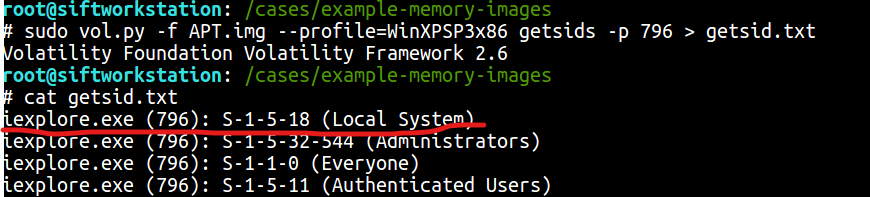
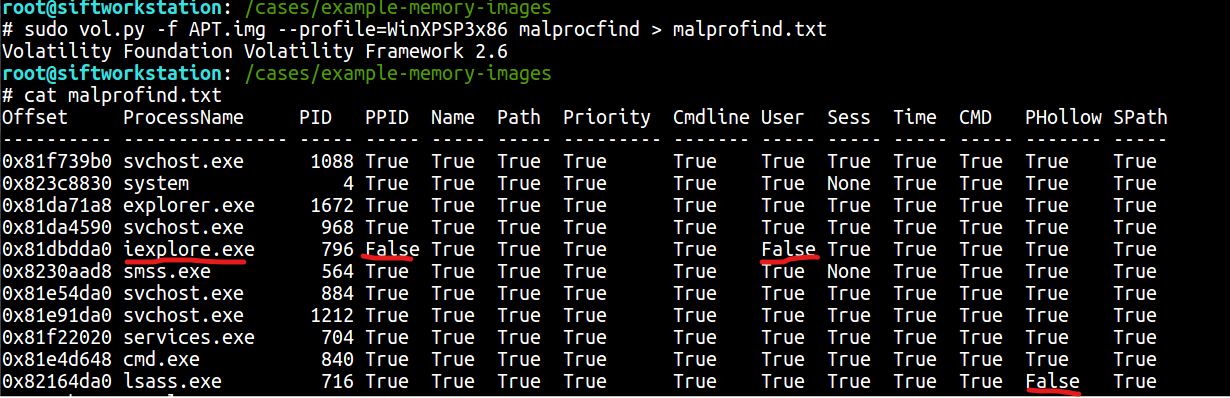
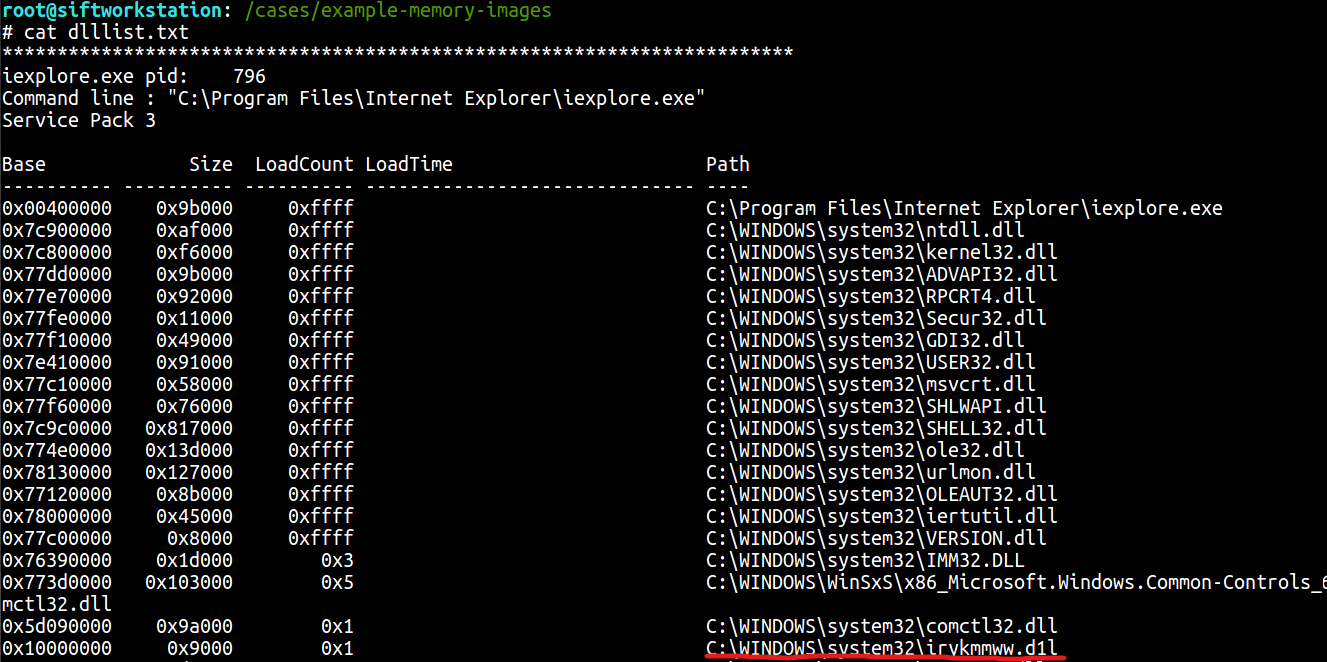


We can notice that iexplorer.exe was spawned from svchost.exe, which should spawn from explorer.exe

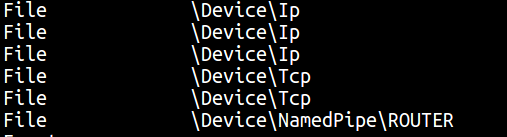


We can also see the process was run under local system, which should have been ran under a user account

  
When running malprocfind, we notice that the plugin had the same suspicions (wrong PPID and user). We also see the common F+ of lsass.exe proving fale in the PHollow column.

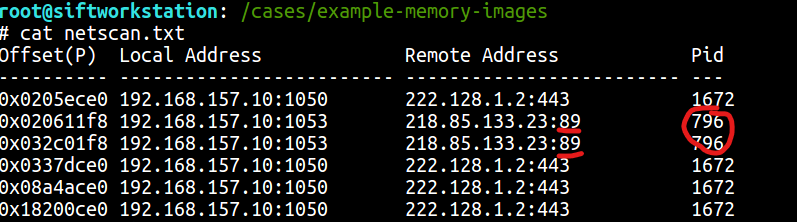
When reviewing the loaded DLLs as well as the file path and parameters, we see a suspicious dll with a random name and a .d1l extension



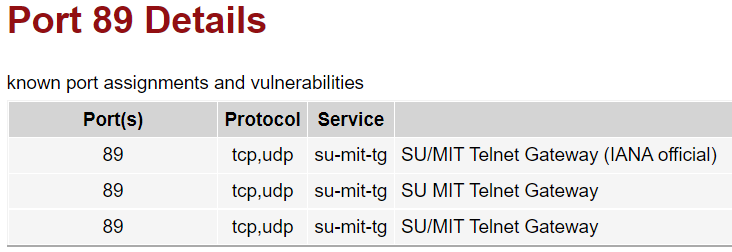




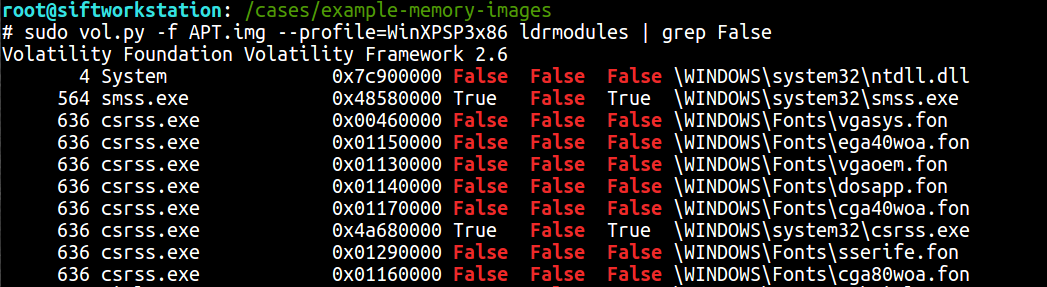
When viewing the handles, we can get some information on the binary - that it uses the TCP/IP stack as well as a mutant RasPbFile which google indicates that some malware uses this mutant (not a good indicator as there was no direct correlation to a specific strand of malware).



When viewing the connscan output, we can see that the process was outbound communicating with 218.85.133[.]23 over port 89. ONSIT reports this to be a Telnet gateway to China.







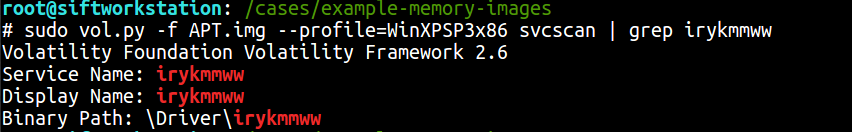
**Code injection**

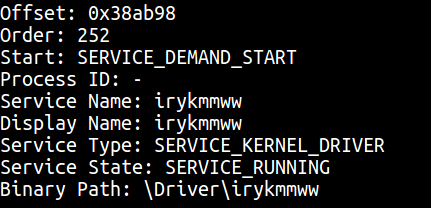
When running ldrmodules across all processes, we only see common occurring F+

The same result occurred when running malfind. This should be expected as this binary was mapped to the program files folder as seen in dlllist.   
  
The same result occurred when running hollowfind.

**How does the code execute?**

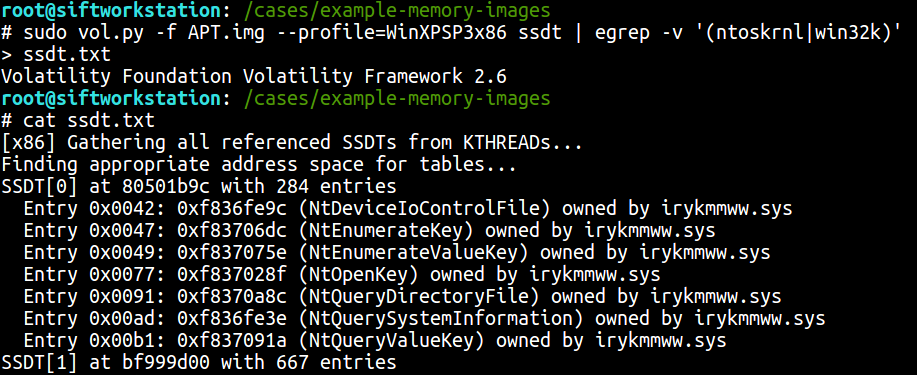
When running svcscan we can see that the driver is executed as a service



****

To do: dump hives and grep the hives for the driver (already did NTuser.dat)

**Rootkit detection**

****

We can see that the driver in question is hooking the system service descriptor table. It is hooking into the NtDeviceIoControlFile system function as well as gaining access to various registry keys/values as well as various directory and system information system functions

* NtDeviceIoControlFile - This API provides a consistent view of the input and output data to the system while still providing the application and the driver a device-dependent method of specifying a communications interface.