TP - **Forensic - Memory Analysis**

March 4, 2024

Environnement de travail

The working VM : IR-Workstation, `Login : investigator, password : L34rn1ng!`
SAMBA server is running on the guest VM to share files with the host OS. The shared directory inside the guest VM is : `~/data`
Launch the command `$> fast-update install_volatility investigator` to install Volatility environment if it is not already the case
Volatility documentation:
- Official Documentation - Volatiltiy
- Cheat Sheet Volatility
- The Art of Memory Forensics
Memory images extracted during the evidence collection step : IMAGES

indication:

Replace below the `vol.py` command by `python2.7 /bin/vol.py` command

Introduction

1. Run the `vol.py --info` command.
2. Run the `vol.py --help` command with and without a plugin name.

   indication:

   One of the options from the global help menu that you will use most frequently is `--profile`. This option tells Volatility what type of system your memory dump came from, so it knows which data structures, algorithms, and symbols to use.

3. Run the `imageinfo` plugin against a Windows memory sample `sample003.bin`.
   (a) What is the date and time when the memory sample was collected ?
   (b) What is the number of CPUs ?
   (c) What profile does it suggest?
4. Run the `kdbgscan` plugin against a Windows memory sample `sample003.bin`.
   (a) Do you confirme the suggested profile ?

indication:

The debugger data structure `_KDDEBUGGER_DATA64` is typically located inside the NT kernel module (`nt!KdDebuggerDataBlock`). It contains a build string such as `3790.srv03_sp2_rtm.0702161710`, numerical values that indicate the major and minor build numbers, and the service pack level for the target operating system. It is important for many things that Volatility and debuggers do. For example, it has a reference to the `PsActiveProcessHead` which is the list head of all processes required for process listing.

In some cases, you'll see slightly inaccurate profiles (for example Win7 SP1 versus Win7 SP0) because the OS data structures look similar. In these cases, look at the `Service Pack` value in the `kdbgscan` output to determine which suggestion is correct.

5. Run the `pslist`, `psscan`, `pstree` and `psxview` plugins against `sample003.bin`.
   - (a) which process(es) are active?
   - (b) which process(es) have terminated?
   - (c) which process(es) are leftover from a previous reboot?
   - (d) Which process(es) are hidden, by comparing the `pslist`, `psscan` and `psxview` views ?
   - (e) In what ways did the malware attempt to hide ? What type of malware uses this technique ?

   indication:

   See for example this before answering https://www.ired.team/miscellaneous-reversing-forensics/windows-kernel-internals/manipulating-activeprocesslinks-to-unlink-processes-in-userland.

   For more comfortable visualization, it is recommended to use `--output=dot --output-file=graph.dot` options of `pstree`

   ```
   $> dot graph.dot > graph.png
   ```

6. Run the `getsids` plugin against `sample005.bin` a lambda salary machine.
   - A) How many users are logged on?
   - B) What are their names?
   - C) Is there any evidence of privilege escalation attacks?

7. Run the `privs` plugin against `sample004.bin` and `memory.06454c20.img`.
   - A) In `sample004.bin`, which process(es) have the ability to load kernel drivers?
   - B) In `memory.06454c20.img`, is there anything strange with `explorer` process token ?

8. Mutex are good IoCs, which process is currently accessing the `"}!VoqA.I4"` mutex in `sample004.bin`?

   indication:

   Use `mutantscan` and `handles -t mutant` plugins.

Registry in Memory

1. Run the `hivelist` plugin on `sample004.bin`.
   (a) What's the virtual address of the `HKEY_LOCAL_MACHINE\SOFTWARE` hive?
   (b) What's the virtual address of the "administrator" user's `HKEY_CURRENT_USER` hive?
2. What is the function of the `Microsoft\Windows\CurrentVersion\Run` key. Use the `printkey` plugin with `-o` and `-K` options to check this key in both identified hives above. Do you see any entries that are worth further investigation?

Networking

1. Was RDP enabled on `sample004.bin` ?

   indication:

   You can use the `sockets` plugin.

2. How many active connections did `sample004.bin` have ? What websites (`port 80 or 443`) did it access in the recent past?

   indication:

   You will use `connections` and `connscan` plugins.

3. Use the `filescan` plugin and `dumpfiles` plugin with `-Q` and `-D` options to extract and analyze the `hosts` file from `sample007.bin` memory dumps.

Services

1. The `6to4` service is a legitimate component of the OS (helps migrate IPv4 to IPv6), but it's just often hijacked. Analyze `sample001.bin` with `svcscan` plugin and pay attention to the `6to4` service.
   (a) How do you know it's new?
   (b) What is the path to the suspicious DLL that implements the service?
   (c) Is the service running? If so, what is the host process ID?
   (d) Dump the malicious DLL from memory for static analysis by using `dlldump` plugin.
   (e) Open the extracted DLL with IDA PRO and quickly analyse it in order to find the IoCs.

   indication:

   You will use `svcscan` plugin with `--verbose` option and `dlldump` plugin.
   You will use `dlldump` plugin with `--pid` and `--dump-dir` options.

Malware Hunting

1. Analyze `sample007.bin` by using `malfind` plugin.
   (a) Google the notion of code and DLL injection.
   (b) Are any processes hosting injected code? If so, which one(s) ? Is it a PE file or shellcode ? Try to extract the malicous code and to analyze it quickly with IDA PRO to infer IoCs.

indication:

What is the signature of a PE file ?
The `procdump --memory` plugin allows to extract the image of a process.

Searching in process memory

1. Try to get the password hash of a user in the `memory.06454c20.img`.
2. By printing the process list with `pstree` on `win10-2.raw`, we notice that the `notepad` process is running. Try to recover what the user was writing.

   indication:

   Use the `vadtree` plugin with `--output=dot --output-file=graph.dot`. The heap chunks are colored in red.

   You can use the `vaddump` plugin to dump the heap chunks into files and analyze them with `strings -e l` or `xxd -a`
3. By printing the process list with `pstree` on `memory.06454c20.img`, we notice that the `mspaint` process is running. Try to recover what the user was drawing.

   indication:

   Use the `memdump` plugin with `--dump-dir`.
   Copy the resulted file as `$> .data`, open it with `gimp` and play with parameters to display the hidden image.
4. By printing the process list with `pstree` on `memory.06454c20.img`, we notice that an `Internet browser` process is running. By using the python code 1 in the `volshell` plugin with `--pid` option, try to find any passwod in the http messages.

Listing 1: searchpassword.py

```
process = proc() # pour r cup rer  l objet  processus courant
process_space = process.get_process_address_space() # pour r cup rer  l espace
     d adressage  utilis
criteria = []
criteria.append("&Email")
criteria.append("&Passwd")

for addr in process.search_process_memory(criteria):
    tz = obj.Object("String",offset = addr, vm = process_space,length = 64)
print str(tz)
```