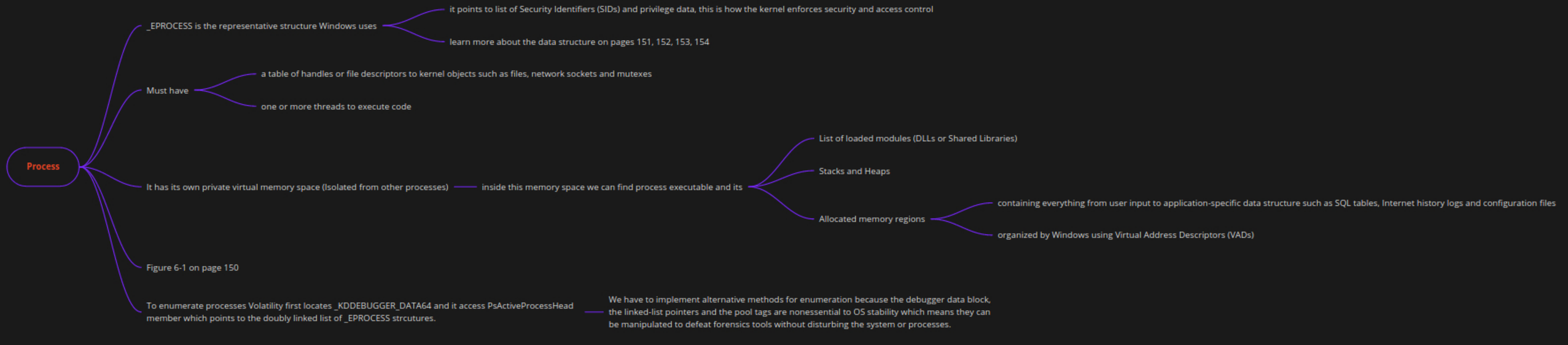
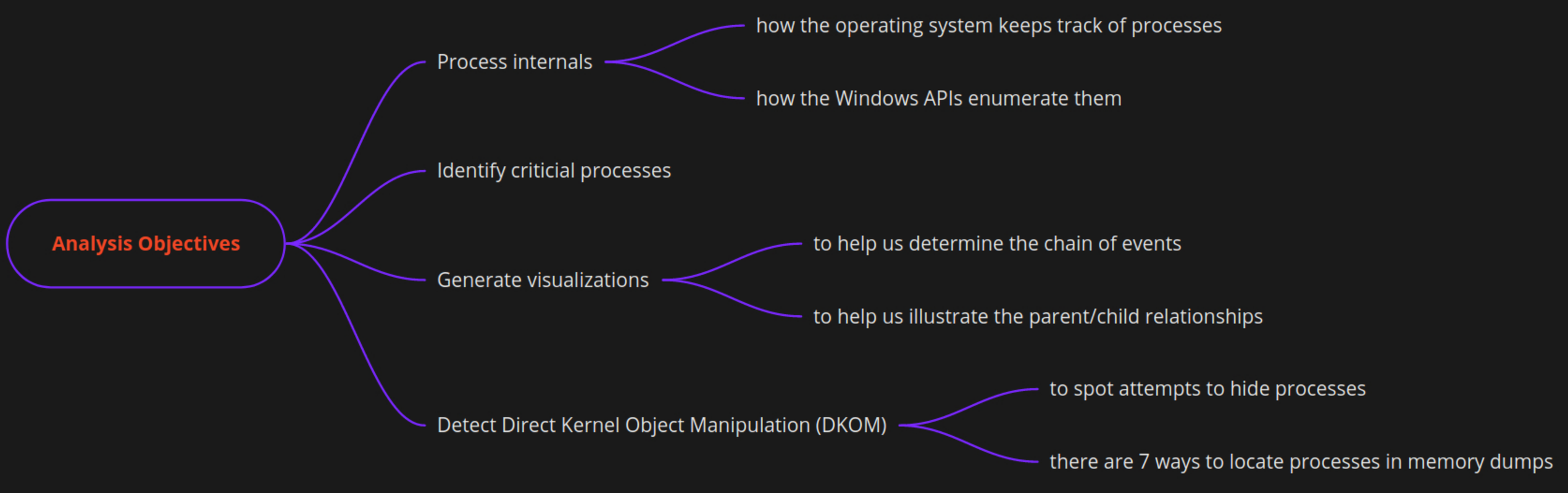
The Art of Memory Forensics

Ch.6 - Processes, Handles and Tokens

0x1411

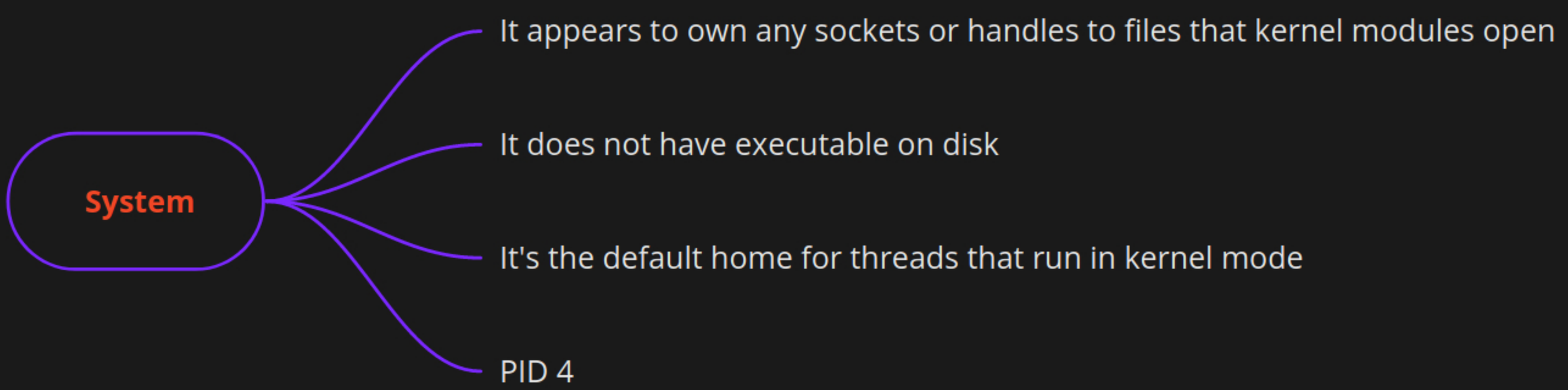

Process
_EPROCESS is the representative structure Windows uses
it points to list of Security Identifiers (SIDs) and privilege data, this is how the kernel enforces security and access control
learn more about the data structure on pages 151, 152, 153, 154
Must have
a table of handles or file descriptors to kernel objects such as files, network sockets and mutexes
one or more threads to execute code
It has its own private virtual memory space (Isolated from other processes)
inside this memory space we can find process executable and its
List of loaded modules (DLLs or Shared Libraries)
Stacks and Heaps
Allocated memory regions
containing everything from user input to application-specific data structure such as SQL tables, Internet history logs and configuration files
organized by Windows using Virtual Address Descriptors (VADs)
Figure 6-1 on page 150
To enumerate processes Volatility first locates _KDDEBUGGER_DATA64 and it access PsActiveProcessHead member which points to the doubly linked list of _EPROCESS strcutures.
We have to implement alternative methods for enumeration because the debugger data block, the linked-list pointers and the pool tags are nonessential to OS stability which means they can be manipulated to defeat forensics tools without disturbing the system or processes.

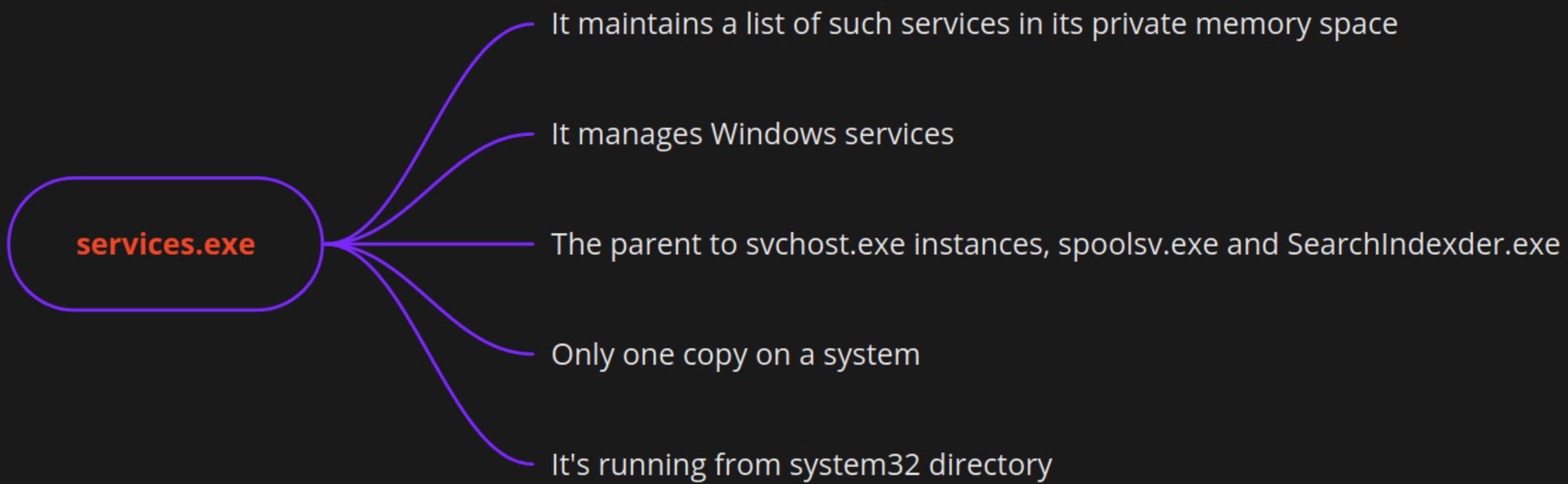

Analysis Objectives
Process internals
how the operating system keeps track of processes
how the Windows APIs enumerate them
Identify criticial processes
Generate visualizations
to help us determine the chain of events
to help us illustrate the parent/child relationships
Detect Direct Kernel Object Manipulation (DKOM)
to spot attempts to hide processes
there are 7 ways to locate processes in memory dumps

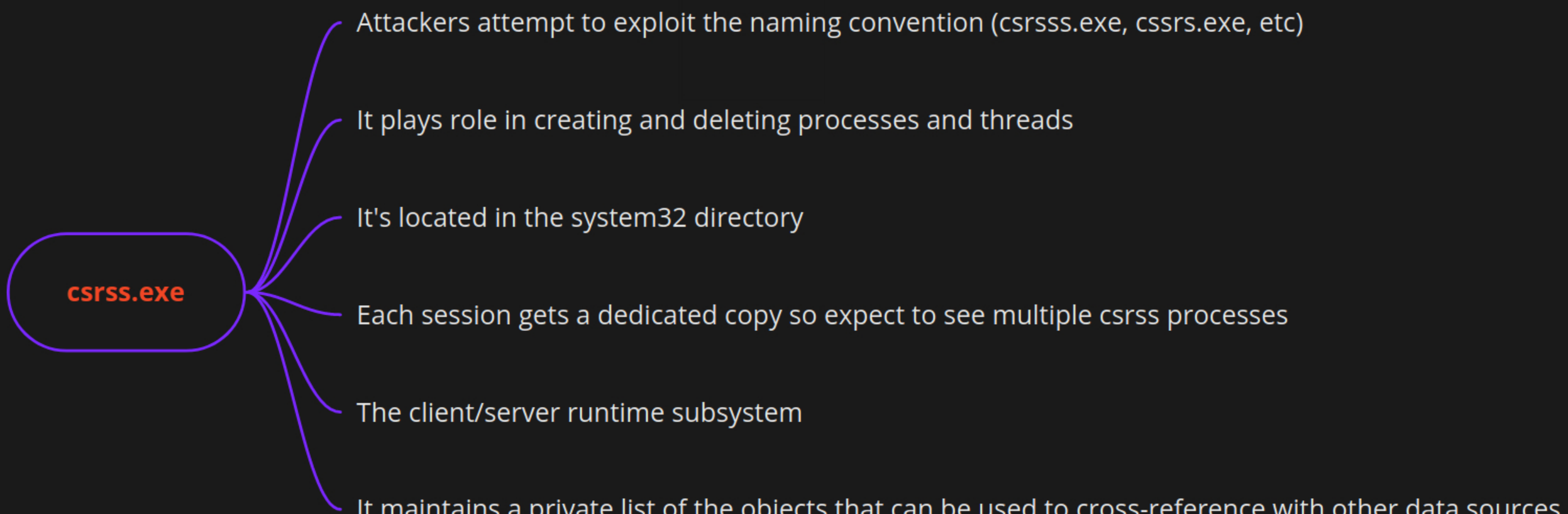
Critical System Processes

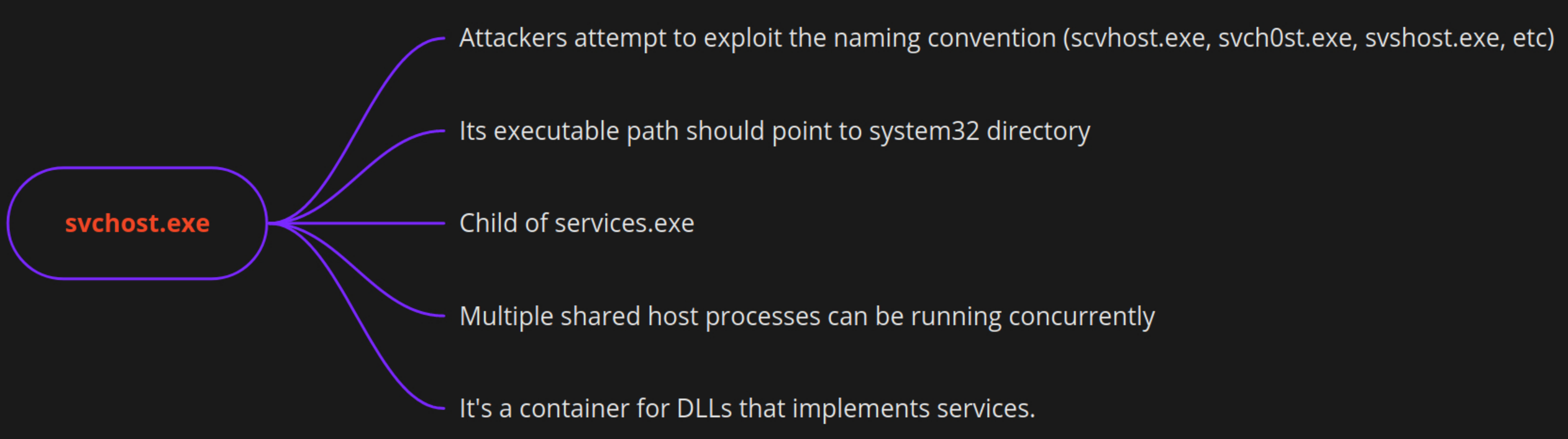
It's a container the kernel uses to charge CPU time for idle threads

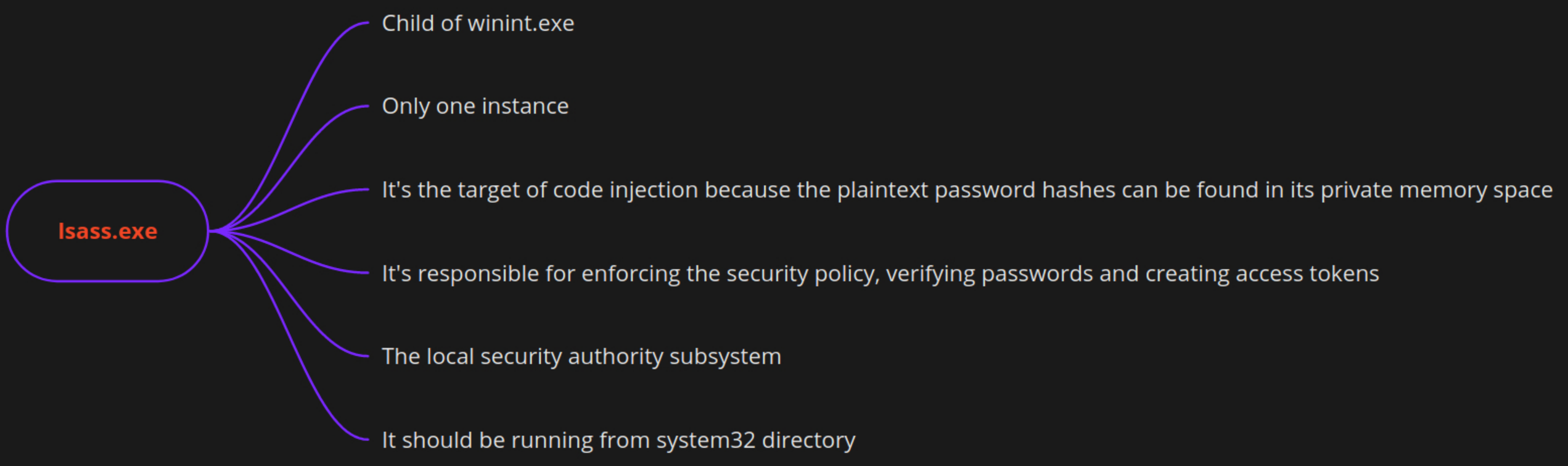
It does not have executable on disk

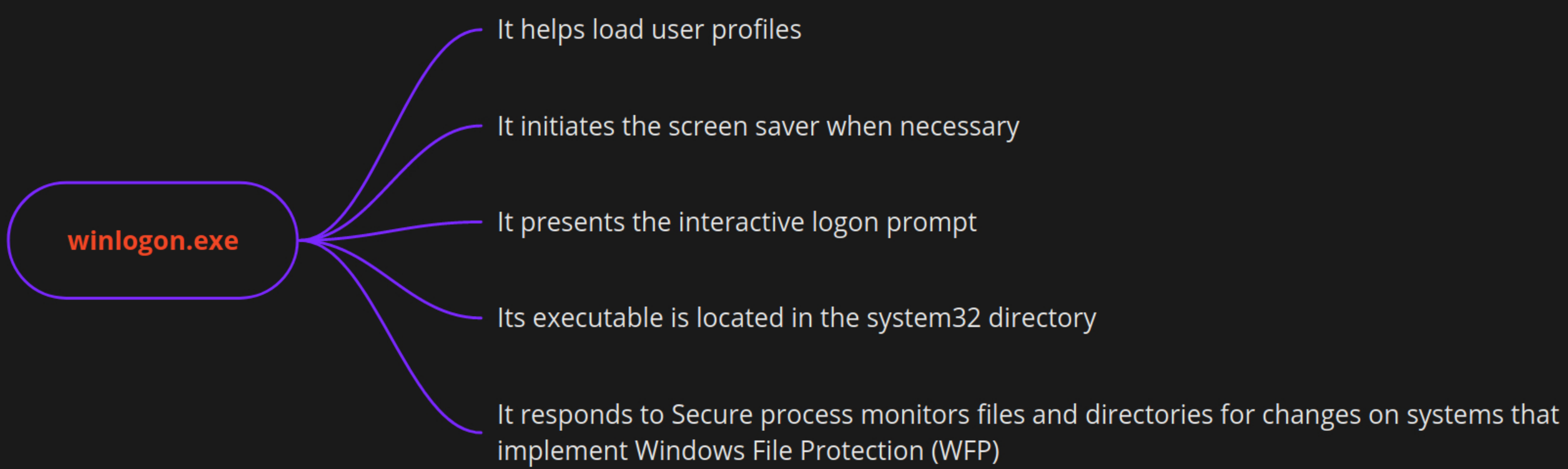

System
It appears to own any sockets or handles to files that kernel modules open
It does not have executable on disk
It's the default home for threads that run in kernel mode
PID 4

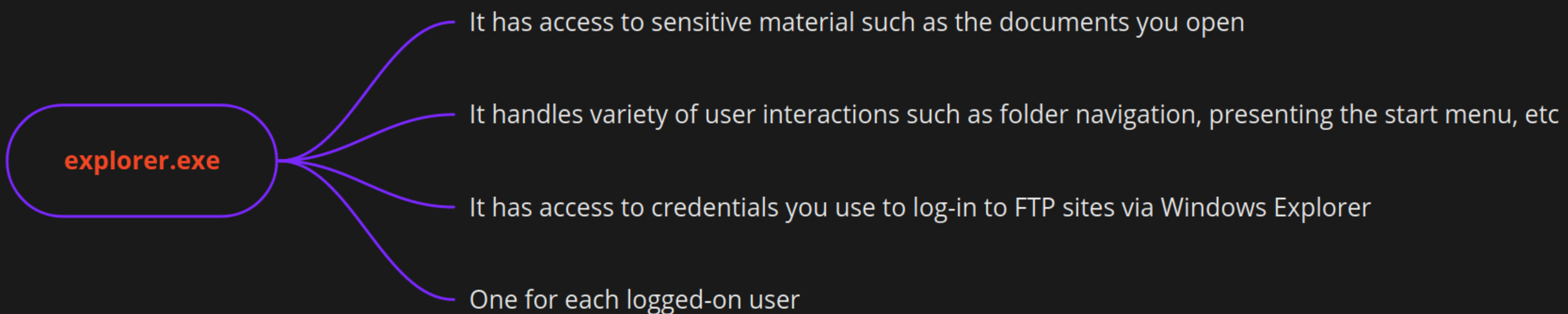

services.exe
It maintains a list of such services in its private memory space
It manages Windows services
The parent to svchost.exe instances, spoolsv.exe and SearchIndexder.exe
Only one copy on a system
It's running from system32 directory

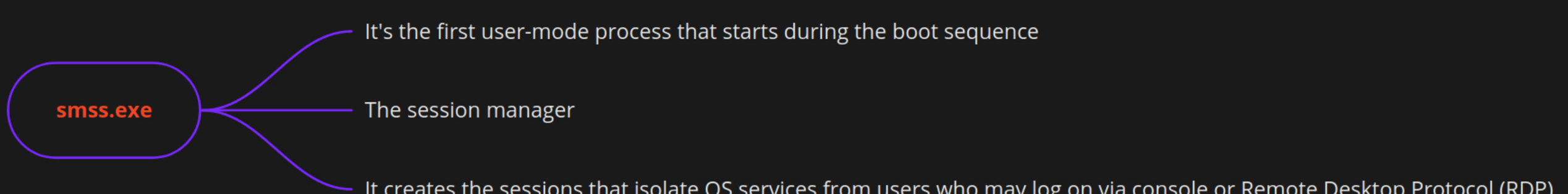

csrss.exe
Attackers attempt to exploit the naming convention (csrsss.exe, cssrs.exe, etc)
It plays role in creating and deleting processes and threads
It's located in the system32 directory
Each session gets a dedicated copy so expect to see multiple csrss processes
The client/server runtime subsystem
It maintains a private list of the objects that can be used to cross-reference with other data sources

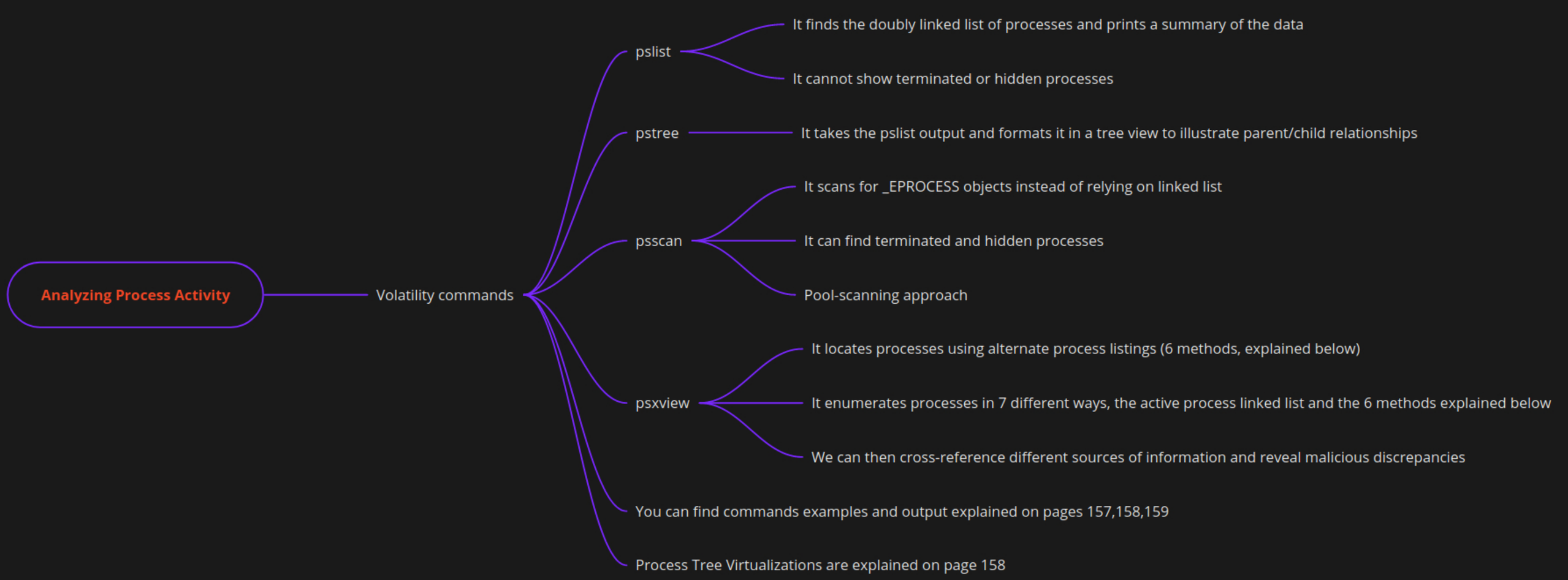

svchost.exe
Attackers attempt to exploit the naming convention (scvhost.exe, svch0st.exe, svshost.exe, etc)
Its executable path should point to system32 directory
Child of services.exe
Multiple shared host processes can be running concurrently
It's a container for DLLs that implements services.

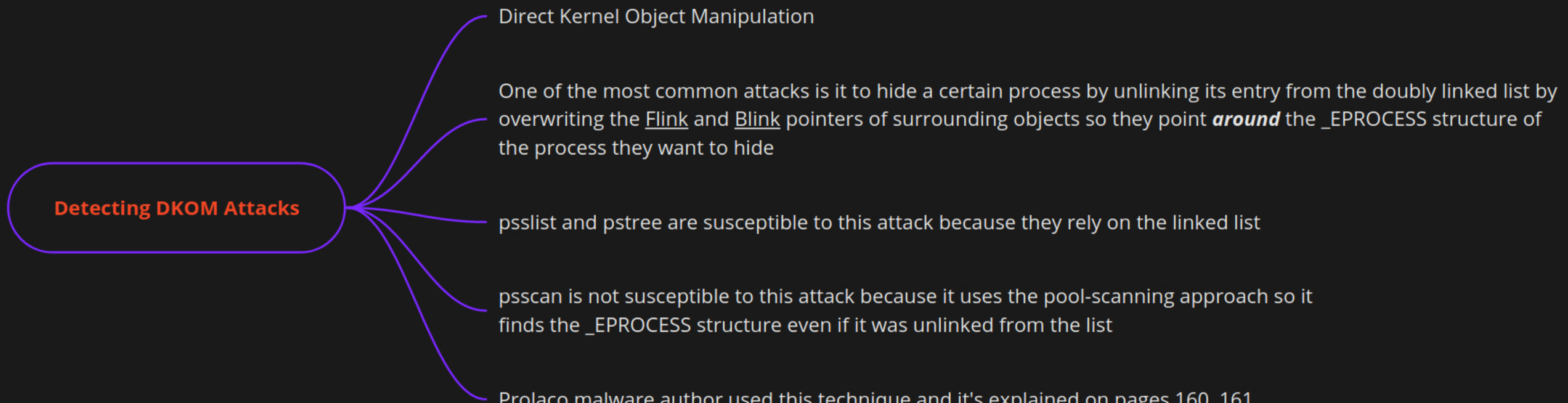

lsass.exe
Child of winint.exe
Only one instance
It's the target of code injection because the plaintext password hashes can be found in its private memory space
It's responsible for enforcing the security policy, verifying passwords and creating access tokens
The local security authority subsystem
It should be running from system32 directory


winlogon.exe
It helps load user profiles
It initiates the screen saver when necessary
It presents the interactive logon prompt
Its executable is located in the system32 directory
It responds to Secure process monitors files and directories for changes on systems that implement Windows File Protection (WFP)


explorer.exe
It has access to sensitive material such as the documents you open
It handles variety of user interactions such as folder navigation, presenting the start menu, etc
It has access to credentials you use to log-in to FTP sites via Windows Explorer
One for each logged-on user

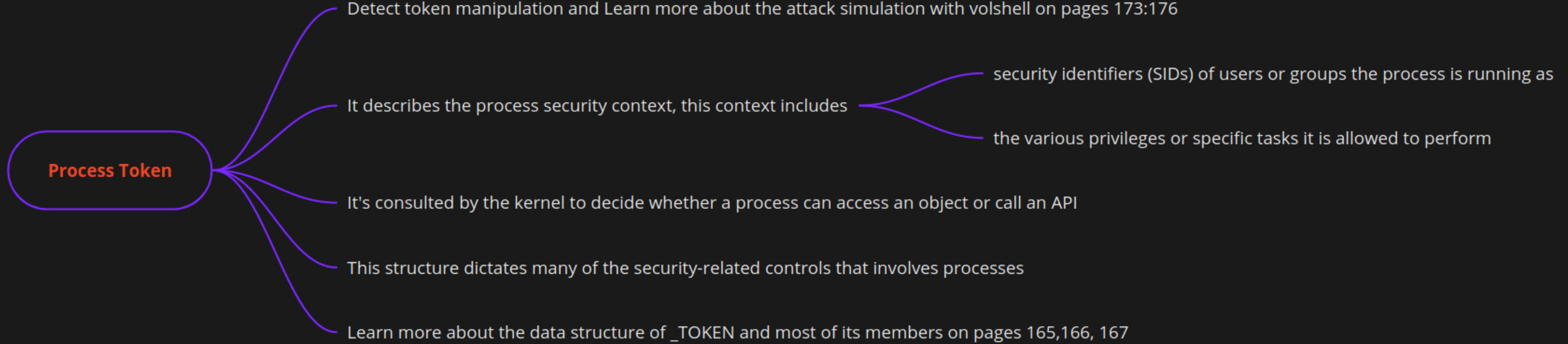

smss.exe
It's the first user-mode process that starts during the boot sequence
The session manager
It creates the sessions that isolate OS services from users who may log on via console or Remote Desktop Protocol (RDP)

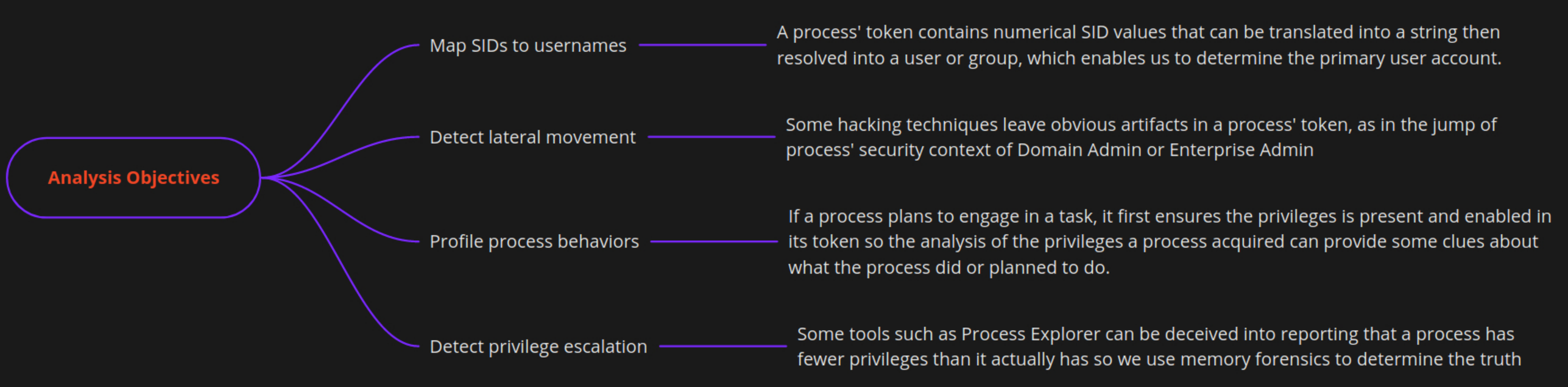

Analyzing Process Activity
Volatility commands
pslist
It finds the doubly linked list of processes and prints a summary of the data
It cannot show terminated or hidden processes
pstree
It takes the pslist output and formats it in a tree view to illustrate parent/child relationships
psscan
It scans for _EPROCESS objects instead of relying on linked list
It can find terminated and hidden processes
Pool-scanning approach
psxview
It locates processes using alternate process listings (6 methods, explained below)
It enumerates processes in 7 different ways, the active process linked list and the 6 methods explained below
We can then cross-reference different sources of information and reveal malicious discrepancies
You can find commands examples and output explained on pages 157,158,159
Process Tree Virtualizations are explained on page 158

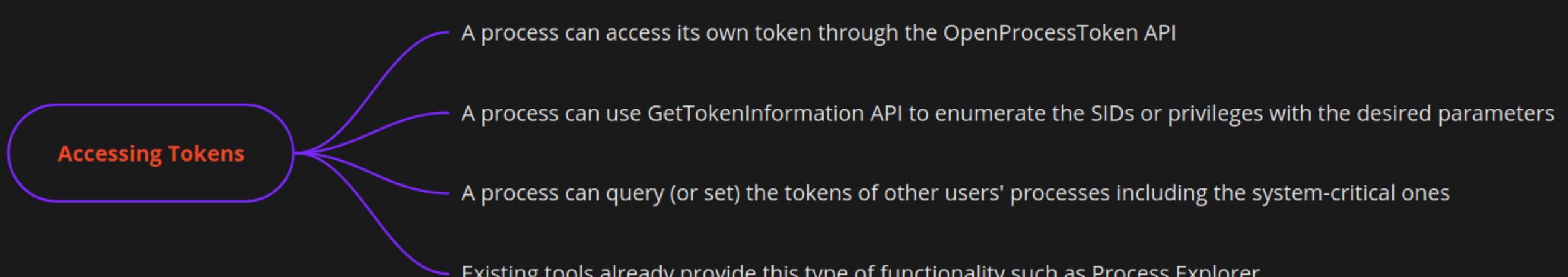

Detecting DKOM Attacks
Direct Kernel Object Manipulation
One of the most common attacks is it to hide a certain process by unlinking its entry from the doubly linked list by overwriting the Flink and Blink pointers of surrounding objects so they point around the _EPROCESS structure of the process they want to hide
psslist and pstree are susceptible to this attack because they rely on the linked list
psscan is not susceptible to this attack because it uses the pool-scanning approach so it finds the _EPROCESS structure even if it was unlinked from the list
Prolaco malware author used this technique and it's explained on pages 160, 161

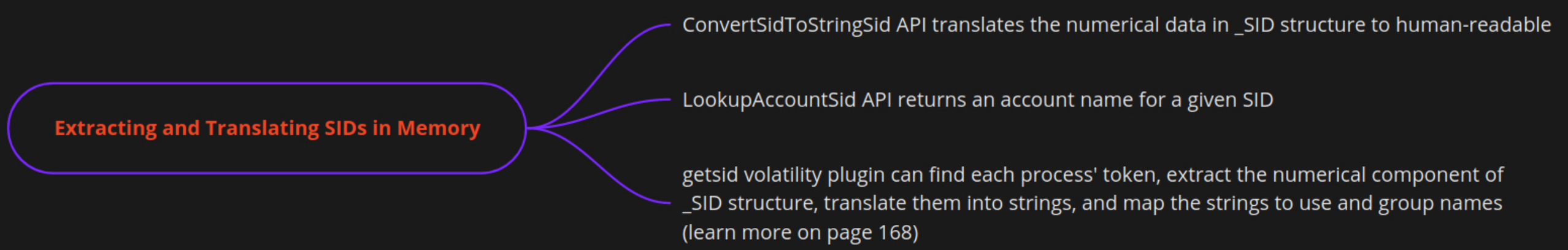
Alternate Process Listings

- Process object scanning
  - As discussed in chapter 5
  - Since the pool tags are not essential, they can manipulated to evade the scanner
- Thread scanning
  - Every process must have at least one thread that execute code
  - Scanner searches for _ETHREAD objects then map them back to their owning processes
  - The member used for mapping is _ETHREAD.Tcb.Process
- CSRSS handle table
  - csrss.exe is involved in creating every process and thread except for itself and the processes started before it
  - Scanner walks the handle table to identify all _EPROCESS objects
- PspCid table
  - It's a special handle table located in kernel memory
  - It stores a reference to all active processes and thread objects
  - The PspCidTable member of the kernel debugger data structure points to the table
- Session processes
  - The SessionProcessLinks member of _EPROCESS associates all processes for every logged-on user session
  - It is not hard to unlink a process from this list, as apposed to the ActiveProcessLinks list
  - Attackers rarely find value in targeting it because live system APIs do not depend on it
- Desktop threads
  - Discussed in details in chapter 14
  - They store list of all threads attached to each desktop
  - The can be easily mapped to their owning process


Process Token
Detect token manipulation and Learn more about the attack simulation with volshell on pages 173:176
It describes the process security context, this context includes
security identifiers (SIDs) of users or groups the process is running as
the various privileges or specific tasks it is allowed to perform
It's consulted by the kernel to decide whether a process can access an object or call an API
This structure dictates many of the security-related controls that involves processes
Learn more about the data structure of _TOKEN and most of its members on pages 165,166, 167

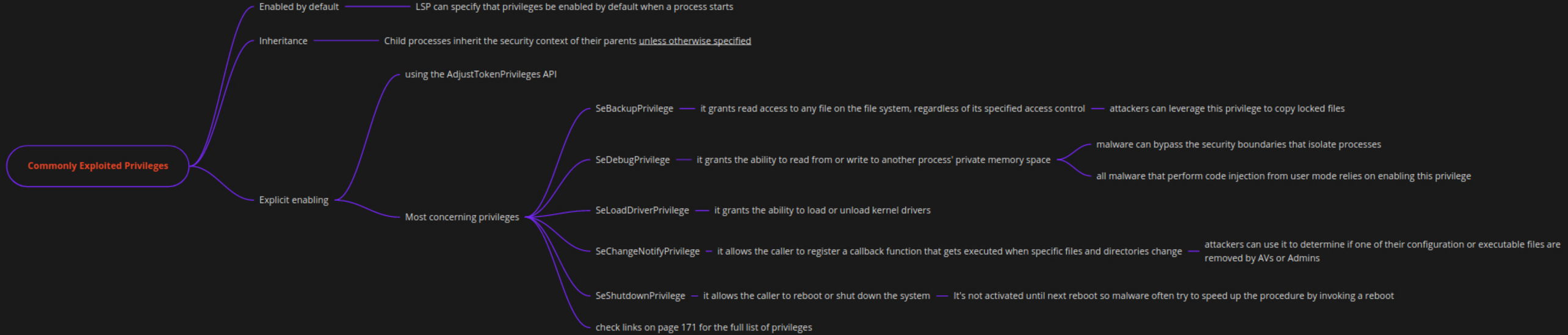

Analysis Objectives
Map SIDs to usernames
A process' token contains numerical SID values that can be translated into a string then resolved into a user or group, which enables us to determine the primary user account.
Detect lateral movement
Some hacking techniques leave obvious artifacts in a process' token, as in the jump of process' security context of Domain Admin or Enterprise Admin
Profile process behaviors
If a process plans to engage in a task, it first ensures the privileges is present and enabled in its token so the analysis of the privileges a process acquired can provide some clues about what the process did or planned to do.
Detect privilege escalation
Some tools such as Process Explorer can be deceived into reporting that a process has fewer privileges than it actually has so we use memory forensics to determine the truth


Accessing Tokens
A process can access its own token through the OpenProcessToken API
A process can use GetTokenInformation API to enumerate the SIDs or privileges with the desired parameters
A process can query (or set) the tokens of other users' processes including the system-critical ones
Existing tools already provide this type of functionality such as Process Explorer


Extracting and Translating SIDs in Memory
ConvertSidToStringSid API translates the numerical data in _SID structure to human-readable
LookupAccountSid API returns an account name for a given SID
getsid volatility plugin can find each process' token, extract the numerical component of _SID structure, translate them into strings, and map the strings to use and group names (learn more on page 168)

Privileges

- A privilege must be present in the process' token before the process can enable it
- Administrators decide which privileges are present by configuring them in the Local Security Policy (LSP) as shown in figure 6-9 on page 171
- Volatility plugin `privs` is used to analyze privileges, learn more from pages 172,173


Commonly Exploited Privileges
Enabled by default
LSP can specify that privileges be enabled by default when a process starts
Inheritance
Child processes inherit the security context of their parents unless otherwise specified
Explicit enabling
using the AdjustTokenPrivileges API
Most concerning privileges
SeBackupPrivilege
it grants read access to any file on the file system, regardless of its specified access control
attackers can leverage this privilege to copy locked files
SeDebugPrivilege
it grants the ability to read from or write to another process' private memory space
malware can bypass the security boundaries that isolate processes
all malware that perform code injection from user mode relies on enabling this privilege
SeLoadDriverPrivilege
it grants the ability to load or unload kernel drivers
SeChangeNotifyPrivilege
it allows the caller to register a callback function that gets executed when specific files and directories change
attackers can use it to determine if one of their configuration or executable files are removed by AVs or Admins
SeShutdownPrivilege
it allows the caller to reboot or shut down the system
It's not activated until next reboot so malware often try to speed up the procedure by invoking a reboot
check links on page 171 for the full list of privileges

- **Process Handles**
  - A handle is a reference to an open instance of a kernel object such as files, registry keys, mutexes, processes or threads
  - There are close to 40 different types of kernel objects
  - Before a process can access an object, it first opens a handle to the object by calling an API such as CreateFile, RegOpenKeyEx, or CreateMutex (example on page 176)
  - Kernel Handles example on page 177
  - Analysis Objectives
    - Handle table internals — learning more about handles internals can give greater understanding of objects
    - Targeted object attribution — with a given indicator such as registry key, we can trace it back to its creator process
    - Detect registry persistence
      - learn how to analyze open registry handles to determine which key is used to store configurations
      - example on pages 183, 184
    - Identify remote mapped drives — example on pages 185,186
  - Handle Table Internals
    - _EPROCESS.ObjectTable member points to a handle table (_HANDLE_TABLE)
    - TableCode serves two critical purposes
      - it specifies the number of levels in the table
      - it points to the base address of the first level
    - All processes start with a single-level table
      - example on page 178,179
      - Figure 6-12 on page 179
    - Data structures on page 179,180
  - The volatility plugin `handles` is used to enumerate handles, example on pages 181,182