Memory Forensics

- Windows
  - Tools
    - Capture memory
      - WinPMEM
      - DumpIt
      - KnTDD
    - Examine memory
      - Volatility
        - volatility -f <file> kdbgscan -> OS profiling
        - After OS profile, we can run different commands
          volatility -f <file> --profile=<> <command (plugins)>
          - pslist or pstree
            - These two rely on the doubly linked list mentioned in [Process Organization] below
          - psscan
            - This rely on pool scanning
          - psxview
            - It gets scan info using 7 different methods described in [Process Organization]
              - Option --apply-rules will exclude few exceptions
          - getsids
            - Finding each process' token
              - If process is terminated, we can get offset using psscan and then use --offset=<> option
          - privs
            - Privileges of the process
              - -p <process pid>

- handles
- cmdline
- memdump
  - -p <pid> -D <directory>
- dlllist
- ldrmodules
  - -> same as dlllist but searches the memory in more comprehensive manner
- malfind
  https://dfrws.org/sites/default/files/se…
  - examines Virtual Address Descriptor (VAD) table
  - malfind looks for two things in a malicious Dlls
    - Short virtual address descriptor tag (VadS)
    - Protection bit marked as Executable
- vaddump
  - In case malfind doesn't work
- apihooks
  - This may take a long time to run. To fasten, we can give filters like -- -p (pid of specific process), --quick, --skip-kernel
- impscan
  - examine the process and extract API name and address info. malfind doesn't have API names. So this can be used to save the info in IDC file which can later be used by IDA PRO for analysis (File->IDC file)
    - -p <pid> -b 0x.... --output=idc
  - This has to be added in the idc output file
    - #include <idc.idc>
      static main(void){

      }

- Internals
  - Paged and Non-Paged Pool
    - Paged pool consists of virtual memory that can be swapped in and out of the system.
    - Non paged pool are those which are guaranteed to reside in physical memory as long as corresponding kernel object are allocated

- Kernel Pool Allocation Example:
  Assuming that a process wants to create a new file using the Windows API, the following steps will occur:
  - 1. The process calls CreateFileA (ASCII) or CreateFileW (Unicode)—both are exported by kernel32.dll.
  - 2. The create file APIs lead into ntdll.dll, which subsequently calls into the kernel and reaches the native NtCreateFile function.
  - 3. NtCreateFile will call ObCreateObject to request a new File object type.
  - 4. ObCreateObject calculates the size of _FILE_OBJECT, including the extra space needed for its optional headers.
  - 5. ObCreateObject finds the _OBJECT_TYPE structure for File objects and determines whether to allocate paged or nonpaged memory, as well as the four-byte tag to use
  - 6. ExAllocatePoolWithTag is called with the appropriate size, memory type, and tag.
- Windows Executive objects
  - A structure becomes a executive object when OS prepends few headers to manage services

- PROCESS
  - Process Organization
    - _EPROCESS is the name of the structure which windows uses to represent a process
    - _EPROCESS contains a _LIST_ENTRY structure called ActiveProcessLinks
      struct EPROCESS
    - _LIST_ENTRY structure contains two members: a Flink (forward link) that points
      to the _LIST_ENTRY of the next _EPROCESS structure, and the Blink (backward link) that points to the _LIST_ENTRY of the previous _EPROCESS structure
      - Process Explorer and Task Manager, rely on walking the doubly linked list of _EPROCESS structures.
        - An API commonly used for this purpose is "NtQuerySystemInformation()"
    - Alternate process listing:
      - 1. Process object scanning - Pool scanning
      - 2. Thread scanning

        - Every process has a thread running, that can be found by scanning for _ETHREAD and map back to process using _ETHREAD.ThreadsProcess (XP) and _ETHREAD.Tcb.Process (Vista and later)
    - 3. CSRSS handle table
    - 4. PspCid table
        - This is a special handle table located in kernel memory that stores a reference to all active process and thread objects.
    - 5. Desktop threads
    - 6. Session processes
        - The "SessionProcessLinks" member of _EPROCESS associates all processes that belong to a particular user's logon session
- Enumerating Processes in memory:
    - 1. Kernel Debugger data block, _KDDEBUGGER_DATA
    - 2. Access "PsActiveProcessHead" member, which points to head of the doubly linked list of _EPROCESS
- Direct Kernel Object Manipulation, DKOM
    - By loading kernel driver
    - By using special native API function, ZwSystemDebugControl
- Windows known processes 10
  Standard Windows processes: a brief…
- TOKENS
    - Describes security context of a process. Includes SIDs and privileges
        - For example- S-1-5-21-4010035002-774237572-2085959976-1000
          - S: Prefix indicating that the string is a SID
          - 1: The revision level (version of the SID specification) from _SID.Revision
          - 5: The identifier authority value from _SID.IdentifierAuthority.Value
          - 21-4010035002-774237572-2085959976: The local computer or domain identifier from the _SID.SubAuthority values
          • 1000: A relative identifier that represents any user or group that doesn't exist by default
        - Subtopic 2
    - Accessing Tokens
        - A process can access it's own token using "OpenProcessToken" API
        - To enumerate SIDs -> "GetTokenInformation"
    - Privileges
        - Ways to enable Privs

- Enabled by default
  - Local Security Policy (LSP) can specify what privs will be available by default
- Inheritance
  - from parent process
- Explicit
  - "AdjustTokenPrivileges" API can be used for this
    - This API doesn't allow enabling a privilege that isn't present in a token
- Commonly exploited privs
  - SeBackupPrivilege
    - Read access to any file on the system regardless of it's ACL
  - SeDebugPrivilege
    - gives ability to read from and write to process' private memory space. Used in code injection
  - SeLoadDriverPrivilege
  - SeChangeNotifyPrivilege
  - SeShutdownPrivilege
- Detecting TOKEN manipulation
  https://media.blackhat.com/bh-us-12…
- Handles
  - Lifetime of a handle
    - APIs like createFile return a special data type HANDLE, which is simply index into process' handle table.
      Then APIs like writeFile/ReadFile work in the following manner
      - 1. Find the base address of the calling process’ handle table.
      - 2. Seek to index 0x40.
      - 3. Retrieve the _FILE_OBJECT pointer.
      - 4. Carry out the requested operation
  - Each process’ _EPROCESS.ObjectTable member points to a handle table (_HANDLE_TABLE).

- OS Concepts
  - Process
    - It's a container of set of resources used when executing the instance of program
  - VAD (Virtual Address Descriptor)

        - Structure which is kept to track which virtual addresses are reserved in the process's address space
    - Semaphores
        - Types
            - Binary
            - Counting Semaphore
        - Signalling mechanism
        - It's an integer value
        - It's a process synchronization tool
    - Mutex object
        - Locking mechanism
        - It's an object
    - Memory Management mechanisms
        - Segmentation
        - Paging
- MAC
- Linux