Muninn - The Volatility Reporter

About

Muninn was built to allow an easier approach to initial memory forensics on Windows 7 and Windows XP machines. Usually, when approaching a memory analysis we start by plotting out the basics and looking for the exceptions. This usually involves a lot of commandlining for each and every data set with Volatility. Muninn will take a case number and a memory image and will try to grab the basic pieces of data ***we*** usually look for and export them into a readable txt file which will be 'nicer' to read by a human being. It does not try to lead the memory forensics from a to z but rather to help the auditor through the initial plotting. To check for updates or submit changes follow this repository at the official repository This program is licensed under GPLv3.

Installation

Clone this repository using:

```
git clone https://www.github.com/ytisf/muninn
```

Make sure you have all the dependencies installed:

```
sudo pip install prettytable
```

Make sure Volatility is installed and linked to vol.py .

```
sudo apt-get install subversion pcregrep libpcre++-dev python-dev build-essentia libgmp3-dev
sudo apt-get install python-pycryptopp sqlite3 libsqlite3-dev
wget https://volatility.googlecode.com/files/volatility-2.3.tar.gz
tar xfv volatility-2.3.tar.gz
cd volatility-2.3/
sudo python setup.py install
```

How To

The basic command line arguments for Muninn are:

```
Options:
    -h, --help                          show this help message and exit
    -f FILENAME, --file=FILENAME        The path to memory image to analyse
    -c CASENUMBER, --case=CASENUMBER    Case number to use
```

The image location and case number are mandatory.

Muninn can be tested using the memory dumps which were published by the guys of Volatility here

Documentation

Basic structure of Muninn is:

- **imports**
- vol_handler.py
- error_handler.py
- report_manager.py
- muninn.py
- README.md

muninn.py

The main execution file. This file just calls other imports. This file manages the flow of the application and is a bit documented. Function names and calls are simple to understand. ###error_handler.py This manages errors in the program. It is very simple and not documented (since there is nothing to document). Every other python module in this application will call error_handler.py for output to the user (screen). ###report_manager.py Will be called to write the report file. It manages the functions:

- **__init__** -
- **InitiateDocument** - Will create the first block of the document and create the file_handler.
- **print_title** - Will add a header to the file.
- **print_table** - Will add a table to the report (since we have many).
- **save** - This will save the document properly and close the file_handler.

vol_handler.py

Warning! Black magic regexing here! You've been warned!

- **__init__** - This will initialize constructs. In general, all of the function will try to store the output in the main class as attributes to the class and not as a return option or anything like that.
- **regex_search** - Just what it says.
- **check_if_vol_is_installed** - Diddo.
- **get_image_type** - First time we use Volatility, and we use it to get image type.
- **document_image_details** - Generates basic image details such as MD5.
- **get_process_list** - Takes the process list from the memory image.
- **hive_list** - Gets all the hives. Used also at *find_hashes*
- **find_hashes** - Extract hashes (and users) from mem image.
- **get_network_connections** - Extract all UDP and TCP connections. (black craft magic van-dam regex vodoo here)
- **get_runkey_from_reg** - Gets the startup keys from the Registry.
- **drivers** - creates the self._drivers object and fills it we the drivers' list.

README.md

Just this readme file.

GPLv3