

If you specify the -d (don't wait) switch, the service exits after starting the executable; otherwise, the service waits for the executable to terminate, then sends the exit code back to PsExec for it to print on the local console.

The switch -c which allows a file to be copied to the remote is powerful, executed and then deleted: -f overwrites the remote file and -v check the version of the file

Parameter	Description
	If you want to interact with a command prompt or PowerShell interface running on the remote computer is switch is required.
-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.**
	Can be used against a local machine and is required for most remote actions if in a workgroup (non-domain)
-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.

Parameter	Description
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.
arguments	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.*

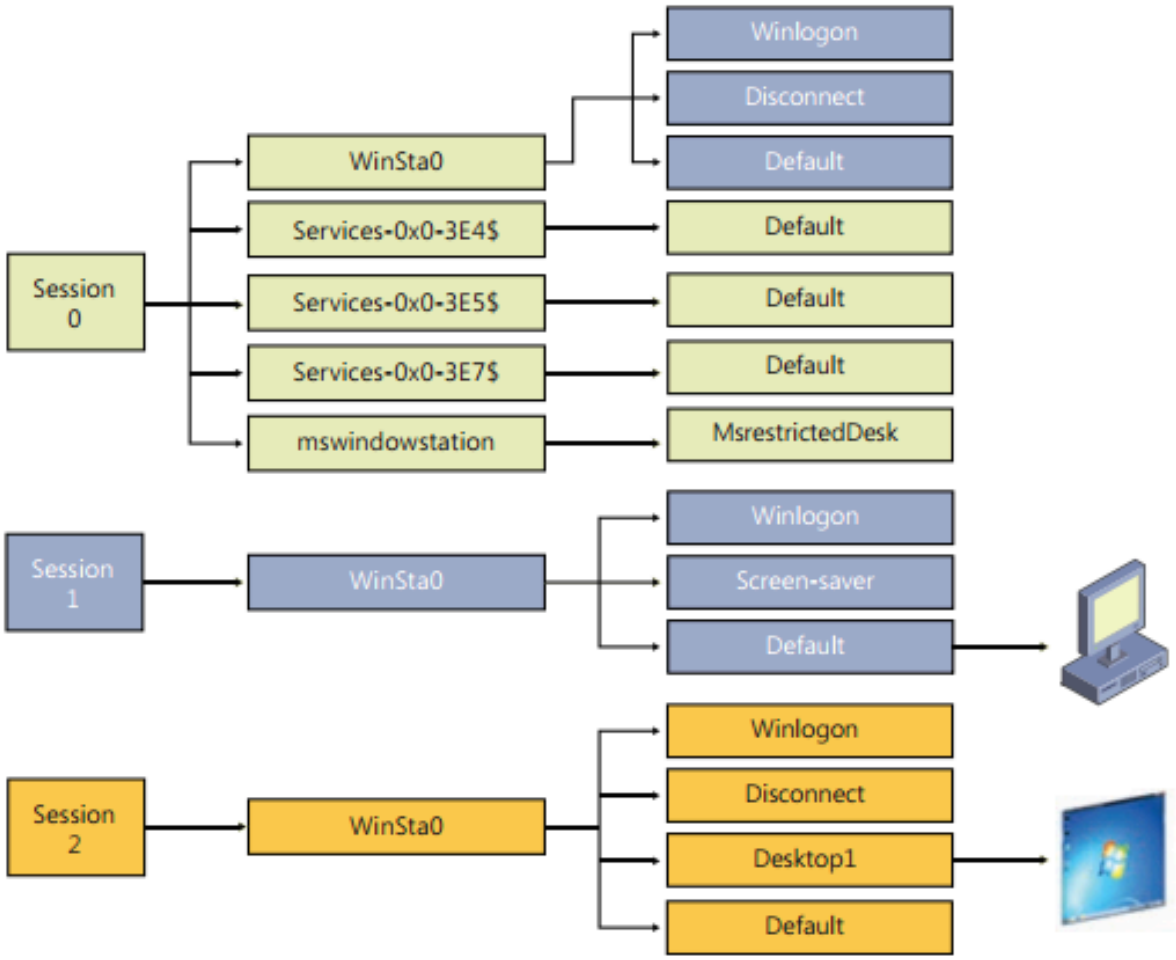
You can enclose applications that have spaces in their name with quotation marks e.g.

Micah 6:8

New International Version

8 He has shown you, O mortal, what is good.
And what does the LORD require of you?
To act justly and to love mercy
and to walk humbly[a] with your God.

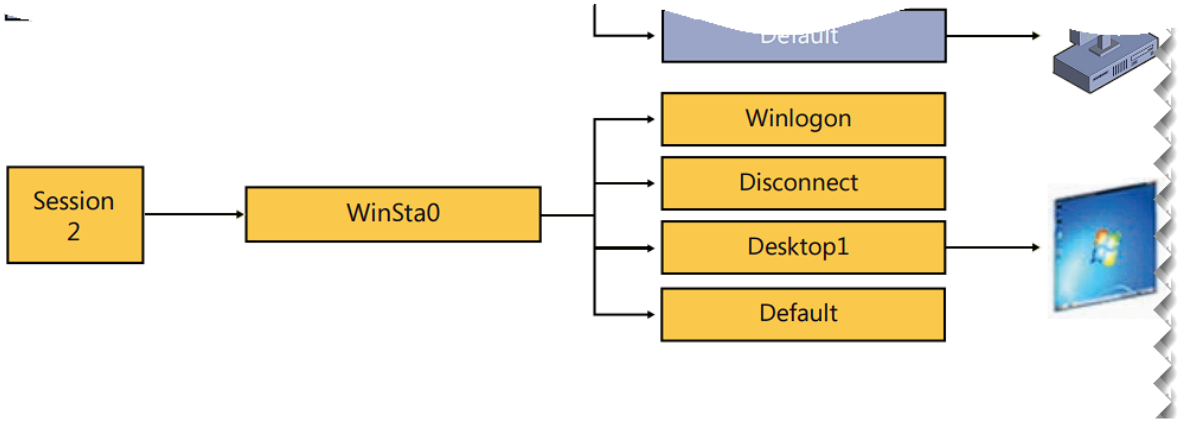
Sessions, window stations, desktops, and window messages

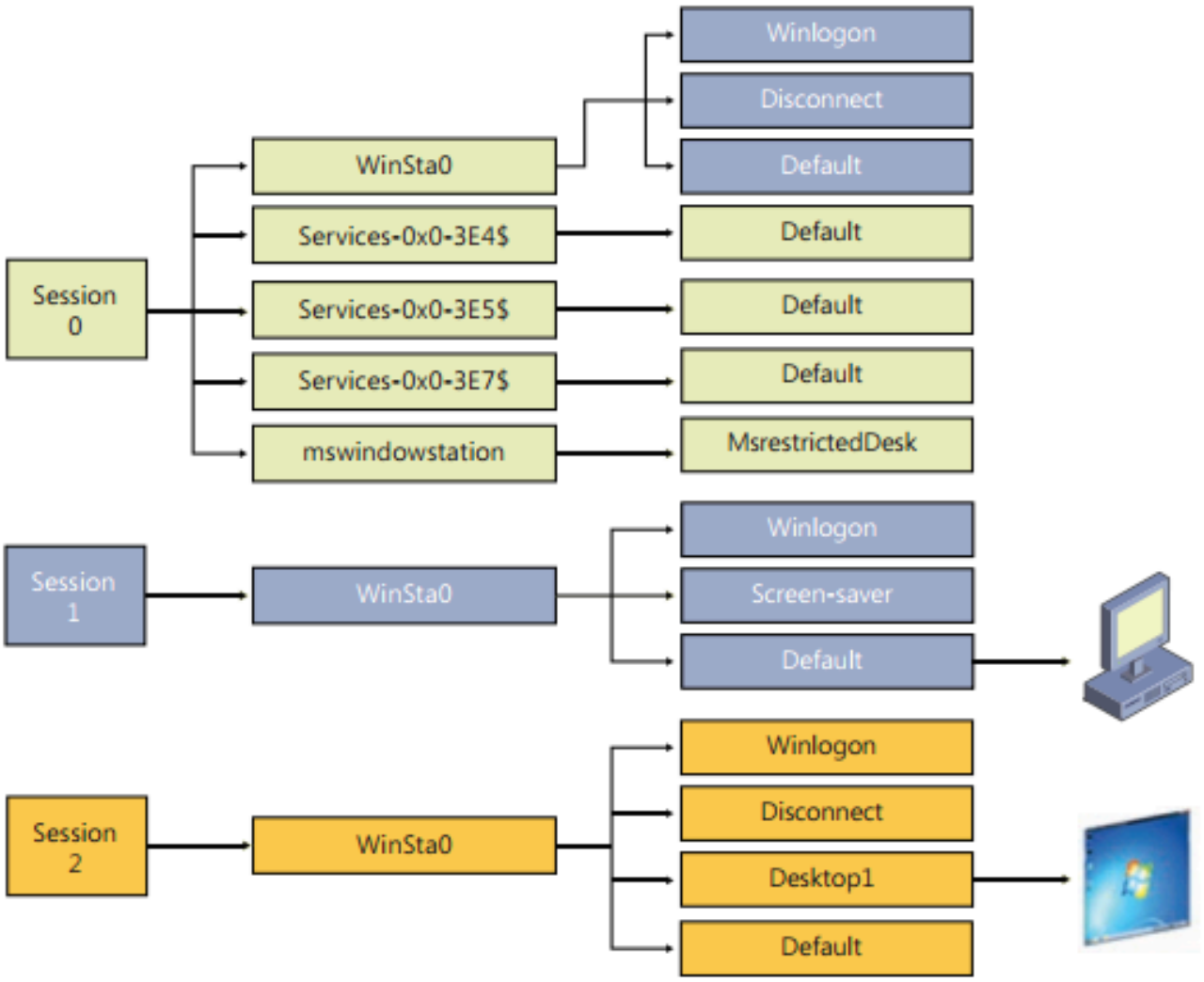


The descriptions of several of the Sysinternals utilities—including Process Explorer, Process Monitor, PsExec, AdInsight, Desktops, and LogonSessions—refer to sessions, session IDs, the "console session," and "session 0"; interactive and noninteractive window stations; and other programs running on the "same desktop."

These concepts, although not widely understood, can be critical to problem solving on the Windows platform. Let's start with an overview of the hierarchy, an example of which is depicted in Figure below, and then define the terms.

- At the bottom of the graphic are remote desktop services (RDS) sessions, formerly known as terminal services (TS) sessions. Each session contains one or more window stations, which contain desktops. Each of these securable objects has resources allocated for its sole use. There is also a loose relationship between these and logon sessions created by the LSA. Although Windows documentation doesn't always make a clear distinction between LSA logon sessions and RDS sessions, they are separate entities.





Remote desktop services sessions

Remote desktop services support multiple interactive user sessions on a single computer. Introduced in Windows NT 4.0 Terminal Server Edition, they were not incorporated into the Windows client operating system family until Windows XP. Features they support include Fast User Switching, Remote Desktop, Remote Assistance, Remote Applications Integrated Locally (RAIL, a.k.a. RemoteApps), and virtual machine integration features. An important limitation of Windows clients (Windows XP, Windows Vista, Windows 7, Windows 8.x, and Windows 10) is that only one interactive session can be active at a time. That is, while processes can continue to run in multiple disconnected sessions

simultaneously, only one session can update a display device and receive keyboard and mouse input. A further limitation was that a *domain-joined* Windows XP computer supported at most only one interactive session. For example, if a user were logged on at the console, you could log on to the computer via Remote Desktop using the same account and continue that session, but you could not log on with a different user account unless the first user were logged off. Remote desktop services sessions are identified by an incrementing numeric session ID, starting with session 0.

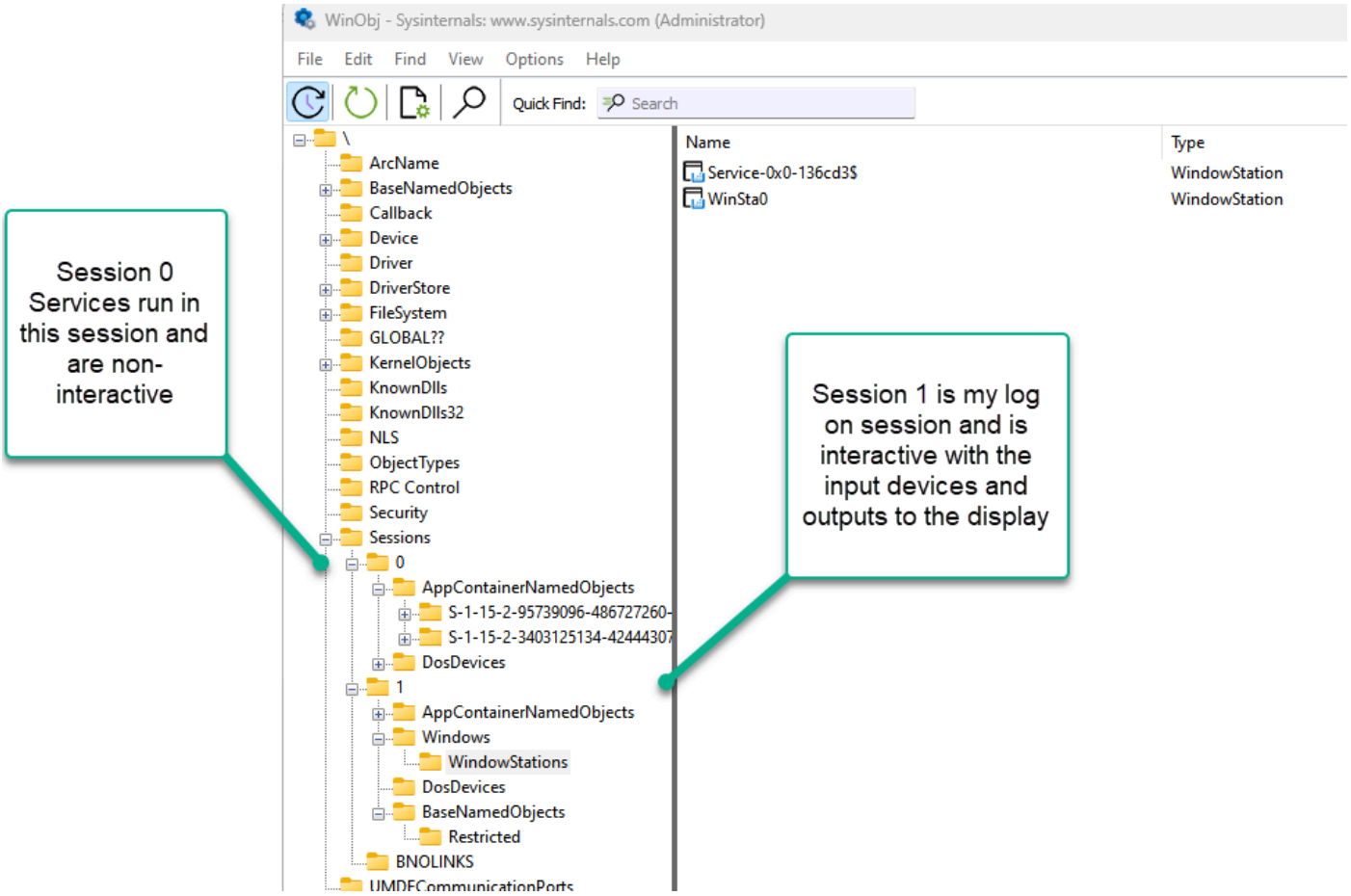
Windows defines a global namespace in the Object Manager and a session-private "local" namespace for each session numbered 1 and higher to provide isolation between sessions. The global namespace serves as the local namespace for processes in session 0. (WinObj offers a graphical view of the Object Manager namespace.)

System processes and Windows services always run in remote desktop services session 0. In Windows XP and Windows Server 2003, the first interactive user to log on to a computer also used session 0 and, consequently, used the same local namespace as services. Windows XP and Windows Server 2003 created sessions 1 and higher only when needed; if the first user logged off before a second one logged on, the second user used session 0 as well. Consequently, on a domain-joined Windows XP, session 0 was always the only session.

In Windows Vista and newer, services run in session 0, but for security reasons all interactive user sessions run in sessions 1 and higher. This increased separation between end-user processes and system processes is called *session 0 isolation*.

Note The term *console session* is sometimes mistaken as a synonym for *session 0*. The console session is the remote desktop services session associated with the locally attached keyboard, video, and mouse. If all active sessions on a computer are remote desktop sessions, the console session remains connected and displays a logon screen. It might or might not happen to be session 0 on Windows XP/Windows 2003, but it is never session 0 on Windows Vista or newer.

The utility called WinObj.exe allows us to see Sessions in a different way. See below.

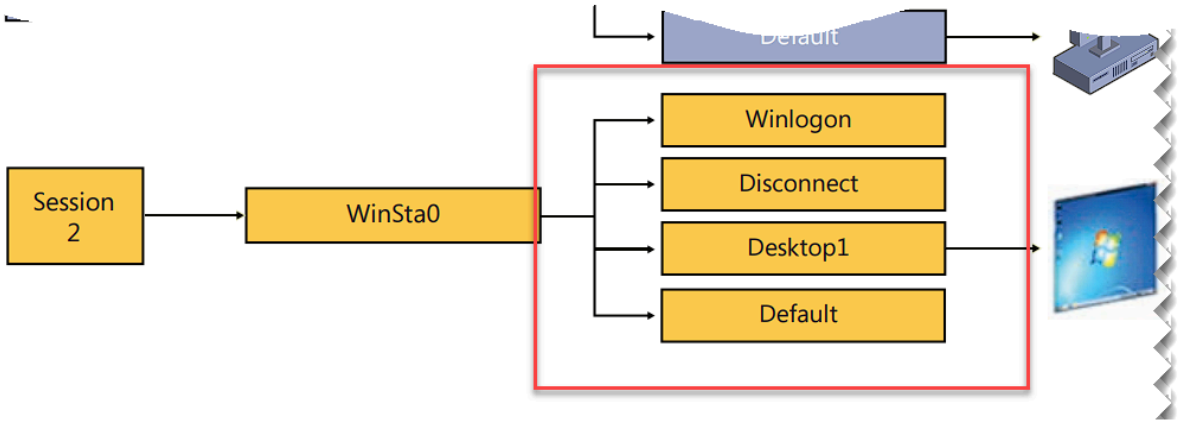


Window stations

Each remote desktop services session contains one or more named *window stations*. A window station is a securable object that contains a clipboard, an atom table, and one or more desktops. Every process is associated with one window station.

Within a session, only the window station named *WinSta0* can display a user interface or receive user input. In sessions 1 and higher, Windows creates only a WinSta0 window station. (See Figure below) In session 0, in addition to WinSta0, Windows creates a separate window station for every LSA logon session

associated with a service, with the locally unique identifier (LUID) of the logon session incorporated into the window station name.



For example, service
processes that run as System run in the *Service-0x0-3e7$* window station, while those that run as Network Service run in the *Service-0x0-3e4$* window station. These window stations cannot display a user interface or receive user input.

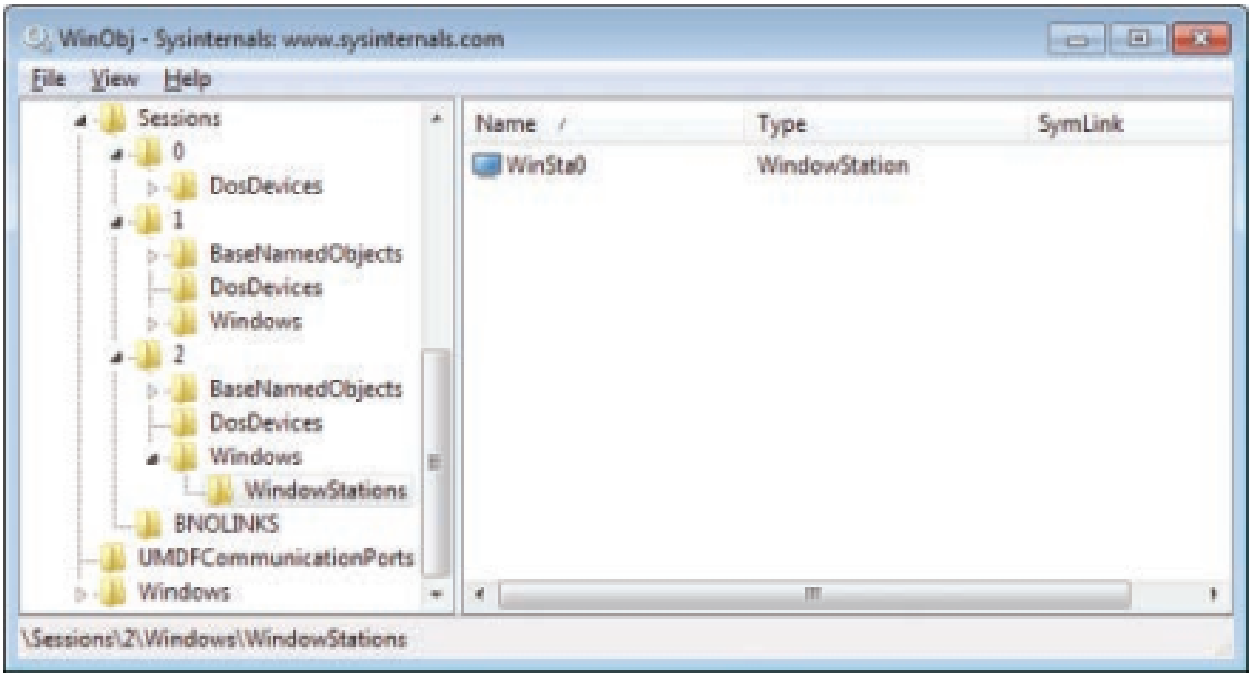


FIGURE WinObj showing the interactive window station in session 2's private namespace.

PsExec –s cmd.exe runs a command prompt in the *Service-0x0-3e7$* window station and redirects its console I/O to PsExec. PsExec's *–i* option lets you specify the remote desktop services session and runs the target process in its WinSta0 window station.

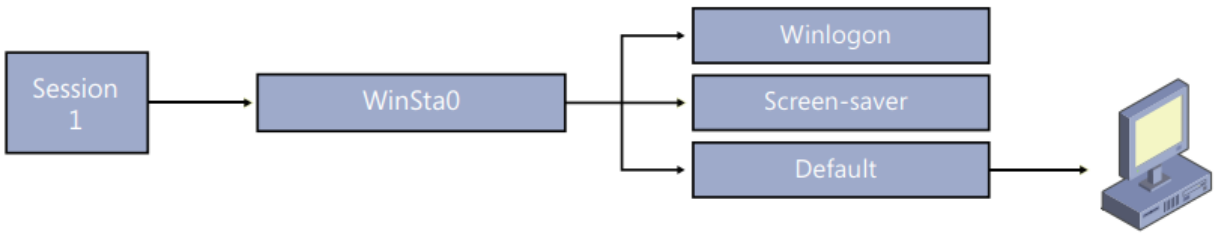
A service configured to run as System can also be configured to Allow Service To Interact With Desktop. When so configured, the service runs in session 0's *WinSta0* instead of *Service-0x0-3e7$*.

When the interactive user was also in session 0, this allowed the service to interact directly with the end user through the display and user input such as the mouse and keyboard. In hindsight, this wasn't a good idea as I'll describe shortly, and Microsoft has recommended against using this technique— and with session 0 isolation, this no longer works. (The Interactive Services Detection service, UI0Detect, offers partial mitigation.)

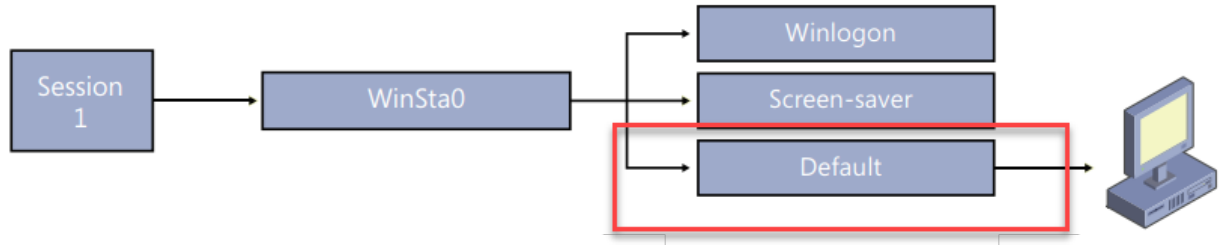
Desktops

Each window station contains one or more desktops. A desktop is a securable object with a logical display surface on which applications can render UI in the form of windows.

Multiple desktops can contain UI, but only one can be displayed at a time. There are typically three desktops in the interactive window station: Default, Screen-saver, and Winlogon.

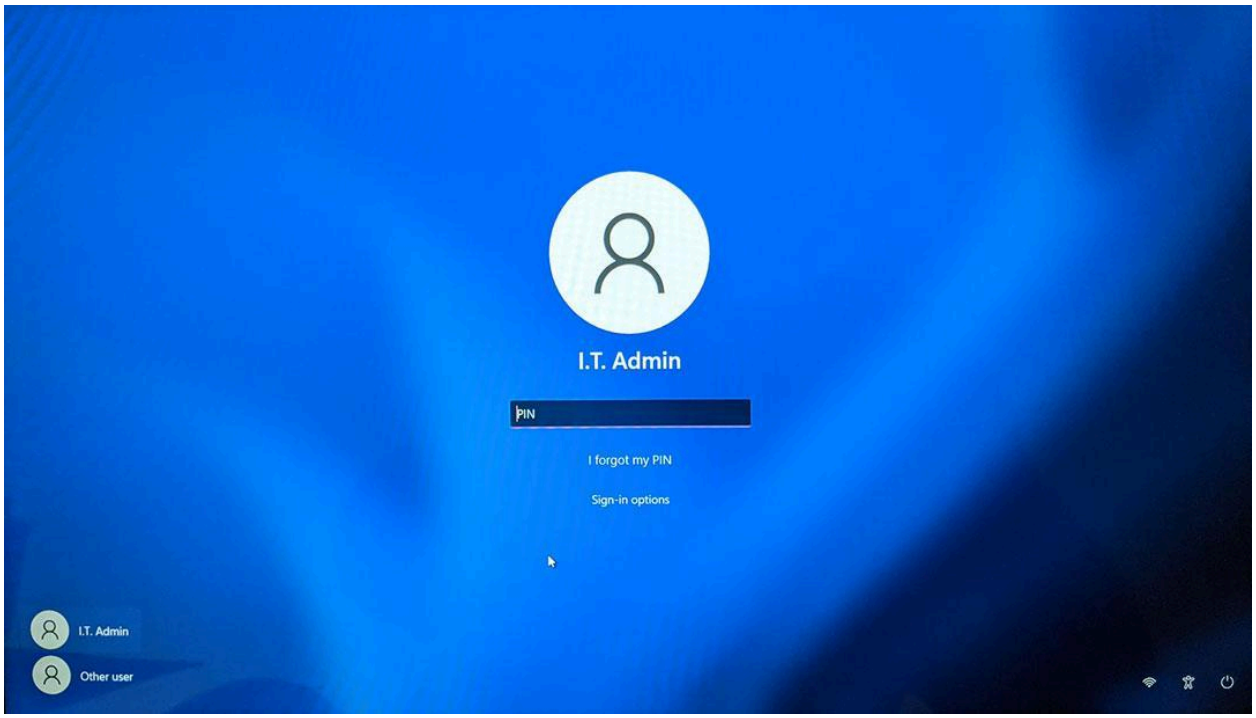


The Default desktop is where user applications run by default. (The Sysinternals Desktops utility creates up to three additional desktops on which to run applications.

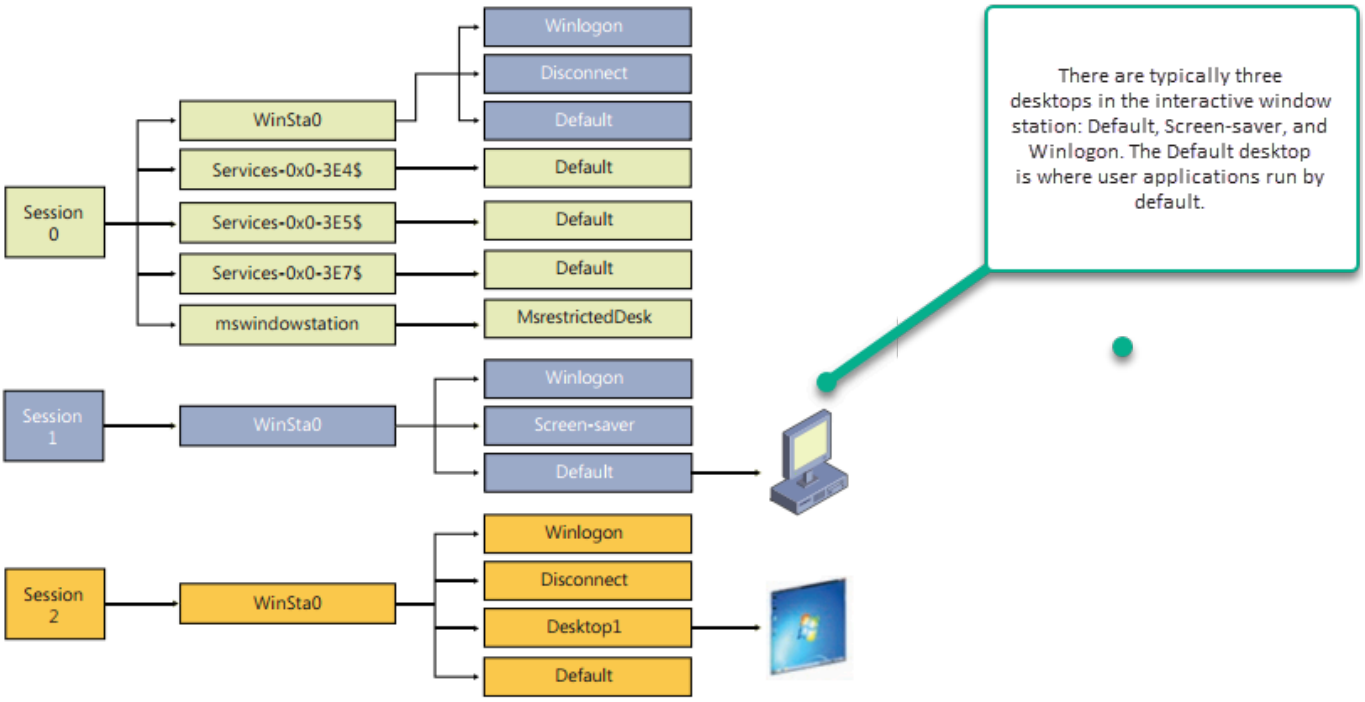


TheScreen-saver desktop is where Windows runs the screen saver if password protection is enabled.

The Winlogon desktop, also known as the *secure desktop*, is where Windows transfers control when you press Ctrl+Alt+Del and the default place to display UAC elevation dialog boxes.

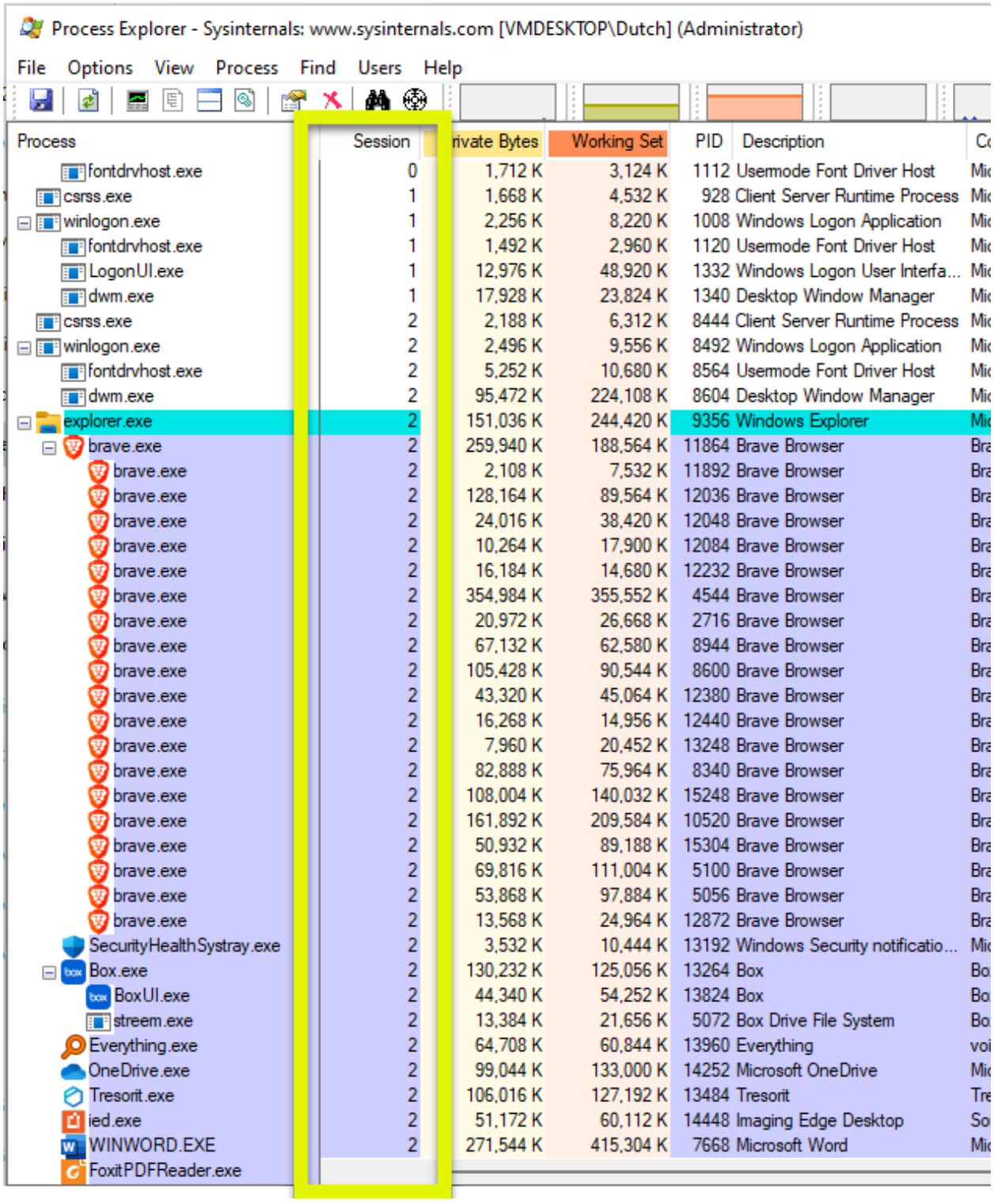


Permissions on the Winlogon desktop restrict access only to programs running as System, which protects secure operations involving password entry.



As a process is associated with a window station, each of its threads is associated with a desktop within the window station. Although individual threads of a process can be associated with different desktops, they are usually associated with a single desktop.

Several Sysinternals utilities, including Process Explorer (see example in fig below) and Process Monitor, identify the session ID to which a process belongs. Although none of the utilities directly identify the window station or desktops that a process is associated with, Process Explorer's Handle View can offer hints in the form of open handles to window stations or desktop objects.

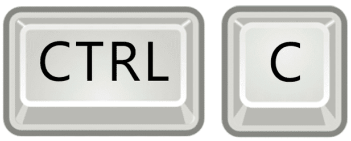


Learning how Psexec functions:

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.



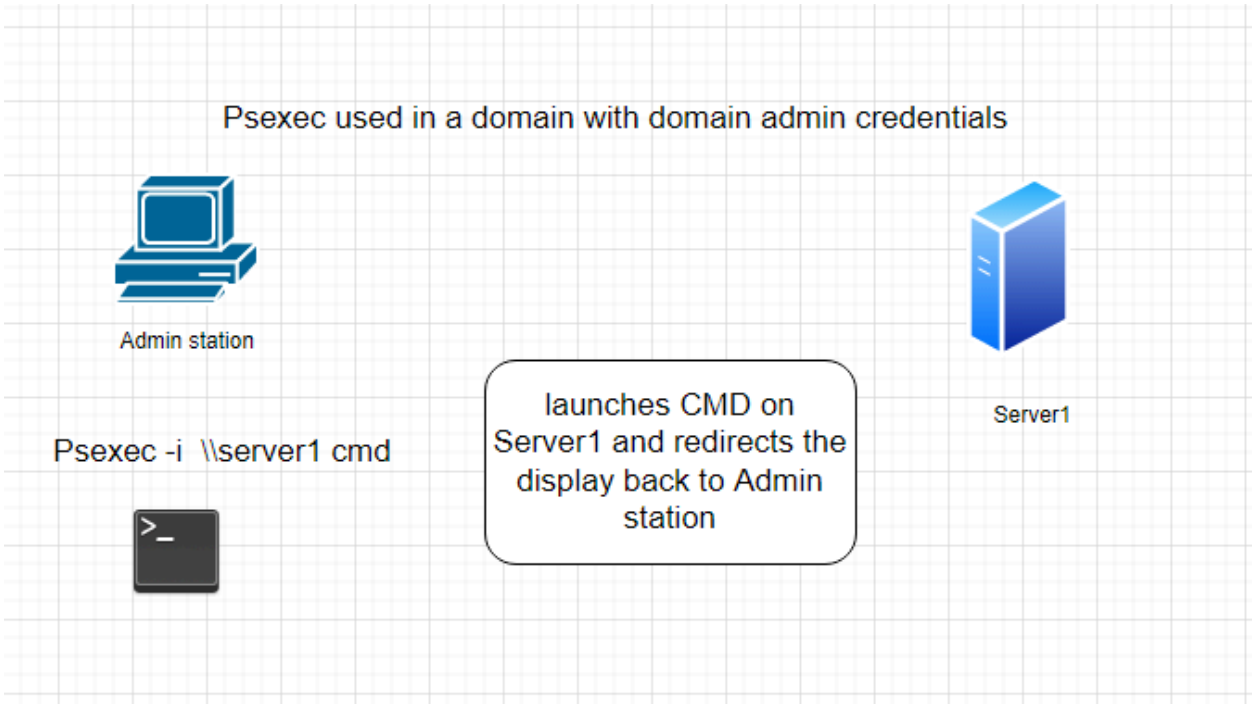
- If you omit a user name, the process will run in the context of your account on the remote system, but will not have access to network resources (because it is impersonating).
- Specify a valid username in the Domain\User syntax if the remote process requires access to network resources or to run in 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.

Examples

The following command launches an interactive command prompt on \\marklap:

psexec -i \\marklap cmd



This command executes IpConfig on the remote system with the /all switch, and displays the resulting output locally:

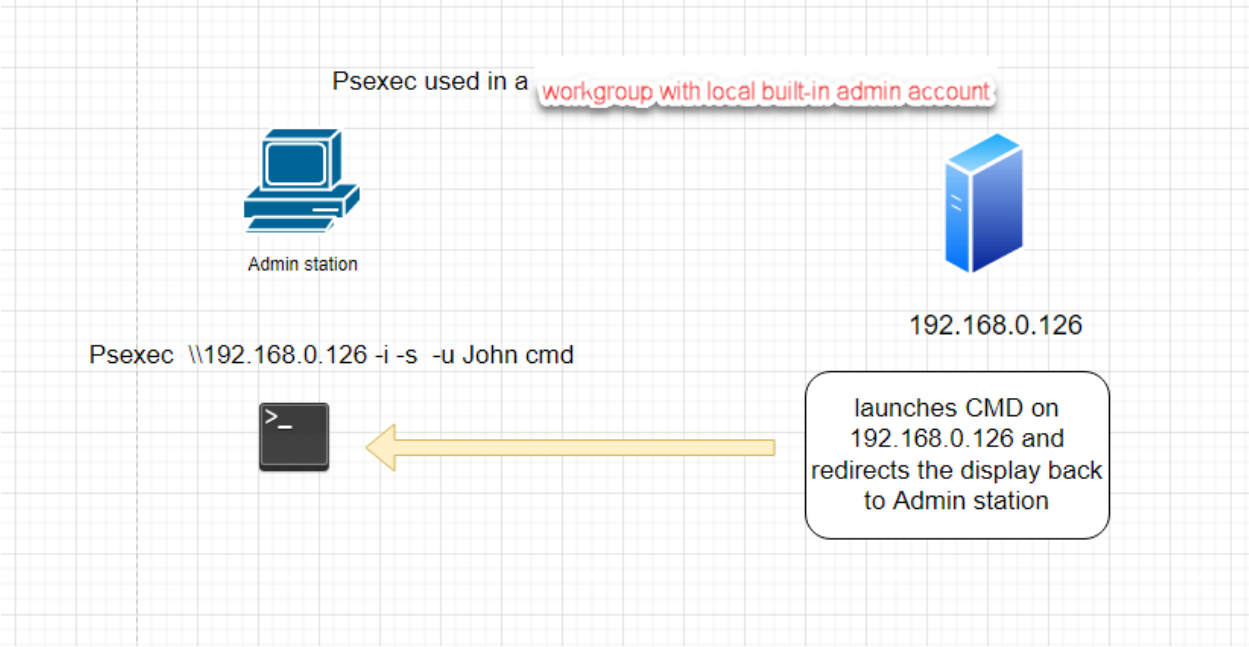
psexec -i \\marklap ipconfig /all

Here below we are running the cmd on 192.168.0.126 in a workgroup using the admin account called “John” we add -s for “SYSTEM account”, this is interactive and we can run a variety of CLI tools on the remote host

```
PS C:\Users\Dutch> psexec \\192.168.0.126 -s -u John cmd

PsExec v2.34 - Execute processes remotely
Copyright (C) 2001-2021 Mark Russinovich
Sysinternals - www.sysinternals.com

Password:

Microsoft Windows [Version 10.0.22000.651]
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>ipconfig /all

Windows IP Configuration

   Host Name . . . . . . . . . . . . : Windows11VM1
   Primary Dns Suffix  . . . . . . . :
   Node Type . . . . . . . . . . . . : Hybrid
   IP Routing Enabled. . . . . . . . : No
   WINS Proxy Enabled. . . . . . . . : No

Ethernet adapter Ethernet 3:

   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : Microsoft Hyper-V Network Adapter #3
   Physical Address. . . . . . . . . : 00-15-5D-00-BE-06
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes
   Link-local IPv6 Address . . . . . : fe80::dd6e:f7e4:9bbf:bb72%8(Preferred)
   IPv4 Address. . . . . . . . . . . : 192.168.0.126(Preferred)
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Lease Obtained. . . . . . . . . . : Saturday, April 16, 2022 9:50:26 AM
   Lease Expires . . . . . . . . . . : Saturday, April 16, 2022 11:50:26 AM
   Default Gateway . . . . . . . . . : 192.168.0.1
   DHCP Server . . . . . . . . . . . : 192.168.0.1
   DHCPv6 IAID . . . . . . . . . . . : 369104221
   DHCPv6 Client DUID. . . . . . . . : 00-01-00-01-27-89-6F-C6-00-15-5D-00-BE-00
   DNS Servers . . . . . . . . . . . : 1.1.1.1
                                       192.168.0.1
   NetBIOS over Tcpip. . . . . . . . : Enabled
```

Psexec now shows a cmd prompt from 192.168.0.126 commands are run on remote host but displayed on local PC

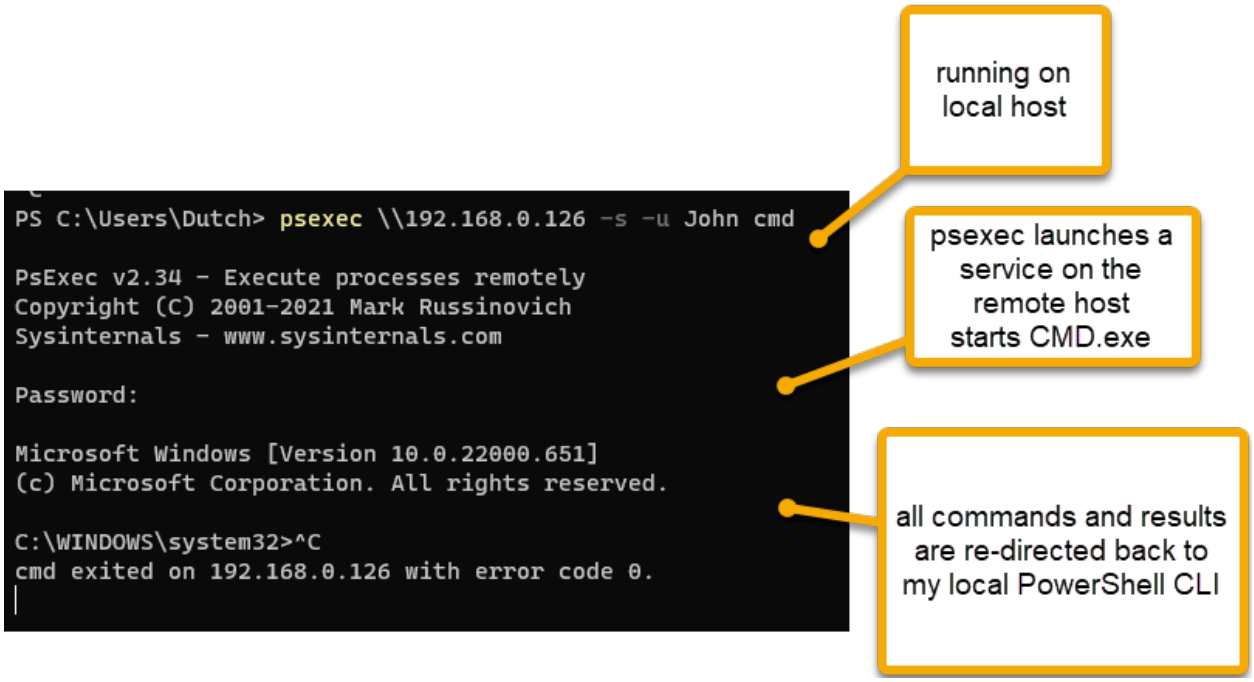
Because my CLI interface is interactive with the remote PC and can use any CMD command and see the results on my admin station



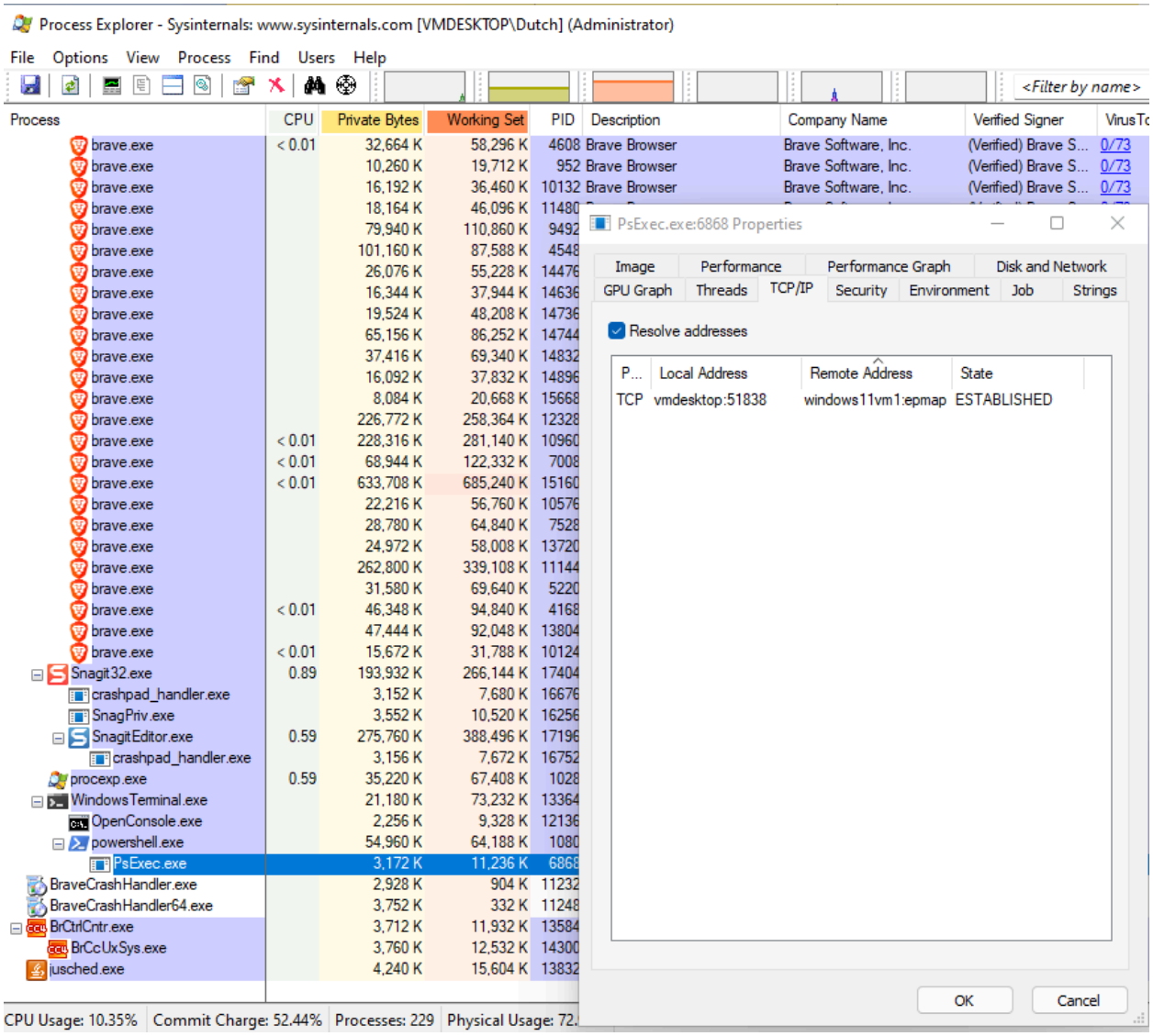
Typing Ctrl-C terminates the remote process.

```
PS C:\Users\Dutch> psexec \\192.168.0.126 -s -u John cmd

PsExec v2.34 - Execute processes remotely
Copyright (C) 2001-2021 Mark Russinovich
Sysinternals - www.sysinternals.com

Password:

Microsoft Windows [Version 10.0.22000.651]
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>^C
cmd exited on 192.168.0.126 with error code 0.
```

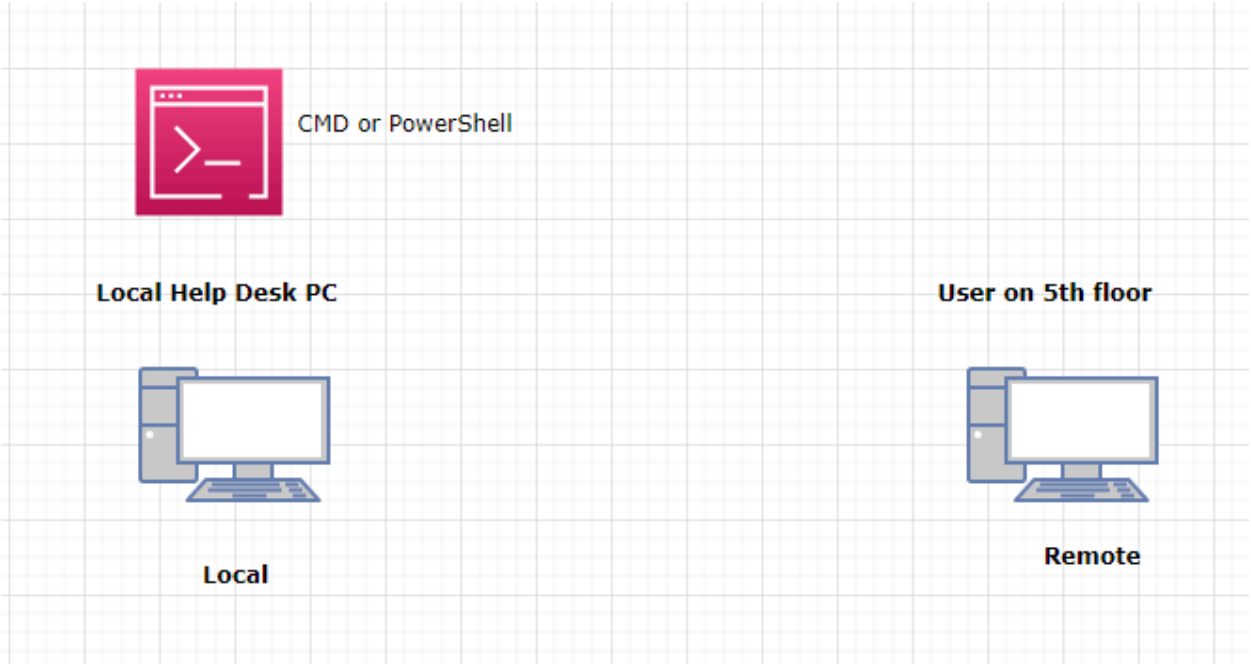
Typing Ctrl-C terminates the remote process.



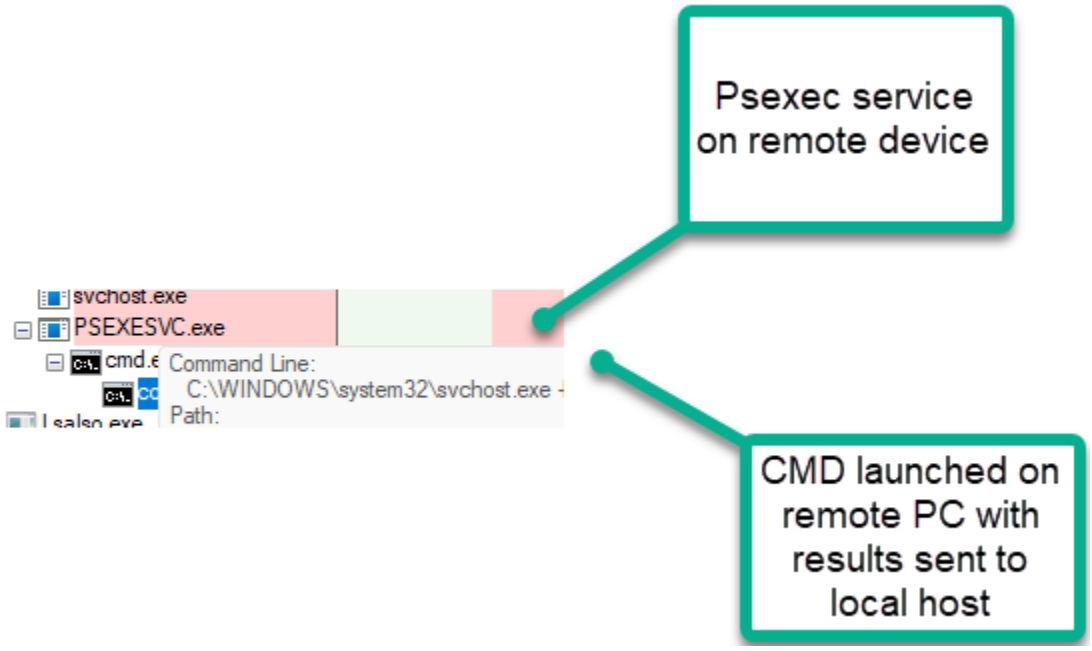
Here in the graphic we see Process Explorer showing Psexec running on the host PC, looking at the TCP properties we see it connecting to the remote host



Above we see psexec.exe process connected to remote host 192.168.0.126



Below shows the service PSEXESVC.exe on the remote host.

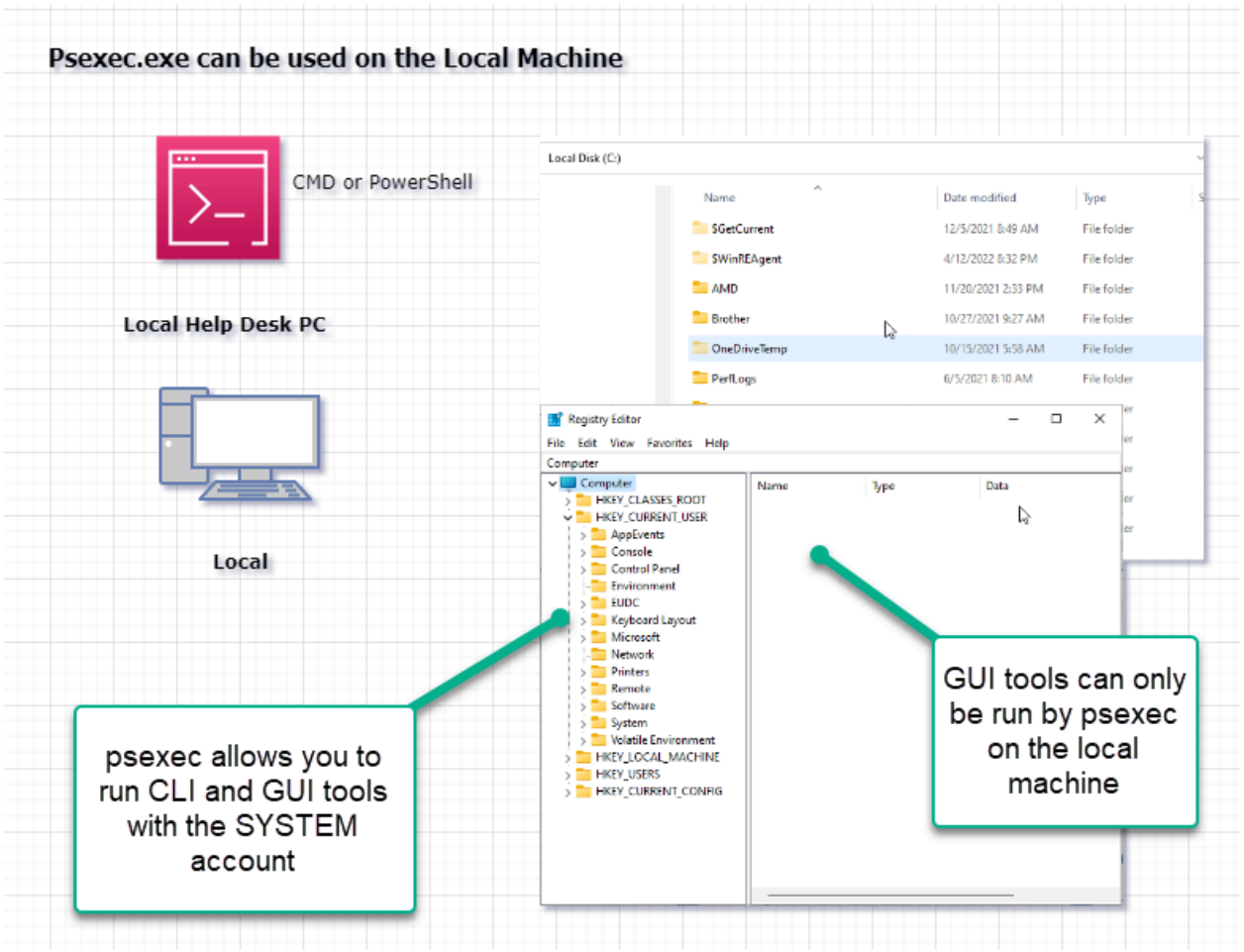


This command copies the program test.exe to the remote system and executes it interactively:

psexec -i \\marklap -c test.exe

Specify the full path to a program that is already installed on a remote system if its not on the system's path:

psexec -i \\marklap c:\bin\test.exe



Psexec allows you to use the SYSTEM Account

- is one of the most powerful accounts in Windows, the SYSTEM account is like ROOT for Linux.
- Psexec can run command line, PowerShell and GUI utilities with SYSTEM account privileges on the local machine. This is the only time you can execute GUI programs with Psexec is locally.

The SYSTEM account is used by the operating system and by services that run under Windows. There are many services and processes in the Windows operating system that need the capability to sign in internally, such as during a Windows installation. The SYSTEM account was designed for that purpose, and Windows manages the SYSTEM account's user rights. It is an internal account that does not show up in User Manager, and it cannot be added to any groups.

On the other hand, the SYSTEM account does appear on an NTFS file system volume in File Manager in the **Permissions** portion of the **Security** menu. By default, the SYSTEM account is granted Full Control permissions to all files on an NTFS volume. Here the SYSTEM account has the same functional rights and permissions as the Administrator account.

Local Use only: Run Regedit interactively in the System account to view the contents of the SAM and SECURITY keys::

psexec -i -d -s c:\windows\regedit.exe

To run Internet Explorer as with limited-user privileges use this command:

psexec -l -d "c:\program files\internet explorer\iexplore.exe"

Setup Psexec on local Admin station

- *Environment variables configured*
- *Firewall opened*
- *Know what user/admin account to use*
- *File and Print Sharing enbled*

A number of dials and knobs need to be set just right for PsTools to work on remote systems. Obviously, they all require connectivity to the necessary network interfaces, which involves firewall settings and ensuring that services are running. Most of the utilities require administrative rights. And finally, User Account Control (UAC) applies restrictions to local accounts that must be taken into consideration.

Basic connectivity

1. File and printer sharing must be enabled on both local and remote computers (TCP port 445 must be open on remote computers).

a. See difference steps in articles found included in these notes
b. You can open your firewall
c. PowerShell command let's you see if these ports are open:
   `Get-TCPConnection`

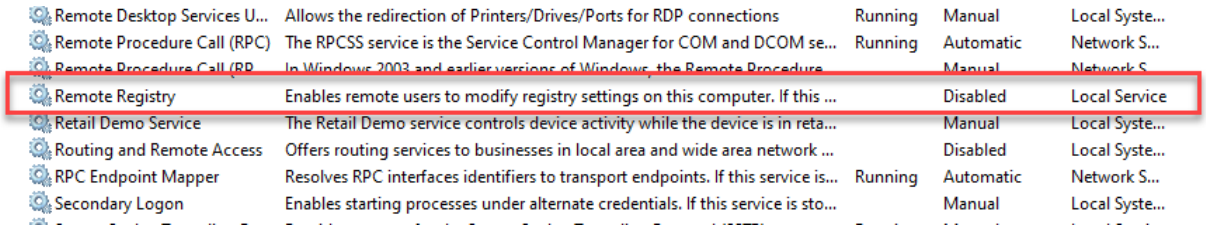
Unless you specify an IP address, name resolution needs to work. If DNS is not available, NetBIOS over TCP (NBT) might suffice, but it requires that 137 UDP, 137 TCP, 138 UDP, and 139 TCP be opened on the firewall of the target system.

2. Some of the utilities require that the administrative Admin$ share be available. This requires that file and print sharing be enabled (the Workstation service locally and the Server service on the target system), that the firewall not block the ports that are needed to support file and printer sharing, and also that "simple file sharing" be disabled.

```
PS C:\Users\Dutch> Get-SMBshare

Name   ScopeName Path        Description
----   --------- ----        -----------
ADMIN$ *         C:\WINDOWS  Remote Admin
C$     *         C:\         Default share
IPC$   *                     Remote IPC
```

3. Some of the utilities require that the Remote Registry service be running on the target system.

Note that in the newer versions of Windows, this service is not configured for automatic start by default. It therefore needs to be manually started or configured for automatic start before some of these tools will work.



4. User accounts

If you are using a domain and have domain admin credentials you can very effectively use the tools.

If you are in a workgroup (non-domain) environment, be sure to use the local built-in administrator account on the local machines. By default, Microsoft disables this account after setup so you will have to "enable" and create a new complex password for this account.

5. If you do not install your Sysinternal tools via the Microsoft Store you will need to set either "system" or "user" environment variables.
    a. See many different way of doing this step in articles in video notes below.

PsExec Explainer by Mark Russinovich

Find out how PsExec works and how it allows you to execute processes on a remote system and redirect output to the local system.

Mark Russinovich

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 12 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.

Related: PsExec to the Rescue

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.

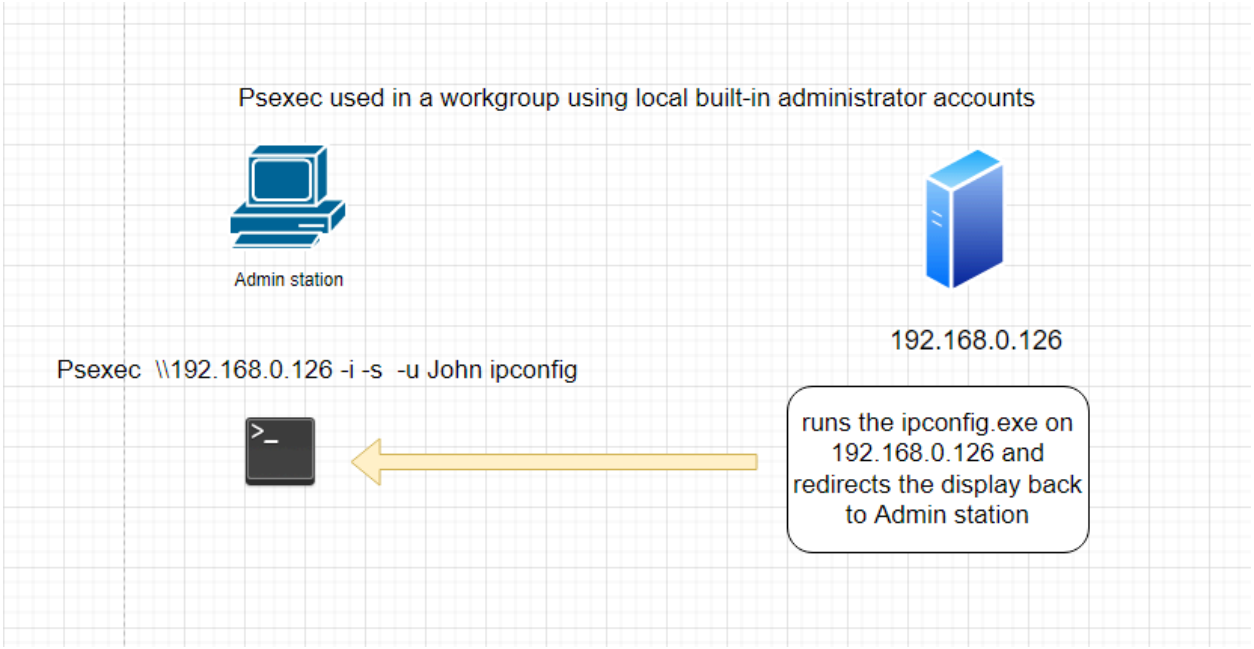
- PsExec lets you remote-enable any of them. For example, PsExec lets Ipconfig, the Windows utility that displays the TCP/IP configuration for a system's network adapters, show a remote system's configuration. A sample command for that use is

psexec \\remotePC 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.

```
PS C:\Users\Dutch> psexec \\192.168.0.126 -s -u John ipconfig

PsExec v2.34 - Execute processes remotely
Copyright (C) 2001-2021 Mark Russinovich
Sysinternals - www.sysinternals.com

Password:


Windows IP Configuration


Ethernet adapter Ethernet 3:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::dd6e:f7e4:9bbf:bb72%8
   IPv4 Address. . . . . . . . . . . : 192.168.0.126
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.0.1
ipconfig exited on 192.168.0.126 with error code 0.
```

Non domain computers you have to add -u -s John is a administrator on this PC but I had to add the -s for local system rights



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.

```
Password:
Starting PSEXESVC service on 192.168.0.126...
```

PSexec begins a service on the remote host called "psexesvc"

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.

```
PS C:\Users\Dutch> psexec \\192.168.0.126 -s -u John notepad.exe

PsExec v2.34 - Execute processes remotely
Copyright (C) 2001-2021 Mark Russinovich
Sysinternals - www.sysinternals.com

Password:

PsExec could not start notepad.exe on 192.168.0.126:
The system cannot find the file specified.
```

Look at the failure above, notepad.exe is in the C:\Windows directory not in the C:\Windows\system32 folder

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.

The Psexesvc service creates a named pipe, psexecsvc, to which PsExec connects and sends commands that tell the service on the remote system which executable to launch and which options you've specified.

If you specify the -d (don't wait) switch, the service exits after starting the executable; otherwise, the service waits for the executable to terminate, then sends the exit code back to PsExec for it to print on the local console.

Configure an Enterprise Switch via a serial console port using Putty

PsExec: What It Is and How to Use It

Execute commands remotely from a Windows PC

By
Tim Fisher

https://www.lifewire.com/psexec-4587631

- Step-by-Step: Set Up PsExec
- Step-by-Step: Use PsExec
- PsExec Command Examples
- Extra: PsExec Can Be Dangerous

PsExec is a portable tool from Microsoft that lets you run processes remotely using any user's credentials. It's a bit like a remote access program but instead of controlling the computer with a mouse, commands are sent via Command Prompt.

You can use PsExec to not only manage processes on the remote computer but also redirect an application's console output to your local computer, making it appear as though the process is running locally.

No software is needed on the remote computer to make PsExec work, but there are a few things to keep in mind if the tool doesn't run correctly the first time you try it.

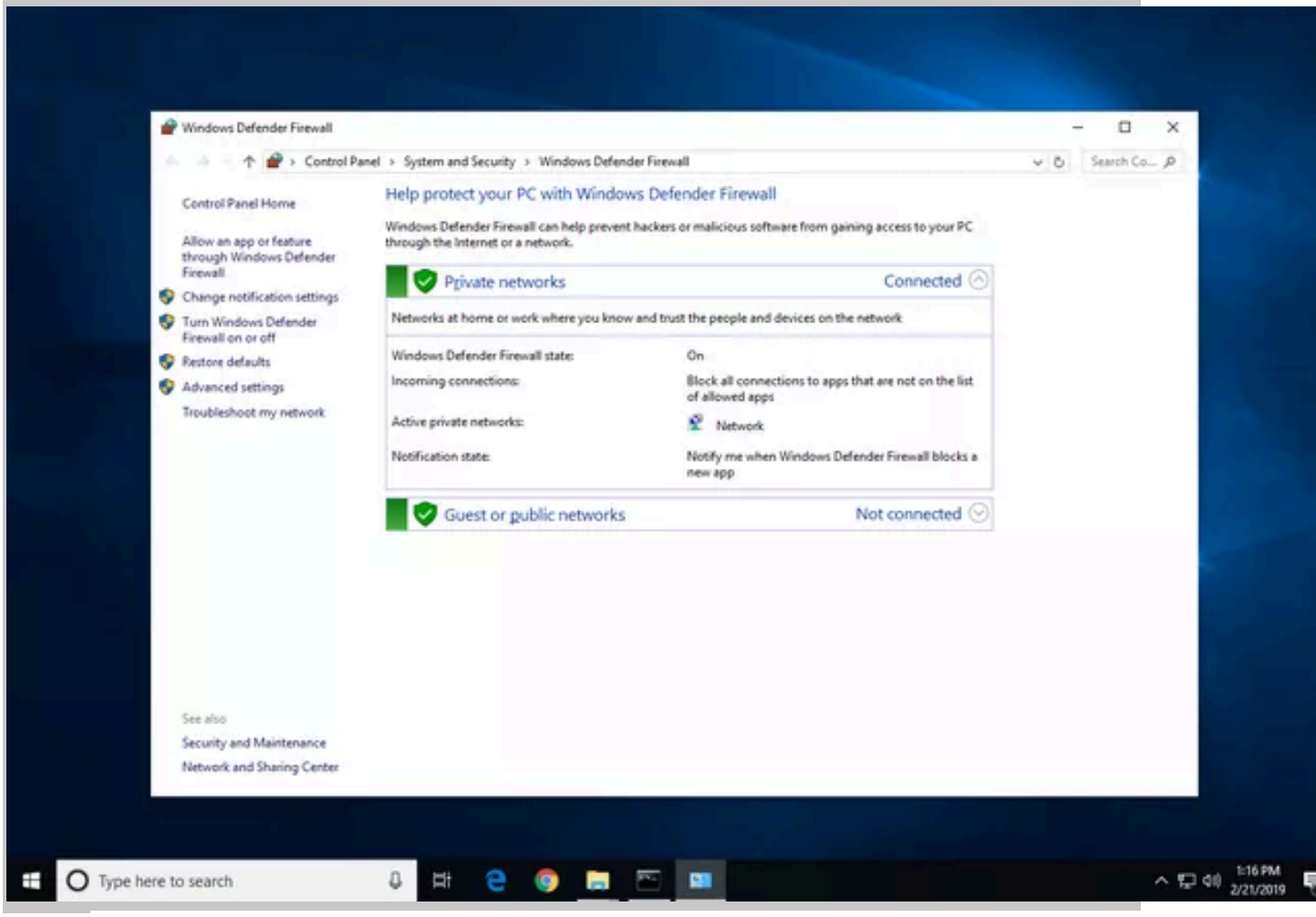
How to Set Up PsExec

If PsExec is portable and doesn't need to be copied to the remote computer, what type of setup does it really need?

The tool works only under certain conditions. Namely, when file and printer sharing is enabled on both the local and remote computer, and when the remote machine has the *$admin* share set up correctly to provide access to its *Windows*\ folder.

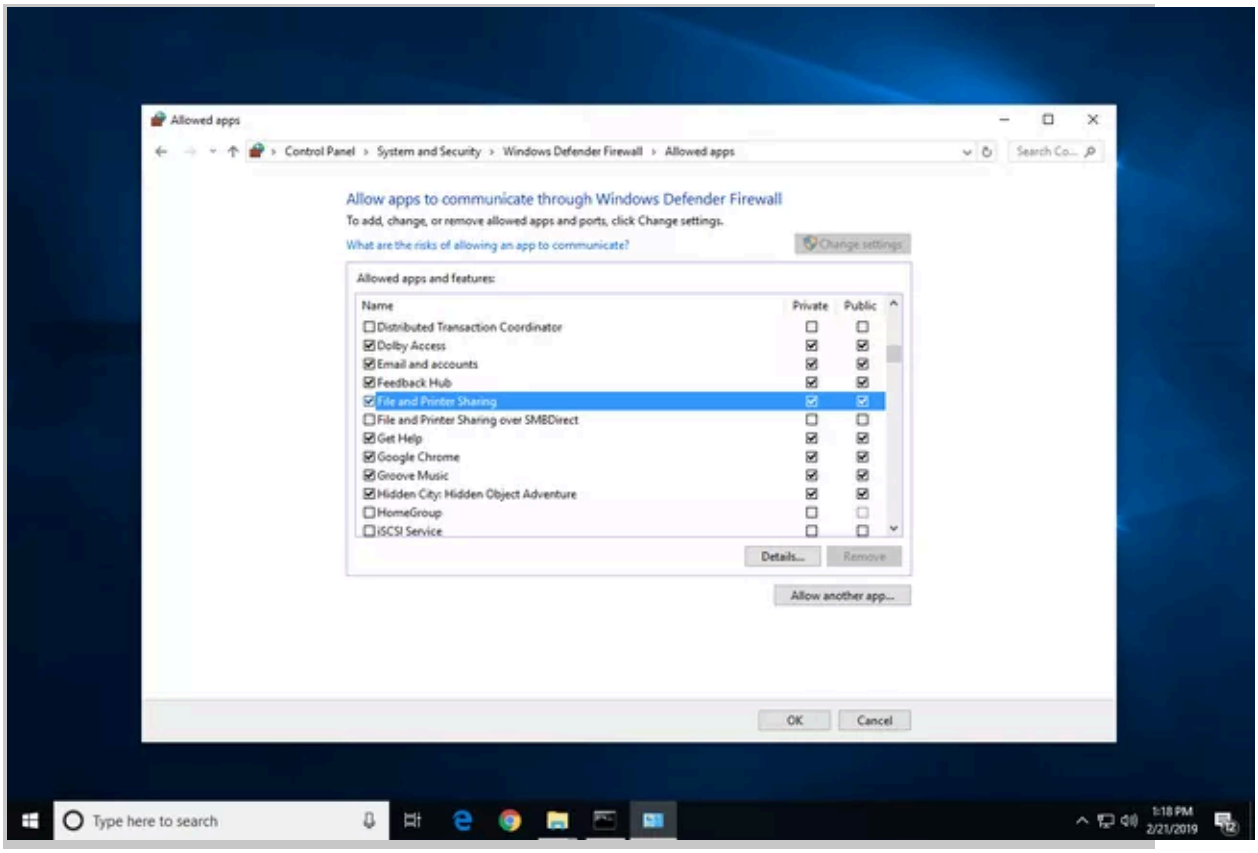
You can double-check that file and print sharing is enabled by looking in the Windows Firewall settings:

1. Enter **firewall.cpl** in the Run dialog box. One way to open Run is through the **WIN+R** keyboard shortcut.
2. Select **Allow an app or feature through Windows Firewall** from the left side of the window.



This might read as **Allow an app or feature through Windows Defender Firewall** depending on how your computer is set up, but it's the same option.

3. Make sure **File and Printer Sharing** has a checkmark in the **Private** box to its right. If it doesn't, put a check in that box and select **OK**.



If you can't change the firewall settings because they're greyed out, select **Change settings** at the top of the window.

4. You can now exit any open Windows Firewall settings.

With the Windows Firewall now set up correctly for PsExec, you should have no problem accessing the *$admin* share on the remote machine so long as the following are true:

- Both computers belong to the same Workgroup
- You know the password to an administrator's account on the remote computer

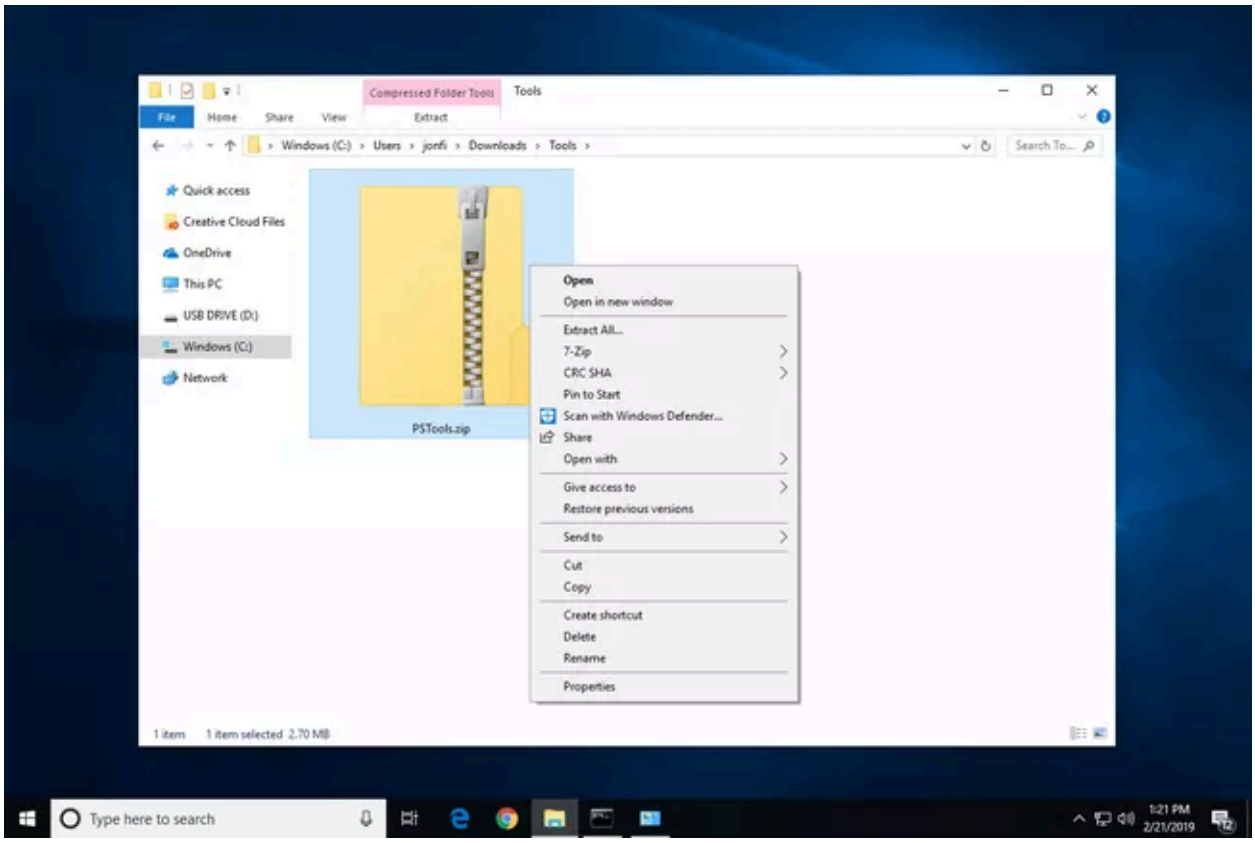
See this tutorial at Wintips.org if you need help doing those things or if you've done them correctly but later on, after trying to use PsExec as described below, you get an "access denied" error.

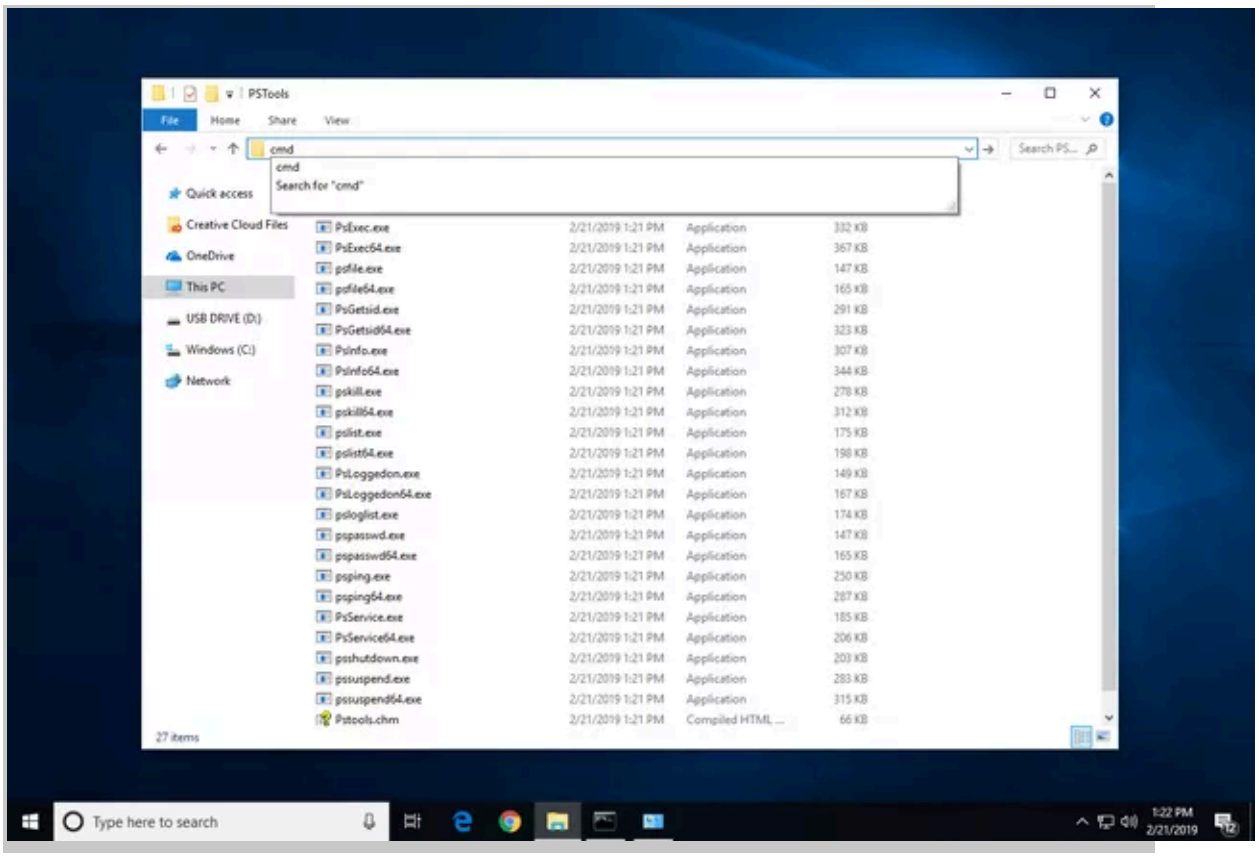
How to Use PsExec

Before using PsExec to execute remote commands, you have to download the program and position Command Prompt in a way where you can utilize the tool correctly.

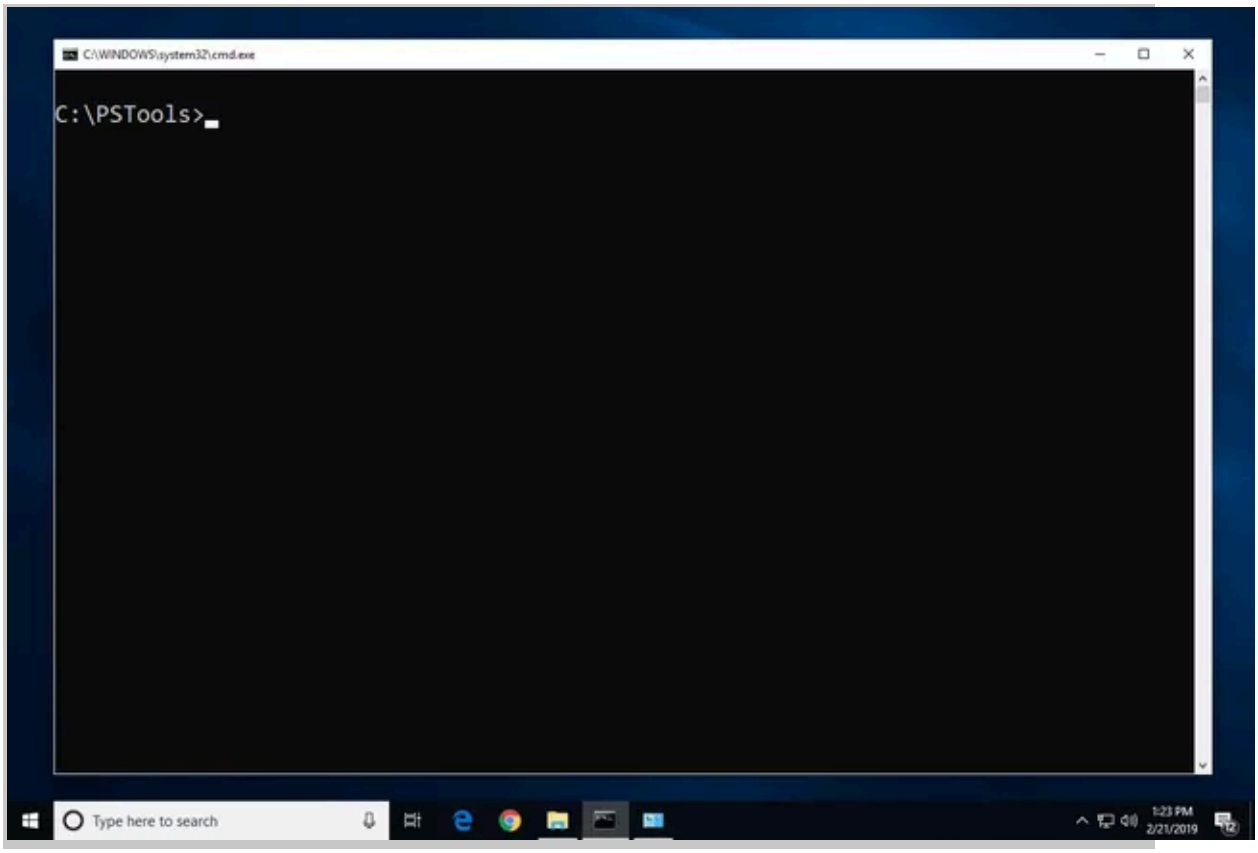
Download and Open It

1. Download PsExec on the computer that will be running the remote commands. It's available for free from Microsoft at Sysinternals as part of PsTools.
2. Extract the files from the *PsTools.zip* download. You can do that by right-clicking the ZIP file and selecting **Extract All**. Any third-party file extractor will work, too.





3. Open the folder where the extracted files are located, and from the navigation bar at the top of the folder, erase what's there and enter **cmd**.



Another way to do this, at least in some versions of Windows, is to **Shift+Right Click** an empty space in the PsTools folder and choose **Open command window here**.

This will open Command Prompt in that folder so that you can run commands through PsExec.

4. Optional: You can add your new directory to the path of Windows environment variables which allows you to execute your new tools in any command prompt. See instructions on how to do this: https://www.computerhope.com/issues/ch000549.htm
5. With Command Prompt now open to the folder that contains PsExec.exe, you can start entering commands on the remote machine.

Understanding the Syntax

Much like any command-line tool, PsExec works only when its syntax is followed exactly. Once you understand how to type commands in the way the tool understands them, you can control the program from any Command Prompt.

This is how PsExec commands must be entered:

psexec [*****computer*[**,***computer2*[,...] | **@file**]][**-u** *username* [**-p** *password*][**-n** **s**][**-r** *servicename*][**-h**][**-l**][**-s**|**-e**][**-x**][**-i** [*session*]][**-c** *executable* [**-f**|**-v**]][**-w** *directory*][**-d**][-<*priority*>][**-a n,n**,...] *cmd* [*arguments*]

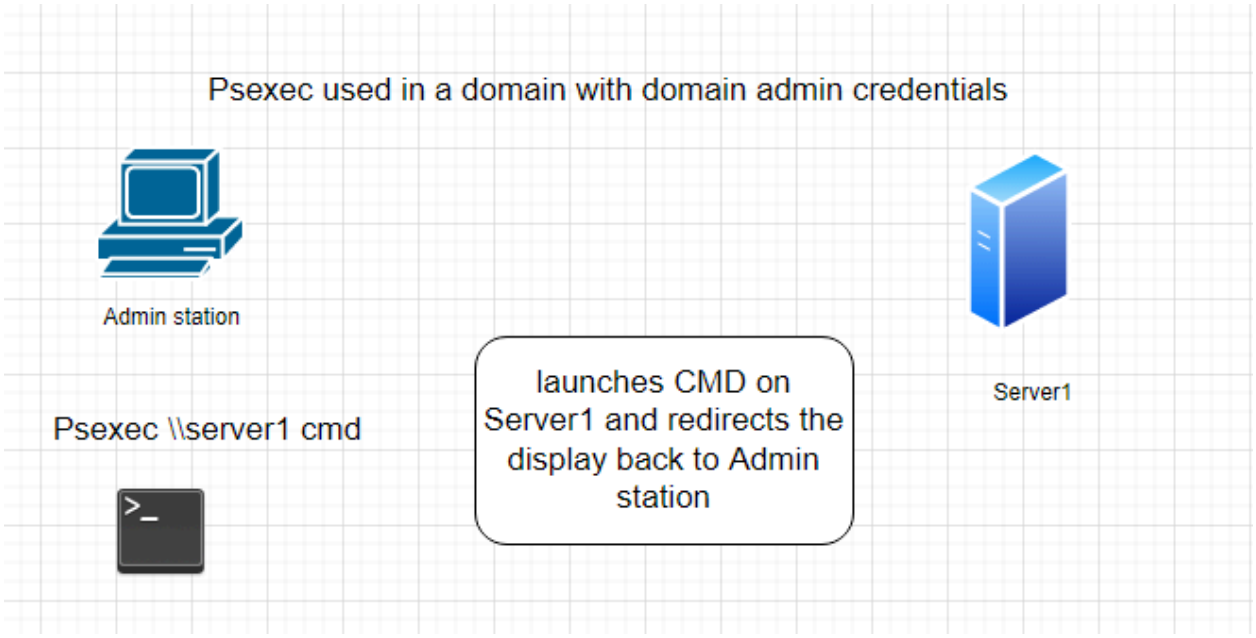
This may look complex and confusing but don't worry! There are some examples at the bottom of this page that you can use to practice.

PsExec Command Examples

Here are a few examples of how to use PsExec to do things like run remote Command Prompt commands, manage Windows Services, and launch or install programs.

Open and using a CMD Remotely

```
psexec \\server1 cmd
```



One of the easiest ways to use PsExec to run Command Prompt commands on a remote computer is to execute *cmd* following the machine's computer name Server1 in this example.

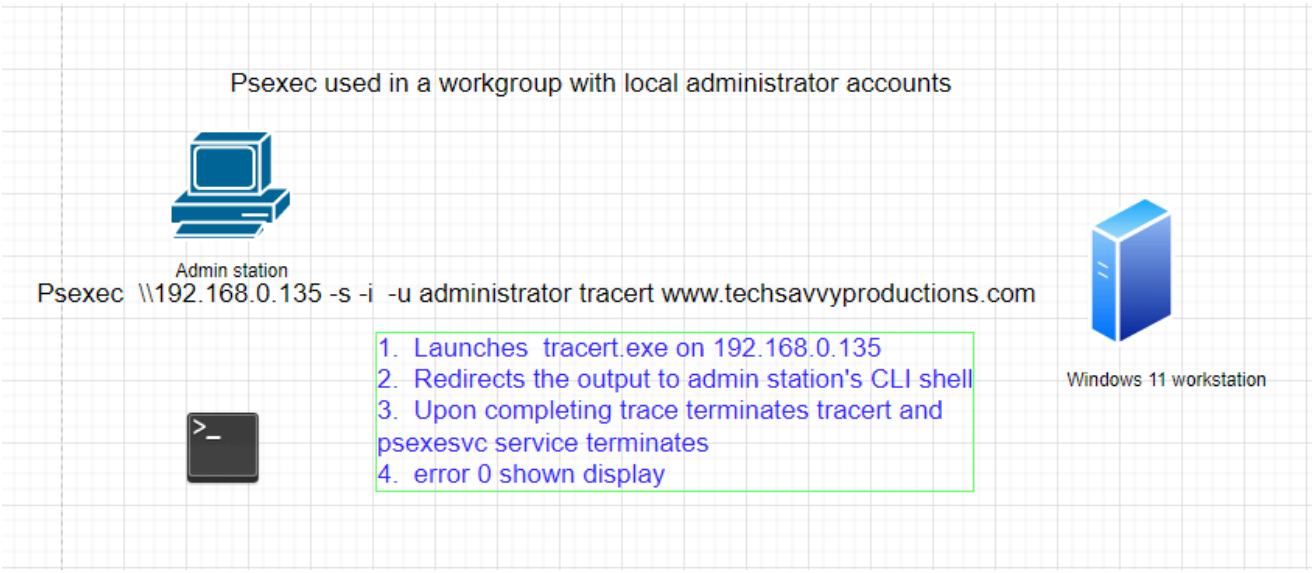
Same action but in a workgroup environment (non-domain). More complex options may need to be used -s (system account) -I (interactive mode) -u (domain/username)

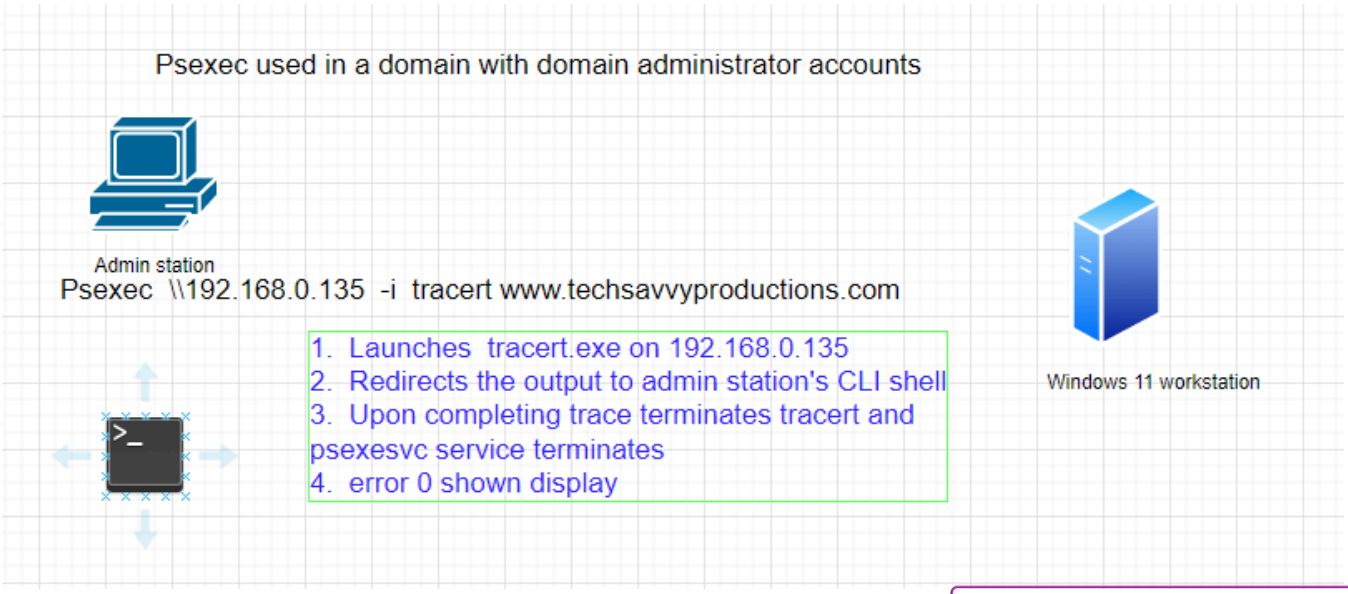
Doing this will launch a regular Command Prompt window within the existing one, and let you enter every command as if you were sitting in front of the remote computer. For example, you could then enter ipconfig to get those results from the other computer, or **mkdir** to create a new folder, dir to list the folder's contents, etc.

Run a Remote Command that exists on the remote host

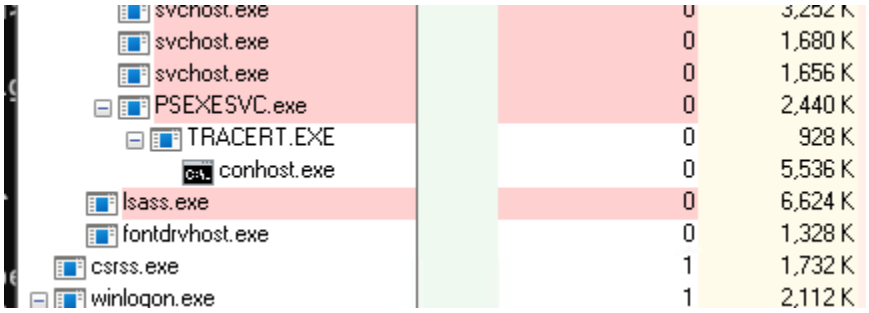
```
psexec \\mediaserver01 tracert lifewire.com
```

Another way to use PsExec is to enter individual commands but without starting a full Command Prompt. In this example, we're executing the tracert command against *lifewire.com*, and because we've specified the remote computer's name, *mediaserver01*, the command results are relevant to *that* machine, not the local one (i.e., the one you're on).





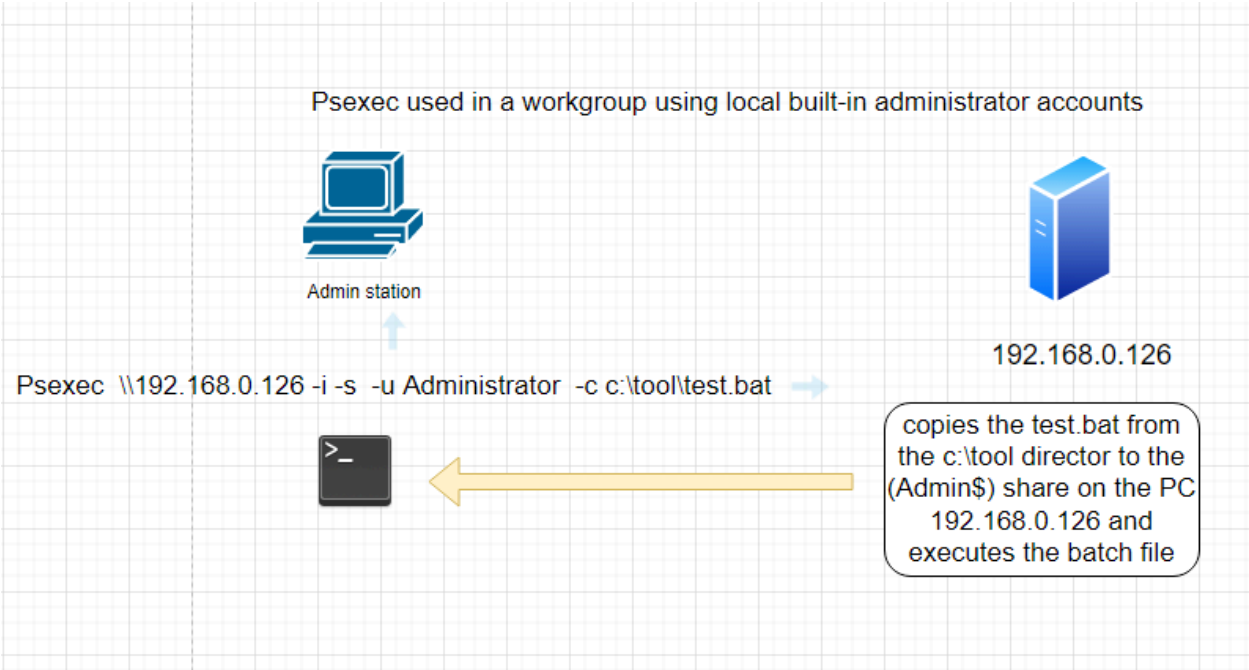
This is the tracert.exe running on the remote host



Run a Remote Command that does not exists on the remote host

- Copy a program from our PC to the remote PC
  - Psexec puts files into the C:\Windows directory (Admin$ share) by default
- Execute the program
- Redirect the output if needed back to our PC
- Terminate the program and delete the file
- Terminate the psexec service

Psexec \\192.168.0.135 -s -i -u Administrator -c c:\tool\test.bat



Some programs many not terminate gracefully you might add the -d switch so the "psexesvc service" will terminate and then run the program. Remember Ctlr+C terminates the "psecesvc" service

Here I copied Nirsoft's cmd utility from my PC c:\tool\ called "whosip.exe" to the remote host and asked for info on www.ebay.com

NirSoft

https://www.nirsoft.net/ Another set of my favorite "portable app" utilities for the IT professionals.

Tech Savvy Productions

```
C:\Users\John>psexec \\192.168.0.135 -s -i -u Administrator -c c:\tool\whosip.exe www.ebay.com

PsExec v2.34 - Execute processes remotely
Copyright (C) 2001-2021 Mark Russinovich
Sysinternals - www.sysinternals.com

Password:


WHOIS Source: ARIN
IP Address:   184.84.137.37
Country:      USA - Massachusetts
Network Name: AKAMAI
Owner Name:   Akamai Technologies, Inc.
CIDR:         184.84.0.0/14
From IP:      184.84.0.0
To IP:        184.87.255.255
Allocated:    Yes
Contact Name: Akamai Technologies, Inc.
Address:      145 Broadway, Cambridge
Email:        ip-admin@akamai.com
Abuse Email:  abuse@akamai.com
Phone:        +1-617-444-2535
Fax:
whosip.exe exited on 192.168.0.135 with error code 0.
```

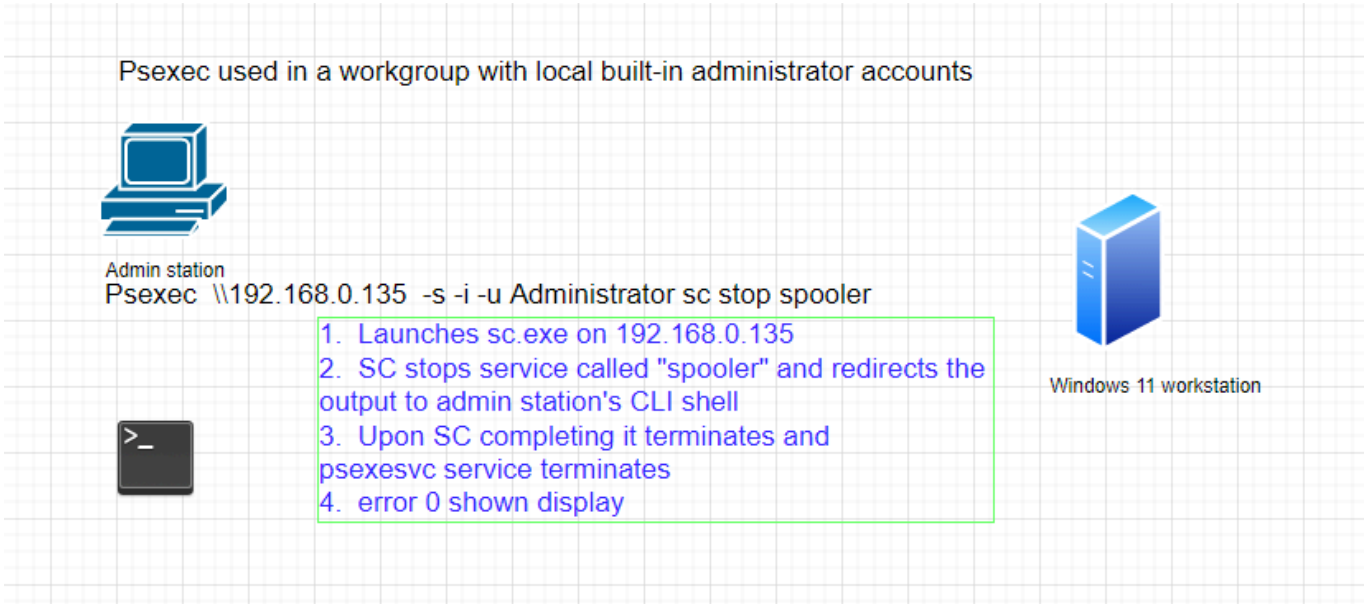
Stop/Start a Service Remotely

```
psexec \\FRONTDESK_PC -u tomd -p 3*(tom#87 net start spooler
```

The PsExec command example shown above starts the Print Spooler service, *spooler*, remotely on the *FRONTDESK_PC* computer using the *tomd* user's password, *3*(tom#87*.

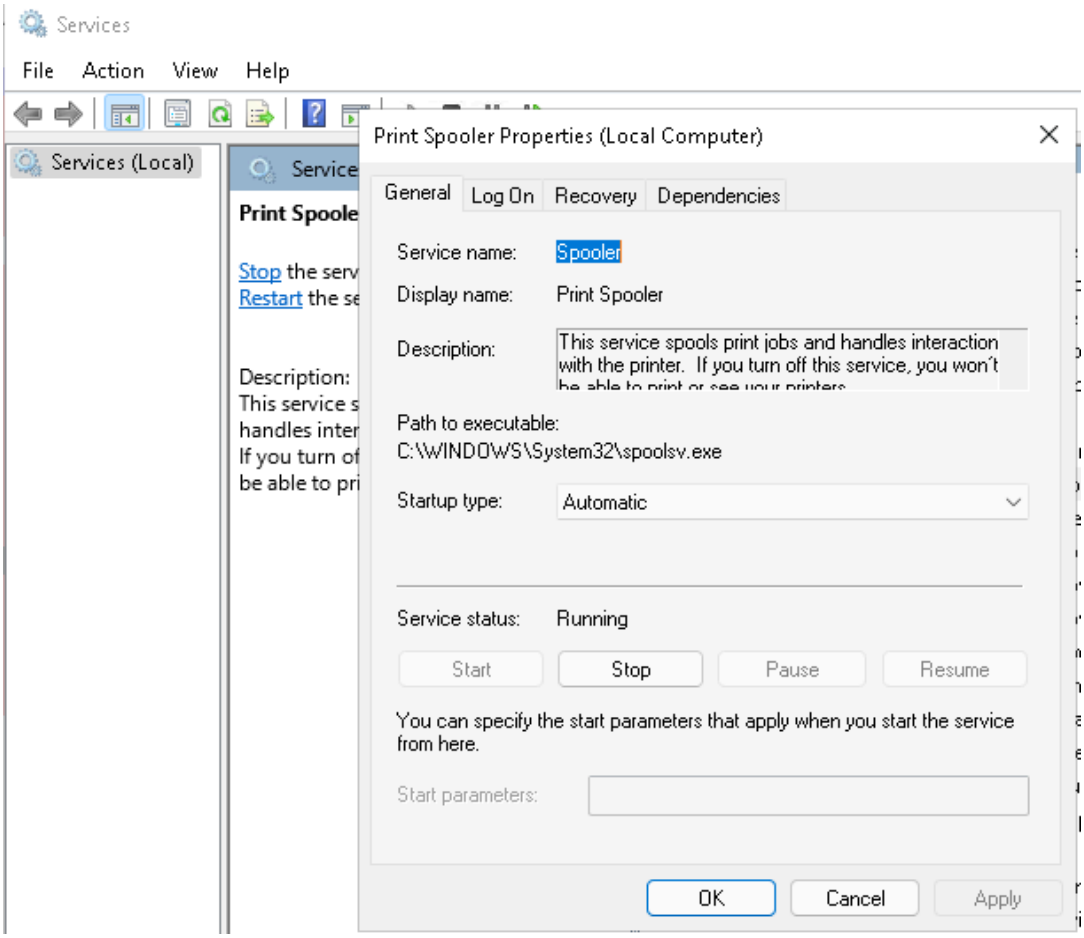
The same command can be used to stop a service remotely, but you'd type "stop" instead of "start."

Below shows a workgroup and an admin stopping the printer spooler on a workstation



You can use the SC query spooler command to make sure the service has stopped.

```
C:\Users\John>psexec \\192.168.0.135 -s -i -u Administrator sc query spooler

PsExec v2.34 - Execute processes remotely
Copyright (C) 2001-2021 Mark Russinovich
Sysinternals - www.sysinternals.com

Password:


SERVICE_NAME: spooler
        TYPE               : 110  WIN32_OWN_PROCESS  (interactive)
        STATE              : 1  STOPPED
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x0
sc exited on 192.168.0.135 with error code 0.
```

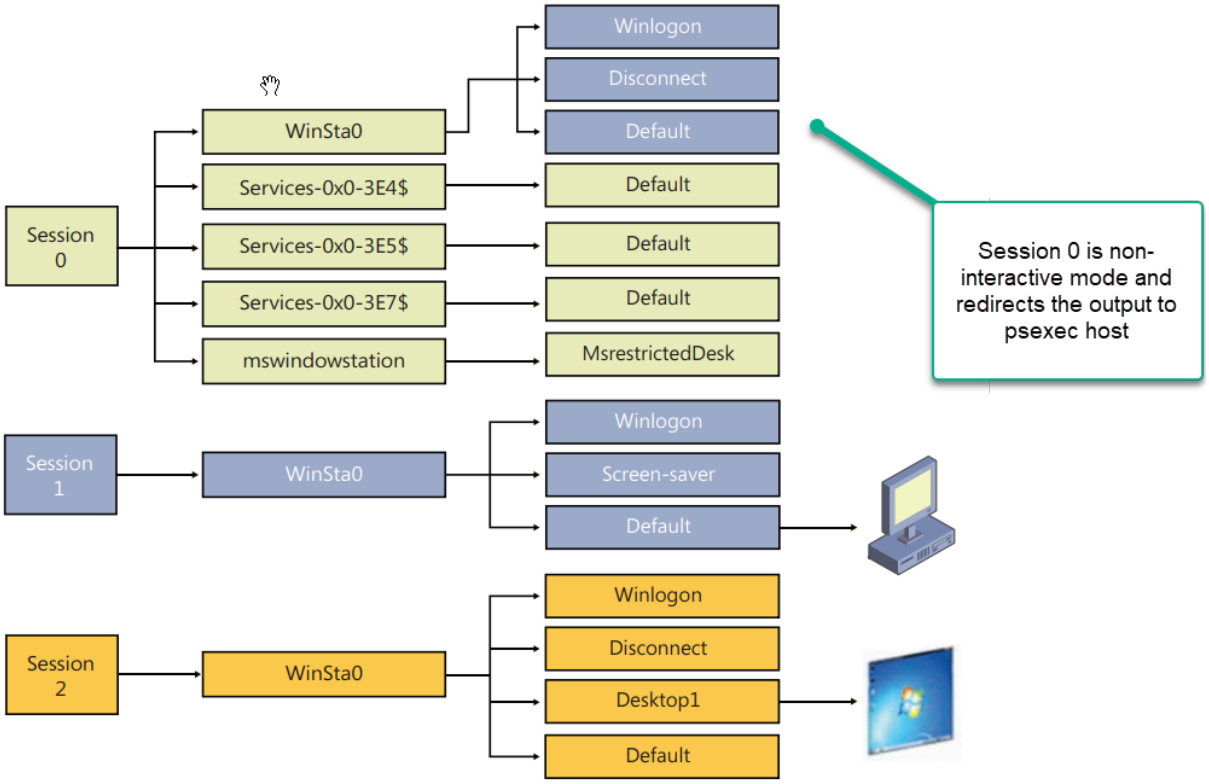


Run a GUI tool on the Remote Host: Open the Registry Editor

```
psexec \\mikelaptopw10 – I 1 -s C:\Windows\regedit.exe
```

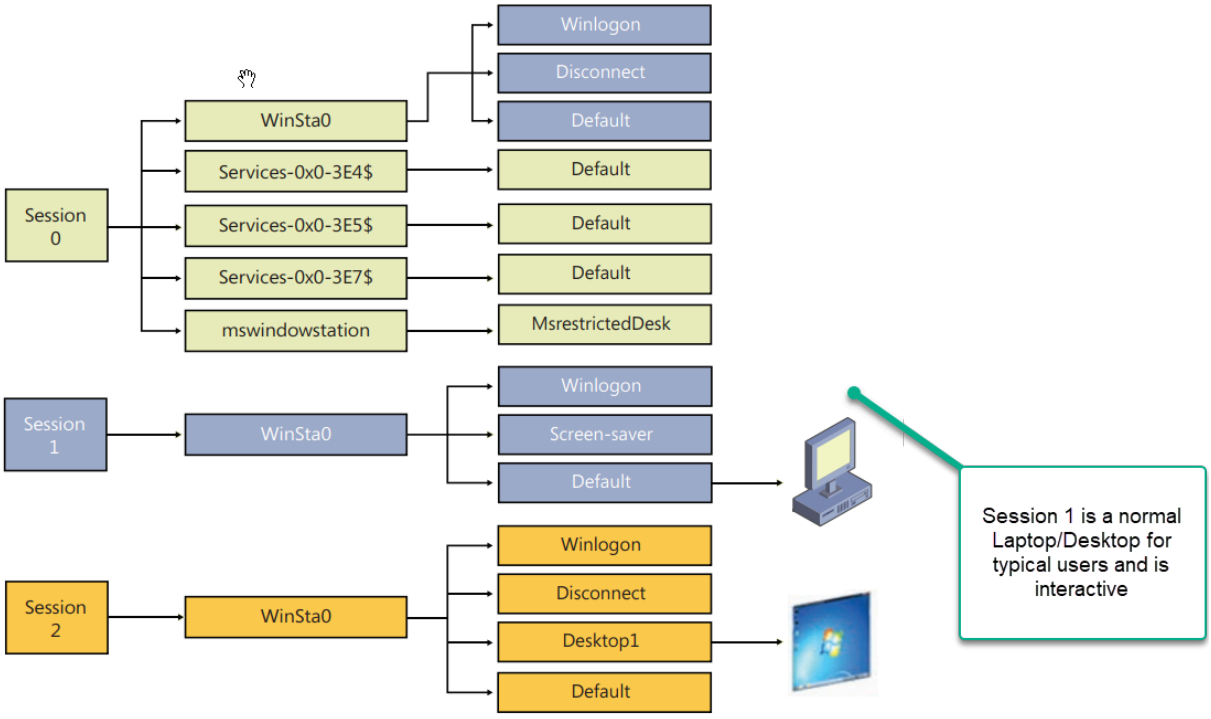
Here, we're using PsExec to launch Registry Editor on the remote machine, *mikelaptopw10*, in the System account. Because *-I 1* is used, the program will open in interactive mode, meaning that it will actually launch on the remote machine's screen. Be sure to add the session number or it defaults to 0 session (this is non interactive)

If *-i* were omitted from the above command, it would execute in hidden mode to avoid showing any dialog boxes or other windows. You can not launch or execute GUI tools in Session 0

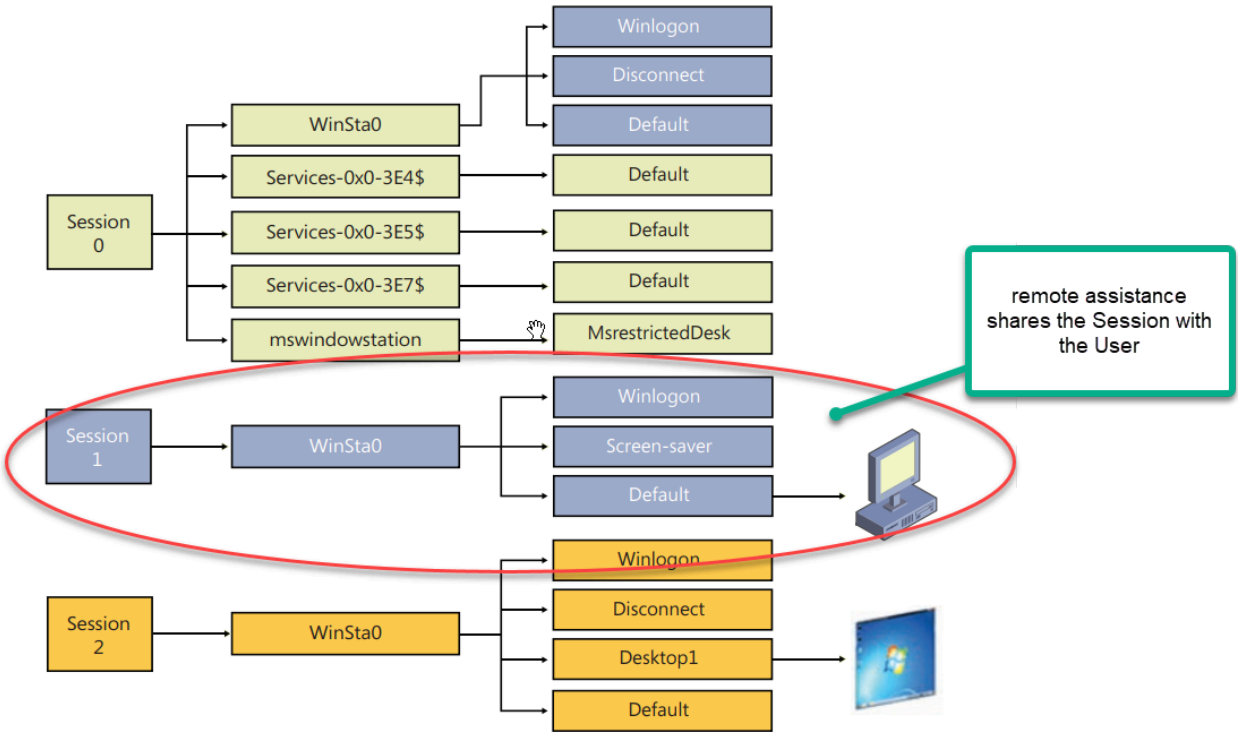


Below in the graphic we are doing the same thing in a workgroup and using session 1 while interactive -i 1 A typical laptop or desktop the user is running is

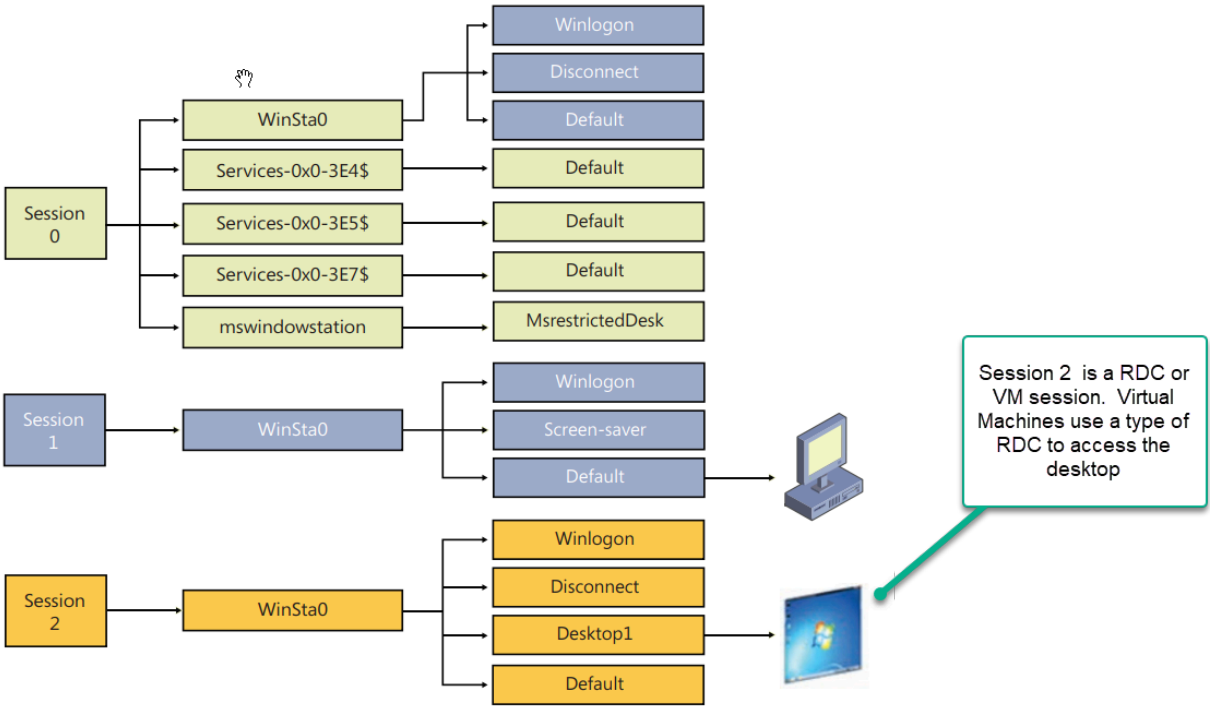
session 1. You can launch a GUI tool for the remote user to see using Session 1 or the -i 1 switch



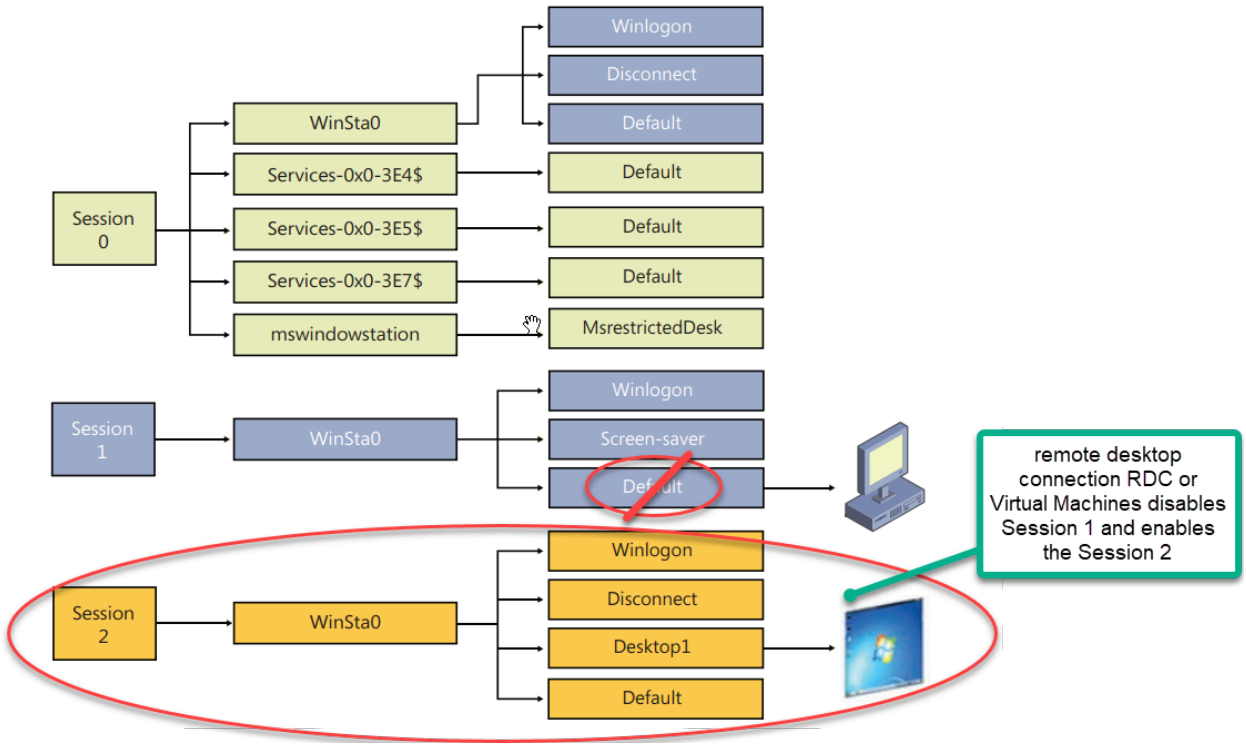
Remember GUI programs can be launched only on remote devices or local but only CLI programs allows redirection of outputs.



If you are working with a Virtual Machine or you are using RDC you need Session 2. You can launch GUI tools to the remote user in Session 2 if you are using RDC or VM.



When you RDC or remote into a VM, you use Session 2 and disable the “default” Desktop in Session 1



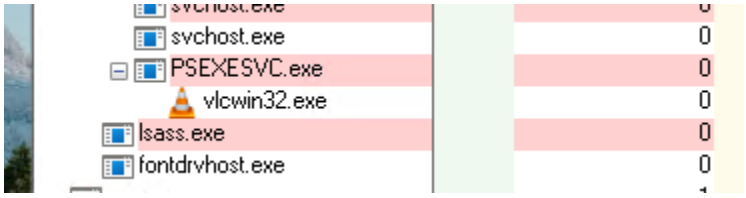
Install Program on Remote Computer

```
psexec \\J3BCD011 -c "Z:\files\ccleaner.exe” cmd /S
```

In this last example of how to use PsExec, we’re using *-c* to copy the *ccleaner.exe* program to the remote computer *J3BCD011*, and then executing it with the */S* parameter since that's what CCleaner uses to enable a silent install (requiring no user input). Adding an argument like that requires *cmd*.

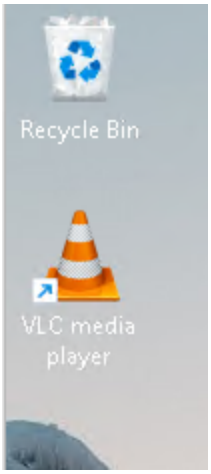
Here I am installing VLC “vlcwin32.exe” using the /S switch for silent install, I am copying the file to the remote host and executing it.

```
C:\Users\John>psexec \\192.168.0.135 -s -i -u Administrator -c c:\tool\vlcwin32.exe /S

PsExec v2.34 - Execute processes remotely
Copyright (C) 2001-2021 Mark Russinovich
Sysinternals - www.sysinternals.com

Password:

vlcwin32.exe exited on 192.168.0.135 with error code 0.
```



We had to do some research on how “vlcwin32.exe” installs and what switch to use with the install.

VLC installed!!



PsExec Can Be Dangerous

It's very important to understand how powerful PsExec is and how it could be used to compromise your computer when used in an otherwise unsecure environment.

For example, combining **-c**, **-u**, and **-p**, specifically will let anyone with a network connection to your computer, and knowledge of the admin credentials, execute secret malware with anyone's credentials.

Even that last, perfectly acceptable example in the previous section takes on a whole new purpose when you consider that instead of CCleaner, someone could install anything else they desired, in the background, and no windows pop up to show that anything is happening.

All that said, considering the firewall changes required and administrator credentials knowledge someone would have to have, there's little reason to worry so long as the admin password on the remote computer is complex and other basic security measures have been taken.

Some antivirus programs falsely identify PsExec as a dangerous file, but those warnings can be ignored if you know for sure that the program you're using is from the Microsoft source above. The reason this happens is because malware has been known to use PsExec to transfer viruses.

Windows WMI: WMI repository, Providers, Infrastructure, namespaces and more

How to use PsExec

https://4sysops.com/archives/how-to-use-psexec/

Surender Kumar

Surender Kumar has more than twelve years of experience in server and network administration. His fields of interest are Windows Servers, Active Directory, PowerShell, web servers, networking, Linux, virtualization, and penetration testing. He loves writing for his blog.

Contents

7. Run a process as the LocalSystem account

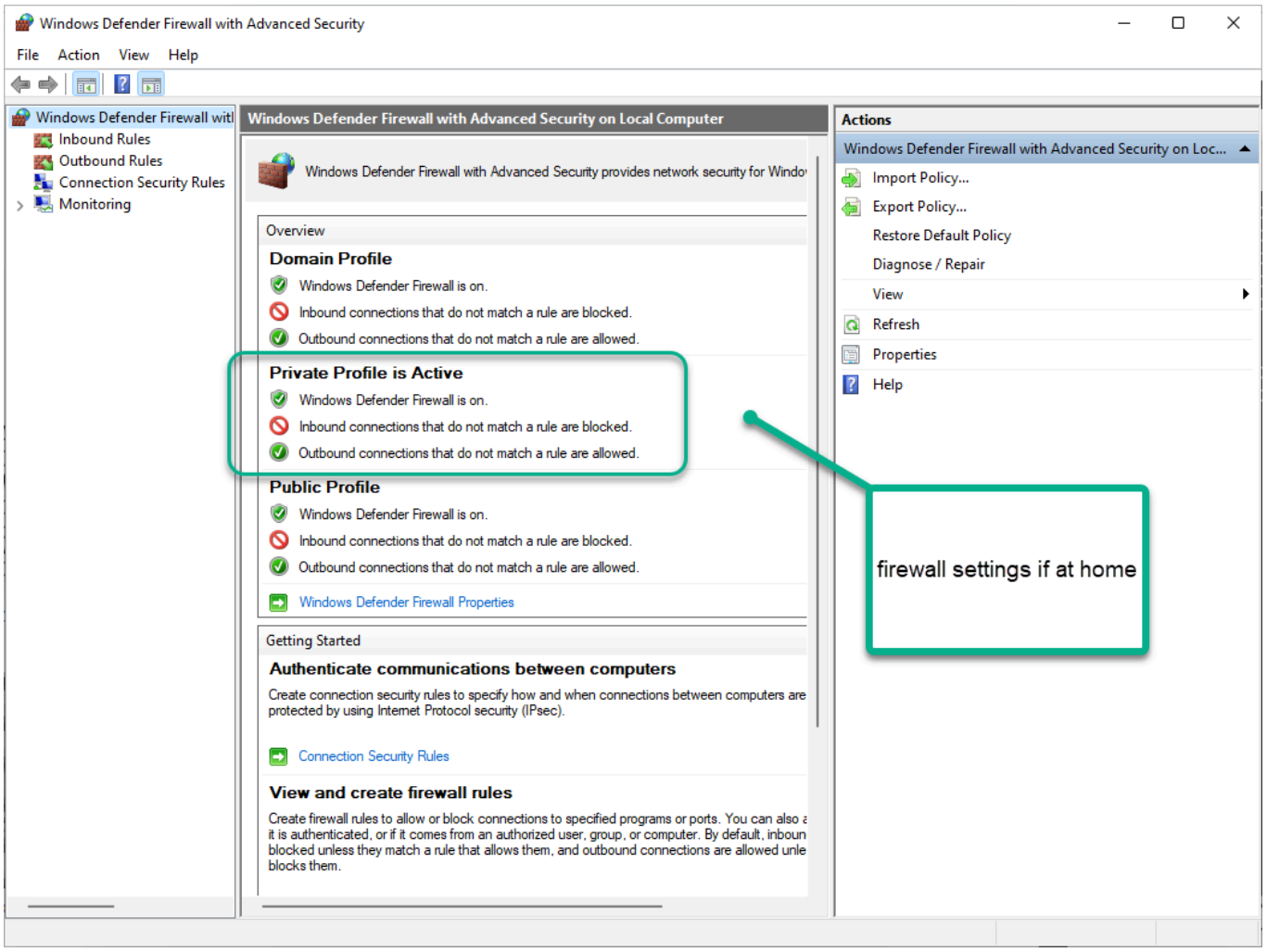
Why use PsExec

Although PsExec's ability to run processes on remote computers might not be a big deal for new admins (since PowerShell remoting is already available), it is still popular among long-time admins. So instead of considering PsExec to be an old, outdated tool, it is very powerful if you use it along with PowerShell. In my next post, I will tell you how you can use PsExec to complement PowerShell and achieve great results that are hard to achieve with PowerShell alone.

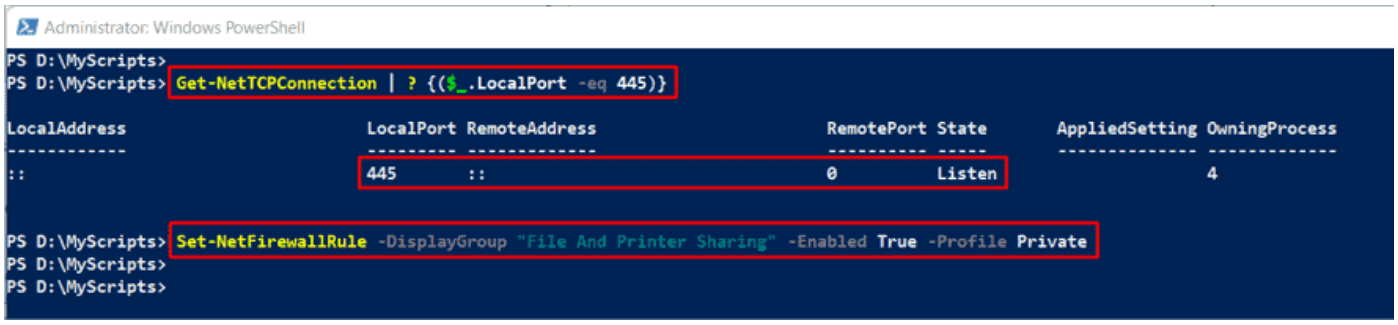
Prerequisites for PsExec

Your system and the remote systems in your network must meet the following prerequisites before you can start using PsExec:

File and printer sharing must be enabled on both local and remote computers (TCP port 445 must be open on remote computers). It could be a potential security risk, so make sure you enable this port for the Private profile in the Windows firewall.




```
Set-NetFirewallRule -DisplayGroup "File And Printer Sharing" -Enabled True -Profile Private
```



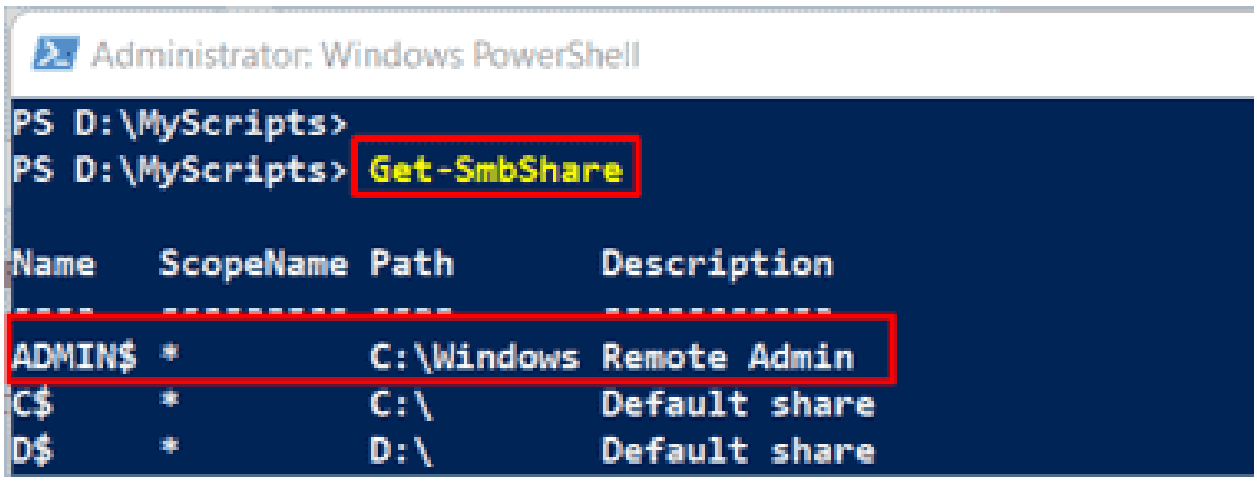
Enable File and Printer Sharing firewall rule using PowerShell

You could also use the *Invoke-Command* cmdlet in PowerShell to set this firewall rule in multiple remote computers at once.

Get-NetTCPConnection

```
PS C:\Users\Dutch> Get-NetTCPConnection

LocalAddress                        LocalPort RemoteAddress                       RemotePort State       AppliedSetting
------------                        --------- -------------                       ---------- -----       --------------
::                                  49674     ::                                  0          Listen
::                                  49673     ::                                  0          Listen
::                                  49670     ::                                  0          Listen
::                                  49667     ::                                  0          Listen
::                                  49666     ::                                  0          Listen
::                                  49665     ::                                  0          Listen
::                                  49664     ::                                  0          Listen
::                                  7680      ::                                  0          Listen
::                                  5357      ::                                  0          Listen
::                                  3389      ::                                  0          Listen
::                                  2869      ::                                  0          Listen
::                                  445       ::                                  0          Listen
::                                  135       ::                                  0          Listen
0.0.0.0                             54027     0.0.0.0                             0          Bound
0.0.0.0                             54019     0.0.0.0                             0          Bound
0.0.0.0                             54018     0.0.0.0                             0          Bound
0.0.0.0                             54017     0.0.0.0                             0          Bound
```

The default admin$ share must be enabled.




```
PS C:\Users\Dutch> Get-SMBshare

Name   ScopeName Path        Description
----   --------- ----        -----------
ADMIN$ *         C:\WINDOWS  Remote Admin
C$     *         C:\         Default share
IPC$   *                     Remote IPC
```

View default admin shares using PowerShell

Security considerations

If a remote computer is in the domain, the members of the domain admins group will be able to use PsExec without a problem.

An earlier version of PsExec wasn't that safe since it was used to transmit passwords and commands in clear text over the network. However, starting with version 2.1, it encrypts the alternate credentials and commands in transit. Please make sure you are using the latest version of PSExec (2.34 is the latest version at the time of writing).

PsExec is kind of like a double-edged sword. In the right hands, it can be a great tool, but in the wrong hands, it can be a disaster. Due to its abilities, it is often used by hackers in combination with other tools, such as the Metasploit framework and Mimikatz, to carry out malicious attacks. This is the reason that major antivirus products have started flagging PsExec as malware. In a nutshell, PsExec is a safe sysadmin tool in itself, but in the end, it totally depends on how the person is using it.

Since PsExec uses the file and printer sharing service, which is vulnerable to various viruses and ransomware, please make sure you properly secure and restrict this service and its associated ports to the local network only; never expose it to a public network.

Downloading and installing PsExec

As mentioned above, PsExec is part of the PsTools suite. To use PsExec utility, we need to download the PsTools suite from the Sysinternals website.

There is no installer in the PsTools zip file. All you need to do is extract the files from the zip archive and start using them. The installation of PsTools depends on how and where you want to extract the files. Some people prefer directly extracting them inside the **%systemroot%\System32** directory so that the executables can run directly without having to specify the full path.

I like extracting these files into a separate directory inside **%systemdrive%\Program Files** alongside other programs and then setting up the **Path** environment variable. You could choose either way; it's just a matter of your own preference. Let's extract the zip archive using the following PowerShell command:

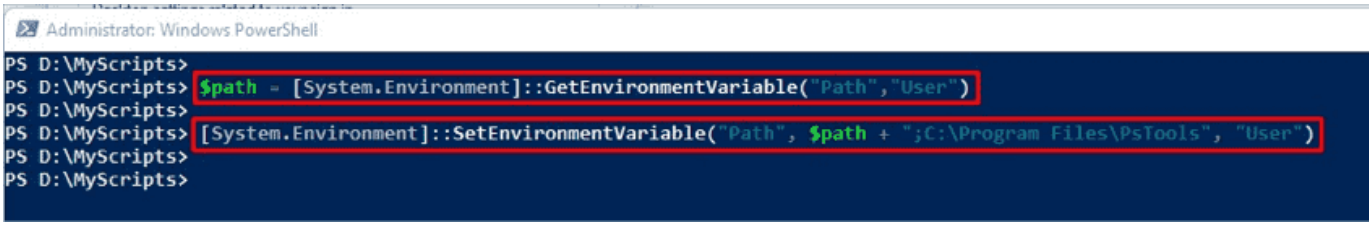
```
Expand-Archive -Path "$env:USERPROFILE\downloads\PsTools.zip"
-DestinationPath "C:\Program Files\PsTools" -Force
```

The PsTools are now in the right place. You just need to modify the **Path** environment variable and specify the full path of your destination directory. Again, there are two **path** environment variables—**System** and **User**. It depends on how you want to use the PsExec tool. If you want PsExec (and other tools) to be available for all users in the current system, you can add it to the **System Path** variable. If you want PsExec to be available only to your current user profile, you could add it to the **User Path** variable. To do this, you can use one of the following commands:

For the User variable:

"If you want to setup tools for logon only"

```
$path = [System.Environment]::GetEnvironmentVariable("Path","User")
[System.Environment]::SetEnvironmentVariable("Path", $path + ";C:\Program
Files\PsTools", "User")
```

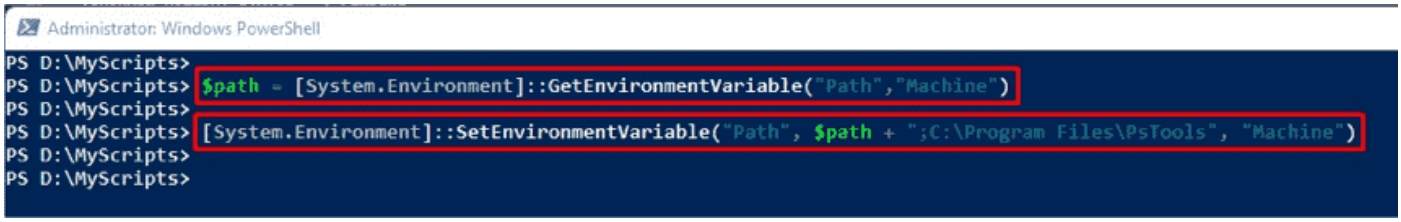


Add PsTools to the User Path environment variable

For the System variable

"If you want to setup tools for anyone who logs on to your PC"

```
$path = [System.Environment]::GetEnvironmentVariable("Path","Machine")
[System.Environment]::SetEnvironmentVariable("Path", $path + ";C:\Program Files\PsTools", "Machine")
```



Add PsTools to the System Path environment variable

No matter which environment variable you set, be sure to specify the full path of the directory to which you extracted the **PsTools** executables. Also note that the semicolon (;) before the directory path is intentionally added to separate the new value from the old values.

Similarly, you can also view the **System Path** environment variable using the following command:

```
[System.Environment]::GetEnvironmentVariable("Path","Machine") -Split ";"
```

At this point, PsExec (and other utilities in PsTools) are ready for use in your system.

Use PsExec on a single remote computer

To start using PsExec, just close the existing PowerShell console and launch a new one. If you want to use it in a command prompt, you can launch a command prompt. Whichever you choose, just make sure you launch an elevated session since PsExec requires administrator privileges to run programs on remote

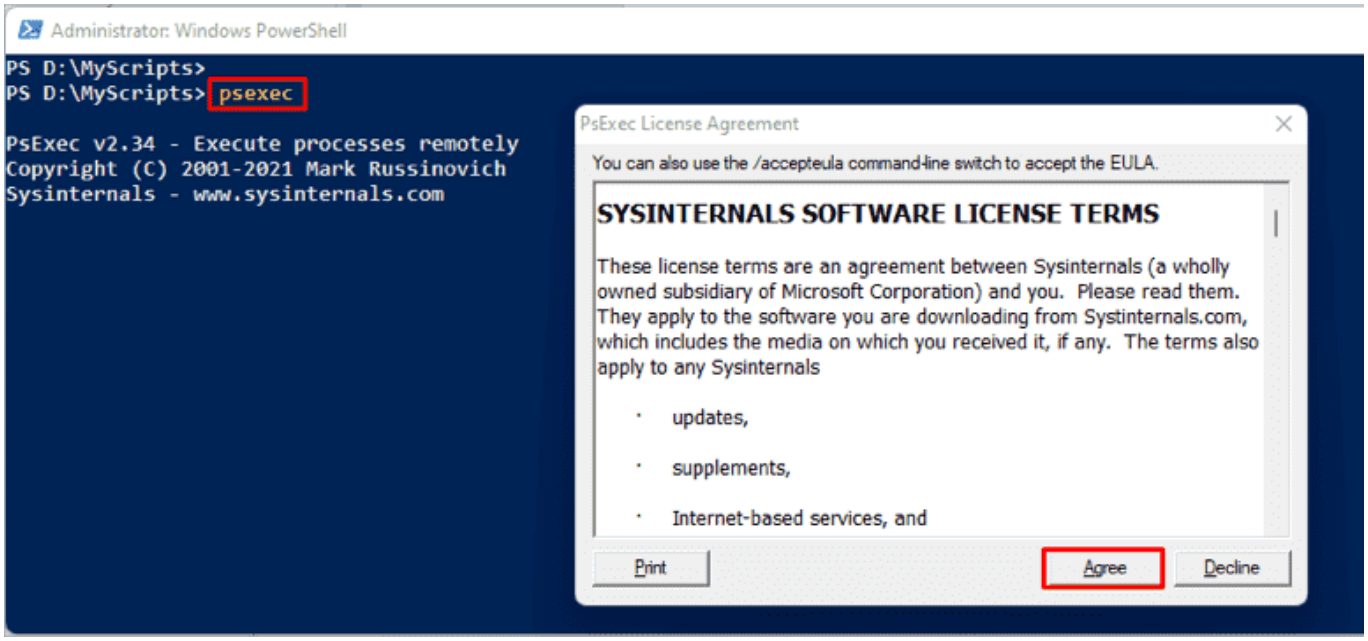
computers. Non-admins will also be able to use PsExec locally, but there is not much you can do without admin privilege.

PsExec syntax

The syntax of PsExec is fairly simple, as shown below:

psexec [\\computer[,computer2[,...] | @file]][-u user [-p password][-n s][-r servicename][-h][-l][-s|-e][-x][-i [session]][-c executable [-f|-v]][-w directory][-d][-<priority>][-a n,n,...] cmd [arguments]

If the syntax sounds complicated to you, trust me, it will become clear as we use it. When you run PsExec for the first time, you will see an end-user license agreement (EULA), as shown in the following screenshot:



Displaying the PsExec EULA

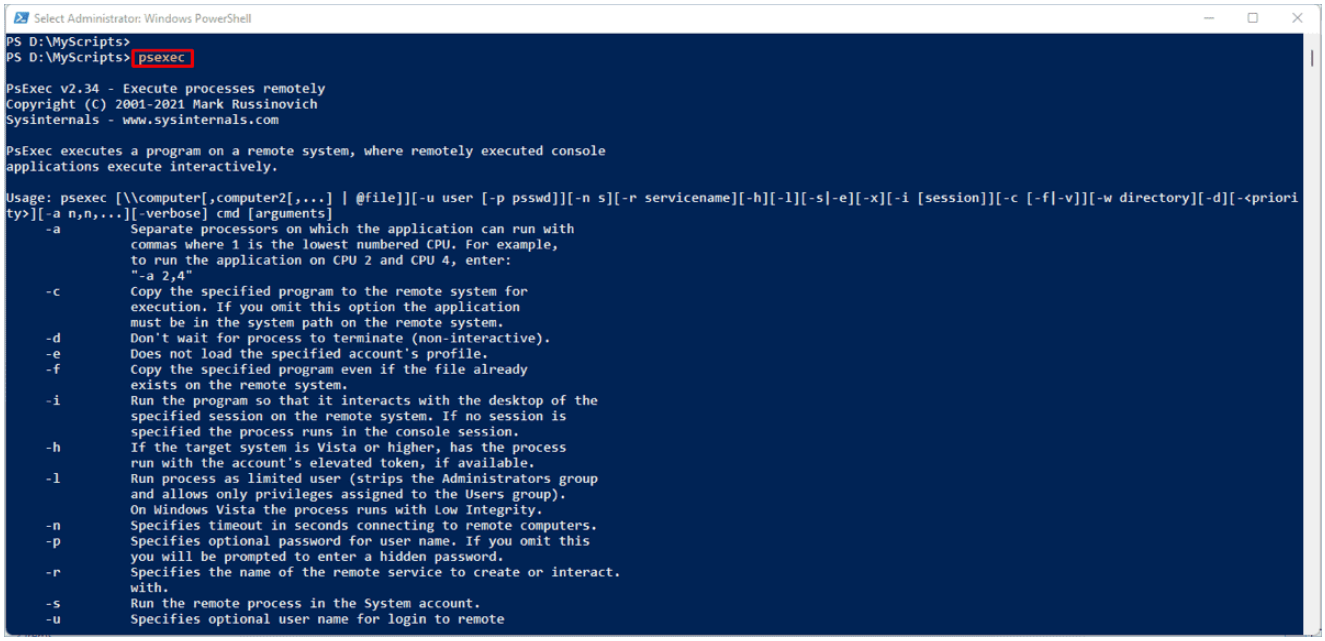
Click the **Agree** button to accept the EULA. To suppress this EULA, you could use the */accepteula* switch as shown below:

psexec /accepteula

Getting help

Help documentation is the most important thing for getting started with any tool. If you run the **psexec** command without any switch, you will see detailed help,

including its usage syntax and all the supported switches. The following screenshot shows what the help section looks like.

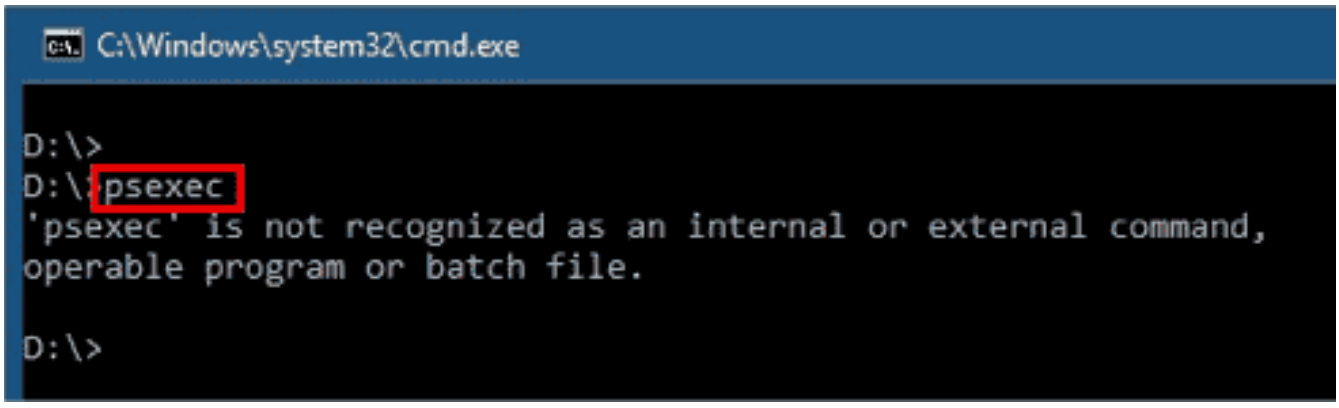


Displaying the PsExec help

I will not discuss every switch in detail, but will try to cover the most important ones. While researching and experimenting with PsExec, you can always refer to the built-in help.

PsExec command error

If you do not see this help page but instead see an error that says *'psexec' is not recognized as an internal or external command, operable program or batch file*, it means you did something wrong while setting up the **Path** environment variable.



'psexec' is not recognized as an internal or external command operable program or batch file

To fix this error, make sure you set the **Path** environment variable correctly.

Exit and error codes

When you use PsExec to run a program (or a process), it might display an exit code for that program and returns an error code. The error code returned is not generated by PsExec; it is generated by the original program that you run using PsExec. So, if you get any error while running a program, you should consult the program's documentation to determine what that error code means. An **exit code of 0** typically indicates successful execution.

Impersonation and alternate credentials

User impersonation occurs when a program runs in the security context of a user other than the currently logged-in user. When you run a program (or process) with PsExec, the program is executed on the remote computer impersonating the user account that you're using on your local computer to run PsExec. In other words, the process that you start on a remote computer runs by impersonating your user account on a local system. Impersonation has certain limitations in Windows. The main limitation is that the remote process cannot access any resources on the

network. To get around this problem, PsExec allows you to provide alternate credentials using the -u and -p switches.

To run a process on a remote computer using alternate credentials, you could use PsExec, as shown below:

psexec \\remote_computer -u domain\admin -p password -i process_name

where:

- The \\remote_computer should be replaced with the name or IP address of the remote computer. You could specify multiple computers separated by a comma (,).
- The -u switch specifies the alternate user name. The -u should be followed by the user name, which can be either a local admin user on the remote computer or a domain admin.
- The -p switch is used to specify the password of the alternate user account you specified in the -u switch. Note that -p requires you to type the password in clear text, so even though PsExec will encrypt these credentials in transit, someone could still see the password as you type it in clear text. Therefore, it is a good idea to skip the -p switch and use the -u switch alone. PsExec will automatically prompt you to type the password when you press Enter, which is a safer way.
- The -i switch specifies the interactive mode. This switch was not needed until PsExec version 2.20. You should use this switch with version 2.30 and above; otherwise, you will get an error.
- The process name should be replaced with the command or process you want to run on the remote computer.

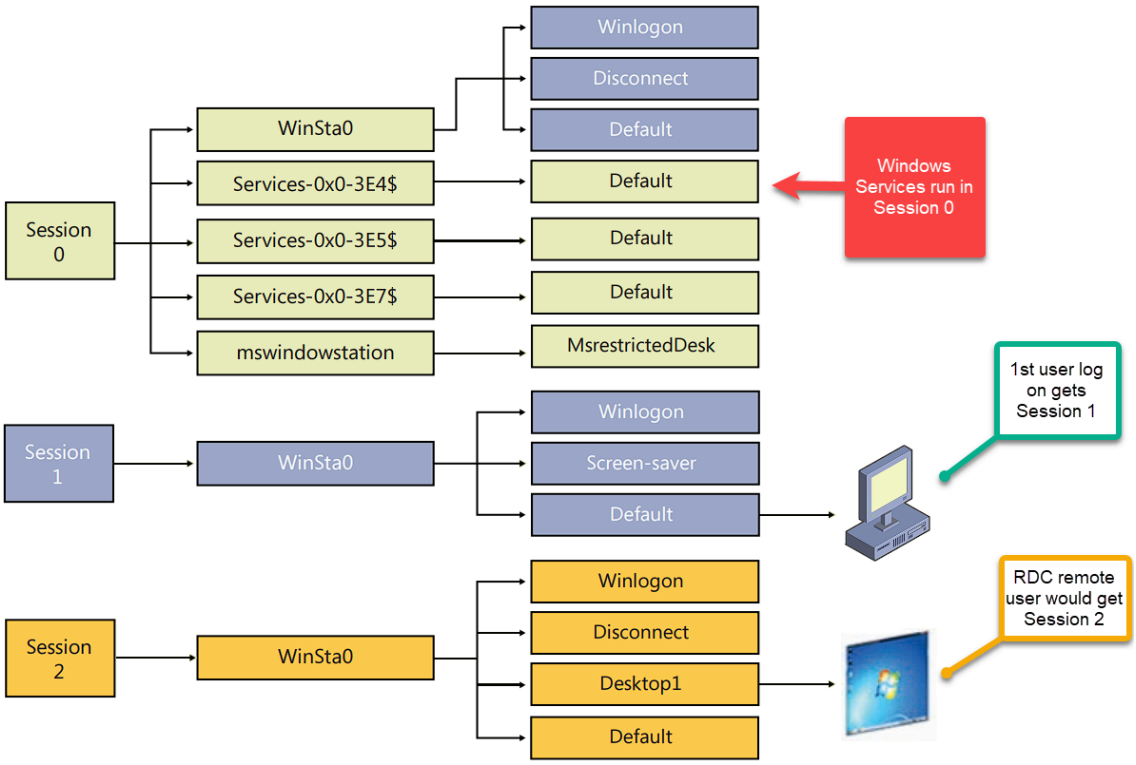
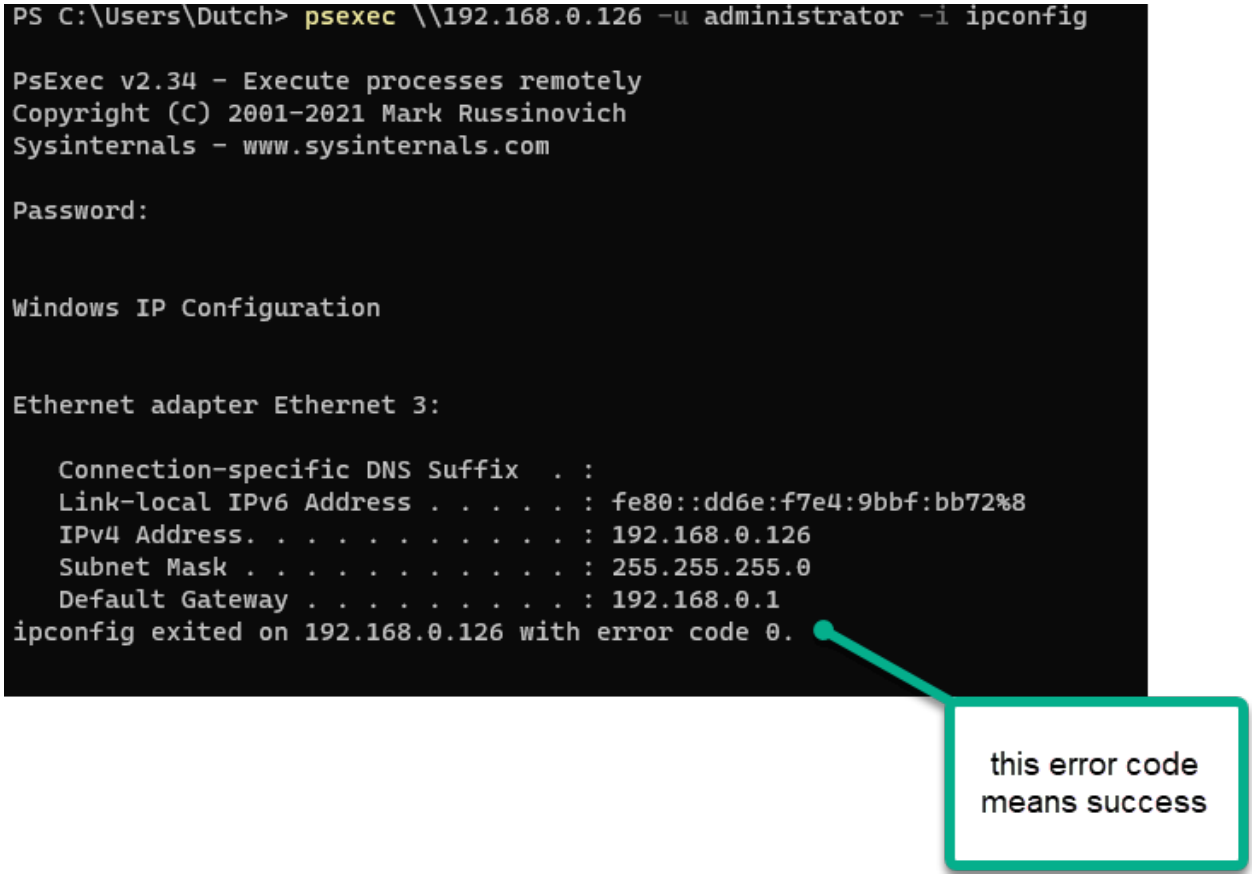


FIGURE 2-11 Relationship between sessions, window stations, and desktops.

Example

psexec \\webserver -u domain\admin -i ipconfig

This command runs the **ipconfig** command on a remote computer named **webserver** using the alternate credentials of the domain admin.

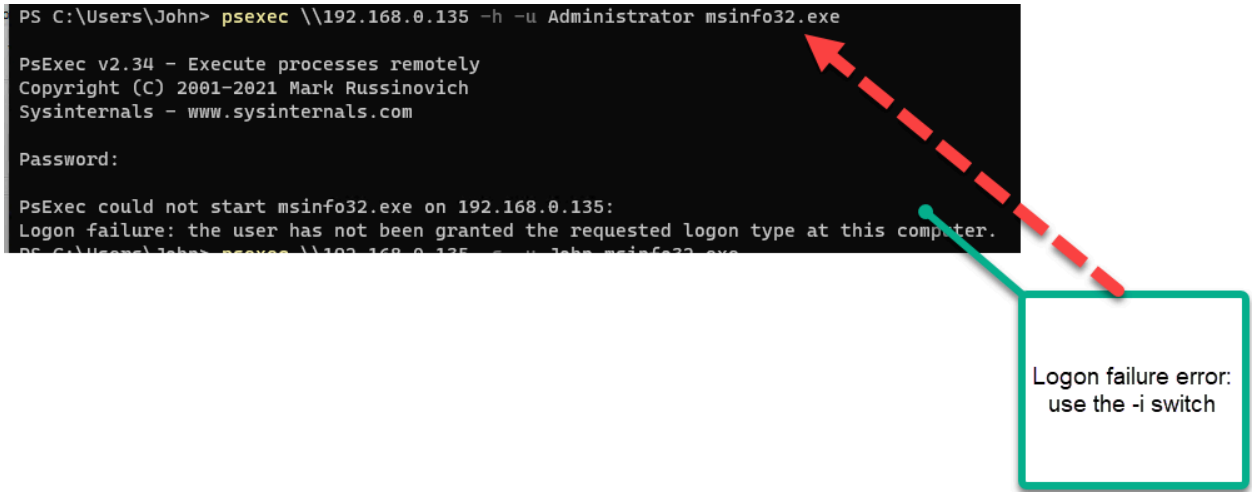


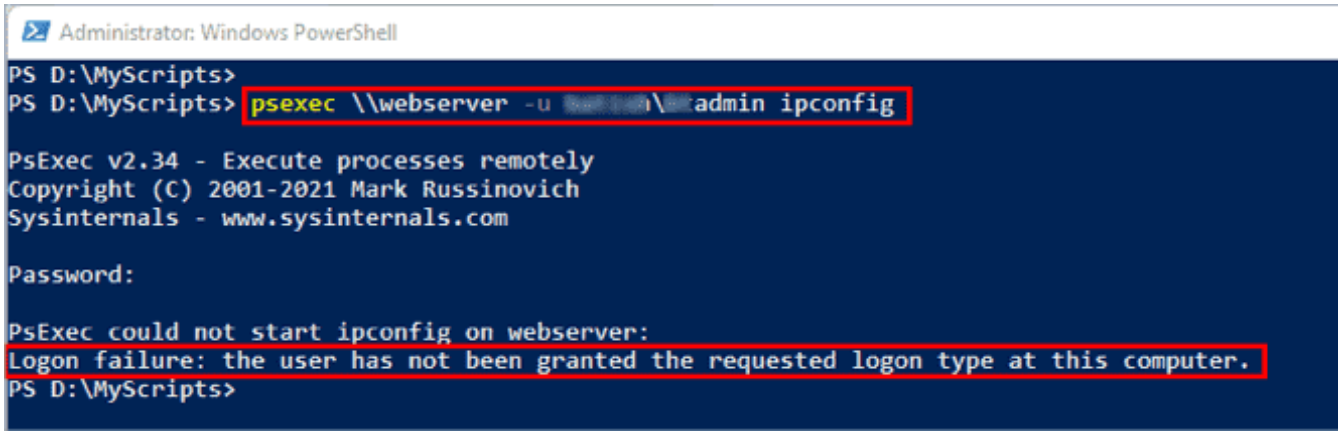
Running a command on a remote computer using alternate credentials with PsExec

Running a command on a remote computer using alternate credentials with PsExec

The screenshot shows that PsExec prompted for the password on the fly, since I did not use the -p switch. **Error code 0** indicates that the **ipconfig** command exited with success.

If you get an error that says "Logon failure: the user has not been granted the requested logon type at this computer," as shown in the following screenshot, make sure to use the -i switch with the command. It is needed with PsExec version 2.30 and above.





Logon failure the user has not been granted the requested logon type at this computer

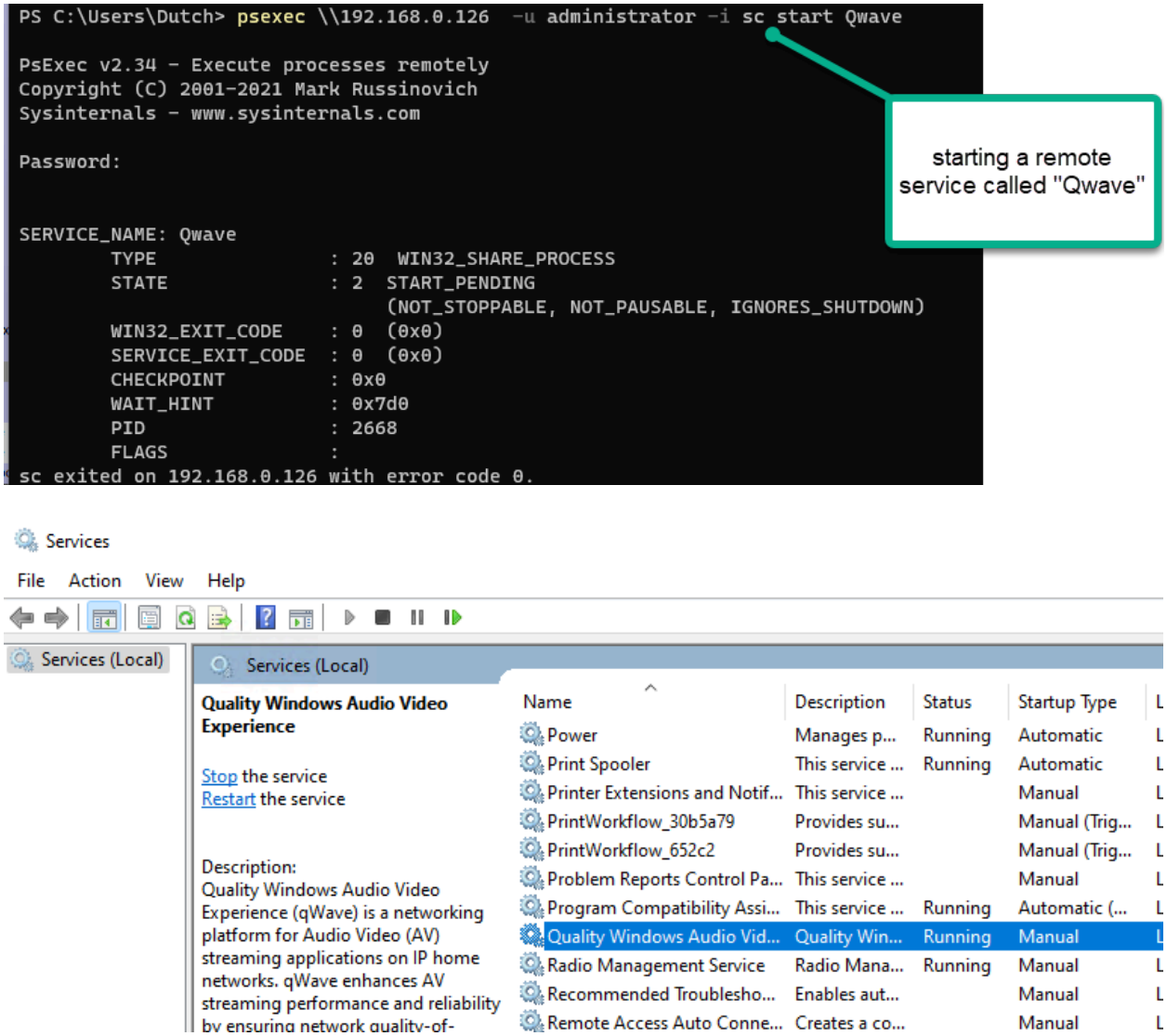
Use PsExec on multiple remote computers

One cool feature of PsExec is that you can specify multiple computers on which to run a program (or process) at the same time. There are several ways to specify multiple remote computers.

Using comma-separated computer names

PsExec allows you to specify a comma-separated list of computers in a domain or workgroup.

psexec \\192.168.0.126 -u administrator -i sc start Qwave



Qwave service started on remote host

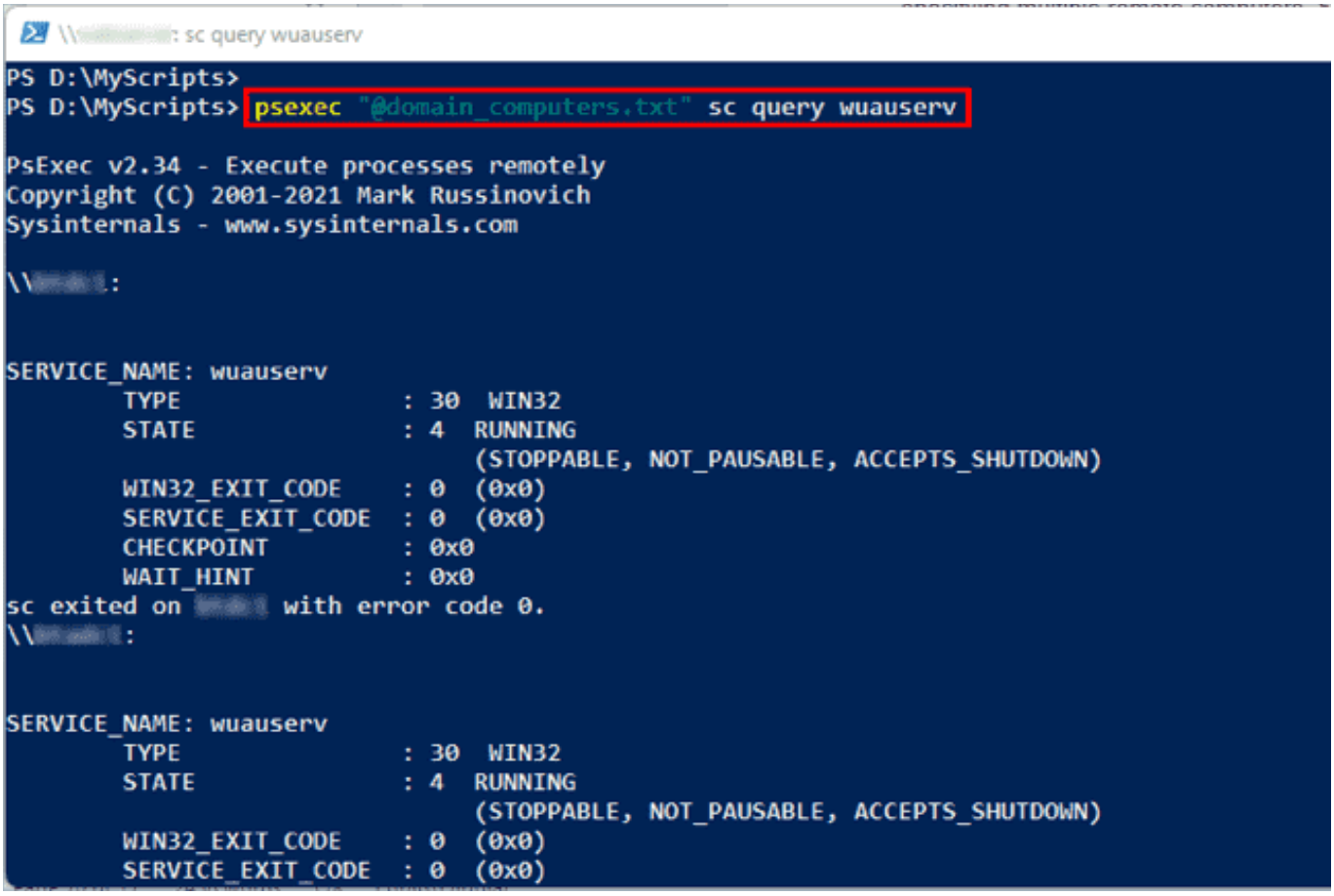
psexec \\webserver01,webserver02,fileserver01,fileserver02 sc start wuauserv

The above command will run **sc start wuauserv** to start the **Windows update** service on multiple remote computers specified in a comma-separated string. Remember not to use a space between computer names and commas.

Using a file to specify multiple computer names

You can also use a text file containing multiple computer names (one per line) using the @file format to specify the file name.

psexec @domain_computers.txt sc query wuauserv



Running a command on multiple remote computers using files with PsExec

The above command runs the **sc query wuauserv** command to view the status of the **Windows update** service on all the computers mentioned in the specified file (@domain_computers.txt in our example). Make sure there is one computer name per line in the file; otherwise, you will get an error.

Note that I am using PowerShell to run PsExec, so I had to enclose the file name in single or double quotes (for example, "@domain_computers.txt"). This is necessary, since PowerShell, by default, interprets the @ symbol as an array subexpression operator.

If you're running PsExec at the normal command prompt (cmd.exe), there is no need to enclose the @filename in quotes.

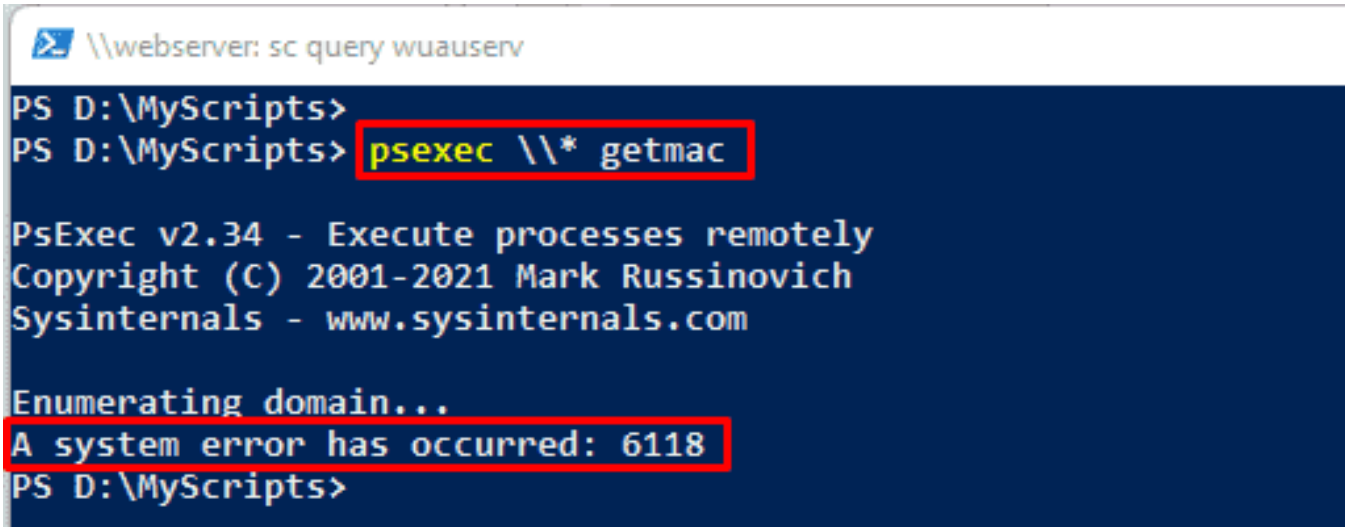
Using a wildcard to run the process on all computers

PsExec also allows you to use the wildcard character (*) to specify all the computers in the current domain or workgroup.

psexec * getmac

This will run the **getmac** command on all the remote computers that are in the same workgroup or domain.

Using * causes PsExec to enumerate all the computers in the current domain or workgroup by running the **net view /all** command under the hood. Since the **net view** command depends on NetBIOS, if you run this command in your domain or workgroup environment, it will most likely fail with an error stating: "A system error has occurred: 6118".



PsExec: A system error has occurred 6118

This error essentially means that the firewall (either the Windows firewall or the antivirus firewall) is blocking the SMB connections, or the network discovery is turned off on the remote computer(s).

This is an outdated and inefficient way of enumerating network computers. Later in this guide, I will tell you how you can utilize PowerShell to build a file containing multiple (or all) domain computers so that you can easily run processes on those computers with the help of PsExec, and you won't have to use * to enumerate computers.

Run a process as the LocalSystem account

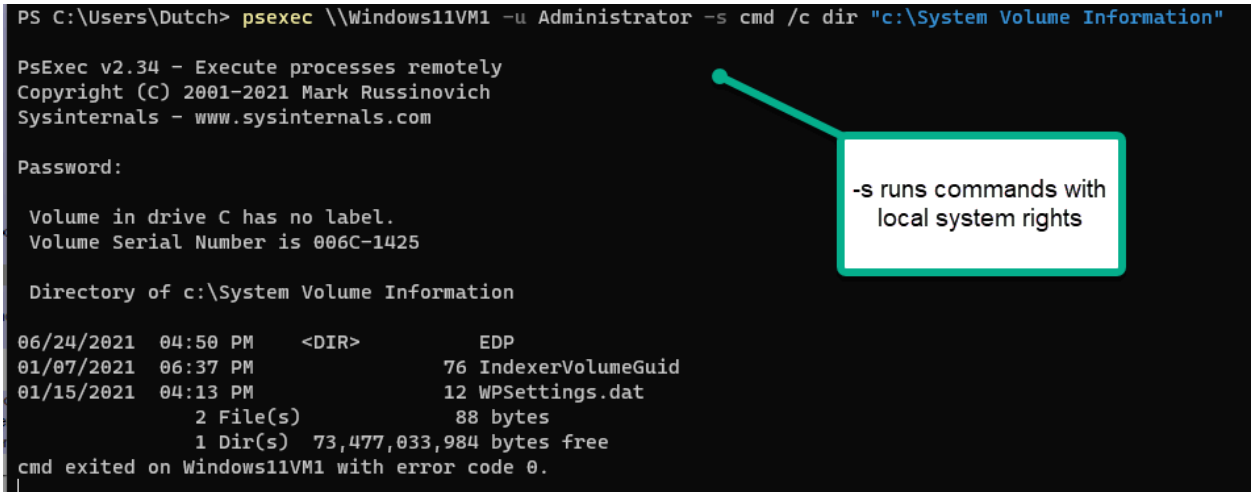
The LocalSystem account is a predefined account with extensive privileges on the local computer. It is created by the operating system during installation. The security token of the LocalSystem account includes the security identifiers "NT AUTHORITY\System" and "BUILTIN\Administrators".

These accounts have unrestricted access to most of the system objects. The advantage of running a process or program under the LocalSystem account is that the process will have unrestricted access to all system resources.

One of the most powerful features of PsExec is its ability to launch a process (or program) with LocalSystem account privileges. **This feature is most often exploited by hackers and other malicious users**. You can imagine what malicious users can do on your system if they succeed in running a malicious process with the LocalSystem account privilege.

To launch a process as a LocalSystem account, you can use the -s switch with PsExec, as shown below:

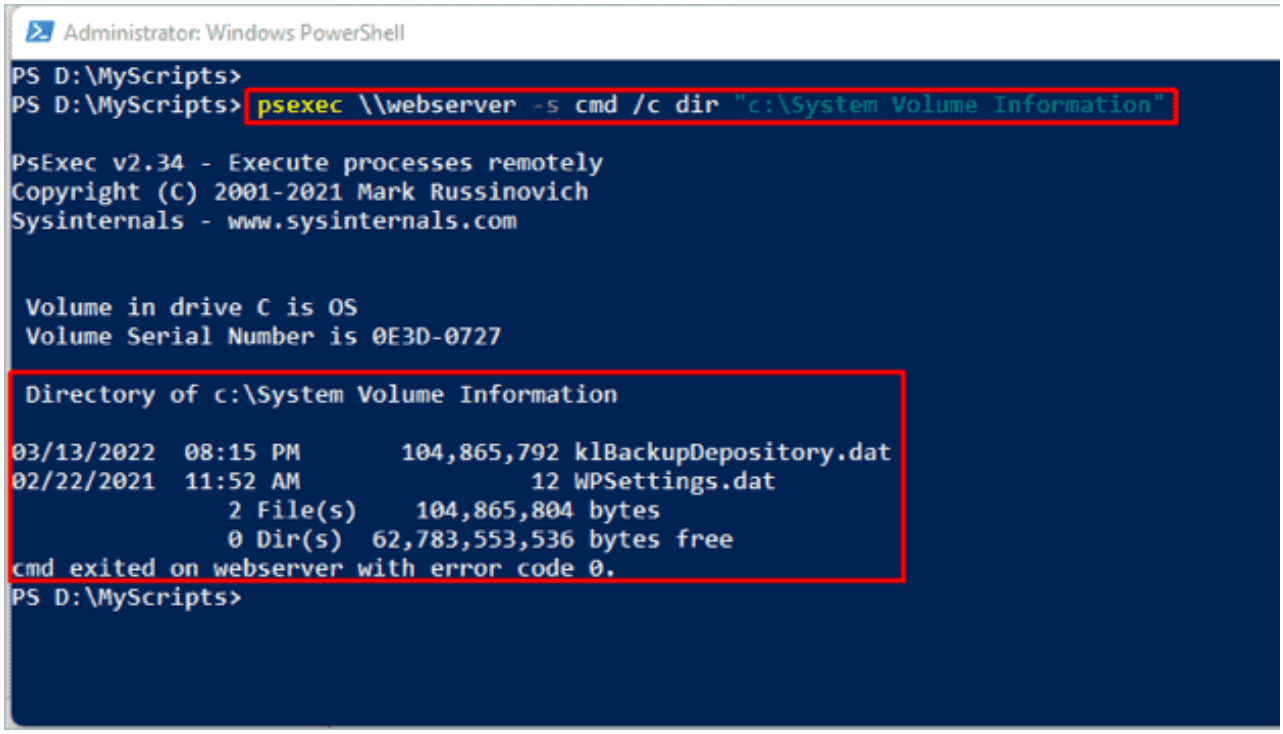
```
psexec \\webserver -s cmd /c dir "c:\System Volume Information"
```



Because I am not on a domain with admin rights alternate credentials are required.
Your computers are in a workgroup

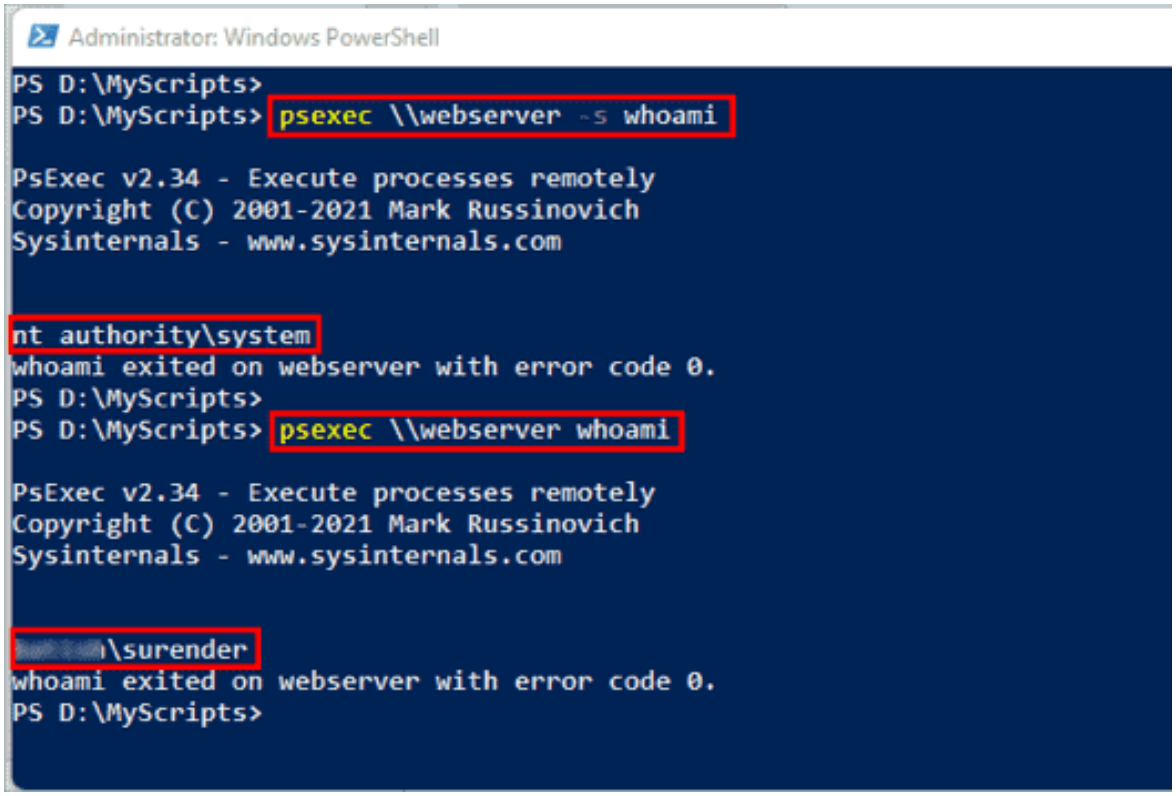
The -s switch causes PsExec to launch the **cmd.exe** process with the LocalSystem account on the webserver. Note that the /c switch used here belongs to **cmd.exe** itself, which is useful for running a command on the fly and then terminating it. It is particularly useful when you want to run a single command without having to launch an interactive console session of **cmd.exe** on a remote computer. Don't confuse this with the -c switch of PsExec, which is useful for copying a program to a remote computer when it doesn't exist in the default path. We will discuss the -c switch in greater detail in a separate section below.

This command displays the contents of the System Volume Information directory on the remote webserver, which is not accessible by regular user accounts.



Run a process on a remote computer under the LocalSystem account with PsExec

The following image shows how the -s switch makes a difference in the PsExec command:



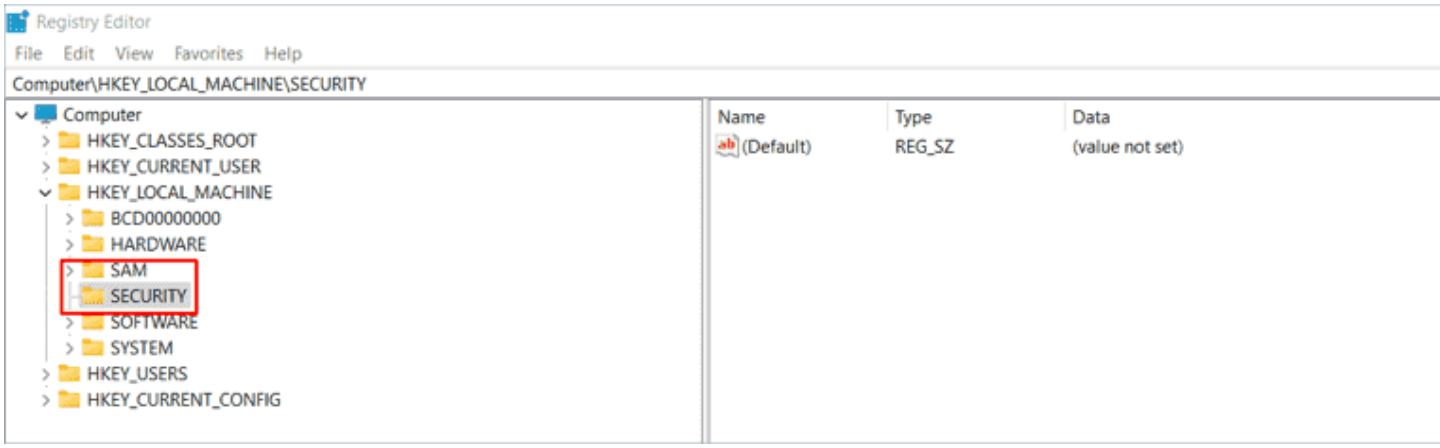
Local System Account vs. Impersonate User

This image shows that when you run a process with the -s switch, it is launched with the "NT AUTHORITY\System" privilege. When you run a process without specifying any alternate credentials, your currently logged-in user is impersonated.

Interactive mode “running local apps with “system” rights”

Another important feature of PsExec is its ability to launch a process in an interactive mode using the -i switch. It is most commonly used along with the -s switch to launch a GUI-based program (or process) on a local computer with the "NT AUTHORITY\System" privilege.

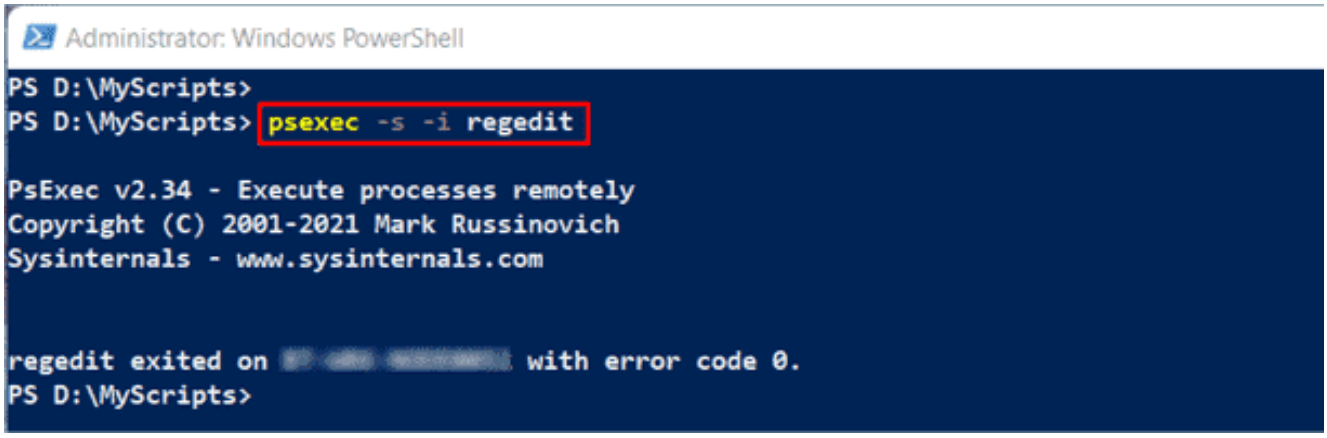
For example, when you normally launch the Windows registry editing tool (regedit.exe*)* and try to open the HKEY_LOCAL_MACHINE\SYSTEM and HKEY_LOCAL_MACHINE\SAM subkeys, you will not see anything there because these subkeys are only accessible to the SYSTEM account.



Direct access to SAM and SECURITY registry subkeys

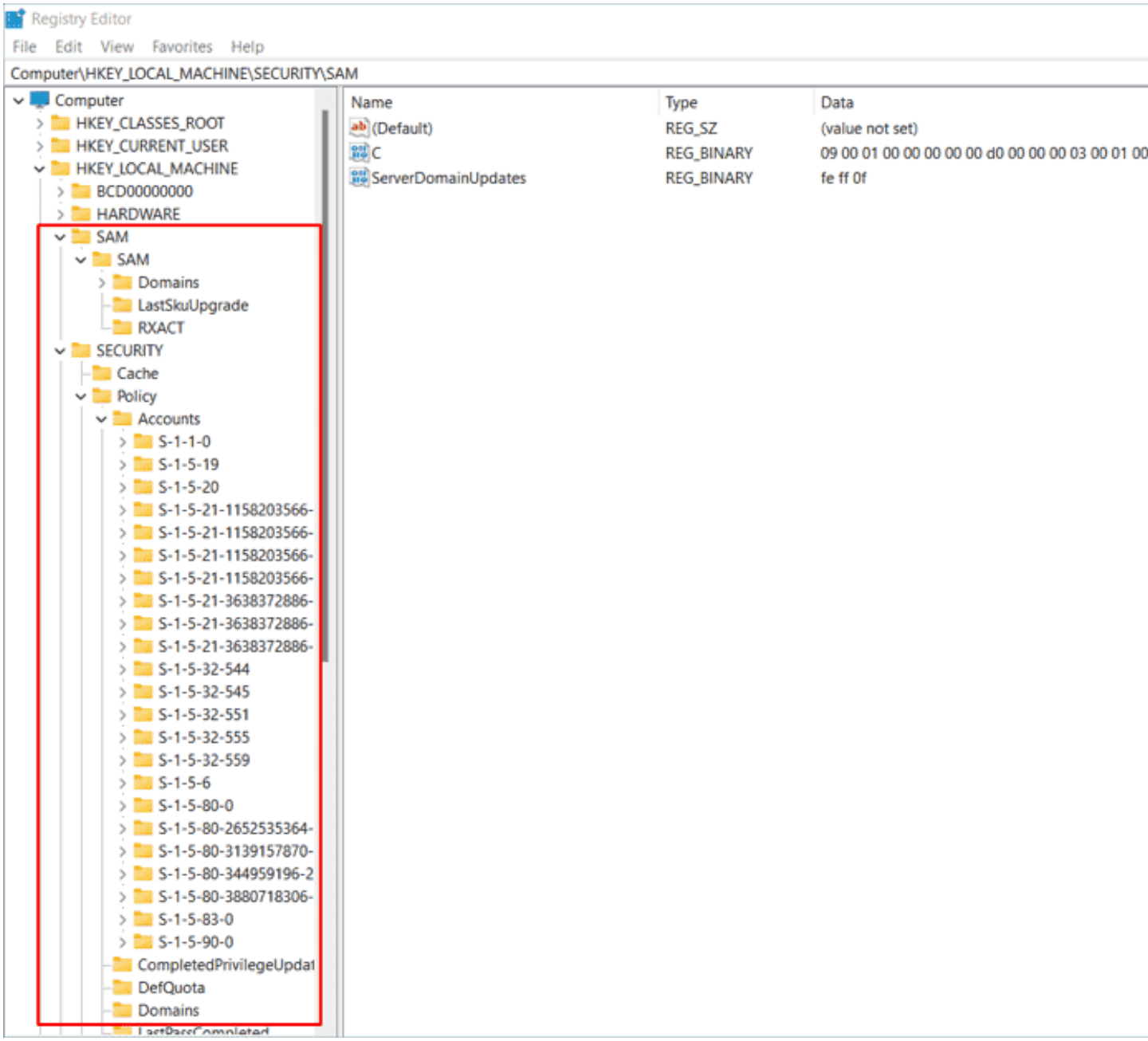
However, if you launch regedit.exe with PsExec using the -s and -i switches, as shown in the following command, you will be able to see everything in these subkeys.

psexec -s -i regedit



Interactively launch regedit under the System account with PsExec

Take a look at the following screenshot showing all the subkeys under SYSTEM and SAM. This became possible since PsExec launched the *regedit.exe* with "*NT AUTHORITY\System*" privilege.



Accessing SAM and SECURITY registry subkeys using System Account

When you launch the process this way, PsExec waits for the program to finish running before the exit code is returned. If you do not want to wait, you can use the -d switch, as shown below:

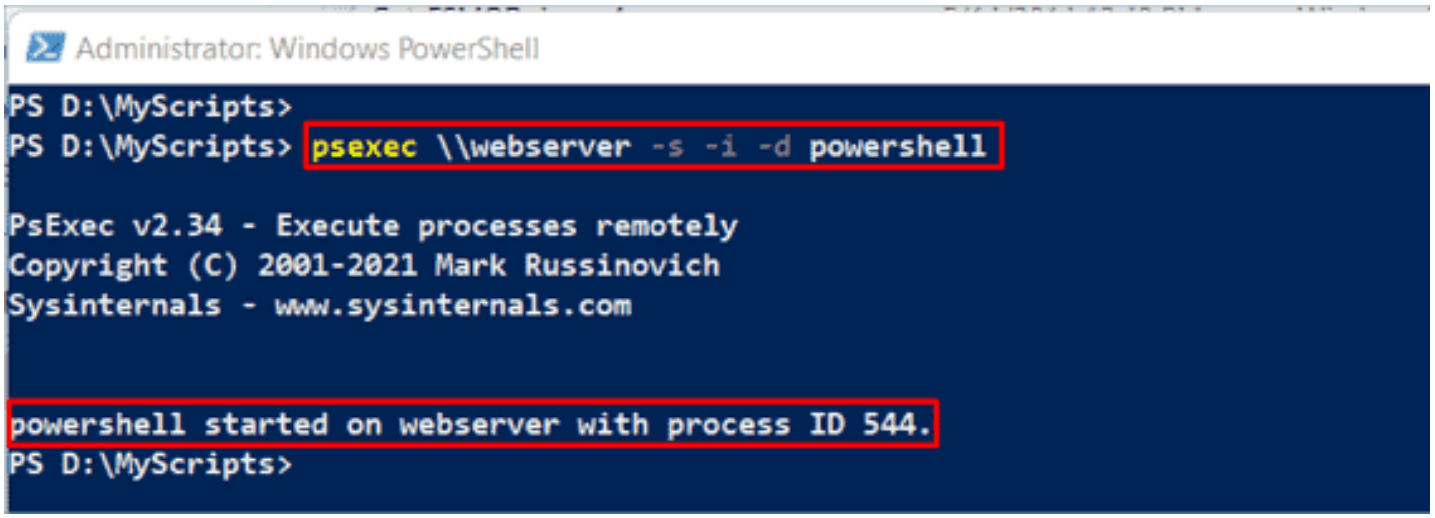
psexec -s -i -d regedit

This command returns the exit code, and you will get the command console control back immediately.

You can also use this technique to launch any GUI-based program interactively on a remote computer, as shown in the following command:

psexec \\webserver -s -i -d powershell

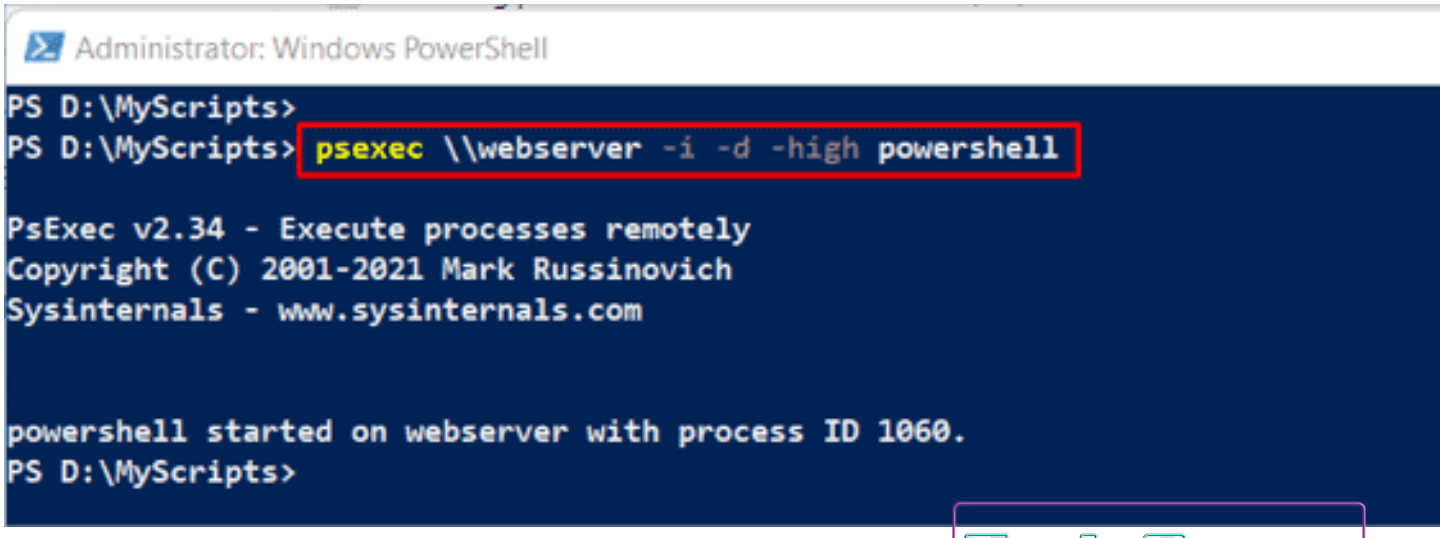
This command launches PowerShell in interactive mode on a remote webserver with the SYSTEM account privilege and returns the process ID.



Launch PowerShell in interactive mode on a remote computer using PsExec

PsExec even allows you to control the priority of the process that you're running with the help of the -low, -belownormal, -abovenormal, -high, or -realtime keywords. The following command launches the PowerShell process on a remote webserver with *a* priority of high.

psexec \\webserver -i -d -high powershell



Launch a process in interactive mode and high priority using PsExec

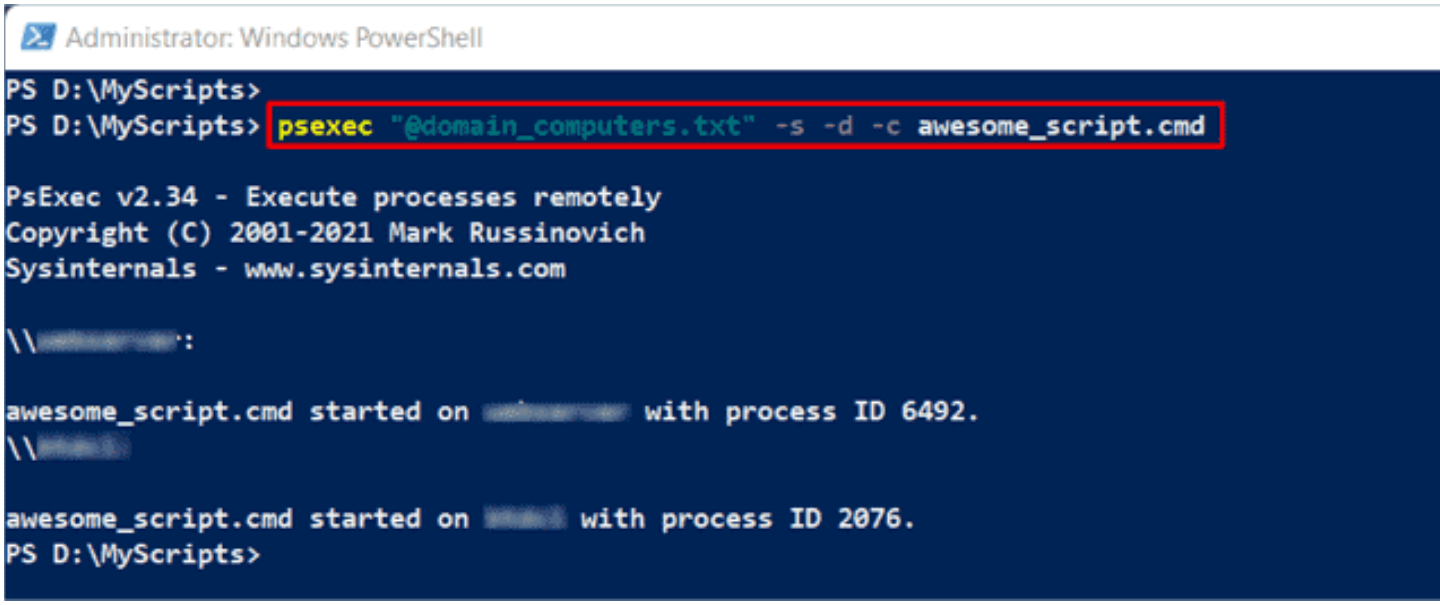
On Windows Vista, you can also use -background to run the process at low memory and I/O priority.

Copy a program to a remote computer

The most unique feature of PsExec is its ability to temporarily copy a program (or process) and then execute it on one or more remote computers. This is particularly useful when you have a program on your local computer, and you want to run it on the remote computer(s). All you need to do is use PsExec with the -c switch and specify the name of the program on the local computer. Have a look at the following command:

psexec @domain_computers.txt -s -d -c awesome_script.cmd

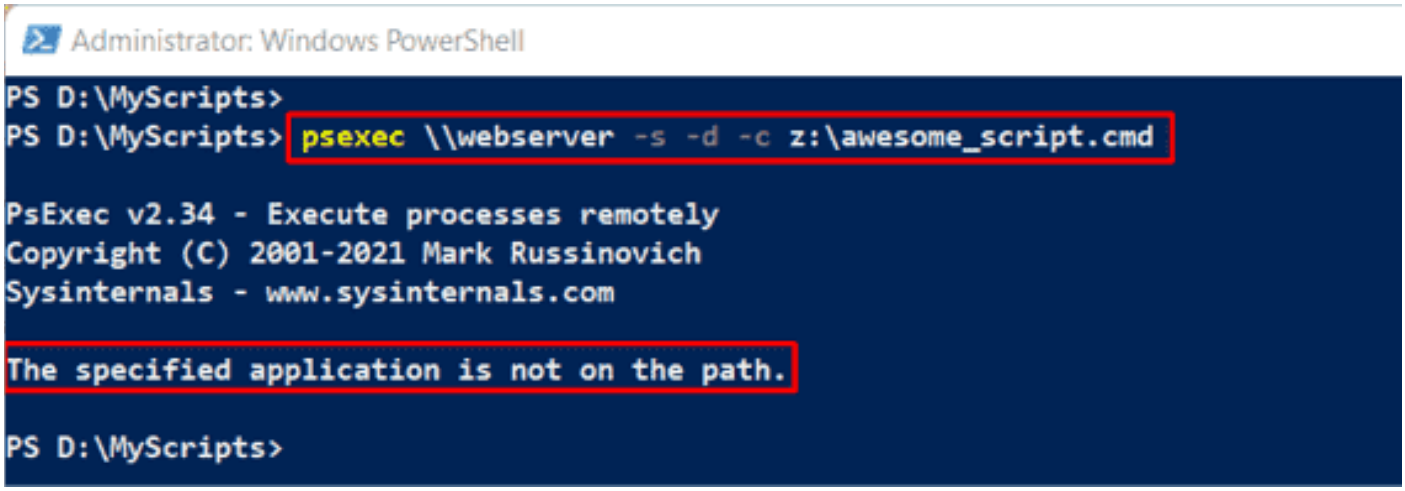
This command copies the awesome_script.cmd script and executes it on all the computers mentioned in the domain_computers.txt file using the SYSTEM account.



Copy a program to a remote computer using PsExec

The -c switch allows PsExec to temporarily copy the program from the local computer to the admin$ share (which is mapped to the Windows directory) on a remote computer, execute it, and then delete it as soon as the program finishes executing.

If you get an error that says, "The specified application is not on the path," as shown in the following screenshot, make sure the program or script you're trying to copy using the -c switch exists in your current working directory. In my case, the awesome_script.cmd must exist in the D:\MyScripts directory.



PsExec error: The specified application is not on the path

If the program or script already exists in the remote computer, but you still want to copy it anyway, use the -f switch. You can also use the -v switch to copy only the program if its version on the local computer is newer than on the remote computer.

Respecting UAC and Windows security

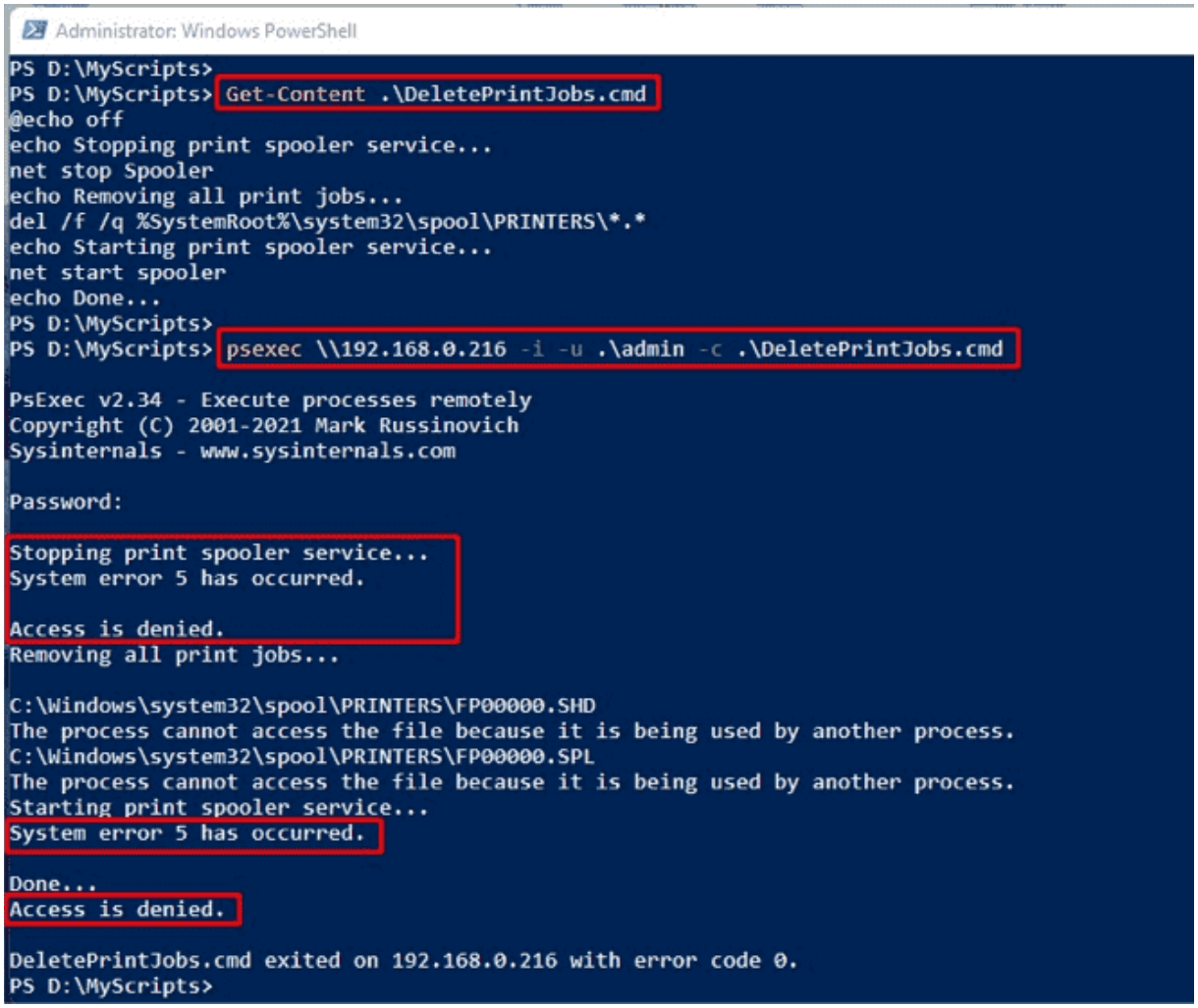
Windows Vista and above have an access control feature known as User Account Control (UAC), which allows programs and processes to always run in the security context of a non-administrator account unless the administrator specifically authorizes that program to run with elevated access to the system. This feature prevents malware and other potentially dangerous programs from making changes to your computer.

PsExec provides the -h switch, which allows the process to run with the account's elevated token if the target system supports it. With the help of this switch, admins can launch processes without having to worry about being blocked by UAC on a remote computer. Let's understand this with an example:

I have a batch script on my workstation that stops the print spooler service, deletes all the files from the **%SystemRoot%\system32\spool\printers** directory,

and then starts the print spooler service again. This is just a simple batch script; there's nothing fancy here. When I get a call from any user complaining about stuck print jobs, all I need to do is run this script remotely on their PC with the help of PsExec. The following screenshot shows that when I try to run the script without the -h switch, it fails with an error stating: "Access is denied:"

psexec \\192.168.0.216 -i -u .\admin -c .\DeletePrintJobs.cmd

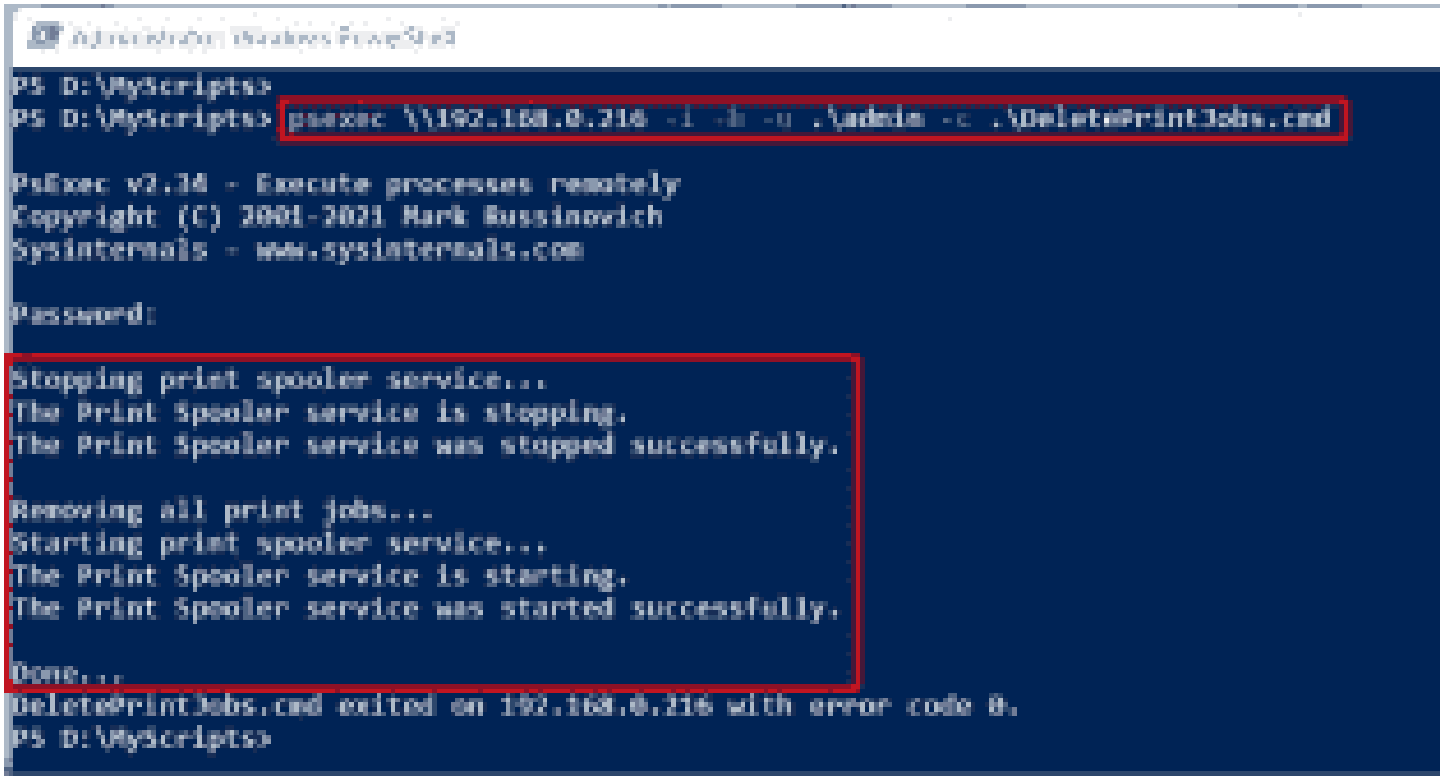


Running a script on a remote computer without an elevated security token

The script failed because UAC on the remote computer is restricting it from making changes to the computer, even though the alternate user specified is a member of the administrators group. This is the main purpose of UAC.

Now, let's see what happens when we add the -h switch to command:

psexec \\192.168.0.216 -i -h -u .\admin -c .\DeletePrintJobs.cmd



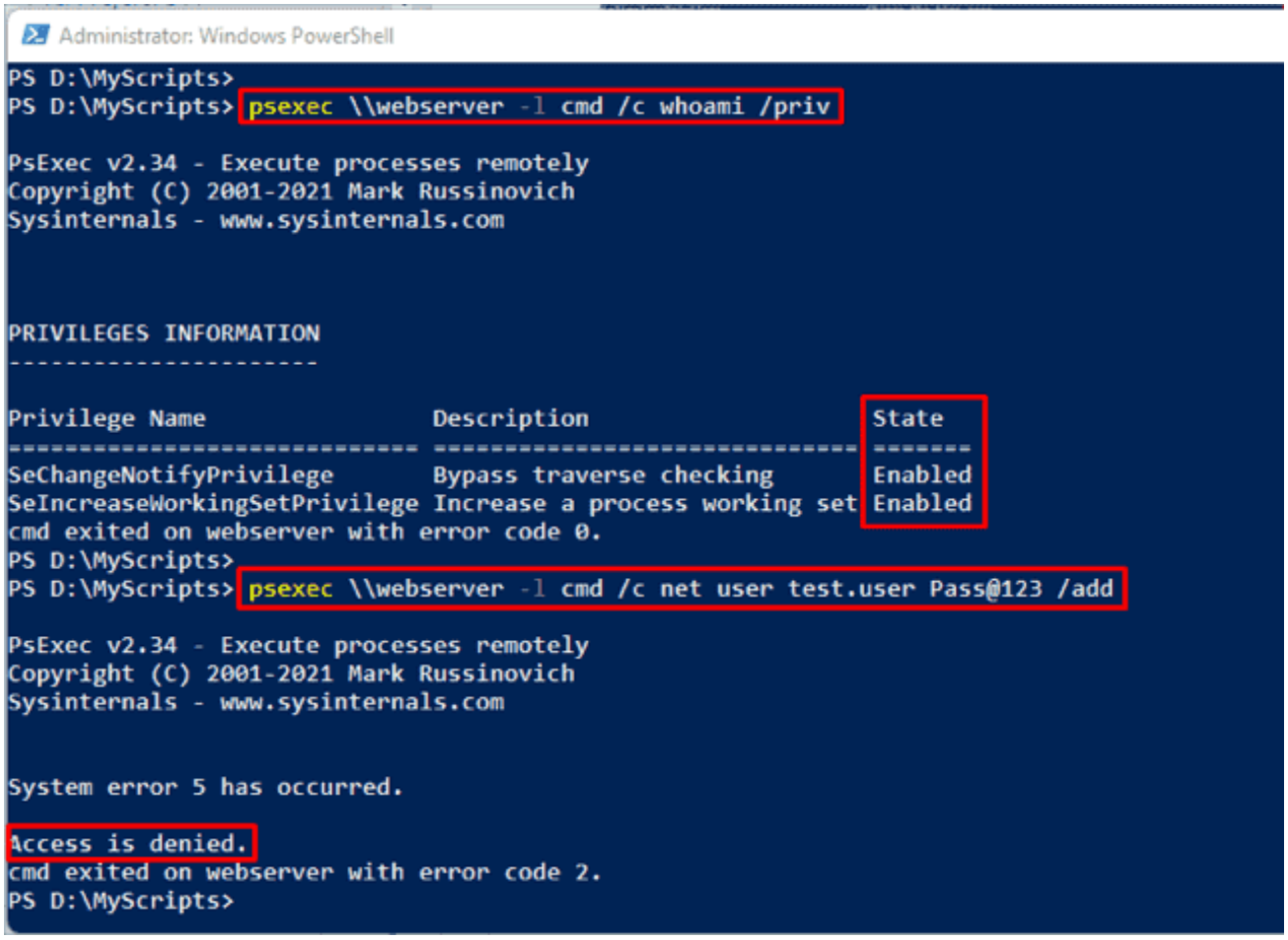
Running a script on a remote computer with an elevated security token

You can see that the script was executed successfully. The -h switch allowed the script to run with the user's elevated security token on the remote computer.

In the same way, if you think your current user account has too much privilege but you want to run the process on a remote computer with limited privileges, you can do so with the help of the -l switch. This allows PsExec to run the process as a limited user by stripping off the administrators group and allowing the privileges assigned to the Users group only. The following command shows how to run cmd.exe with the least privilege on a remote webserver:

```
psexec \\webserver -l cmd /c whoami /priv
```

This command launches **cmd** on the remote webserver with the least privilege. So, even though my currently logged-in user is a member of the domain admins AD group, it will still have restricted permissions on the remote computer within that process. Have a look at the following screenshot:



Running a process using limited privileges on the remote computer with PsExec

This screenshot shows that I had only a few privileges. When I tried creating a user, I got an Access Denied error since the **cmd** process was launched using the -l switch with PsExec. It is a great way to run business applications either on remote or local computers with the user's default security context.

Task Scheduler: Learn how to Analyze and Troubleshoot!

How to run PowerShell commands remotely with PsExec

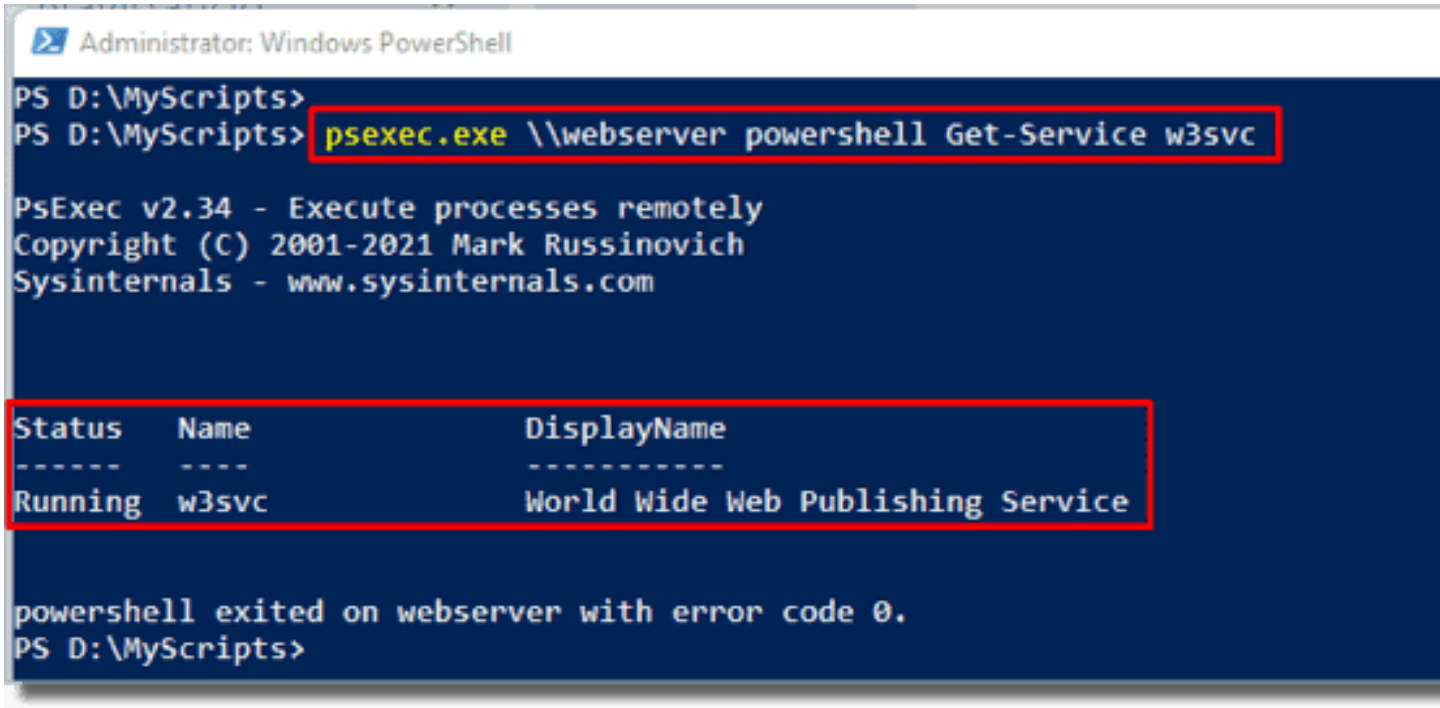
https://4sysops.com/archives/use-psexec-and-powershell-together/

First, let's discuss how to run PowerShell commands on remote computers with the help of PsExec. This ability of PsExec might not be useful for people working in domain environments since they can use PowerShell remoting. However, it is still worth discussing as not everyone works in a domain. There are many small organizations that are still operating in a workgroup environment.

To run a PowerShell command on a remote computer with PsExec, you need to use the -command parameter, as shown in the following example:

```
psexec.exe \\webserver Powershell -command Get-Service
w3svc
```

Since the -command parameter is the default for PowerShell, you can even skip typing it and just type the command itself, as shown in the following screenshot:



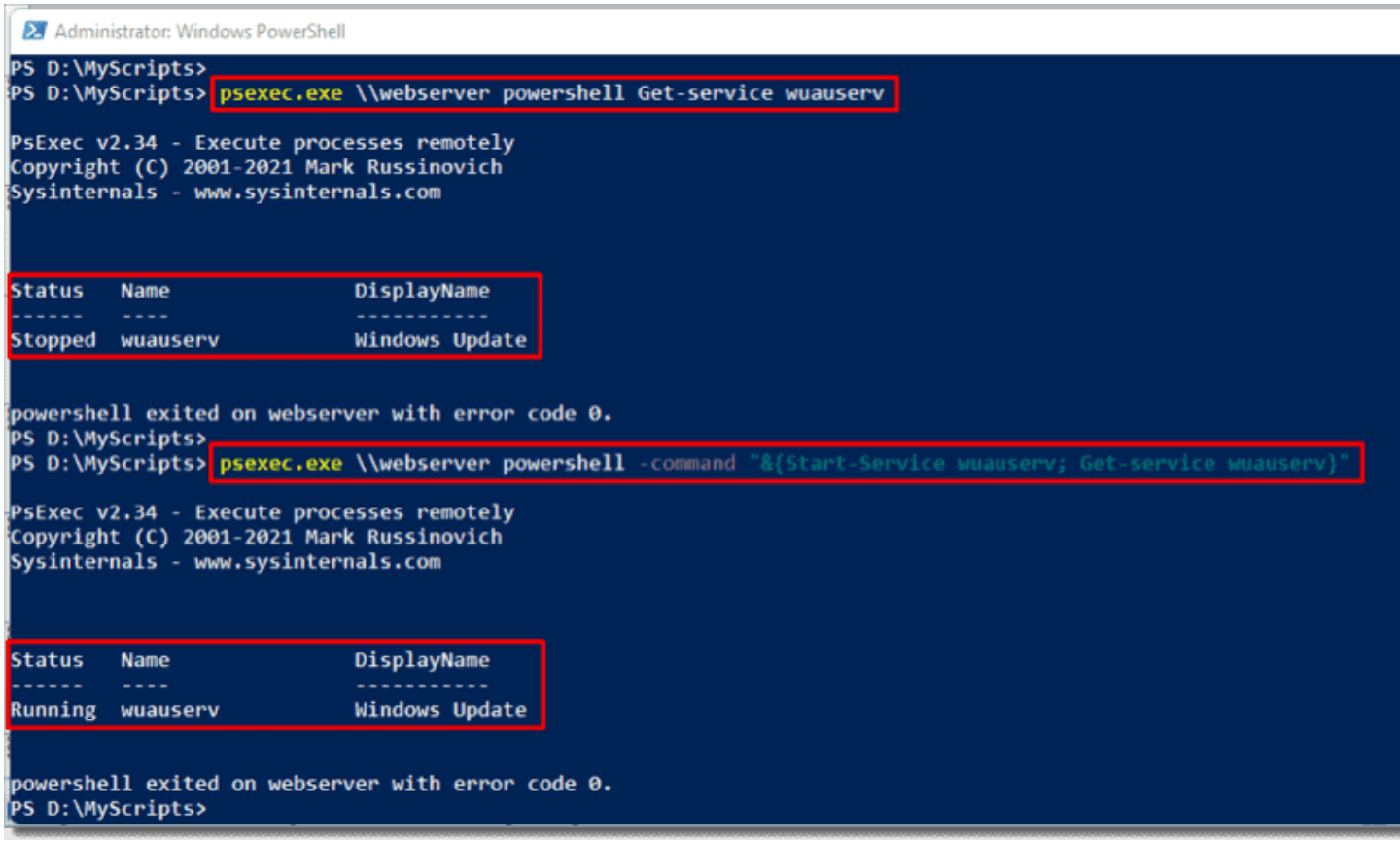
Running a single PowerShell command on a remote computer with PsExec

This command starts the PowerShell process on a remote web server. Then, the PowerShell process in turn runs the Get-Service command specified as a parameter. The command actually gets the status of the worldwide web publishing service on the remote webserver, and PsExec brings the result to your local computer.

There are situations in which you might want to run multiple PowerShell commands on remote computers. If you have even a small amount of experience with PowerShell, you probably know that a quick way to run multiple PowerShell commands is to separate them using a semicolon (;). Unfortunately, this will not work straight with PsExec, but we can use the call operator or ampersand (&) to do the job for us. The following example shows how to do this:

```
psexec.exe \\webserver powershell
"&{start-service wuauserv; get-service wuauserv}"
```

This command starts the Windows Update service on the remote webserver, and then shows the status of the service. The call operator (&) causes PowerShell to treat the string following it as a command. The commands will work if you omit the braces, but I prefer adding them for readability. Note that the call operator itself needs to be enclosed in quotes. The following screenshot shows the command in action:



Running multiple PowerShell commands on a remote computer with PsExec

```
PS C:\Users\Dutch> psexec \\192.168.0.126 -s -u administrator powershell -command Get-Service

PsExec v2.34 - Execute processes remotely
Copyright (C) 2001-2021 Mark Russinovich
Sysinternals - www.sysinternals.com

Password:


Status   Name               DisplayName
------   ----               -----------
Running  AarSvc_30b5a79     Agent Activation Runtime_30b5a79
Stopped  AarSvc_652c2       Agent Activation Runtime_652c2
Stopped  AJRouter           AllJoyn Router Service
Stopped  ALG                Application Layer Gateway Service
Stopped  AppIDSvc           Application Identity
Running  Appinfo            Application Information
Stopped  AppMgmt            Application Management
Stopped  AppReadiness       App Readiness
Stopped  AppVClient         Microsoft App-V Client
Running  AppXSvc            AppX Deployment Service (AppXSVC)
Stopped  aspnet_state       ASP.NET State Service
Stopped  AssignedAccessM... AssignedAccessManager Service
Running  AudioEndpointBu... Windows Audio Endpoint Builder
Running  Audiosrv           Windows Audio
```

In workgroup have to add the -s

Using this technique, you can run multiple PowerShell commands with PsExec on multiple remote computers.

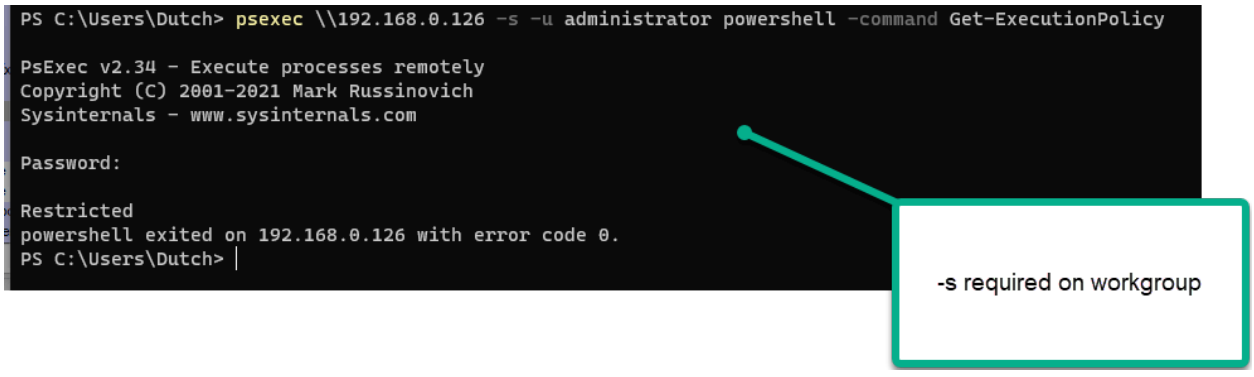
Run a PowerShell script remotely using PsExec ^

PowerShell remoting is great since it allows system admins to run commands on remote computers. But PsExec can help you take PowerShell remoting to the next level, since it enables you to run PowerShell scripts on multiple remote computers. The following command shows how to execute a PowerShell script on a remote computer:

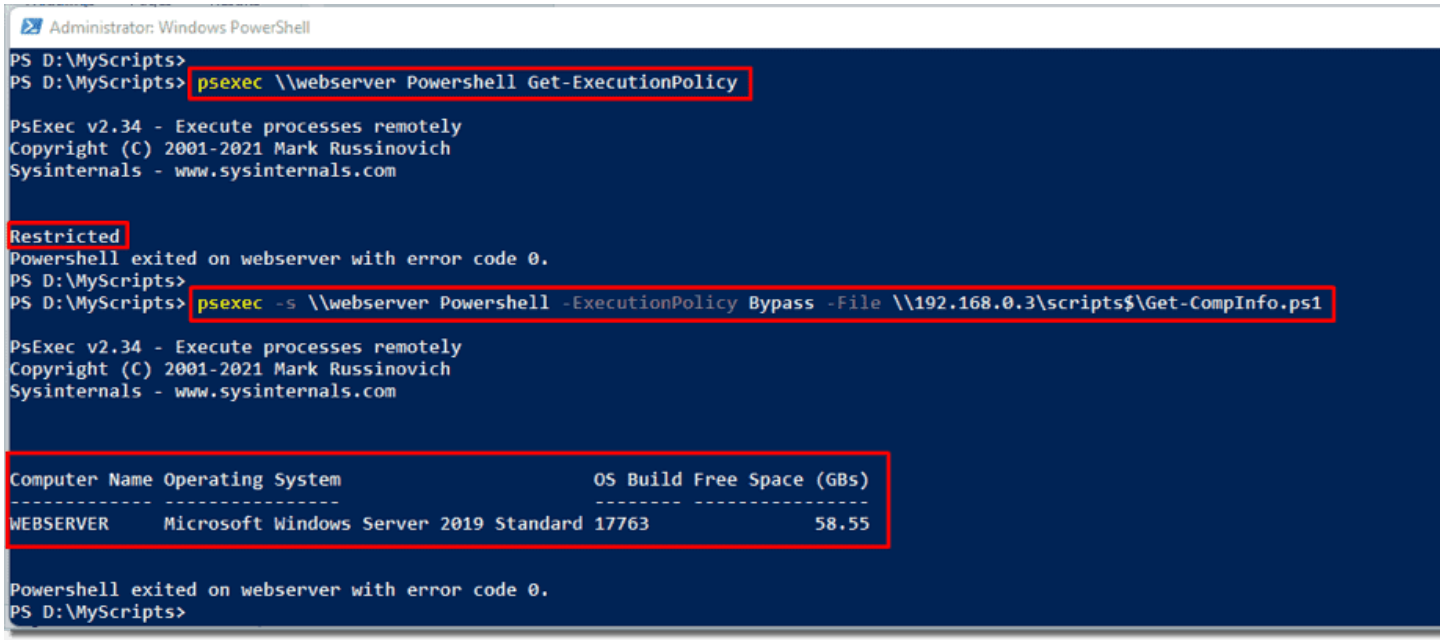
```
psexec -s \\webserver Powershell -ExecutionPolicy Bypass -File
\\192.168.0.3\scripts$\Get-CompInfo.ps1
```

This command executes a PowerShell script (Get-CompInfo.ps1) on a remote web server; the script itself is stored in a network share. The -ExecutionPolicy Bypass parameter allows the PowerShell script to execute even if the execution policy on that computer is set to **Restricted** mode. To run the script on multiple computers, you can either use a file (with entries in the format @computer.txt) or specify a list of comma-separated computer names, as discussed in the *How to use PsExec* guide.

A quick note on PowerShell execution policy—it is a security measure that controls how PowerShell scripts are handled by the computer. When in Restricted mode, it doesn't allow execution of any PowerShell scripts.

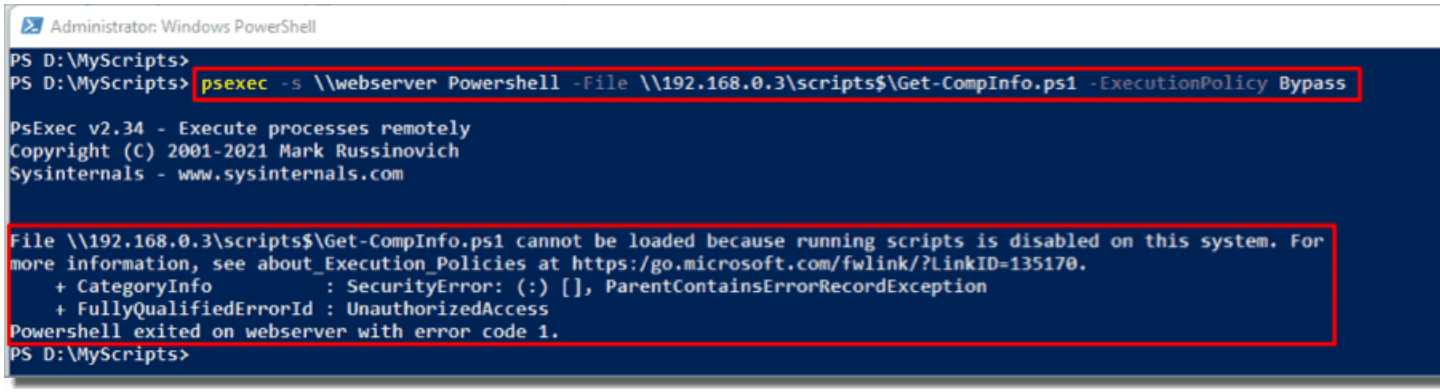


Add the -s in workgroup to execute PowerShell on remote hosts



Running a PowerShell script on a remote computer with PsExec

The screenshot shows that the current execution policy on the remote webserver is Restricted, but the script was still executed anyway. Note that the command needs to be run carefully; order matters here with regard to the parameters. If you specify the -File parameter first and the -ExecutionPolicy parameter later on, the command won't work. See the following screenshot for reference:



PowerShell error The file cannot be loaded because running scripts is disabled on this system

You will see a different error depending on the execution policy set on the remote computer. In Windows client operating systems (Windows 10, Windows 11, etc.), the default execution policy is **Restricted,** and in Windows Server operating systems (Server 2016, Server 2019, etc.), the default execution policy is **RemoteSigned**.

This error occurs because the PowerShell process (which is launched by PsExec on remote computers) complies with the execution policy before actually executing the script you specified using the -File parameter. Since you specified the script with the first parameter, it is blocked in the first place and the -ExecutionPolicy Bypass parameter isn't even considered. I hope this makes sense.

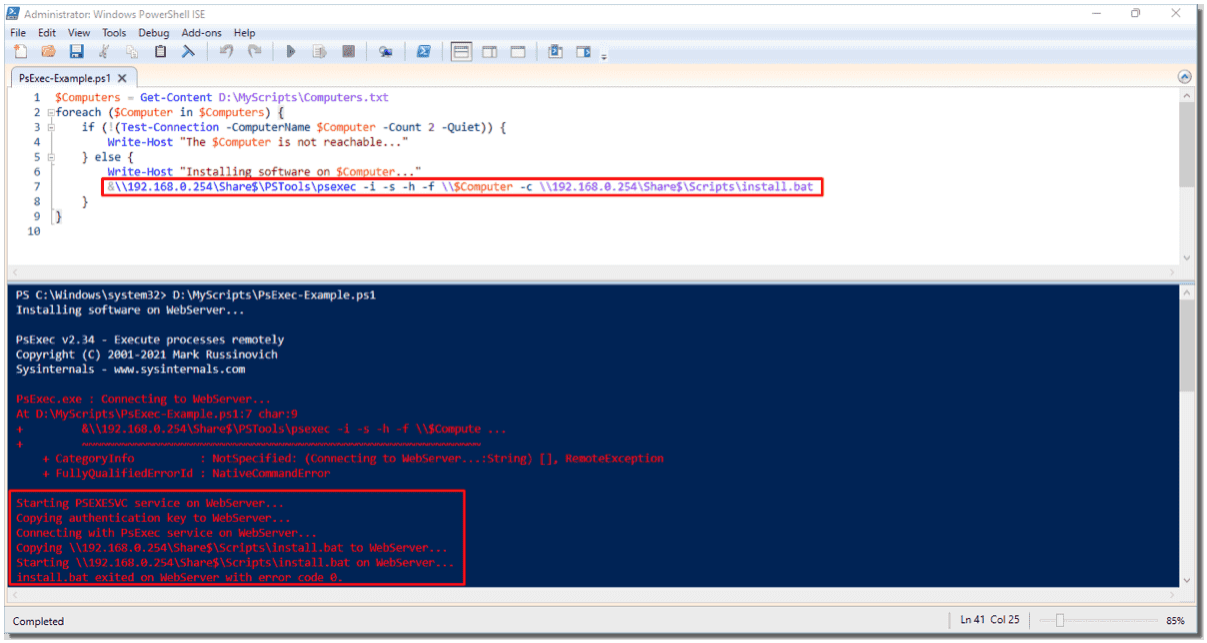
Run PsExec via PowerShell script ^

To use PsExec in a PowerShell script, we can again use the call operator or ampersand (&). See the following PowerShell script for an example:

```
$Computers = Get-Content D:\MyScripts\Computers.txt
foreach ($Computer in $Computers) {
if (!(Test-Connection -ComputerName $Computer -Count 2 -Quiet)) {
Write-Host "The $Computer is not reachable..."
} else {
Write-Host "Installing software on $Computer..."
&\\192.168.0.254\Share$\PSTools\psexec -s -h -f \\$Computer -c
\\192.168.0.254\Share$\Scripts\install.bat
}
}
```

This is a simple PowerShell script that I use to perform unattended deployment of a few applications in my environment. The interesting thing is in the line number 7. Look how the call operator or ampersand (&) character is used to call PsExec from a remote share. This technique is particularly useful if you don't want to install PsExec on all computers. If you have PsExec set up on all computers, you can specify the complete local path to the PsExec executable.

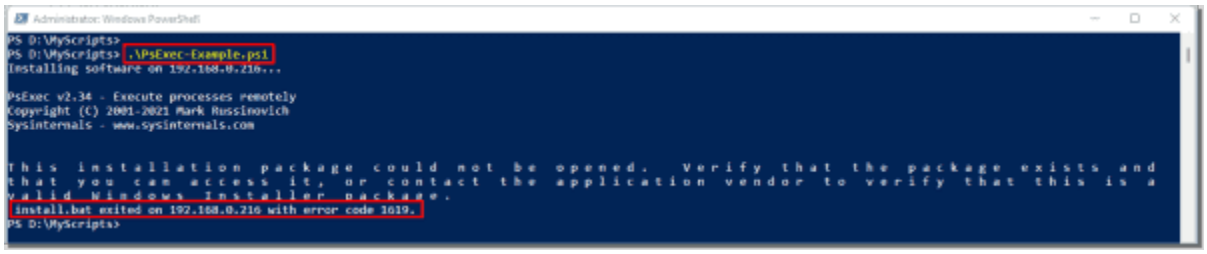
Have a look at the following screenshot:



Running PsExec within a PowerShell script on multiple remote computers

I intentionally executed this script in PowerShell ISE since it shows some detailed information (in the section marked in red) that you won't see in the regular PowerShell console.

If you get error code 1619 while doing a package installation using PsExec, make sure the installation package is accessible on the remote computer. See the following screenshot as an example:



This installation package could not be opened - error code 1619

The installation package might not be accessible by the user you specified in the script, or you haven't specified the alternate user at all. If you don't specify the alternate user with PsExec, your currently logged-in user is impersonated, and it won't have access to network resources. In this situation, you need to use the -u switch with PsExec to specify an alternate user that has sufficient access to the network share containing the installation package.

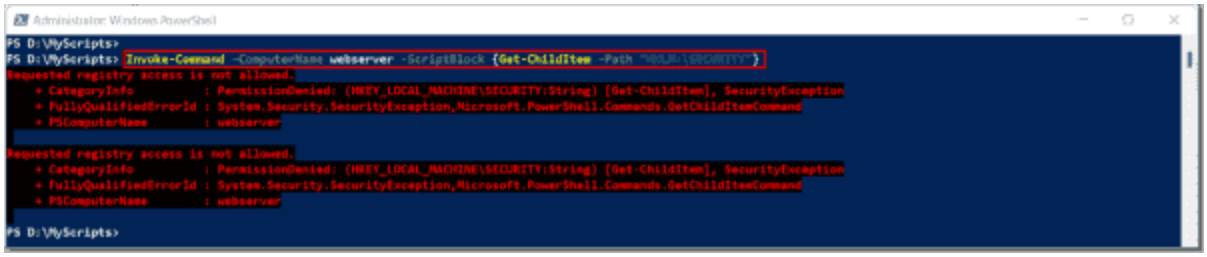
Usage examples with PsExec and PowerShell ^

In this section, I will cover a few use cases in which you can use PsExec and PowerShell together.

Access to highly restricted objects

We know that there are certain objects in Windows that are accessible to highly privileged users only (such as NT AUTHORITY\System). The HKEY_LOCAL_MACHINE\SYSTEM and HKEY_LOCAL_MACHINE\SAM subkeys are good examples of such objects. If you try to access these objects with PowerShell alone, you will get an error stating, "Requested registry access is not allowed," as shown in the screenshot.

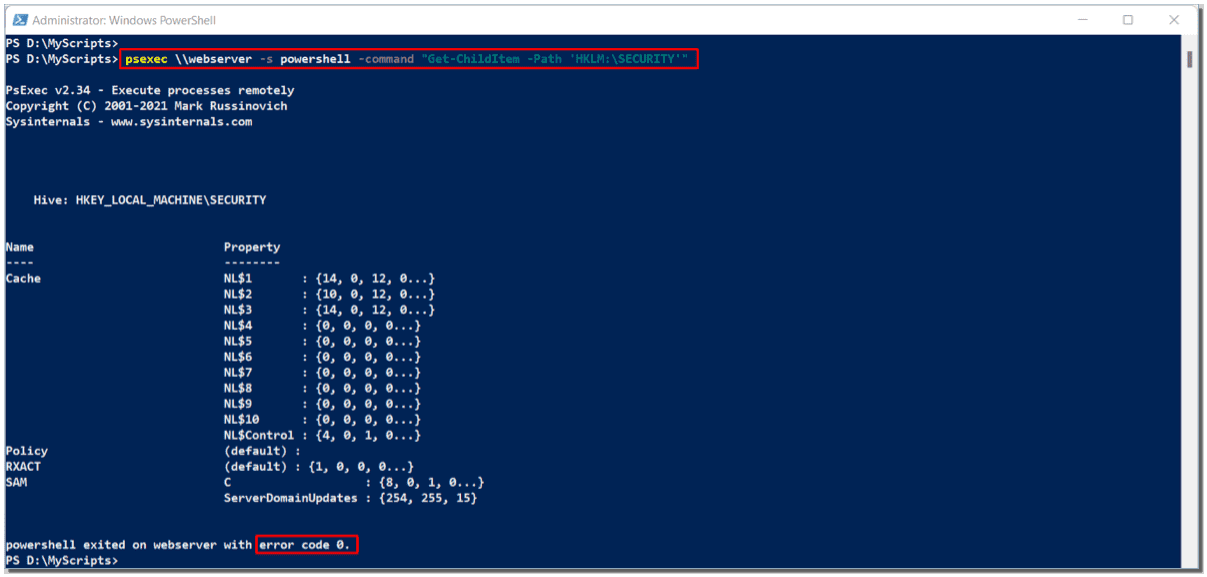
```
Invoke-Command -ComputerName webserver
-ScriptBlock {Get-ChildItem -Path
"HKLM:\SECURITY"}
```



Registry Error- Requested registry access is not allowed

To get around this access problem, you could use PsExec and PowerShell together, as shown in the following command:

```
psexec \\webserver -s powershell -command
"Get-ChildItem -Path 'HKLM:\SECURITY'"
```

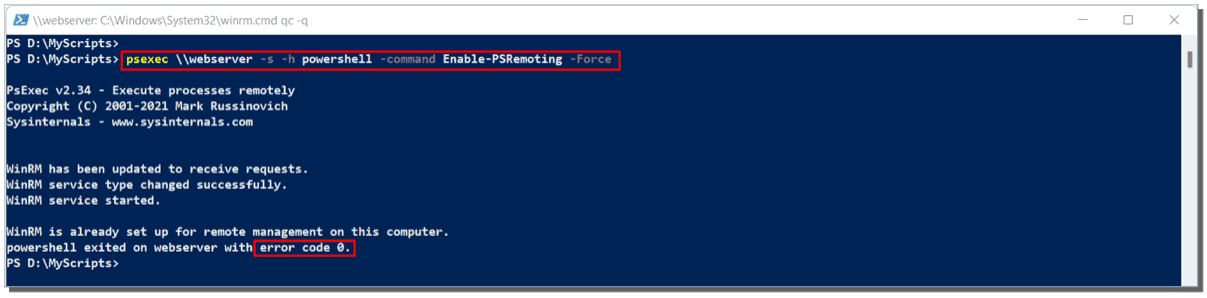


Accessing the restricted registry subkeys using the PsExec System switch

Enable PowerShell remoting on all computers

We all know that PowerShell remoting requires manual configuration on remote computer(s) before you can start using it. In these cases, PsExec proves to be really useful. The following command enables PowerShell remoting on all the computers using the domain_computers.txt file that we created earlier:

```
psexec @domain_computers.txt -s -h powershell -command
Enable-PSRemoting -Force
```



Enable PowerShell remoting using PsExec

Change the service start mode on all computers

Every organization has a different requirement when it comes to setting up Windows services. Some organizations prefer disabling unnecessary services and setting up the service startup mode. The following command is a mix of PsExec and PowerShell to set the startup mode of the Windows Update service to *auto* on all domain computers:

```
psexec "\\$((get-adcomputer -filter *).name -join ',')"
-s -h sc config wuauserv start=auto
```

Note that the command must be executed in the PowerShell console instead of the regular command prompt. The "\\$((get-adcomputer -filter *).name -join ',')" part uses PowerShell to dynamically build the computer names in a comma-separated list by pulling all computers from AD. It then feeds the list to PsExec to run whatever process you want to run.

Change the start mode of a service using PsExec and PowerShell on all computers

Conclusion ^

These are just a few examples of use cases where you can use PsExec and PowerShell together. Once you start using them yourself, you will realize that the potential is unlimited. You will be able to accomplish things that are hard to achieve with a single tool alone. You can remotely manage services, manage software, troubleshoot problems, perform security assessments, perform penetration testing, and much more.

Understanding and Troubleshooting Fiber-Optic Communication

Everything You Wanted to know About Psexec

Adam Bertram

Read more tutorials by Adam Bertram!

https://adamtheautomator.com/psexec/

If there was a command-line utility that could compete with robocopy in terms of usefulness, it's PsExec. The Sysinternals PsExec utility is as ubiquitous as they come in an IT admin arsenal. This tool allows administrators to remotely run commands just as if they were on the local computer.

To cover the PsExec tool in depth, it was fitting to cover this coveted tool in an ATA Ultimate Guide. In this guide, you will learn what psexec is, what it's capable of any many examples of how to use this useful tool.

What is PsExec.exe?

If you're new to IT or perhaps haven't had the need to run commands and tools on remote computers, you might not know what psexec is.

PsExec or *psexec.exe* is a command-line utility built for Windows. It allows administrators to run programs on local and more commonly remote computers. It is a free utility part of the Sysinternals pstools suite built by Mark Russinovich many years ago.

It was built to replace tools like telnet that forced you to open up ports and introduce security vulnerabilities. Nowadays, we have other options like PowerShell Remoting and the `Invoke-Command` PowerShell cmdlet but PsExec still has its place.

PsExec allows full interactivity for console application without having to install any software. As you'll see through this Ultimate Guide, PsExec can launch interactive command prompts, run as local system on remote computers, run commands on multiple computers at once and more.

It supports all versions of Windows since Windows XP. That means, that yes, PsExec on Windows 10 is a thing too. It's a simple tool to run that works on nearly everything but don't confuse its simplicity with its capabilities!

Prerequisites

You simply need to be running a modern Windows operating system for PsExec to run on your local computer. However, you're going to want to run psexec against remote computers. To do that, you'll need to ensure a few items are in place.

If you don't have these items in place now or unsure, don't worry. In the next section, we'll cover how to write some PowerShell to test your remote computers.

- A modern Windows computer (local)
- *File and Printer Sharing* open (remote computer, TCP port 445)
- The *admin$* administrative share available (remote computer)
- You know a local account's credential (remote computer)

As of this writing, PsExec is at v2.2 and will be the version you'll be learning about in this article.

Installing PSexec (With Remote Computer Setup)

Technically, you don't *install* PsExec since it's just a command-line utility but close enough. Since no installation is necessary, you simply need to download and extract it from the PsTools zip file. PsExec isn't available as a standalone utility and is part of the PsTools suite of tools.

Downloading PSExec

You can either extract the ZIP file manually or here's a handy PowerShell snippet to download and extract PsExec from its pstools ZIP file. Note that this removes all of the other PsTools tools. Many are still handy but we're not going to cover those in this article.

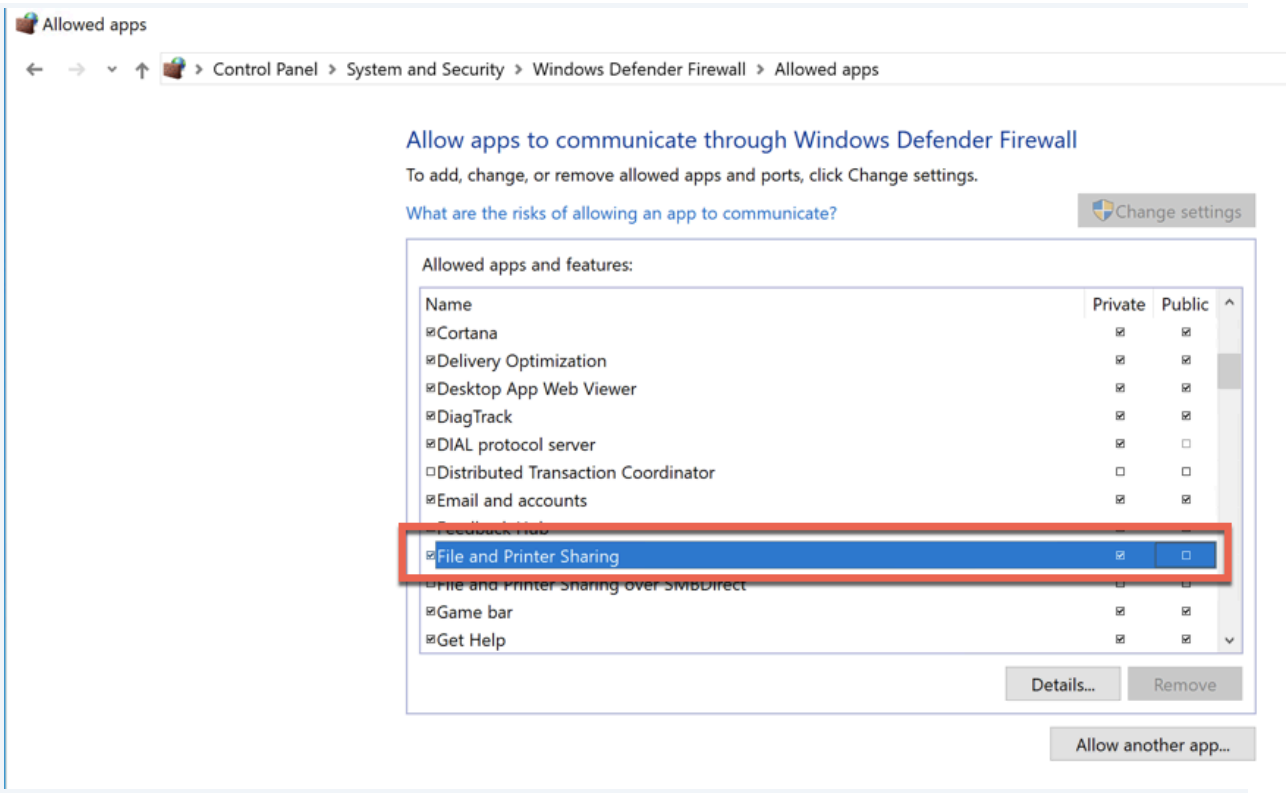
```
PS> Invoke-WebRequest -Uri
'https://download.sysinternals.com/files/PSTools.zip' -OutFile
'pstools.zip'
PS> Expand-Archive -Path 'pstools.zip' -DestinationPath
"$env:TEMP\pstools"
PS> Move-Item -Path "$env:TEMP\pstools\psexec.exe" .
PS> Remove-Item -Path "$env:TEMP\pstools" -Recurse
```

Remote Computer Configuration

Once you have PsExec downloaded, you'll then need to ensure any remote computer you're going to run it on is open. PsExec has simple requirements; *File and Printer Sharing* enabled and the *admin$* administrative share available.

You could go to all of the remote computers, open up the *Windows Firewall* applet, go to *Allowed Apps* and enable *File and Printer Sharing* on all computers as you see below.

Note that *File and Printer Sharing* is a known security risk so ensure only the *Private* firewall profile is enabled.



Allowing File and Print Sharing in the Windows Firewall

Or you could visit each computer and run the netsh utility to open it up via:

```
> netsh advfirewall firewall set rule group="File and Printer Sharing"
new enable=Yes
```

Or you could use PowerShell's **`Set-NetFirewallRule`** cmdlet to do it.

```
PS51> Set-NetFirewallRule -DisplayGroup "File And Printer Sharing"
-Enabled True -Profile Private
```

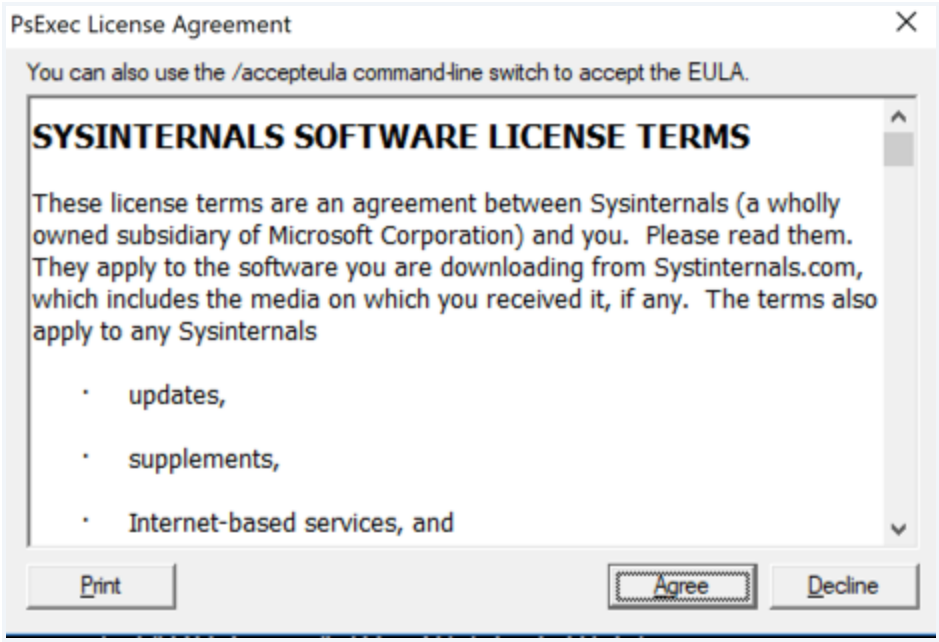
If you'd rather not visit each computer, you have PowerShell Remoting available and you're in an Active Directory domain, you could also open up the firewall on many computers at once using the **`Invoke-Command`** cmdlet.

```
PS51> Invoke-Command -ComputerName PC1, PC2, PC3 -ScriptBlock {
Set-NetFirewallRule -DisplayGroup "File And Printer Sharing" -Enabled
True -Profile Private }
```

Using PsExec

Before you can run, you need to walk. If you've never used PsExec before, you're in for a treat! Be sure to read this section first to get your feet wet to learn the basics before jumping in the deep end later in this article.

The first time you run PsExec on a new system, you will immediately see the PsExec license agreement come up. You'll have to click on the *Agree* button to begin using it.



PSExec license agreement (EULA)

If you'd like to prevent the license agreement from being displayed, you can silently accept it using the `/accepteula` switch as shown below.

```
> psexec /accepteula
```

You'll learn a few tricks at silencing this EULA popup on local and remote computers later in the article.

Running a Simple Remote Command

At its most basic, PsExec requires two parameters: a computer name and a command to run. If you have a command to run on the remote computer that doesn't require any arguments like `hostname`, you can simply add it after the computer name.

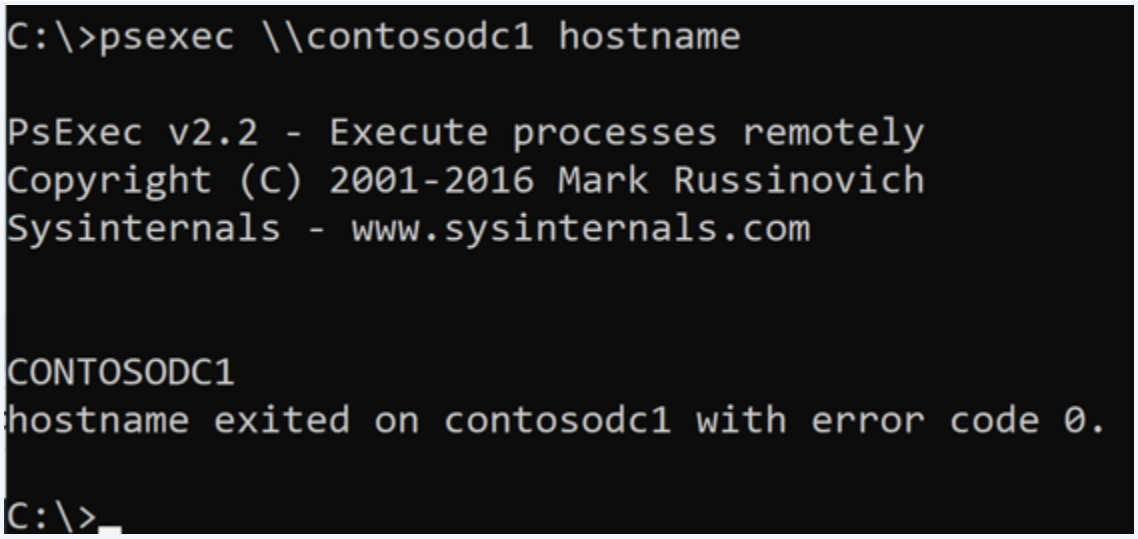
Note that if you don't specify a full file path, the command to run must be in the user or system path. Also, if you have a program with spaces in the name, you can always enclose the program in spaces such as *"my application.exe"*.

```
> psexec \\REMOTECOMPUTER hostname
```

You can see below that to execute the **hostname** command on the *CONTOSODC1* computer, you define it's UNC path followed by the command. PSExec will then connect to the remote computer securely, execute the command and return the output. In this case, the **hostname** command returned the hostname of the computer which is *CONTOSODC1*.

If the command isn't **cmd** or another console, PsExec will quickly exit the remote session and return the exit code the *remote* process returned.

Note: The error or exit code returned from psexec is not coming from PsExec itself. Instead, it's coming from the command that psexec executed on the remote computer.



Successful psexec remote command execution

How PsExec Works on Remote Computers

PsExec goes through a few steps to execute programs on remote computers.

1. Create a *PSEXESVC.exe* file in *C:\Windows*.

2. Create and start a Windows service on the remote computer called *PsExec*.
3. Execute the program under a parent process of *psexesvc.exe*.
4. When complete, the PsExec Windows service will be stopped and removed.

When the process doesn't work 100% correctly you may have to manually remove the service using the *sc* command.

Running a Simple Local Command

Even though PsExec is best known for running commands on remote computers, you can also run commands locally.

You can run commands locally by simply not providing a computer name like below.

```
> psexec <local command or EXE file>
```

Why would you do this? One reason would be to execute commands as the local *SYSTEM* account. You can use the `-s` switch to run any command as *SYSTEM* locally or remotely as you'll learn more about later.

Take a look at the short video below. Notice that you simply need to provide the `-s` switch along with the command interpreter executable for psexec to launch a new command session as *NT AUTHORITY\SYSTEM*.

Running Psexec as SYSTEM

PsExec Commands (Getting More Advanced)

Once you've got the basics down, you can then start learning more advanced techniques in psexec. PsExec can do a lot more than just run a single command on a single computer.

Running commands on multiple computers

PsExec isn't just limited to running commands on one remote computer at a time. This tool also has support to copy programs and run commands on multiple computers at once.

You can run PsExec on multiple computers at once a few different ways.

Comma-separated Computer Names

Typically when running a command on a single remote computer, you will specify a single computer name like `\\REMOTECOMPUTER`. You can also specify multiple computers separated by commas like below.

```
> psexec \\REMOTECOMPUTER1,REMOTECOMPUTER2,REMOTECOMPUTER3
```

All Computers in an Active Directory Domain

If you're running PsExec on an Active Directory domain-joined computer and you'd like to blast out a command execution on all computers in that domain, use a wildcard.

PsExec will search your entire Active Directory domain and attempt to run a command on every computer. Below is example syntax on how PsExec will attempt to connect to every computer on the domain the executing computer is a part of and run the `hostname` command.

```
> psexec \\* hostname

PsExec v2.2 - Execute processes remotely
Copyright (C) 2001-2016 Mark Russinovich
Sysinternals - www.sysinternals.com

Enumerating domain...
```

Note that if you use an asterisk to find all computers in a domain while the local computer is part of a workgroup, you will receive the error `A system error has occurred: 6118`.

Using a wildcard forces PsExec to essentially run the command `net view /all` to first find all computers in the domain. This is an outdated way to find computer information due to its dependency on NetBIOS.

Reading from a File

Another way you can run commands on multiple computers at once is to use a text file. Using the syntax `@<filename.txt>`, PsExec will read every line in the text file as if it were a computer name. It will then process each computer individually.

Below you can see an example of using PowerShell to create a text file of line-delimited computer names and using that as input for psexec.

```
PS51> (Get-AdComputer -Filter *).Name | Out-File computers.txt
PS51> psexec @computers.txt hostname
```

Copying local programs to the remote computer

Using the `-c` switch, psexec will copy any local program to the remote computer prior to execution.

Perhaps you have an EXE on your local computer in a *C:\Tools* folder and would like to run it on a remote computer. You can do so using the following syntax:

```
> psexec \\REMOTECOMPUTER -c C:\Tools\program.exe
```

When you use the `-c` switch and don't specify an executable file, PsExec will still copy the file but you'll receive an error stating *system cannot find the file specified.* This happens because PsExec will always attempt to run the file you copy.

If you need to copy files to remote computers prior to using PsExec, use the **`Copy-Item`** PowerShell cmdlet instead.

Running Remote Processes under Alternate Credentials

Another popular use case of PsExec is to run commands under alternative accounts. By default, PsExec will attempt to connect to the remote computer under your currently-logged-in account. More specifically, it will *impersonate* your account on the remote computer.

Using the `-u` and optional `-p` switch allows you to connect to the remote computer with an alternative user account. PsExec will then encrypt both the username and password and send them to the remote computer for authentication.

For example, if you're in a workgroup, you'll always need to specify the username to authenticate to the remote computer as.

```
> psexec \\REMOTECOMPUTER hostname -u localadmin -p secret-p@$$word
```

If both computers are a member of Active Directory, be sure to preface the user account with the domain name.

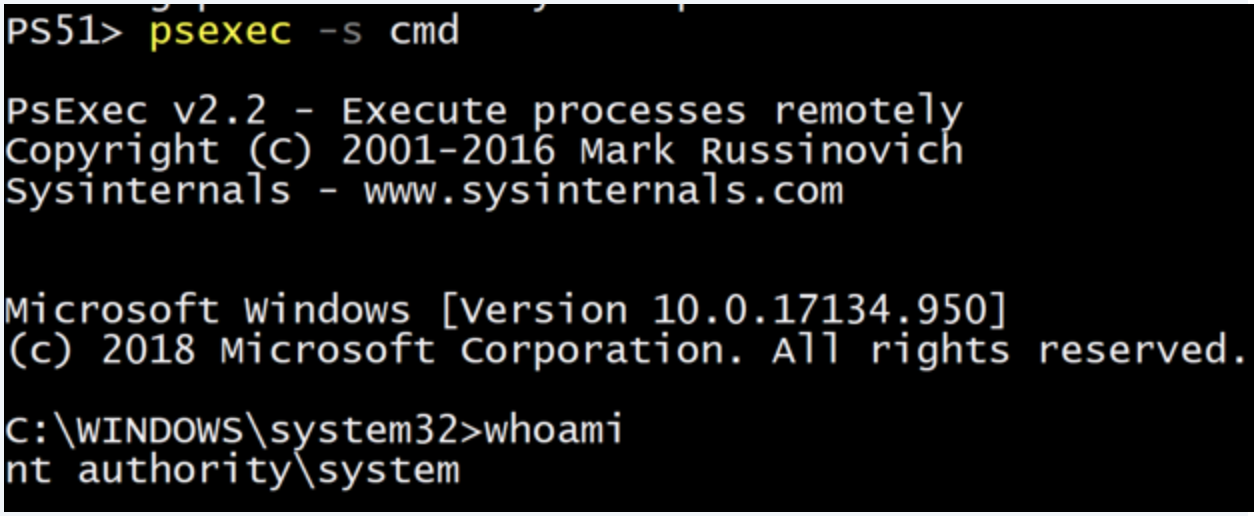
```
> psexec \\REMOTECOMPUTER hostname -u contoso.local\domainadmin -p
secret-p@$$word
```

Note that when you do not use the `-u` switch, psexec impersonates your logged-in account on the remote computer. It will not have access to any network resources.

Running Processes as the LOCAL SYSTEM Account

One of the most useful features of running PsExec under an alternative account is using the `-s` switch. This switch allows PsExec (and your remotely-executed application) to run under the remote (or local) computer's LOCAL SYSTEM account.

Notice below I didn't include a remote computer name. PsExec will just as gladly run on the local computer as well. In this instance, I'm using the `-s` option to tell PsExec to launch a local command prompt as the LOCAL SYSTEM account.



Running psexec as LOCAL SYSTEM

To run a command prompt as LOCAL SYSTEM on a remote computer, add the computer name to the reference like below:

```
> psexec -s \\REMOTECOMPUTER cmd
```

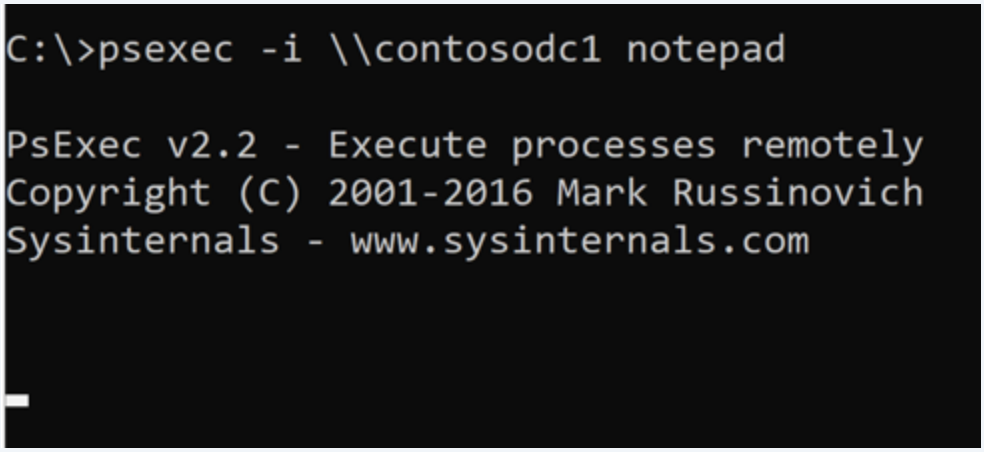
Launching GUI Applications Remotely

Another useful PsExec switch is `-i`. By default, PsExec does not allow the remotely-executed command to bring up any windows on the remote computer. This is helpful because if you're executing commands remotely, you're not going to see the screen anyway.

But perhaps you need to bring up programs for your users. You personally won't be using the application but an end-user will. In that case, use the `-i` switch.

Maybe you need to bring up a notepad window on a remote computer. Not a problem. Run *notepad.exe* with the `-i` switch and PsExec will open up Notepad.

```
> psexec -i \\REMOTECOMPUTER notepad
```



Running psexec in interactive mode

Be sure to also use the `-d` switch to disconnect when the interactive window is brought up though. By default, PsExec will wait for the process it executed to complete. If the remote process (Notepad in this case) is kept running, PsExec will never return control.

Using the `-d` switch with `-i` will tell PsExec to not wait for the remote process to finish. Instead, it will disconnect and return control to you as soon as the remote process is executed.

Redirecting Output

Psexec will rely any output sent from the remote process to your local session. Typically, this output will go directly to your local console. But if you'd like to redirect it, you can do so using typical redirection operators.

For example, if you'd like to run a command and silence all output, you could redirect output and errors to null using `^> nul ^2^&1`.

Note the special characters are escaped with a hat. (^).

PsExec Use Cases

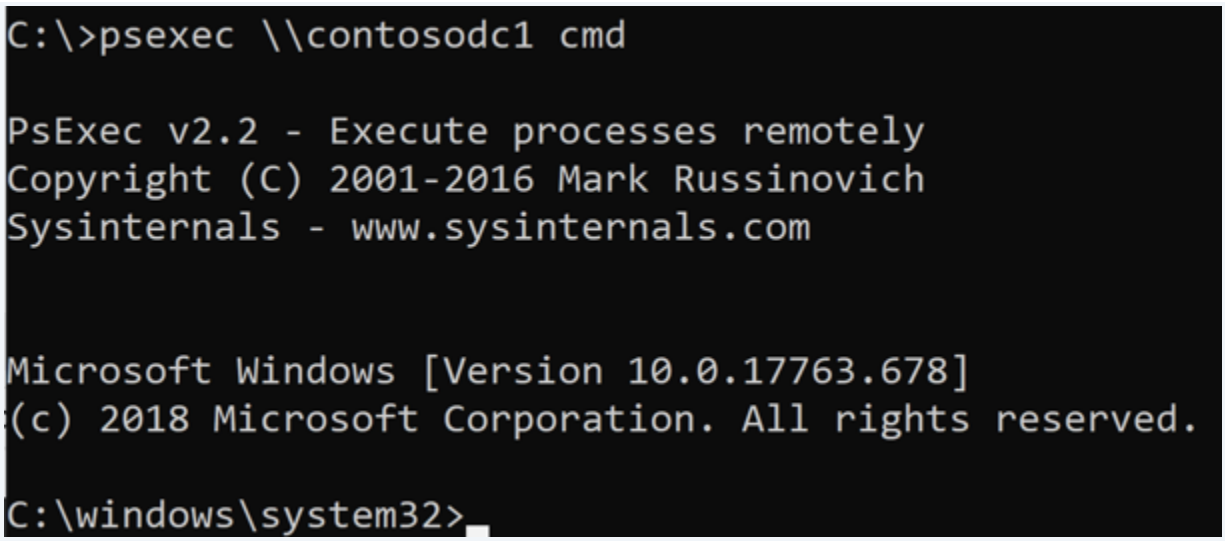
Once you've learned how to use psexec, you'll inevitably come across various specific use cases. In this section, you'll learn some real-world use cases and examples using psexec.

Launching a Remote Command Prompt (`psexec cmd`)

One of the most common use cases is launching PsExec as an interactive command prompt. PsExec doesn't just run commands remotely. It can also send command output back to your console. Because of this, it can make a great telnet (if anyone is still using that) or perhaps PowerShell **`Enter-PSSession`** replacement.

To launch a remote command, specify the remote computer name and run the *cmd* application. Cmd is the Windows command interpreter. Since PsExec supports interactive use, it will gladly return a flashing cursor and a prompt.

```
> psexec \\REMOTEPC cmd
```



Opening a command prompt on a remote computer

At this point, the world is your oyster. You can run commands on your local computer via this "nested" command prompt and they will be executed on the remote computer.

To exit from the command prompt, type **exit**. PsExec will stop the **cmd** process on the remote computer and return focus to the local computer.

Do NOT use Ctrl-C to close out of an interactive cmd session. Always use **exit**. If you use Ctrl-C, the psexec session will remain running on the remote computer.

Installing Software Remotely

You can use PsExec as a poor-man's software deployment tool. Perhaps you have an MSI installer that you need to run on one or more remote computers called *setup.msi*. This installer needs to be copied to the remote computers and then executed with the *msiexec.exe* utility with a few switches.

Below is an example of how you could use PsExec to remotely deploy software. This example copies *setup.msi* to the remote computer then launches the MSI installer interactively on as the *SYSTEM* account.

```
> psexec.exe \\REMOTECOMPUTER -i -s "msiexec.exe /i setup.msi" -c
setup.msi
```

Accepting the EULA without the `/accepteula` switch

As mentioned earlier, the first time PsExec runs, you'll have to accept a EULA. You could use the **/accepteula** switch but you could also "stage" it in the registry.

When launched for the first time, PsExec creates a registry key at *HKCU\Software\Sysinternals\PsExec*. Instead of that registry key, it creates a registry value called *EulaAccepted* with a DWORD value of *1*.

Using your favorite method to modify the registry on remote computers, you simply need to create this key/value on computers you'd like to run PsExec on. Once created, no need to run **/accepteula**!

Marrying PowerShell and PsExec

Before PowerShell, all we had was PsExec. Now, we have options. PowerShell can replace PsExec in many situations but complement it in others.

Building Computer Names with PowerShell

Instead of using `\\*` to find all computers in the domain, you can use PowerShell instead. By using PowerShell, you can not only pick certain computers but you don't have to use the firewall-prone **`net view /all`** behavior.

You can use PowerShell to create a string containing all computer names separated by a comma. You can then pass that string to PsExec which will merrily process each one like you typed each one manually.

You can see below an example of using the **`Get-AdComputer`** cmdlet part of the *ActiveDirectory* PowerShell module.

```
PS51> psexec "\\$((Get-AdComputer -Filter *).Name -join ',')" hostname
```

Enabling PowerShell Remoting Remotely

If you have remote computers you'd rather use PowerShell Remoting with instead of PsExec, you can use PsExec to enable them.

By running **`Enable-PSRemoting`** or the *winrm.cmd* batch file on remote computers, you can quickly turn on PowerShell Remoting across many computers at once.

Below you can see an example of calling the *winrm.cmd* batch file on a remote computer running as the SYSTEM account. Because the output from that command isn't needed, it's silenced with **`2>&1> $null`**.

```
$computerName = 'REMOTECOMPUTER'
```

```
psexec "\\$Computername" -s c:\windows\system32\winrm.cmd quickconfig
-quiet 2&>&1> $null
```

PsExec Error Messages

It's worth mentioning again upfront that most error codes you see returned from PsExec are from the remote process; not from PsExec. But it's helpful to have an understanding of these error codes and what they might mean.

If you'd like a reference on all Windows error codes, I recommend checking out this exhaustive list of Windows error codes.

Below is a list of common error codes you may see returned by PsExec.

ERROR CODE	EXPLANATION
-2146232576	Typically returned by Windows Update when an error occurs.
0	Command executed successfully
1	Incorrect function. A problem happened. That's about it.

ERROR CODE	EXPLANATION
1603	Fatal error during installation. This typically is returned by msiexec.
2	The system cannot find the file specified
4	The system cannot open the file.
5	Access is denied.
6	The handle is invalid.
6118	The list of servers for this workgroup is not currently available

Lateral movement: A deep look into PsExec

PsExec was originally conceived as a sysadmin tool. It quickly turned into the *de facto* standard for network pivoting. This article will explain how it works and give you the background to understand under which conditions it can be used.

https://www.contextis.com/en/blog/lateral-movement-a-deep-look-into-psexec

By Daniel Muñoz

Security Consultant

04 SEP 2018

Vulnerabilities And Exploits

Moving laterally during an engagement is a common practice which consists in accessing or controlling other systems on a network after compromising a machine. For this purpose, the ability to execute code on a remote host is really useful. There are many techniques available and the landscape keeps evolving as monitoring and defence mechanisms improve. Although you may be familiar with them or even used them quite often, there is a chance you are not entirely aware of the underlying mechanisms supporting it.

This is an in-depth analysis that will hopefully help having a clear picture of one of the best-known techniques, which has been around for long time now: PsExec.

What is PsExec?

As you probably know, PsExec is a tool included in the Sysinternals Suite created by Mark Russinovich. Originally, it was intended as a convenience tool for system administrators so they could perform maintenance tasks by running commands on remote hosts.

By providing the address of a target host, a valid user and a password, you can get control of a machine remotely.

What is happening under the hood? In a nutshell:

- Uploads PSEXESVC.exe to the $ADMIN shared folder.
- Remotely creates a service that will run PSEXESVC.exe.
- Remotely starts the service.

The PSEXECSVC service acts as a wrapper. It runs the specified executable (cmd.exe in our example) on the remote system (note that whatever you want to run must exist already on the remote system) while it redirects the input/output of the process back and forth between the hosts via named pipes. In other words, the PsExec tool running on the sysadmin´s computer writes to *client**pipe**PSEXESVC-stdin* and reads from *client**pipe**PSEXECSVC-stdout* and *client**pipe**PSEXECSVC-stderr*, while the process running on the remote machine has its output redirected to *client**pipe**PSEXECSVC-stdout* / *client**pipe**PSEXECSVC-stderr* and gets its input from *client**pipe**PSEXESVC-stdin*.
Let us see the process in detail:

1. Open an SMB session using the supplied credentials to authenticate. It doesn´t matter here if it uses NTLM or Kerberos.

2. Access the default shared folder $ADMIN (alias for C:\Windows) over SMB and upload an executable: PSEXESVC.exe.

3. Open handle to *client**pipe**svcctl* to talk to the Service Control Manager (SCM), which gives us the ability to create and start/stop services remotely, among other things. It uses the SVCCTL protocol which goes on top of DCE/RPC calls sent to the svcctl pipe, i.e., DCE/RPC goes on top of SMB.

4. Call the CreateService function, using the recently uploaded PSEXESVC.exe as service binary.

5. Call the StartService function.

As you can see in the following Wireshark capture, it creates the named pipes to redirect stdin, stdout, stderr.

In total, there are 4 named pipes created, one for the service itself, and the pipes to redirect the process´ stdin/stdout/stderr.

What is pass-the-hash?

Pass-the-hash is a technique which takes advantage of the constant fight between security and usability. Windows implements single-sign-on for users' convenience, without it, every time a user accessed, e.g., a network share, it would have to prompt him for the password. To be able to do this, the "key" must be stored in memory. At some point the designers must have thought "well, we don´t want to have cleartext passwords laying around in memory. Let´s just keep its hash". The problem then is that applications only rely on the password´s hash to authenticate against remote services. As a result, many authentication protocols in the Windows ecosystem end up using the NTLM hash of the password as the "key", instead of the password itself. The consequence is that the hash is often enough to authenticate.

In the first step described before, PsExec authenticates to SMB. Oddly enough, the authentication process itself only requires the user´s NTLM hash. If we are using NTLM authentication the hash will be used to encrypt the challenge or nonce. If it is Kerberos, we will be able to get a Service Ticket from the KDC only using the hash (pass-the-ticket). This means that remote code execution can be achieved without knowing the password itself.

Does PsExec pass the hash?

Sysinternal´s PsExec **does NOT pass the hash**. There are other tools, such as **Metasploit´s PsExec module**, which use the same technique and **do pass the hash**. Unfortunately, this has led to confusion and you´ll see misleading articles about this on the web.

If you do want to pass the hash using Sysinternal´s PsExec, from a windows box, you can do so with the aid of Mimikatz.

First, let´s lay out some facts:

- To be able to access resources locally, the Windows kernel checks if the user SID contained in the Process´ AccessToken has permission to access such resource.
- To access resources remotely, the credentials from the logonSession are used to authenticate with the remote system.
- AccessTokens are linked to a logonSession, specified in the AccessToken´s logon SID field.
- Mimikatz can create a forged logonSession using any user, domain and NTLM hash you provide it with.
- Mimikatz can make a process´ AccessToken point to a forged logonSession.
- **Sysinternal**´s PsExec will authenticate to a remote system using the credentials of the current process if no credentials are given as an argument.

Hence you can execute **Sysinternal**´s PsExec in such conditions that when it tries to authenticate, it will be using the stolen NTLM hash. In the picture below, you can see the steps taken:

- Spawn cmd.exe with cooked logonSession using Mimikatz

- Run PsExec within that cmd.exe to connect to the remote computer with the stolen hash

This is only one of many ways you could do this. If, instead of having GUI access to this pivoting machine, you only had a Meterpreter session you could use the Mimikatz module to spawn a hidden bogus process instead of cmd.exe and make your session impersonate its token using the Meterpreter´s steal_token command.

It's worth noting the following:

- In these examples it's being assumed that you already have a valid NTLM hash. Typically, to obtain them, you would need to have compromised a host and have a privileged account to extract the hashes. This would not be possible at all in newer versions of Windows 10 and Windows Server 2016 with isolated lsass (see references for more information).
- Again, even if you already had the NTLM hash, if you want to use Mimikatz to forge a logonSession to pass-the-hash, you'll need admin privileges on the attacker machine.

Minimum requirements to use the PsExec method

These are the bare minimum requirements:

1. Port 139 or 445 open on the remote machine, i.e., SMB.
2. Password or NTLM hash of the password (*)
3. Write permissions to a network shared folder (**). It doesn´t matter which one (***).
4. Permissions to create services on the remote machine: SC_MANAGER_CREATE_SERVICE (Access mask: 0x0002).
5. Ability to start the service created: SERVICE_QUERY_STATUS (Access mask: 0x0004) + SERVICE_START (Access mask: 0x0010).

Notes:

- *: Remember that NTLM ≠ NTLMv1/v2, so captured hashes over the network are not valid, unless you crack them or use an NTLM relay technique, but that´s another story.
- **: Bear in mind that NTFS permissions ≠ Share permissions. It´s not enough to have permissions to write locally.
- ***: Note that if you are using Sysinternal´s PsExec, it will try to copy PSEXESVC.exe to ADMIN$ (alias for C:\Windows), so you need access to it, not to just any folder.

In most scenarios, your stolen account won´t be able to comply with requirements 4 & 5 unless it is a privileged account (read the FAQ for more information on what type of privileged accounts would work). Mainly because, even if you had the ability to create services (SC_MANAGER_CREATE_SERVICE), the default DACL template (Discretionary Access Control List) will be applied to the service you just created. As you can imagine, this template won´t allow your user SERVICE_QUERY_STATUS + SERVICE_START access unless it belongs to an administrators group. In short: with a normal account you won´t have the permissions to start the service you just created, provided you could create one in the first place.

As a proof of concept, we are going to follow the steps manually. We are going to use Alice´s hash again. For the sake of simplicity in this scenario she belongs to the local ***Administrators*** group.

STEP 1: IMPERSONATE USER

Again, we impersonate the user.

STEP 2: UPLOAD THE EXECUTABLE

First, create a simple Meterpreter executable. Don´t forget to build it with the *exe-service* format.

Within the cmd.exe running with the forged token, copy the executable to the remote share.

STEP 3: START THE SERVICE

Start the service remotely

When the service starts, it launches a reverse shell back to our Kali box.

Differences with this approach:

- Note that you need to either know the local path of the network share or use the UNC path (as seen in the screenshot).
- The session is running as NTAUTHORITY\System. That is because, by default, all services run as SYSTEM, not as the user who created the service. Despite that, Sysinternals´PsExec runs as the user whose credentials you are using.
- In this case, the executable is a Meterpreter payload, not PSEXECSVC.exe. Thus, communication happens over TCP session, instead of over named pipes.

FAQ about token filtering and local users

Q: CAN I USE THIS METHOD AGAINST NON-DOMAIN-JOINED HOSTS?

A: Yes, but be aware that local users will fail trying to do so, even if they belong to the local “Administrators” group. The reason for this is that their AccessToken on the victim´s system will be filtered to a medium-integrity level by UAC and, as such it won´t be able to write to disk or to control the SC Manager, as those require a high-integrity token. This is an intentional security measure. There is an unadvisable workaround to prevent this which consists in editing/creating the registry

key **HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\LocalAccountTokenFilterPolicy** and set it to '1'.

On the other hand, this policy does not apply to the **built-in** local Administrator account (read, the one with RID 500), who is not enrolled in UAC by default. This account is disabled by default, but if you enable it you can test it yourself.

Q: WHAT ABOUT DOMAIN-JOINED HOSTS?

A: Local users belonging to the ***Administrators*** group will only get a medium-integrity token as well. Nonetheless, domain users belonging to the local ***Administrators*** group will receive a full-fledged high-integrity token and will be able to perform this technique.

TO WRAP THINGS UP:

Remember that you need to write to network share and control the SC Manager and a service. You could write to the network share with a medium-integrity token, but to create and start a service, you do need to be running with a high-integrity level. Regardless of whether the victim is part of a domain or not, local users will be constrained to a medium-integrity level, domain users (if that applies) and the built-in Administrator account will not.

	Admin-like permissions	Domain/Non-domain-joined	Will it work?	Why?
Domain member	Y	Domain	Yes	Not affected by token filtering.
Domain member	N	Domain	No	Not enough permissions.
Local user	Y	Doesn't matter	No	Token filtered down to medium-integrity level.
Built-in administrator user (RID 500)	Y (implicit)	Doesn't matter	Yes	Not affected by token filtering.

Reading list and references

Access tokens:

- Microsoft – Access tokens: https://msdn.microsoft.com/en-us/library/aa374909(v=vs.85).aspx.
- Book: Windows Internals – Part 1, Chapter 6, Security Descriptors and Access control (page 522 in the 6th edition). ISBN: 978-0-7356-4873-9

UAC Token filtering and remote access:

- https://support.microsoft.com/en-us/help/951016/description-of-user-account-control-and-remote-restrictions-in-windows

Service security and access rights:

- Microsoft – Service security and access rights: https://msdn.microsoft.com/en-us/library/windows/desktop/ms685981%28v=vs.85%29.aspx
- You can find the definition of the access rights masks for services either in the Windows winsvc.h header file or at https://rapid7.github.io/metasploit-framework/api/Msf/Exploit/Windows_Constants.html.

ACL and SDDL:

- Microsoft – Access Mask Format: https://msdn.microsoft.com/en-us/library/aa374896(v=vs.85).aspx.
- https://blogs.technet.microsoft.com/askds/2008/04/18/the-security-descriptor-definition-language-of-love-part-1/.
- https://blogs.technet.microsoft.com/askds/2008/05/07/the-security-descriptor-definition-language-of-love-part-2/.
- Book: Windows Internals – Part 1, Chapter 6, Security Descriptors and Access control (page 522 in the 6th edition). ISBN: 978-0-7356-4873-9

Windows security model:

- Book: Windows Internals – Part 1, Chapter 6 – Security. ISBN: 978-0-7356-4873-9
- Microsoft – Parts of the Access Control Model: https://msdn.microsoft.com/en-us/library/aa374862(v=vs.85).aspx.

Isolated Lsass, Secure Kernel Mode and Isolated User mode:

- Microsoft Technet Blog – Windows 10 Device Guard and Credential Guard demystified: https://blogs.technet.microsoft.com/ash/2016/03/02/windows-10-device-guard-and-credential-guard-demystified/
- Alex Ionescu (Black Hat USA 2015) - Battle Of The SKM And IUM: How Windows 10 Rewrites OS Architecture: https://www.youtube.com/watch?v=LqaWIn4y26E

Windows's Registry: Understand and Troubleshoot

How to: become the LOCAL SYSTEM account with PsExec

https://specopssoft.com/blog/how-to-become-the-local-system-account-with-psexec/

If you are an administrator using Specops Deploy, you may have had the following experience: an application can be deployed without any problems when you are trying it on your local machine but when you try to deploy it you can't seem to get it to work. This blog post might shed some light on your issue.

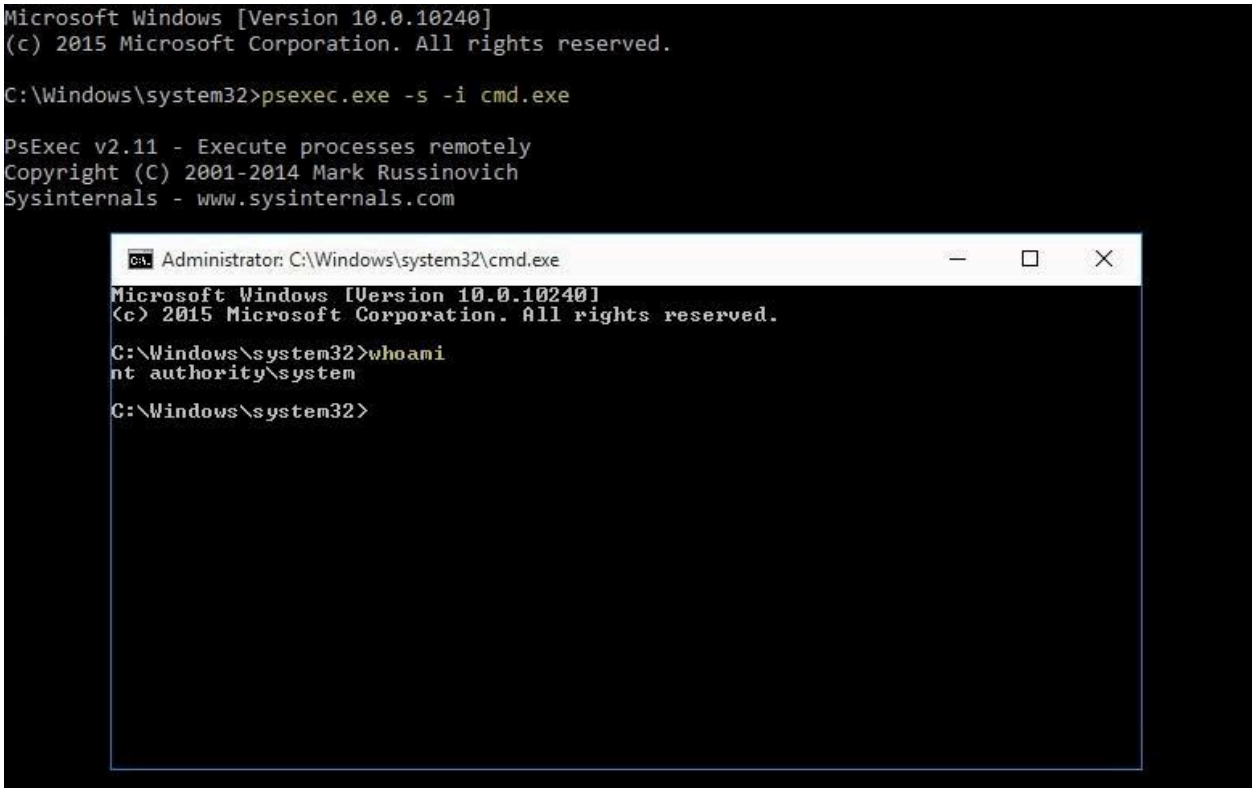
When an application is deployed using Specops Deploy, LOCAL SYSTEM is the account performing the installation. To ensure that your application is installed in the exact same way as Specops Deploy App, you need to act as the LOCAL SYSTEM account. To do this I recommend a little tool from Sysinternals called PsExec.exe

1. Download PsTools from https://download.sysinternals.com/files/PSTools.zip
2. Unzip the content and copy PsExec.exe to C:\Windows\System32
3. Open a Command Prompt as admin and enter the command below:

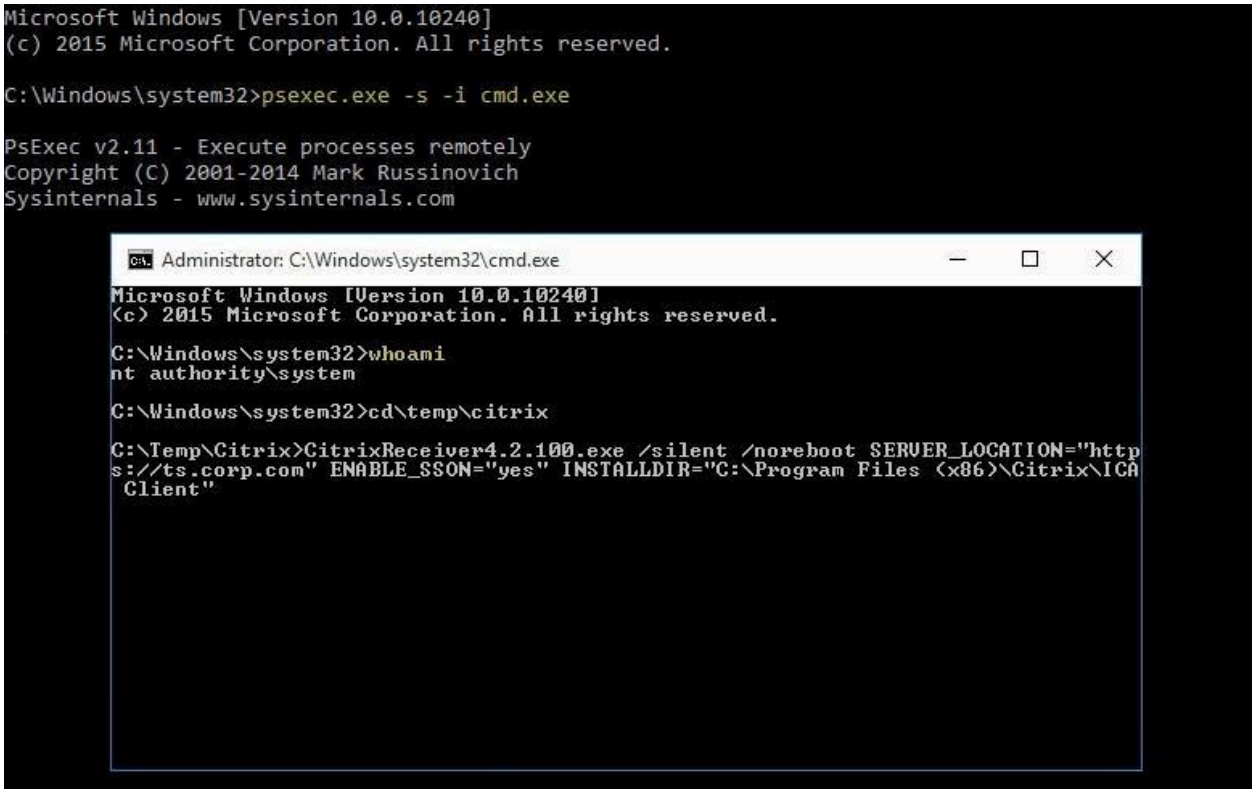
```
PsExec.exe -s -i cmd.exe
```

Copy

By using PsExec.exe you will open the new Command Prompt in the System Context and the account doing all the operations will be the LOCAL SYSTEM account. This is the same account Specops Deploy App uses when installing applications.



In the Command Prompt opened as LOCAL SYSTEM so that you can test your installations before you move them into Specops Deploy.



In other words, this is the perfect way to test your installations. There is no way to get closer to how Specops Deploy App is deploying software to the machines in your network.

Cool Command-line tools to use remotely

When you type **Net Accounts** and press Enter, you see the current settings for both automatically forcing a logoff and the system password requirements

```
C:\WINDOWS\system32>net accounts
Force user logoff how long after time expires?:       Never
Minimum password age (days):                          0
Maximum password age (days):                          42
Minimum password length:                              0
Length of password history maintained:                None
Lockout threshold:                                    Never
Lockout duration (minutes):                           30
Lockout observation window (minutes):                 30
Computer role:                                        WORKSTATION
The command completed successfully.
```

Need Local user account control?

The following procedure helps you construct a Net Accounts subcommand that configures passwords:

1. Type Net Accounts to start the subcommand.

2. (Optional) Type /MinPWLen:Length, where Length is the number of characters the user must supply for a password as a minimum. The default setting varies by Windows version. For example, Windows 7 has a minimum password length of 0. If you don't want to set a minimum password length, type /MinPWLen:0.

3. (Optional) Type /MaxPWAge:Days, where Days is the maximum number of days the user can use a password before changing it. If you don't want to set a maximum password age, type /MaxPWAge:Unlimited instead. The default setting is 42 days for Windows 7.

4. (Optional) Type /MinPWAge:Days, where Days is the minimum number of days that must pass before a user can change their password. This setting comes in handy if you have a user who changes their password often, but tends to forget the new password. Generally, more password changes are better, but too often can raise support costs. If you don't care how often the user changes their password, type /MinPWAge:0.

5. (Optional) Type /UniquePW:Number, where Number is the number of unique passwords that the user must provide before repeating an old password. This setting prevents users from recycling the same two passwords. The maximum number of passwords that Windows 7 will track is 24. If you don't care how often the user relies on the same password when changing passwords, type /UniquePW:0.

6. (Optional) Type /Domain if you want the command to modify the domain controller settings, rather than the local machine settings.

7. Press Enter. The Net Accounts subcommand makes the required changes. For example, if you want the user to change their password at least every 90 days but no more than once a month, use a password of 146 Chapter 8 3 Managing the Network seven characters, and provide at least five unique passwords, you'd type Net Accounts /MinPWLen:7 /MaxPWAge:90 /MinPWAge:30 / UniquePW:5 and press Enter.

Is someone connected via the network to the local PC?

View and Close Sessions

The Net Session subcommand makes it possible to view and control sessions on any computer. The sessions listed are those with external connections to the computer, not the local connection you're using. Consequently, unless you share resources on your workstation with others on the network, you're unlikely to see anything listed for a typical workstation—this subcommand is more applicable to serve following sections describe common tasks you perform using the Net Session subcommand.

View All Sessions

Before you view a specific session, you normally check which sessions are available by viewing all of the sessions. To view all of the sessions, `type Net Session` and press Enter.

```
C:\WINDOWS\system32>net sessions

Computer              User name           Client Type       Opens Idle time

-------------------------------------------------------------------------------
\\192.168.0.126       Administrator                            4 00:00:00
The command completed successfully.
```

You'll see an overview of all of the current sessions that include the computer name or TCP/IP address, the username, the client type, the number of open resources, and the system's idle time. The idle time tells you how much time has elapsed since the computer has performed any action, which you can use to determine when computers have crashed and are no longer connected to the server.

Deleting those sessions is normally safe after you verify that the user is no longer connected.

Delete a Session

Deleting a session essentially disconnects the user from the host machine. The most common reason for deleting a session is that the client machine has frozen or otherwise become unable to use the session. The server still has the session open because it doesn't know that the session is no longer in use. Use the Net Session ComputerName /Delete subcommand, where ComputerName is the name or the TCP/IP address of the computer you want to work with, to delete the session. For example, if you want to delete the session for the computer at TCP/IP address 192.168.0.244, you'd type

```
Net Session \\192.168.0.244 /Delete
```

and press Enter.

Need to view a local account?

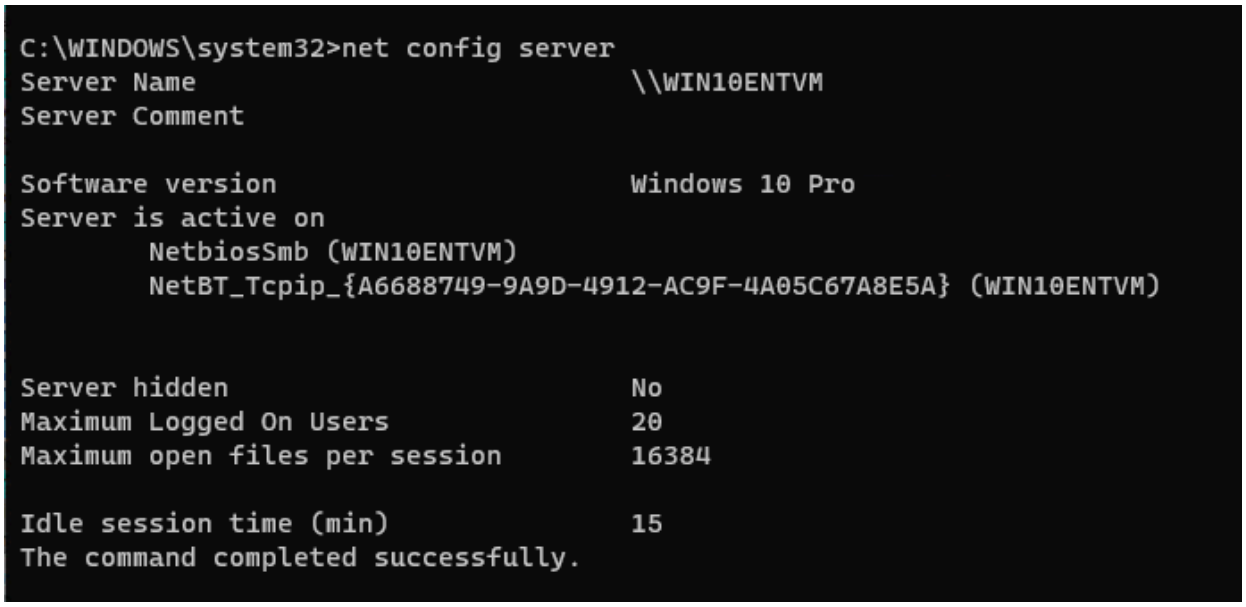
NET USER "administrator"

```
net user administrator
User name                    Administrator
Full Name
Comment                      Built-in account for administering the computer/domain
User's comment
Country/region code          000 (System Default)
Account active               Yes
Account expires              Never

Password last set            4/19/2022 6:50:40 AM
Password expires             Never
Password changeable          4/19/2022 6:50:40 AM
Password required            Yes
User may change password     Yes

Workstations allowed         All
Logon script
User profile
Home directory
Last logon                   4/20/2022 9:14:46 AM

Logon hours allowed          All

Local Group Memberships      *Administrators
Global Group memberships     *None
The command completed successfully.
```

View information on any PC's Server Service:

Perform Server Configuration

The `Net Config Server` subcommand provides the means to access the server part of any machine. The machine could be a workstation that shares resources or a full-fledged domain controller. As long as the machine is running the Server service, you can configure the server elements on it using this subcommand. In fact, you can see the current configuration by typing Net Config Server and pressing Enter. You'll see statistics like the ones shown in Figure



Remote removal, upgrade and install Firefox across a domain

In a previous post where I talked about How to silently and remotely remove Symantec Antivirus I mentioned how to use the pstools and msiexec combination

to achieve our goals. Firefox and other programs use a different installer called the Nullsoft Scriptable Install System which is open source and evolved from the winamp installer.

From this website which has information on most installation programs and how they can be remotely controlled I found the information to silently run the installer for Firefox. It can also work for other programs provided the packager did their homework. It was actually quite simple and just involved a single argument of "/S". Note that it is a capital S as the case sensitivity is important. There are a couple of way's that it can be done. For a single machine a remote installation or upgrade of Firefox is simply.

psexec -c -s computer_name firefox_setup_program.exe /S

This copies the program over to the remote computer and silently installs or upgrades Firefox. If you want to do a complete domain at once you can put the installation executable on a publicly accessible share and run it from there like this.

psexec -s * "share_serverpublic_sharefirefox_setup_program.exe" /S

Notice how I used the quotes. With both methods the psexec "-s" argument may or may not be needed. To uninstall Firefox for whatever reason (I can't think of any 🙂 you need to find out its uninstallation program in the same manner I described in How to silently and remotely remove Symantec Antivirus but to save some time it is generally "C:Program FilesMozilla Firefoxuninstallhelper.exe". Using the same "/S" switch it will silently uninstall Firefox. The complete command line will look like this.

psexec -s computer_name "C:Program FilesMozilla Firefoxuninstallhelper.exe" /S

Again if you wish to do a complete domain at once use an asterisk for the computer name and again be careful with the quoting. As windows likes to use spaces in its path names pstools will break the path into several arguments if it is not quoted. This methodology should work for other programs as well but it is

suggested to test them out on a test or virtual machine first before you let loose in the production environment.