Andrea Oliveri (a.k.a. Iridium)

-> PhD student at Eurecom in S3 research group, under the supervision of Prof. Davide Balzarotti.

**Research interests:**

-> Zero-knowledge memory forensics
-> Reversing engineering
-> CPUs/machines architectures
-> Network security
● International University (Grande École) and research center in digital sciences in Sophia-Antipolis (France)

● 9 universities and 7 industry partners

● 3 principal research themes:
  ○ Telecommunication
  ○ Data science
  ○ Cybersecurity

● 4 ERC grants
Software and system security group (S3)

- **3 faculty members:**
  - Prof. Davide Balzarotti
  - Prof. Aurelien Francillon
  - Prof. Daniele Antonioli

- **1 Post-Doc**
- **13 PhD students**

- **Research areas:**
  - Android Security
  - Malware Collection, Detection and Analysis
  - Memory Forensics
  - Security in Embedded Systems
  - System Security
  - Web Security
What is digital forensics?

Digital forensics is the **application of investigation and analysis techniques to gather and preserve evidence from a particular computing device**.
Why digital forensics?

- Police investigation
- Incident response
- Malware analysis
Where are the evidence?

- Packets and connections
- Services
- Transferred data
- Firewall logs
- Volumes and partitions
- File systems
- Data in empty regions
- OS & apps artifacts
Where are the evidence?

- Packets and connections
- Services
- Transferred data
- Firewall logs

- Volumes and partitions
- File systems
- Data in empty regions
- OS & apps artifacts

- Volatile memory (RAM)
Memory Forensics

The process of capturing a copy of the system memory (RAM) and extracting a number of artifacts that are useful for an investigation

- Essentially, **it consists in:**
  - **Acquiring a snapshot** of the system memory
  - **Locating known data structures** in the raw image to extract OS and process information
  - **Carving** de-allocated data structures, strings, encryption keys, credit card numbers...

- Relatively new field (~2005), and still a very active research area
Available Information

- Running, terminated, and hidden processes
  - Code, data, open files, buffers, ...
- Kernel modules, drivers and privileged services
- Hardware devices connected at dump time
- Open sockets and active connections
- Memory-mapped files
  - Executables, shared libraries, ...
- Clipboard's content
- Browsers data
  - History, cached pages, passwords, ...
- Cached data
  - Open documents, emails and IM messages, ...
Memory Forensics - Pro

- Memory is relatively small compared to hard disks
- Attackers often overlook their memory footprint
- Many of the kernel artifacts can be used for forensics
- Even rootkits designed to hide data in a running system need to be located somewhere in memory
- Certain information (loaded kernel modules, open sockets, ...) may be difficult to extract otherwise
- Some malware samples only reside in memory
Memory Forensics - Cons

- On physical machines the content of the memory keeps changing
  - Data collection requires an efficient approach with a small footprint
  - Consecutive imaging acquisitions give different results
  - Forget about comparing hashes

  ⇒ It is impossible to verify the authenticity of the acquired data

- Data structures change among different OSs and OS versions

- Semantic Gap: going from a raw sequence of bytes to high-level artifacts
Memory Acquisition
Memory Acquisition Internals

- Physical address space is not continuous!
  - It contains physical RAM regions and other stuff...
  - `# cat /proc/iomem` <- show the physical memory layout of the system
  - Hardware peripherals map registers or parts of their integrated memory into the physical address space via Memory Mapped I/O
  - Any attempt to read the memory mapped to a device would probably crash the system

- Some parts of the RAM contain encrypted data or fundamental code needed for the system (e.g, Intel SGX, UEFI code, SMRAM...
Memory Acquisition

- **Software Acquisition**
  - Relies on a program running inside the system to read and store a copy of the memory
  - The software is altering the system, so its footprint is very important

- **Hardware Acquisition**
  - Relies on hardware devices to read the memory, often bypassing the CPU
  - It does not introduce any new artifact in the system

- Most of the existing approaches don't freeze the system during the acquisition, potentially (often!) leading to inconsistencies

- **Special case: Virtual machines**
  - If the analyst has access to the hypervisor it is possible to perform an atomic dump
  - New technologies like AMD Secure Encrypted Virtualization can block any type of memory dump from the hypervisor
Software Acquisition

- **Pseudo-device files**
  - A file-like device that can be copied using `dd`
  - `/dev/mem` and `/dev/kmem` in Linux
    - Access to user- and kernel-memory
    - Not available on recent systems
  - `\\\PhysicalMemory` in Windows
    - Not accessible from user space since Windows 2003 SP1

- **Kernel-mode drivers**
  - Exist to overcome the previous limitations
  - On modern systems with Secure Boot enabled a signed driver is required

- **Special tools**
  - Acquire memory from special CPU modes/special code regions (e.g., SMM mode, IPMI controllers, Intel ME, UEFI code,...)
Crash Dumps

- Windows can be configured to create a full memory dump in response to a Blue Screen of Death (BSoD)
  - Dumps are created in the swap area and then copied to regular files at the next boot
  - Memory dumps can be forced by pressing CTRL + Scroll Lock (x2)

- Very accurate
  - the system is frozen while the dump is taken, allowing to take an atomic snapshot

- Impact on disk analysis
  - several GB are written to the disk, possibly overwriting other evidence

- Hard to deploy:
  - several configuration options that need to be set in advance in the registry or a system reboot
Hibernation Files

- When the OS is hibernated (suspended to disk) a copy of the RAM is stored in a file (hiberfil.sys in Windows, swap partition in Linux)

- If hibernation is supported, this is a good method to obtain a memory dump, with few limitations (e.g., it works in 64-bit architecture with more than 4GB or RAM)
Tools

- **winpmem**
  - Supports *Windows XP SP2* to *Windows 10*
  - Outputs **raw dumps** or **AFF4 dumps**
    (an archive that can contain multiple streams and additional information)
  - A version for Mac OS is also available (*OSXPmem*) and for Linux (*linpmem*)

- **LiME**
  - **Linux kernel module**
  - Outputs **raw dumps** or **LiME dumps**
  - Permits to **save dumps on external disk** or **send it through the network**
Hardware Acquisition

- **Based on DMA transfer on an external bus**
  - requires to have that bus on the motherboard, have access to it, and some security features not enabled (e.g., IOMMU)
  - **Not forensically sound**

- **Firewire**
  - Directly access the main memory from a FireWire device
  - Not common, no more used

- **PCI-Express**
  - PCILeech, FPGA
  - Internal acquisition cards which need to be connected to the bus **before** the incident

- **Thunderbolt / USB 4**

- **Intel DCI (on Intel CPUs only) / JTAG**
Cold Boot Attack

● Main memory is normally stored in DRAM chips
  ○ Information is stored in a capacitor, whose charge needs to be refreshed every few milliseconds (or the charge would decay to the ground state)
  ○ If not refreshed, the content of DRAM is completely lost after several seconds (the actual time depends on the machine)
  ○ **RAM is not zeroed at reboot!**

● Acquisition:
  ○ Cut the power of the target machine
  ○ Boot from network or from a USB drive
  ○ Copy the RAM – the process is relatively fast (~30 sec per GB over the network to 4 min over USB)
  ○ On modern systems with UEFI this technique is **no more a viable option** (due to MOR bit protection)
  ○ But what if the computer is not configured to boot from network or USB?
Memory Degradation at Room Temperature

After: 5 seconds  30 seconds  60 seconds  5 minutes
Mona Lisa on the Rocks

- A multi-purpose “canned air” duster spray canister, held upside-down, discharges very cold liquid refrigerant instead of gas.
- It can be used as a fast and cheap refrigerant for memory chips :)
- 1% of bits decayed after 10 minutes
- Drop it in a liquid nitrogen can, and information is preserved unchanged for hours
- 0.17% bits decayed in 60 min
On the Practicability of Cold Boot Attacks

- The attack was originally designed for DDR1 and DDR2

- On DDR3, the memory controller scrambles the data before writing it to memory to reduce electromagnetic interference
  - Intel uses a Linear-feedback shift registers that can be reverted if a small plaintext is known
  - See “Lest we forget: Cold-boot attacks on scrambled DDR3 memory” (DFRWS EU 2016) for more information
Memory Acquisition: RESUME

- **Software Acquisition**
  - Easy method but... software alters the system

- **Hardware Acquisition**
  - Does not introduce new artifacts but.. relies on hardware devices

- **Best case: Virtual machines** *(without AMD SEV!)*
  - Fast, easy atomic dump
Memory Analysis
Address Translation

● Each process lives in a (quasi) separated Virtual Address Space
● The virtual memory system stores the **mapping between virtual and physical addresses** inside Page Tables
● The actual address translation depends on the CPU architecture and on certain CPU registers
Strings

- A freshly booted machine roughly generates ~100MB of strings per GB of RAM*
  - ~580,000 strings of length 5
  - ~66,000 Unicode strings of length 5
- Starting notepad and IDA adds another ~7,000 new Unicode strings!!

⇒ Much better to focus on something in particular (IP addresses, email headers, ...)

*Post-Mortem RAM Forensics – Tim Vidas 2007
Locating Structures

- **Fixed offsets**
  - Useful to find the kernel base image

- **Data structure traversal** (list walking, tree climbing, ...)
  - Extract allocated data structures
  - Requires knowledge of the kernel internals

- **Linear scanning**
  - Search the memory for known patterns
  - It can detect de-allocated structures
  - Requires knowledge of the kernel internals
  - Fields validation to reduce false positives
    - Permitted values
    - Pointers target
Problems

- The modeling and extraction tools need a *very precise definition of the kernel structures*

- But *kernel structures change quite rapidly and they are often unknown*
  - In *Windows*, source code are not always available for documentation
  - In *Linux*, users can install different kernel, apply patches, or *recompile the kernel* from scratch
  - In general a deep knowledge of the kernel internals is required

- ⇒ *We need a profile*: for each operating system, and each version of it we need a description and location of all the kernel data structures necessary to perform the analysis!
Interesting Structures
(from now on, we are talking about Windows)

- Each process is identified by an Executive Process Block (EPROCESS)

- All the EPROCESS are connected in a double linked list (EPROCESS→ActiveProcessLinks→flink/blink)
  - Processes are removed from the list when they exit
  - Rootkits often hide processes by taking them out of the linked list

- The EPROCESS structure contains a link to the Process Environment Block (PEB) located in the process address space
Interesting Info

- **Eprocess**
  - Creation and Exit time
  - Process ID and parent Process ID (who started the process)
  - Pointer to the handle table
  - Virtual Address Space descriptors (VAD)

- **PEB**
  - Pointer to the **image base address** (where you can find the executable image)
  - Pointer to the **process parameters structure** (full path of binary, DLLs, and command line used to start the executable)
  - Pointer to the **DLLs loaded** by the process (three lists, ordered by loading time, initialization time, and memory address)
  - **Heap size** information (the pointer to the heap is located just after the PEB structure)
Kernel Global Variables

- A number of **hidden kernel variables are extremely helpful** to examine the state of the running system
  - **PsActiveProcessHead** points to the start of the kernel's list of EPROCESS structures
  - **PsLoadedModuleList** points to the list of currently **loaded kernel modules**
  - **HandleTableListHead** points to the head of list of handle tables (resources used by each process)
  - **MmPfnDatabase** is an array of structures describing each **physical page in the system**
Locating Kernel Variables

- The **variables are always at a fixed location** in memory
  - But unfortunately the **location changes between Windows versions**, patch levels, and even single hotfixes

- **Windows keep a structure** (_KDDEBUGGER_DATA64) for debugging purposes, that contains the memory address of dozens of global kernel variables
  - In Windows {XP, 2003, Vista} **this structure can be found through a KPCR structure** that is located at a fixed address in memory
  - In Windows 2000, the structure has to be located by scanning the memory
  - **Windows 8 encodes the KDBG block making memory analysis more difficult**
Volatility

- Open source memory analysis framework written in Python

- Supports:
  - 32-bit and 64-bit Windows OSs
  - Linux 32 and 64-bit
  - macOS
  - FreeBSD
  - Android

- Volatility supports raw dumps, crash dump, virtual machine snapshots, and hibernation files
Volatility Plugins

- **Collection of tools** implemented as plugins
- **Plugins are just Python scripts** and can be easily installed by copying them into the plugin directory
  - The current version contains ~50 profiles and ~265 plugins
  - A few plugins have been developed specifically to find signs of malware infections
  - Additional (more research-oriented) plugins implemented by Brendan Dolan-Gavitt
    - [http://www.cc.gatech.edu/~brendan/volatility/](http://www.cc.gatech.edu/~brendan/volatility/)
  - Volatility plugins developed and maintained by the community:
    - [https://github.com/volatilityfoundation/community](https://github.com/volatilityfoundation/community)
- `$ vol.py --info` → list the available plugins
Catching the bad Guys
Starting the memory analysis

- Suppose you have received a memory dump of a machine which performs suspected activity... How to perform a memory analysis?

  - The analysis often starts by listing and investigating the processes that were running in the system
    - Open files, loaded DLLs, or network sockets can help identifying suspicious cases
    - The starting point for the analysis may come from another source (e.g., a network sensor detected a suspicious connection)

  - The analysis also includes the inspection of the kernel, to locate malicious kernel modules

  - The analysis may end when you locate a known malicious file, or dump an unknown suspicious file that require some further binary analysis
Image Identification

```bash
andrea@ubuntu-20:~/volatility$ ./vol.py -f ~/Downloads/blank.vmem --profile=WinXPSP3x86 imageinfo
Volatility Foundation Volatility Framework 2.6.1
INFO : volatility.debug : Determining profile based on KDBG search...
  Suggested Profile(s) : WinXPSP2x86, WinXPSP3x86
  AS Layer1 : IA32PagedMemoryPae (Kernel AS)
  AS Layer2 : FileAddressSpace (/home/andrea/Downloads/blank.vmem)
  PAE type : PAE
    DTB : 0x3190000L
    KDBG : 0x80545ae0L
  Number of Processors : 1
  Image Type (Service Pack) : 3
    KPCR for CPU 0 : 0xffdff0000L
    KUSER_SHARED_DATA : 0xffdf0000L
  Image date and time : 2011-06-03 04:31:36 UTC+0000
  Image local date and time : 2011-06-03 00:31:36 -0400
```
Detecting Malicious Processes

Hidden through DKOM (Direct Kernel Object Manipulation), by removing the process from the EProcess linked list or disguised by renaming the process to match a system or innocuous one.

- Compare the output of the **plist** and **pscan** plugins
- **Psxview** outputs the list of process extracted in six different ways
### List of Processes

<table>
<thead>
<tr>
<th>Offset(V)</th>
<th>Name</th>
<th>PID</th>
<th>PPID</th>
<th>Thds</th>
<th>Hnds</th>
<th>Sess</th>
<th>Wow64</th>
<th>Start</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x823c8830</td>
<td>System</td>
<td>4</td>
<td>0</td>
<td>59</td>
<td>403</td>
<td>------</td>
<td>0</td>
<td>2010-10-29 17:08:53 UTC+0000</td>
</tr>
<tr>
<td>0x820df020</td>
<td>smss.exe</td>
<td>376</td>
<td>4</td>
<td>3</td>
<td>19</td>
<td>------</td>
<td>0</td>
<td>2010-10-29 17:08:54 UTC+0000</td>
</tr>
<tr>
<td>0x821a2da0</td>
<td>csrss.exe</td>
<td>600</td>
<td>376</td>
<td>11</td>
<td>395</td>
<td>0</td>
<td>0</td>
<td>2010-10-29 17:08:54 UTC+0000</td>
</tr>
<tr>
<td>0x81da5650</td>
<td>winlogon.exe</td>
<td>624</td>
<td>376</td>
<td>19</td>
<td>570</td>
<td>0</td>
<td>0</td>
<td>2010-10-29 17:08:54 UTC+0000</td>
</tr>
<tr>
<td>0x82073020</td>
<td>services.exe</td>
<td>668</td>
<td>624</td>
<td>21</td>
<td>431</td>
<td>0</td>
<td>0</td>
<td>2010-10-29 17:08:54 UTC+0000</td>
</tr>
<tr>
<td>0x81e70020</td>
<td>lsass.exe</td>
<td>680</td>
<td>624</td>
<td>19</td>
<td>342</td>
<td>0</td>
<td>0</td>
<td>2010-10-29 17:08:54 UTC+0000</td>
</tr>
<tr>
<td>0x823315d8</td>
<td>vmacthlp.exe</td>
<td>844</td>
<td>668</td>
<td>1</td>
<td>25</td>
<td>0</td>
<td>0</td>
<td>2010-10-29 17:08:55 UTC+0000</td>
</tr>
<tr>
<td>0x81fc5da0</td>
<td>VMwareTray.exe</td>
<td>1912</td>
<td>1196</td>
<td>1</td>
<td>50</td>
<td>0</td>
<td>0</td>
<td>2010-10-29 17:11:50 UTC+0000</td>
</tr>
<tr>
<td>0x81e6b660</td>
<td>VMwareUser.exe</td>
<td>1356</td>
<td>1196</td>
<td>9</td>
<td>251</td>
<td>0</td>
<td>0</td>
<td>2010-10-29 17:11:50 UTC+0000</td>
</tr>
<tr>
<td>0x82100478</td>
<td>juspced.exe</td>
<td>1712</td>
<td>1196</td>
<td>1</td>
<td>26</td>
<td>0</td>
<td>0</td>
<td>2010-10-29 17:11:50 UTC+0000</td>
</tr>
<tr>
<td>0x82279998</td>
<td>imapi.exe</td>
<td>756</td>
<td>668</td>
<td>4</td>
<td>116</td>
<td>0</td>
<td>0</td>
<td>2010-10-29 17:11:54 UTC+0000</td>
</tr>
<tr>
<td>0x8222ba10</td>
<td>wuaucatl.exe</td>
<td>976</td>
<td>1032</td>
<td>3</td>
<td>133</td>
<td>0</td>
<td>0</td>
<td>2010-10-29 17:12:03 UTC+0000</td>
</tr>
<tr>
<td>0x81c543a0</td>
<td>Procmon.exe</td>
<td>660</td>
<td>1196</td>
<td>13</td>
<td>189</td>
<td>0</td>
<td>0</td>
<td>2011-06-03 04:25:56 UTC+0000</td>
</tr>
<tr>
<td>0x81fa5390</td>
<td>wmiprvse.exe</td>
<td>1872</td>
<td>856</td>
<td>5</td>
<td>134</td>
<td>0</td>
<td>0</td>
<td>2011-06-03 04:25:58 UTC+0000</td>
</tr>
<tr>
<td>0x81c498c8</td>
<td>lsass.exe</td>
<td>868</td>
<td>668</td>
<td>2</td>
<td>23</td>
<td>0</td>
<td>0</td>
<td>2011-06-03 04:26:55 UTC+0000</td>
</tr>
<tr>
<td>0x81c47c00</td>
<td>lsass.exe</td>
<td>1928</td>
<td>668</td>
<td>4</td>
<td>65</td>
<td>0</td>
<td>0</td>
<td>2011-06-03 04:26:55 UTC+0000</td>
</tr>
<tr>
<td>0x81c0cda0</td>
<td>cmd.exe</td>
<td>968</td>
<td>1664</td>
<td>0</td>
<td>------</td>
<td>0</td>
<td>0</td>
<td>2011-06-03 04:31:35 UTC+0000</td>
</tr>
</tbody>
</table>
### List of Processes

<table>
<thead>
<tr>
<th>Offset(V)</th>
<th>Name</th>
<th>PID</th>
<th>PPID</th>
<th>Thds</th>
<th>Hnds</th>
<th>Sess</th>
<th>Wow64</th>
<th>Start</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x823c8830</td>
<td>System</td>
<td>4</td>
<td>0</td>
<td>59</td>
<td>403</td>
<td>------</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0x820df020</td>
<td>smss.exe</td>
<td>376</td>
<td>4</td>
<td>3</td>
<td>19</td>
<td>------</td>
<td></td>
<td>0 2010-10-29 17:08:53 UTC+0000</td>
</tr>
<tr>
<td>0x821a2da0</td>
<td>csrss.exe</td>
<td>600</td>
<td>376</td>
<td>11</td>
<td>395</td>
<td>0</td>
<td></td>
<td>0 2010-10-29 17:08:54 UTC+0000</td>
</tr>
<tr>
<td>0x81da5650</td>
<td>winlogon.exe</td>
<td>624</td>
<td>376</td>
<td>19</td>
<td>570</td>
<td>0</td>
<td></td>
<td>0 2010-10-29 17:08:54 UTC+0000</td>
</tr>
<tr>
<td>0x82073020</td>
<td>services.exe</td>
<td>668</td>
<td>624</td>
<td>21</td>
<td>431</td>
<td>0</td>
<td></td>
<td>0 2010-10-29 17:08:54 UTC+0000</td>
</tr>
<tr>
<td>0x81e70020</td>
<td>lsass.exe</td>
<td>680</td>
<td>624</td>
<td>19</td>
<td>342</td>
<td>0</td>
<td></td>
<td>0 2010-10-29 17:08:54 UTC+0000</td>
</tr>
<tr>
<td>0x823315d8</td>
<td>vmacctlp.exe</td>
<td>844</td>
<td>668</td>
<td>1</td>
<td>25</td>
<td>0</td>
<td></td>
<td>0 2010-10-29 17:08:55 UTC+0000</td>
</tr>
<tr>
<td>0x81fc5da0</td>
<td>VMwareTray.exe</td>
<td>1912</td>
<td>1196</td>
<td>1</td>
<td>50</td>
<td>0</td>
<td></td>
<td>0 2010-10-29 17:11:50 UTC+0000</td>
</tr>
<tr>
<td>0x81e6b660</td>
<td>VMwareUser.exe</td>
<td>1356</td>
<td>1196</td>
<td>9</td>
<td>251</td>
<td>0</td>
<td></td>
<td>0 2010-10-29 17:11:50 UTC+0000</td>
</tr>
<tr>
<td>0x8210d478</td>
<td>juscled.exe</td>
<td>1712</td>
<td>1196</td>
<td>1</td>
<td>26</td>
<td>0</td>
<td></td>
<td>0 2010-10-29 17:11:50 UTC+0000</td>
</tr>
<tr>
<td>0x82279998</td>
<td>imapi.exe</td>
<td>756</td>
<td>668</td>
<td>4</td>
<td>116</td>
<td>0</td>
<td></td>
<td>0 2010-10-29 17:11:54 UTC+0000</td>
</tr>
<tr>
<td>0x822b9a10</td>
<td>wuauclt.exe</td>
<td>976</td>
<td>1032</td>
<td>3</td>
<td>133</td>
<td>0</td>
<td></td>
<td>0 2010-10-29 17:12:03 UTC+0000</td>
</tr>
<tr>
<td>0x81c543a0</td>
<td>Procmon.exe</td>
<td>660</td>
<td>1196</td>
<td>13</td>
<td>189</td>
<td>0</td>
<td></td>
<td>0 2011-06-03 04:25:56 UTC+0000</td>
</tr>
<tr>
<td>0x81fa5390</td>
<td>wmpiprvse.exe</td>
<td>1872</td>
<td>856</td>
<td>5</td>
<td>134</td>
<td>0</td>
<td></td>
<td>0 2011-06-03 04:25:58 UTC+0000</td>
</tr>
<tr>
<td>0x81c498c8</td>
<td>lsass.exe</td>
<td>868</td>
<td>668</td>
<td>2</td>
<td>23</td>
<td>0</td>
<td></td>
<td>0 2011-06-03 04:26:55 UTC+0000</td>
</tr>
<tr>
<td>0x81c47c00</td>
<td>lsass.exe</td>
<td>1928</td>
<td>668</td>
<td>4</td>
<td>65</td>
<td>0</td>
<td></td>
<td>0 2011-06-03 04:26:55 UTC+0000</td>
</tr>
<tr>
<td>0x81c0cda0</td>
<td>cmd.exe</td>
<td>968</td>
<td>1664</td>
<td>0</td>
<td>------</td>
<td>0</td>
<td></td>
<td>0 2011-06-03 04:31:35 UTC+0000</td>
</tr>
</tbody>
</table>
LSASS a.k.a. Local Security Authentication Subsystem Service

- Responsible for authenticating users
- Only one per system 🔥
- Associated with Local System Account
- Parent process: winlogon.exe
- Executable in %SystemRoot%\System32\lsass.exe
- Starts within seconds of boot times
Process SIDs

Andrea@ubuntu-20:/volatility$ ./vol.py -f ~/Downloads/ volmem --profile=WinXPSP3x86 getsids -p 680,868,1928
Volatility Foundation Volatility Framework 2.6.1

lsass.exe (680): S-1-5-18 (Local System)
lsass.exe (680): S-1-5-32-544 (Administrators)
lsass.exe (680): S-1-1-0 (Everyone)
lsass.exe (680): S-1-5-11 (Authenticated Users)
lsass.exe (868): S-1-5-18 (Local System)
lsass.exe (868): S-1-5-32-544 (Administrators)
lsass.exe (868): S-1-1-0 (Everyone)
lsass.exe (868): S-1-5-11 (Authenticated Users)
lsass.exe (1928): S-1-5-18 (Local System)
lsass.exe (1928): S-1-5-32-544 (Administrators)
lsass.exe (1928): S-1-1-0 (Everyone)
lsass.exe (1928): S-1-5-11 (Authenticated Users)
LSASS a.k.a. Local Security Authentication Subsystem Service

- Responsible for **authenticating users**
- Only **one** per system 🔥
- Associated with **Local System Account** ✅
- Parent process: winlogon.exe
- Executable in `%SystemRoot%\System32\lsass.exe`
- Starts within seconds of boot times
<table>
<thead>
<tr>
<th>Offset(V)</th>
<th>Name</th>
<th>PID</th>
<th>PPID</th>
<th>Thds</th>
<th>Hnds</th>
<th>Sess</th>
<th>Wow64</th>
<th>Start</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x823c8830</td>
<td>System</td>
<td>4</td>
<td>0</td>
<td>59</td>
<td>403</td>
<td>------</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0x820df020</td>
<td>smss.exe</td>
<td>376</td>
<td>4</td>
<td>3</td>
<td>19</td>
<td>------</td>
<td>0</td>
<td>2010-10-29 17:08:53 UTC+0000</td>
</tr>
<tr>
<td>0x821a2da0</td>
<td>csrss.exe</td>
<td>600</td>
<td>376</td>
<td>11</td>
<td>395</td>
<td>0</td>
<td>0</td>
<td>2010-10-29 17:08:54 UTC+0000</td>
</tr>
<tr>
<td>0x81da5650</td>
<td>winlogon.exe</td>
<td>624</td>
<td>376</td>
<td>19</td>
<td>570</td>
<td>0</td>
<td>0</td>
<td>2010-10-29 17:08:54 UTC+0000</td>
</tr>
<tr>
<td>0x82073020</td>
<td>services.exe</td>
<td>668</td>
<td>624</td>
<td>21</td>
<td>431</td>
<td>0</td>
<td>0</td>
<td>2010-10-29 17:08:54 UTC+0000</td>
</tr>
<tr>
<td>0x81e70020</td>
<td>lsass.exe</td>
<td>680</td>
<td>624</td>
<td>19</td>
<td>342</td>
<td>0</td>
<td>0</td>
<td>2010-10-29 17:08:54 UTC+0000</td>
</tr>
<tr>
<td>0x823315d8</td>
<td>vmaccthr.exe</td>
<td>844</td>
<td>668</td>
<td>1</td>
<td>25</td>
<td>0</td>
<td>0</td>
<td>2010-10-29 17:08:55 UTC+0000</td>
</tr>
<tr>
<td>0x81fc5da0</td>
<td>VMwareTray.exe</td>
<td>1912</td>
<td>1196</td>
<td>1</td>
<td>50</td>
<td>0</td>
<td>0</td>
<td>2010-10-29 17:11:50 UTC+0000</td>
</tr>
<tr>
<td>0x81e6b660</td>
<td>VMwareUser.exe</td>
<td>1356</td>
<td>1196</td>
<td>9</td>
<td>251</td>
<td>0</td>
<td>0</td>
<td>2010-10-29 17:11:50 UTC+0000</td>
</tr>
<tr>
<td>0x8210d478</td>
<td>jusesched.exe</td>
<td>1712</td>
<td>1196</td>
<td>1</td>
<td>26</td>
<td>0</td>
<td>0</td>
<td>2010-10-29 17:11:50 UTC+0000</td>
</tr>
<tr>
<td>0x82279998</td>
<td>imapi.exe</td>
<td>756</td>
<td>668</td>
<td>4</td>
<td>116</td>
<td>0</td>
<td>0</td>
<td>2010-10-29 17:11:54 UTC+0000</td>
</tr>
<tr>
<td>0x822b9a10</td>
<td>wuauscredited.exe</td>
<td>976</td>
<td>1032</td>
<td>3</td>
<td>133</td>
<td>0</td>
<td>0</td>
<td>2010-10-29 17:12:03 UTC+0000</td>
</tr>
<tr>
<td>0x81c543a0</td>
<td>Procmom.exe</td>
<td>660</td>
<td>1196</td>
<td>13</td>
<td>189</td>
<td>0</td>
<td>0</td>
<td>2011-06-03 04:25:56 UTC+0000</td>
</tr>
<tr>
<td>0x81fa5390</td>
<td>wmicprintf.exe</td>
<td>1872</td>
<td>856</td>
<td>5</td>
<td>134</td>
<td>0</td>
<td>0</td>
<td>2011-06-03 04:25:58 UTC+0000</td>
</tr>
<tr>
<td>0x81c498c8</td>
<td>lsass.exe</td>
<td>868</td>
<td>668</td>
<td>2</td>
<td>23</td>
<td>0</td>
<td>0</td>
<td>2011-06-03 04:26:55 UTC+0000</td>
</tr>
<tr>
<td>0x81c47c00</td>
<td>lsass.exe</td>
<td>1928</td>
<td>668</td>
<td>4</td>
<td>65</td>
<td>0</td>
<td>0</td>
<td>2011-06-03 04:26:55 UTC+0000</td>
</tr>
<tr>
<td>0x81c0cda0</td>
<td>cmd.exe</td>
<td>968</td>
<td>1664</td>
<td>0</td>
<td>------</td>
<td>0</td>
<td>0</td>
<td>2011-06-03 04:31:35 UTC+0000</td>
</tr>
</tbody>
</table>
LSASS a.k.a. Local Security Authentication Subsystem Service

- Responsible for authenticating users
- Only one per system 🔥
- Associated with Local System Account ✅
- Parent process: winlogon.exe 🔥
- Executable in %SystemRoot%\System32\lsass.exe
- Starts within seconds of boot times
<table>
<thead>
<tr>
<th>Offset(V)</th>
<th>Name</th>
<th>PID</th>
<th>PPID</th>
<th>Thds</th>
<th>Hnds</th>
<th>Sess</th>
<th>Wow64</th>
<th>Start</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x823c8830</td>
<td>System</td>
<td>4</td>
<td>0</td>
<td>59</td>
<td>403</td>
<td>------</td>
<td></td>
<td>0 2010-10-29 17:08:53 UTC+0000</td>
</tr>
<tr>
<td>0x820df020</td>
<td>smss.exe</td>
<td>376</td>
<td>4</td>
<td>3</td>
<td>19</td>
<td>------</td>
<td></td>
<td>0 2010-10-29 17:08:54 UTC+0000</td>
</tr>
<tr>
<td>0x821a2da0</td>
<td>csrss.exe</td>
<td>600</td>
<td>376</td>
<td>11</td>
<td>395</td>
<td>0</td>
<td></td>
<td>0 2010-10-29 17:08:54 UTC+0000</td>
</tr>
<tr>
<td>0x81da5650</td>
<td>winlogon.exe</td>
<td>624</td>
<td>376</td>
<td>19</td>
<td>570</td>
<td>0</td>
<td></td>
<td>0 2010-10-29 17:08:54 UTC+0000</td>
</tr>
<tr>
<td>0x82073020</td>
<td>services.exe</td>
<td>668</td>
<td>624</td>
<td>21</td>
<td>431</td>
<td>0</td>
<td></td>
<td>0 2010-10-29 17:08:54 UTC+0000</td>
</tr>
<tr>
<td>0x81e70020</td>
<td>lsass.exe</td>
<td>680</td>
<td>624</td>
<td>19</td>
<td>342</td>
<td>0</td>
<td></td>
<td>0 2010-10-29 17:08:54 UTC+0000</td>
</tr>
<tr>
<td>0x823315d8</td>
<td>vmacthlp.exe</td>
<td>844</td>
<td>668</td>
<td>1</td>
<td>25</td>
<td>0</td>
<td></td>
<td>0 2010-10-29 17:08:55 UTC+0000</td>
</tr>
<tr>
<td>0x81fc5da0</td>
<td>VMwareTray.exe</td>
<td>1912</td>
<td>1196</td>
<td>1</td>
<td>50</td>
<td>0</td>
<td></td>
<td>0 2010-10-29 17:11:50 UTC+0000</td>
</tr>
<tr>
<td>0x81e6b660</td>
<td>VMwareUser.exe</td>
<td>1356</td>
<td>1196</td>
<td>9</td>
<td>251</td>
<td>0</td>
<td></td>
<td>0 2010-10-29 17:11:50 UTC+0000</td>
</tr>
<tr>
<td>0x8210d478</td>
<td>jussched.exe</td>
<td>1712</td>
<td>1196</td>
<td>1</td>
<td>26</td>
<td>0</td>
<td></td>
<td>0 2010-10-29 17:11:50 UTC+0000</td>
</tr>
<tr>
<td>0x82279998</td>
<td>imapi.exe</td>
<td>756</td>
<td>668</td>
<td>4</td>
<td>116</td>
<td>0</td>
<td></td>
<td>0 2010-10-29 17:11:54 UTC+0000</td>
</tr>
<tr>
<td>0x822b9a10</td>
<td>wuauclt.exe</td>
<td>976</td>
<td>1032</td>
<td>3</td>
<td>133</td>
<td>0</td>
<td></td>
<td>0 2010-10-29 17:12:03 UTC+0000</td>
</tr>
<tr>
<td>0x81c543a0</td>
<td>Procmon.exe</td>
<td>660</td>
<td>1196</td>
<td>13</td>
<td>189</td>
<td>0</td>
<td></td>
<td>0 2011-06-03 04:25:56 UTC+0000</td>
</tr>
<tr>
<td>0x81fa5390</td>
<td>wmicrvse.exe</td>
<td>1872</td>
<td>856</td>
<td>5</td>
<td>134</td>
<td>0</td>
<td></td>
<td>0 2011-06-03 04:25:58 UTC+0000</td>
</tr>
<tr>
<td>0x81c498c8</td>
<td>lsass.exe</td>
<td>868</td>
<td>668</td>
<td>2</td>
<td>23</td>
<td>0</td>
<td></td>
<td>0 2011-06-03 04:26:55 UTC+0000</td>
</tr>
<tr>
<td>0x81c47c00</td>
<td>lsass.exe</td>
<td>1928</td>
<td>668</td>
<td>4</td>
<td>65</td>
<td>0</td>
<td></td>
<td>0 2011-06-03 04:26:55 UTC+0000</td>
</tr>
<tr>
<td>0x81c0cda0</td>
<td>cmd.exe</td>
<td>968</td>
<td>1664</td>
<td>0</td>
<td>------</td>
<td>0</td>
<td></td>
<td>0 2011-06-03 04:31:35 UTC+0000</td>
</tr>
</tbody>
</table>
LSASS a.k.a. Local Security Authentication Subsystem Service

- Responsible for **authenticating users**
- **Only one per system** 🔥
- Associated with **Local System Account** ✅
- **Parent process:** winlogon.exe 🔥
- Executable in `%SystemRoot%\System32\lsass.exe`
- **Starts within seconds of boot times** 🔥
Detecting Injected DLLs

Injecting a DLL inside another process is a very common way for malware to hide their presence by not showing up in the process list.

- Examine the VAD for areas associated to DLLs
  - Even more suspicious if the page permissions are RWE
  - The malfind plugin is automatically searching for these cases
- Use ldrmodules to detect unlinked DLLs that are not listed
### The missing DLL...

<table>
<thead>
<tr>
<th>Pid</th>
<th>Process</th>
<th>Base</th>
<th>InLoad</th>
<th>InInit</th>
<th>InMem</th>
<th>MappedPath</th>
</tr>
</thead>
<tbody>
<tr>
<td>1928</td>
<td>lsass.exe</td>
<td>0x00080000 False False False False</td>
<td>\WINDOWS\system32\ntdll.dll</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x7c900000 True True True True</td>
<td>\WINDOWS\system32\ntdll.dll</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>\WINDOWS\system32\ntdll.dll</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x00870000 True True True True</td>
<td>\WINDOWS\system32\ntdll.dll</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>\WINDOWS\system32\ntdll.dll</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x76f20000 True True True True</td>
<td>\WINDOWS\system32\ntdll.dll</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>\WINDOWS\system32\ntdll.dll</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
andrea@ubuntu-20:~/volatility$ ./vol.py -f ~/Downloads/\*vmem --profile=WinXPSP3x86 clipboard
Volatility Foundation Volatility Framework 2.6.1

<table>
<thead>
<tr>
<th>Session</th>
<th>WindowStation</th>
<th>Format</th>
<th>Handle</th>
<th>Object</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>WinSta0</td>
<td>CF_UNICODETEXT</td>
<td>0x136012b</td>
<td>0xe29b0c68</td>
<td>74dcd49a7c121a61b8d05c03f92d0c13.exe</td>
</tr>
<tr>
<td>0</td>
<td>WinSta0</td>
<td>CF_LOCALE</td>
<td>0x270101</td>
<td>0xe1bdab58</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>WinSta0</td>
<td>CF_TEXT</td>
<td>0x1</td>
<td>----------</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>WinSta0</td>
<td>CF_OEMTEXT</td>
<td>0x1</td>
<td>----------</td>
<td></td>
</tr>
</tbody>
</table>
```
### Hidden services/kernel drivers...

```bash
andrea@ubuntu-20:/volatile~$ ./vol.py -f ~/Downloads/*******_vmem --profile=WinXPSP3x86 servicediff
Volatility Foundation Volatility Framework 2.6.1

<table>
<thead>
<tr>
<th>Missing service</th>
<th>Description</th>
<th>DisplayName</th>
<th>ErrorControl</th>
<th>Group</th>
<th>ImagePath</th>
<th>Start</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>MRxNet</td>
<td>(REG_SZ) MRXNET</td>
<td>(REG_SZ) MRXNET</td>
<td>(REG_DWORD) 0</td>
<td>(REG_SZ) Network</td>
<td>(REG_SZ) ??\C:\WINDOWS\system32\Drivers\mrxnet.sys</td>
<td>(REG_DWORD) 1</td>
<td>(REG_DWORD) 1</td>
</tr>
<tr>
<td>MRxCls</td>
<td>(REG_SZ) MRXCLS</td>
<td>(REG_SZ) MRXCLS</td>
<td>(REG_DWORD) 0</td>
<td>(REG_SZ) Network</td>
<td>(REG_SZ) ??\C:\WINDOWS\system32\Drivers\mrxcls.sys</td>
<td>(REG_DWORD) 1</td>
<td>(REG_DWORD) 1</td>
</tr>
</tbody>
</table>
```
And it was...

Full analysis here:
And now...

DEMO!

(on a toy Linux ransomware)
The Input/output memory management unit (IOMMU) introduces virtual memory for external devices.

If properly configured, it can be used to prevent certain devices to access some range of memory:

- This is typically the case when a hypervisor is running.
- In this case, it is very hard to get a physical image of the entire memory.

Malware can configure IOMMU to crash the system or returns false data when read from external devices:

- “Beyond The CPU: Defeating Hardware Based RAM Acquisition”
(Very) Short Introduction to Address Translation

- Memory analysis requires the ability to translate Virtual Addresses used by programs into the true memory locations in the memory image.

- Memory is divided into pages of 4KB each (in Intel architecture).

- The OS presents to each program a large private virtual address space.

- Each time a program references a virtual address, the MMU translates that virtual address into a physical location and accesses the requested data.

- MMU uses data-structure managed by the kernel (page tables) to perform automatic address translations.