System Characterization Commands

System Characterization Commands	Description
date /t	Provides current system date
time /t	Provides current system time
tzutil /g	Provides current system time zone Note: Vista+ command
w32tm /tz	Displays detailed system time zone information
ver	Displays system version
net session	Displays users currently logged into the system
psloglist “security” -i 528 -s | find /i “Logon Type: 10”	Indicates the most recent users logged into the system (Windows XP/Server03)
psloglist security -i 4624 -s | find /i “Logon Type: 10”	Indicates the most recent users logged into the system (Vista+)
net statistics	Specifies how long the system has been running
systeminfo	Provides the following: Host name, OS name, OS version, OS manufacturer, OS Config, OS Build Type, Registered owner/org, install date, boot time, processors, BIOS, windows/system directory, locale, time zone, memory, domain, logon server, hotfixes, network information

Process Commands

Type	Description
Start from unexpected locations	Some processes may start from unexpected locations. For example, explorer.exe running out of C:\windows\system32 (or just about any executable running from %WINDIR%\temp).
Start sooner than expected	Generally, there are certain processes, including system services such as svchost.exe and smss.exe, as well as winlogon.exe, that you expect to see when the system is first coming up. Other processes starting around the same time as these could be questionable. You can determine this by reviewing timestamps and PIDs.
Possess unusual options or arguments	Viewing the process list using the syntax wmic process list (this may be easier to read if you redirect the output to a text file) enables you to see the Parent Process ID to determine what process spawned the process in question. Examine these processes to look for anomalies, such as a shell process (for example, svchost.exe or lsass.exe) descending from a service process.
Possess unexpected ancestors/PPIDs	Finding PPIDs is not as straightforward in Windows as it is in UNIX systems, so some creative command crafting using wmic or pslist may be necessary to view process lineage.
Are run by unexpected users	In some cases you may observe a process being run by an unexpected user. For example, explorer.exe or cmd.exe seldom run as SYSTEM – if seen doing so, they should be investigated further.

Exhibit strange or intentionally malformed names	Malware will often use randomly generated executable names, but there are also names intentionally chosen to look like and be mistaken for legitimate processes. For example, explorer.exe is a legitimate process; however, explorerer.exe is not a valid process.

Other known examples of intentional malformed names include:

- WINW0RD.EXE (with a zero "0" instead of the letter "O")
- iexplorer.exe (iexplore.exe is Internet Explorer and explorer.exe is the Windows environment)
- svchst.exe (the legitimate services process is svchost.exe)
- csrss .exe (the space between the executable name and the period [“.”] can easily be overlooked)

Network Commands

On a Windows machine, the syntax that gives the best info in one shot is `netstat -anob` where the options used are as follows:

System Characterization Commands	Description
-a:	Displays all active TCP connections as well as the TCP and UDP ports on which the computer is listening
-n:	Displays numerical addresses instead of trying to determine host, port, or user names
-o:	Displays active TCP connections and includes the Process ID (PID) for each connection; you can find the application based on the PID on the Processes tab in Windows Task Manager
-b:	Displays which process is involved in the connection; however, this only returns the executable name and not the full path of the executable

You can also view just TCP connections by using `-p TCP`. UDP connections are usually much less of a concern; however, to view them, you can use `-p UDP`. (Note: It is acceptable to leave out the -p option.) The Sysinternals Suite contains a command line tool called tcpvcon that can be used to return information about the open sockets as well; however, it does not necessarily offer any additional benefit over the built-in Windows tools.

Quick Checks

However, as a general rule, the most common locations for executables on a Windows system include the following (in order of likelihood):

- c:\windows\system32
- c:\program files
- c:\windows

Conversely, a good first indicator that you may be on the track of an anomalous process is when a process is running from locations, such as:

- c:\windows\temp
- c:\windows\user

Windows Timestamps

Filetime is a 64-bit value that represents the number of 100-nanosecond intervals that have elapsed since 12:00 A.M. January 1, 1601 Coordinated Universal Time (UTC). The system records file times when applications create, access, and write to files.

The NTFS file system stores time values in UTC format, so they are not affected by changes in time zone or daylight saving time. The FAT file system stores time values based on the local time of the computer. For example, a file that is saved at 3:00pm PST in Washington is seen as 6:00pm EST in New York on an NTFS volume, but it is seen as 3:00pm EST in New York on a FAT volume. Sometimes referred to as MAC times.

Modified/Written Time – changes/updated anytime a file/directory's contents are changed.

Access Time – changes/updates anytime the file/directory contents is touched to perform an action (including metadata).

Creation Time – changes/updated anytime a file/directory is created from scratch or a copy.

Not all file systems can record creation and last access times, and not all file systems record them in the same manner. Resolution of create time on FAT is 10 milliseconds, while write time has a resolution of 2 seconds and access time has a resolution of 1 day, so it is really the access date. The NTFS file system delays updates to the last access time for a file by up to 1 hour after the last access, while Modified/Written time is 2 minutes.

NTFS keeps track of lots of time stamps. Each file has a time stamp for 'Create', 'Modify', 'Access', and 'Entry Modified'. The latter refers to the time when the MFT entry itself was modified. These four values are commonly abbreviated as the 'MACE' values. Note that other attributes in each MFT record may also contain timestamps that are of forensic value.

`HKLM\System\CurrentControlSet\Control\FileSystem\NtfsDisableLastAccessUpdate = 1`

Has to be added to XP, Is there in Vista+

MAC times might be referred to as WAC times because:

dir /t displays:W Last written

C Creation

A Last Access

Note that FAT32 and VFAT volumes store the date but not the time of the last access. On these drives the time of last access will always be 00:00. The registry key for this is:

HKLM\System\CurrentControlSet\Control\FileSystem\NtfsDisableLastAccessUpdate = 1

A value if 1 is disabled.

- Has to be added to XP, Is there in Vista+
- It can take up to 55 minutes to update last access times, when enabled.

The attributes that store these times on an NTFS system are known as the $STANDARD_INFO attribute and the $FILE_NAME attribute. The $STANDARD_INFO attribute is the timestamp collected (or shown) by Windows explorer, timestomp and other standard utilities. This timestamp when displayed is also skewed to display the local time for that region unless the system has been configured for the GMT/UTC timezone. Time will also be effected by daylight savings. To determine if a system is configured to automatically adjust for daylight savings time and the offset from the UTC, the following registry key can be queried:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\TimeZoneInformation

Additional timezone information can be found in the following key:

hklm\software\microsoft\windows nt\currentversion\time zones\<TimeZoneKeyName>

where TimeZoneKeyName is the timezone the box is configured for. The key contains information such as when daylight savings time is configured to start and end.

Understanding the many ways timestamps can be skewed will prevent errors when normalizing to UTC. Normalizing all timestamps to UTC is important to keep a consistent timeline of events for each system.

The chart below lists the various ways the MACE time entries (also known as MACB (C for Change (Entry Modified) and B for Born instead of Created)) can change based on various processes applied to a file:

$STANDARD_INFO	Rename	Local Move	Vol Move	Copy	Access	Modify	Create	Delete
Modification (LastWriteTime)						X	X	
Accessed			X	X	X	X	X	
Change / Entry Modified *	X	X	X	X			X	X
Born / Created				X			X	

* Refers to changes in MFT entry, not the file itself. Not normally visible using native tools.

Timestamps can be useful for finding out more information. For example, you can use the timestamp of a suspicious/anomalous executable to find other things on disk that were created/added/modified around the same time that may be related, or provide more information.

Windows Service Control Manager

To interact with and review service configurations of a Windows host, Microsoft has developed the tool "**sc.exe**". sc.exe provides capabilities similar to Services in the Administrative Tools item in Control Panel and is installed by default on all modern versions of Windows. A full set of examples on the use and capabilities of the **sc** tool can be found at: https://web.archive.org/web/20160808113139/https://technet.microsoft.com/en-us/library/bb490995.aspx and also at: https://web.archive.org/web/20160721135112/http://ss64.com/nt/sc.html .

With the **sc** tool, you can create, modify, delete, start and stop Windows services on local or remote Windows hosts. However, for our purposes of Windows triage, you will mainly interact with options such as "**query**" or "**getdisplayname**".

A few useful examples of using **sc** for Windows triage are provided below:

sc getdisplayname *service_name* Provides the descriptive name of the service.

Example: "**sc getdisplayname wuauserv**" returns the value" Name = Automatic Updates

sc qc *service_name* Provides configuration information for a service.

Example" "**sc qc wuauserv**" returns information such as Start_Type, Path to the Binary that started the process, Display_Name, Dependencies and the account which started the service (Service_Start_Name)

Many of the sc arguments allow additional options. An example is the "query" argument:

sc query state= all will output all installed services, whether running or stopped.

sc query state= inactive will output only stopped services.

You can also use pipes to filter your queries. For example, to list the names of all the stopped services, you can use the following syntax:

sc query state= inactive | findstr "DISPLAY_NAME"