**Introduction to Understanding Windows Operating System Basics**

The three most used endpoint operating systems are Windows, Linux, and Mac. When investigating security incidents, security analysts often encounter these operating systems running on servers or user end hosts. This section covers basic Windows operations principles.

Before an analyst investigates Windows security incidents, it is important that they understand some of the basic operations of the Windows operating system and how to examine the Windows event logs. The content in this section assumes that you already understand how to use various Windows operating systems, and introduces basic internal Windows operations and terminologies.

**Windows Operating System History**

Microsoft Windows has been a family of operating systems for PCs since 1985. In addition to Windows operating system for PCs, Microsoft offers operating systems for servers and personal mobile devices. Originally developed by Microsoft for IBM, MS-DOS was the standard operating system for IBM-compatible PCs. The latest Windows operating system is Windows 10 (the successor to Windows 8), which debuted on July 29, 2015.

It is interesting to note that although Microsoft operating systems have several vulnerabilities, they are not in the top three, as shown below.

| **Operating System** | **Vendor** | **Number of Vulnerabilities** |
| --- | --- | --- |
| Debian Linux | Debian | 3067 |
| Android | Google | 2563 |
| Linux Kernel | Linux | 2357 |
| Mac X | Apple | 2212 |
| Ubuntu Linux | Canonical | 2007 |
| IPhone | Apple | 1655 |
| Windows Server 2008 | Microsoft | 1421 |
| Windows 7 | Microsoft | 1283 |
| Windows 10 | Microsoft | 1111 |
| Windows Server 2012 | Microsoft | 105 |

CVE vulnerability data are taken from National Vulnerability Database (NVD) XML feeds provided by National Institute of Standards and Technology (NIST).

<https://www.cvedetails.com/top-50-products.php?year=0>

**Quiz**

**1.**

**Question 1**

**When you are researching a Windows operating system vulnerability (such as CVE-2016-7211), which organization can provide detailed information about the specific vulnerability?**

Institute of Electrical and Electronics Engineers (IEEE)

Control Objectives for Information and Related Technologies (COBIT)

National Institute of Standards and Technology (NIST)

International Organization for Standardization (ISO)

**2.**

**Question 2**

**Which is not a valid Windows operating system version?**

Windows 10

Windows 9

Windows 8.1

Windows 8

Windows 7

Windows Server 2012

Windows Vista

**Windows Operating System Architecture**

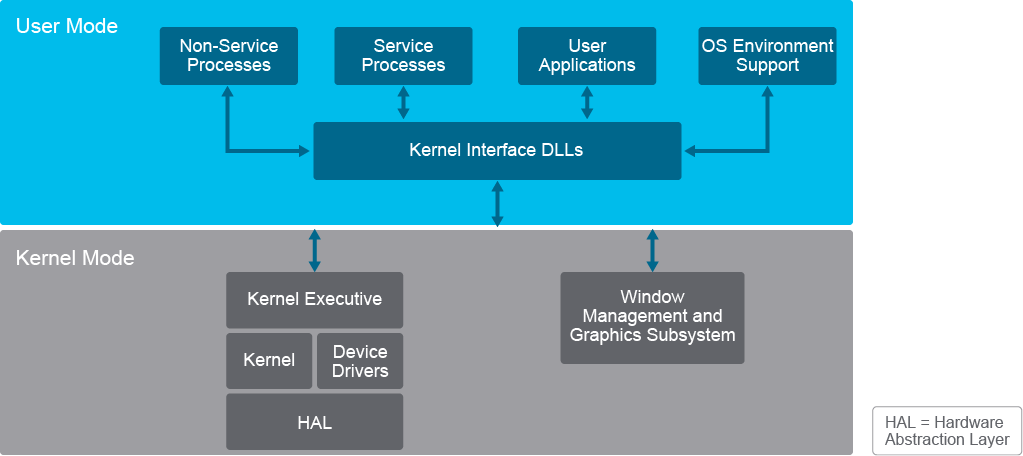
The Windows operating system has evolved into a robust enterprise computing platform that ranges in use from everyday end-user desktop environments to data center-scale server platforms. Its architecture has matured to include features that provide system stability and security. Although Windows is used most often as a desktop computing platform, its server platforms are used extensively in corporate data centers by IT professionals.

The figure illustrates the various architectural components of the Windows operating system, and outlines major processes that each component handles.

The Windows operating system operating system’s architecture is common across all its offerings and consists of two major components:

* **User mode:** The component that manages processes on behalf of system users. It runs with lower privilege so users can’t directly interfere with system level processes. It also provides communication pathways known as APIs so that user processes can request access to system level resources.
* **Kernel mode:** The component in which the core processes of the operating system run. Its processes run with the highest level of privilege so it can manage system CPU and memory resources.

The list does not include all the elements for each component, but it does outline the major processes and which mode is responsible for each. One of the concepts of the Windows architecture is how processes are treated relative to the mode in which they reside. For example, user processes appear to occupy their own distinct spaces, which roughly equate to the memory resources that the operating system dedicates to each one, preventing a user mode process from accessing the memory space of another process. This mechanism enhances the security and stability of the system.



Kernel mode processes also operate within the same memory space. However, kernel mode processes are higher privileged functions that cannot be directly accessed by lower privileged user mode processes. Executing processes in the same memory space increases the overall efficiency of the system.

User mode and kernel mode processes are the Windows subsystems that form the core of the operating system. They are started at system initialization by the Session Manager (Smss.exe). The following registry key will identify the system’s operating mode HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\SubSystems.

**User Mode**

81User mode processes are specific processes that are initiated or owned by Windows users. User mode processes operate within the confines of their own memory space or spaces, which prevent one user’s process from accessing another user’s process space or resources. The system does provide ways for one user process to request resources from another: Specific API calls are tightly controlled by the system, which prevents a process from arbitrarily impinging on another user’s process space while reducing the potential of system corruption. Remember, user mode processes operate at lower system privileges than kernel mode processes to maintain the separation between the user and kernel mode spaces.

The types of processes that run in user mode can fall into one of four general categories:

* **Non-service processes:** Fixed processes that support user access to system resources such as login services and Session Manager. Non-service processes are not services in a strict sense because they are initiated by user activity and not started by the service control manager.
* **Service processes:** Services that run in support of user system requirements; they are not system services that run independent of users, and are generally initiated by the system through the service control manager. Some examples include the Task Scheduler and Print Spooler.
* **User applications:** As the name implies, processes that are needed to run applications that are initiated by users.
* **Operating system environment support:** Windows NT originally shipped with support for Windows, OS/2, and POSIX subsystems to support a Windows design goal of code portability. Recent versions of Windows have dropped support for OS/2 and POSIX in favor of an enhanced version of POSIX that is called Subsystem for UNIX-based Applications (SUA). The operating system environment support subsystem manages the subsystem components to use based on what is required by running applications.

User mode applications and services often need to utilize resources that are managed by kernel mode processes. For example, a user mode process may need to write to a file or initiate a network connection. To support user application requirements, Windows can allow a process to switch contexts from user to kernel mode. When the operation requiring the system level resource completes, the system will switch the context back to user mode. This function is handled by a kernel interface layer that manages context switches and will prevent user mode processes direct access to system level resources.

**Kernel Mode**

A large portion of the Windows operating system runs as kernel mode processes. Unlike user mode processes, where each process runs in its own protected memory space, kernel mode processes share the same memory space. Kernel mode processes provide speed and efficiency, along with the opportunity for a misbehaving process to potentially impact the entire system. To mitigate this possibility, Windows enforces kernel mode code signing, which means that drivers and critical system files must be signed by a cryptographic key from a public certification authority.

Major processes that run in kernel mode include the following:

* **Kernel executive:** Performs base operating system services such as I/O, networking, and memory management
* **Kernel:** Responsible for managing low-level system operations such as thread scheduling and managing system interrupts
* **Device drivers:** Handles the requests for I/O to and from connected hardware devices
* **Hardware abstraction layer (HAL):** A layer of operating system code that handles interaction between the kernel and hardware to account for differences between platforms such as motherboards. HAL allows the operating system to be insulated from the hardware so that the operating system itself does not need to adapt to hardware differences. Rather, the HAL handles low-level hardware contingencies by translating them to standard operating system functions.
* **Window management and graphics subsystem:** Implements the GUI and manages graphic functions for Window management and interface controls
* **Ntoskrnl.exe:** Windows executive and kernel
* **Hal.dll:** HAL module
* **Win32k.sys:** Kernel-mode device driver for managing Windows displays and screen output. Also collects input from devices such as the keyboard and mouse
* **Ntdll.dll:** System service dispatcher to executive
* **Kernel32.dll:** Kernel interface core Windows subsystem
* **Advapi32.dll:** Core Windows subsystem
* **User32.dll:** User-mode management subsystem
* **Gdi32.dll:** GUI management subsystem

**Quiz**

**1.**

**Question 1**

**Malicious Windows operating system codes that share a single virtual address space, and can manage the system CPU and memory resources directly are running in which mode?**

safe

user

kernel

privileged

**2.**

**Question 2**

**Which type of Windows processes operate within the confines of their own memory space?**

service

EXEC

user

kernel

privileged

**Question 3**

**When investigating Windows-based security incidents, which can cause the most damage to the integrity of the Windows operating system?**

malicious Windows drivers that run in kernel mode

malicious applications that run in kernel mode

malicious applications that run in safe mode

malicious rootkits that run in user mode

**Windows Processes, Threads, and Handles**

A Windows application consists of one or more processes. In the simplest terms, a process is an instance of an executing program.

One or more threads run in the context of the process. A thread is the basic unit the operating system allocates processor time to. A thread can execute any part of the process code, including parts currently being executed by another thread. Each process provides the resources to execute a program and contains the following characteristics:

* Runs in virtual address space
* Consists of executable code
* Opens handles to system objects
* A security context
* A unique process identifier
* Environment variables
* A priority class
* Minimum and maximum working set sizes
* At least one thread of execution

Each process is started with a single thread that is often referred to as the primary thread. However, the process can create additional threads from any of its existing threads.

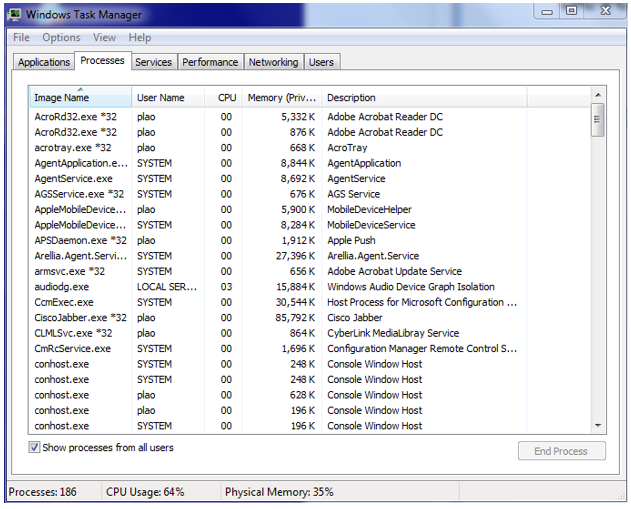
A thread is the entity within a process that can be scheduled for execution. All threads of a process share its virtual address space and system resources. In addition, each thread maintains exception handlers, a scheduling priority, thread local storage, a unique thread identifier, and a set of structures the system will use to save the thread context until it is scheduled. The thread context includes the thread's set of machine registers, the kernel stack, a thread environment block, and a user stack in the address space of the thread's process. Threads can also have their own security contexts, which can be used for impersonating clients. All threads within a process can access its virtual address space. In general, threads cannot access the memory space that belongs to another process, protecting the process from being corrupted by another process.

Microsoft Windows supports preemptive multitasking, which creates the effect of simultaneous execution of multiple threads from multiple processes. On a multiprocessor computer, the system can simultaneously execute as many threads as there are processors on the computer.

An object is a data structure that represents a system resource, such as a file, thread, or graphic image. An application cannot directly access object data or the system resource that an object represents. Instead, an application must obtain an object handle, which it can use to examine or modify the system resource. Each handle has an entry in an internally maintained table. These entries contain the addresses of the resources and the means to identify the resource type.

**Windows Task Manager**

The Windows Task Manager can be used to view the running applications and processes.



Here are two ways to start the Windows Task Manager:

* Right-click the taskbar and choose the Start Task Manager option.
* Run the **taskmgr** command from the Windows command line.

**1.**

**Question 1**

**Which Windows component is used by the applications to modify the system resources?**

process thread

task manager

Windows registry

object handle

**2.**

**Question 2**

**When investigating a malicious Windows application, which two Windows components that are associated with the application will also need to be investigated? (Choose two.)**

threads

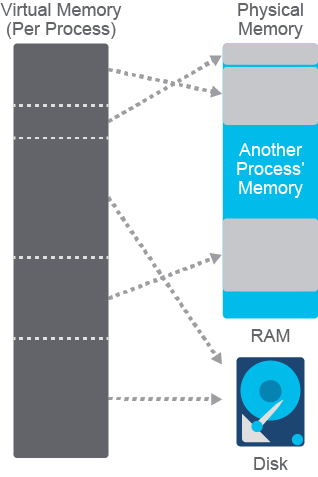
Correct

DLLs

processes

**Windows Virtual Memory Address Space**

In modern computing systems, temporary data instructions that are processed by the CPU are stored in the computer’s RAM. Virtual memory is a memory management technique that helps the system execute programs when the environment is running low on RAM memory. Virtual memory temporarily transfers content directly from RAM into the disk storage system. This process helps to compensate for a shortage of RAM.



A process is allocated specific memory addresses from the available virtual address space. The address space for each process is private and cannot be accessed by other processes unless it has permission to open the process for read or write access. All Windows applications and processes have to be granted specific access to various kernel system resources or objects, such as TCP ports and Windows sockets. By design, processes are not allowed direct access to kernel level functions. A process handle provides the process with access to a specific kernel level resource. Typically, the process has to release the handle for other processes to be able to access the same kernel level resource. However, if the process has kernel level access to a function and the handle has granted PROCESS\_VM\_READ (function) access, other processes can be granted read access to the same kernel level function’s memory space via the ReadProcessMemory function. In addition, if the process has kernel level access to a function and the handle has granted PROCESS\_VM\_WRITE (function) access, other processes can write to the same kernel level function’s memory space via the WriteProcessMemory function. It should be noted that processes need SeDebugPrivilege to open such handles. This potential violation of the process's memory privacy is very common when dealing with malware, and is a technique that is used to insert malicious code with the intent of gaining access to a system.

A virtual address space does not represent the actual physical location of an object in memory; instead, the system maintains a page table for each process. The page table is an internal data structure used to translate virtual addresses into corresponding physical addresses. Each time a thread references an address, the system translates the virtual address to a physical address.

Each process on 32-bit Microsoft Windows supports a virtual address space that enables addressing up to 4 gigabytes of memory. Each process on 64-bit Windows supports a virtual address space of 8 terabytes.

**1.**

**Question 1**

**Malware that can corrupt the virtual memory space of another process can alter the content of which two resources? (Choose two.)**

another process’s content stored in the RAM

another process’s content stored in the hard disk

another process’s content stored in the BIOS

another process’s content stored in the NVRAM

**2.**

**Question 2**

**What condition does the size of the total addressable virtual memory space depend on?**

if the Windows version is a server version

whether the Windows version is 32-bit or 64-bit

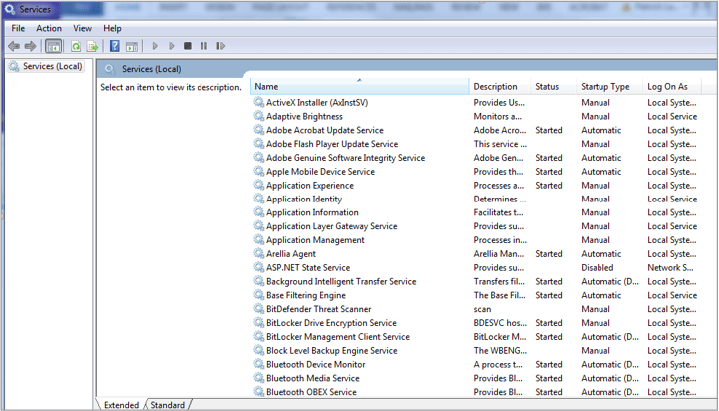
the drivers that are loaded with the Windows version

the physical size of the hard disk

the size of the RAM

**Windows Services**

Microsoft Windows services (formerly known as NT services) will create long-running executable applications that run in their own Windows sessions. These services can be automatically started when the computer boots. These services can be paused and restarted, and do not show any user interface. These features make services ideal for use on a server or whenever you need long-running functionality that does not interfere with other users who are working on the same computer.



Services Control Manager is used to start, stop, pause, resume, and configure the services.

To start Services Control Manager using the Windows interface, complete these steps:

1. Click the **Start** button.
2. Click **Control Panel.**
3. Click **Administrative Tools.**
4. Double-click **Services.**

To start the Services Control Manager by using a command line, open a Windows command prompt and enter the **services.msc** command.

**Question 1**

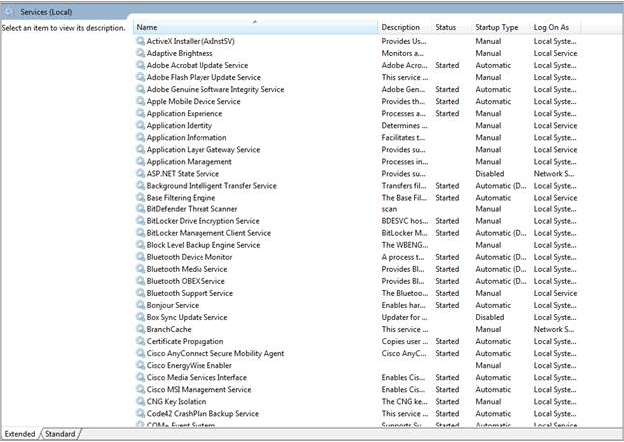
**You encountered malware that automatically runs upon bootup in its own Windows sessions, and without any user interface. Which Windows component can be used to configure the malware from starting automatically?**

startup manager

performance manager

services control manager

device manager



**Question 2**

**Referring to the following figure, which statement is correct?**

The Adobe Acrobat Update Service will automatically start during the boot and login process.

BitDefender Threat Scanner is running in the background.

Bluetooth Support Service has been disabled

Cisco AnyConnect Secure Mobility Agent was started manually.

**Windows File System Overview**

The Microsoft New Technology File System (NTFS) is predominantly utilized in most Windows installations today. A file system is a mechanism for organizing files on storage media. The type of media may influence which file system is appropriate to use. For example, optical media typically uses ISO 9660 or UDF. The focus of this section is on disk storage since the operating system and user data are typically stored on hard disk media.

The following file system options are supported in Windows installations:

* **File Allocation Table (FAT):** A general-purpose file system that is supported by Windows and other operating systems such as Mac OS X and Linux. Because of its simplicity and portability, it remains a popular file system. It is mostly used as the default format for USB-based storage media such as flash drives. However, it does have some limitations in terms of the file sizes it supports and the maximum partition size, so it is not widely used for hard drives. There are varieties of FAT known as FAT16 and FAT32. The FAT32 variant is the most commonly used because it is the least restrictive of the variants, less than FAT16.
* **exFAT:** The extended version of the FAT file system removes some of the file and partition size limitations of FAT, but is not widely supported on operating systems other than Windows and later versions of Mac OS X.
* **Hierarchical file system plus (HFS+):** The file system that is used on Mac OS X-based systems. HFS+ permits filenames up to 255 characters in length and supports larger file sizes and drive partitions than the original HFS. It is not supported on Windows platforms without the aid of additional software. However, a Windows system can read data from an HFS-formatted drive.
* **Extended file system (EXT):** The default file system on Linux-based installations. Its most current version is EXT4, which has the benefit of maintaining backward compatibility with the previous versions 2 and 3. It is not natively supported on Windows platforms, but a Windows system can read data from EXT-formatted drives with additional software.
* **Microsoft NTFS:** The most commonly used file system in Windows installations. It is natively supported across all versions of Windows and Linux operating systems. Mac OS X-based systems can read NTFS partitions but can only write to them by installing additional driver software.

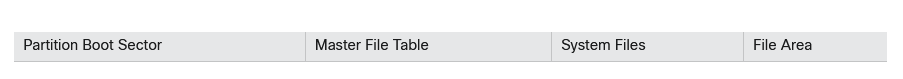
**NTFS Basics**

Microsoft NTFS provides many features that make it a good choice for both enterprise-scale systems and desktop systems. Some key features are:

* Performance and reliability
* Compatibility
* Large file and partition size
* Disk quotas
* Recovery features
* Security
* It also supports file system encryption to further secure data at rest on the storage media. NTFS supports data access control through security descriptors that include file ownership and permission characteristics for each file. This capability is one reason that NTFS is used so extensively throughout Windows deployments.

**NTFS Volume Structure**

Formatting a drive with the NTFS file system creates several system files, and the basic structures for reading and writing files and recording their locations.



The partition boot sector occupies the first 16 sectors of the drive. It contains information that points to the system bootstrap instructions and the location of the Master File Table (MFT), which contains an index of all the files on the partition. The last of the 16 sectors is a copy of the boot sector, which can be referenced if the primary is corrupted.

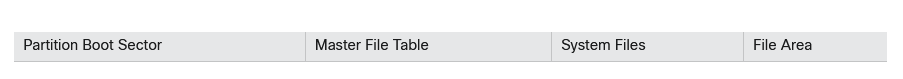
The MFT contains a file that maps the location of all the files and directories on the partition. NTFS also maintains a mirror or copy of the MFT that can be used if the primary is corrupted. In addition to file location, the MFT stores file attributes such as filename, security descriptor, and time stamps.

The system files are created when the NTFS volume is first initialized. These files are hidden and are used to store metadata about files, NTFS volumes, logs, and file attribute definitions.

The file area is where the files and directories are stored.

**NTFS Volume Structure**

Formatting a drive with the NTFS file system creates several system files, and the basic structures for reading and writing files and recording their locations.



The partition boot sector occupies the first 16 sectors of the drive. It contains information that points to the system bootstrap instructions and the location of the Master File Table (MFT), which contains an index of all the files on the partition. The last of the 16 sectors is a copy of the boot sector, which can be referenced if the primary is corrupted.

The MFT contains a file that maps the location of all the files and directories on the partition. NTFS also maintains a mirror or copy of the MFT that can be used if the primary is corrupted. In addition to file location, the MFT stores file attributes such as filename, security descriptor, and time stamps.

The system files are created when the NTFS volume is first initialized. These files are hidden and are used to store metadata about files, NTFS volumes, logs, and file attribute definitions.

The file area is where the files and directories are stored.

**NTFS Alternate Data Streams**

An interesting feature of the NTFS file system is the alternate data streams (ADS) technique. Files in NTFS are stored as a series of attributes such as the filename and date stamp. One attribute is called $DATA, which represents the actual file data, and is known as a data stream. However, NTFS allows you to attach ADSs to a file that provides for various interesting cases. ADSs are primarily used by applications to store additional information about a file.

ADS data is not readily visible and can be difficult to detect. In fact, recent versions of Windows just started providing tools for identifying files with ADS. A file with ADS can be referenced with the following format:

myFile.txt:ADS<*name*>

In the example below, the base filename is **myFile.txt**. The colon character (:) separates the base filename from the name of the ADS, which is **ADStest**. To test this feature, you can easily create a file with an ADS as follows:

C:\Users\Ed\test>**echo "My ADS data." > myFile.txt:ADStest**

This command sends the text that is specified in the **echo** command to an ADS that is associated with a file called **myFile.txt**. List the files in the directory with the **dir** command to confirm that the file was created.

C:\Users\Ed\test>**dir** Volume in drive C has no label. Volume Serial Number is 4428-917B Directory of C:\Users\Ed\test 09/09/2019 11:20 AM <DIR> . 09/09/2019 11:20 AM <DIR> .. **09/09/2019 11:20 AM 0 myFile.txt** 1 File(s) 0 bytes 2 Dir(s) 18,152,321,024 bytes free C:\Users\Ed\test>

Note that the **myFile.txt** file exists, but the size of the file is 0 bytes. The reason is that the data that was entered into the file did not go to the primary data stream, which is what the dir command is reporting. Instead, it went to an alternate data stream that is called **ADStest**.

To see the data that you put into the alternate stream, do the following:

C:\Users\Ed\test>more **< myFile.txt:ADStest** “My ADS data.” C:\Users\Ed\test>

The exercise illustrates an important ADS issue: It is easy to hide things in an ADS. This technique has been used for malicious purposes, so it is important to be aware of it. Recent versions of Windows have added tools to help administrators identify the presence of ADS. For example, the **dir** command now includes support for the **/r** parameter, which lists the ADSs that are associated with a file. Consider the following example:

C:\Users\Ed\test>**dir /r** Volume in drive C has no label. Volume Serial Number is 4428-917B Directory of C:\Users\Ed\test 09/09/2019 11:20 AM <DIR> . 09/09/2019 11:20 AM <DIR> .. **09/09/2019 11:20 AM 0 myFile.txt 17 myFile.txt:ADStest:$DATA** 1 File(s) 0 bytes 2 Dir(s) 18,157,400,064 bytes free C:\Users\Ed\test>

Note that the **dir** command with the **/r** parameter is now reporting the presence of the ADS attached to the file. Also note that the byte size of the ADS is also being reported and that the byte size of the file remains 0.

**1.**

**Question 1**

**When performing disk forensics on a Windows machine, which file system will most likely be involved?**

FAT

HFS+

EXT

NTFS

**2.**

**Question 2**

**Which part of the NTFS formatted disk drive can be damaged by corrupting the first 16 sectors of the drive?**

master file table

partition boot sector

system files

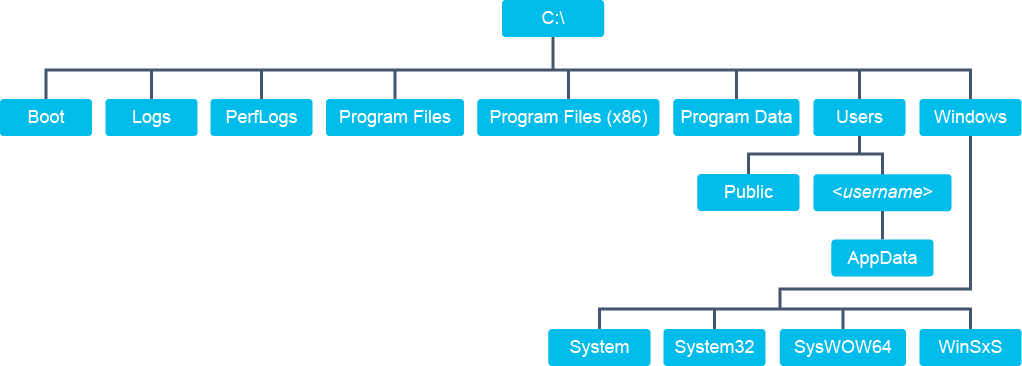
file area

**Windows File System Structure**

In a Windows-based system, every storage device that is connected to a host contains its own file system. The Windows operating system can be installed on virtually any drive that is connected to the host. However, it is most often installed on the first physical hard drive that is connected to the host. Hard drives, or any storage media for that matter, are identified with letters.

The first hard drive is given the letter C, followed by a colon (:). The Windows file system is not case-sensitive by default (although Windows can be configured to support case-sensitivity). Therefore, accessing the **C:** and **c:** drives will reference the same device. Additional storage media is given the next sequential letter designation.

The hard drive on which the Windows operating system is installed is typically organized in the following manner:



The following lists the major directories of of the Windows file system:

* **Boot**: A hidden directory that contains system boot files.
* **Logs**: As of Windows 10, the directory where event logs are stored.
* **PerfLogs**: Stores performance logs if this feature is enabled in the operating system.
* **Program files**: In 64-bit Windows installations, stores 64-bit applications.
* **Program files (x86)**: In 64-bit Windows installations, stores 32-bit applications.
* **ProgramData**: Directory used to store application-need data. Applications are not able to write directly to this directory, but they can create subdirectories used for their own purposes.
* **Users**: Directory where users store their data. Each user with an account on the system has a folder that is dedicated to them. Users with standard account privileges are only able to access the data in their assigned directory.
  + **Public**: This subdirectory of the Users directory can be shared among all the users of a system. Normally, Windows access controls do not allow users to access data in other users’ directories. However, the public directory allows anyone on the system to store data, making it ideal for sharing data among users.
  + **<username>**: All users with accounts on a Windows host are given areas for their use. The area for each user resides under the Users directory and bears the name of an individual user’s account. This area allows users to privately store their documents and other data since other users are permitted access to their own directory only. Each user’s directory includes a subdirectory that is called AppData. This location is used by applications that need to store things specific to a user, for example a user’s browsing history.
* **Windows**: The directory that stores the operating system and its various subsystems.
  + **System**: This directory is used to store 16-bit system dynamic link library (DLL) files. In 64-bit Windows installations, this directory is normally empty.
  + **System32**: This directory is used to store 64-bit system DLL files in 64-bit Windows installations.
  + **SysWOW64**: 64-bit Windows is backward compatible with 32-bit applications. This directory stores 32-bit system DLL files.
  + **WinSxS**: This directory stores copies of system DLL files. Often, these files represent older versions of the DLL files that may be required by older applications. The folder name stands for Windows side by side because it allows older applications to co-exist with newer ones that require the most current DLL files.

File System Paths

The Windows file system is hierarchical. The root of a disk drive represents the top of hierarchy with multiple levels of directories that can store files or other directories. The term “path” refers to the location of an object, such as a file, which includes the drive and hierarchy of directories that uniquely identify the location of the object. For example, if you use your browser to download a PDF document, the system will store the document in the downloads directory of your user directory, which resides on the disk drive on which the operating system is installed—usually the C: drive.

C:\Users\Ed\Downloads\myDoc.pdf

Each element of the path is separated by a backslash (\) character. Note that the path starts with a reference to the disk drive, followed by the series of directories that you must traverse to get to the file, and lastly, the filename. Known as a “fully qualified” path, it includes all the information that is needed to identify the specific location of the file.

Windows also supports relative paths to identify the location of an object. For example, when you open the command-line window, the system puts you in your user directory. The command prompt itself specifies the directory that you are in. To locate the PDF file that you downloaded with your browser, you only need to specify the location of the file relative to your current location. In the example that follows, the user wishes to copy the PDF file from the downloads directory to the documents directory:

C:\Users\Ed> copy Downloads\myDoc.pdf Documents\myDoc.pdf

Note that with the relative notation, the leading backslash is omitted since that would represent the root of the drive. Because the user was already in their user directory, and both the downloads and documents directories reside in the user directory, only those directories that need to be referenced in the command, along with the file to be copied.

**File Naming Conventions**

Filenames in the Windows file system consist of two components:

* Base filename
* File extension

The maximum number of characters that can be used to express a file path is 256 characters. Four additional characters are included in fully qualified names (bringing the total maximum number of characters that is supported to 260), which includes the drive letter, the colon that follows it, the backslash that represents the root of the drive and a null character at the end of the filename to terminate the filename.

However, it should be noted that Microsoft allows you to increase the MAX-PATH length within Windows 10.

The Unicode version of Windows file I/O functions allows for longer paths: approximately 32,767 characters. By using the extended paths, files and directories can be hidden and some tools are not able to find them. The Windows cmd shell only supports ANSI file paths. Any tool that uses ANSI file I/O functions or fails to support \\?\C:\ paths cannot access the directories and files that exist after the first 260 path characters.

In a Windows filename, a period or dot character (.) is used to separate the base filename from the extension. However, most other punctuation characters cannot be used in a filename because they have special meanings to the Windows command interpreter.

In Windows, the file extension is used to identify an association between a file and the application that is needed to open or execute the file. The list below contains examples of common file extensions and the applications with which they are typically associated:

* **.exe**: Executable file
* **.txt**: Text file
* **.rtf**: Rich text file
* **.pdf**: Portable Document Format
* **.bmp**: Bitmap graphic image

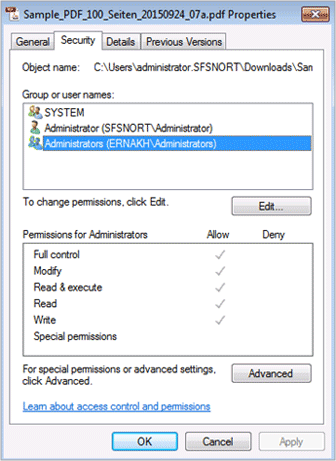
The list is just a small sample of file extensions. You will mostly see them expressed with three characters but that is not a requirement. File extensions are mapped in the operating system to specific applications. Often, a file format that is used by an application is assigned a unique icon by the application when the application is installed. It registers the extension association so that when a user double-clicks the icon that represents the file, the operating system will know which application to launch. Users can change these mappings but it is not done frequently.

**File Properties, Attributes, and Permissions**

File properties are essentially details about a file, including the file’s modification and creation date, file size, and other meta information about the file. Specific information that is stored as a property of a file can differ by file type. For example, an audio file will allow you to add a rating, but that same attribute is not supported for text files. Some properties can be modified depending upon the type of file and its ownership permissions.

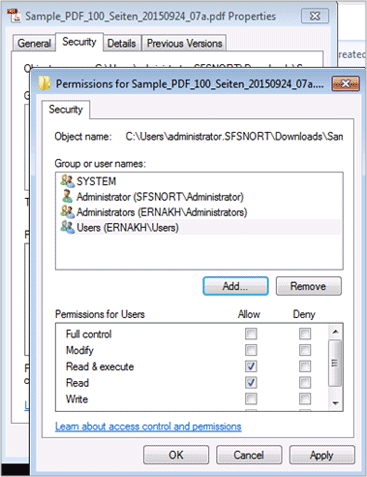
Another aspect of a file’s properties is file ownership, which allows Windows to enforce a degree of access control so that only the user that creates a file and the administrator initially have permission to access the file. File owners and administrators can grant permission to others. Note that to implement Windows file permissions, you must be using the NTFS file system.

To view or modify the ownership and permission properties of a file, right-click the file, choose **Properties**, and then select the **Security** tab.



Currently, the administrator account (represented by the single person icon) and the members of the administrators group (represented by the two-person icon) can access the file. The system user that is listed is a special account that is used by the operating system. By default, the system user is granted full control to every file. You can add users to the list to give others access to the file by clicking the **Edit** button under the **Group or user names** box.

In the figure below, the user has added the **Users** group to the file properties.



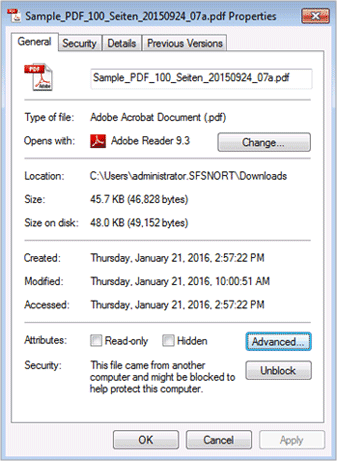
Once you have decided which users should have access to the file, you can specify the permissions to grant each user or user group. The permissions are defined below:

* **Read**: Allows users to view and list files and subdirectories.
* **Write**: Allows users to write to files or add files and subdirectories to folders.
* **Read and execute**: Allows users to view the contents of files, list files and subdirectories, and execute files if they are of an executable type. The read and execute permission does not grant permission to write to the file.
* **Modify**: Permits users to read and write to files and subdirectories and grants permission to delete files and directories.
* **Full control**: Permits reading, writing, modifying, and deleting files and subdirectories.

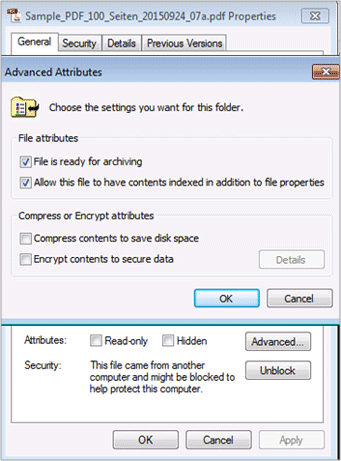
You have the ability to explicitly allow or deny file/directory permissions by checking the boxes in the appropriate column. If a specific permission level is not set, the deny permission will take precedence.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| [ ] | Allow | [ ] | Deny | = Deny |
| [X] | Allow | [ ] | Allow | = Allow |
| [ ] | Allow | [X] | Deny | = Deny |
| [X] | Allow | [X] | Deny | = Deny |

Another aspect of Windows file properties is file attributes, which are metadata associated with every file. File attributes are used to define how the system, applications, or users can interact with a file. You can view many of the attributes that are associated with a file by right-clicking the file and selecting Properties from the context menu.



Note the attributes section that is near the bottom of the window. Click the Advanced button to expose more attributes.



The following is a complete list of available file attributes:

* **Archive**: Used to mark files for backup
* **Compressed**: Indicates that the data in a file is compressed
* **Device**: Reserved for system use
* **Directory**: Marks the object as a directory rather than a file
* **Encrypted**: Indicates that the file or directory is encrypted
* **Hidden**: Hides the file or directory from standard directory listings
* **Normal**: Identifies files that have no other attributes set. It is only valid when used alone
* **Not Indexed**: The file or directory is not indexed by the content indexing service.
* **Offline**: File data is not available because it has been physically moved to offline storage; used by remote storage applications.
* **Read-only**: Applications can read the file data but cannot write or delete it
* **Reparse Point**: A file that is a symbolic link
* **System**: A directory that is reserved for use by the operating system
* **Temporary**: Indicates that the file is used for temporary storage. Typically, the application using the file deletes it when the file handle is closed.
* **Virtual**: Reserved for system use.

There are several ways to set or view file attributes. For example, from the file properties box, you can view or modify several, but not all, file attributes. You can also view and modify file attributes from the command line with the **attrib** command. The help text from the following command-line example shows the various options that you can use with the **attrib** command:

C:\Users\administrator.SFSNORT\Downloads>**attrib /?** Displays or changes file attributes. ATTRIB [+R | -R] [+A | -A ] [+S | -S] [+H | -H] [+I | -I] [drive:][path][filename] [/S [/D] [/L]] + Sets an attribute. - Clears an attribute. R Read-only file attribute. A Archive file attribute. S System file attribute. H Hidden file attribute. I Not content indexed file attribute. [drive:][path][filename] Specifies a file or files for attrib to process. /S Processes matching files in the current folder and all subfolders. /D Processes folders as well. /L Work on the attributes of the Symbolic Link versus the target of the Symbolic Link

Not all the attributes are available for viewing or change with this command because some attributes are only modified by applications or the system. In the example that follows, the **attrib** command is used to set the read-only attribute on a file:

attrib +R clamdoc.pdf

**Question 1**

**Which Windows directory stores the 64-bit applications?**

**Program Files (x86)**

Program Files

System

System64

**Question 2**

**Which Windows file attribute is used so that the file will not be listed from the standard directory listings?**

read-only

hidden

special permissions

not indexed

**Windows Domains and Local User Accounts**

Windows systems can act as stand-alone hosts, or they can be members of domains. Domains are groups of Windows hosts that are managed by a centralized authority, which is known as a domain controller. One administrative aspect of system management that is handled at the domain level is users and user groups. Stand-alone systems are capable of hosting multiple user accounts; however, each account is managed locally on the host.

**Windows Domains**

Windows domains represent a closed system of users and computers that can share resources and adhere to one centrally controlled management structure. Each user and machine belonging to that domain must authenticate with a domain controller to access the system. A domain controller is a server that is running a version of the Windows Server operating system and has Active Directory domain services installed.

A small organization might need only one domain with two domain controllers for high availability and fault tolerance. A larger organization with many network locations will need one or more domain controllers in each site to provide high availability and fault tolerance.

One of the greatest advantages of Windows domain setup is the ability to use group policy to control all the settings of each workstation in granular detail. The Active Directory is based on the LDAP which is designed to store information about virtually any type of object in a network that anyone may need to locate. Objects can be users, hosts, devices, or virtually any type of resource. Objects are grouped in structures that are known as containers. Containers can be nested with sub-containers to create a hierarchy structure. Domains are the entities in which containers are stored, and an organization may have one or more domains that can be organized by department or geographic location, for example.

**Local Users and User Groups**

A default Windows system ships with two default user accounts and several user groups. The default user accounts are guest and administrator. Recent versions of Windows no longer enable the administrator account because it runs with elevated privileges. At system initialization time, the system prompts you to create a custom user account so that you can access the system without the administrator account. The administrator account can be enabled at any time but the best practice is not to do so. If an administrator account or an account with administrator privileges is compromised, malicious code has greater access to critical system resources. Users that need to perform a task with administrator-level access can select the **Run as Administrator** option when they select the object to act on, such as a file or connected device.

The second default user account is called guest. This account is disabled by default and for most installations it should remain this way. It is intended for situations where a Windows host may be deployed in a public space or a location where it is likely to be accessed by many people that do not necessarily have accounts on the host. It grants a limited amount of privilege and is not protected with a password. Normally, customizations that a guest user makes during a session are erased when the guest user exits. When a new guest accesses the host, they are presented with a default desktop environment from which the user can launch authorized applications and access authorized system resources.

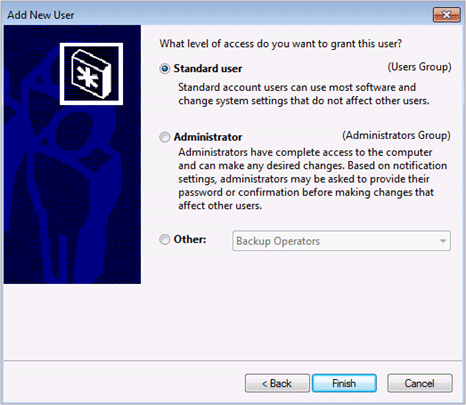
User groups determine the level of privilege that users have on the system. For example, placing a user in the administrator’s group grants that user administrator privilege. Several user groups are present on a standard Windows installation that grant specific rights to the users that belong to the group. For example, a user can belong to the backup operator’s group, to perform backup and restore operations. Most users will be assigned to the users group, allowing them to perform common tasks, run applications, use resources such as network printers, or lock and shutdown the system.

Starting with the release of Windows 8, Windows products have been gaining greater integration with Microsoft cloud services. To support these features, you could log in to a Windows system with a Microsoft account (rather than a local account), which allows user data to be synchronized between multiple desktop devices and mobile devices. For example, your files that are stored in the Microsoft cloud drive are automatically available at login time. And your desktop profile migrates to whatever host you log in to with your Microsoft account. A Microsoft account is available to any user that subscribes to a public Microsoft service, such as Hotmail or its web-based applications. Users of these versions of Windows can still create and use a local account on the host, but they lose cloud functionality unless they sign in to these applications manually.

**Creating a Local User Account**

There are several ways to access the application to manage user accounts on a Windows system, but the most common method is through the control panel. The exact method for accessing the control panel depends on the version of Windows, but, usually you can get to it from the Start menu in the lower-left portion of the screen. From there, you will either see an option to navigate to the control panel or, in recent versions of Windows, you will see a Settings option that opens a tablet-friendly version of the control panel. With the control panel or settings window open, you can navigate to the user accounts selection, where you can create or manage local user accounts.

When you create an account, you are prompted for the username and domain. If you are creating a local account, you can leave the domain field blank. Next, you are prompted to provide the level of access to grant to the user. Choose standard user access, administrator access, or other, which lets you specify the privilege level from a drop-down list of options.



After making your selections, click **Finish** to complete the process. When the user attempts to log in with the new account, they are prompted to create a password for the account. Note that to create an account, you must either be logged in as the administrator or be a member of the administrator group.

User accounts can also be created through the **net user** menu from the command shell.

**1.**

**Question 1**

**What is required for an attacker to create another local user account on a Windows host?**

edit the Windows registry

access to the Windows domain controller

log in as the administrator or be a member of the administrator group

root access

**2.**

**Question 2**

**A group of Windows hosts can be managed by which central controller?**

group policy controller

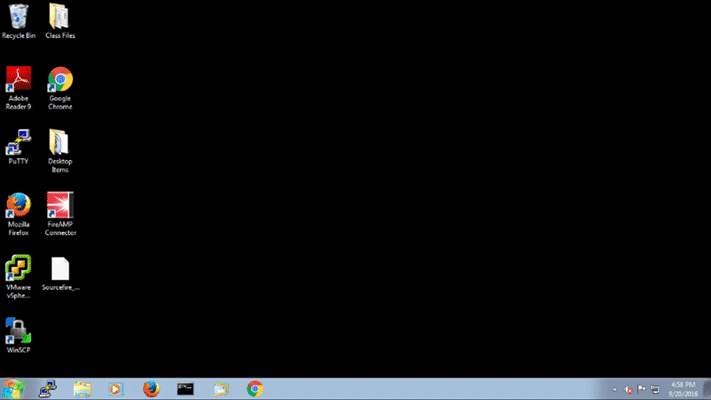
domain controller

LDAP controller

windows event controller

**Windows GUI**

One distinguishing feature of the Windows operating system is that it is operated primarily from the graphical interface. It has a command line from which you can execute commands and navigate the file system, but usually a Windows-based host is operated from the graphical interface. The following figure shows a typical Windows 7 desktop.



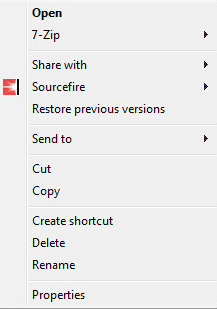
The main panel of the screen is known as the desktop. Each user is given a desktop, which is an environment that the user can customize in various ways, such as using an image for the desktop background or changing the color scheme. The desktop can also be used to store files, folders, or application icons. The recycle bin icon in the upper-left portion of the desktop is reserved for deleted files. Rather than making files unavailable immediately after deleting them, Windows stores deleted files in the recycle bin. Files are not fully deleted until you empty the recycle bin.

The lower portion of the screen contains the task bar which is further divided into three primary areas:

* **Start menu**: The icon that is at the far left portion of the task bar is the Start menu. The start menu gives you access to all your applications and other special features of the system such as the control panel. It also contains a field that you can type in to search for items or run commands.
* **Quick launch icons**: The middle portion of the task bar contains a series of quick launch icons, which act as launch buttons for applications that you place in the task bar. It also creates an icon for each application that is currently running. Clicking the icon of a running application brings it to the foreground so you can use it. You can also pin application icons to the task bar for easy access to frequently used applications.
* **Notification area**: This area contains several features for viewing notifications and controlling applications or processes that run in the background. The arrow icon opens the system tray which is where you can access options for controlling background applications. The flag icon lets you view system notifications and the icon to its right lets you view and control networking status. The shape of this icon will change depending upon how you are connected to the network. If you are wired to the network over an Ethernet connection, the icon will appear as a computer terminal; if you are connected over a wireless adapter, you will see the wireless icon here instead. This area also contains a clock so you can view the date and time. Lastly, there is a rectangular button in the far right portion of the task bar that is used to minimize all the running applications and show you the desktop.

**Windows Context Menu**

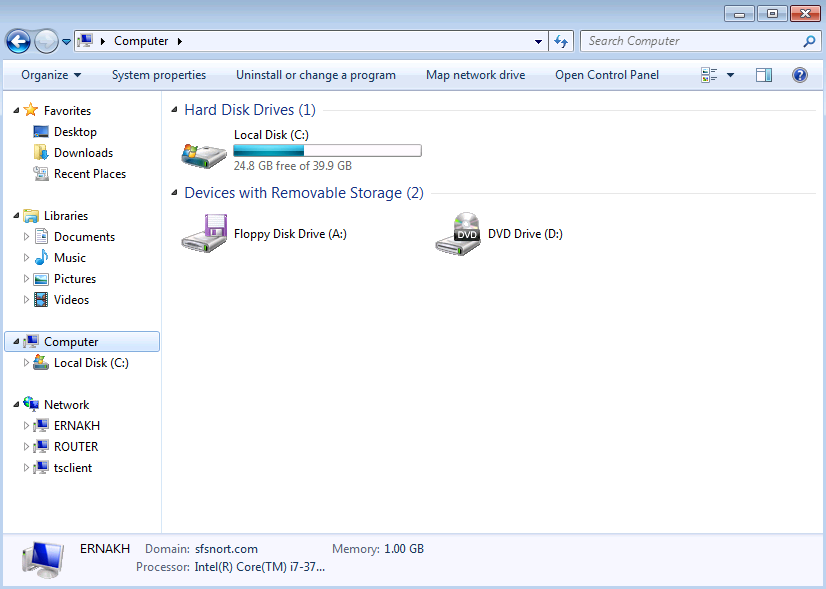
Another feature of the Windows desktop is the ability to bring up a context menu in virtually any part of the graphical interface. Right-click an item to open a context menu. Windows keyboards also contain a context Menu key that exposes a context menu at the location of the mouse pointer. The context menu presents menu options that are relevant to where you opened it. Also, some third-party applications add items to the context menu. For example, right-clicking a file produces the following context menu:



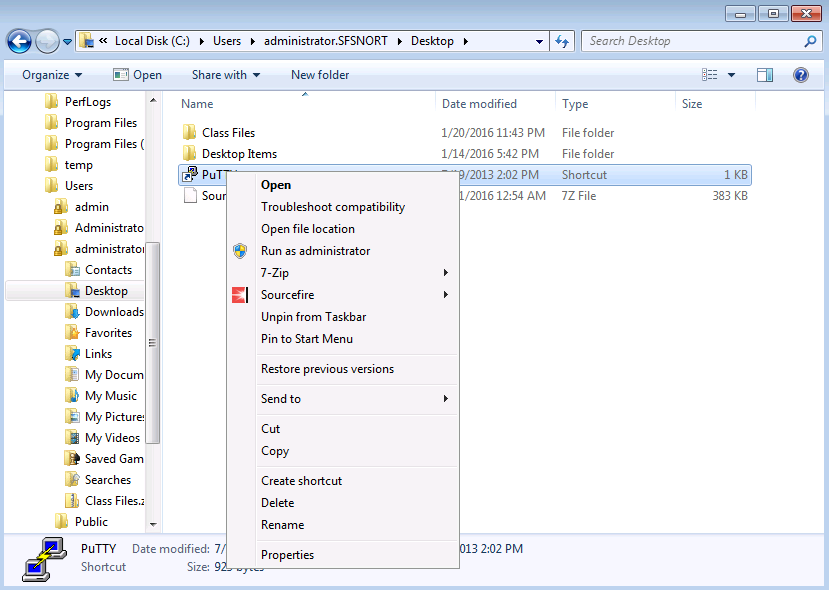
**Windows File Explorer**

Windows provides a tool for managing files and navigating the file system, which is known as the Windows File Explorer and is sometimes referred to as Windows Explorer. It is a utility that runs in the graphical environment and can apply many of the features of the graphical environment, such as drag-and-drop and the context menu.

Note: Windows File Explorer, previously known as Windows Explorer, is a file manager application that is included with releases of the Microsoft Windows operating system from Windows 95 onwards.



The navigation panel along the left side of the window provides shortcuts to various locations that you may frequently visit both in the local file system and network locations. A triangular icon in this panel indicates that these areas may be expanded to show more subsections that reside hierarchically. Click a selection in the left panel to populate the main panel a detailed view of the selection. In the example, the user selected computer, which displays the drives that are connected to the computer and basic information about each of them, such as how utilized the drive is. Double-click one of these icons to drill further down into the file system.



When you arrive at a file system location that interests you, right-click an item in that location to get the context menu. The context menu provides several file management options (for example, cut and copy) or other management options (for example, to view the file’s properties.

The Windows operating system is known for its rich graphical interface which makes for an intuitive and easy-to-use end-user platform. Though it does provide a CLI which is also quite robust, most users choose to operate the system using the graphical interface.

**1.**

**Question 1**

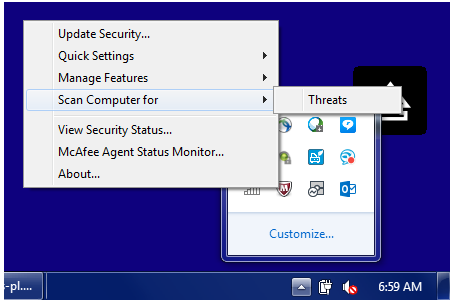
**Referring to the screenshot below, which area of the Windows GUI is accessed to initiate the McAfee scan?**

Start Menu

Quick Launch icons

Notification area

Windows File Explorer



**2.**

**Question 2**

**Which component of the Windows operating system GUI task bar is used for navigating the file system?**

Windows Context Menu

Windows Internet Explorer

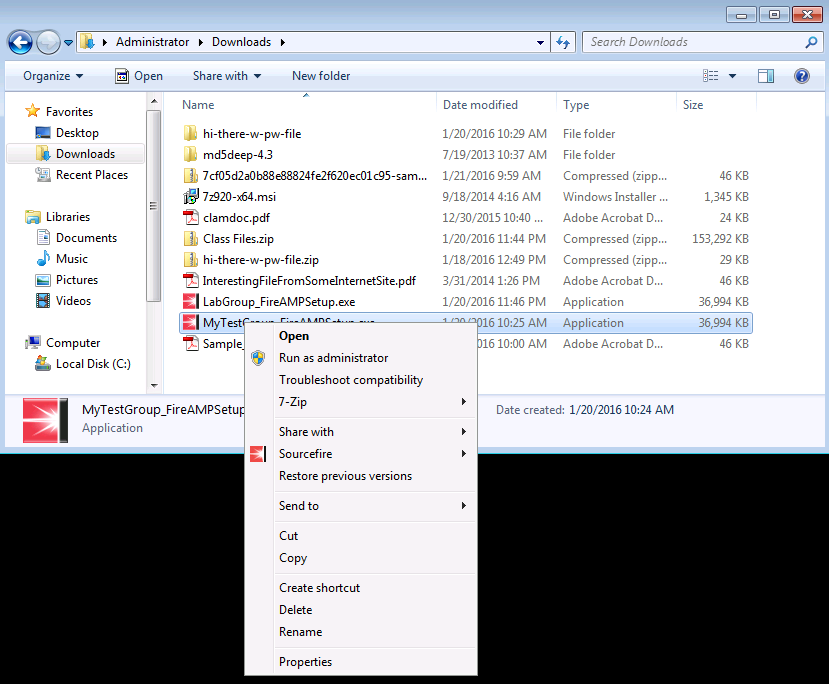
Windows File Explorer

Windows PowerShell

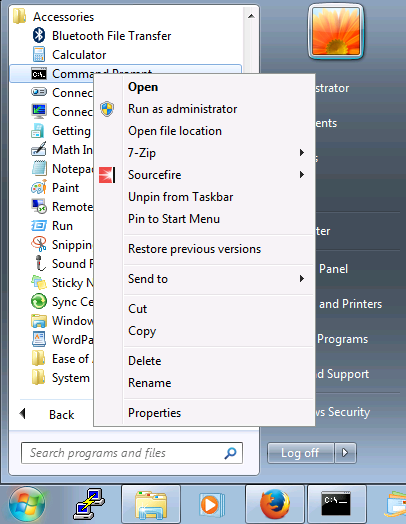
**Run as Administrator**

Certain tasks that you need to perform to administer a Windows system may require administrator privileges. Normally, the command interpreter runs with the permissions that are configured for the user who is operating the command line. However, you can elevate your permissions to administrator level using the following methods:

* **Right-click a command icon**: If there is a command you need to run as administrator, you can use the Windows File Explorer to find the command and right-click it to bring up its context menu. From there, you can **Run as administrator**, as seen in the figure. This technique can be useful in situations where you need to install an application with the administrator context. In the figure, the user is installing an AMP for endpoints connector which requires administrator privilege.



* **Open the command interpreter as administrator**: Right-click the icon for the command line to see that the same option, Run as administrator, is also available. The difference is that all the commands that you execute from the command line will run with the administrator context.
* **Windows Key-X** brings up a special context menu that contains a menu item, hotkey A, which opens an administrative command shell. The shell can be configured to Powershell from the Windows Registry.



**1.**

**Question 1**

**When you are logged in as a standard user, how do you run an application with administrator privilege?**

Left-click the application.

Double left-click the application.

Right-click the application, then select the **Run as administrator** option.

Launch the command line, enter the su root command, then left-click the application.

**2.**

**Question 2**

**How can an attacker who is using the Windows command line run all the commands with administrator privilege?**

Run the command prompt program as administrator.

Use the **su root** command.

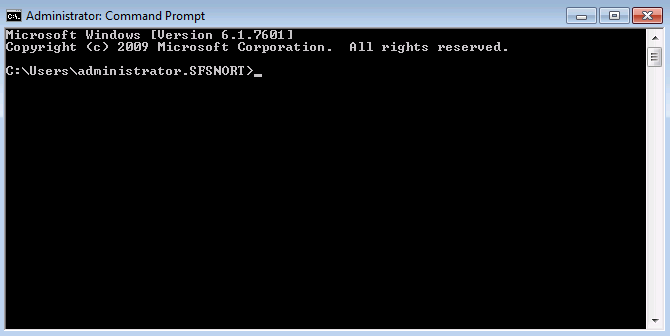
Boot Windows in Safe mode.

Use the **–admin** option in the command-line commands.

**Windows CLI**

In addition to the rich graphical environment, Windows includes a robust CLI that you can use to execute applications, manage files, and navigate the file system. You can also store commands in a text file and execute them, just as you would a shell script in Linux environments.

The Windows command line can be accessed in various ways. Click the **Start** button and enter **cmd.exe** in the search field, or navigate the menu system and select **command prompt** from the **Accessories** menu to begin entering commands, as shown in the figure.



The prompt displays the home directory of the current user, followed by the greater than symbol (>) and the blinking cursor, which is where you enter commands.

When using the Windows CLI, consider the following:

* **Case sensitivity**: By default, Windows commands, file paths, and filenames are not case-sensitive. However, the case-sensitivity of the text is preserved when creating a file or directory. Windows will not take case-sensitivity into account when expressing an item in the command line. However, Windows can be configured to support case-sensitivity.
* **Directory references**: Directory structures in a Windows installation are referenced with a starting point of the storage media on which the file you need to reference is located. Storage media is assigned a drive letter. After the drive is identified, you list the directories that the system must traverse to access the file. Drives, directories, and files are delimited with a backslash (\) character. For example:

c:\users\administrator\desktop\myFile.txt

* **Commands and command options**: You can execute commands from the command line. Often, the commands you execute will accept options. Command options are preceded with a forward slash (/) character. In the example that follows, a ping command is shown with an option to send a specific number of echo request messages:

ping /n 10 192.168.7.1

* **Accounting for spaces in filenames**: The Windows file system allows for filenames to contain spaces, which can cause problems in the command line because a space is typically used to separate a command from its arguments. To ensure that a space in the filename is interpreted correctly, enclose the filename in double quotation marks (“).

copy “Class Files.zip” d:

In the example, the user is copying the Class Files.zip file to the D: drive. Note that the filename contains a space and that the user enclosed the filename in double quotation marks (") to prevent the command line from misinterpreting the command.

* **Auto-complete**: A useful feature of the command line is the ability to automatically complete commands that reference directories or files. Press the Tab key as you enter file or directory references in the command line to automatically complete the statement.

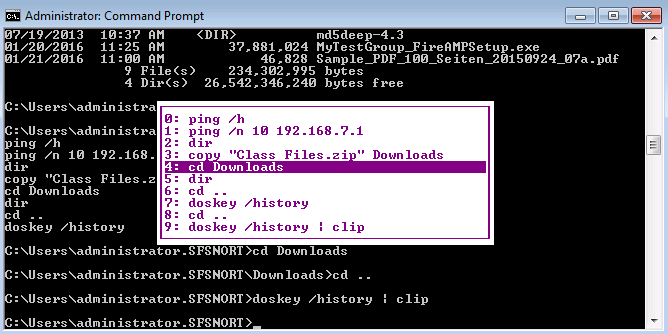
C:\Users\administrator> **copy Clas[Tab]** C:\Users\administrator> **copy “Class Files.zip”**

In the example, the user entered the **copy** command followed by a portion of the filename. Then the user pressed the **Tab** key to autocomplete the filename reference. If the command interpreter can determine the remaining portion of the file, it will complete the filename automatically. If the file does not exist in the location that was specified by the user, the command interpreter does nothing. In this case, check your syntax or spelling. Also note that the command interpreter recognized that the file contained a space character in the name and correctly enclosed the filename in double quote characters.

* **Command-line history**: The Windows command line maintains a history of the commands you used throughout the command-line session. Once you exit the session, the command history is erased. A new command history is created the next time that you open the command line. To access recently used commands, use the up or down arrow keys. You can list the command history with the following command:

doskey /history The output from this command would appear as follows: C:\Users\administrator>**doskey /history** ping /h ping /n 10 192.168.7.1 dir copy "Class Files.zip" Downloads cd Downloads dir cd .. doskey /history C:\Users\administrator>

Another option is to press the **F7** key when you are in the command-line window, which opens a box in your command-line session that lists the command history. Navigate the box with the up and down arrow keys or use the **Home** and **End** keys to go to the beginning and end of the command history list. You can also use the **Page Up** and **Page Down** keys to move a page at a time if the list spans multiple pages. When you highlight the command that you wish to use from the list, press the **Enter** key to execute it. The figure shows an example of the command history box after opening it using the **F7** key:



The Windows command-line history provides another feature that you may find useful if you are entering a lot of long commands that you don't want to type. Type a portion of a command and press **F8** to have the command interpreter fetch the most recent command that begins with the string that you typed. If the command that is presented is not the one you want, press F8 again and it will search the history for the next most recent instance of the string. For example, given the following history:

C:\Users\administrator>**doskey /history** ping /h ping /n 10 192.168.7.1 dir copy "Class Files.zip" Downloads cd Downloads dir cd .. doskey /history

If you type **cd** at the command prompt, press **F8** to draw the first instance of a command that begins with that string:

C:\Users\administrator>cd [**Press F8**] C:\Users\administrator>cd .. [**Press F8 again**] C:\Users\administrator>cd Downloads

**File and Directory References from the Command Line**

The Windows operating system designates all storage media as unique entities with their own file systems. Therefore, referencing a file or directory location will always begin with a specific drive letter. A user can easily switch from one drive to another by simply specifying the drive letter followed by a colon.

C:\Users\administrator\Desktop> **d:** D:\>

With the drive that you wish to use selected, you can begin to navigate its directory structure, manage files, or execute commands from the command prompt.

The Windows command prompt allows you to reference files and directory locations in one of two ways:

* **Fully qualified reference**: Using this method, you must enter the entire directory starting from the root of the current drive or a drive letter if the item that you wish to reference resides on a different drive.

D:\>**type c:\Users\ed\Desktop\hi-there.bat** @echo off REM Next command inputs the greeting echo Hello! Just wanted to say hello. REM Next command displays “Press any key to continue” and waits for user to press a key pause dir c:\windows D:\>

In the example, the user executed the **type** command, which is used to print the contents of a file to the screen. The user’s command prompt was in drive D:, but the file that the user was interested in was on the C: drive. The user referenced the file using the fully qualified path to the file.

* **Relative reference**: Relative references to file or directory locations take into account the current file system location as the starting point of the reference. If a file or command resides in a directory level that is higher than the current directory, use the double dot (..) notation to move up the directory structure, as in the example that follows:

C:\Users\administrator\Desktop> **copy “Class Files.zip” ..**

**Windows PowerShell**

Although the Windows command line has a rich set of features, including the ability to create batch file scripts, it lacks the ability to interact with core Windows operating system functions and graphical interface. The Windows PowerShell overcomes these drawbacks by providing a scripting environment that is deeply integrated with the operating system and graphical interface. It also provides a command-line interpreter for executing commands and directing their output to the screen, as you would by executing a regular command from the command line. The true power of the PowerShell however, is the ability to use it as a scripting language for automating tasks. It is also interoperable with other operating systems such as Linux and Mac OS X.

As of Windows 7 and Windows Server 2008, PowerShell has been included as an integrated component of the operating system. It can run on versions before Windows 7, but it has to be installed separately. PowerShell became an Open Source project in August of 2016, with the introduction of PowerShell Core, which is based on the NET Core Open Source framework. Open Source code can be obtained from GitHub.

Windows PowerShell can execute several types of commands:

* **Cmdlets**: Pronounced “command-lets,” cmdlets are programs that are designed to interact with PowerShell.
* **PowerShell scripts**: Sequences of PowerShell commands that are contained in a file that can be executed. PowerShell script files have a **.ps1** file extension.
* **PowerShell functions**: Functions are code snippets that can be referenced from within a script. Functions can be crafted to accept parameters.
* Standard executable files.

PowerShell was also designed with security in mind. First, to execute PowerShell commands locally with escalated privileges, you must use the **Run as Administrator** option. Second, the PowerShell Remoting function is disabled by default, and once enabled, administrator rights are required to connect to a remote computer via PowerShell. Lastly, PowerShell includes an execution policy to determine whether the system can run PowerShell scripts at all, or, if it can, the execution policy can enforce whether the script has to be digitally signed.

The following are the PowerShell execution policy elements:

* **Restricted**: Default execution policy that completely restricts the use of PowerShell scripts on the system.
* **AllSigned**: Indicates that any PowerShell script that you attempt to execute must be digitally signed.
* **Remote Signed**: The recommended execution policy mode that indicates that externally downloaded scripts must be digitally signed, but scripts that are created locally do not need to be signed. Windows considers a PowerShell script to be an externally downloaded script when the file contains an alternate data stream (ADS) to indicate that the file came from the Internet zone.
* **Unrestricted**: Places no restrictions on running PowerShell scripts.

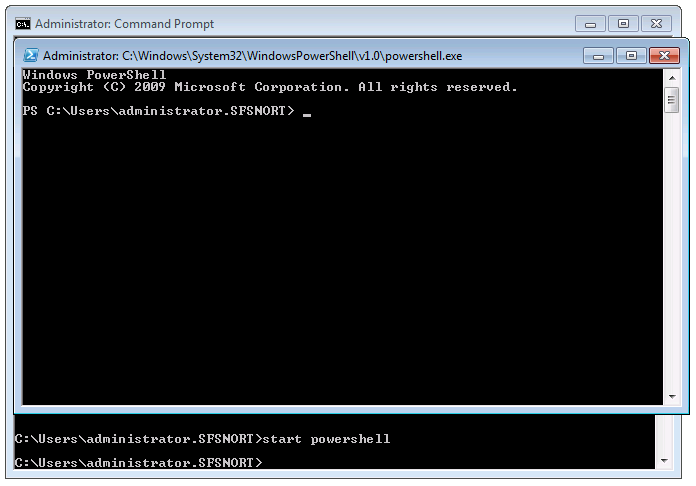
From within the PowerShell environment, you can determine the current execution policy by using the **get-executionpolicy** command. You can also set the execution policy by using the **set-executionpolicy** command from within the PowerShell environment.

Another aspect of the security that is built in to PowerShell is that it does not run commands from the current directory. It searches the directories that are stored in the PATH environment variable. If the PowerShell script that you are trying to call up is not in one of the directories that are specified in the variable, the system will return a “file not found” error, even if the script is in the same directory where you attempted to execute it. To execute a PowerShell script, it must be in the PATH variable, or you can provide the fully qualified path. For scripts in the same directory, put a period+backslash (.\) or period+forward slash (./) in front of the script name to instruct the system to search for the file in the current directory.

**Entering PowerShell**

The Windows PowerShell environment can be initiated in several ways. First, you can go through the start menu and find the icon to start a PowerShell session. If you simply click the icon, you will be restricted in what you can do. However, if you right-click the icon and choose the “Run as Administrator” option from the context menu, you will have full control of the PowerShell session.

Another option is to start a PowerShell session from the standard command-line window. Again, for full control, you should be utilizing the command line as an administrator. The command to start the PowerShell is **start powershell**. The new window opens with the PowerShell command interpreter (it may look similar to the standard command-line window, or have a different color scheme). What you see depends on the PowerShell version that is running on your system. For example, in the figure, you can see a PowerShell session that was started from a Windows 7 desktop.



In all cases, the PowerShell command prompt starts **packets per second**, immediately followed by the directory path from which you started the PowerShell session. Note that the banner indicates that you are in the PowerShell command interpreter environment, as does the title bar of the Window running the PowerShell.

Test the PowerShell session by running a simple command as shown below:

Windows PowerShell Copyright (C) 2009 Microsoft Corporation. All rights reserved. PS C:\Users\administrator.SFSNORT> **get-executionpolicy** Restricted PS C:\Users\administrator.SFSNORT>

In the example, the user executed the command to check the PowerShell execution policy. The value that it returned was “restricted” which means this host is not configured to run PowerShell scripts. To enable PowerShell script execution, enter the following command:

PS C:\Users\administrator.SFSNORT> **set-executionpolicy RemoteSigned** Execution Policy Change The execution policy helps protect you from scripts that you do not trust. Changing the execution policy might expose you to the security risks described in the about\_Execution\_Policies help topic. Do you want to change the execution policy? [Y] Yes [N] No [S] Suspend [?] Help (default is "Y"): **y** PS C:\Users\administrator.SFSNORT>

Note that the system returns a warning and gives the option to continue or exit. The default is to continue. Press the **Enter** key, or enter the letter **y** followed by the **Enter** key, to accept the default.

Another alternative for executing PowerShell commands or executing scripts is to use the **-command** option with the PowerShell command line **start** command, as follows:

start powershell –command “<PowerShell command>”

The **start** command starts the PowerShell and executes the command that you enter as the argument to the **-command** option. This method closes the PowerShell window after the command executes, so it might be better suited for launching scripts. You, can, however, use the **-NoExit** option to leave the window open when the command finishes.

**Using Windows PowerShell**

When you first start using PowerShell, you should explore the help system, which provides a quick means of giving you information and usage examples. Invoke the help system with the **get-help** command. Include the **more** command to prevent information from scrolling off the page before you have a chance to read it.

PS C:\Users\administrator.SFSNORT> **get-help | more** TOPIC Get-Help SHORT DESCRIPTION Displays help about Windows PowerShell cmdlets and concepts. LONG DESCRIPTION SYNTAX get-help {<CmdletName> | <TopicName>} help {<CmdletName> | <TopicName>} <CmdletName> -? "Get-help" and "-?" display help on one page. "Help" displays help on multiple pages. Examples: get-help get-process : Displays help about the Get-Process cmdlet. get-help about\_signing : Displays help about signing scripts. help where-object : Displays help about the Where-Object cmdlet. help about\_foreach : Displays help about foreach loops in PowerShell. set-service -? : Displays help about the Set-Service cmdlet. You can use wildcard characters in the help commands (not with -?). If multiple help topics match, PowerShell displays a list of matching topics. If only one help topic matches, PowerShell displays the topic. Examples: -- More--

The following is a partial representation of the help page, but it shows syntax examples, and examples of how to get help on specific commands.

To get the most out of the PowerShell, explore the four levels of help:

* **get-help** <*PS command*>: With no further parameters, returns the least amount of help information.
* **get-help** <*PS command*> **-examples**: Provides basic help information with examples.
* **get-help** <*PS command*> **-detailed**: Provides a more detailed amount of help information with examples.
* **get-help** <*PS command*> **-full**: Provides the most information with examples and greater technical depth.

The previous example, where the **get-help** command output was piped to the **more** command, is another powerful capability of PowerShell. Piping the output of one command to the input of another greatly increases your productivity with PowerShell by getting directly to information that you may find useful, rather than wading through long lists of command output feedback. In the example that follows, you may be interested in finding out about services that are installed on the host that you are evaluating. Use the **get-service** command to do so, but it yields a great deal of information including stopped services and running services. You may only be interested in the running services, so pipe the output of **get-service** to the **where-object** command to specify the output criteria.

The following is an example of how to execute these commands to filter the output to just running services. This output is also piped to the **more** command due to the size of the PowerShell window, so that the output does not scroll off the screen before you have a chance to read it.

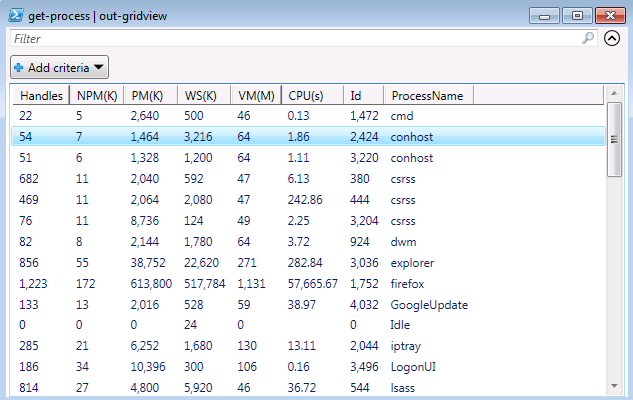
PS C:\Users\administrator.SFSNORT> **get-service | where-object {$\_.status -eq "Running"} | more** Status Name DisplayName ------ ---- ----------- Running AudioEndpointBu... Windows Audio Endpoint Builder Running AudioSrv Windows Audio Running BFE Base Filtering Engine Running BITS Background Intelligent Transfer Ser... Running Browser Computer Browser Running CertPropSvc Certificate Propagation Running CryptSvc Cryptographic Services Running CscService Offline Files Running DcomLaunch DCOM Server Process Launcher Running Dhcp DHCP Client Running Dnscache DNS Client Running DPS Diagnostic Policy Service Running eventlog Windows Event Log Running EventSystem COM+ Event System Running fdPHost Function Discovery Provider Host Running FDResPub Function Discovery Resource Publica... -- More --

Now the command output is more precise and usable. The parts of the command that make this work are the parameters of the **where-object** command. The first parameter specifies the field of information that you are interested in. The second parameter uses an equality operator to let you specify the string in the field that you wish to search for.

Another interesting feature of PowerShell is the ability to send output from a command to an interactive table in the graphical environment that you can sort and filter. The example that follows places the output of the **get-process** command to the interactive table.

PS C:\Users\administrator.SFSNORT> **get-process | out-gridview**

This command produces an interactive table as shown in the figure:



The window is the output of the command in a graphical table that you can interact with. Click column headings to sort on a column, filter items based on a fstring that you enter in the Filter field near the top, and add criteria to further filter the output. For example, if you click the **Add Criteria** button and select handles, you can enter logical operators as a filter, such as handles with a number greater than, less than, or equal to a value that you enter.

In this example, the output from the **get-process** cmdlet includes the following:

* **Handles**: The number of handles that are opened by the process
* **NPM(K)**: Nonpaged memory in kilobytes
* **PM(K)**: Pagable memory in kilobytes
* **WS(K)**: The working set of memory pages that are used by the process, in kilobytes
* **VM(M)**: Virtual memory, in megabytes, that is consumed by the process
* **CPU(s)**: Processor time, in seconds, that is consumed by the process
* **ID**: The process ID
* **ProcessName**: The name of the process

At a glance, you get an idea of the processing resources that are consumed by individual processes at the time that the command was executed. Obviously, the Firefox application was consuming most of the processing resources when the **get-process** cmdlet was executed.

**Importing PowerShell Functions**

PowerShell users have an active community that features sites, blogs, and various resources for users to support each other and share code. By taking advantage of these resources, you may find that a problem that you need to solve with PowerShell has already been solved by others. If you obtain code from another source or you wrote a function that you wish to use as a script, use the **import-module** command to do so.

For example, the Microsoft blogging community makes many useful code examples available. One entry, which was written by Shay Levy, includes a PowerShell function to find running processes that are listening for connections with their associated port numbers. You can copy the code example and import it, so that the function is available to run on your system as if it were a standard cmdlet. The process for performing the import is as follows:

PS C:\Users\administrator.SFSNORT> **import-module -name C:\Users\administrator.sfsnort\Downloads\Get-NetworkStatistics.psm1 -verbose** VERBOSE: Loading module from path 'C:\Users\administrator.sfsnort\Downloads\Get-NetworkStatistics.psm1'. VERBOSE: Exporting function 'Get-NetworkStatistics'. VERBOSE: Importing function 'Get-NetworkStatistics'.

In the example, the **import-module** command uses the **-name** parameter to identify the file with the code sample that you wish to import. It also adds the **-verbose** parameter to provide verbose output as it processes the file, which could provide useful information if the import were to fail. The extension of the file name is also provided; the convention is to use the .psm1 extension for importing this type of code sample.

With the code sample imported, execute it as if it were a regular cmdlet as follows:

PS C:\Users\administrator.SFSNORT> **Get-NetworkStatistics | format-table** Protocol LocalAddr LocalPort RemoteAdd RemotePor State ProcessNa PID ess ress t me -------- --------- --------- --------- --------- ----- --------- --- TCP 0.0.0.0 135 0.0.0.0 0 LISTENING svchost 752 TCP 0.0.0.0 445 0.0.0.0 0 LISTENING System 4 TCP 0.0.0.0 5357 0.0.0.0 0 LISTENING System 4 TCP 0.0.0.0 49152 0.0.0.0 0 LISTENING wininit 436 TCP 0.0.0.0 49153 0.0.0.0 0 LISTENING svchost 800 TCP 0.0.0.0 49156 0.0.0.0 0 LISTENING lsass 544 TCP 0.0.0.0 49157 0.0.0.0 0 LISTENING svchost 1872 TCP 127.0.0.1 5357 127.0.0.1 50464 TIME\_WAIT Idle 0 TCP 127.0.0.1 49765 127.0.0.1 49766 ESTABL... firefox 1752 TCP 127.0.0.1 49766 127.0.0.1 49765 ESTABL... firefox 1752 TCP 192.16... 139 0.0.0.0 0 LISTENING System 4 TCP 192.16... 3389 192.16... 62490 ESTABL... svchost 992 TCP 192.16... 50394 192.16... 445 ESTABL... System 4 TCP 192.16... 50415 168.63... 443 ESTABL... firefox 1752 TCP 192.16... 50463 104.21... 443 ESTABL... firefox 1752 TCP 192.16... 50466 192.16... 139 TIME\_WAIT Idle 0 TCP 192.16... 50469 157.55... 443 ESTABL... firefox 1752 TCP 192.16... 50479 204.79... 443 ESTABL... firefox 1752 TCP 192.16... 50481 137.11... 443 ESTABL... firefox 1752 TCP 192.16... 50482 137.11... 443 ESTABL... firefox 1752 TCP 192.16... 50483 93.184... 443 ESTABL... firefox 1752 TCP 192.16... 50484 52.4.2... 443 ESTABL... sfc 1276 UDP 0.0.0.0 123 \* \* svchost 576 UDP 0.0.0.0 500 \* \* svchost 984

The example output is truncated, but you can see that it provides a great deal of information which could be useful if you are investigating suspicious processes that have opened a port to listen on, for example. Note that the data within some of the columns is truncated as well. Adjust the width of the PowerShell window to see the entire message for each field in the output.

**Windows net Command**

The Windows operating system contains a robust suite of commands that are comparable to the command sets that are available in any other operating system. However, Windows operating system includes a set of commands that are specific to Windows administration and maintenance. These commands are known as the **net** commands since the commands start with the term **net** and are often used to manage network resources, although there are also options that focus on local system resources.

To get information about using these commands, enter **net help** at the command prompt to display a list of various **net** command options, and instructions for getting help with specific options.

C:\Users\administrator.SFSNORT>**net help**

The syntax of this command is:

**Windows net Command**

NET HELP command -or- NET command /HELP Commands available are: NET ACCOUNTS NET HELPMSG NET STATISTICS NET COMPUTER NET LOCALGROUP NET STOP NET CONFIG NET PAUSE NET TIME NET CONTINUE NET SESSION NET USE NET FILE NET SHARE NET USER NET GROUP NET START NET VIEW NET HELP NET HELP NAMES explains different types of names in NET HELP syntax lines. NET HELP SERVICES lists some of the services you can start. NET HELP SYNTAX explains how to read NET HELP syntax lines. NET HELP command | MORE displays Help one screen at a time.

An example of using a net command is to view or edit local user account login policy. You can use the net accounts command as follows:

C:\Users\administrator.SFSNORT>**net accounts** Force user logoff how long after time expires?: Never Minimum password age (days): 1 Maximum password age (days): Unlimited Minimum password length: 7 Length of password history maintained: 24 Lockout threshold: Never Lockout duration (minutes): 30 Lockout observation window (minutes): 30 Computer role: WORKSTATION

With this usage, you can view the local user login policy, and modify the policy if you wish. Currently, based on the feedback from the **net accounts** command that was issued previously, the Maximum Password Age is unlimited. You prefer that the user of this host periodically changes their password, but you are not sure how to structure the command. The following syntax to will provide help:

C:\Users\administrator.SFSNORT>**net help accounts**

There are many more uses for the **net** commands. For example, if you need to mount a file share to your Windows host, use the **net use** option of the **net** command.

C:\Users\administrator.SFSNORT>**net use z: \\192.168.7.88\users /user:emendez** Enter the password for 'emendez' to connect to '192.168.7.88':

This command mounts a share to the remote host 192.168.7.88 and it connects to a share that is named “users.” When the share mounts, it will be mapped to the driver letter **z**:. So, the local users of this host will be able to use the remote share as if it were a locally connected disk drive. Lastly, the command specified a username to log in with on the remote host. If the username is a valid user on the remote host, the user is prompted for a password. Upon entering the correct password, the share becomes available to use on the local host.

This last example demonstrates how to use the command to stop and start network services. First, you must know the name of the service that you wish to control. In the example, the system admin will use a service that is called **FileInfo**. The following represents the sequence of commands to stop and start the service:

C:\Users\administrator.SFSNORT>**net stop fileinfo** The File Information FS MiniFilter service was stopped successfully.

C:\Users\administrator.SFSNORT>**net start fileinfo** The File Information FS MiniFilter service was started successfully.

Becoming proficient in **net** commands is an important skill for Windows administrators. You've seen just a few examples, but there are many more useful features of the **net** commands.

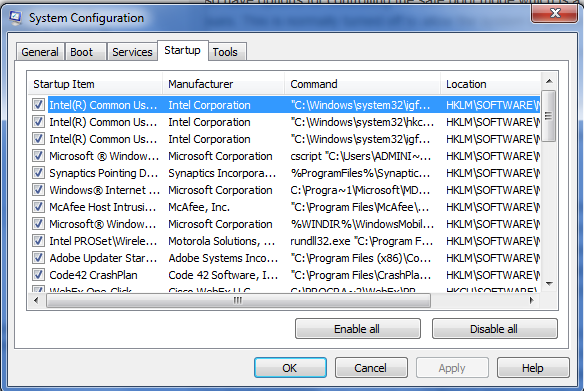
**Controlling Startup Services**

It is possible to manually edit the registry keys directly with the registry editor to control auto-start services, but if you edit something incorrectly or delete a key that is required by the system, you can damage a Windows system. Tools are available to view auto-start services and change their start-up properties without directly manipulating the registry. For example, the **Msconfig.exe** tool ships with the operating system and allows you to view and edit system start-up properties. Access the tool by entering the **msconfig** command at the prompt, or in the search field of the start menu.

The Msconfig.exe tool allows you to control various aspects of the system configuration. Options are grouped as tabs in the following categories:

* **General**: This group of settings allows you to control various options for starting up the system. The default is a normal startup, but you can choose a diagnostic startup or a selective startup where you can control whether to start system services, startup items or whether to use the original boot configuration.
* **Boot**: If multiple versions of Windows are installed, you can choose which one to boot on startup. You also have options for controlling the safe boot mode, which is a boot process that is used for troubleshooting system issues. Safe boot mode is normally turned off to allow the system to perform a normal boot.
* **Services**: This section lists the services that are loaded in the system and their state: Stopped or Running. It also gives you the option to disable or enable the services, which will require a reboot to take effect.
* **Startup**: This section lists the applications and services that are configured to automatically start at boot time. It also gives you the option to disable or enable the applications and services to start at boot time, which will require a reboot to take effect.
* **Tools**: This section lists the tools that ship with the operating system. You can launch the tools directly from this window.

The figure shows the **Startup** tab. Here, you can see a listing of the applications and services that are configured to start up on system boot. Note that each entry has a check box to let you determine whether to start each service. Other information about each entry is presented which could be useful if you are tracking down a suspicious service or application. For example, it lists the manufacturer of the service, the command that is used to start it, and the location in the registry that instructs the system to start the service.



**Windows System Shutdown**

The best practice when halting a Windows system is to perform a proper shutdown, rather than simply turning it off. A proper shutdown gives the system a chance to terminate applications and services with dependencies in the proper order, and write any configuration changes or pending edits to their respective files. Shutting down the system incorrectly can corrupt open files or damage applications that were in the process of doing something when the system was turned off.

Part of the shutdown process is to inform applications and services that a user requested a shutdown. The system communicates this request to each process so that the processes can start terminating open threads, and cleaning up open file handles. If a process thread does not respond within a predetermined period (default is 20 seconds), the system displays the hung application warning box, giving you the option to wait or close the application. After user mode applications have closed, the system begins terminating system processes. These processes are also bound by the timeout, however, no warning is displayed. The system may appear to hang at this point while it waits for the offending process to terminate. In this situation, after you have waited a reasonable period, you may have no choice but to manually turn off the system.

You can use various methods to shut down a Windows system:

* **From the Start menu**: The Start menu configuration may differ slightly from one version of Windows to another, but you can always find it in the lower left portion of the screen. From there, you will see an option to shut down the system. Selecting that option presents a message box for you to select the type of shutdown to do.
* **Ctrl-Alt-Delete**: Affectionately known as the “three-finger salute” because this method requires the user to press the Ctrl, Alt, and Delete keys at the same time. A screen pops up with a series of options, including system shutdown. Note that the most recent versions of Windows eliminated this option from the list and replaced it with an icon in the lower right that brings up the list of shutdown options.
* **Command line**: Execute the shutdown command from the command line to begin the shutdown process. The command can take various parameters to determine the type of shutdown to perform.

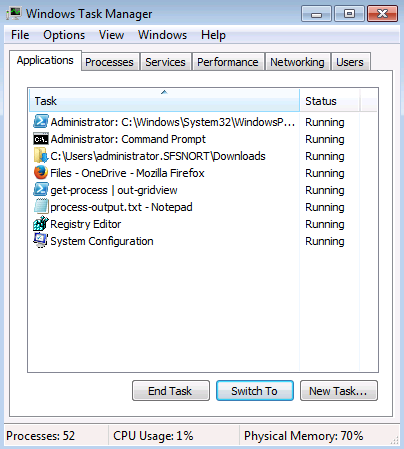
The following are three ways to shut down/restart a Windows host:

* **Shutdown**: The operating system completely shuts down.
* **Restart**: The operating system completely shuts down and the system automatically reboots.
* **Sleep/hibernation**: Captures system state and writes it to a file. The system then enters a low-power state but it is not completely turned off, allowing users to resume operation quickly without having to boot the host. The feature is primarily designed to accommodate laptop and mobile devices.

**Controlling Services and Processes**

When Windows is up and running, administrators need to know about the services and processes that are operating on a given host. Obtaining that information can be the result of simple system maintenance or it can be part of an investigation to see if there are suspicious processes that are running on a compromised host. This topic presents several tools that are available to get information about processes and manage them.

The task manager is a primary tool that every Windows administrator should be familiar with. The figure shows an example of the Task Manager window from a Windows 7 installation.

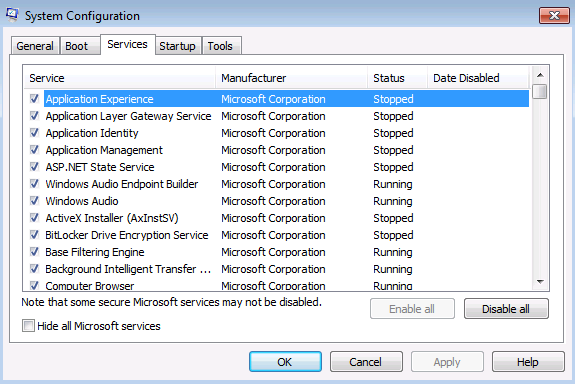


The task manager provides a great deal of information regarding what is running on the system including system performance metrics. Because it pertains to services and processes, the first three tabs provide the most useful information.

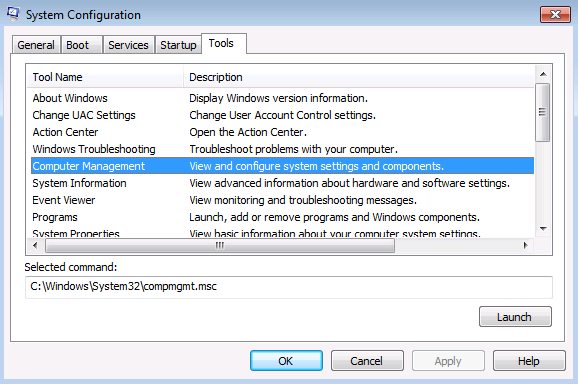
* **Applications**: Lists the applications currently running on the system and gives you the ability to end the task or switch to it. This area can be useful for killing hung applications.
  + Right-clicking an application in this list gives you several options in a context menu. One option that you may find useful is to jump to the process that runs the application.
* **Processes**: Lists processes running on the system and it also gives you the ability to sort applications several ways: by name, by the user that owns the process, by CPU resource consumption, by memory resource consumption and description. This view can help you identify processes consuming resources or acting strangely. You can also terminate misbehaving processes.
  + Right-clicking a listed process opens a context menu with various options, such as opening the location of the file that starts the process or opening the processes properties.
* **Services**: Lists the services that are loaded on the system, and the service status: Running or Stopped. You can also open the Services management console applet, where you have greater control over loaded services and configuring their properties.
  + Right-clicking an item in the services list allows you to open a context menu with several options. For example, you can stop or start the service depending on its current state, and you can jump to the process that runs the service in the process list.

**The msconfig Utility**

The msconfig utility also has the ability to show the status of the services. The figure below shows the Services tab of the **msconfig** utility.



The amount of management that you can perform from **msconfig** is somewhat limited: you can only enable or disable the services at the next reboot. But it allows you to view the services and potentially identify any that appear suspicious. You may also find useful tools to give you greater control through the **Tools** tab.

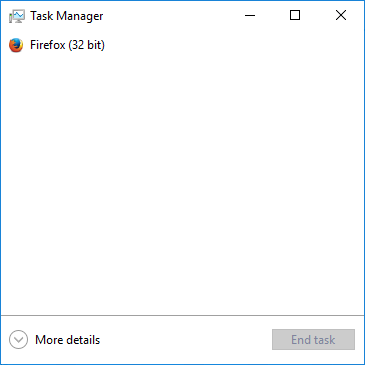


In the **Tools** tab, you can select a tool and launch it from this application window. In this example above, the user has selected the Computer Management tool which gives access to control a wide range of system settings, including services.

**Windows 8 (and Greater) Task Manager**

The task manager was updated in recent versions of Windows (version 8 or greater), and added some key enhancements and tabs that provide more useful information. This section provides an overview of the updated task manager.

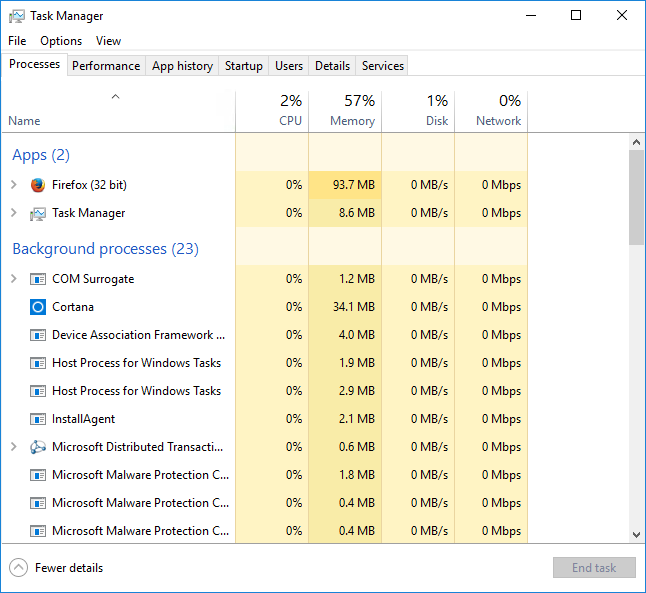
First, the task manager opens as a simple window that lists applications that are currently running on the host.



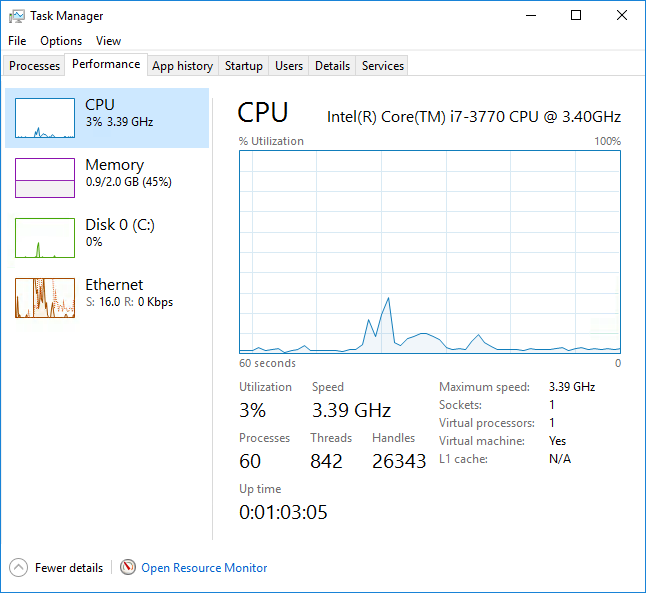
To get to all the task manager features, click **More Details** to expand the window to reveal the full set of task manager features as seen in the figure below.

Note that the application tabs have changed. The information is better organized graphically.

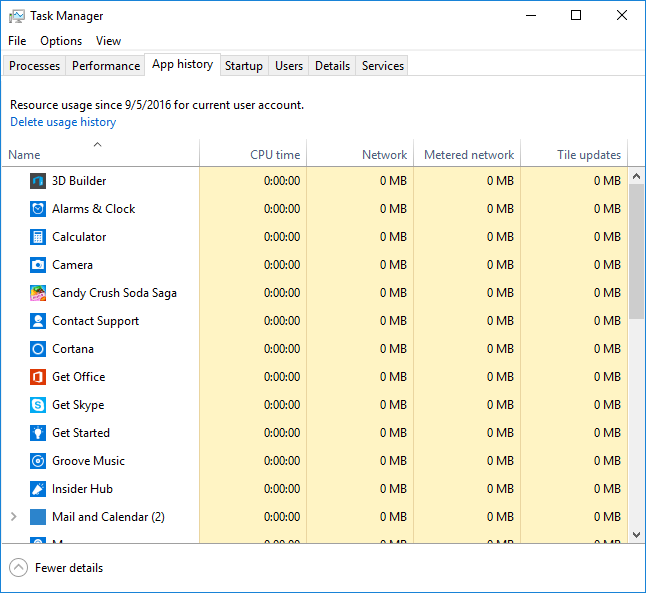
* **Processes**: This presentation includes both applications and processes in a single list. Applications and processes consuming large amounts of resources relative to others get shaded in various ways. Generally, the darker the shade of yellow and then orange, the higher the resource consumption. If resource utilization has reached a critical level, the value will be shaded in red. Processes are organized in terms of Background processes and Windows processes. Each section of the list is labeled accordingly.



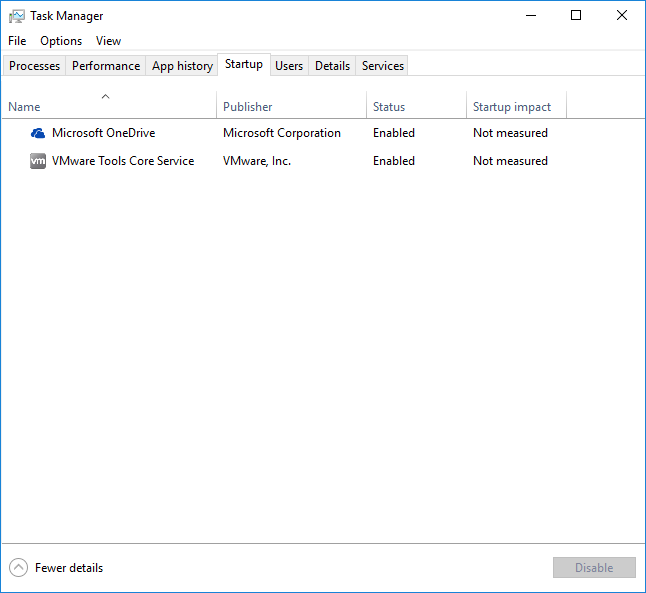
* Another enhancement is that processes running sub-processes are identified with a button in the shape of a greater-than symbol. Clicking this button expands the selection to show the sub-processes. In the case of applications that are running multiple windows, such as a browser with multiple tabs open, clicking the button lists all the windows or tabs belonging to that instance of the application.
* One enhancement that could be particularly useful is embedded in the context menu. Right-click an application or process in the list to see the context menu that is associated with that item . The option that is called “Search Online” initiates a search for the item using the system’s default search engine. A key use case for this feature is to do a quick lookup on a process that is suspicious. However, many classes of malware either misrepresent what is displayed in the task manager or disable it all together.
* **Performance**: This tab aggregates all the major performance metrics (CPU, memory, disk, and network) into a single section. You can click a resource in the left portion of the tab and the main panel populates with the performance details of that resource including a general performance graph over time and values of key metrics that are dynamically updated as resource consumption changes.



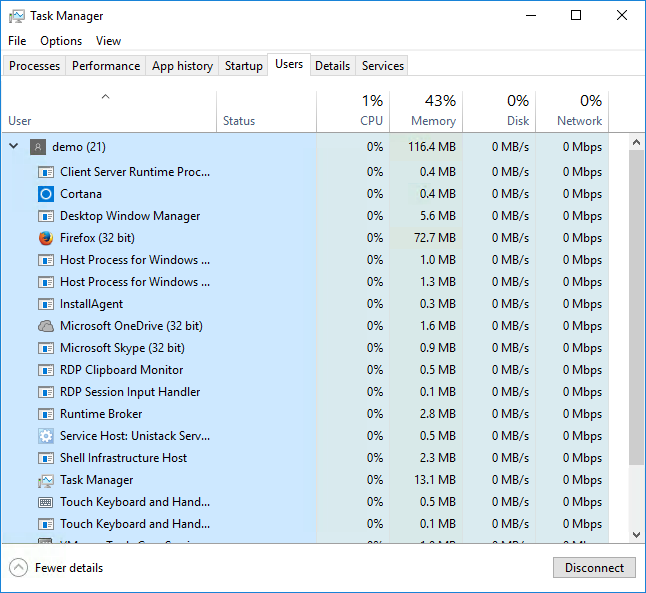
* **App history**: By default, this tab shows historical resource utilization of applications over time, giving you an idea of which applications may be consuming high amounts of resources. Edit the properties of this page to display historical resource consumption for all applications by clicking the Options menu at the top of the window.



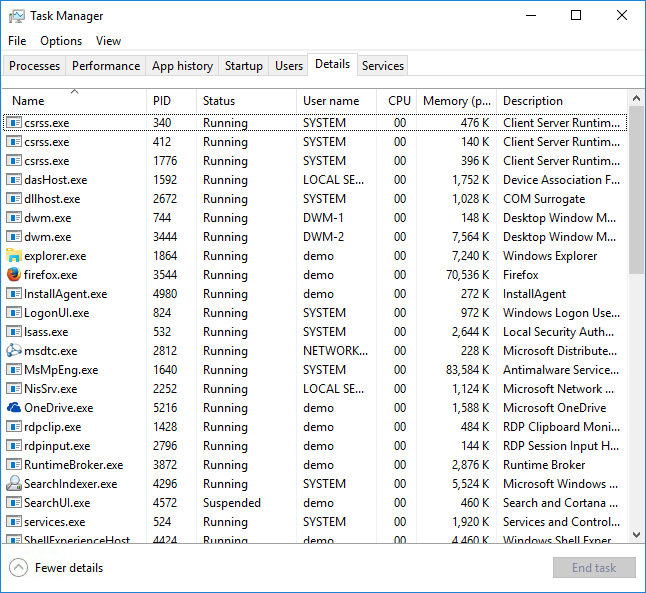
* **Startup tab**: This tab presents a list of applications and services that start up at boot time. You can select an item from this list and click the Disable button in the lower-right portion of the window to prevent the application from initializing at boot time.



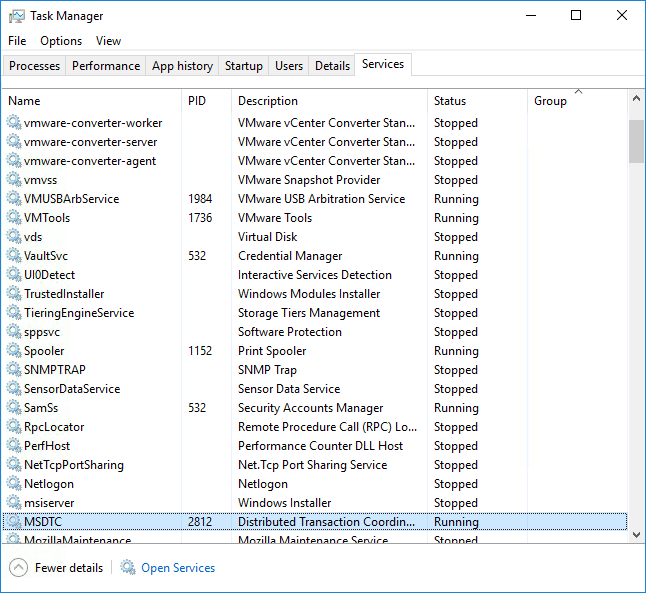
* **Users tab**: This tab displays users who are currently in the system. It also displays the resources that are consumed by the applications and processes that belong to each user. You can see a list of these processes by clicking the arrow button in front of the listed user to expose the list of processes belonging to the user. You can also select a user and disconnect him or her from the system if you have administrator privilege.



* **Details tab**: This tab displays a list of all the applications and processes running on the system. It expands on the capabilities of the Processes tab in that it displays status information for each item and it makes more advanced options for managing processes available through the context menu when you right-click an item. For example, it allows you to set a processing priority for an item. The context menu also allows you to set CPU affinity. Affinity is a feature that is used primarily on multi-core or multi-CPU systems, allowing a process to run on a specific core or CPU. Another option on the context menu is called, Analyze wait chain. This option will display the status of a process that might appear to be hung up. You can easily determine from this option whether a non-responsive application is waiting on another process.



* **Services tab**: This tab, as the name implies, lists the services that are loaded on the system and the service status (Running or Stopped). It is not much different than its predecessor from previous versions of Windows but it does include the ability to do an online search for a service that may not be familiar to you. It also allows you to open the Services applet from the management console application to give you greater control over the properties and configuration of services.

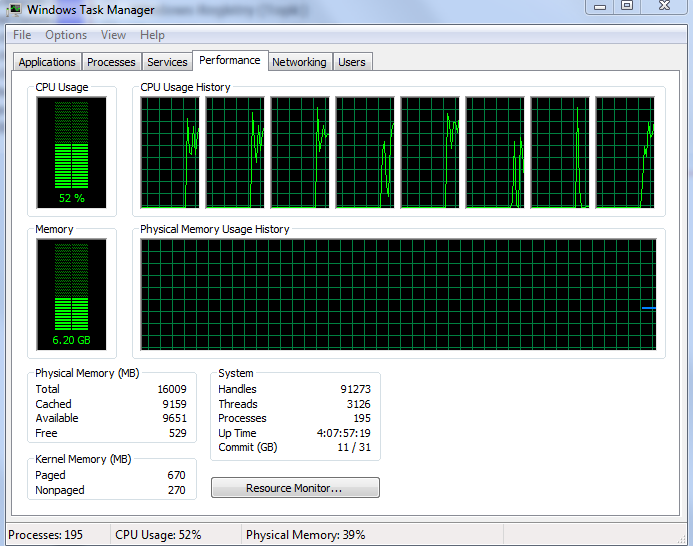


**Monitoring System Resources**

In addition to information that is related to services and applications that can be gathered from the Task Manager, you can also gather information that is related to system performance. Three Task Manager tabs provide information about various aspects of system performance:

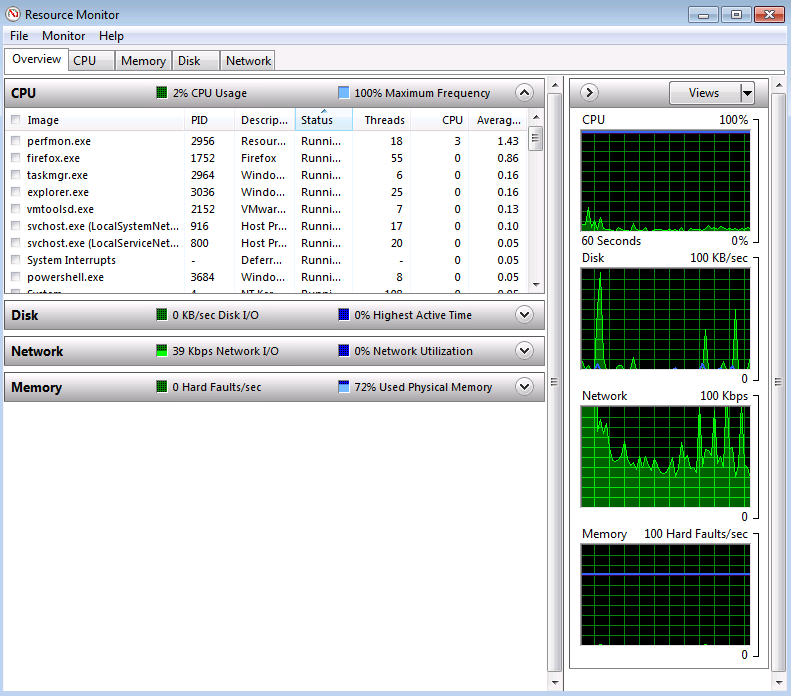
* **Performance**: This tab, as shown below, shows the general system resource utilization. It contains a graph of CPU utilization and a second graph of memory consumption. It also contains more detailed memory utilization counters.
* **Networking**: This tab shows a graphical representation of network utilization for each network adapter on the system. It also shows a table of the processes consuming network resources as a percentage of the total. The information can be helpful in tracking down applications or processes consuming large amounts of network bandwidth.
* **Users**: This tab shows the users who are currently logged in to the host. It gives you the option to disconnect users that are remotely logged in or log them off entirely.

The Task Manager provides a lot of information about the system at-a-glance.



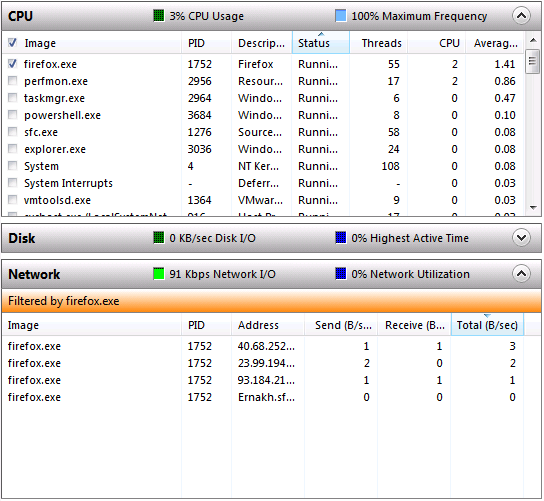
**Resource Monitor**

If you need to dig deeper to get more detailed information, Windows ships with a tool that is called Resource Monitor that provides just that. The figure shows an example of Resource Monitor from a Windows 7 installation. This application is available on all the most current versions of Windows, however, its functionality is unchanged from prior versions. You can access this application several ways: open it from the **Start menu->All Programs->Accessories->System Tools->Resource Monitor**, or type resmon.exe into the Start Menu search box, or you can call it up from the Performance tab of the task manager.

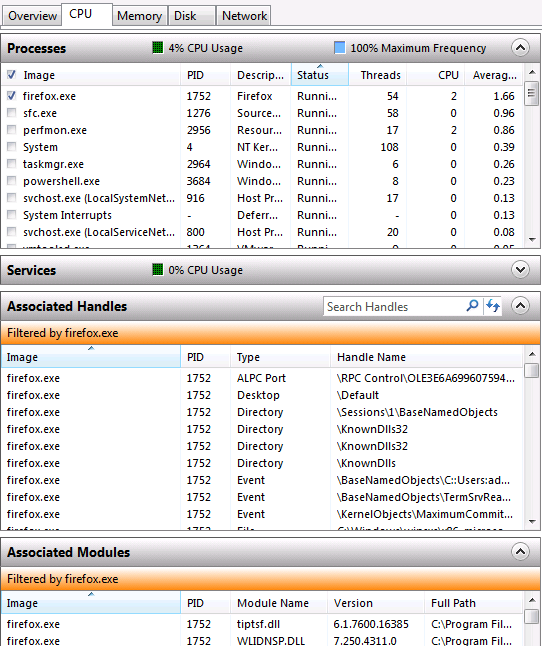


The resource monitor tool gives you a detailed view of resource utilization on your system. The application is focused on the major resources on your system: CPU, memory, disk, and network resources. The first section of the application is represented by the contents of the Overview tab, which provides general utilization statistics for each of the resources. You can also select a process from the process list at the top of the page by clicking the check box in front of the item, to filter the utilization data on the selected item or items and lets you see, on a per process basis, how an item is utilizing available resources.

When you select an item, expand each of the remaining sections to show utilization statistics for each of the resource categories. In the figure that follows, the user has selected the Firefox process in the top portion of the window and has expanded the network section by clicking the arrow button in the right side of the section title bar.



The network section inserts a message to indicate that the results are filtered on firefox.exe, giving you a means for understanding the resource utilization of specific applications and processes. The filter persists across all the tabs, so if you want greater detail on CPU utilization for a selected application, click the CPU tab, and the statistics that are rendered by the resource monitor remain filtered by the selected application, as seen below.

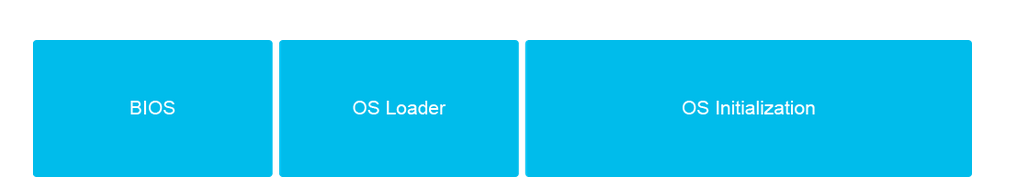


Remove the filter by deselecting the item in the top portion of the window. Doing so reverts the resource monitor to showing statistics that are based on all the running applications and processes. Each of the remaining tabs provides greater detail regarding the resource that you selected. If you need to investigate a system that is exhibiting erratic behavior, this tool can help you track down the source of the behavior.

**Windows Boot Process**

The figure that is shown below shows the three main phases of the Windows boot process:

* During the **BIOS initialization** phase, the platform firmware identifies and initializes the hardware devices, and then runs a power-on self-test (POST). The POST process ends when the BIOS detects a valid system disk, reads the master boot record (MBR), and starts Bootmgr.exe. Bootmgr.exe finds and starts Winload.exe on the Windows boot partition, which begins the operating system Loader phase.
* During the **operating system loader** phase, the Windows loader binary (Winload.exe) loads essential system drivers that are required to read minimal data from the disk, and initializes the system so the Windows kernel can begin execution.
* During the **operating system initialization** phase, most of the operating system work occurs. This phase involves kernel initialization, Plug and Play activity, service start, login, and Explorer (desktop) initialization.



The Windows boot process actually begins at system installation time. The Windows setup builds the structures that are necessary for the system to boot properly. Therefore, these structures must be in place for any subsequent successful system initializations to take place. There are also some differences in the boot sequence that take into account the type of BIOS that is in place. BIOS-based systems are essentially legacy but they are still very common. The newer type of BIOS is known as the extensible firmware interface (EFI). A more recent version of EFI, known as unified extensible firmware interface (UEFI), is rapidly becoming the new standard.

When an instance of Windows is installed on a host, the setup application does some preliminary preparation of the disk that the system will boot from. It will use the first sector of the disk to create an area that is known as the master boot record (MBR), which contains some executable instructions and the partition table that is used to map out the structure of the disk. Once MBR is created and present on the system disk, the host can begin the boot process.

When the system boots, it executes code that is stored in the flash memory of the BIOS. The code that it executes is enough to identify the boot disk it will then communicate with and load the partition table information from the MBR into memory. It also transfers control to the executable code in the MBR to continue the boot sequence. With the partition table loaded into memory, the system can locate the Bootmgr file. When the Bootmgr is found, the MBR executable code transfers control to the Bootmgr. Bootmgr starts in “real mode,” which is the original MS-DOS memory model where programs ran in a memory space of 1MB. Bootmgr’s job is to switch the system to “protected mode,” which makes the remaining memory of the system available so the boot sequence can continue. If Bootmgr is not found at this point, the boot process will fail and the system will respond with the message Bootmgr not found.

Next, the Bootmgr reads the Boot Configuration Database (BCD) to read in any options that are required to start the system. The BCD also contains information about whether the system was put into hibernation the last time that it was used, rather than fully shut down. If a hibernation state is detected, the BCD redirects the boot process to execute **Winresume.exe**, which reads the hibernation file (Hiberfil.sys) to rebuild the state of the system to the state it had when it was put into hibernation.

At this point, Bootmgr executes **Winload.exe** to continue the boot sequence. The **Winload.exe** program queries the system BIOS to get information about devices connected to the system. The hardware information is held so the system knows what is there as the boot sequence continues and is later written to the registry so the system can have a record of the hardware configuration at boot time. Next, **Winload.exe** begins loading the files that are needed to start initializing the kernel. It will also begin reading in the device drivers of the hardware devices connected to the host.

An important part of what **Winload.exe** does at this point is to enforce the Kernel Mode Code Signing (KMCS) infrastructure. The KMCS is there to ensure that all the drivers and components that **Winload.exe** reads in are digitally signed to maintain the security of the system as it boots.

Systems that use UEFI BIOS do things a bit differently. In UEFI-based systems, the boot code is stored and executed in the UEFI firmware itself. Therefore, the system never goes into the real mode memory management system. Instead, it goes directly to protected mode, which improves the host’s security at boot time.

The last thing **Winload.exe** does is call up Ntoskrnl to initialize the Windows kernel memory and set up the Hardware Abstraction Layer (HAL) so the system can communicate with the connected hardware. Once the kernel is initialized, control is handed off to the Session Manager Subsystem (SMSS), which is in charge of initializing the user environment. The SMSS reads in values from the registry files to prepare the system to present the environment in the state that it was last left in. The SMSS also invokes Winlogon at this point to wait for a user to log in. When a user successfully logs in, Winlogon reads the registry keys for the user to prepare the user’s desktop environment.

**Starting Services at Boot Time**

Part of the system initialization process is to start the services that the system needs to run, including system services and services required by user-installed applications. The user space initialization process, handled primarily by SMSS and Winlogon, reads the registry keys that identify which services to load and which to start. Much of this information is stored in the Windows registry.

Registry hives can influence the services and applications that automatically start:

* **HKEY\_LOCAL\_MACHINE**: This hive stores systemwide information. Services that are registered in this hive start with each boot.
* **HKEY\_CURRENT\_USER**: This hive stores information about the user that is currently logged in. Services in this hive start when the user logs in.

In each hive, various registry locations define the services to auto-start. Some may not be present in a default installation because they are created by applications, but the following list includes services that you are likely to encounter:

* Run
* RunOnce
* RunServices
* RunServicesOnce
* Userinit

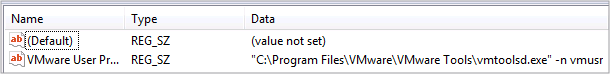
You can add these keys to these locations yourself, or applications that you install may populate them automatically if they are required for the application to run. These are also interesting areas to investigate if you suspect some unwanted application is automatically starting at boot time or after you log in. For example, use the registry editor to view applications or services that start when the system boots, in the HKEY\_LOCAL\_MACHINE hive:

[HKEY\_LOCAL\_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run]

Entries for any application or services that auto-start at boot time are in this location. 64-bit Windows installations also reserve a registry location for starting 32-bit applications, as follows:

[HKEY\_LOCAL\_MACHINE\Software\WOW6432Node\Microsoft\Windows\CurrentVersion\Run]

The figure below is an example of an entry in these registry locations:



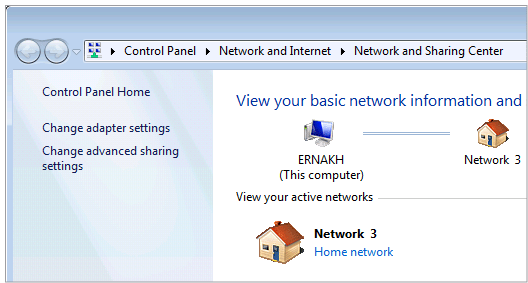
In the example, there is a registry key to start VMware Tools. Note that it points to the fully qualified path of the file that starts the service, and includes any command-line parameters that the system needs to start the service.

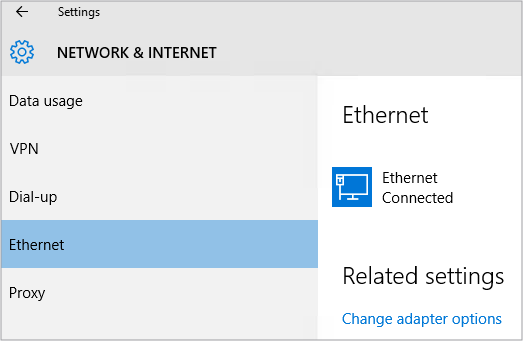
**Windows Networking**

One of the first tasks to perform when setting up a Windows host is to configure its networking properties. You should also be familiar with the basic tools for testing whether your network configuration is working properly. With the network configuration in place and operational, you will want to begin using network resources to get things done.

**Configuring Basic Networking Properties**

As with many Windows features, there are several ways to access network properties. One way is to click the network adapter icon in the system tray, which opens a list of options that are related to the type of network adapter that you are using. You will also see an option to open the Network and Sharing Center (the precise verbiage may differ among Windows versions). From the window that opens, select change adapter settings. The first figure below shows a portion of this window from a Windows 7 installation. The second figure shows a portion of this window from a Windows 10 installation.

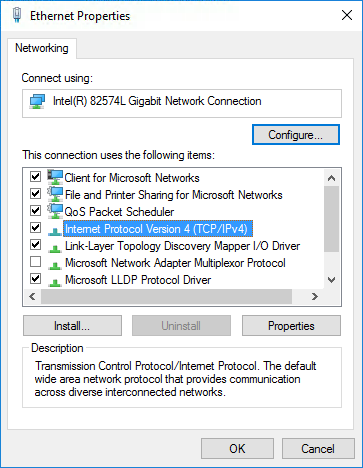




**Network Adapter Properties**

In either Windows 7 or Windows 10, select Change adapter options to open a window that shows each of the network adapters that you have installed on the host. Right-click the icon of the adapter that you wish to configure and select Properties from the context menu to open the properties box for the adapter.

From the properties box, select the IP version that you wish to configure and click the **Properties** button. The example below uses IPv4.



With the IPv4 properties window open, you can enter the values that are appropriate for your network, or choose Obtain IP address automatically to get your IP information dynamically from a local DHCP server. If you choose to statically assign the network values, you must enter the following:

* **IP address**: The IPv4 address that you wish to assign to the network adapter.
* **Subnet mask**: The subnet mask that you wish to assign to the network adapter.
* **Default gateway**: The IP address of the default gateway that the network adapter will use to access networks beyond the local network.
* **Preferred/alternate DNS server**: The IP address of your DNS server. You can enter an alternate server here as well.

When you complete the configuration, click **OK** to accept the changes.

The **netsh.exe** tool can also configure networking parameters from the cmd shell. This scripting utility can display and modify the network configuration of the system.

**Testing Network Connectivity**

After configuring your network settings, test the host to make sure that it has connectivity to the local network, and if the host requires access to external resources or the Internet, test that as well. Two utilities are available to quickly perform these tests from the command line. First, to test basic connectivity to the network, use the **ping** command. The **ping** command sends an ICMP echo request message to the remote host that you are testing with. The example command-line session follows:

C:\Users\administrator.SFSNORT>**ping 192.168.133.253** Pinging 192.168.133.253 with 32 bytes of data: Reply from 192.168.133.253: bytes=32 time<1ms TTL=64 Reply from 192.168.133.253: bytes=32 time<1ms TTL=64 Reply from 192.168.133.253: bytes=32 time<1ms TTL=64 Reply from 192.168.133.253: bytes=32 time=4ms TTL=64 Ping statistics for 192.168.133.253: Packets: Sent = 4, Received = 4, Lost = 0 (0% loss), Approximate round trip times in milli-seconds: Minimum = 0ms, Maximum = 4ms, Average = 1ms C:\Users\administrator.SFSNORT>

The **ping** command in Windows issues four ICMP echo request messages. If the host that is targeted in the command replies, you will see one reply for each request. Then the command will return some general statistics of the ICMP traffic that is generated in the session. This simple test has confirmed that you have connectivity at least to the local network. Test connectivity to networks that are beyond the local network by pinging hosts in other networks.

**The nslookup Command**

You should also test DNS to make sure it is working properly, because applications that use network resources will likely reference hosts by their names, rather than their IP addresses. In Windows, use the nslookup command to quickly test your DNS settings as follows:

C:\Users\administrator.SFSNORT>**nslookup cisco.com** Server: r1.sfsnort.com Address: 192.168.133.253 Non-authoritative answer: Name: cisco.com Addresses: 2001:420:1101:1::a 72.163.4.161 C:\Users\administrator.SFSNORT>

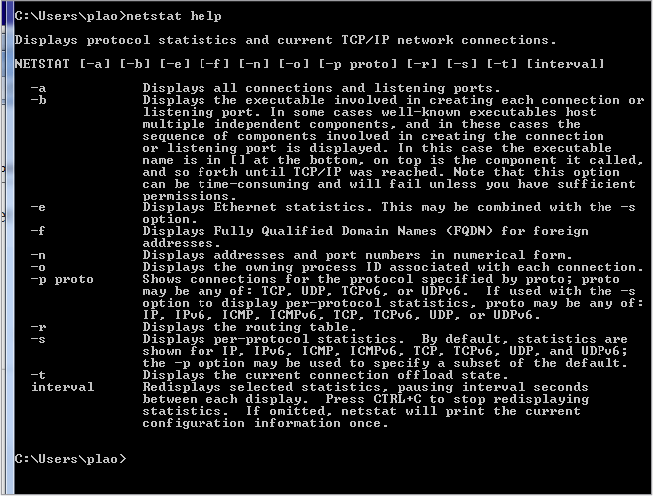
This command simply uses the DNS servers that you configured in the network properties to resolve the name of the host that you specified in the command to its IP address. If it returns an address, you know that DNS is working properly on the host.

With both IP connectivity and DNS resolution confirmed, the host is properly configured to use network resources. As the administrator of a Windows host, you can use other tools to check other aspects of networking, such as ports and services that the host is running. The netstat tool that ships with Windows allows you to do just that, and lists connections to and from the host.

**Windows netstat Command**

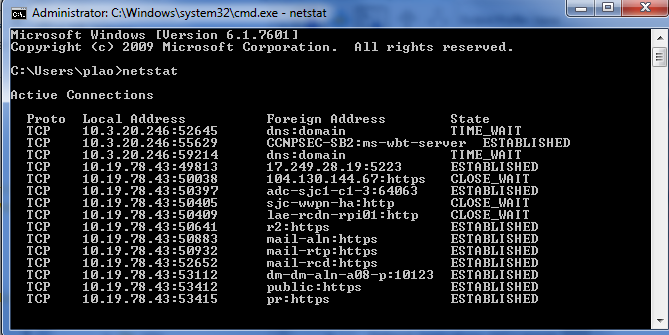
With the IP connectivity and DNS resolution confirmed, the host is properly configured to use network resources. As the administrator of a Windows host, you can use other tools to check other aspects of networking, such as ports and services that the host is running. The **netstat** command allows you to do just that, and list connections to and from the host.

If malware is present, the **netstat** command is a useful analytical tool for recognizing abnormal inbound or outbound network connections. Malware will typically open backdoor network communication ports on the host to generate outbound traffic to CnC servers under the attacker’s control. The figure below shows the help screen for the **netstat** command with all the options.



The **netstat** command displays active TCP connections—ports on which the computer is listening—Ethernet statistics, the IP routing table, IPv4 statistics (for the IP, ICMP, TCP, and UDP protocols), and IPv6 statistics (for the IPv6, ICMPv6, TCP over IPv6, and UDP over IPv6 protocols).

When used without parameters, **netstat** displays active TCP connections.



By examining the active TCP connections, an analyst should be able to determine if there are any suspicious programs that are listening for incoming connections on the host. You can also trace that process to the Windows Task Manager and cancel the process.

To link a connection that is displayed by **netstat** to a process that is listed in Windows Task Manager, run **netstat -abno** and find the appropriate process name listed in Windows Task Manager. If there are two or more processes with the same name, use the PID to find an exact match. From the Windows Task Manager Processes tab, click **View > Select Columns**, then check the PID box to display the PID.

**Accessing Network Resources with Windows**

Networking is integral to the Windows operating system. You can use all the typical client applications to access remote resources such as your browser application for web access or email client for email access like any other operating system. However, Microsoft pioneered the SMB protocol for sharing network resources. Though it was originally developed by IBM, Microsoft continued its development and has been a part of the Windows operating system throughout its life cycle. SMB also has an Open Source counterpart, which is known as Samba, that allows other operating systems to share resources with Windows-based hosts.

One of the primary uses of SMB is for accessing files or file systems on remote hosts. Windows uses the Universal Naming Convention (UNC) format to identify the remote resource, expressed as follows:

\\servername\sharename\file

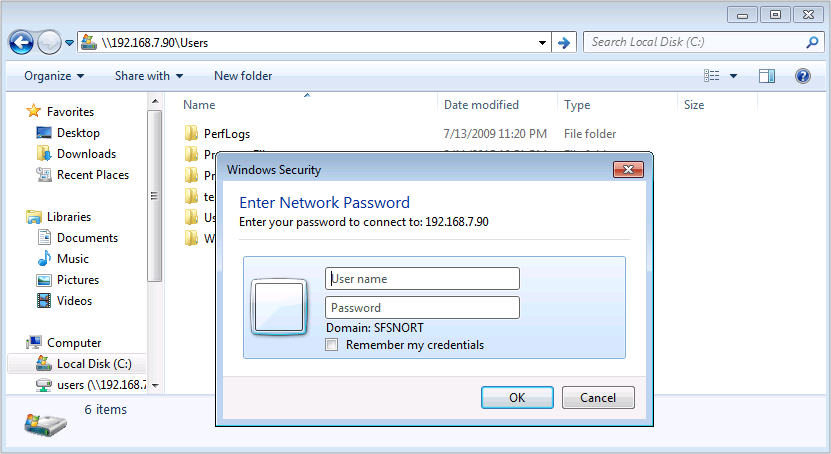
The **servername** component identifies the server hosting the remote resource. The server name can be an IP address, the host’s NetBIOS name (NetBIOS names are used to uniquely identify hosts in Windows networking environments) or the host’s DNS name (**ernakh.sfsnort.com** for example). The **sharename** component refers to the root of the file system that is reserved for sharing the resource. The **file** component is the actual resource the client is seeking. The file can also contain a path of subdirectories if that’s where the file resides.

Another note about the share name is that you must identify the portion of the file system that you wish to share. You can use access control features to restrict what remote users can do in the share, or grant them full permissions to read, write, and execute files from the share. **Kshares**, which are known as administrative shares, are a special group that allow only those users with administrative privileges to access these shares. Common administrative shares include shares that represent entire disk drives. For example, the share that represents the **C:** drive is called **C$**. The **ADMIN$** share points to the **c:\windows** directory. An administrative share that is called **IPC$** is used for interprocess communications.

Use the **net share** command to view the shares that are available on a Windows host.

C:\Users\administrator.SFSNORT>**net share** Share name Resource Remark -------------------------------------------------------------- C$ C:\ Default share IPC$ Remote IPC ADMIN$ C:\Windows Remote Admin Users C:\Users The command completed successfully. C:\Users\administrator.SFSNORT>

In the example above, the host is showing three administrative shares and a user-created share. The user-created share is called Users, and points to the **c:\Users** directory. So, users with credentials on the host can access that share from remote locations. One way to access the share is from the file explorer window. Simply enter the UNC name of the share and you will be prompted to enter the credentials on the remote host as shown in the figure.

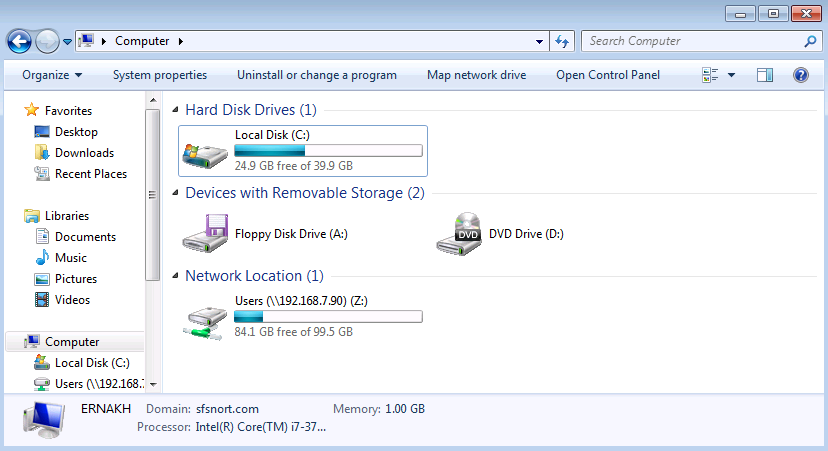


After entering the credentials, you will be able to access the files and folders stored in that directory on the remote host.

You can also mount the remote share as if it were a disk drive. From the perspective of the local host, it will appear as if it is just another disk drive. In reality, it uses the disk resources on the remote host. One way to mount the remote share is from the command line as follows:

C:\Users\administrator.SFSNORT>**net use z: \\192.168.7.90\Users /user:joe** The command completed successfully. C:\Users\administrator.SFSNORT>

In this example, the **net use** command was used to map drive letter **z:** to the share on the remote host. The username was also supplied as an argument in the command. If the user had previously mounted a share with valid credentials, the credentials are cached on the local host and the user is not prompted for a password. If the credentials are not cached, the user is prompted for a password before the operation can complete successfully. The result is a new icon in the file explorer that appears as another drive, as shown in the figure.

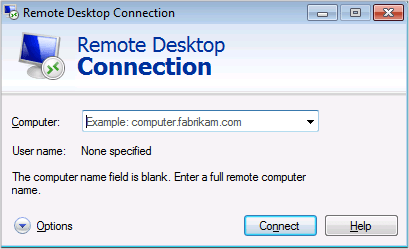


**Remote Desktop Protocol**

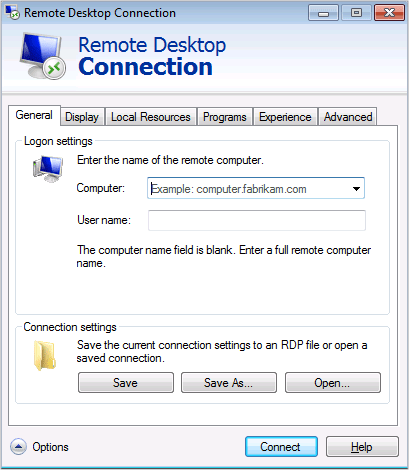
Another commonly used network resource is the ability to log in to remote hosts and remotely operate the host’s desktop. Windows ships with a client application that allows you to do this using the Remote Desktop Protocol (RDP). Professional and business-oriented versions of Windows only allow one remote session at a time. However, Windows server products with terminal services allow many concurrent sessions.

As a security analyst investigating security incidents on the remote Windows machines, RDP is often used to access the remote Windows machines.

The client application that you use to access a remote desktop is called Remote Desktop Connection. The following figure provides an example of the application interface.

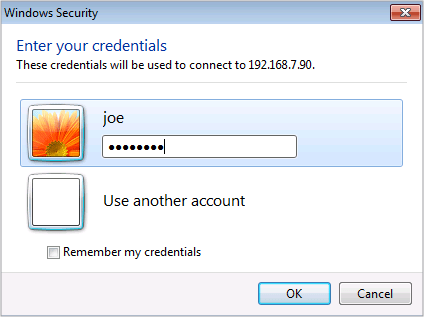


If you previously used the RDP client to connect to another host, you may see the host and user account already listed in the application window. If you want greater control over the connection, click the **Options** button to expose settings to customize the remote access experience.

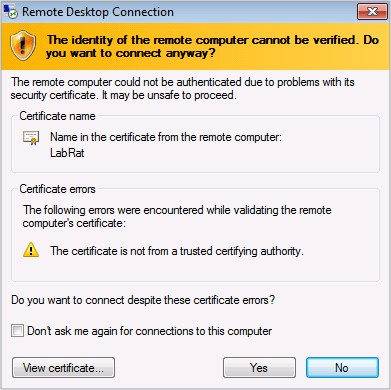


With the options exposed, you can control various aspects of the session, such as which computer to access, the username for logging in to the remote host, properties to display, or even if you wish to share any local resources, such as sound, files, or printers, with the remote desktop. Once you have configured the properties of the session, save the properties if you want to simply recall the session profile the next time.

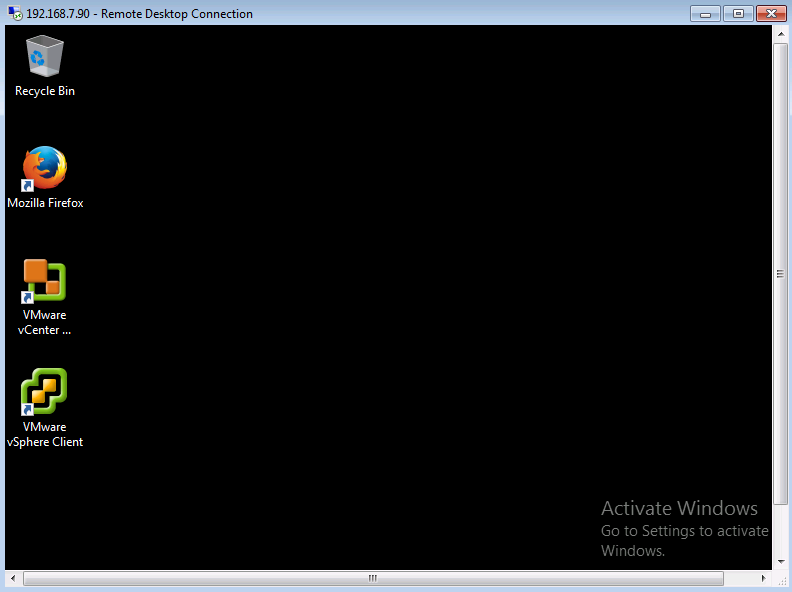
After you enter the host and username information, click the **Connect** button to initiate the session. If the user account is a valid account on the remote host, you will be prompted for the user’s password.



After providing the user credentials, you will see a window where the remote session will be built. You may see a warning message regarding the system’s inability to identify the remote system’s certificate. If a valid certificate is not installed on the remote system, but you trust the system, click the **Yes** button to proceed.

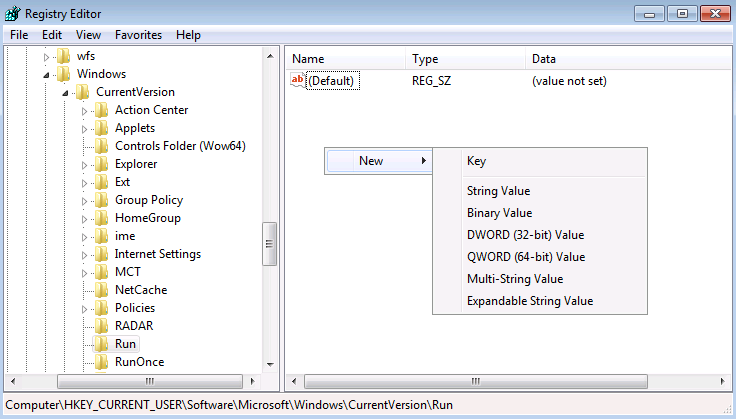


At this point, the remote desktop displays in the window, and you can now control the system as if you were sitting in front of it.



**Windows Registry**

Windows registry is a central hierarchical database in which Microsoft Windows stores information that is necessary to configure the system for one or more users, applications, and hardware devices.



The registry contains information that Windows continually references during operation, such as profiles for each user, applications that are installed on the computer, types of documents that each can create, property sheet settings for folders and application icons, hardware that exists on the system, and the ports that are being used.

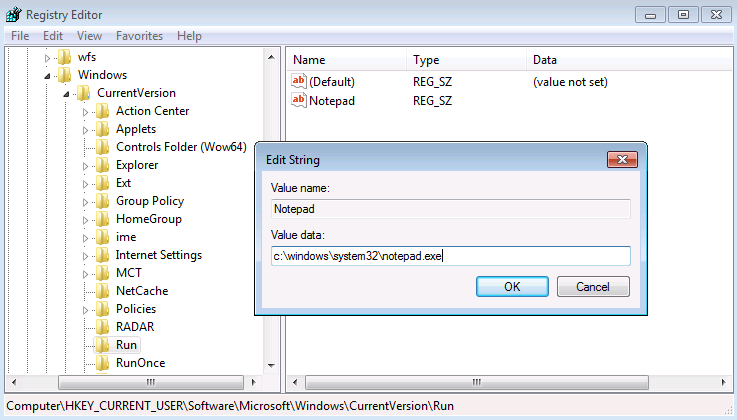
The registry is organized in a hierarchical manner. The highest element of the hierarchy is known as a hive, which maps to a file in the file system that contains a binary database of registry keys and values. Hives are designed to store specific types of information.

Hives and their intended purpose are described below:

* **HKEY\_CURRENT\_USER (HKCU)**: Stores data that is associated with the currently logged in user.
* **HKEY\_USERS (HKU)**: Stores information about all the user accounts on the host.
* **HKEY\_CLASSES\_ROOT (HKCR)**: Stores information about file associations and Object Linking and Embedding (OLE) registrations.
* **HKEY\_LOCAL\_MACHINE (HKLM)**: Stores system-related information.
* **HKEY\_CURRENT\_CONFIG (HKCC)**: Stores information about the current hardware profile.

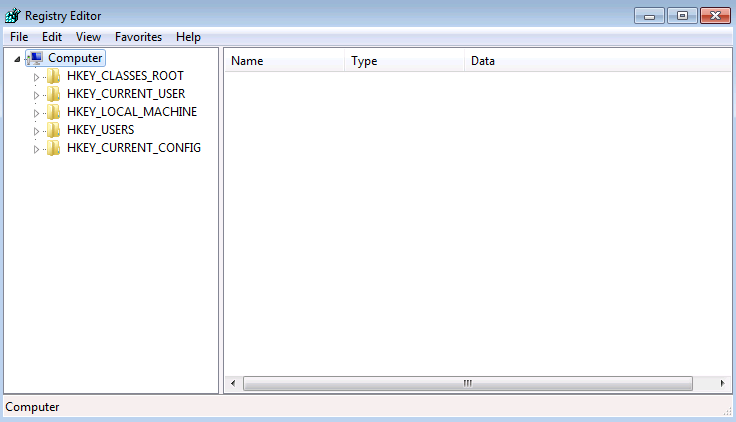
The names in the list are written in their actual long form as you would see them in the registry editor, but they also include abbreviations that are commonly used to reference a given hive in documentation or reference material. You cannot create or delete a hive, however, as an administrator, you can delete or modify registry keys and values.

The names of the hives are derived in the following manner: the letter H represents handles to keys; the format HKEY is followed by the specific name of the hive. For example, HKEY\_CURRENT\_USER contains the root of the configuration information for the user who is currently logged on. The user's folders, screen colors, and control panel settings are stored here. This information is associated with the user's profile.



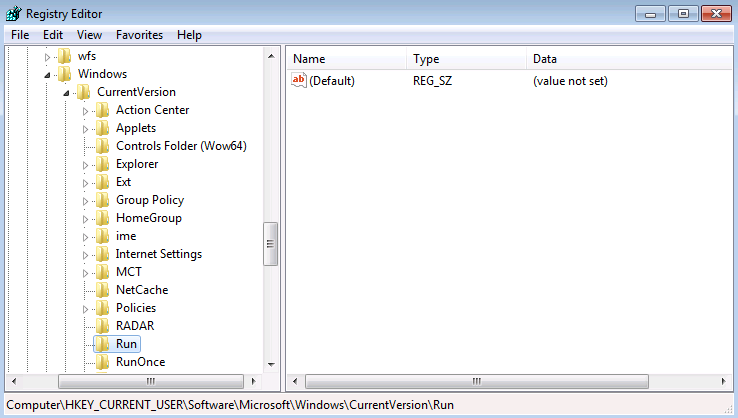
**Registry Editor**

The Windows registry cannot be modified by a non-administrative user on the system. Windows ships with an administrative tool that is called called **regedit.exe**, that will allow an administrator to view or edit the registry. An administrator can also use a tool that is called **regedt32.exe**, that will ultimately launch the **regedit.exe** tool. To open the registry editor, click the **Start** button and enter **regedit.exe** in the search field. An example of the registry editor is shown below:



To navigate the registry, use the same technique as the Windows file explorer: The left panel lists the hives and you click the triangular icon to drill into the structures within a hive. The main panel exposes a detailed view of an item that you select on the left. One difference between regedit and the file explorer is that the main panel only shows what is in the currently selected item. If that item has sub-keys, they will not show in the main panel, but they will be listed in the left navigation panel. The example that follows uses regedit to access a specific registry key:

(Computer\HKEY\_CURRENT\_USER\Software\Microsoft\Windows\CurrentVersion\Run).



The key path is rather lengthy, but you can see the folders that represent keys and their sub-keys in the left panel. Also note that the entire path is displayed in the lower portion of the regedit application window.

In this example, the Run registry key is used to cause programs to run each time a user logs on, which is a good place to look for potentially unwanted applications that start when you reboot the system.

In general, you should not modify registry settings unless you are an experienced Windows user or acting under the guidance of an experienced user. It is easy to induce a problem if you set something incorrectly or delete a critical registry key.

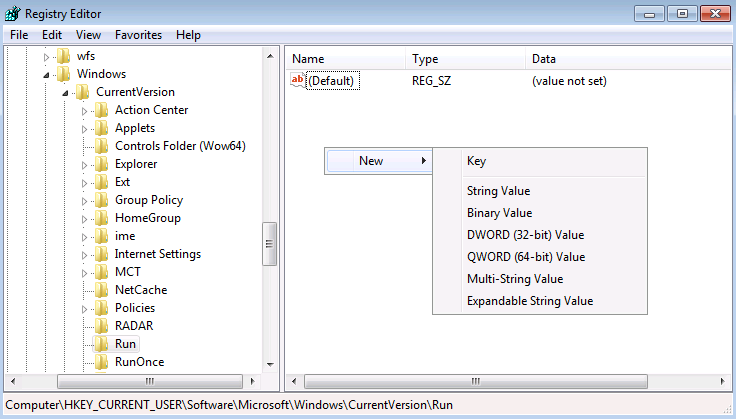
A registry key is a container object. It acts as if it is a folder in the file system. It can contain key/value pairs or sub-keys. Therefore, a registry key reference looks like a path in the file system, and keys and their sub-keys are delimited with the same backslash (\) character that you would use in a Windows file path.

A key can hold a value or another key. Values that are assigned to a key can be in several formats, so you can express values using various data types. Some of the more common types are listed below:

* **REG\_BINARY**: Stores numbers or Boolean values (on or off).
* **REG\_DWORD**: Stores numbers greater than 32 bits or raw data.
* **REG\_SZ**: Stores string values in Unicode format.

In the registry editor, the key/values displayed in the main panel indicate the format that the system is expecting as the value for a given key. For example, in the previous figure, the Run registry key is expecting a REG\_SZ value. The REG\_SZ value in this example is the blank default value.

Define the REG\_SZ value by right-clicking the main panel to bring up the context menu. Select New, then the key type that you want to use.



Select **String Value** to add a value and you will be prompted to provide a name for the value. In this case, the user created a value that is named **Notepad**, and the value data is the path and filename to the Notepad executable.

**Security Concerns Regarding the Windows Registry**

The Windows registry is a critical component to the security of a system. The registry controls virtually every aspect of the system’s configuration and operation. For example, malicious or potentially unwanted applications, such as browser toolbars, may add entries to the registry keys that start up applications at system boot time. A security administrator who is remediating an infected host may want to review the system registry keys to see if there are unwanted applications. Registry keys can be removed if the security administrator determines that they are malicious in nature.

Another key aspect of the registry is that it tracks much of the activity that is performed by users and applications on the host. Forensic analysts will find tracked activity to be a very useful place to investigate. For example, you can track the history of USB storage devices by examining the HKLM\SYSTEM\CurrentControlSet\Enum\USBSTOR key. Here you will find a list of all the USB devices that have connected to the system at some point. It also includes information such as the vendor, product number, and serial number of the device—all valuable forensic information in advancing a cyber investigation.

**Windows Management Instrumentation**

Windows Management Instrumentation (WMI) is the infrastructure for management data and operations on Windows-based operating systems. You can write WMI scripts or applications to automate administrative tasks on remote computers. WMI also supplies management data to other parts of the operating system and products.

WMI can be used in all Windows-based applications, and is most useful in enterprise applications and administrative scripts. System administrators can find information about using WMI at the TechNet ScriptCenter, and in various books about WMI.

An example of a network security device using WMI is to retrieve the Windows user log in and log off security events from the Windows domain controller.

To avoid detection and to carry out broad commands on compromised systems, today's attacks often use WMI to connect to remote systems, modify the registry, access event logs, and execute commands. Besides the initial login event, remote WMI commands leave little evidence on the accessed system. Therefore, Windows administrators should follow the proper Windows guidelines to secure the WMI access.

WMI supports a limited form of security that validates each user before the user is allowed to connect to WMI, on a remote or local computer. This security is layered on top of the Windows operating system security. By default, only the local computer Administrator account has full control of the WMI services on the computer that is being managed. Members of the Administrators group have access to remote computers, but may not have access to all the data. Permissions can be changed by adding a user to the Administrators group on the managed computer or by authorizing users or groups in WMI and setting their permission level.

**Common Windows Server Functions**

Although Windows is used most often as a desktop computing platform, its server platforms are used extensively in enterprise data centers. Windows Server is actually a brand name that refers to the Microsoft family of server products, starting with Windows Server 2003. Windows Server products host many popular services, which make the platform suitable for fulfilling many roles in the enterprise.

Windows Server services include the following:

* HTTP, HTTPS, and FTP
* DHCP
* DNS
* File services: NFS, SMB, and DFS
* Hyper-V for hosting and managing virtual machines
* Terminal services for hosting remote desktop access
* Deployment services
* Group policy management
* Active Directory domain services control

Microsoft also offers its Exchange Server product for email and calendaring, but among the services that are listed above, one of the most common applications for Windows Server products is the Active Directory, which is a service that is used to manage Windows domain-based networks.

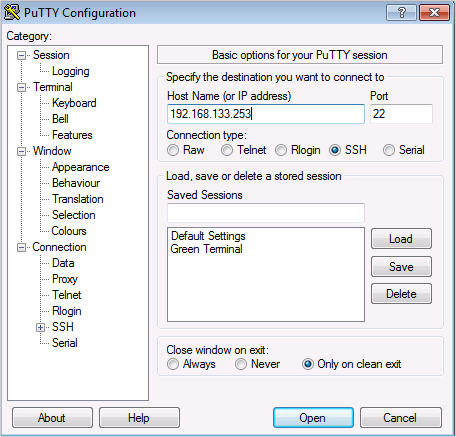
**Common Third-Party Tools**

Although Windows ships with many tools to manage and monitor the system, some users find that these tools lack specific features. Therefore, many Windows administrators install their own preferred tools. This section explores several popular tools in use today to manage and monitor systems. Some can be particularly useful in environments with various operating systems.

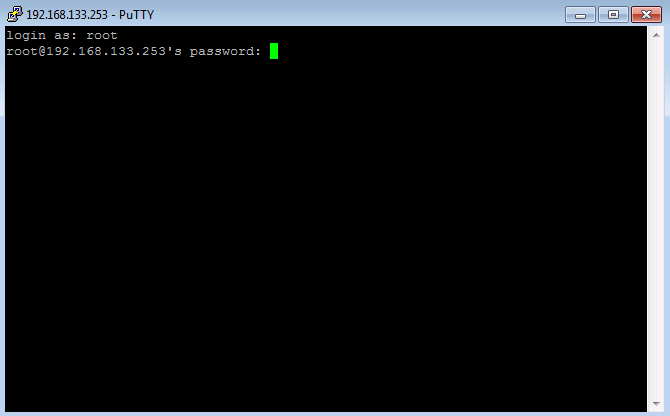
**PuTTY Application**

PuTTY is a very popular application that is on many Windows administrator desktops. PuTTY is an SSH client application. SSH is a protocol that is used to remotely access a host’s shell or command line. Linux operating system and Mac operating system X use this protocol because it encrypts the remote session, so that a third party cannot view the contents of the communication. Unfortunately, Windows does not ship with an SSH server or an SSH client. PuTTY is popular because it is free to download and easy to install. To download PuTTY, go to www.putty.org, find the version that matches your version of Windows, and install it.

PuTTY starts in an application window that allows you to configure your session. Usually, the default values suffice, so you really only have to enter the host that you want to reach.



After you enter the host information, click the Open button to initiate the SSH session in a terminal window. You will be prompted to enter the username and password to remotely log in to the system.



After you provide the log in credentials, you will be logged in to the remote host where you can begin executing commands from its command line.

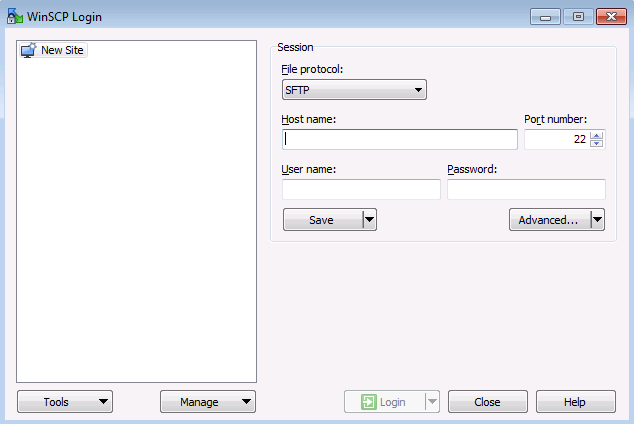
The features from PuTTY are natively supported in bash that runs on the Linux for Windows subsystem within the Windows 10 operating system. However, it is still in a limited beta deployment.

**WinSCP**

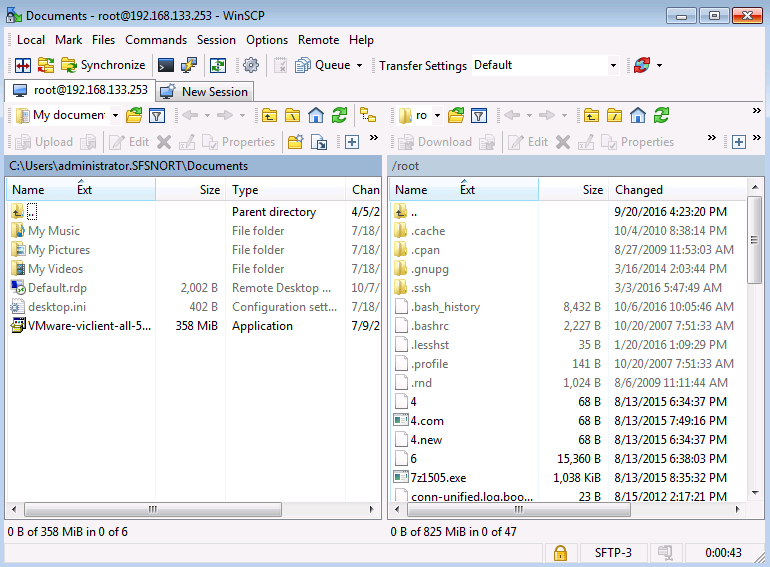
WinSCP is an application that is used to securely copy files to and from a remote host. If the host is running Linux or Mac OS X, it is probably running an SSH server that can be used to transfer files. This is another example of how the lack of SSH support on Windows can hamper an administrator’s efforts, especially in mixed operating system environments. WinSCP is a simple solution.

To download WinSCP, browse to www.winscp.net, navigate to its download page, and find the installation package that matches the version of Windows that you are running. It installs like any other Windows application and, for most users, the default values at install time will suffice.

Once the application is installed, double-click its icon to start the application. The first window that you see allows you to enter the remote host that you wish to share files with and the log in credentials for the host.



When you have entered the log in credentials and the remote host information, click the log in button to proceed. WinSCP presents a screen that is divided into two panels: on the left is your local host and on the right is the remote host. Navigate the file systems of each by double-clicking the folders that you wish to access. Simply drag-and-drop between the panels to exchange files. File transfers can be done in either direction: local host to remote host, or remote host to local host.

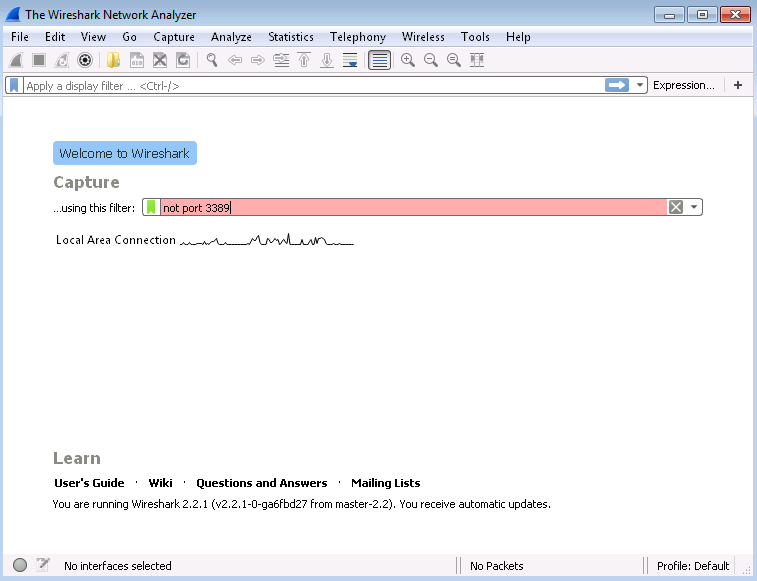


**Wireshark**

The Wireshark application is natively found on Unix-like OSs. Wireshark allows you to capture network traffic for further analysis. This feature is not natively found in Windows. However, the Wireshark application can be installed on the system to capture traffic on a Windows host.

To download Wireshark, go to www.wireshark.org, and navigate to its download page and find the installation package that matches the version of Windows that you are running. It installs like any other Windows application, however; you need to reboot your system when the installation completes.

Wireshark is a “feature-rich” application for capturing and analyzing network traffic. Details of its use are covered in other sections of the course. The figure below shows an example of the application window of Wireshark.



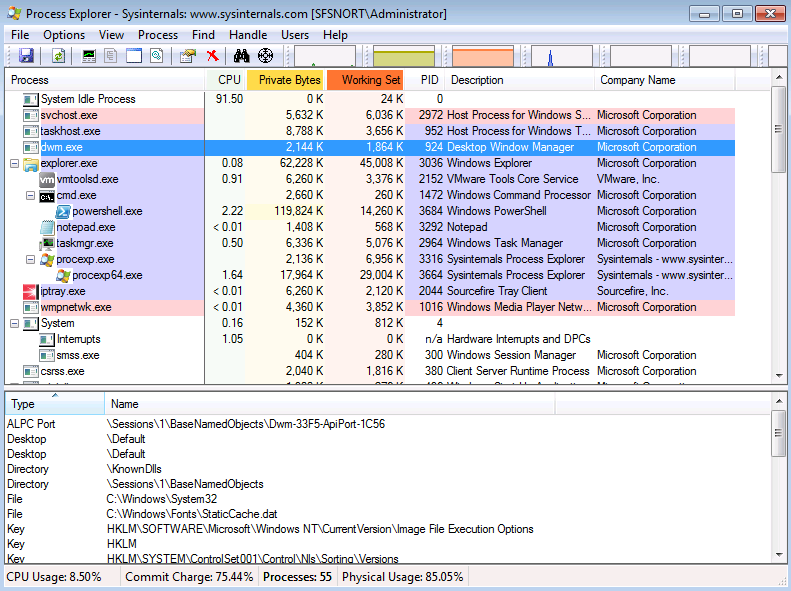
**Sysinternals**

Sysinternals is a package of tools that is specifically designed for the Windows operating system. The project was created by Mark Russinovich to overcome some of the shortcomings of the tools for monitoring and managing a Windows installation. There are over 70 tools in the entire package, and they have become very popular with Windows administrators over the years because they provide a higher degree of utility than their Windows counterparts. Microsoft acquired Sysinternals in 2006 and has since remained committed to maintaining the package, and making the tools available as a free download to anyone. Download the tools from http://technet.microsoft.com/en-us/sysinternals/default.aspx.

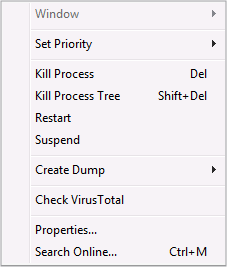
Sysinternal tools fall into three categories: tools to help programmers, tools for system troubleshooting, and tools for system management. With more than 70 in the set, this section focuses on the tools that are of most interest to a security professional.

To run many of these tools, you must have administrator privilege. Also, many of the tools are command line-based tools. To ensure that they run properly, open the command line as administrator so that what you execute in the command-line session runs with administrator privilege.

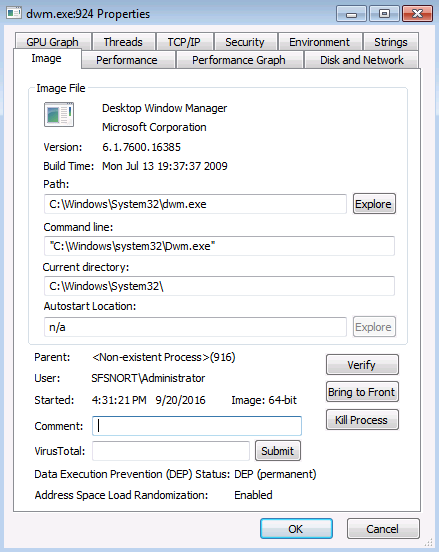
One of the more popular tools in the set is Process Explorer. It has a graphical interface and can best be described as a combination of the Task Manager and the Resource Monitor. An example of its application window is shown in the figure below. Process Explorer shows parent/child process relationships in a tree view.



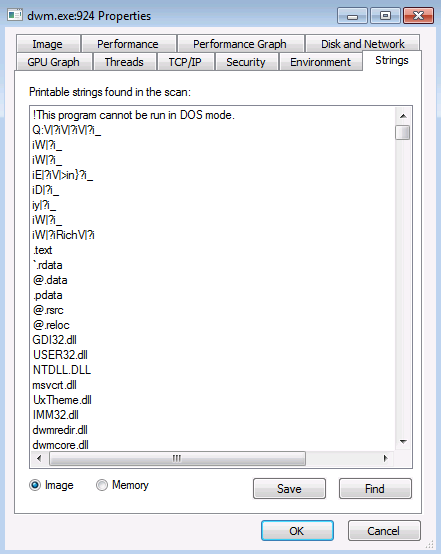
Right-click a process to open a context menu with several options, as shown in the next figure. Similar to Task Manager, you can select a process and kill it if it is misbehaving. You can also suspend or restart a process as well. If the process you are interested in is suspicious, select the **Check Virus Total** option to see if the file is known to be malicious. You can also select **Search Online** to initiate a web search that is based on the name of the process. Lastly, the **Properties** option opens a box detailing various properties that are related to the process.



As shown in the figure below, many categories of properties are available, depending on the process that you selected. The **Strings** category yields potentially interesting information.



The **Strings** tab exposes text strings that were found in the file that initiated the process. A string in this context is any sequence of three or more printable characters. Malware will sometimes embed text strings, such as IP addresses or URLs, in its code. For certain classes of malware, they could point to CnC sites or places to download other malware components.



Another feature of the Process Explorer tool is that it more accurately indicates CPU consumption. Processes that show less than 1% utilization do not appear to be idle. Many more features are worth exploring. Review the download site for more Sysinternals tools and documentation on each.

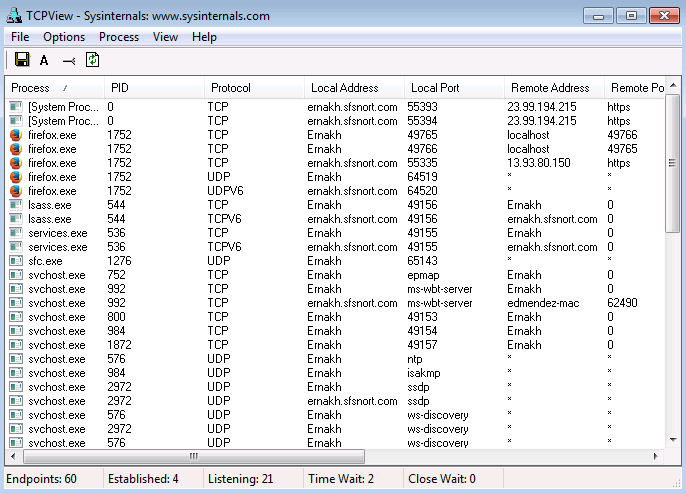
You can configure the tool to replace the Task Manager, which lets you launch this tool anywhere that you would normally launch the Task Manager, such as from the context menu of the task bar or the Task Manager option when you press **Ctrl+Alt+Delete**.

**Process Monitor**

Process Monitor is a utility provided by Windows Sysinternals that can monitor system activity and events. ProcMon can report and monitor attempts to access the Windows registry and can be used to identify registry keys and files that are accessed by an application. ProcMon can provide insight into what a process is doing on a system and can also be used to identify a suspicious process that may indicate the presence of malicious software. ProcMon requires administrative right privileges to execute on a system.

**TCPView**

The **TCPView** tool shows processes that are communicating with other hosts, which may be normal behavior for many processes. But some malware hijacks the memory space of a running process and disguises itself as a legitimate process. For example, if you see the Windows calculator application suddenly making outbound connections, you may want to investigate further by doing a search to see if the sites that it is communicating with are known to host malware or known CnC servers.



Right-click an entry in TCPView to open the context menu (shown below), which lets you take several actions. The Process Properties option opens a properties box that displays detailed information about a process. If you suspect that the selected process is malicious, you can end the process or close the connection. You can also run a **whois** search to find out who the registered owner of the offending IP address is.

