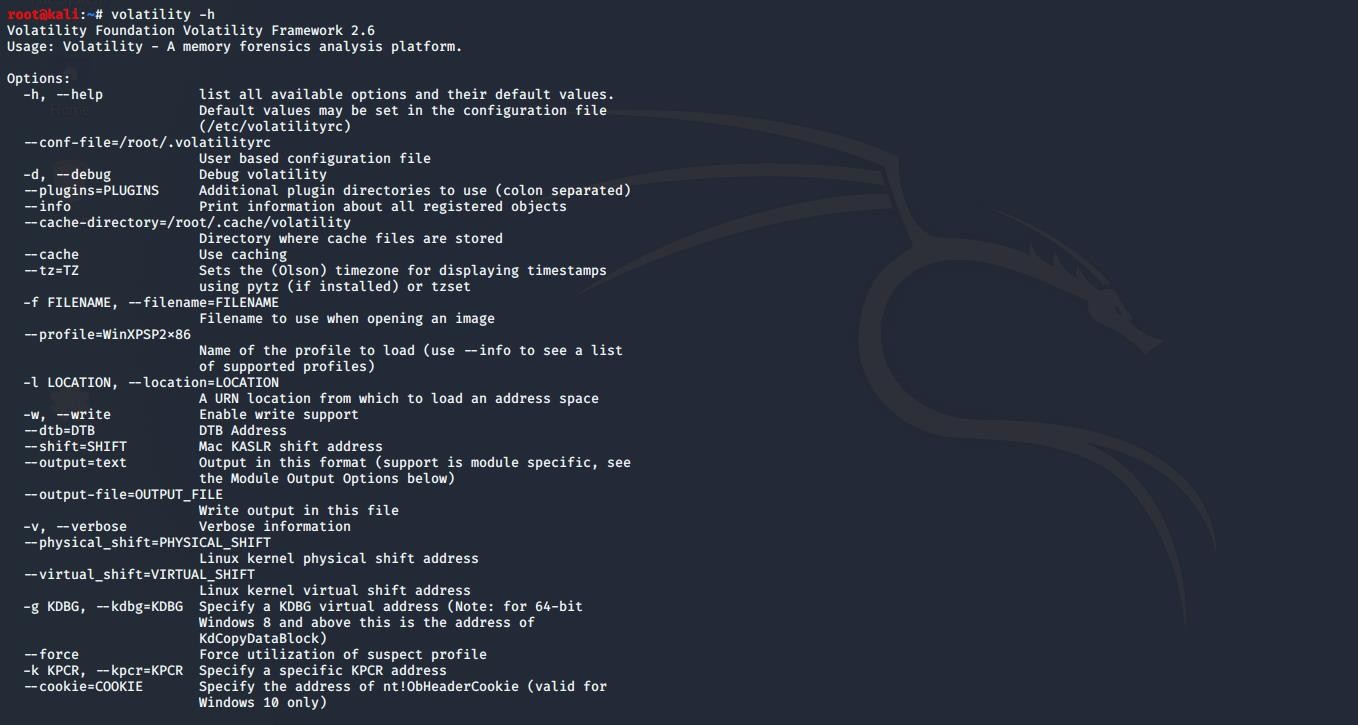
**Memory Forensics using Volatility**

Volatility is a free memory forensics tool developed and maintained by Volatility labs. Regarded as the gold standard for memory forensics in incident response, Volatility is wildly expandable via a plugins system and is an invaluable tool for any Blue Teamer. This tool is used to determine that either the PC is infected or not. The Volatility Framework is a completely open collection of tools, implemented in Python under the GNU General Public License, for the extraction of digital artifacts from volatile memory (RAM) samples. Volatility supports memory dumps from all major 32- and 64-bit Windows versions and service packs including XP, 2003 Server, Vista, Server 2008, Server 2008 R2, and Seven. Whether your memory dump is in raw format, a Microsoft crash dump, hibernation file, or virtual machine snapshot, Volatility is able to work with it. We also now support Linux memory dumps in raw or LiME format and include 35+ plugins for analysing 32- and 64-bit Linux kernels from 2.6.11 – 3.5.x and distributions such as Debian, Ubuntu, OpenSuSE, Fedora, CentOS, and Mandrake .

**Installation:**

**In Kali Linux it is installed by default.**

**Command:** volatility –h



Download the memory dump from the link provided and open **volatility**(memory forensics tool)

<https://github.com/volatilityfoundation/volatility/wiki/Memory-Samples>

**For this experiment we will use cridex memory sample**.

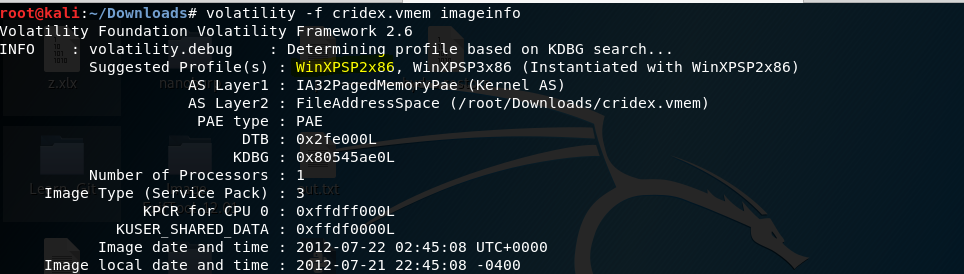


**Now we will analyse the use of various plugins in volatility**

* **Imageinfo** :This plugin is used to identify the operating system, service pack, and hardware architecture, it also contains other useful information such as the DTB address and time the sample was collected

 First, let’s figure out what profile we need to use. Profiles determine how Volatility treats our memory image since every version of Windows is a little bit different. Let’s see our options now with the command

`**volatility -f cridex.vmem imageinfo**`

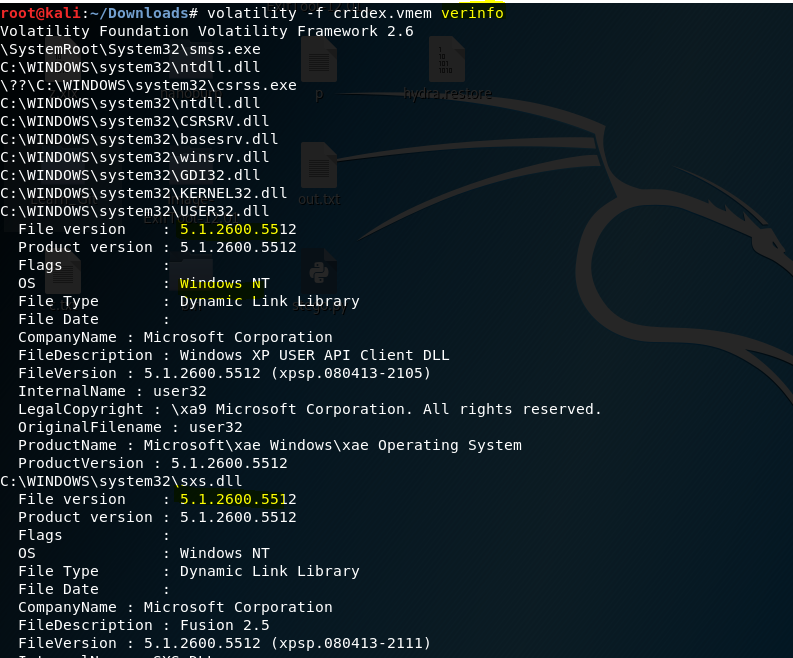


**Observation**

Running the imageinfo command in Volatility will provide us with a number of profiles we can test with, however, only one will be correct. We can test these profiles using the pslist command, validating our profile selection by the sheer number of returned results.

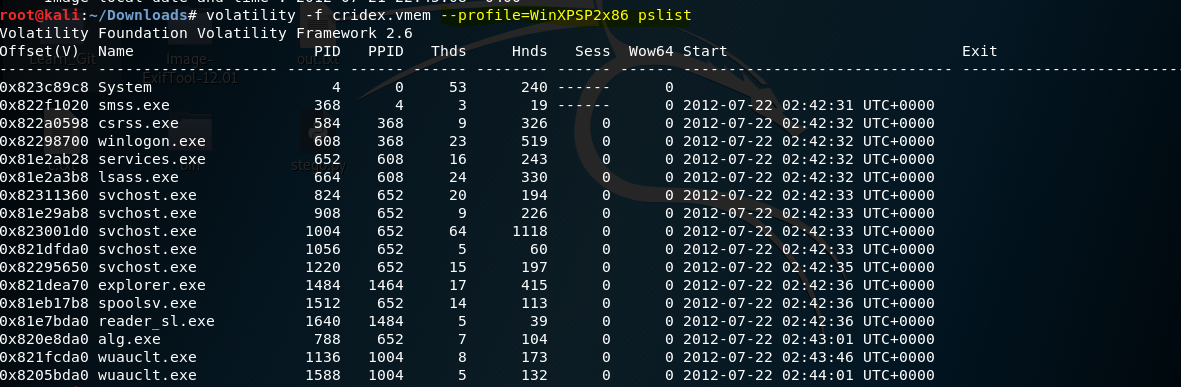
* **Verinfo**: To display the version information embedded in PE files, use the verinfo command. Not all PE files have version information, and many malware authors forge it to include false data, but nonetheless this command can be very helpful with identifying binaries and for making correlations with other files. Command:

**Volatility cridex.vmem verinfo**



* **Pslist** :This plugin is used to list the processes of a system. It shows the offset, process name, process ID, the parent process ID, number of threads, number of handles, and date/time when the process started and exited. The command will display all the processes that were running at the time of creation . **Command:**

`**volatility -f MEMORY\_FILE.vmem— profile=PROFILE pslist**`.

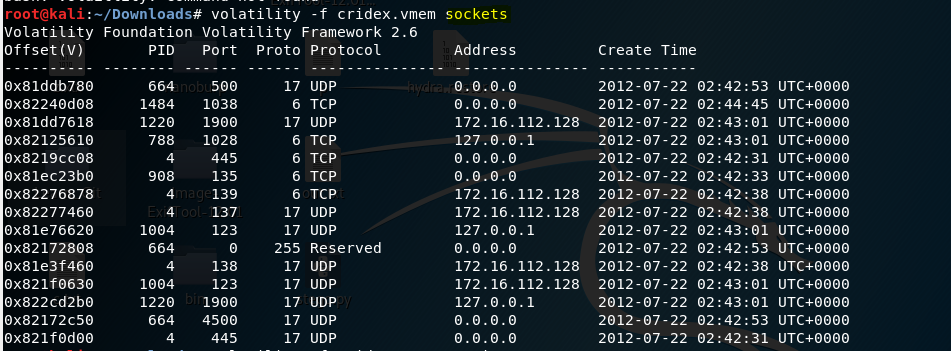
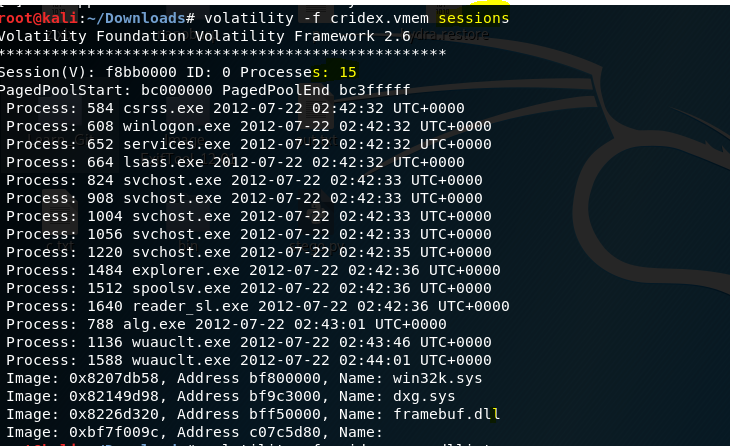


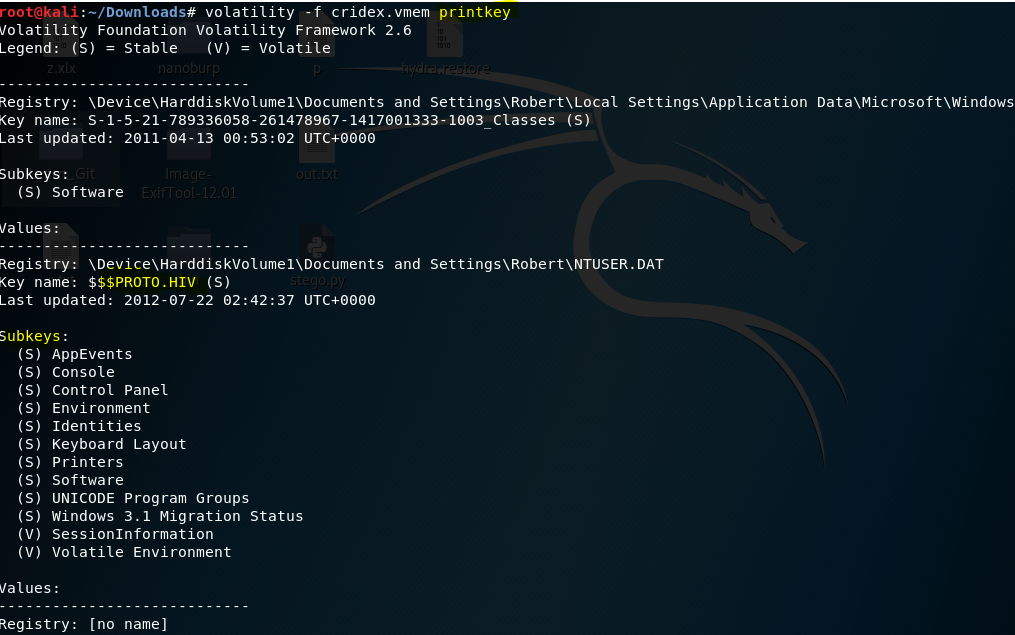
**Observations**: The profile that is correct for this memory image is WinXPSP2x86. All the child of the svchost should have same ppid of the service id else it may be malicious.

* **Psxview:** This plugin compares the active processes indicated within psActiveProcessHead with any other possible sources within the memory image. It’s fairly common for malware to attempt to hide itself and the process associated with it. That being said, we can view intentionally hidden processes via the command `**psxview**`.

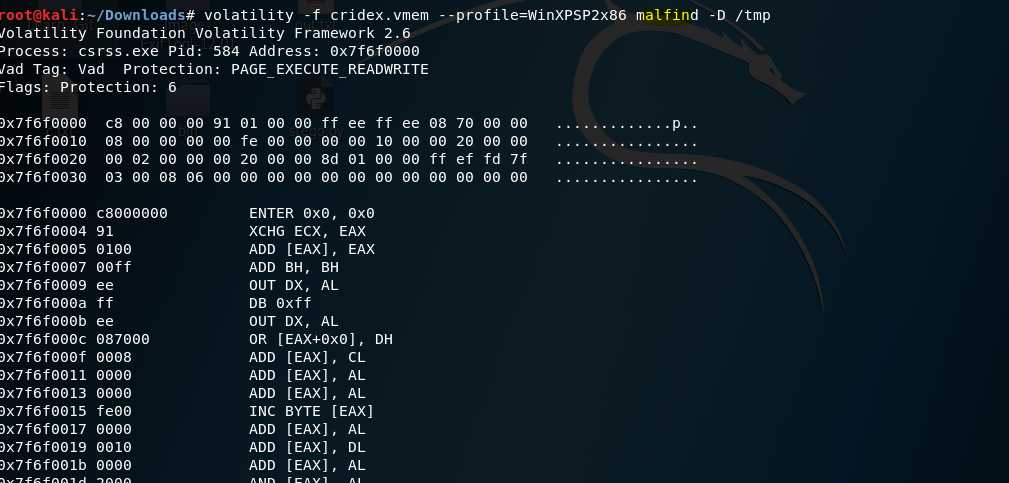


**Observation :** The process has only one ‘False’ listed is the **csrss.exe** and can be the malware to attempt to hide itself.

* **Socket** :This plugin is used to detect listening sockets for any protocol.
* **Session:**This plugin analyses theunique \_MM\_SESSION\_SPACE objects and prints details related to the processes running in each logon session, mapped drivers.   
  **Command:**  **volatility-f cridex.vmem sessions**
* **Printkey**:To display the subkeys, values, data, and data types contained within a specified registry key, use the printkey command. By default, printkey will search all hives and print the key information (if found) for the requested key. Therefore, if the key is located in more than one hive, the information for the key will be printed for each hive that contains it.



* **Malfind**: This plugin that we will use is malfind, which is a plugin that searches for malicious executables (usually DLLs) and shellcode inside of each process. Injected code can be a huge issue and is highly indicative of very very bad things. We can check for this with the command `malfind`. Using the command `**volatility -f MEMORY\_FILE.raw — profile=PROFILE malfind -D <Destination Directory>**` we can not only find this code, but also dump it to our specified directory



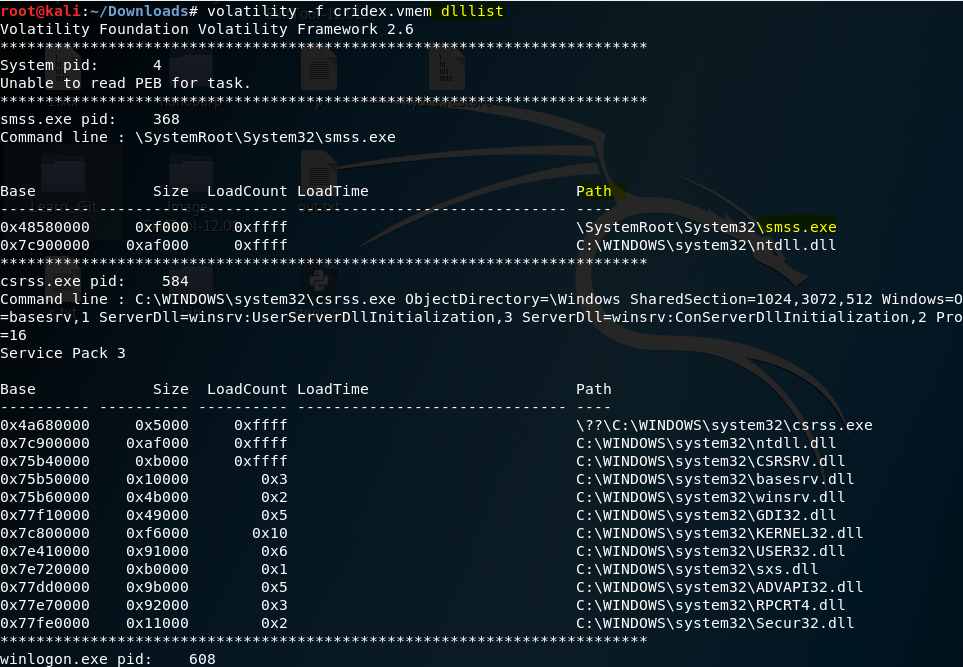


Observation: There are 12 proccess.dmp

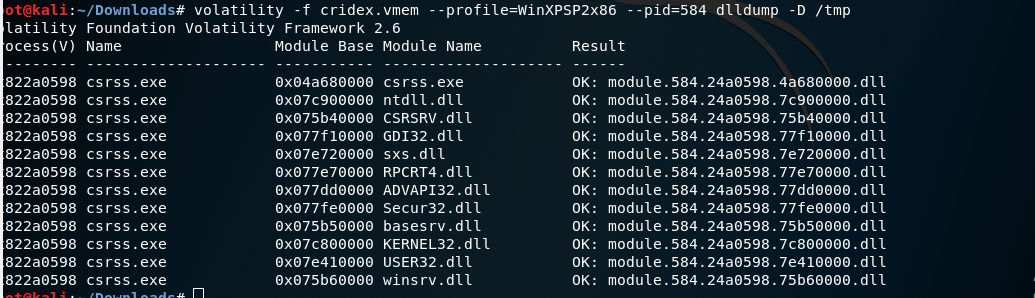
* **Dlllist:** The plugin dlllist in the Volatility Framework can also be used to list all DLLs for a given process in memory and find DLLs injected with the CreateRemoteThread and LoadLibrary technique. This technique does not hide the DLL and therefore will not be detected by the plugin malfind

**Volatility cridex.vmem imageinfo**

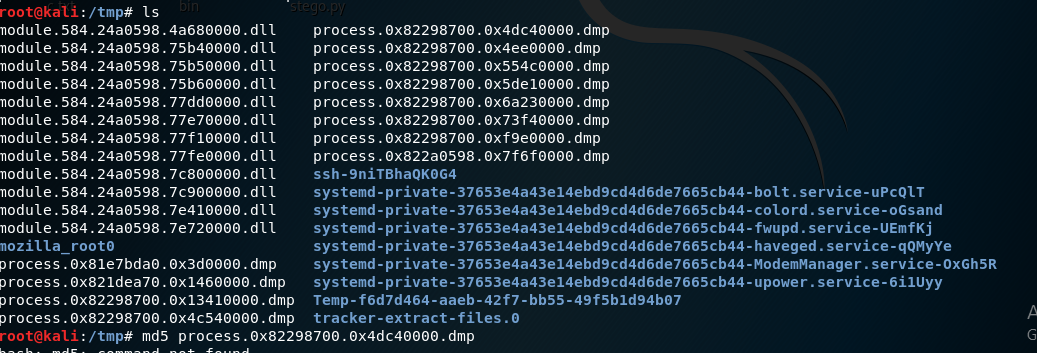
 DLLs are shared system libraries utilized in system processes. These are commonly subjected to hijacking and other side-loading attacks, making them a key target for forensics. Let’s list all of the DLLs in memory now with the command `dlllist`



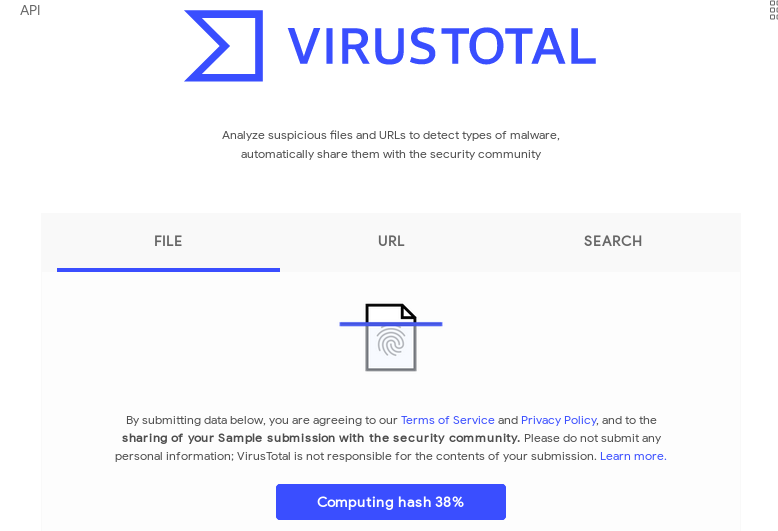
* **dlldump**: To extract a DLL from a process's memory space and dump it to disk for analysis, use the dlldump command. Now that we’ve seen all of the DLLs running in memory, let’s go a step further and pull them out! Do this now with the command `**volatility -f MEMORY\_FILE.raw — profile=PROFILE — pid=PID dlldump -D <Destination Directory>`** where the PID is the process ID

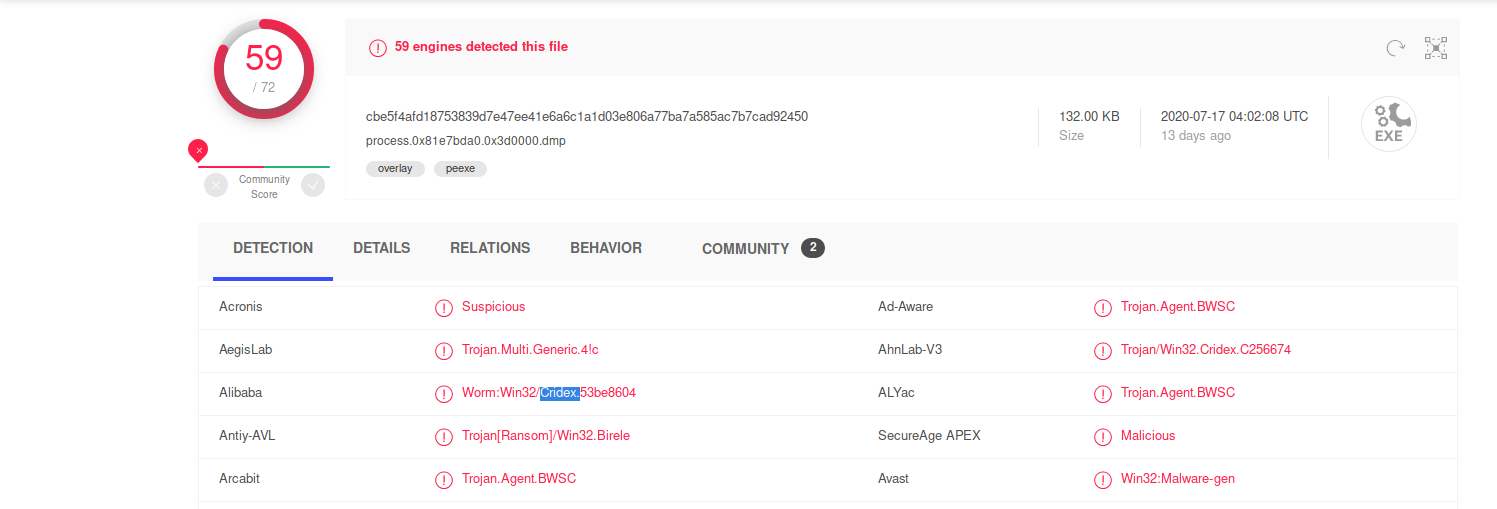


**Observation:** There are 12 DLLs does this end up pulling for the choosen PID

****

**Upload the extracted files to VirusTotal for examination.**





**Observation**: What malware has our sample been infected with? You can find this in the results of VirusTotal - **Cridex**