Codestock 2017 Script

- Objective: Find out what type of image it is.
  - Command:
    - `./volatility -f stuxnet.vmem imageinfo`

- Objective: Set path to file and profile in environment variables.
  - Command:
    - `export VOLATILITY_LOCATION=file://stuxnet.vmem`
    - `export VOLATILITY_PROFILE=WinXPSP3x86`

- Objective: See what processes were running on the machine at the time the image was taken.
  - Command:
    - `./volatility pstree`
  - Point out:
    - Multiple lsass.exe processes. Windows should only spawn one lsass.exe process, which does user authentication, from the winlogon.exe at boot time. However, here, two additional lsass.exe processes are started from the services.exe.
  - Command:
    - `./volatility pstree | egrep '(lsass.exe)'`
  - Point out:
    - Of the three lsass.exe processes, two of them share the services.exe parent ID, and one is winlogon.exe (normal). Additionally, the odd lsass.exe's start at the exact same time over a year after the first one starts, which is also weird.
    - This is the first indication that something is weird on this system, notably stuxnet.

- Objective: See what other weird things are going on with these two processes. Look at loaded libraries, DLL's. **DLL** is a dynamic link library **file** format used for holding multiple codes and procedures for Windows programs.
  - Command:
    - `./volatility dlllist -p 624 | wc -l`

    - Point out:
        - There are a lot of DLL’s under the normal lsass.exe, that’s good, because Windows uses DLL’s everywhere, and its usually common to see 40+ DLL’s under most software, system or otherwise (if benign)
- Command:
    - `./volatility dlllist -p 1928 | wc -l`
- Command:
    - `./volatility dlllist -p 868 | wc -l`
    - Point out:
        - This is bad, both of these have very few DLL’s even further reason to believe these are the malicious processes.

- Objective: See what other weird things are going on with these two processes. Look at internal sockets/network communication.
    - Command:
        - `./volatility sockets | egrep ‘(680|868|1928)’`
    - Point out:
        - The normal lsass.exe process (PID 680) has open UDP ports, which other windows processes use for sending messages to do authentication, BUT the suspicious lsass.exe’s don’t have any open sockets, which is all the more reason to believe they are bad.

- Objective: Let’s use a volatility command designed to look for weird malicious issues with processes in memory, specifically for things like writeable regions where it shouldn’t be possible.
    - Command:
        - `./volatility malfind -p 680`
    - Point out:
        - Malfind doesn’t find anything weird with the normal lsass.exe process.
    - Command:
        - `./volatility malfind -p 868,1928`
    - Point out:
        - Here, malfind shows us that there are several regions in memory where the memory page is executable **and** writable, which is not good. This means the malware has done something with the injected DLL to be able to write code into the memory space of other processes.

- Objective: Stuxnet calls the Windows LoadLibrary function with a specially crafted file name that does not exist on disk and normally causes LoadLibrary to fail. However, Stuxnet has hooked another windows DLL, Ntdll.dll, which is the windows API, to monitor for requests to load specially crafted file names. These specially crafted filenames are mapped to another location instead—a location specified by Stuxnet. That location is generally an area in memory where a .dll file has been decrypted and stored by the threat previously. The filenames used have the pattern of KERNEL32.DLL.ASLR.[HEXADECIMAL]. Knowing this, we can use the dlllist command to look for the string 'ASLR'.
  - Command:
    - `./volatility dlllist | grep ASLR`
  - Point out:
    - Even though the file does not exist on disk and the malware hooks the DLLs loading windows API's, the process environment block (PEB), an internal Windows OS structure that has system data stored when running, still shows it.

- Objective: So, we see this DLL in memory, and by running another command, ldrmodules, that will show which DLLs are missing from the PEB. We can find that stuxnet is injecting code via the ASLR DLL with this command.
  - Command:
    - `./volatility ldrmodules -p 1928`
- Point out:
  - The three lines in red are either suspicious because an entry is missing from one of the 3 PEB lists or because the path name is blank.
  - Based on that information, we can tell that code has been injected both by the last artifact we found and by seeing that these DLLs were shoved into memory without being linked into the PEB.

- Objective: Let's dump out these process's executables to disk and look at them.
  - Command:
    - `mkdir output`
    - `./volatility procdump -p 680,868,1928 -D output`
  - Command:

    - `strings out/executable.680.exe`
    - `strings out/executable.1928.exe`
  - Point Out:
    - Different!
    - Submit to VirusTotal

- Objective: Let's look at the registry now with Volatility. From analysis of stuxnet in other writeups, we know that the malicious lsass.exe drops two of the drivers used by Stuxnet into the registry entry for drivers so it can use them. It then creates corresponding registry keys for them. If we just search for these two drivers, we will find them in the registry!
  - Command:
    - `volatility  printkey -K 'ControlSet001\Services\MrxNet'`
  - Command:
    - `volatility printkey -K 'ControlSet001\Services\MrxCls'`

- Objective: What if we want to examine these drivers as loaded by the kernel?
- Command:
    - `./volatility modules`
  - Point out:
    - In this list are the two drivers we just looked up in the registry.

- Objective: Let's dump out the drivers for further examination. Look up the driver name in the list, look at what it's called, then let's examine it through strings and on VirusTotal.
- Commands:
    - `mkdir output`
    - `./volatility moddump -D output`
    - `strings output/driver.b21d8000.sys`
  - Point out:
    - This will dump out all the modules loaded in the OS, and we want the mrxnet.sys module. Upload to VirusTotal and look at results.