June, 2020

Windows Memory Forensics I

Introduction to Memory Forensics. Unstructured Analysis.

LIFARS
your digital world, **secured**

Contents

OVERVIEW

Memory forensics has been a crucial part of an investigation process for some time now. RAM can provide – and provides – invaluable information on what is happening on the system at any given moment. It is also able to reveal traces of activity that has already taken place, leaving a trail of breadcrumbs behind – in memory. In this series we will introduce the process of memory analysis – from acquiring a sample of RAM to extracting juicy data and interpretation of contained information.

MEMORY ACQUISITION

MEMORY CAPTURE TOOLS

Before we can even think about memory analysis, we need to figure out a way to obtain an image of RAM. There are several tools available to us that allow for the RAM capture. An important thing to remember is that any tool we choose will leave a trace on the system we are investigating, which is why we never install any memory acquisition tool on the source system (if not absolutely necessary for some reason). Be sure to document your steps so that in the case of a lawsuit, acquired evidence may be accepted in the court of law and the entire investigation process would not be disputable.

FTK IMAGER

Access Data FTK Imager is a well-known forensic tool. Besides allowing an investigator to mount and image physical disk or logical partition, create a custom content image or export specified files from evidence, it allows us to capture memory from a running system. This tool must be run with privileges of the local Administrator. Using a portable version of FTK Imager running from a USB stick, we create the RAM image by selecting "Capture Memory" from the top-left dropdown menu. Do not copy your FTK Imager executable files onto the machine that is being investigated.

Including pagefile can be beneficial to our investigation as this file contains memory sections that have been "paged out" to the disk. Although it lacks structured data like (kernel) memory structures contained in RAM image, pagefile can provide valuable information about "raw" data present in the memory. For example, the IP address of a C2 server or malicious domain names found in pagefile.sys which can give us a clue about adversary activities happening on the system, even if it does not give us precise context.

The output format from FTK Imager is by default ".mem". If desired, we can choose to store memory in .ad1 format. The default format works well with most of the memory parsing and analysis tools, including the ones discussed in this series: Rekall and Volatility.

When selecting the location to write the output capture to, remember that we must not point it to the device we are imaging. Configure FTK Imager to store the resulting image on an attached external media, brought in by an examiner.

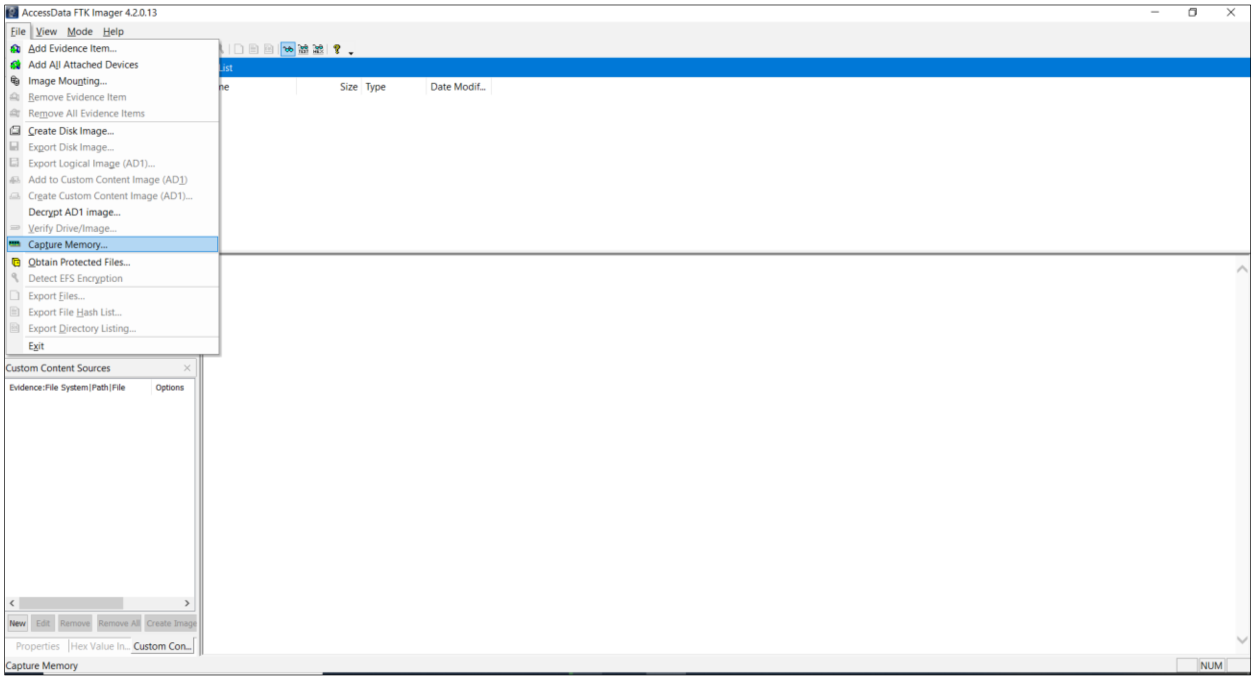
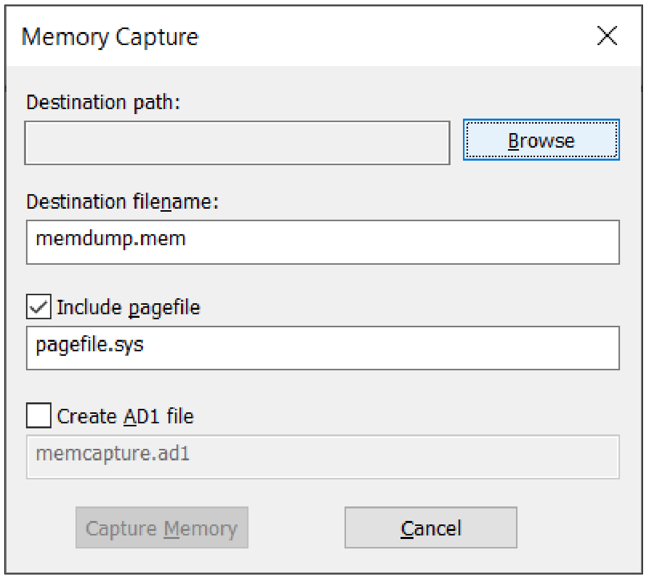


Figure 1: FTK Imager


Memory Capture

Destination path:

Browse

Destination filename:

memdump.mem

[✓] Include pagefile

pagefile.sys

[ ] Create AD1 file

memcapture.ad1

Capture Memory Cancel

Figure 2: FTK Imager - Memory Capture Dialog

WINPMEM

WinPMEM is another well-known tool used for obtaining memory images from Windows systems. It is an easy-to-use command line program which, by default, stores the resulting memory image in AFF4 format. AFF4 is an open-source standard which is designed to store multiple types of data – including disk and RAM images. For subsequent analysis we may need to extract raw memory from acquired AFF4 files. WinPMEM allows for such extraction, as well as Volatility's and Rekall's imagecopy plugins.

```
winpmem_v3.3.rc3.exe -o E:\memdump.aff4
winpmem_v3.3.rc3.exe E:\ memdump.aff4 -ExtractPhysicalMemory -o
E:\memdump.raw
```

Capture memory in raw format:

```
winpmem_v3.3.rc3.exe --format raw -o E:\memdump.raw
```

UNSTRUCTURED ANALYSIS

Runtime information about a live system is stored in the computer's memory in a structured way. Kernel structures have a defined format and store data about running processes, handles, DLLs, other information that is stored about drivers, console content, network packets etc. But what if we are not overly concerned in all the information the operating system needs to keep the system up and living? What if we focus on some clue itself, not on the context surrounding it? If that were to be the case, unstructured analysis then comes into place.

STRINGS SEARCH & GREP

Strings is a well-known tool pre-installed on most Linux/UNIX-based OS. Strings searches through the content of a disk/directory/file and prints character arrays that appear to be (human readable) strings. As basic as it is, strings is a powerful tool to search for malicious domain names, IP or email addresses, filenames, contacts etc.

By default, strings outputs sequences of 4+ characters in length, formatted as 8-bit ASCII chars. Using switches, we can override default settings and search for big *(-e b)* and little *(-e l)* endian Unicode strings with a minimal length of our choice *(-n XX)*. Another hack to remember is printing the offset within the file where the string was found. For decimal or hexadecimal offsets, use *-td* or *-tx*, respectively. Finally, the *-f* switch comes in handy in case we need to perform a string search on a folder or disk image: it will include a filename where the string was found into the output.

In most cases, strings output becomes quite lengthy. Consider redirecting the output into a file or piping it to the *less* command. When looking for a specific string or pattern, we can use GREP/EGREP for postprocessing. GREP scans a file and prints out any matching strings. Use *-i* option to make the search case-insensitive, or *-v* to show only lines that do not match the specified term. You should consult GREP's man page before using it to reveal the true power of this tool. EGREP, or Extended GREP, supports extended regular expressions search. Finally, AGREP a.k.a. Approximate GREP, allows for using various queries, arbitrary wildcards, regular expressions – virtually everything that can be used with GREP, but more general. (Note that tools are not fully compatible.) The processing of AGREP is usually faster, compared to the same query performed by GREP. Additionally, AGREP has several built-in searching algorithms. Also, for processing each query, it will select the best-suited to complete the task.

Of course, GREP can be used not only to scan STRINGS output, but to scan an entire process's memory or any file or directory. However, if the file's encoding is different from ASCII, GREP will need some tweaks to fully cooperate with you 😊.

BULK EXTRACTOR

To go beyond the basic strings search and filtering, we will need a more sophisticated instrument. Bulk Extractor is a powerful tool which scans through the file or image (not limited to memory dump, crash dump or hibernation file) without parsing the filesystem structure. The output consists of so-called feature files with data extracted. These features include MAC and IP addresses, domain names, phone numbers, e-mail addresses, websites visited and even network packets that were in memory at the time of acquisition. When multiple of these information types are collected, histograms become populated. Histograms can be of great use: for example, we can infer that the most frequently hit MAC addresses and the topmost used email account belong to the device being investigated and to the nearest network switch/access point.
Bulk Extractor does not run every scanner by default. Scanners can by disabled or enabled to address specific requirements of the ongoing investigation. For example, information scanners such as Outlook and Facebook need to be directly invoked.

One of BulkExtractor's useful features, although not run by default, is the wordlist scanner. It works similarly to STRINGS – it looks for character arrays and saves them to the output feature file. Carving features include jpeg, rar or zip extraction, as well as previously mentioned network packets. The latter allows an investigator to open pcap files in Wireshark and look at network communication – note that packets will not have timestamps included. This is quite logical – packets themselves do not carry any such information; timestamps are added by packet capturing tools when speaking about real-time network monitoring. If Bulk Extractor carves for packets in memory, no timestamp can be present or reasonably added.

Bulk Extractor comes with a graphical interface, BEViewer, to view results. After opening a XML file with the report, the investigator can go through any extracted data listed in the left pane. The middle region shows the precise location of the scanned file where the data were found. On the right side of the window, data are shown in context, in a hexadecimal view.
Not every piece of data is shown in BEViewer. For example, a pcap file is present in the output folder, but cannot be investigated in the GUI, with the same applying to MFT,

INDX or LogFile: these NTFS filesystem files (or their parts currently present in memory) are carved from memory and stored for further analysis, performed by the investigator.

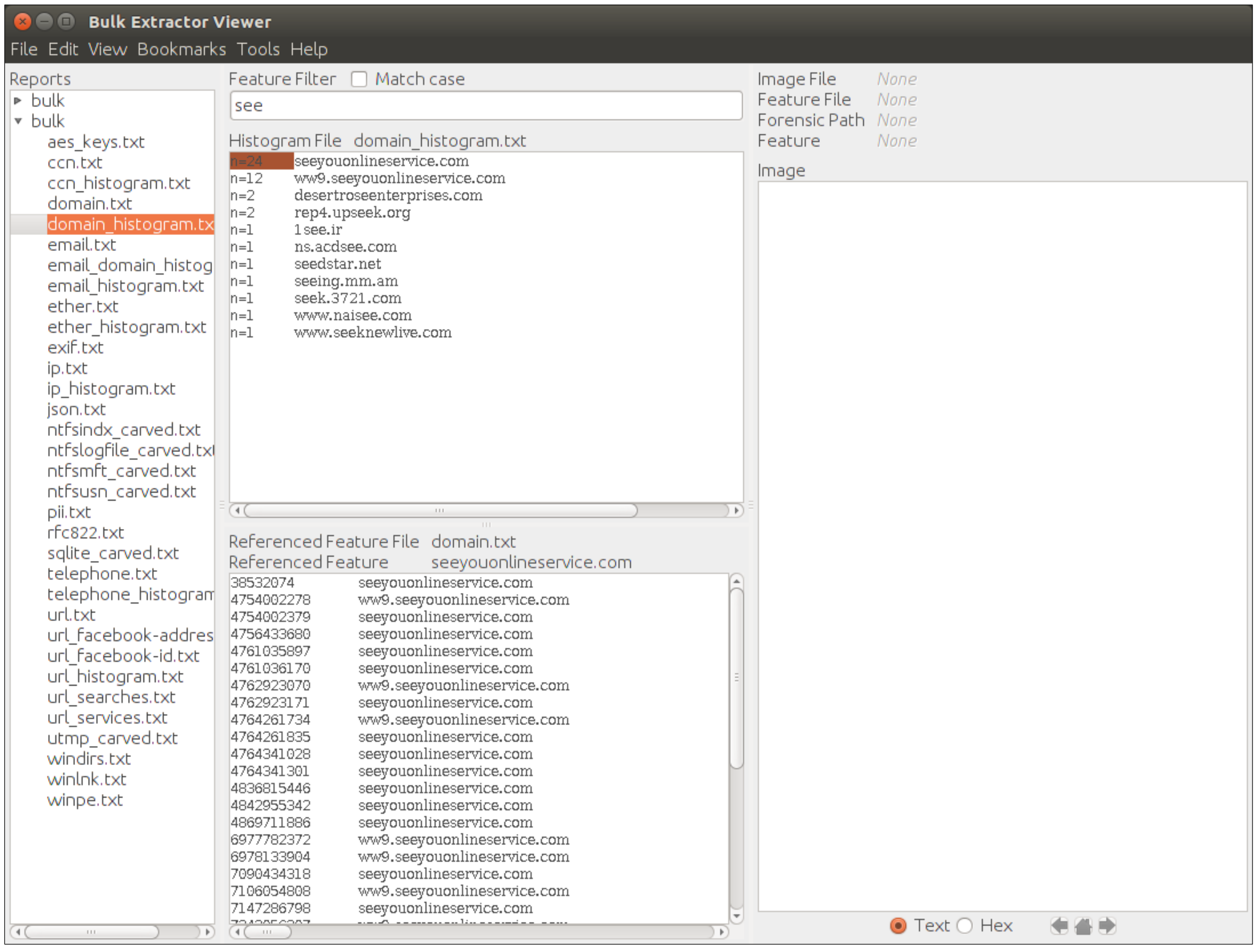
Before we proceed, let us speak a bit about an image that we are about to analyze. It was captured from a system infected by malware known as Khalesi, which has been a part of the Kpot campaign. It exhibits multiple standard malware behaviors, such as network communication with C2 server, creating persistence via scheduled tasks, Run registry key, multiple files dropped on the disk etc. However, as it employs several checks to ensure that the sample is not being debugged/analyzed on a Virtual machine, it eventually crashed, so not all actions were performed and thus cannot be deeply analyzed.
VT analysis:
https://www.virustotal.com/gui/file/4e87a0794bf73d06ac1ce4a37e33eb832ff4c89fb9e4266490c7cef9229d27a7/behavior/Dr.Web%20vxCube

Example of running BulkExtractor on memory image:

```
$ bulk_extractor -o whitepaper/pony/bulk /mnt/hgfs/AnalystVMShare/memdump-
pony-2.mem
bulk_extractor version: 1.6.0-dev-rec02
Hostname: siftworkstation
Input file: /mnt/hgfs/AnalystVMShare/memdump-pony-2.mem
Output directory: whitepaper/pony/bulk
Disk Size: 9126805504
Threads: 2
Attempt to open /mnt/hgfs/AnalystVMShare/memdump-pony-2.mem
 9:13:04 Offset 67MB (0.74%) Done in  0:13:25 at 09:26:29
 9:13:06 Offset 150MB (1.65%) Done in  0:07:37 at 09:20:43
 9:13:07 Offset 234MB (2.57%) Done in  0:05:37 at 09:18:44
 9:13:08 Offset 318MB (3.49%) Done in  0:04:33 at 09:17:41
 9:13:09 Offset 402MB (4.41%) Done in  0:03:51 at 09:17:00
 9:13:10 Offset 486MB (5.33%) Done in  0:03:22 at 09:16:32
 ...
 9:18:49 Offset 8875MB (97.24%) Done in  0:00:09 at 09:18:58
 9:18:51 Offset 8959MB (98.16%) Done in  0:00:06 at 09:18:57
 9:18:57 Offset 9042MB (99.08%) Done in  0:00:03 at 09:19:00
All data are read; waiting for threads to finish...
Time elapsed waiting for 2 threads to finish:
     (timeout in 60 min.)
All Threads Finished!
Producer time spent waiting: 274.653 sec.
Average consumer time spent waiting: 8.30007 sec.
*******************************************
** bulk_extractor is probably CPU bound. **
**    Run on a computer with more cores  **
**      to get better performance.       **
*******************************************
MD5 of Disk Image: fca945d20e5a1222672c98879959cd45
Phase 2. Shutting down scanners
Phase 3. Creating Histograms
Elapsed time: 365.561 sec.
Total MB processed: 9126
Overall performance: 24.9666 MBytes/sec (12.4833 MBytes/sec/thread)
Total email features found: 1099

analyst@sift:~$ BEViewer &
[1] 20920
analyst@sift:~$ Bulk Extractor Viewer Version 1.6.0-dev-rec02
ReportSelectionManager.setReportSelection image file:
/mnt/hgfs/AnalystVMShare/memdump-pony-2.mem, features file: null, selection
type: None
```

After the memory image has been processed, run BEViewer and point it to the report.xml file included in output directory. We are now able to go through the extracted artifacts, perform searches etc. In the picture below, domain_histogram is reviewed. The results depict the domain described in public reports has been found in the memory image. However, this cannot be taken as evidence that the domain has indeed been visited. If the domain were present in a file loaded into memory, it could have been carved out by BulkExtractor without visiting it. Nevertheless, if the domain were to be malicious at the time of taking the memory image, it is probable that the system has been compromised.



```
bulk
bulk
    aes_keys.txt
    ccn.txt
    ccn_histogram.txt
    domain.txt
    domain_histogram.tx
    email.txt
    email_domain_histog
    email_histogram.txt
    ether.txt
    ether_histogram.txt
    exif.txt
    ip.txt
    ip_histogram.txt
    json.txt
    ntfsindx_carved.txt
    ntfslogfile_carved.txt
    ntfsmft_carved.txt
    ntfsusn_carved.txt
    pii.txt
    rfc822.txt
    sqlite_carved.txt
    telephone.txt
    telephone_histogram
    url.txt
    url_facebook-addres
    url_facebook-id.txt
    url_histogram.txt
    url_searches.txt
    url_services.txt
    utmp_carved.txt
    windirs.txt
    winlnk.txt
    winpe.txt
```

Feature Filter: see

Histogram File domain_histogram.txt

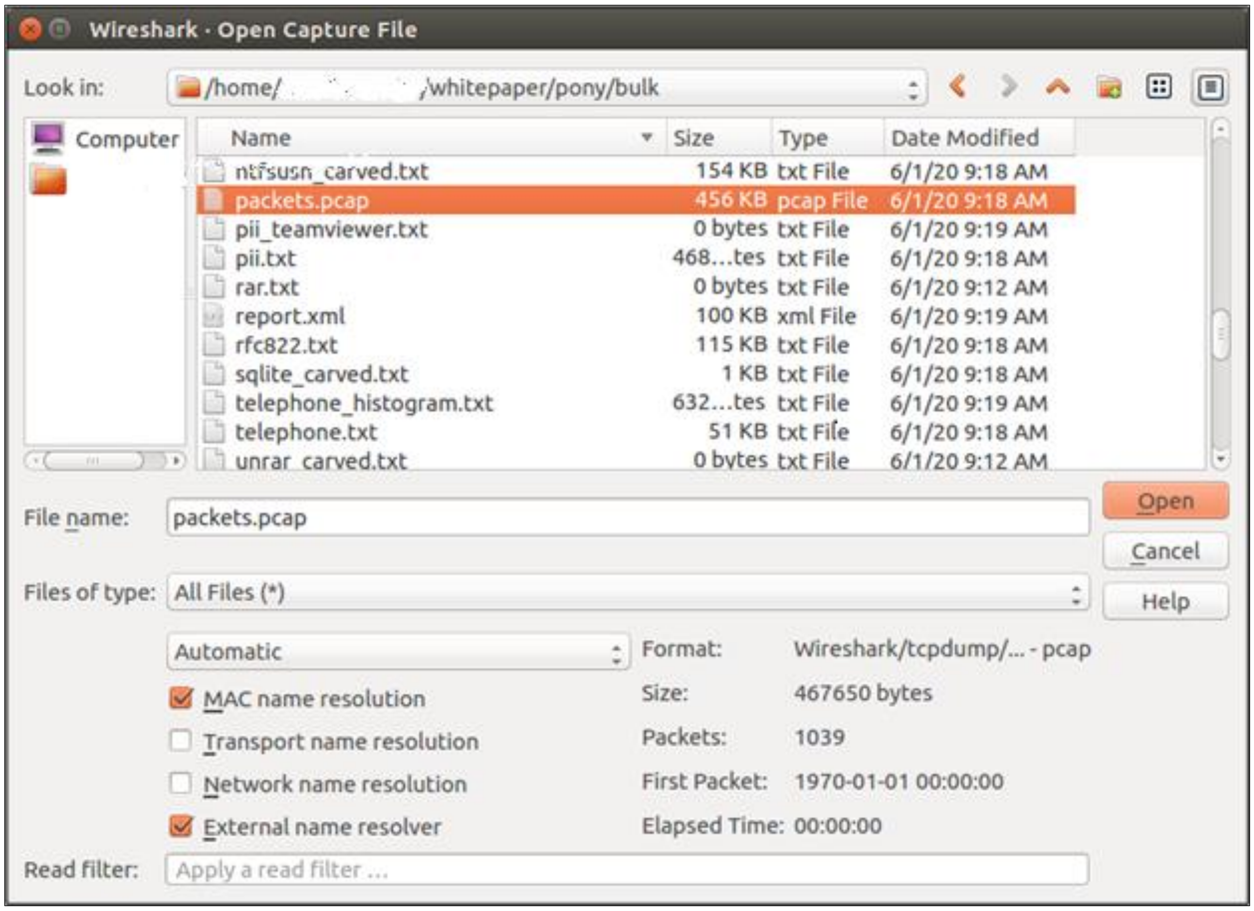
```
n=24    seeyouonlineservice.com
n=12    ww9.seeyouonlineservice.com
n=2     desertroseenterprises.com
n=2     rep4.upseek.org
n=1     1see.ir
n=1     ns.acdsee.com
n=1     seedstar.net
n=1     seeing.mm.am
n=1     seek.3721.com
n=1     www.naisee.com
n=1     www.seeknewlive.com
```

Referenced Feature File domain.txt
Referenced Feature seeyouonlineservice.com

```
38532074        seeyouonlineservice.com
4754002278      ww9.seeyouonlineservice.com
4754002379      seeyouonlineservice.com
4756433680      seeyouonlineservice.com
4761035897      seeyouonlineservice.com
4761036170      seeyouonlineservice.com
4762923070      ww9.seeyouonlineservice.com
4762923171      seeyouonlineservice.com
4764261734      ww9.seeyouonlineservice.com
4764261835      seeyouonlineservice.com
4764341028      seeyouonlineservice.com
4764341301      seeyouonlineservice.com
4836815446      seeyouonlineservice.com
4842955342      seeyouonlineservice.com
4869711886      seeyouonlineservice.com
6977782372      ww9.seeyouonlineservice.com
6978133904      ww9.seeyouonlineservice.com
7090434318      seeyouonlineservice.com
7106054808      ww9.seeyouonlineservice.com
7147286798      seeyouonlineservice.com
```

Figure 3: BEViewer GUI

Besides data shown in the GUI, BulkExtractor provides additional output. As mentioned previously, the tool carves network packets from the memory and these can be subsequently analyzed in WireShark or other packet viewers of the analyst's choice. In this case, we have searched for DNS requests for the above-mentioned domain with no luck. This is not at all evidence that the domain has not been contacted, only that the packet may have been overwritten in the memory by the time we managed to acquire it. Only one of the packets from C2 communication was left in the memory, TCP with FIN-ACK flags set – so most likely the last one left from that part of the communication.


Wireshark · Open Capture File

Look in: /home/[redacted]/whitepaper/pony/bulk

Name	Size	Type	Date Modified
ntfsusn_carved.txt	154 KB	txt File	6/1/20 9:18 AM
packets.pcap	456 KB	pcap File	6/1/20 9:18 AM
pii_teamviewer.txt	0 bytes	txt File	6/1/20 9:19 AM
pii.txt	468...tes	txt File	6/1/20 9:18 AM
rar.txt	0 bytes	txt File	6/1/20 9:12 AM
report.xml	100 KB	xml File	6/1/20 9:19 AM
rfc822.txt	115 KB	txt File	6/1/20 9:18 AM
sqlite_carved.txt	1 KB	txt File	6/1/20 9:18 AM
telephone_histogram.txt	632...tes	txt File	6/1/20 9:19 AM
telephone.txt	51 KB	txt File	6/1/20 9:18 AM
unrar_carved.txt	0 bytes	txt File	6/1/20 9:12 AM

File name: packets.pcap

Files of type: All Files (*)

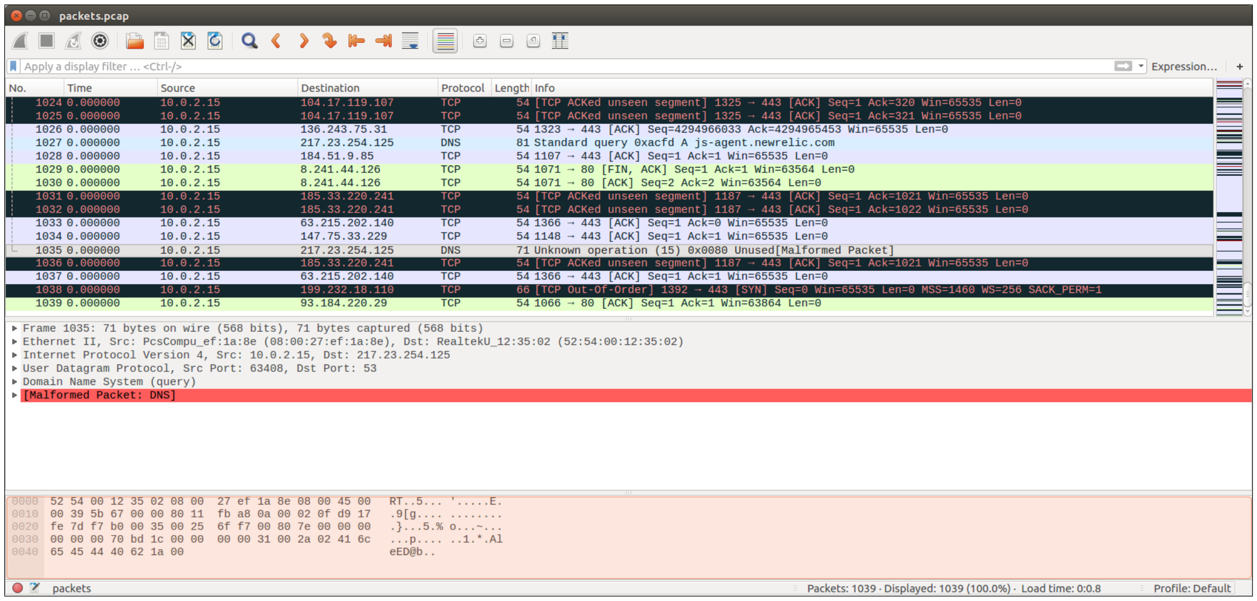
Automatic

[✓] MAC name resolution
[ ] Transport name resolution
[ ] Network name resolution
[✓] External name resolver

Format: Wireshark/tcpdump/... - pcap
Size: 467650 bytes
Packets: 1039
First Packet: 1970-01-01 00:00:00
Elapsed Time: 00:00:00

Read filter: Apply a read filter ...

Figure 4: Opening PCAP file with extracted packets.


packets.pcap

Apply a display filter ... <Ctrl-/>

```
No.   Time      Source      Destination      Protocol Length Info
 1024 0.000000  10.0.2.15   104.17.119.107   TCP      54 [TCP ACKed unseen segment] 1325 → 443 [ACK] Seq=1 Ack=320 Win=65535 Len=0
 1025 0.000000  10.0.2.15   104.17.119.107   TCP      54 [TCP ACKed unseen segment] 1325 → 443 [ACK] Seq=1 Ack=321 Win=65535 Len=0
 1026 0.000000  10.0.2.15   136.243.75.31    TCP      54 1323 → 443 [ACK] Seq=4294966033 Ack=4294965453 Win=65535 Len=0
 1027 0.000000  10.0.2.15   217.23.254.125   DNS      81 Standard query 0xacfd A js-agent.newrelic.com
 1028 0.000000  10.0.2.15   184.51.9.85      TCP      54 1107 → 443 [ACK] Seq=1 Ack=1 Win=65535 Len=0
 1029 0.000000  10.0.2.15   8.241.44.126     TCP      54 1071 → 80 [FIN, ACK] Seq=1 Ack=1 Win=63564 Len=0
 1030 0.000000  10.0.2.15   8.241.44.126     TCP      54 1071 → 80 [ACK] Seq=2 Ack=2 Win=63564 Len=0
 1031 0.000000  10.0.2.15   185.33.220.241   TCP      54 [TCP ACKed unseen segment] 1187 → 443 [ACK] Seq=1 Ack=1021 Win=65535 Len=0
 1032 0.000000  10.0.2.15   185.33.220.241   TCP      54 [TCP ACKed unseen segment] 1187 → 443 [ACK] Seq=1 Ack=1022 Win=65535 Len=0
 1033 0.000000  10.0.2.15   63.215.202.140   TCP      54 1366 → 443 [ACK] Seq=1 Ack=0 Win=65535 Len=0
 1034 0.000000  10.0.2.15   147.75.33.229    TCP      54 1148 → 443 [ACK] Seq=1 Ack=1 Win=65535 Len=0
 1035 0.000000  10.0.2.15   217.23.254.125   DNS      71 Unknown operation (15) 0x0080 Unused[Malformed Packet]
 1036 0.000000  10.0.2.15   185.33.220.241   TCP      54 [TCP ACKed unseen segment] 1187 → 443 [ACK] Seq=1 Ack=1021 Win=65535 Len=0
 1037 0.000000  10.0.2.15   63.215.202.140   TCP      54 1366 → 443 [ACK] Seq=1 Ack=1 Win=65535 Len=0
 1038 0.000000  10.0.2.15   199.232.18.110   TCP      66 [TCP Out-Of-Order] 1392 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM=1
 1039 0.000000  10.0.2.15   93.184.220.29    TCP      54 1066 → 80 [ACK] Seq=1 Ack=1 Win=63864 Len=0
```

```
▸ Frame 1035: 71 bytes on wire (568 bits), 71 bytes captured (568 bits)
▸ Ethernet II, Src: PcsCompu_ef:1a:8e (08:00:27:ef:1a:8e), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)
▸ Internet Protocol Version 4, Src: 10.0.2.15, Dst: 217.23.254.125
▸ User Datagram Protocol, Src Port: 63408, Dst Port: 53
▸ Domain Name System (query)
▸ [Malformed Packet: DNS]
```

```
0000  52 54 00 12 35 02 08 00  27 ef 1a 8e 08 00 45 00   RT..5... '.....E.
0010  00 39 5b 67 00 00 80 11  fb a8 0a 00 02 0f d9 17   .9[g.... ........
0020  fe 7d f7 b0 00 35 00 25  6f f7 00 80 7e 00 00 00   .}...5.% o...~...
0030  00 00 00 70 bd 1c 00 00  00 00 31 00 2a 02 41 6c   ...p.... ..1.*.Al
0040  65 45 44 40 62 1a 00                               eED@b..
```

packets   Packets: 1039 · Displayed: 1039 (100.0%) · Load time: 0:0.8   Profile: Default

Figure 5: Wireshark

CONCLUSION

In Part I of *Windows Memory Forensics* series, we have covered several topics. First, how to acquire a memory image. We have introduced important functions of FTK Imager and WinPMEM, both of which are well-known tools for memory acquisition. Then, we described how to analyze memory in an unstructured way. We have also provided basic information on strings search. Lastly, we have discussed Bulk Extractor, along with its operation and features.

In the next part of *Windows Memory Forensics* series, we will start with the structured analysis of Windows memory images, using Volatility.