

The 'Obvious' Secrets of Malware Analysis

Editor Tzah M.

Special thanks to Nofar II



This booklet is a summarization of information available online, All Rights reserved to the original authors.

INDEX

Modes and permissions	3
Process Introduction.....	3
Process explorer.....	3
Threads intro.....	4
Memory.....	4
Object and Handles.....	4
General architecture overview.....	5
Function call flow.....	6
Core system files.....	7
Processes.....	7-8
Wow64	9
• Architecture	
• Restrictions	
Process fundamentals.....	10
Threads.....	11-12
Mutex and Semaphore.....	13
Thread, Jobs and Object Manager.....	14
Windows processes.....	15-17
Common APIs.....	18-19
Registry.....	20
Persistence.....	20
Malware analysis Ideas.....	21
PDF file format.....	22

Modes and permissions

- User Mode:
 - Access to operation system code and data only.
 - No hardware accesses.
 - Exceptions are handled by the OS, not crashing the system.
- Kernel Mode:
 - Used by the Kernel and device drivers.
 - Allows access to all system resources.
 - Exceptions can crash the system.
 - Unhandled exceptions cause Blue Screens (Of death).

Processes

- Description:
 - A set of resources used to execute a program.
 - A process does not run, it manages resources and threads that actually run.
- Contains:
 - A private virtual address space (reserved on the hard drive).
 - An executable program – Image file which has executable data.
 - A table of handles to kernel objects. (A private number to identify the handle in its context).

Process explorer (default)

- Colors
 - Yellow – Running a dot.net applications.
 - Brown – Running processes that are under a job. (A job object allows groups of processes to be managed as a unit).
 - Pink – Running under the 'Service Control Manager' ((SCM) starts, stops and interacts with Windows **service** processes).
 - To check all the colors and make changes: Process Explorer=>configure colors.
 - To change the highlights duration: options=> Difference highlights duration

Threads intro:

- Description:
 - While a process is a manager and a container of various objects, threads are actual entity that execute code.
- A Thread contains:
 - The state (CPU registers, etc..).
 - Current access mode (User, Kernel).
 - Two Stacks:
 - A Stack for User space and a Stack for Kernel space.
 - TLS – Thread local storage
 - Optional security token.
 - Will usually run under the parent security context.
 - Can use impersonation to temporarily use a different role (for permissions).
 - Optional message queue and windows created by the thread.
 - Windows assumes that every new thread will be a worker thread (CPU, I/O, etc..) but if the thread creates user interaction, it gets the message queue that will hold the UI activity to become a UI thread.
 - Threads has a priority 0-31 (0 being the lowest) – it will also inherit priority from his running process.
 - 3 main states:
 - Running – Run.
 - Ready – want to run when possible.
 - Waiting – Does not want to run, waiting for something to happen.

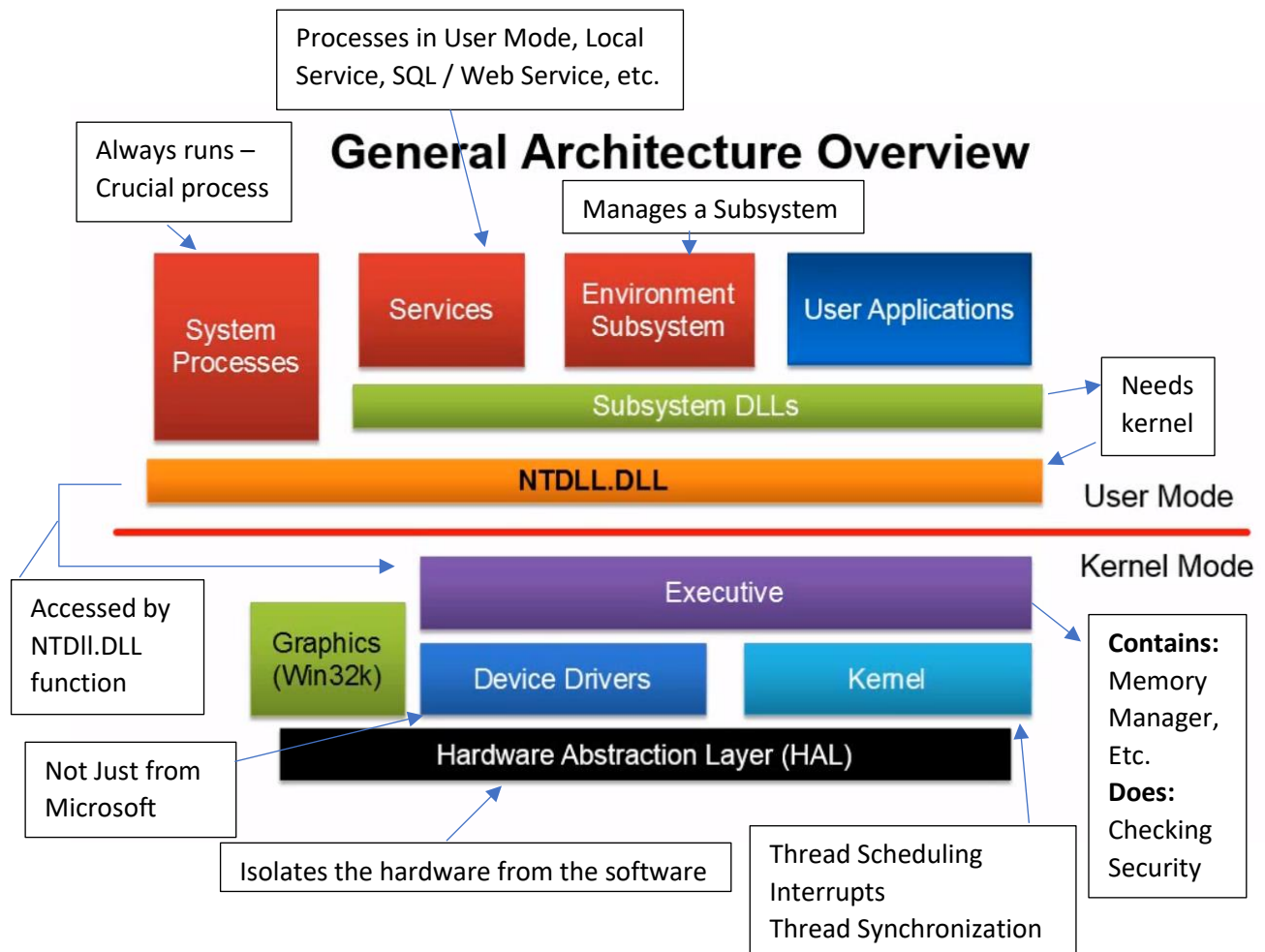
Memory

- **Virtual Memory:** reside on the hard drive, used mostly as RAM for less active processes and when RAM is insufficient.
- **Physical Memory:** RAM (Private Working Set).
- **Private Memory:** Memory that is not shared.
- Tool – vmmap.

Objects and Handles

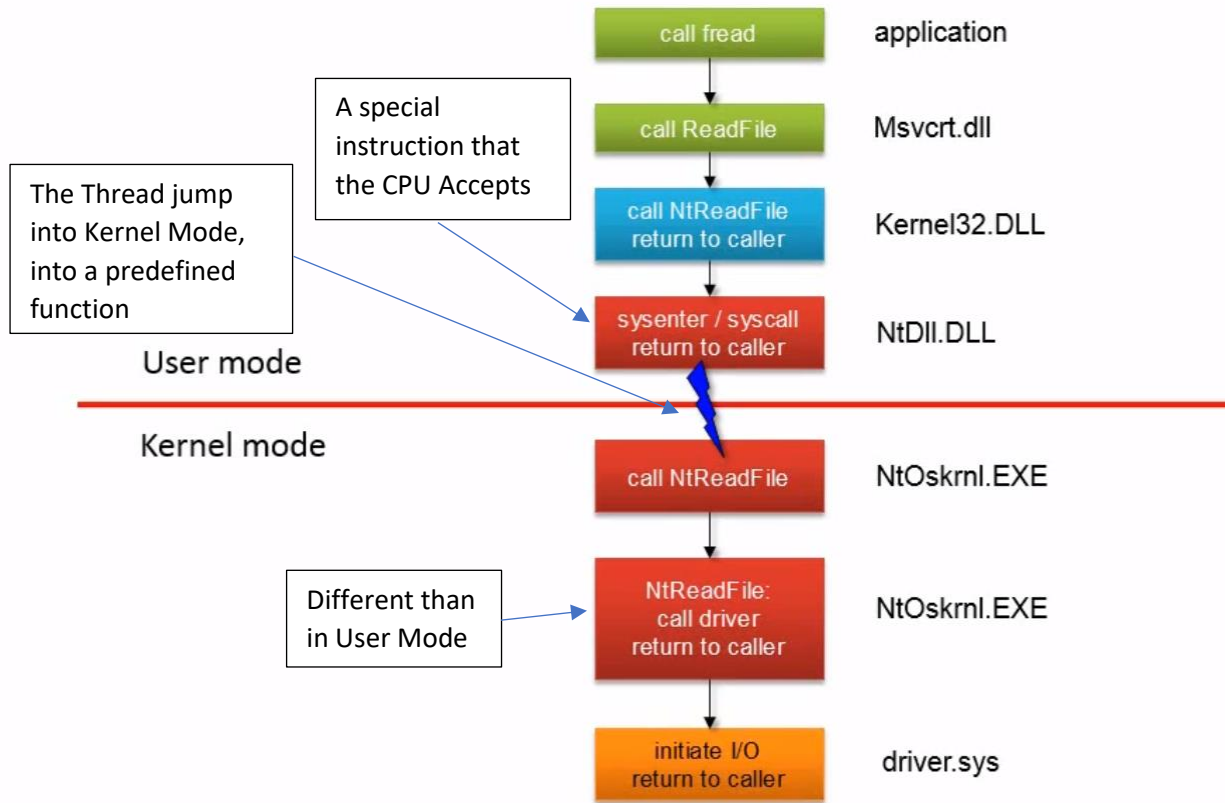
- Objects are runtime instances of static structures.
 - Process, Mutex, Event, Desktop, File.
- Kernel code can obtain a direct pointer to an object.
- User Mode code cannot directly get to the object because system space is unavailable in user mode, therefore, in order for user mode to get to an object, it will use a handle.
- A handle is an indirect reference to an object.
- Objects are reference counted, once the handle closes, the entry will be deleted from the handle table.
- The Object manager is responsible for handling objects.
- The Handle numbers are multiples of the number 4 (i.e. 0x4, 0x8, 0xC) (0 is NULL).
- Kernel object address will always be above 0xFFFFF8A /A8 and will always end with 0 for 64 bit and will always end with either 0 or 8 for 32 bit.
- Access mask is a set of flags which indicate what can be done with this handle.

General Architecture Overview



Function Call flow

Function Call Flow



Core System Files


- Ntoskrnl.exe – Executive and Kernel for 64 Bit systems.
- Ntkrnlpa.exe (Physical address extensions, able to access >4GB RAM) – Executive and Kernel for 32 Bit systems.
- Hal.dll – Hardware Abstraction layer.
- Win32k.sys – Kernel component of the windows subsystem (Windowing GDI graphics)
 - Functions such as:
 - CreateWindow
 - MoveTo
 - Etc..
- Ntdll.dll (The lower layer of User Mode space)
 - System support routines and native API dispatcher to executive services. (jump into kernel mode).
- Kernel32.dll, user32.dll, gdi32.dll, advapi32.dll
 - Applications doesn't call the kernel of the Ntdll directly, they will use one of the above. (official and documented) (Core Windows subsystem DLLs).
- CSRSS.exe (Client Server Runtime Subsystem)
 - Manages the windows subsystem – Always running.

Processes introduction

System – Hosting all device drivers and the kernel.

* All modules in the kernel space can be regarded as DLLs regardless of their extension.

Session Manager - ..\Windows\System32\smss.exe (The first User Mode process created by the system)



Can only use Ntdll.dll because no other is available when the system comes up, therefore, it's a native image.

- Session 0.
- Creating system environment variables.
- Launches the subsystem processes (Normally just CRSS.exe)
- Creating new sessions by launching the instance of itself in other logged on sessions, and then creates winlogon sessions and csrss in that particular session, then terminates.
- Overall, it waits for requests and for csrss.exe instances to terminate.
- Smss and csrss will crash the system if killed.

Winlogon - ..\Windows\System32\winlogon.exe

- Responsible for interactive logon and logoff.
- If terminated, logs off the user session.
- Registered for the ctrl+alt+delete and therefore can't go away and has to stay alive.
- Authenticates the User by presenting a username and password dialog. (logonUI.exe)
 - Can be replaced.
- Sends captured username and password to LSASS.
 - If successfully authenticated, initiates the user's session.

LSASS - ..\Windows\System32\Lsass.exe

- Local session authentication subsystem (service).
- Calls the appropriate authentication package.

Service Control Manager (SCM) - ..\Windows\System32\Services.exe

- Communicates with windows services (Processes that start with windows startup).
- Services
 - Unix – Daemon Processes.
 - Normal windows executables, that interact with the SCM.
 - Not running in Kernel Mode in any way.
 - Can run in special windows accounts like: Local, System, networks, Local service.
 - Start depends on their configuration.

Local session manager - ..\Windows\System32\lsm.exe

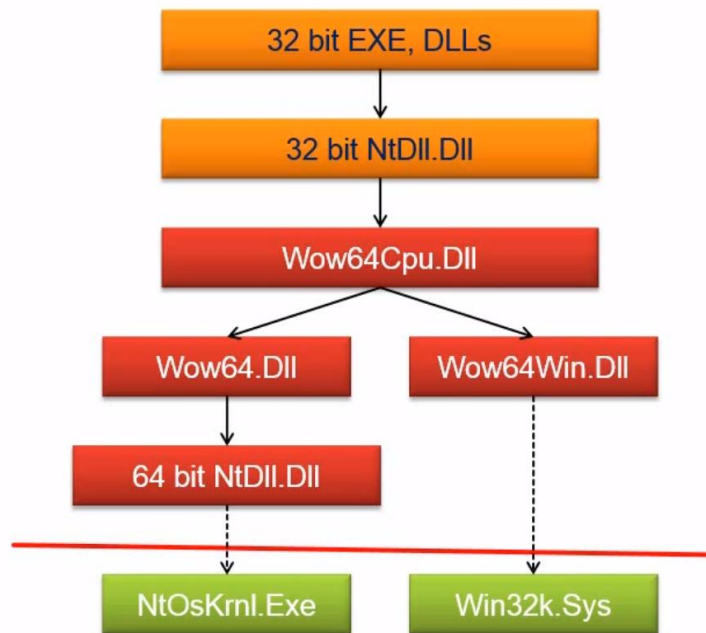
- Introduced in vista and turned to a service in windows 8. (Implemented in a dll)
..\Windows\System32\lsm.dll
- Hosted in a standard svchost.exe
- Manages terminal sessions on the local machine
 - Passing requests to smss.exe.
 - Notified when various events take place at logon, logoff etc.

Wow64 – Relevant for User Mode only, cannot run on Kernel mode.

- Intercept system calls from 32bit applications.
 - Convert 32bit data structure into 64bit aligned structure.
 - Issues the native 64bit system calls.
 - Returns any data from the 64bit system calls -> NtDll.dll
- The IsWow64 Process function call tell whether if a process is running under wow64.
- If a 32bit image is linked with the flag 'LARGEADDRESSAWARE', it will have a 4GB of address space and not 2GB like in 32bit.
- There is no transition layer for drivers, Device drivers must be 64bit as they always run in Kernel mode.
- Every 64bit system contains 2 sets of images:
 - ..\windows\system32 -> contains 64bit images
 - ..\windows\syswow64 -> contains 32bit images

Architecture

Wow64 Architecture



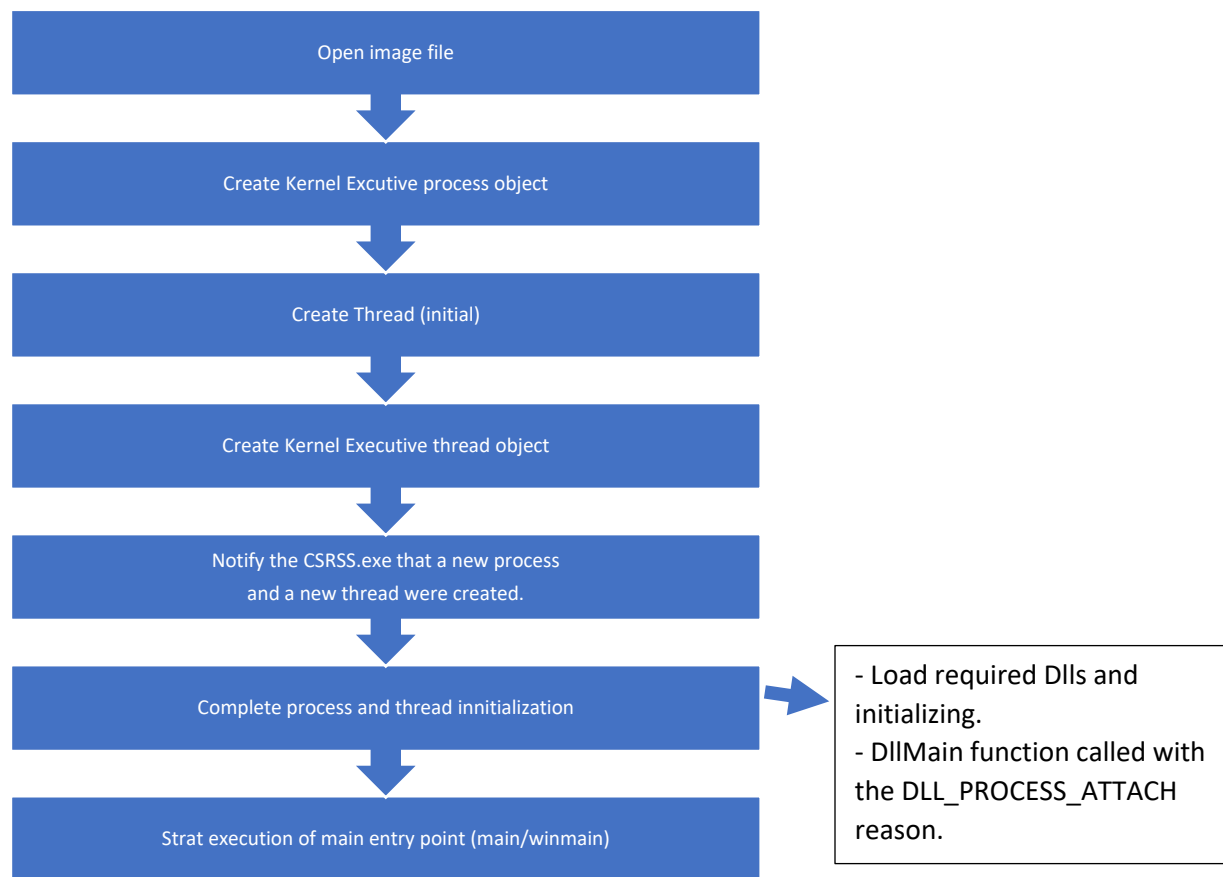
Wow64 Restrictions

- A 64bit process cannot load a 32bit DLL and vice versa.
 - Except for **'resource only' dlls** which can be loaded cross platform.
 - [Contains nothing but resources]
- Some APIs are not supported by the wow64 process, like ReadFileScatter, etc.
- Filesystem redirection.
 - The folder names are the same in 64bit and 32bit but they are not the same files.
 - 32bit must use their own directories.
 - ..\windows\system32 maps to ..\windows\system64.
- Installed applications
 - 32bit in ..\program files (x86)\
 - 64bit in ..\program files\
- Registry redirection
 - Component that will try to register 64bit and 32bit will get the requests to clash.
 - 32bit components will redirect to wow64 registry node (wow64 node).
 - [wow64 node]
 - HKY_LOCAL_MACHINE\software
 - HKY_CLASSES_ROOT
 - HKY_CURRENT_USER\SOFTWARE\CLASSES
- New flags for registry APIs allows access to the 64 and 32 bit nodes.
 - KEY_WOW64_64KEY -> OPENS A 64BIT KEY
 - KEY_WOW32_64KEY -> OPENS A 32BIT KEY

Process fundamentals

- Object management.
- A process does not run, just manages and provides all the resources for the call to execute.
- Holds private address space:
 - 32bit -> 2/3 GB
 - 64bit -> 8TB
- Working set – the amount of physical memory currently used by the process.
- Private handle table to kernel objects. Holds all the kernel objects that have been opened in the process context. On a process by process basis, information from one process table will be meaningless to another process.
- Access token – determines the security context.
- Priority class – given to threads (win32).
 - Affects the threads under the process.
- Basic creation functions: CreateProcess, CreateProcessAsUser.
- Terminates when:
 - All threads in the process terminated.
 - One of the threads call ExitProcess (win32)
 - Killed by Terminate Process (wind32)

Process creation



Threads

- Contains:
 - Context (registry, etc.)
 - 2 stacks (User Mode and Kernel Mode)
 - When created, it assumes to be a worker thread. If it executes code that open windows or UI or using GDI32 or User32 functions, windows creates a message queue for that thread.
 - Security token (of present parent process) can also impersonate in order to perform certain actions and when it's done it will do 'RevertingToSelf'.
 - Scheduling state.
 - Priority 0-31.
 - State (Ready, Wait, Running).
 - Current access mode (User or Kernel).
 - Basic creation function: CreateThread (win32).
 - Destroyed:
 - Thread function return (win32).
 - The thread calls 'ExitThread' (win32).
 - Terminated with 'TerminateThread' (win32).

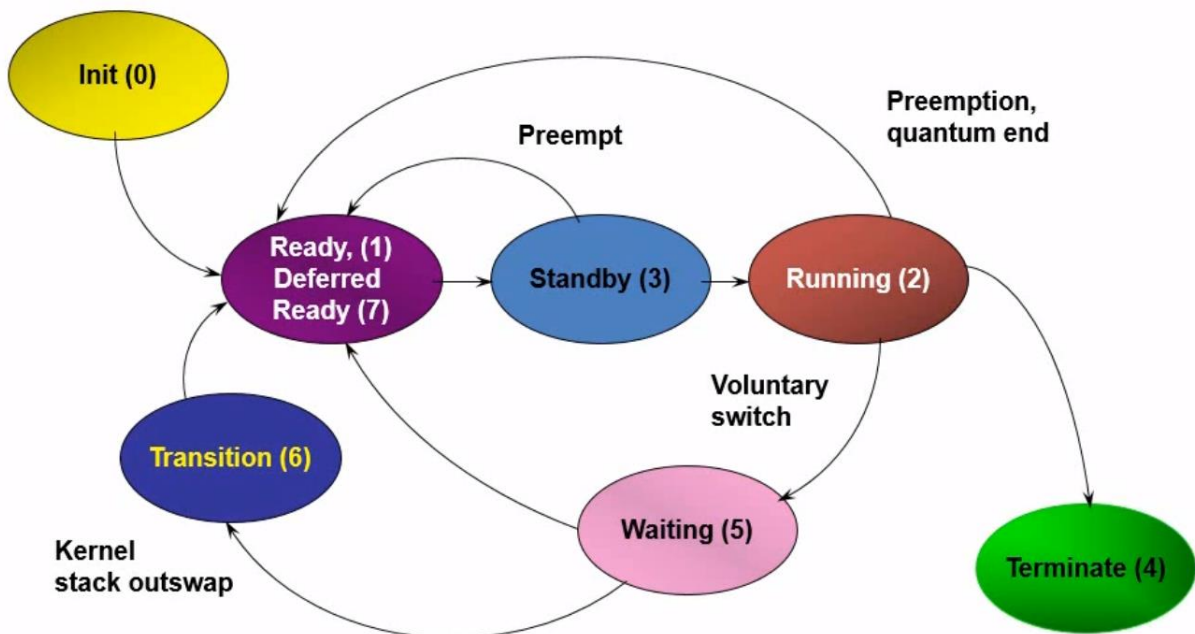
Thread Stacks

- Every User Mode has two stacks:
 - In Kernel space
 - 12K (x86)
 - 24K (x64)
 - It mostly resides in physical memory.
 - In User space:
 - By default, 1MB, 64K committed
 - A guard page causes a special exception to happen if the stack grows too much.
 - Size can be changed
 - Using linker settings.
 - On thread by thread basis in the call to: CreateThread/CreateRemoteThread(Ex)
 - Can specify a new committed or reserved size.
 - Committed assumed unless if the flag: 'STACK_SIZE_PARAM_IS_A_RESERVATION' is used.

Thread scheduling

- Priority based, preemptive, time sliced
 - The highest priority thread runs first.
 - If the time slice (quantum) elapses and there is another thread with the same priority in the ready state – it runs. Otherwise, the original thread will run again.
 - If a thread is running and then another thread will become ready, if the thread which runs has less priority, it will be stopped and the higher priority thread will start.
- Voluntary switch
 - Thread will enter a 'Wait' state to clear space for other threads to run.
- Typical time slice is 30msec for client and 180msec for server.
- Thread states:

Thread States



Thread scheduler

- Scheduling routines are called when scheduling event occurs.
 - Interval timer interrupts, checks for quantum end and timed wait completion.
 - Various events:
 - I/O completion calls.
 - Thread changes priority.
 - Process changes base priority.
 - Changing state.
 - Entering a wait on one or more objects.
 - Entering sleep.

Mutex – (Mutant *In Kernel)

- Mutual Exclusion.
- Allows a single thread to enter a critical region.
- The thread that enters the critical region (its 'wait' state succeeded) is the owner of the mutex.
- Releasing the mutex allows one thread to acquire it and enter the critical section.
- Recursive acquisition is OK.
- It allows one thread to enter a critical block of code that need to execute on a single thread in the system.

Semaphore

- Maintains a counter (set at creation)
- Allows x callers to go through a gate (i.e. multithreading).
- When a thread succeeds a 'wait', the Semaphore counter decreases.
 - When the counter reaches 0 new waits will not succeed.
 - Releasing the semaphore, increments its counter, releasing a waiting thread.
- Has no ownership – A Semaphore with a maximum of 1 thread is not a mutex because of the ownership. A mutex is for protecting data, a semaphore is different.

Event

- Maintains a bullion flag
 - Signal = true
 - Non-signal = false
- Event types
 - Manual reset – Notification in kernel terminology.
 - Auto reset – Synchronization.
- When set (Signaled), threads waiting for it can release any number of threads.
- Events are all about flow synchronization.
 - Manual reset event releases any number of threads.
 - Auto reset event releases just one thread and then the event goes automatically to the non-signaled state.

Critical section

- User Mode replacement mutex
 - Does not have all the powers of a mutex but does have better performance.
- Can only be used to synchronize threads within a single process.
 - Operates on a structure type CRITICAL_SECTION.
- Cheaper than a mutex as when there is no contention it will not transit to kernel mode.
- No way to specify a timeout other than infinite or zero.

More on threads

- Thread Pools – Simplified thread management.
- Can boost performance as threads don't need to be created/destroyed explicitly.
- Simplify operations where order is not important.

Jobs

- Kernel objects that allow us to manage more than one process as a unit.
- The system enforces job quotas and security.
 - Total and per process CPU time.
 - Working sets
 - CPU affinity and priority class
 - Quantum length (Long and fixed quantum only)
 - Security limits
 - UI limits
- API
 - CreateJobObject / OpenJobObject
 - AssignProcessToJobObject (Once a process is inside a job, it can't be removed until termination)
 - TerminateJobObject
 - SetInformationJobObject

Object Manager

- Part of the executive.
- Manages, create, delete and track objects.
- Can be partially viewed with the WinObj sysinternals.
- User Mode clients can do either.
- The Global??? – Holds symbolic links
 - i.e. C: -> \Device\HarddiskVolume2
- The session holds current sessions.
 - Session 0 is for services and other system processes.
- Mutex == Mutant.

Windows Processes

- **System**
 - No parent process.
 - One instance.
 - User account: Local System.
 - Starts at boot.
 - Responsible for most kernel-mode threads, modules run under system are mostly drivers (.sys). Also include several DLLs and Ntoskrnl.exe.
- **SMSS.exe**
 - Image path: %systemRoot%\system32\smss.exe
 - Parent process: system
 - Instances: master instance + child instance per session.
 - User account: Local system
 - Boot time: seconds after boot for master instance.
 - Responsible for creating new sessions
 - The first instance creates a child instance for each new session.
 - The child process will initialize the new session by starting the window session by starting the subsystem (crss.exe) and winint.exe for session 0 or winlogon.exe for session 1 and above.
- **Wininit.exe**
 - Path: %systemRoot%\system32\wininit.exe
 - Created by an instance of smss.exe so usually will not have a parent name.
 - Starts seconds after boot.
 - Starts key background processes within session 0.
 - Service Control Manager – services.exe
 - Local Security Authority Process – lsass.exe
 - For systems with credentials guard enabled – lsaiso.exe
 - Before windows 10 – lsm.exe (moved to lsm.dll by svchost.exe)
- **Taskhostw.exe**
 - Path: %systemRoot%\system32\taskhostw.exe
 - Parent: svchost.exe
 - One instance or more.
 - User account: User / Local service.
 - Generic process for windows tasks, runs in loops listening to trigger events.
 - In windows 10, all scheduled tasks are used by a signed exe or dll.
- **Winlogon.exe**
 - Path: %systemRoot%\system32\winlogon.exe
 - Parent: created by smss.exe, but it will exit afterwards so it will show no parent.
 - One instance or more.
 - Handles interactive user logons and logoffs.
 - Launches 'logonUI.exe' to gather info from the user and then passes the credentials to lsass.exe for validation, upon success, winlogon.exe loads the user's NTUSER.DAT into HKCU (Registry) and starts the user shell (mostly explorer) via userinit.exe.

- **Csrss.exe**
 - Path: %systemRoot%\system32\csrss.exe
 - Parent: created by smss.exe but it exits so no parent process.
 - Two or more instances.
 - User account: Local system.
 - Start within seconds after boot.
 - The 'Client Server Runtime Subsystem' manages processes and threads, import DLLs. Facilitating shutdown of the GUI during system shutdown.
- **Services.exe**
 - Path: %systemRoot%\system32\services.exe
 - Parent: wininit.exe
 - One instance.
 - User account: Local system
 - Implements the 'UBPM' – Unified Background Process Manager which is responsible for background activities such as services and scheduled tasks.
 - Implements the SCM – Service Control Manager which handles the loading of service and device drivers marked for auto start (the driver).
- **Svchost.exe**
 - Path: %systemRoot%\system32\svchost.exe
 - Parent: Services.exe (most of the times)
 - Many instances.
 - User account can be local system (mostly), Network service or local services account.
 - In windows 10 there can also be instances which are running as logged on users.
 - In general, a generic host process for windows services also used for running services DLLs.
 - Windows runs multiple instances using the -k parameter for grouping similar services.
 - Typical -k parameters: Decomlaunch
 - RPCSS
 - LocalServiceNetworkRestricted
 - LocalServiceNoNetwork
 - LocalServiceAndImpersonation
 - NetSvcs
 - NetworkService
 - Malware will try to use svchost.exe either to host a malicious dll as a service or run a malicious process with the svchost.exe name or a similar name.
 - In windows 10 v1703, if the system has more than 3.5GB RAM, most services will run under their own instance, resulting in many svchost.exe instances, Mostly above 50 of them.

- **LsaIso.exe**
 - Path: %systemRoot%\system32\lsaiso.exe
 - Parent: wininit.exe
 - One or more instances
 - User account: Local System
 - When credential guard is enabled, the functionality of Lsass.exe is divided between Lsass and LsaIso, most functionalities remain within Lsass, the role of safely storing account credentials move to LsaIso.exe.
 - LsaIso is running in a context that is isolated from other processes (Hardware Virtualization) and therefore can safely store the credentials.
 - For remote authentication, both processes Lsass and LsaIso will proxy the request using RPC channel.
- **Lsass.exe**
 - Path: %systemRoot%\system32\lsass.exe
 - Parent: Wininit.exe
 - One instance and no children.
 - User account: Local system
 - The 'Local Security Authentication Subsystem' is in charge of authenticating users by calling an appropriate authentication package specified in :
 - HKLM\SYSTEM\CURRENTCONTROLSET\CONTROL\LSA
 - Mostly Kerberos for domain accounts or MSV1_O for localaccounts.
 - Also responsible for implementing the local security policy and for writing events to the security log.
- **Explorer**
 - Path: %systemRoot%\explorer.exe
 - Parent: Userinit.exe but it will exit after finishing so no parent.
 - User account: <Logged on users>
 - Provides users access to files and folders.
 - Also related to: Desktop, Start menu, task bar & control panel.
 - The default user interface as specified at the registry value: HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell.

Common APIs

Virtual API – Page Screen Allocation

- **VirtualAlloc:**
 - Reserves, commits or changes the state of a region of pages in the virtual address space of the calling process. (initialize to 0)
- **VirtualProtect:**
 - Changes the protection on a region of committed pages in the virtual address space of the calling process.
- **VirtualFree:**
 - Releases, decommits or releases and decommits a region of pages within the virtual address space of the calling process.

ScreenGrabber+Spyware+keylogger

- **RULDownloadToFile:**
 - Download bits from the internet, saves it to a file.
- **GetWindowDc:**
 - Retrieves the device context for the entire window. Assign default attributes to the window device context.

Kernel32 functions

- **GetModuleHandle**
 - Retrieves a module handle to the specified module. (The module must have been loaded by the calling process).
- **GetProcAddress**
 - Retrieves the address of an unexported function or variable from the specified Dynamic link library.
- **_wtoi:**
 - Converts a string to an integer.
- **strStr:** [Function]
 - Finds the first occurrence of a substring within a string. (Case Sensitive)
- **Wsprintf:** [function]
 - Writes formatted data to the specified buffer.
- **WinHttpOpen:** [function]
 - Initializing an application, the use of winHTTP function and returns a WinHTTP session.handle.
- **GetModuleFileName:** [function]
 - Retrieves the fully qualified path for the file that contains the specified module. The module must have been loaded by the current process.
- **Load Library:** [function]
 - Loads the specified module into the address space of the calling process. The specified module may cause other modules to be loaded.

- **LocalAlloc:** [function]
 - Allocates the specified number of bytes from the heap.
- **LocalFree:** [function]
 - Frees the specified local memory object and invalidates its handles.
- **ExitProcess:** [function]
 - Ends the calling.
- **CreateFile:**
 - Used to create and open files. Can open existing files, pipes, Strings and I/O devices.
 - dwCreationDisposition controls whether to open a new file or write to an existing file.
- **ReadFile & WriteFile**
 - Used to read and write, to and from a file. Both operate on file as a Stream.
- **CreateFileMapping & MapViewOfFile:**
 - File mappings are commonly used by malware writers as they allow a file to be loaded into memory and can be manipulated easily.
 - CreateFileMapping loads the file to the memory. MappedViewFile returns a pointer to the base address of the mapping.
- **RegCreateKey:**
 - Creates the specified registry key. If it's already exist it will open it.
- **RegSetValueEx:**
 - Sets the data and type of the specified value under a registry key.
- **RegSetKeyValue:**
 - Sets the data and type of the specified value under a registry key plus subkey.
- **IsDebuggerPresent:**
 - Determines if the calling process is being debugged by a User Mode debugger.
- **WriteProcessMemory:**
 - Writes data to an area of memory in a specified process. The entire area needs to be accessible or the operation fails.
- **CreateRemoteThread:**
 - Creates a thread that runs in the virtual address space of another process.
 -
- Indicators of network API's
 - connect
 - accept
 - send
 - recv
 - listen
- Network functions:
 - InternetConnect
 - InternetReadFile
 - InternetWriteFile
 - GetHostByaddr
 - GetHostByName

Registry

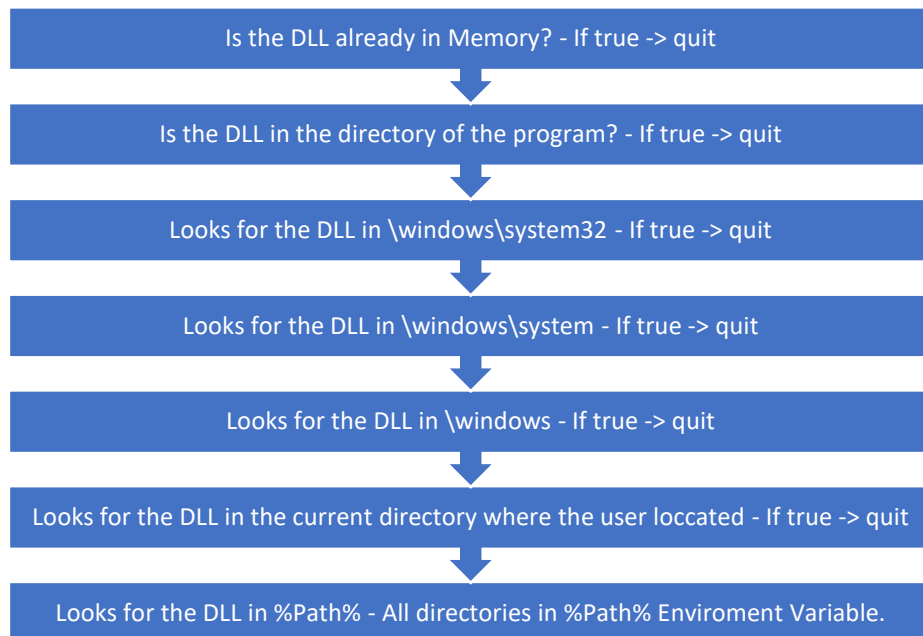
- Root Keys [HKEY / Hive]
 1. Subkey – a subfolder within a folder.
 2. Key – a key is a folder in the registry that can contain additional folders or values.
 3. Value Entry – an ordered pair with name and value.
 4. Value or Data – is the data stored in the registry entry.
- Registry Root Keys:
 1. HKEY_LOCAL_MACHINE(HKLM) – store settings that are global to the local machine.
 2. HKEY_CURRENT_USER(HKCU) – store setting specific to the current user.
 3. HKEY_CLASSES_ROOT – Store information defining types.
 4. HKEY_CURRENT_CONFIG – Store information about the current hardware configuration, specially differences between the current and standard config.
 5. HKEY_USERS – Define settings for the current user, default users and new users.

Registry Persistence:

- \Software\Microsoft\Windows\CurrentVersion\run
- \Software\Microsoft\Windows\CurrentVersion\runOnce
- \Software\Microsoft\Windows\CurrentVersion\RunServices
- \Software\Microsoft\Windows\CurrentVersion\RunservicesOnce
- \Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run
- \Software\Microsoft\WindowsNT\CurrentVersion\Windoews\Applnit_Dlls

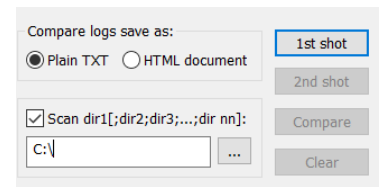
DLL Persistence:

- DLL Search Order Mechanism:
 1. The Operation System will look for imported DLLs in the following order and hierarchy. Once it finds the DLL, it will stop.
 2. When a program attempts to load a DLL, it will check:



Malware Analysis Ideas

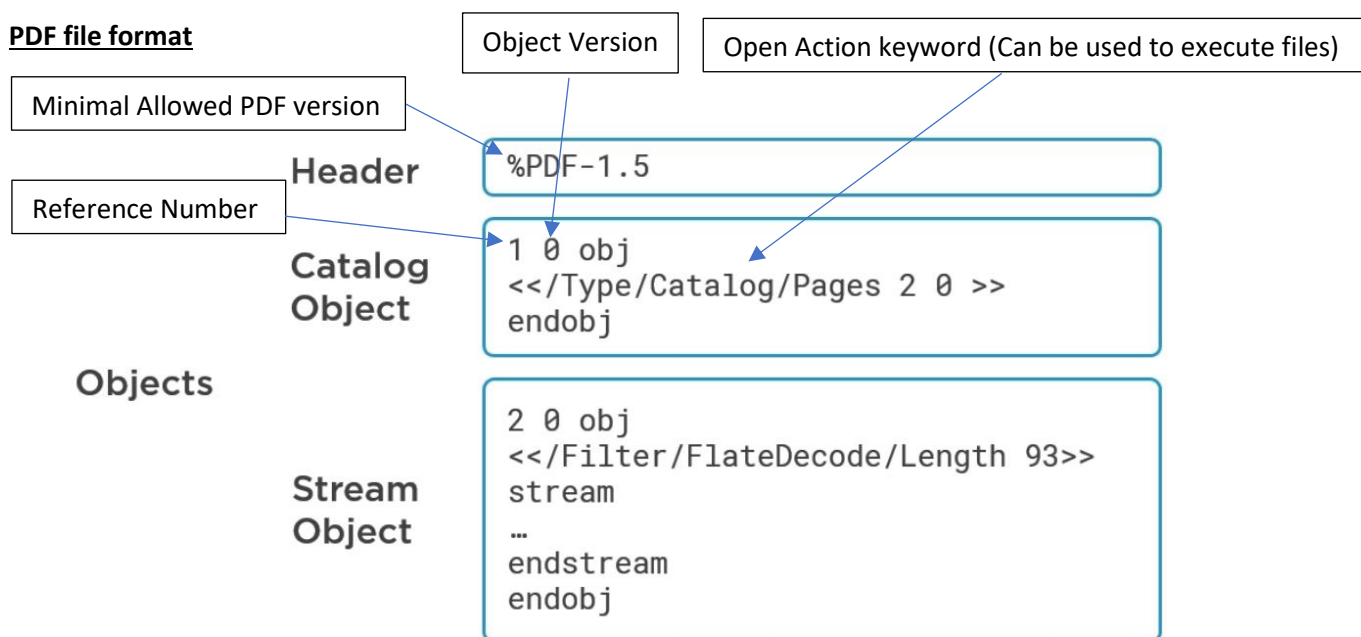
- By Order
 1. RegShot #1 -> Check [Scan dir1] Change the path to C:\
 - Click 1st Shot.
 2. Network – Run Fakenet.
 3. Procmon – Filter ideas:
 - Process name
 - Operation == WriteFile
 - Operation == SetDepositioninformationfile
 - Operation == RegSetValue
 - Operation == ProcessCreate
 - Operation (Start With) TCP or UDP or network keywords.
 4. WireShark
 5. Run Malware - Let the malware run for 5-10 minutes.
 6. RegShot #2 -> Compare
 7. Save ProcMon to CSV and use it in ProcDot.
- *Linux Remnux – Malware analysis tools preset.



File Analysis

- Install popular document reader such as Adobe, Word, etc.
- For .docx file we can use – **ZipDump** to extract the zip (the .docx)
- Extract metadata with **Exiftool**.
- RemNux terminal commands:
 - **Strings** -a [filename] | more
 - **Xorsearch** [FileName] [StringToSearch] (Checks for obfuscation of a string)
 - **Exiftool** [FileName] | more
 - **Yara** -U -msg ~/index.yar [location] [FileName]
 - **pdfid** Identify PDF object types
 - **pdfparser** – search through the PDF for key terms -> options:
 - -s [term]
 - -o [Object Number] (Print object by number)
 - -f (decode data in object)
 - -w (shows raw data)
 - --search
 - --Object [obj number]
 - **Peepdf** – combine multiple tools:
 - Suspicious objects – option:
 - -l (inline)
 - -u (update)
 - Decode data.
 - JS analysis.

PDF file format



* Stream objects can also contain compressed data.

- Suspicious keyword in objects:
 - `/openAction` or `/AA` – point to another object to execute.
 - `/javascript` or `/JS` – Contains code.
 - `/Names` (like file names).
 - `/EmbeddedFile` – A file can be embedded in the PDF.
 - `/URI` or `/submit form` – Can direct to a malicious site.
 - `/launch` – Causes any object associated with the PDF to open when the PDF opens.

Strings to look for in a PDF:

- HexENcoded - `#48#65#6C#6F#72#6C#64#21`
- Octal Encoded - `\110\145\154\157\162\154\144\041`
- Mix - `#48#65#6C` `\110#145` (including whitespaces)

*Filters that can be used to decode data:

- `/ASCIHexDecode` – Hex Encoding.
- `/LZWdecode` – LZW compression algorithm.
- `/FlateDecode` – Zlib Compression.
- `/ASCII85 Decode` – ASCII base 85 representation.
- `/Crypt` – Various encrypting algorithm of choice.

