The Windows operating system

Created by: Zoltán MICSKEI

Last modification: 2017.03.20.

English translation: Gábor NASZÁLY
Edited by: Gábor Hullám

Version: 1.2.4

Budapest University of Technology and Economics
Department of Measurement and Information Systems

1 Introduction

In this laboratory exercise the structure and the basic management tools of the Windows operating systems is presented. The lab requires the prior knowledge of the Windows user interface.

Preparation for the lab is assessed at the beginning of the lab by a small test.

To prepare for this lab exercise, complete the followings:

1. Go through again the relevant lectures of the Operating Systems courses on Windows:
   a. http://mit.bme.hu/~micskeiz/opre/operating-systems.html
2. Read through the information provided in this guide!
3. Try to answer the questions at the end of this guide! This will help to assess your preparation!
4. **Try out the following tools** (further information can be found in chapter 5)!
   a. *VMware Player*: start a virtual machine, so some work, take a screenshot from it, try to copy files between the host and the virtual machine!
   b. *Sysinternals Process Explorer*: examine the elements in the process tree, and look at the properties of a process!

2 The Windows architecture

This section covers the architecture of the Windows operating system and its most important components.

2.1 The most important components of Windows

A simplified view about the structure of Windows can be seen on the following picture.

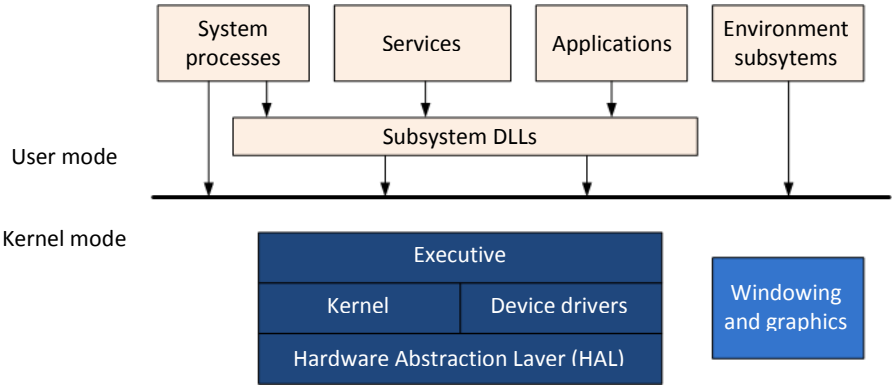

Figure 1: simplified Windows architecture

The most important components of the system:

Components running in **kernel** mode:

- *HAL*: its purpose is to hide the details of the underlying hardware and to provide a unified interface to the components above for operations on the hardware.
- *Kernel*: this component provides the most core functions of the operating system (like scheduling and interrupt handling). Hardware specific code can be found even in this component. This is obvious if we think about context switching. To accomplish a context switch one must know the registers of the underlying CPU. This component is contained in the file `ntoskrnl.exe`. (Note: in many cases the word kernel may refer to all of the kernel mode components.)

- *Device drivers*: these kernel mode modules translate general requests to hardware specific ones. The device driver model in Windows is layered. The device drivers are chained together. (For example it is possible to insert a module in between the devices drivers of the NTFS file system and the hard drive. This module can provide fault tolerance functions or implement various RAID structures transparently.) The device drivers are contained in files having `.sys` as extension.
- *Executive*: this layer provides the higher level functions of the operating system (like memory management and security). It stores data in objects. These objects can be accessed only by their *handles* through well defined interfaces. Every object has security information about who can access it and what operations are allowed to that entity. This is checked on every access by the system. However this component is also contained in the `ntoskrnl.exe` file it reaches the kernel only the interface provided by the kernel.
- *Windowing and graphics*: due to performance reasons the windowing and graphics subsystem is implemented in kernel mode. If one wants to draw to the screen it is necessary to access the display in kernel mode at the end. With this component implemented in kernel mode we need less amount of user mode to kernel mode switching. The file `win32k.sys` contains the functions of this component.

Components running in **user** mode:

- *System processes*: these are required to start and to operate the system however they run in user mode.
- *Services*: these processes are running in the background independently from the user interface and user logins. More than 50 services are shipped with Windows. They can implement various functions like network firewall or managing the printer pool. It is also possible to install custom services.
- *Environment subsystems*: at the time Windows NT launched there were three application programming interface provided: the applications were able to reach the operating system calls through POSIX, Win32 and OS/2 API. This was implemented by first providing a common, inner API by the *Kernel* (or more precisely by the *Executive*) and then the various environment subsystems use this API. The applications do not call the *Executive* directly. They use its functions through one of the subsystems.

Figure 2. presents us a more detailed view about the components of a Windows system. It is worth to notice the followings:

- You can identify the most important components of the *Executive* (like Virtual Memory, I/O Manager).
  - Only a subset of the functions – implemented by these components – is accessible from user mode. One can reach this subset of functions through `NTDLL.DLL`. This library accepts the system calls, checks the parameters then switches to kernel mode and forwards the requests to *System Service Dispatchernek*.
- The environment subsystems are listed:
  - Windows subsystem (`csrss.exe`),
  - The optional POSIX subsystem (psxss.exe). The former name of this component from Windows 8 is *Subsystem for Unix Applications* (SUA). This component was removed from Windows 10, therefore only the Windows subsystem is useable currently.
  - Only the Windows subsystem and the Session Manager call directly `NTDLL.DLL`. All of the other components are able to reach the operating system only through a subsystem DLL. These DLLs are `kernel32.dll`, `advapi32.dll`, `user32.dll` and `gdi32. dll` for the Windows subsystem.
- Services:
  - *SvcHost.exe* (Host Process for Windows Services): it is a general process to run certain services built in Windows. The aim is to spare resources by not having separate processes for all of the services. Only built-in services can be executed in this way (custom services not). The `svchost.exe` instances are distinguished from each other by a unique group name (like NetworkService, LocalServiceNoNetwork). This name indicates the user name under which these services are running and the access limitations they may have. Since Windows Vista the *Task Manager* is able to display the group name of an `svchost.exe` instance on the *Services* tab and also the services it hosts.

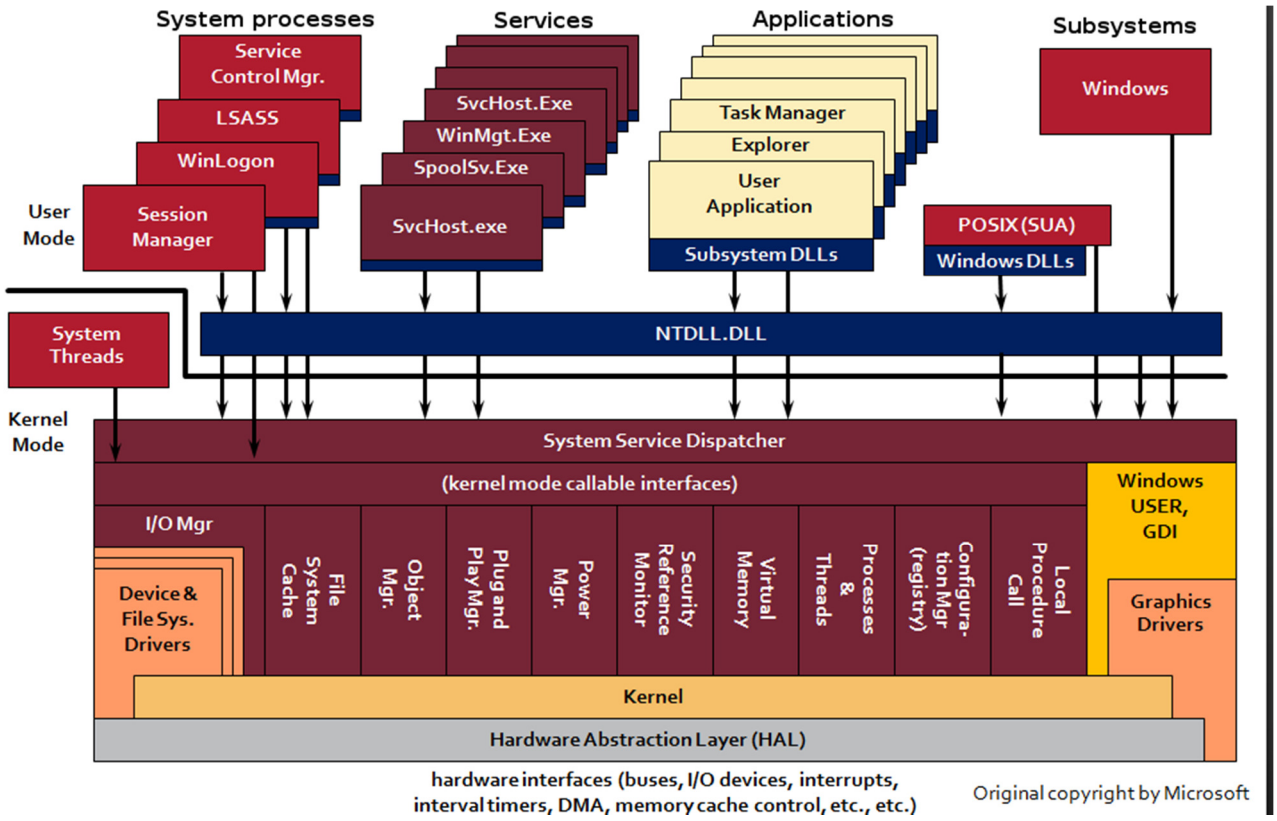

Figure 2: a more detailed view of the Windows architecture

- System processes:
  - Session Manager (`smss.exe`): this is the first process started in user mode in the system. Many users may be logged in to a system simultaneously. Their processes are separated from each other in *sessions.* The duty of `smss.exe` is to start and manage these sessions. Possible types of a session (see Figure 3):
    - Session 0: this is reserved of system processes and services. It has no user interface.
    - Console (Session 1): this is the session of the locally logged in user.
    - Remote sessions: remote logins can be accomplished by using the *Remote Desktop* protocol. The *Remote Desktop Connection* (`mstsc.exe`) application can initiate a session to a remote host. If the remote host runs a client Windows operating system the console session is logged out after a successful remote login. On server operating systems two remote sessions (for administrative purposes) are allowed to run besides the console session by default. If *Remote Desktop Services* component is installed one can have even more simultaneous user sessions.
  - *Windows Startup Application (`Wininit.exe`)*: it is started by `smss.exe` after it created session 0. Its purpose is to start the other system processes.
  - *Local Security Authority Subsystem (`lsass.exe`)*: this process implements the most important security functions like checking users and passwords, security logging and checking accesses.
  - *Service Control Manager (`services.exe`):* manages (start, stop, ...) the services. If a service is set to start automatically, this component starts it.
  - *Local Session Manager (`lsm.exe`):* manages remote connections.
  - *Windows Logon Application (`WinLogon.exe`)*: responsible for user logins.
  - *Windows Logon User Interface Host (`LogonUI.exe`)*: this process displays the welcome screen and offers the various login modes (like password or smart card depending on what has been installed on the system).
  - `Explorer.exe`: this is the graphics shell for users. After a successful login this process is started by winlogon by default (to be honest, this component is not strictly a system process).

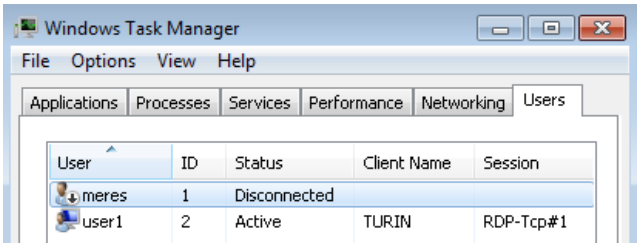

Figure 3: the console session is disconnected because of an active remote login.

On Figure 4. you can see a possible process hierarchy. As displayed in the Session column there are two active sessions in the system (in session 3 only `LogonUI.exe` is running and displaying the logon screen).

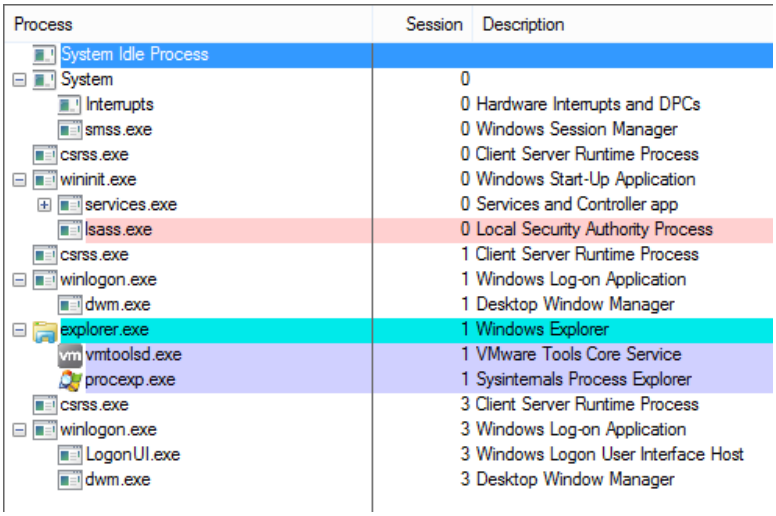

Figure 4: system processes in case of two sessions

The processes are able to reach the resources of the system (like a file, a process or shared memory) through so called *handles.* Before one can open a handle the system checks if the user has enough access rights to the object in question.

Now we have a basic view about the most important components of a Windows operating system.

2.2 Windows Store applications

One of the most significant changes in Windows 8 was the appearance of a new application type: the Windows Store applications (codenamed as Metro style applications previously). They have a new user interface, need to be installed in a different way, have a new life-cycle and there is also a new development environment belonging to them. A few properties of these new applications:

- new user interface (like full screen view, readiness for touch control)
- another installing model (the application arrives in a package what can be acquired (and later updated) from *Windows Store*)
- isolated running environment (there are strict access rules)
- special life-cycle (the OS may suspend applications in the background, and only via special system calls is it possible to do background operations)

Among the design goals of this new model we can find the easier use of tablets, a more efficient use of resources (like consume less power), less possibilities to broke and generally easier usage.

If you are unfamiliar with the new user interface it is worth to mention how to use the corners of the screen:

- Lower left corner: display the *Start screen*

- Upper left corner: switch to another Windows Store application
- Right corners (upper or lower): display the so called *charm bar*

Because the usage of the above mentioned corners are a little bit tedious when using virtual machines it is worth to use *keyboard shortcuts* instead (see appendix).

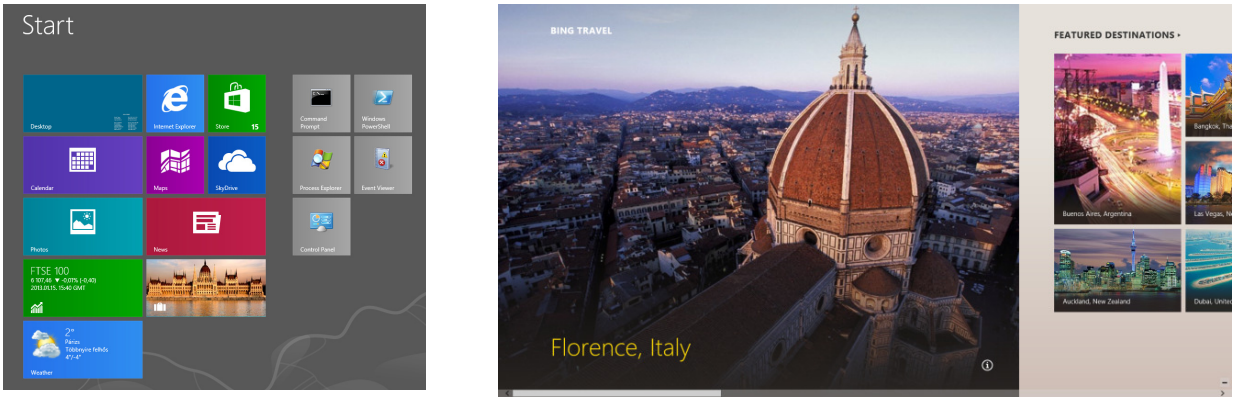

Figure 5: the new Start screen and a Windows Store application (Travel)

The figures below help us to understand the new life-cycle model. If we switch from a *running* application to another then the previous application will be *suspended* by the OS within a few seconds. If an application is about to suspend the OS informs it before and gives 5 seconds (maximum) to the application to save the most important aspects of its state. An application in the *suspended* state does not get CPU time but the memory content belonging to it does not perish immediately. But the OS may decide to free up some memory by *terminating* the application in case of the system is running on low amount of free memory. It is important to mention that this event is not signaled to the application! So it is important to save the state during the suspension process as during termination the application has no chance to do that. If we restart a terminated application whose state has been saved before, then that state can be restored.

On the figure to the right one can see a situation having more than one Windows Store applications started. Note that these processes are started not by `explorer.exe` but rather by a background service running in an `svchost.exe` instance. The user started `SystemSettings.exe` (an application to manage system settings having the new interface), `Map.exe` (the Map application; note the path) and `WWAHost.exe` (in which the Travel application is running). Among these two are already in the suspended state and Travel will also be suspended soon as we just switched from it.

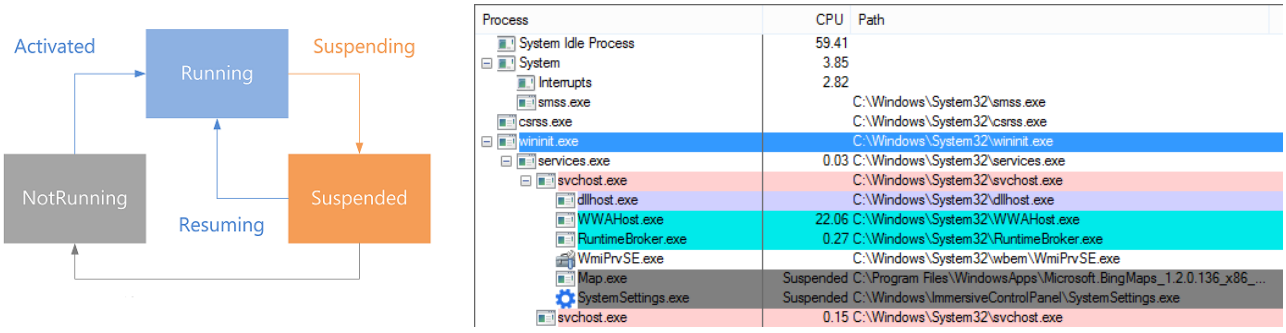

Figure 6: the life-cycle of Windows Store applications

We have only scratched the surface of Windows Store applications. If you are interested in further details we can advise online documentations like MSDN.

2.3 System folders

On figure 7. you can see what are the folders on a newly installed system.

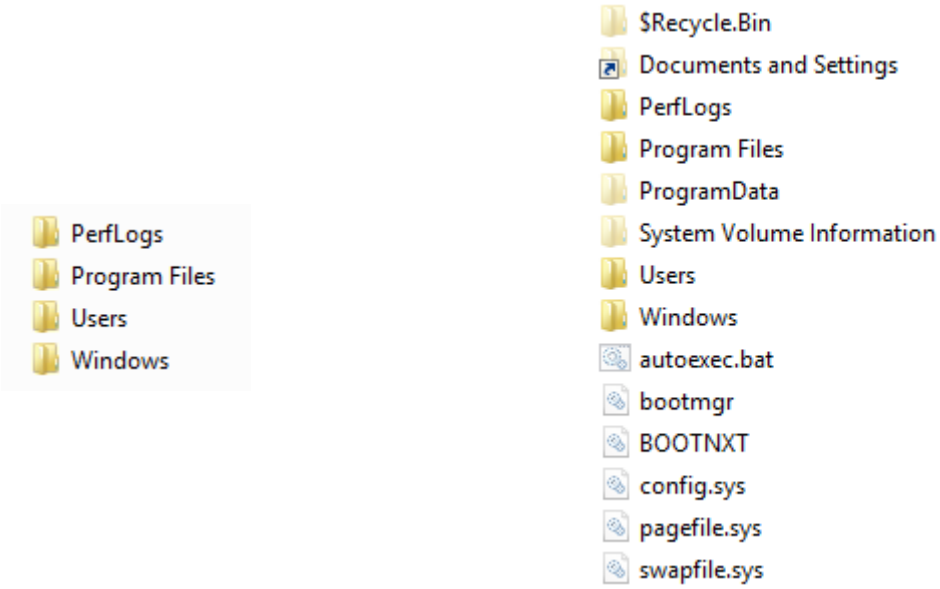

Figure 7: folders created during installation (with and without hidden files)

- *$Recycle.Bin*: under this folders are separate folders for the users to store their recycle bins.
- *Boot*: this folder holds information on how to start the system (BCD, Boot Configuration Database). If we installed the OS to a brand new disk then this folder (and also the one named *Recovery*) are placed to a separate partition (350 MB).
- *Users*: the user profiles (user specific settings and date) are stored here. On older systems this folder was named *Documents and Settings*. For compatibility reasons a folder (actually a link[1]) even on newer versions of Windows is created with this name pointing to the Users folder..
- *Program Files*: the default place for installed programs. Note that the users have only read rights to this folder (to not allow any attempt to replace the programs with potentially malicious versions). So the applications are not allowed to write settings or data to this folder during normal execution (only during installation are they allowed to change the content of this folder). On 64 bit systems this folder is used for 64 bit programs only, and a separate folder named `Program Files (x86)` is created for 32 bit programs.
- *ProgramData*: this folder holds all the common data of the applications (so the data that is not specific to any user). Many of the subfolders here are just links to the corresponding folders in `\Users\Public`.
- *System Volume Information*: *System Restore* feature uses this folder to store restoration points.
- *Windows*: programs, folders, settings and others belonging to the system. Some of its most important subfolders:
  - *Installer:* MSI (*Microsoft Installer*) executables belonging to the installed application to change or remove them.
  - *SoftwareDistribution:* the downloaded updates are stored here temporarily.
  - *System32*: most of the programs and DLLs belonging to the system reside here.
    - *drivers*: device drivers known by the system.
      - *etc*: among others one can find the *hosts* file here, in which it is possible to bind domain names to IP addresses statically.
    - *LogFiles*: some system components store their log files here (in case they do not use the *Event Log* facility).
  - *Winsxs*[2]: the so called component store.

[1] NTFS supports *symbolic links* (one can create them for example with the *mklink* command)
[2] More on this topic: Engineering Windows 7, http://blogs.msdn.com/e7/archive/2008/11/19/disk-space.aspx

The files in the root directory:

- `autoexec.bat` and `config.sys`: remnants of the old DOS days…
- `bootmgr`: reads BCD and displays the boot menu. Loading the system is beyond the purpose of this file. That is accomplished by `%SystemRoot%\System32\Winload.exe`.
- `hiberfil.sys`: during hibernation the system saves the content of the memory here.
- `pagefile.sys`: the paging file. The system may put memory contents here temporarily.

It is worth to investigate the structure of Users folder also. Here are the user profiles stored (by users in separate folders). Upon system install the profiles below are created automatically:

- *Default*: if a just created user logs in at the very first time his or her profile is created by copying the default profile.
- *Public*: the entities located here in the *Desktop* and *Start Menu* subfolders are presented to all of the users. The other subfolders are for storing common files.
- *Administrator*: the profile for the administrator user.

Due to compatibility reasons the folders named *All Users* (→ *\ProgramData*) and *Default User* (→ *Default*) are also created (as links).

Within a profile usually the following items can be found:

- *AppData*: the user specific settings of applications. In a networked environment it is also possible to use *roaming profiles*. In this case the profile is stored on a server. Upon login it is downloaded to the local machine and upon logout the changes are saved and stored on the server. This way the user can see the same environment from any computer. For these reasons the *AppData* folder has the following subfolders:
  - *Local, LocalLow*: those settings that do not need to be stored in a roaming profile are stored here (like the *Temp* folder or the *Temporary Internet Files*).
  - *Roaming*: those application settings that need to be accessed from any of the computers.
- *Desktop*: folders and files on the user's Desktop.
- *Contacts, Documents, Downloads, Favorites, Links, Music, Saved Games, Searches, Videos*
- *`ntuser.dat`*: this file contains the registry data specific to the given user.

2.4 Services

The services are processes that run in the background independently from the currently logged in user. They can be managed here: *Control Panel / System and Security / Administrative Tools / Services*.

On the summary screen the following information can be seen about the services:

- *Name*: this is the long (display) name of the service.
- *Description*: a short description on the purpose of the service.
- *Status*: started or not.
- *Startup Type*: Manual / Automatic / Disabled. There is also an option (Automatic (Delayed Start)). This latter startup type instructs the system to start this service only after all of the automatic services have been started. In this way these services do not slow down the login process of users. In the newer version it is even possible to start a service only upon a specified event (trigger).
- *Log On As*: the user under whose name the service is about to run. Possible options:
  - *Local System:* this user is not capable of interactive login but has greater privileges even than the *Administrator*. In fact it has all local privileges (can access the processes of other users, can read the security database and so on.
  - *Local Service*: a bit weaker than the *Local System* but still a user having a lot of privileges.
  - *Network Service*: this user has the least privileges among this three.

- *Ordinary user*: it is even possible to select any valid user name on the machine (in case the user in question has the *Log on as a service* privilege).

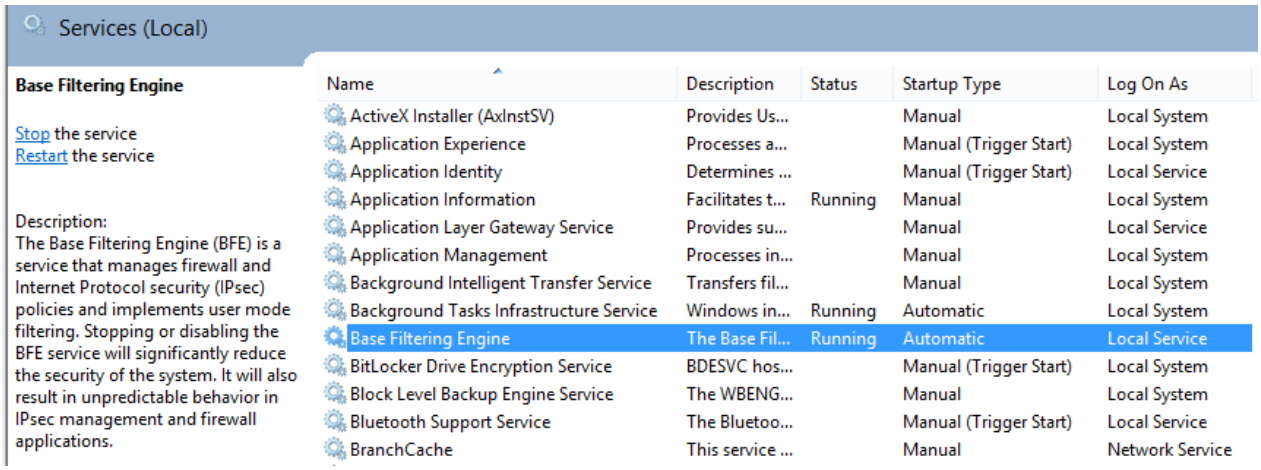

Figure 8: Services control panel

More information can be found on each service in the corresponding properties dialog.

- *General* tab:
  - *Service name*: the short (inner) name of the service.
  - *Path to executable*: the path to the executable that implements the service. On the picture below we can see a built-in service which is run by the LocalServiceNetworkRestricted instance of `scvhost.exe`.
- *Log On* tab: here we can choose the user name under which the service shall run.
- *Recovery* tab: what shall be done if the service fails (like restart the service or running any other program).
- *Dependencies* tab: here we can see all of the services this service depends on (services that have to be started before our service) and all of the services that depend on this service (services that cannot start if this service is not running). These values have to be filled in during the install phase of the service.

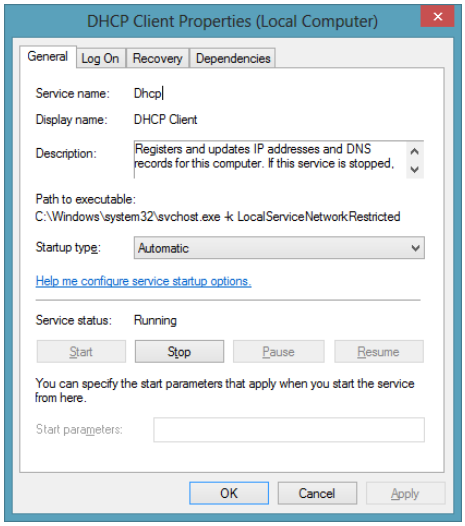

Figure 9: properties of a service

The services store their settings in the *registry* under the key:
`HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services`.

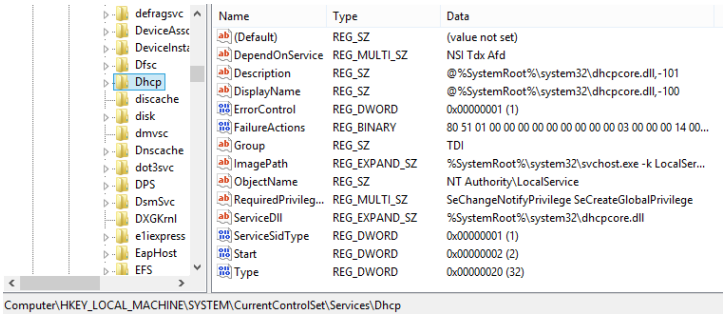

Figure 10: the settings of a service in the registry

We can manage services from the command line too by the *Get-Service* PowerShell cmdlet.

2.5 Advanced Startup Options

In Windows 8 the former boot process and the access of the advanced startup options have been redesigned[3]. Previously, in older systems, we needed to press F8 during system start. But the systems nowadays can start so fast that this method is not reliable enough anymore[4]. For this reason, since Windows 8, we need to make choices in *Advanced Startup Options* to tell the system to show the startup menu during next boot (Figure 11). Certainly if the system was unable to start normally this startup menu will be presented to the user automatically.

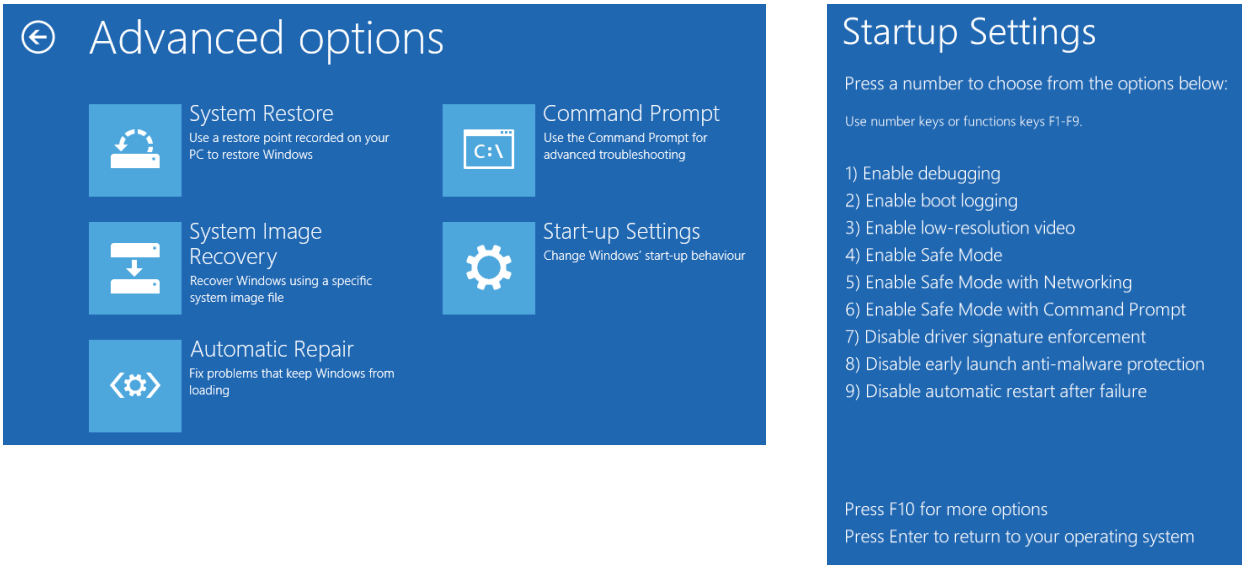

Figure 11: advanced startup options

In case of *Safe Mode* the system loads only the basic, built-in drivers. This way if the system was unable to start in normal mode due to a faulty driver or service then in safe mode we can remove it.

3 System management

In the following sections we investigate the most important technologies and tools that can be used to operate and manage the system. Surely you have met a few of them before but it is worth to get familiar with them more thoroughly.

[3] More on this: Building Windows 8 blog, Reengineering the Windows boot experience, URL: http://blogs.msdn.com/b/b8/archive/2011/09/20/reengineering-the-windows-boot-experience.aspx
[4] More on this: Building Windows 8 blog, Designing for PCs that boot faster than ever before, URL: http://blogs.msdn.com/b/b8/archive/2012/05/22/designing-for-pcs-that-boot-faster-than-ever-before.aspx

3.1 Microsoft Management Console

Most of the management tools have similar interfaces. This is because they are just a module (so called snap-in) to a common console called *Microsoft Management Console* (MMC). The console has three main parts: (1) on the left a tree structure showing the modules and their parts, (2) in the middle the details of the currently selected module can be seen, (3) while on the right we can see the possible operations on the selected item.

After we started MMC (`mmc.exe`) snap-ins can be added by pressing Ctrl+M (see Figure 12.).

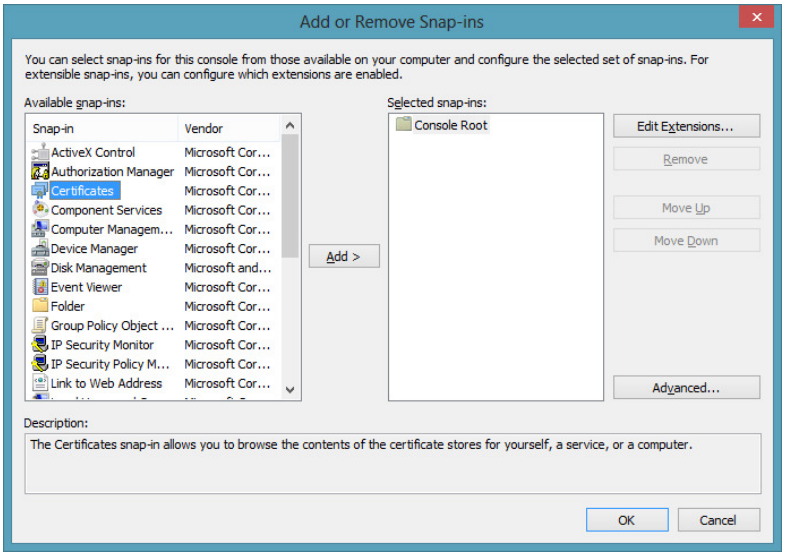

Figure 12: adding a snap-in to MMC

After selecting a module to be added we also have to tell if we want to manage the local or a remote computer (most of the management operations can be done on remote machines too).

3.2 Eventlog

The *Eventlog* is the central logging facility of Windows. It can be started from *Control Panel / Administrative Tools* or can be directly added to a MMC console. After we started the *Event Viewer* a summary view is presented to us showing the most important events recently happened grouped by the level of severity (Figure 13.).

The built-in logs worth to mention (in the *Windows Logs* part):

- *System*: system messages (like startup, problems with services).
- *Security*: security audit logs (who accessed which object with which privileges; this log can be viewed only with proper permissions)
- *Application*: other built-in or third-party applications can write their messages to this log. Since Vista many of the applications (like Backup, Task Scheduler) has a dedicated log under the *Application and Service Logs*.

Since Vista the Event Log has been changed: it stores the events in a new format (an XML based format having *evtx* as extension), can be filtered and searched and under the *View* menu it is possible to sort and group the events. Furthermore it is even possible to collect events from other machines too (*Subscriptions*).

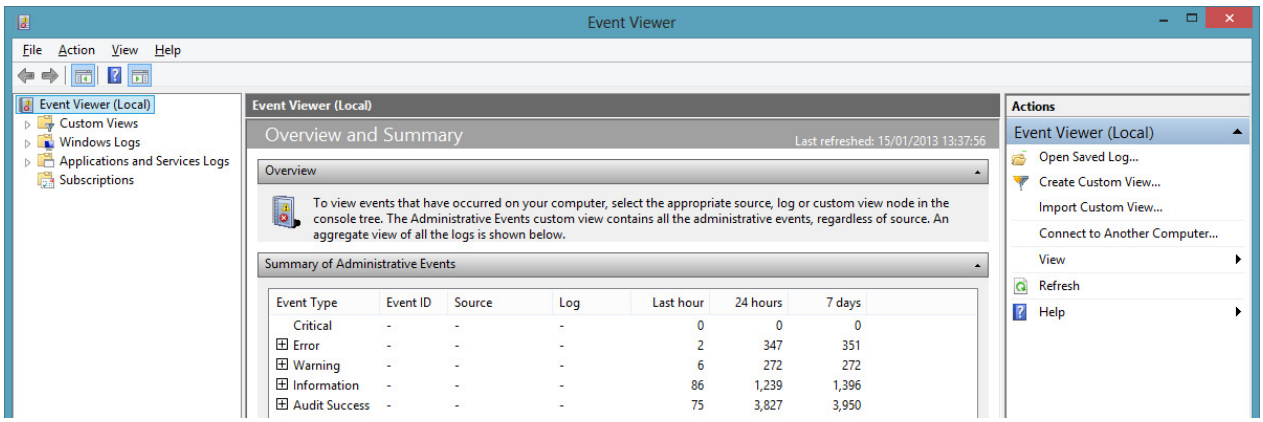

Figure 13: the summary screen of Event Viewer

In the next example we can see the typical fields of an event log entry.

Log Name:	Microsoft-Windows-DHCP Client Events/Admin
Source:	Dhcp-Client
Logged:	2012.03.24. 12:39:55
Event ID:	1002
Task Category:	Address Configuration State Event
Level:	Error
Keywords:	
User:	LOCAL SERVICE
Computer:	voltaic
Description:	The IP address lease 10.224.2.241 for the Network Card with network address 0x001A4B73B022 has been denied by the DHCP server 152.66.252.9 (The DHCP Server sent a DHCPNACK message).

To debug an error, look at the source and ID fields of the event. With this information we can search for the problem on http://support.microsoft.com in the official knowledge base articles or on http://eventid.net. Another option is to click on the online help link in the properties of the event. In this case a query to Microsoft's web based database is made to this entry. For frequent errors usually a solution can be found.

Log files are located in `C:\windows\System32\winevt\Logs`. It is worth to look how many application stores their logs here.

3.3 Registry

The *registry* is the central configuration database in Windows. Settings are stored (or can be stored) here by hardware devices, the operating system or applications. The registry has a tree structure where the nodes are called *keys*. We can assign various properties to keys.

The top level contains the following keys (Figure 14.):

- HKEY_CLASSES_ROOT: it stores various settings belonging to file extensions and the list of COM objects in the system. The essence of COM (Component Object Model) technology is that applications can offer their parts as components to other applications. Such components can be called and used by other applications, and are identified by a GUID (Globally Unique Identifier) called CLSID (Class ID).
- HKEY_CURRENT_USER: the settings belonging to the current user. It is just a link to the corresponding entries in HKEY_USERS.
- HKEY_LOCAL_MACHINE (HKLM): the settings of the computer (detected hardware, list of software installed).
- HKEY_USERS: settings of the users in the system.
- HKEY_CURRENT_CONFIG: information on the current hardware profile, only a link to the key named `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Hardware Profiles\Current`.

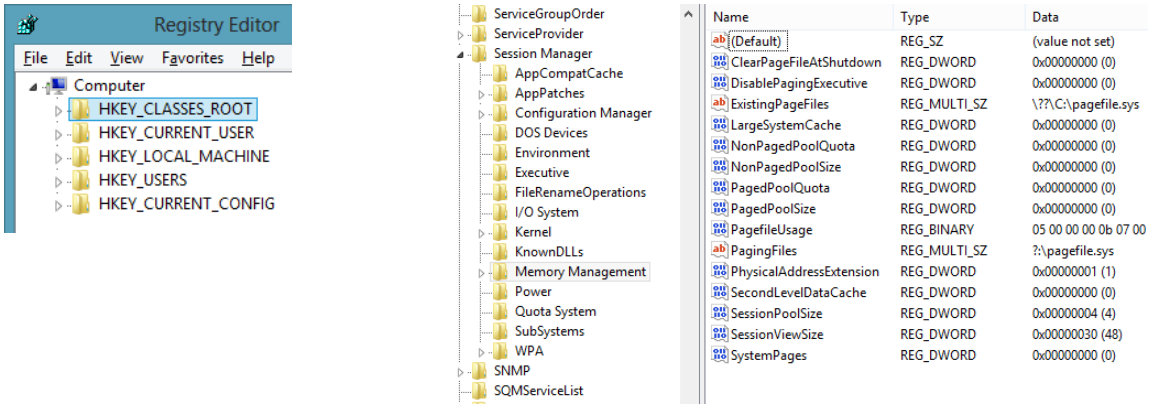

Figure 14: a) the top level keys of the registry b) properties belonging to a given key

The registry is located in the files under the folder `\Windows\System32\config` and can be edited by the application named *regedit*.

3.4 Task Manager

Task Manager is one of the most essential tools to manage processes and to display system load. It can be invoked from the menu displayed after pressing Ctrl+Alt+Del (or by pressing Ctrl+Shift+Esc).

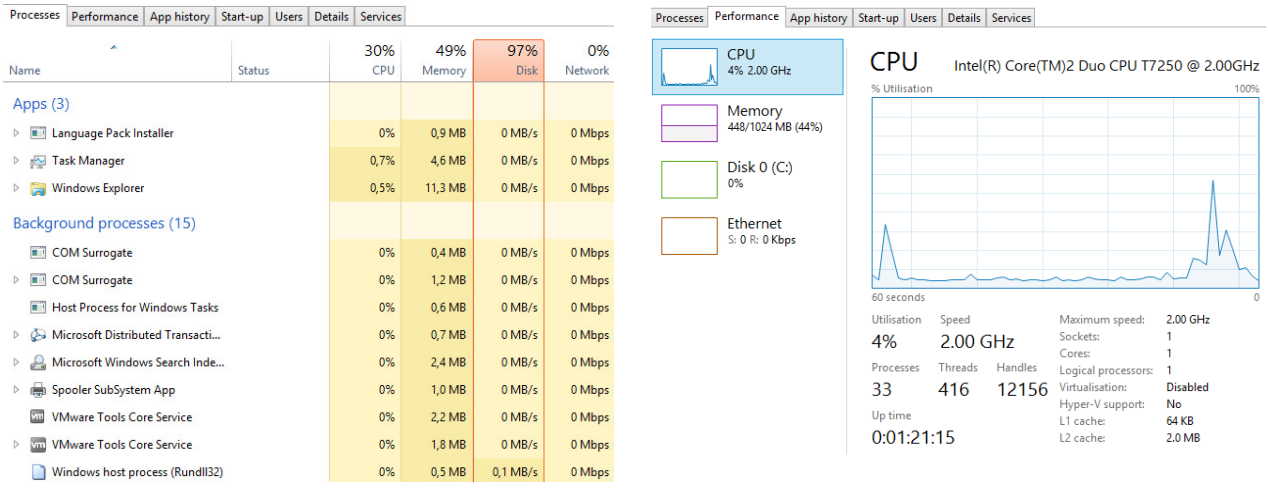

Figure 15: views of the task manager

The *Processes* tab were completely redesigned in Windows 8[5] (Figure 15). It displays less information but that is presented in a much more straightforward way (for example resource usage is shown by a thermal map too). This provides a quick overview on the system showing which processes stress the system most.

The former, more detailed process list can be found under the *Details* tab. It is worth to have a look at the *Select columns* menu (right click on the column names), much more can be displayed compared to the entries selected by default (like PID, caused page fault count, full path and so on). After a right click on the name of a process we can close it or change its priority (be careful with real-time priority).

Task Manager also allows to identify which services are running under the various `svchost.exe` instances. This can be done in the *Services* tab with after right clicking on a service name and selecting *Go to details*. Here we can see the group name under which the system references to the svchost instance in question.

[5] An interesting article on the redesign process of Task Manager: Building Windows 8 Blog, The Windows 8 Task Manager, URL: http://blogs.msdn.com/b/b8/archive/2011/10/13/the-windows-8-task-manager.aspx

On the *Performance* tab we can see a fast overview about how much is the system loaded. The most useful information related to the various resources is placed here.

- *CPU*: current load by cores, number of total processes and threads, CPU hardware data…
- *Memory*: size of active (In use) memory, current and maximum virtual memory (Committed)…
- *Disk*: current write and read speed, average response time
- *Ethernet*: transmit and receive speed, basic network information…

Task Manager is a great tool to provide an overview. But if a more detailed view on the system is needed then the performance counters presented in the next section should be utilized.

3.5 Resource Monitor and Performance Monitor

Detailed performance information can be accessed using *Resource Monitor* and *Performance Monitor*.

The *Resource Monitor* is a bit more detailed than *Task Manager* (Figure 16). A typical usage scenario is the case of a "currently slow system" where we want to identify which resource is running low. Another use case is to identify which file is under a read or a write operation.

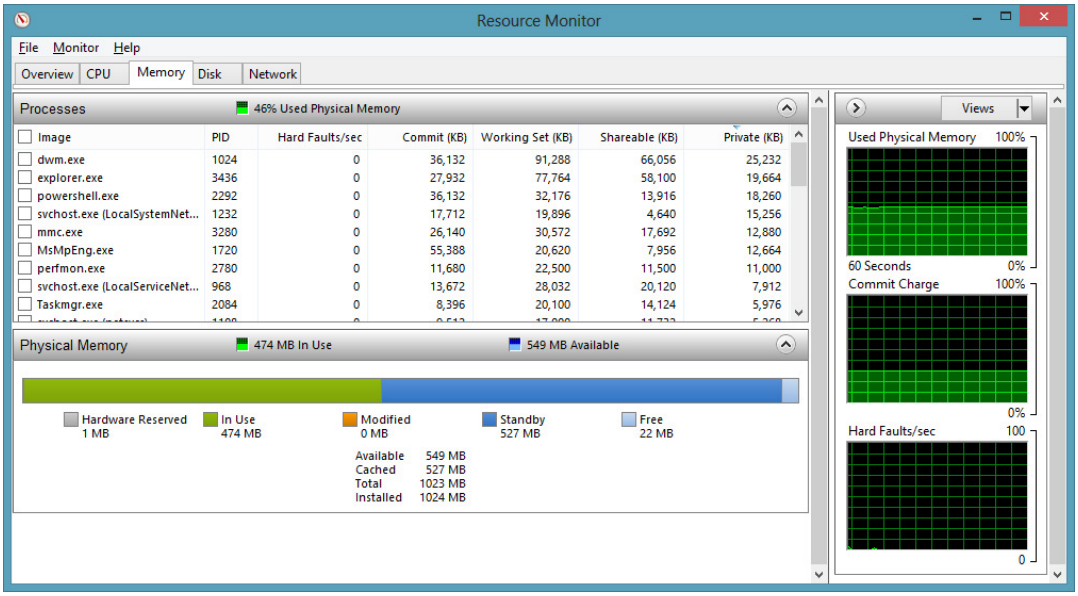

Figure 16: the memory tab of Resource Monitor

The *Performance Monitor* can display the values of the so called *Performance counters*. Most of the OS components provide such counters (like the current size of the virtual memory related to a process, write speed of the physical disk and so on). But other programs are free to register counters too. On the *Add Counters* dialog it is useful to turn on the *Show description* option. After adding counters we can right click on the graph then on *Properties.* Then on the *Data* tab we can tell the scaling factor to the current counter under the *Scale* control (this can be useful if we are about to display counters in very different scales).

Under *Data Collector Sets* we can assemble groups from the above mentioned counters. These counters are then monitored continuously by the system. Such a group is for example the *System Diagnostics* set. At the beginning it collects the most important information about the system (what kind of hardware the computer has, what are the programs started automatically and so on) then it begins to collect data from the most important system performance counters. The data collected can be investigated later at the *Reports* part.

In the *Reliability Monitor* we can see the system and application failures happened in the past. For each event the significant changes made to the system at that time (update, installation of a device driver or an application) are displays. This allows to identify the source of failure given suddenly appeared frequent malfunctions.

3.6 PowerShell

The *PowerShell* is the new command line interface of Windows (Figure 17). The advantage of this new version compared to the original command prompt is that this provides a flexible, object-oriented script language and environment based upon the .NET framework. Since Windows 8 most of the OS components have native PowerShell interface too. E.g. network settings can be managed using PowerShell instead of the old command line tools (like ipconfig and netsh). So far the older variants are still functional but in future Windows versions they may not be supported.

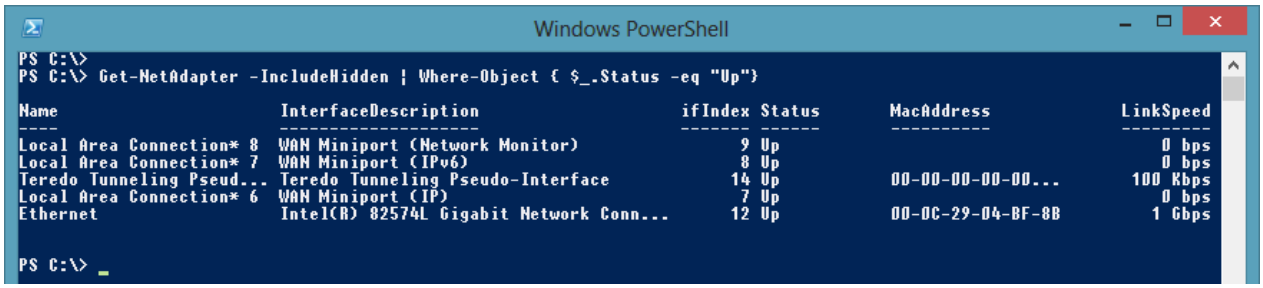

Figure 17: a PowerShell command to query and filter the network adapters

4 Security

The security system of an OS is quite complicated. In this section we discuss only a few important aspects.

4.1 Elements of the security system

The following are the most important security related tasks what have to be accomplished:

- *Authentication*: it is required to identify the entity from which the request is originated. Like at the login procedure we identify ourselves to the system by a user name and a password.
- *Authorization*: it is required to check if the requester has sufficient rights to accomplish the request in question.
- *Auditing*: logging the requests to a security log.

These task are implemented by the following Windows components:

- User authentication is done by LSASS. This component can check a login attempt according to a local database. In case of a domain environment it connects to a central authentication server and asks the server to do the verification procedure.
- The security settings are stored in the *Local Security Policy* in the registry under the key `HKLM\SECURITY\Policy`.
- The base of the security system is the *Security Reference Monitor* component of the *Executive.* This component checks if the requested operation is allowed on a protected object. It is necessary to verify access rights upon accessing a handle.
- Audit events are logged to the built-in *Security* log of *Event Log*.

The most important elements of security:

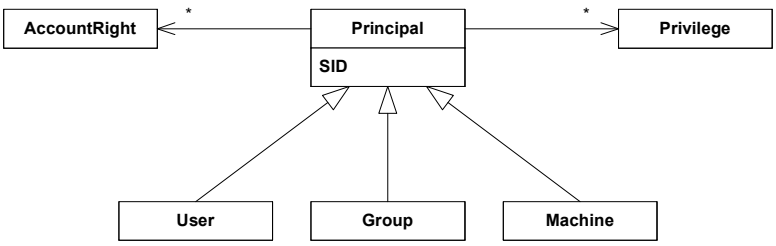

Figure 18: security entities (simplified)

- *Principal*: it is a general name for the entities controlled by the security system. It is a good practice to create *Groups* for easier management; and make security settings only for groups. In this way we only need to put users into or remove users from groups. In the security system the computer is represented by the NT AUTHORITY\System user (at the Services section we referenced it under the name Local System).
- *Security Identifier* (SID): the security system identifies the user by an inner identifier (SID) and not by their name. Here is a typical SID:

  S-1-5-21-13124455-12541255-61235125-500

  The computer gets its SID during OS installation. The SIDs of the local users are generated from it by tailing the machine SID with a *Relative Identifier* (RID). The SID in the above example belongs to the built-in Administrator user (the RID for the Administrator is always 500). Ordinary user RID-s are assigned from 1000. The built-in groups have fixed SIDs (for example the SID belonging to the *Everyone* is S-1-1-0).
- *Privilege*: these are general rights regarding the operating system (like turning the computer off, loading a device driver or changing the priority of a process). The inner name of the privileges are like these: SeShutdownPrivilege, SeLoadDriverPrivilege.
- *Account right*: these rights control how the users are allowed to log in. The login forms can be interactive, through network, log on as service and so on. A given form also can be denied not just allowed.

Information on groups a user is member of and privileges a user has can be listed by the help of *whoami* (see below):

```
PS C:\> whoami /all

USER INFORMATION
----------------

User Name      SID
============== ============================================
ml4-win8\meres S-1-5-21-3647710530-3675882973-2840394154-1001


GROUP INFORMATION
-----------------

Group Name                             Type             SID          Attributes
====================================== ================ ============ ==================================================
Everyone                               Well-known group S-1-1-0      Mandatory group, Enabled by default, Enabled group
BUILTIN\Administrators                 Alias            S-1-5-32-544 Mandatory group, Enabled by default, Enabled group, Group owner
BUILTIN\Users                          Alias            S-1-5-32-545 Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\INTERACTIVE               Well-known group S-1-5-4      Mandatory group, Enabled by default, Enabled group
CONSOLE LOGON                          Well-known group S-1-2-1      Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\Authenticated Users       Well-known group S-1-5-11     Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\This Organization         Well-known group S-1-5-15     Mandatory group, Enabled by default, Enabled group
LOCAL                                  Well-known group S-1-2-0      Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\NTLM Authentication       Well-known group S-1-5-64-10  Mandatory group, Enabled by default, Enabled group
Mandatory Label\High Mandatory Level   Label            S-1-16-12288 Mandatory group, Enabled by default, Enabled group


PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                                        State
============================= ================================================== ========
SeIncreaseQuotaPrivilege      Folyamat memóriakvótájának módosítása              Disabled
SeSecurityPrivilege           A naplózás és a biztonsági napló kezelése          Disabled
SeTakeOwnershipPrivilege      Fájlok és más objektumok saját tulajdonba vétele   Disabled
SeLoadDriverPrivilege         Eszközilleszt?k betöltése és eltávolítása a memóriából Disabled
SeSystemProfilePrivilege      Rendszerteljesítmény kiértékelése                  Disabled
```

Figure 19: groups and privileges of a user

The operating system assigns an *access token* to every process and thread. This token summarizes the most important security information. On the figure bellow we can see the most significant part of a token (an access token may have a vast amount of other fields too (like primary group, the source of the token and so on)).

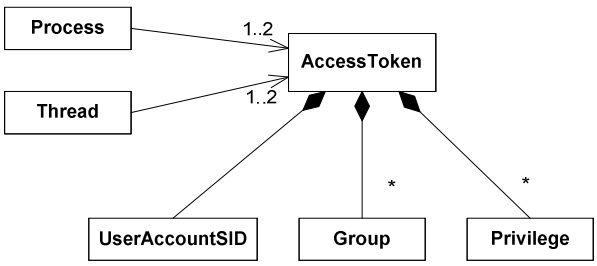

Figure 20: access token (simplified)

When the user logs in the system creates a token and assigns it to the processes of the user. The processes may get temporarily another kind of token, the *impersonation token* too. In this case these processes can access the system under a different user name. Typical usage can be a file server process. If this process gets a request from a user it can first try to impersonate the user to check if he or she really has the required rights to access the file in question. By the help of the command line tool *runas* we can start a program under a different user name (in this case the program gets a completely new token).

The token stores all of the groups the user is member of and all of the privileges either assigned directly to the user or to its groups.

Users and groups can be administered here: *This PC (right click) / Manage / Local Users and Groups*.

To administer the security policy we can use the module *Administrative Tools / Local Security Policy*.

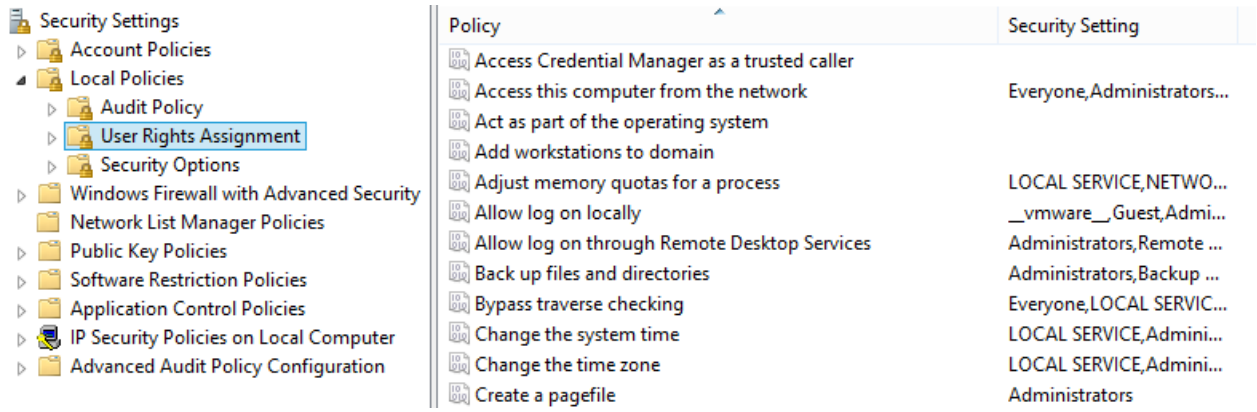

Figure 21: local security policy

- *Password Policy*: here we can set how often it is required to change password, how strong they need to be and so on.
- *Account Lockout Policy*: after how many unsuccessful attempts shall the system lock out the user.
- *Audit Policy*: defines event types that are need to be logged into the Security log (like managing users or using a privilege).
- *User Rights Assignment*: privileges and account rights.
- *Security Options*: other general security options (like: is it required to digitally sign network communication or is the guest account is disabled or nor).

4.2 Access control

In Windows it is possible to assign access protection to every object (no matter if it is a process, a file, a hardware device, shared memory or a key in the registry.

Access control can be accomplished in Windows basically in two ways:

- *Integrity levels*: the *mandatory integrity control* function was introduced in Vista. Every user and object gets an integrity level (low, medium, high or system). Settings can be made to deny read, write or execute operations on the given object to processes started by a user having a lower integrity level.
- *Access Control Lists*: it is possible to assign *Discretionary Access Control* (DAC) to objects. This means that we can tell who can access the object (however this can be altered later by a person having enough rights to do that).

The *Security Descriptor* summarizes the access protection to a protected object:

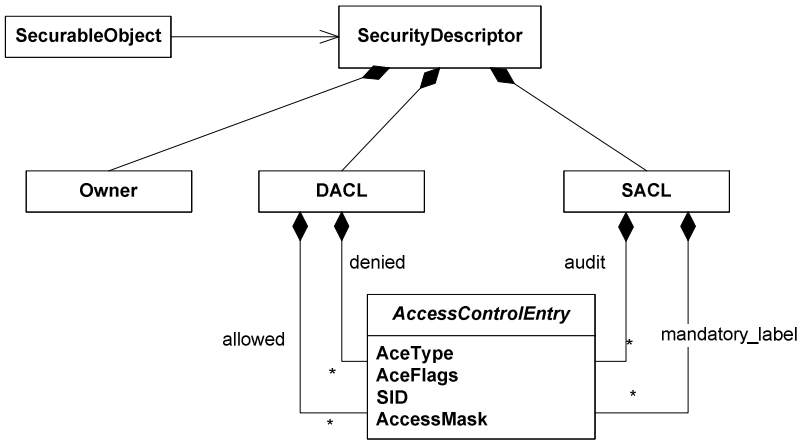

Figure 22: security descriptor (simplified)

To every object belongs a Security Descriptor. The most important parts of this are the following:

- *Owner*: every object has an owner. The owner has the special ability to access the object even when the access rights do not allow access.
- *DACL* (discretionary access control list): DACL is a list containing information about who can access the object and how. There can be two types of list elements: *allow* and *deny* (a deny has always greater priority than an allow). In one element of an access control list (access control entry, ACE) the following information is stored: the type of the element, a flag controlling inheritance, the SID to which this ACE is assigned and an access mask. In case of DACL this mask is generally the combination of write, read, execute, write owner, write DAC and so on operations. (This list can be expanded depending on the type of the given object).
- *SACL* (system access control list): influences access logging and integrity checking. In case of a logged ACE we can set whether we want to log successful or unsuccessful accesses. In case of integrity checking ACE the SID can be any of the three special SIDs (which correspond to the low, medium and high level), and the mask can be NO_WRITE_UP, NO_READ_UP and NO_EXECUTE_UP.

Access checking consists of the following steps (considering what we learned so far):

1. The requester gives the access token and the operation he or she wants to accomplish.
2. The system checks if the integrity level of the requester is high enough for the requested operation. If not access is denied.
3. The system checks the deny rules. If the requested operation is denied to the user or to any group he or she is a member of, then the access will be denied.
4. The system checks the allow rules. If the requested operation is allowed to the user or to any group he or she is a member of then access is granted. In any other cases access is denied.
5. If there is a logging SACL setting related to the user or any group he or she is a member of then the result of the request will be logged by the system.

For the easier administration access control lists can be *inherited*. An ACE can be set to be effective not just to a given object but to its child objects (for example to the values stored in a registry key, or to subfolders and files in a folder). Settings can be made on child object to inherit access rights (which are defined on their parents as inheritable rights) or to break the inheritance chain and use brand new access rights.

Let's have a look via the example of NTFS access control list how can we set access rights!

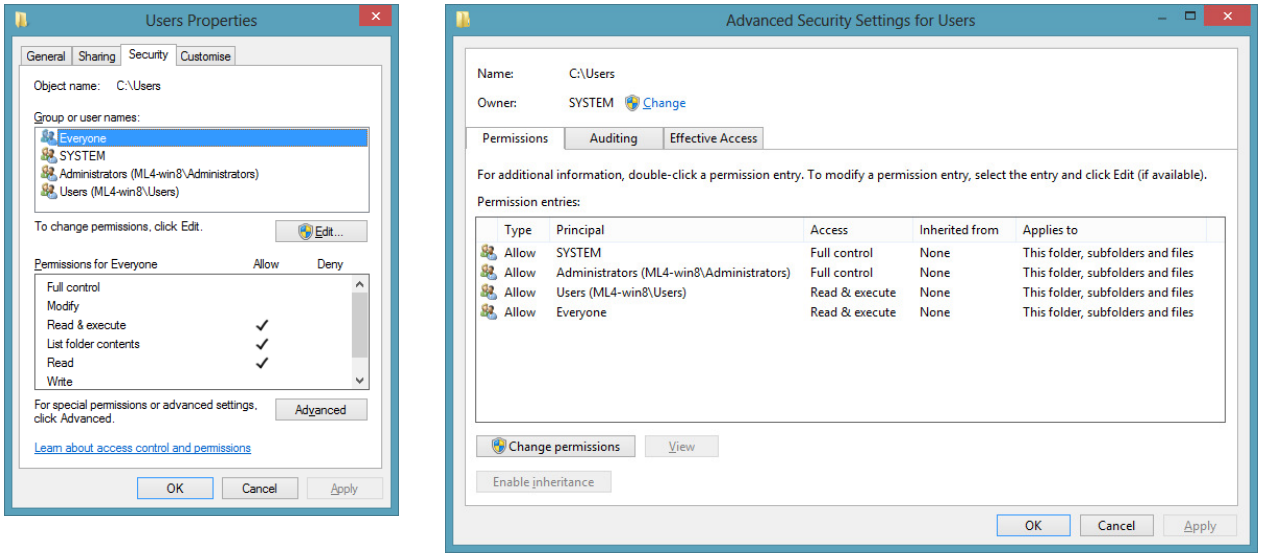

Figure 23: simplified and detailed view on a folder's security settings

On the left you can see the security settings for a folder. This is a simplified view. We cannot see the access elements just a few frequently used groups (made by the elements). For example *Read* involves the following elements: read data, read the attributes and read the permissions.

After clicking on *Advanced* you get a dialog similar to the figure on the right:

- *Permissions*: detailed permission list assigned to the files.
- *Auditing*: it is possible to add logging ACEs here.
- *Owner*: we can display the current owner, and if we have the proper privilege (SeTakeOwnershipPrivilege) then we can even change the owner.
- *Effective Access*: here one can check what are the effective access rights of a given user or group after taking into consideration all ACEs.

At the permission editing dialog one can make detailed settings on the access elements and how it shall be inherited:

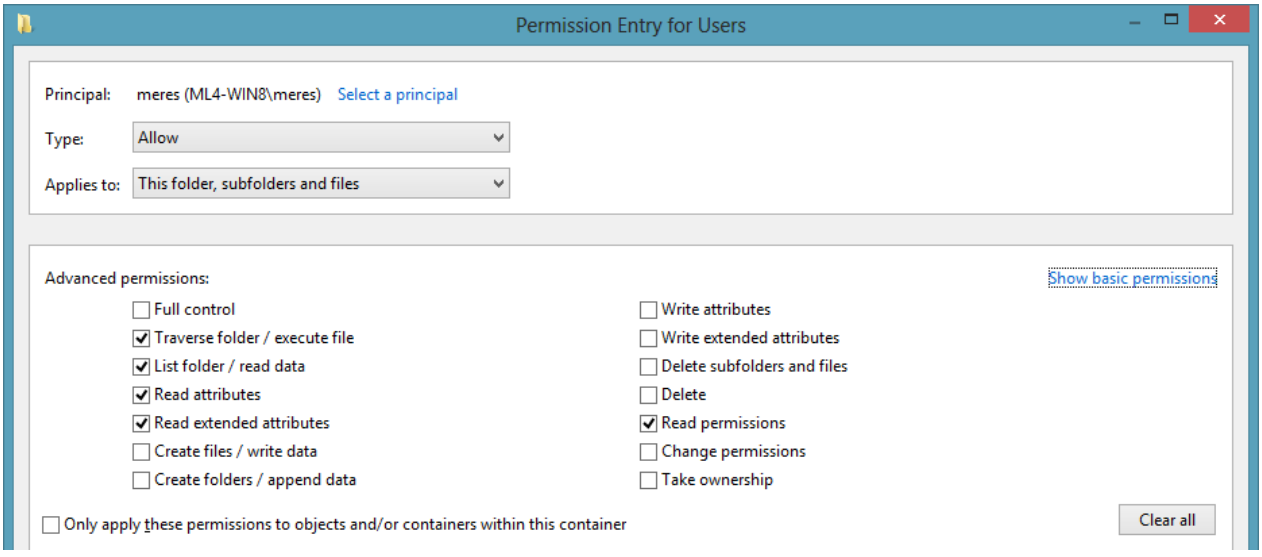

Figure 24: adding an ACE

4.3 User Account Control (UAC)

User Account Control (UAC) is an important change in security which was introduced in Vista.

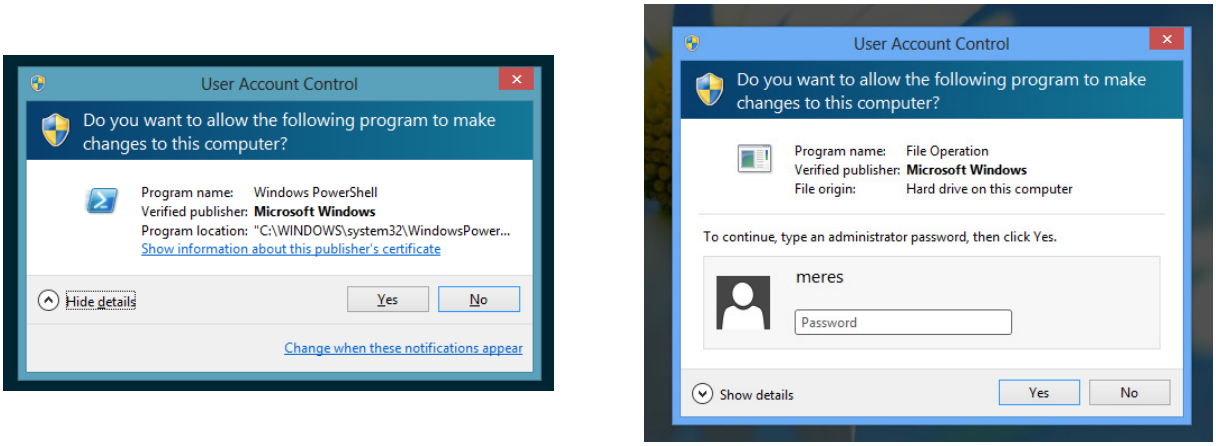

Figure 25: UAC in operation (under a user having administrative rights (left) and not (right))

The principle of UAC is that a user having administrative rights gets two tokens at login. One is an ordinary access token (without administrative privileges). Programs started by the user are usually use this token. If the user wants to do an operation which requires higher level of privileges the system will use the other token. But this is not done automatically. The user has to confirm the operation first. This method can be used even if the user in question does not have administrative privileges at all. The only difference in this case is that the system will prompt us a password.

In this way we really do not need to use the Administrator user all the time (which is a great security risk).

5 Other useful tools

The measurement is conducted in a virtual machine running under VMware Player. VMware Player can be downloaded from http://www.vmware.com/.

The virtual machine consists of the following:

- Windows 10
- Sysinternals tools: http://www.sysinternals.com

Trying out VMware Player and Sysinternals Process Explorer is a part of the preparation phase for the measurement (without any former knowledge of this tools the measurement is hard to be solved in time).

5.1 Sysinternals Process Explorer

Process Explorer can be considered as a much more sophisticated version of *Task Manager*.

On the upper row we can see the CPU, memory and I/O usage graphs. The middle section shows the list of processes. In the lower part we can see the handles or DLLs belonging to the currently selected process (we can switch between these two modes by pressing Ctrl+H and Ctrl+D). Warning: if you see a lot of blank descriptions or company name fields then it is possible that you started Process Explorer without administrative privileges.

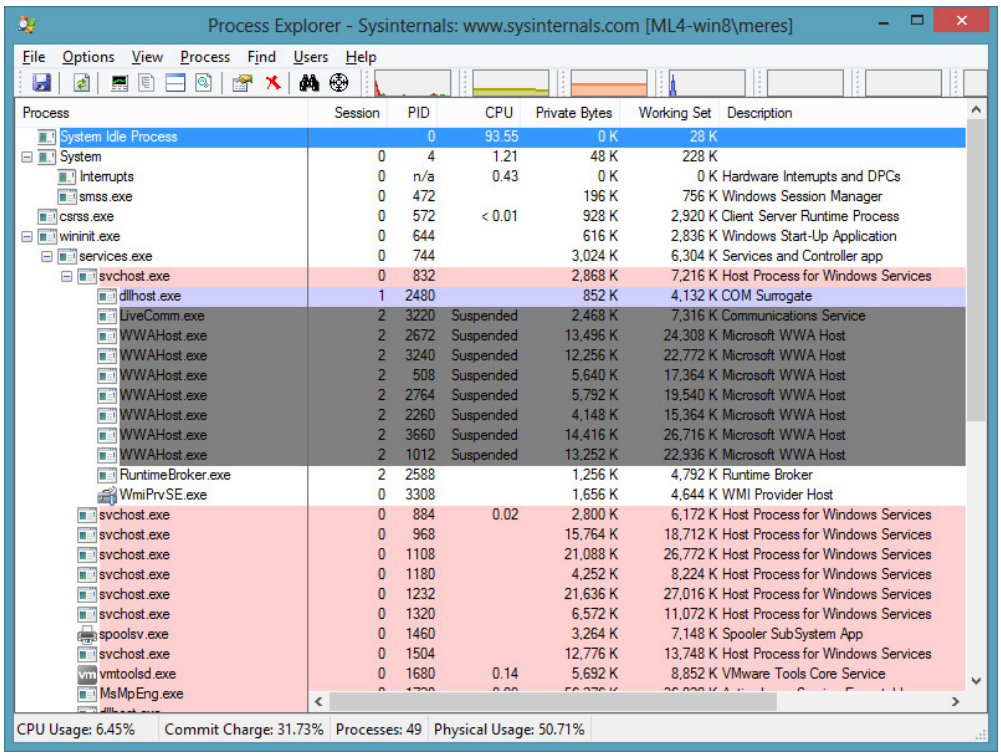

Figure 26: Sysinternals Process Explorer

This software has a lot of capabilities. Let's name a few of them:

- *System Information* (Ctrl+I): a bit similar view to the *Performance* tab of *Task Manager* (just with a bit more information).
- *Find Window's Process*: we can click on a window then Process Explorer tells us which process belongs to that window.
- *View / Select columns*: there are a lot of information that can be displayed besides the ones selected by default.
- *Process / Properties*: detailed information on a process. Like what threads are belonging to it, what functions are these threads are executing currently. Or what TCP/IP connections are opened by the process. Or what *strings* can be found in the binary file belonging to the process (this can tell us a lot about what can be the function of the given process).
- *Find / Find Handle or DLL*: who has open handle to a given object.

5.2 Sysinterals Process Monitor

By the help of *Process Monitor* we can monitor file and registry database accesses in real-time. Even if we do not run any application there are a lot of background operations going on. So it is always worth to filter the captured operations: *Filter* (Ctrl + L). Probably the most common filter criteria is the name of a process but there are a lot of other opportunities.

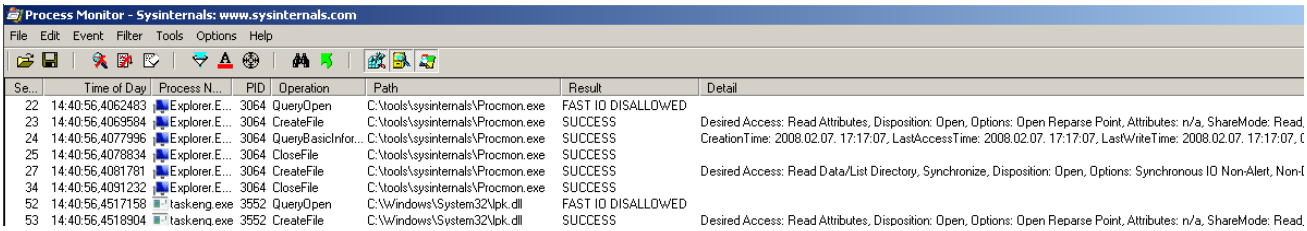

Figure 27: Process Monitor

Process Monitor can help us with really strange access failures. In this case it is worth to use Highlight on the rows in which we are interested in (like the ones having Result values as ACCESS DENIED).

6 Sample test questions

1. What are the subsystems in Windows, and what are their purposes?
2. Create a figure showing the basic structure of the Windows operating system!
3. What is a session?
4. What are the roles of services in Windows? What programs are used to manage them?
5. What are handles in Windows?
6. Where does Windows store the settings of the users?
7. What is the registry, how is it structured?
8. What is the purpose of the svchost.exe program?
9. What tools can be used to monitor performance in Windows?
10. What is a SID? Why is it important?
11. What does an ACL (Access Control List) contain?
12. Give at least 3 performance counters!
13. What built-in event logs are in Windows (at least 3)
14. Give at least 3 example for MMC snap-ins!
15. What is the purpose of the lsass.exe program?
16. How can an administrator access a folder, for which access was denied to him?
17. What kind of elements can be configured in the local security policy (at least 3)?

7 Appendix

7.1 Useful keyboard shortcuts

A few useful keyboard shortcuts to make things a bit easier[6]:

Valid also in versions before Windows 8	
Win (Windows button)	Display Start menu / Start screen
Win + D	Show Desktop (and minimize current windows)
Win + E	Display Windows Explorer
Win + R	Display the Run dialog
Win + X	Display a menu consisting of a few management tools
Win + left arrow	Dock the active window to the left side of the screen
Win + right arrow	Dock the active window to the right side of the screen
Win + up arrow	Make the active window full screen
Introduced in Windows 8	
Win + Q	Open the Search pane (search in everything)
Win + W	Open the Search pane (search in settings)
Win + C	Display the “right side buttons” (Charm bar)

[6] List on more shortcuts: Technet, Common Management Tasks and Navigation in Windows Server 2012, URL: http://technet.microsoft.com/en-us/library/hh831491.aspx#BKMK_keys