**CSEC 730 - Advanced Computer Forensics**

Homework 2

**Due Date:** Please submit your answers to the Homework 2 dropbox by Sunday midnight 10/28/2018.

**Goal**

The open-source toolkit, volatility framework, is one of the best memory forensic analysis tools to extract valuable information from a memory dump or a .vmem file. In this activity, you will practice volatility’s basic plugins for extracting important volatile data from memory images.

**Software**

Volatility is installed on SIFT Workstation.

[Volatility documentation, including a list of image types that Volatility can analyze](https://github.com/volatilityfoundation/volatility)

# Part 1. Windows Memory Analysis Using Volatility

In this activity, you will use volatility to analyze the Zeus memory dump from the Malware Analyst’s Cookbook DVD. Zens is a malware designed to steal credentials.

* Download [Malware Analyst’s Cookbook DVD download link](https://www.sendspace.com/pro/dl/p87m18) (.zip file)
* Extract the zip file and save it to your desktop. The Zeus memory sample is located in the folder named ‘**17**’ and in the sub folder ‘**1**’.

To challenge yourself, you may also use *Redline/Memoryze* from FireEye, <https://www.fireeye.com/services/freeware/redline.html>. However, using Redline is not required for this lab.

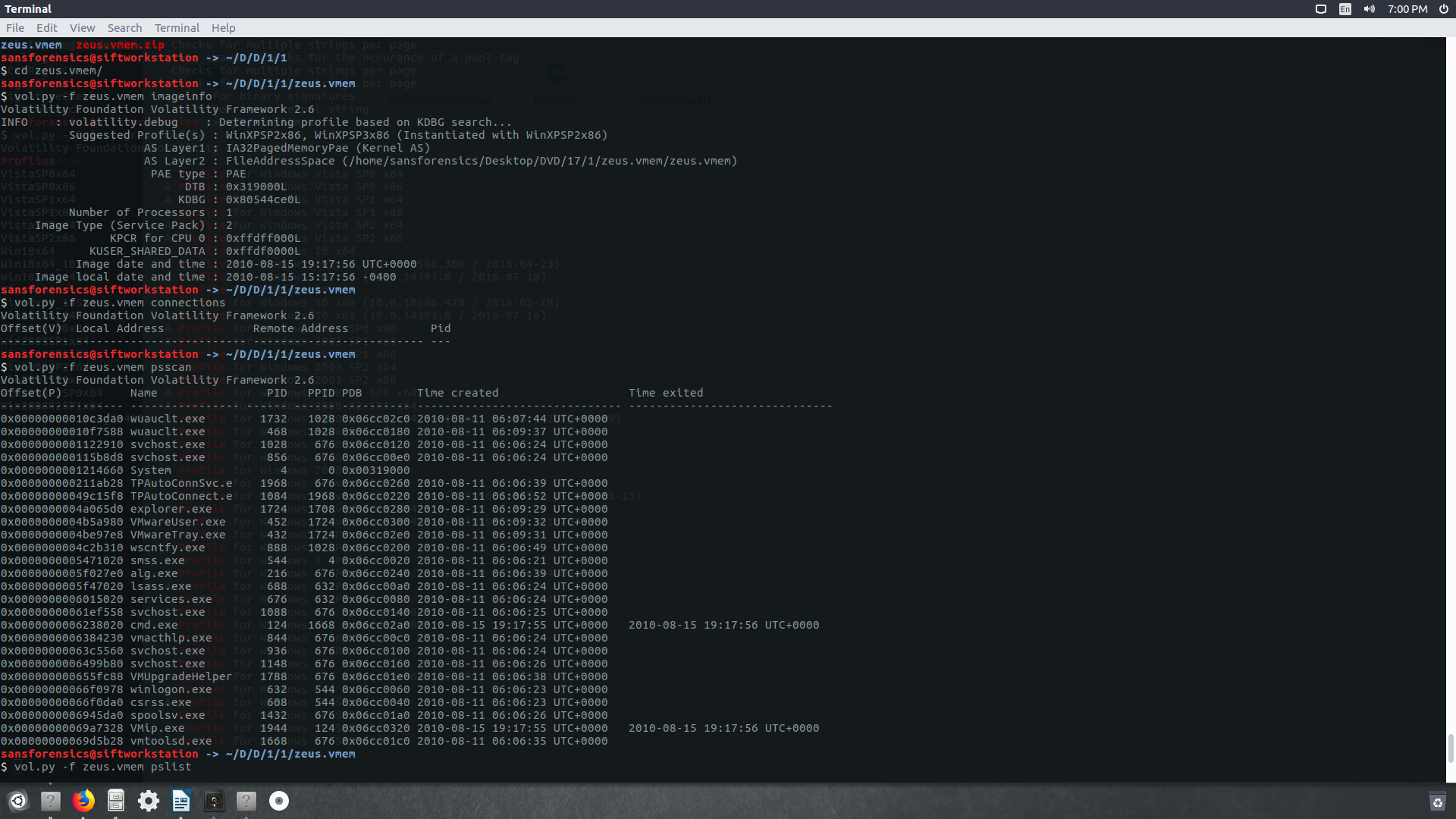
**Instructions**

1. Launch SIFT Workstation 3
2. Run vol.py –h to see volatility’s options and plugins
3. Practice these basic plugins to understand how you can use the result for your investigation.

|  |  |
| --- | --- |
| imageinfo | shows basic system information such as type of OS |
| pslist | Lists the processes of a system |
| psscan | Finds processes that previously terminated (inactive) and processes that have been hidden or unlinked by a rootkit |
| pstree | Displays the process listing in tree form |
| connections | Shows the TCP connections that were active at the time of the memory acquisition |
| connscan | Extracts TCP connections that were active at the time of the memory acquisition and previous connections that have since been terminated. |
| hivelist | Locates the virtual addresses of registry hives in memory and the full paths to the corresponding hive on disk |
| hivescan | Displays the physical addresses of registry hives in memory |
| printkey | Displays the subkeys, values, data, and data types contained within a specified registry key |

**Answer all the questions for part 1.**

### *Run vol.py -f zeus.vmem imageinfo* using volatility’s plugin "imageinfo." What is the suggested type of OS of zeus.vmem and when was the sample was collected? Provide screenshots as your supporting data.



From the screenshot, we can see that the suggested profiles are: WinXPSP2x86, WinXPSP3x86.

Image time: 2010-08-15 19:17:56

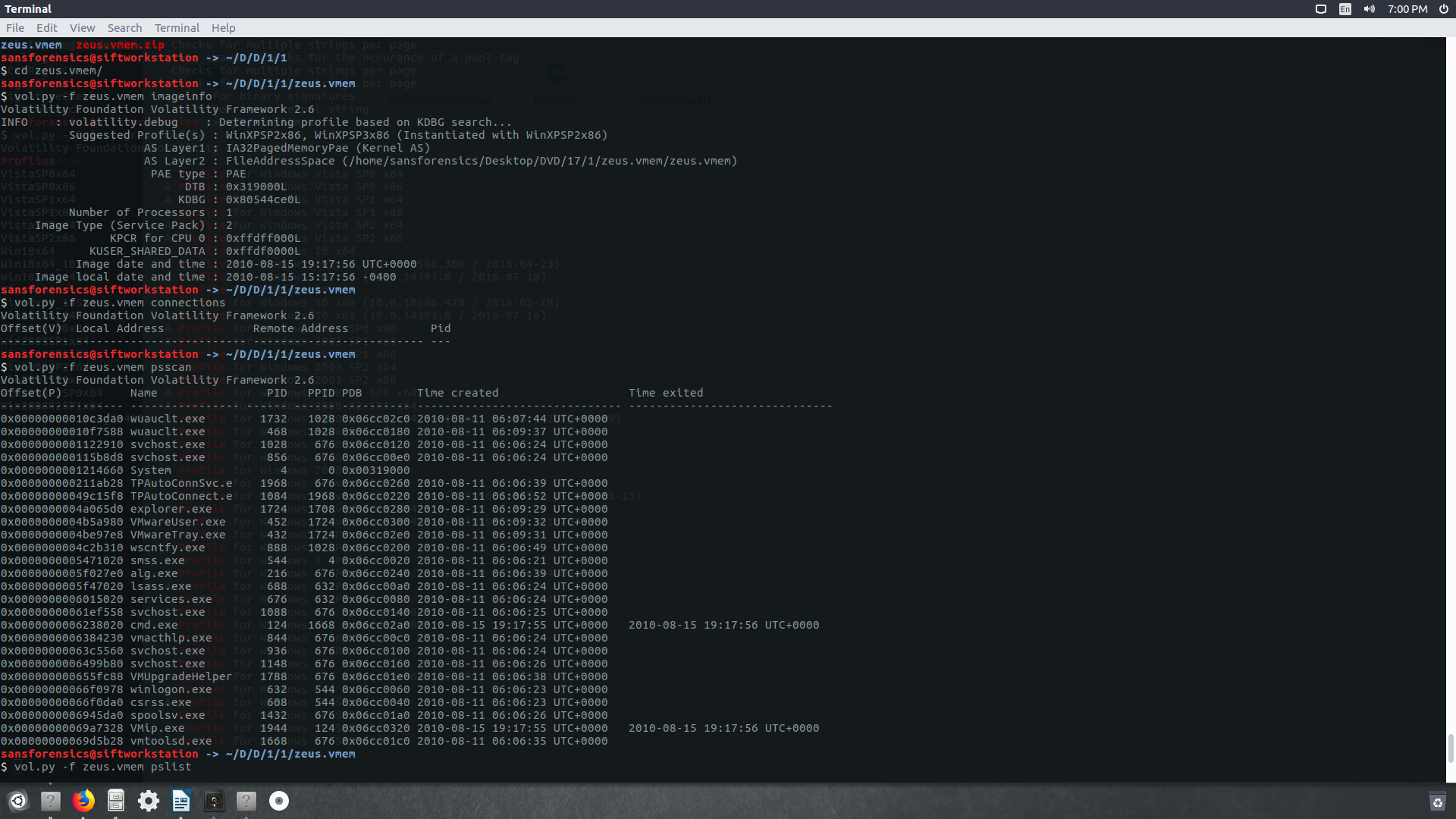
Image Local time: 2010-08-15 15:17:56

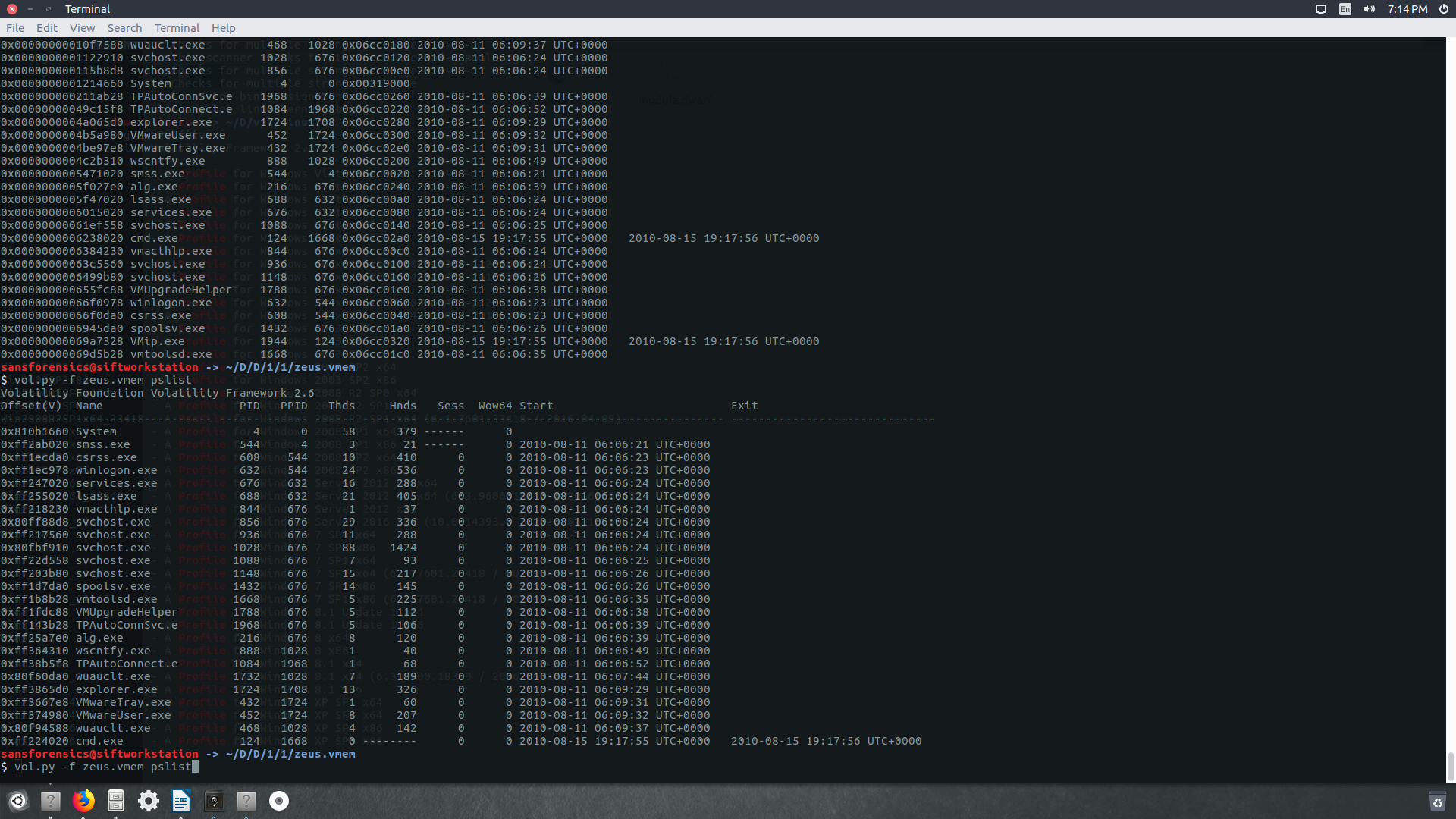
### Run *vol.py –f zeus.vmem pslist* and *vol.py –f zeus.vmem psscan*.

### Which one walks through the doubly-linked list of EPROCESS pointed by PsActiveProcessHead? pslist

### Which one does not rely on the doubly-list of EPROCESS and can detect unlinked (hidden) processes? psscan

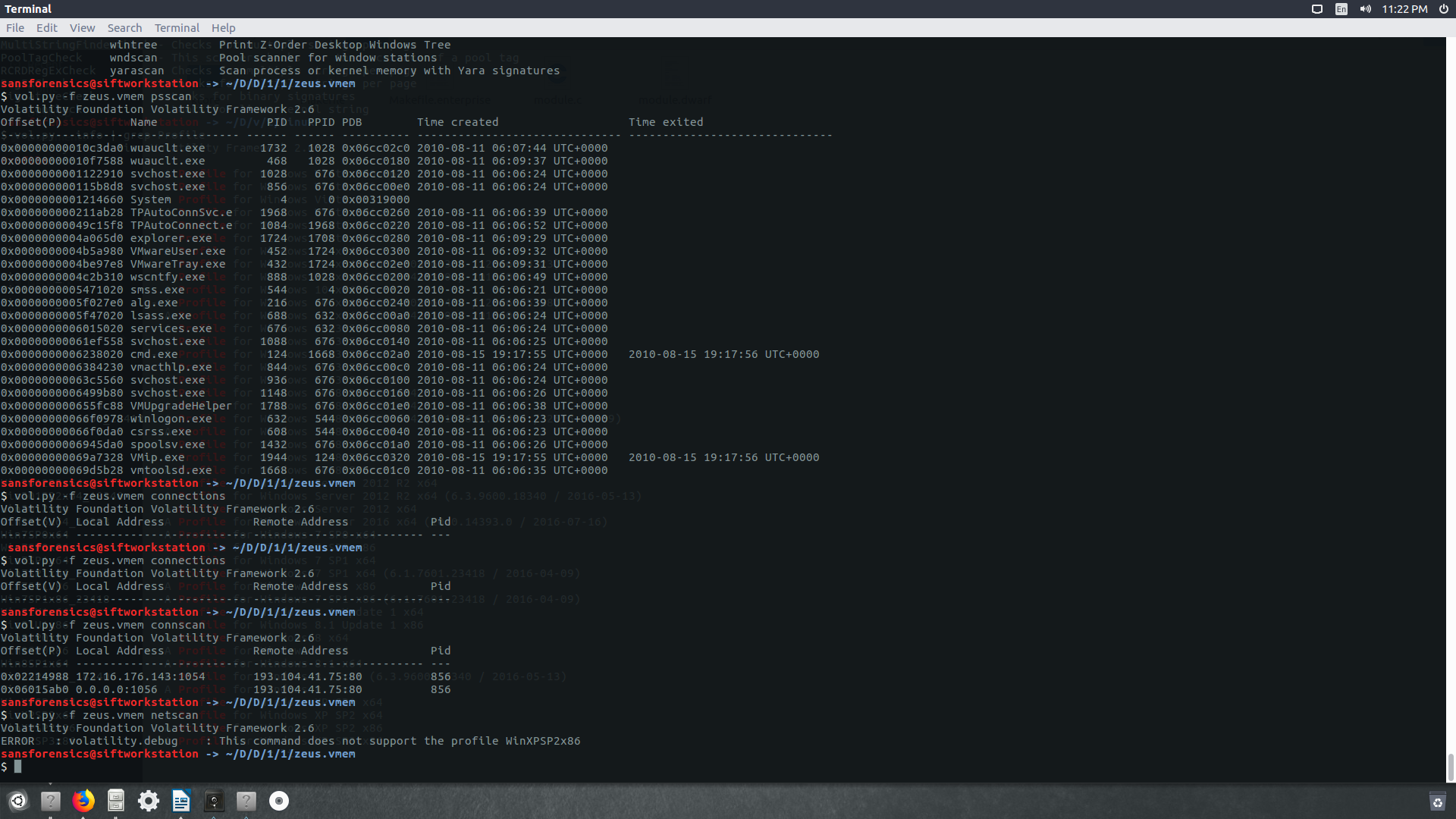
### Show the hidden processes in a screenshot.



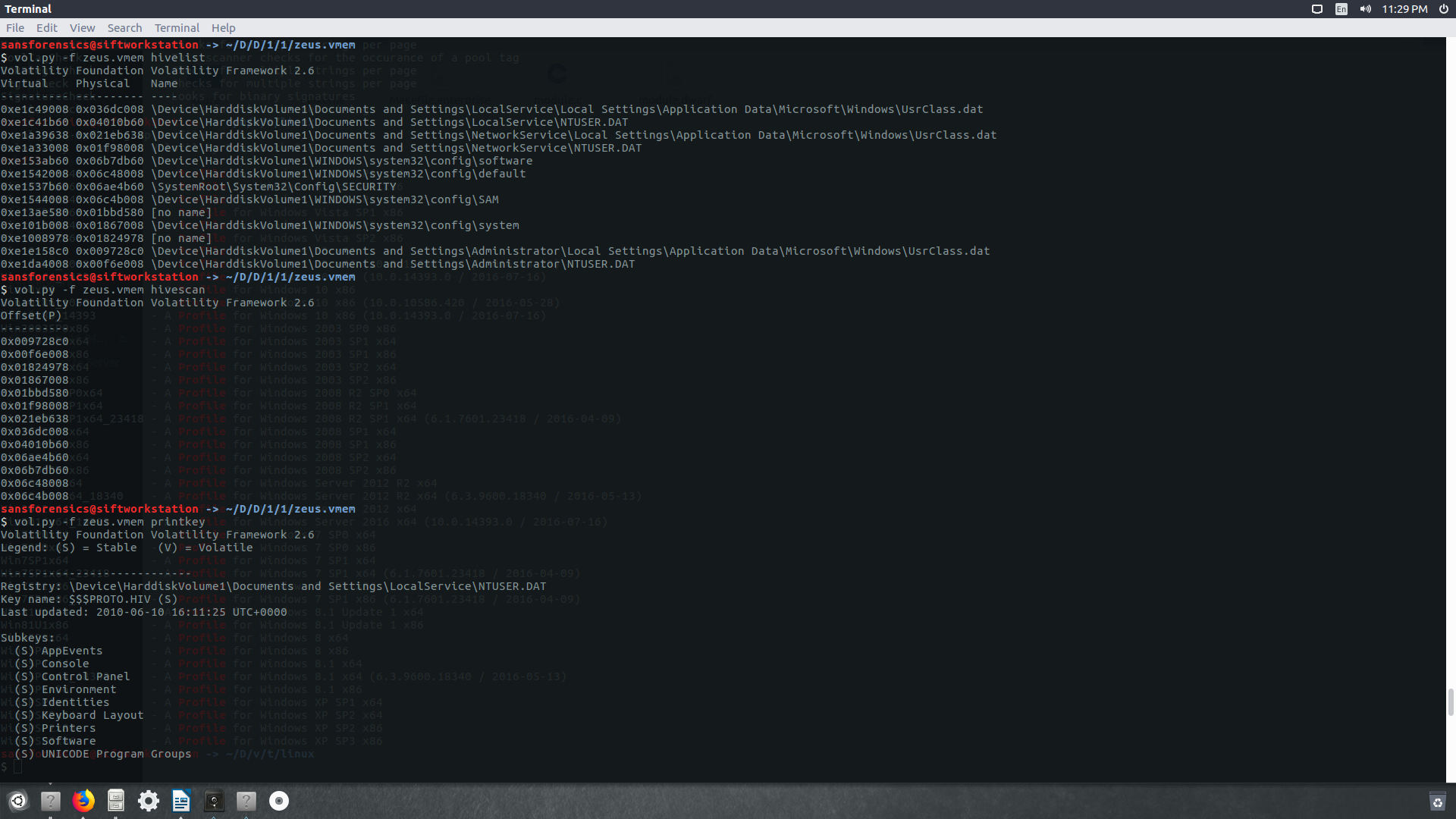


### Run *vol.py –f zeus.vmem connections* and *vol.py –f zeus.vmem connscan*. Do you see any active TCP connections or previous connections? Yes. Two of them.

### Provide screenshots as your supporting data. (Note: both *connections* and *connscan* do not work for Windows Vista and later version memory image. You will use plugin *netscan* instead)



### Run *vol.py –f zeus.vmem hivelist*, *vol.py –f zeus.vmem hivescan*, and *vol.py -f zeus.vmem printkey -K "Microsoft\Windows NT\CurrentVersion\Winlogon"*.



### 

### Which plugin displays the subkeys, values, and data types contained within a specified registry key? printkey

### Provide screenshots as your supporting data.

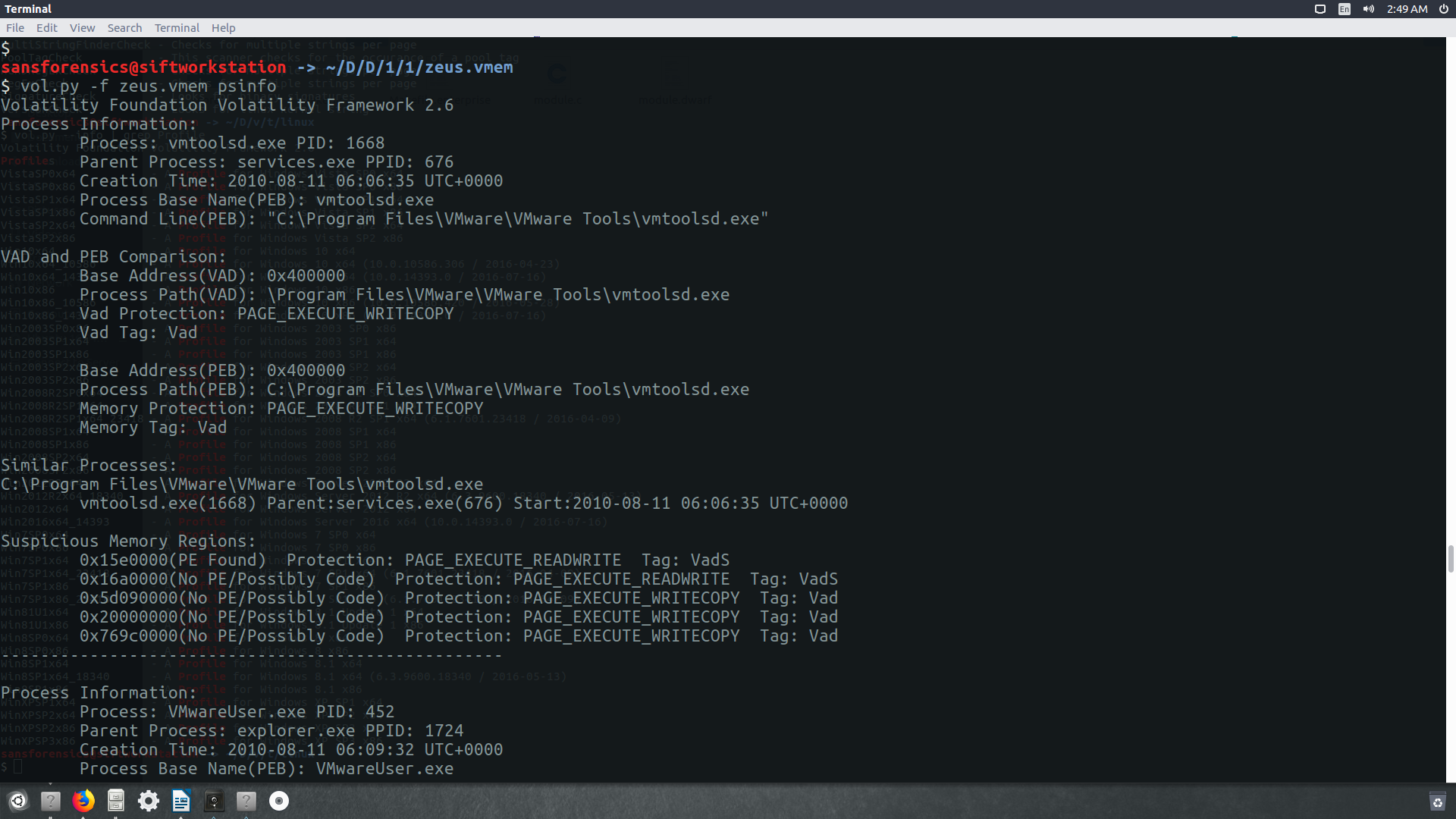
### Which plugin shows the virtual addresses of registry hives in memory along with the full paths to the corresponding hive on disk? Hivelist

### Provide screenshots as your supporting data.

1. Try other plugins from the [Volatility Command Reference](https://github.com/volatilityfoundation/volatility/wiki/Command-Reference), show me one or two other plugins that provide you interesting results. Psinfo

Display process related info and suspicious memory regions

Pstree: print process list as tree





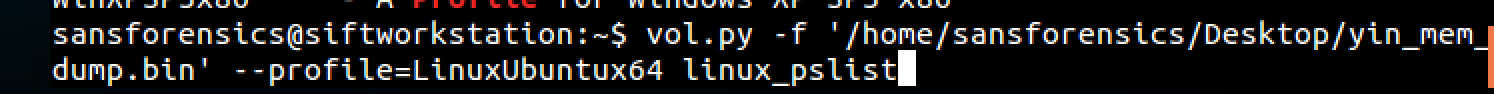
# Part 2. Linux memory acquisition using LiME (bonus points):

I tried to figure out a profile for my machine. Tried couple ones, but it didn’t work.

In homework 1, you dumped out your SIFT memory to the file, *yourusername\_memory\_dump.bin* (assume the file is saved on the SIFT desktop). In this exercise, you will use volatility to extract useful information.

**Instructions**

1. Download Ubuntux64 profile, *Ubuntu.zip*, from myCourses.
2. Move the *Ubuntu.zip* profile to the directory */usr/lib/python2.7/dist-packages/volatility/plugins/overlays/linux/*. This directory contains all the Linux profiles that volatility can use.
3. Run *vol.py --info | grep Profile* to make sure the profile "LinuxUbuntux64" is in the profiles list.
4. Run **vol.py -f ‘/home/sansforensics/Desktop/ *yourusername\_memory\_dump.bin’* --profile=LinuxUbuntux64 *VolatilityLinuxCommand***(Note: replace *VolatilityLinuxCommand* with the volatility Linux commands from <https://github.com/volatilityfoundation/volatility/wiki/Linux-Command-Reference>)

****

**Task for part 2:**

Show me the volatility commands along with the plugins and the data you recovered from the memory.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**If the profile "LinuxUbuntux64" does not work for your memory dump**, and you are not able to find an appropriate profile for your memory dump, then you have to build your own profile from the machine where the memory was exacted. In our case, it is the Ubuntux64 SIFT machine.

**The steps to build Ubuntux64 profile on SIFT:**

1. Install dwarfdump package and kernel headers

$sudo apt-get install dwarfdump linux-headers-generic

2. download volatility repo from <https://github.com/volatilityfoundation/volatility>

cd into tools > linux folder of the downloaded repo

$cd ~/Downloads/volatility/tools/linux

3. Generate the module.dwarf file using make

$make

4. Create the profile zip (sift.zip) and place it in volatility overlays/linux folder where volatility looks for all profiles, given the System.map file's for the kernel version.

$sudo zip /usr/lib/python2.7/dist-packages/volatility/plugins/overlays/linux/Ubuntu.zip module.dwarf /boot/System.map-3.13.0-44-generic

5. Run *vol.py --info | grep Profile* to make sure the profile "LinuxUbuntux64" is in the profiles list.

6. Run **vol.py -f ‘/home/sansforensics/Desktop/ *yourusername\_memory\_dump.bin’* --profile=LinuxUbuntux64 *VolatilityLinuxCommand***(Note: replace *VolatilityLinuxCommand* with the volatility Linux commands from <https://github.com/volatilityfoundation/volatility/wiki/Linux-Command-Reference>)

**Task for part 2:**

Show me the volatility commands along with the plugins and the data you recovered from the memory.