Challenge: QBot Lab

Platform: CyberDefenders

Category: Endpoint Forensics

Difficulty: Medium

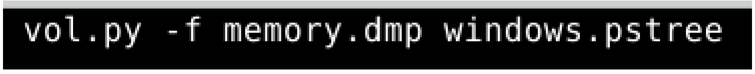
Tools Used: Volatility3, VirusTotal

Summary: This lab involved analysing a memory dump using Volatility3 and VirusTotal to trace malicious activity. I found it really enjoyable and was able to learn new techniques (like the dumpfiles plugin to dump files associated with a process) to better improve my memory forensics skills. Those of you who enjoy memory forensics, especially Volatility, should give this a go.

Scenario: A company's security team detected unusual network activity linked to a potential malware infection. As a forensic analyst, your mission is to investigate a memory dump, identify the malicious process, extract artifacts, and uncover Command and Control (C2) communications. Using Volatility3, analyze the attack, trace its origin, and provide actionable intelligence.

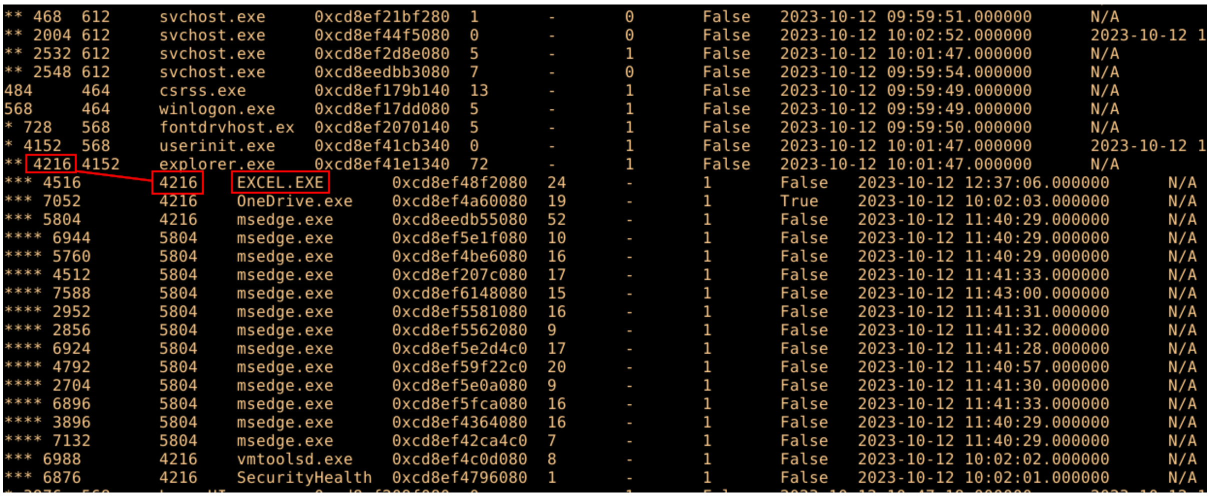
Our first step is identifying the initial point of contact the malware made with an external server. Can you specify the first IP address the malware attempted to communicate with?

To begin, I used the pstree plugin. This plugin shows all running processes in a hierarchical (parent-child) format at the time of the memory capture. It's especially helpful in spotting unusual relationships, for example, legitimate software spawning tools like PowerShell or cmd.exe:



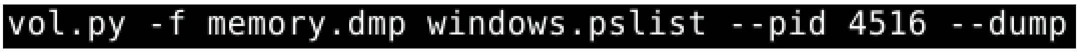
```
vol.py -f memory.dmp windows.pstree
```

In the output, I started by looking for any suspicious processes, focusing on those with weird parent-child relationships. I eventually identified an instance of EXCEL.EXE running under explorer.exe. This suggests a user opened an Excel file manually, which is common behaviour. However, Excel is a known vector for malware delivery via malicious macros, so further scrutiny is required:

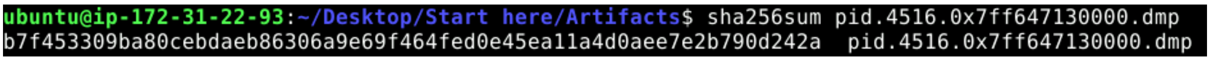


```
** 468  612     svchost.exe     0xcd8ef21bf280  1       -       0       False   2023-10-12 09:59:51.000000      N/A
** 2004 612     svchost.exe     0xcd8ef44f5080  0       -       0       False   2023-10-12 10:02:52.000000      2023-10-12 1
** 2532 612     svchost.exe     0xcd8ef2d8e080  5       -       1       False   2023-10-12 10:01:47.000000      N/A
** 2548 612     svchost.exe     0xcd8eedbb3080  7       -       0       False   2023-10-12 09:59:54.000000      N/A
484     464     csrss.exe       0xcd8ef179b140  13      -       1       False   2023-10-12 09:59:49.000000      N/A
568     464     winlogon.exe    0xcd8ef17dd080  5       -       1       False   2023-10-12 09:59:49.000000      N/A
* 728   568     fontdrvhost.ex  0xcd8ef2070140  5       -       1       False   2023-10-12 09:59:50.000000      N/A
* 4152  568     userinit.exe    0xcd8ef41cb340  0       -       1       False   2023-10-12 10:01:47.000000      2023-10-12 1
** 4216 4152    explorer.exe    0xcd8ef41e1340  72      -       1       False   2023-10-12 10:01:47.000000      N/A
*** 4516        4216    EXCEL.EXE       0xcd8ef48f2080  24      -       1       False   2023-10-12 12:37:06.000000      N/A
*** 7052        4216    OneDrive.exe    0xcd8ef4a60080  19      -       1       True    2023-10-12 10:02:03.000000      N/A
*** 5804        4216    msedge.exe      0xcd8eedb55080  52      -       1       False   2023-10-12 11:40:29.000000      N/A
**** 6944       5804    msedge.exe      0xcd8ef5e1f080  10      -       1       False   2023-10-12 11:40:29.000000      N/A
**** 5760       5804    msedge.exe      0xcd8ef4be6080  16      -       1       False   2023-10-12 11:40:29.000000      N/A
**** 4512       5804    msedge.exe      0xcd8ef207c080  17      -       1       False   2023-10-12 11:41:33.000000      N/A
**** 7588       5804    msedge.exe      0xcd8ef6148080  15      -       1       False   2023-10-12 11:43:00.000000      N/A
**** 2952       5804    msedge.exe      0xcd8ef5581080  16      -       1       False   2023-10-12 11:41:31.000000      N/A
**** 2856       5804    msedge.exe      0xcd8ef5562080  9       -       1       False   2023-10-12 11:41:32.000000      N/A
**** 6924       5804    msedge.exe      0xcd8ef5e2d4c0  17      -       1       False   2023-10-12 11:41:28.000000      N/A
**** 4792       5804    msedge.exe      0xcd8ef59f22c0  20      -       1       False   2023-10-12 11:40:57.000000      N/A
**** 2704       5804    msedge.exe      0xcd8ef5e0a080  9       -       1       False   2023-10-12 11:41:30.000000      N/A
**** 6896       5804    msedge.exe      0xcd8ef5fca080  16      -       1       False   2023-10-12 11:41:33.000000      N/A
**** 3896       5804    msedge.exe      0xcd8ef4364080  16      -       1       False   2023-10-12 11:40:29.000000      N/A
**** 7132       5804    msedge.exe      0xcd8ef42ca4c0  7       -       1       False   2023-10-12 11:40:29.000000      N/A
*** 6988        4216    vmtoolsd.exe    0xcd8ef4c0d080  8       -       1       False   2023-10-12 10:02:02.000000      N/A
*** 6876        4216    SecurityHealth  0xcd8ef4796080  1       -       1       False   2023-10-12 10:02:01.000000      N/A
```

To verify if this EXCEL.EXE process was legitimate, I used the pslist plugin with the --dump option to extract the binary of this process from memory:

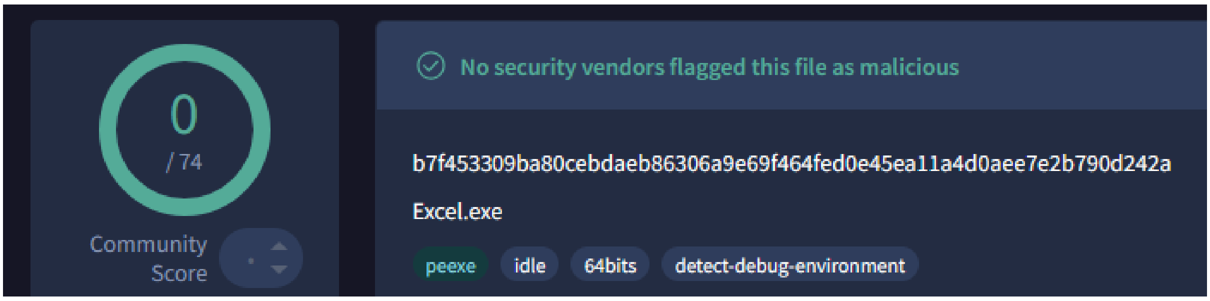


```
vol.py -f memory.dmp windows.pslist --pid 4516 --dump
```



```
ubuntu@ip-172-31-22-93:~/Desktop/Start here/Artifacts$ sha256sum pid.4516.0x7ff647130000.dmp
b7f453309ba80cebdaeb86306a9e69f464fed0e45ea11a4d0aee7e2b790d242a  pid.4516.0x7ff647130000.dmp
```

After submitting the hash to VirusTotal, it received no detections, which likely indicates that it is the legitimate Excel binary:

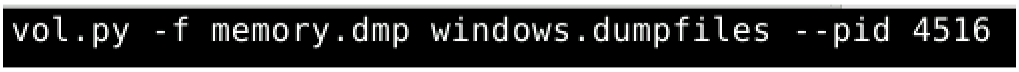


No security vendors flagged this file as malicious

b7f453309ba80cebdaeb86306a9e69f464fed0e45ea11a4d0aee7e2b790d242a

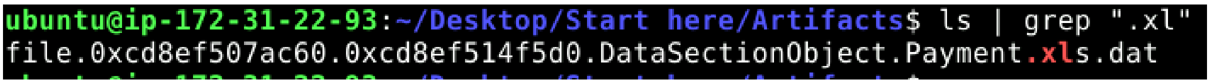
Excel.exe

Using the dumpfiles plugin, I extracted all files in memory associated with the EXCEL.EXE process:



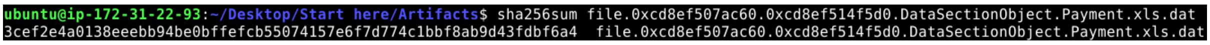
```
vol.py -f memory.dmp windows.dumpfiles --pid 4516
```

Among them, I found an Excel document that is likely what the user opened:

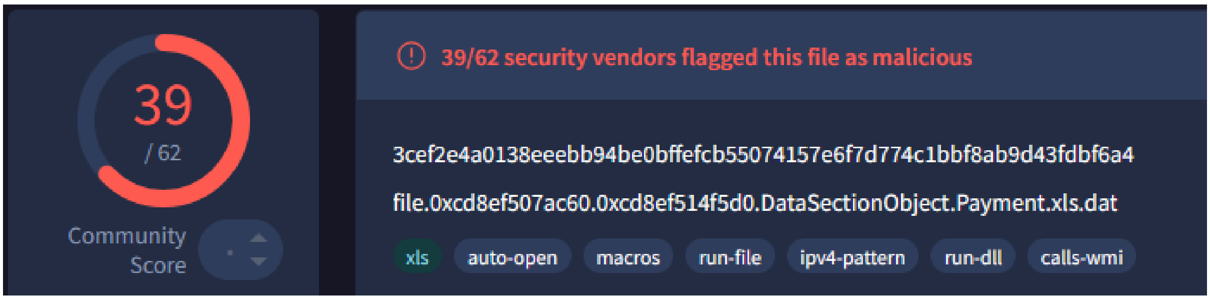


```
ubuntu@ip-172-31-22-93:~/Desktop/Start here/Artifacts$ ls | grep ".xl"
file.0xcd8ef507ac60.0xcd8ef514f5d0.DataSectionObject.Payment.xls.dat
```

After generating the SHA256 hash of the file, I submitted it to VirusTotal:



```
ubuntu@ip-172-31-22-93:~/Desktop/Start here/Artifacts$ sha256sum file.0xcd8ef507ac60.0xcd8ef514f5d0.DataSectionObject.Payment.xls.dat
3cef2e4a0138eeebb94be0bffefcb55074157e6f7d774c1bbf8ab9d43fdbf6a4  file.0xcd8ef507ac60.0xcd8ef514f5d0.DataSectionObject.Payment.xls.dat
```



39 / 62

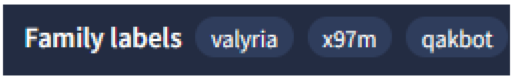
Community Score

39/62 security vendors flagged this file as malicious

3cef2e4a0138eeebb94be0bffefcb55074157e6f7d774c1bbf8ab9d43fdbf6a4

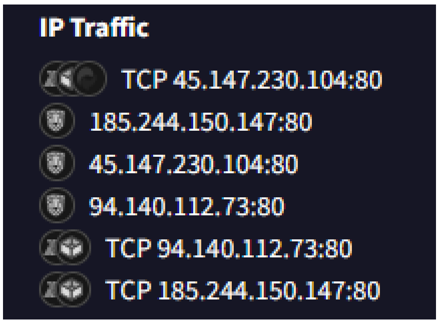
file.0xcd8ef507ac60.0xcd8ef514f5d0.DataSectionObject.Payment.xls.dat

This file has 39/62 detections, meaning that it is certainly malicious. We can also see that it is associated with qakbot aka QBot:



Family labels valyria x97m qakbot

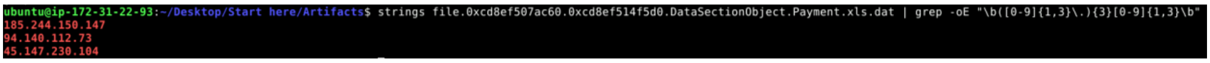
Within the behaviour section we can find a list of IPs this file has communicated with:



IP Traffic

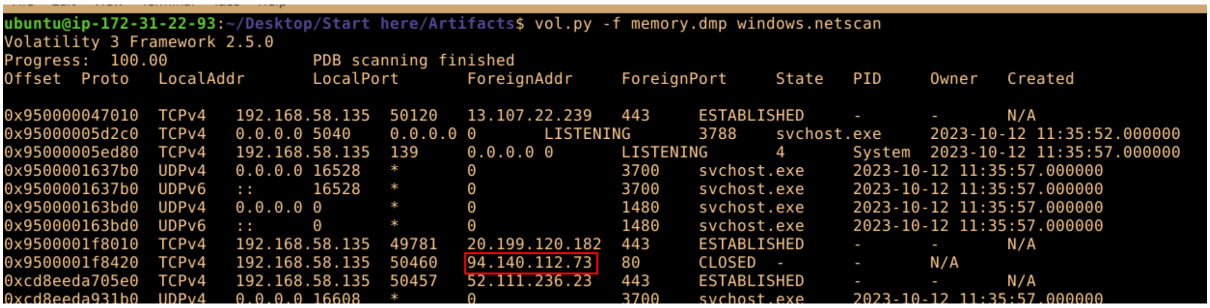
TCP 45.147.230.104:80
185.244.150.147:80
45.147.230.104:80
94.140.112.73:80
TCP 94.140.112.73:80
TCP 185.244.150.147:80

If you use strings against the file we dumped, you can find some of these IPs:



```
ubuntu@ip-172-31-22-93:~/Desktop/Start here/Artifacts$ strings file.0xcd8ef507ac60.0xcd8ef514f5d0.DataSectionObject.Payment.xls.dat | grep -oE "\b([0-9]{1,3}\.){3}[0-9]{1,3}\b"
185.244.150.147
94.140.112.73
45.147.230.104
```

To verify which of these IPs were contacted, I used the netscan plugin. This plugin reveals live or recently closed network connections, and sometimes the process associated with that connection:



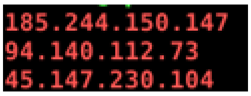
```
ubuntu@ip-172-31-22-93:~/Desktop/Start here/Artifacts$ vol.py -f memory.dmp windows.netscan
Volatility 3 Framework 2.5.0
Progress:  100.00               PDB scanning finished
Offset  Proto   LocalAddr       LocalPort       ForeignAddr     ForeignPort     State   PID     Owner   Created

0x950000047010  TCPv4   192.168.58.135  50120   13.107.22.239   443     ESTABLISHED     -       -       N/A
0x95000005d2c0  TCPv4   0.0.0.0 5040    0.0.0.0 0       LISTENING       3788    svchost.exe     2023-10-12 11:35:52.000000
0x95000005ed80  TCPv4   192.168.58.135  139     0.0.0.0 0       LISTENING       4       System  2023-10-12 11:35:57.000000
0x9500001637b0  UDPv4   0.0.0.0 16528   *       0               3700    svchost.exe     2023-10-12 11:35:57.000000
0x9500001637b0  UDPv6   ::      16528   *       0               3700    svchost.exe     2023-10-12 11:35:57.000000
0x950000163bd0  UDPv4   0.0.0.0 0       *       0               1480    svchost.exe     2023-10-12 11:35:57.000000
0x950000163bd0  UDPv6   ::      0       *       0               1480    svchost.exe     2023-10-12 11:35:57.000000
0x9500001f8010  TCPv4   192.168.58.135  49781   20.199.120.182  443     ESTABLISHED     -       -       N/A
0x9500001f8420  TCPv4   192.168.58.135  50460   94.140.112.73   80      CLOSED  -       -       N/A
0xcd8eeda705e0  TCPv4   192.168.58.135  50457   52.111.236.23   443     ESTABLISHED     -       -       N/A
0xcd8eeda931b0  UDPv4   0.0.0.0 16608   *       0               3700    svchost.exe     2023-10-12 11:35:57.000000
```

In the output, I matched one of the IPs from the strings within the .xls file to an entry under the ForeignAddr field.

Answer: 94.140.112.73

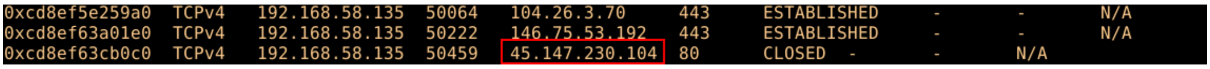
We need to determine if the malware attempted to communicate with another IP. Which IP address did the malware attempt to communicate with again?

We previously extracted and analysed the malicious Excel file that we dumped from memory. We were able to identify three IP addresses within the strings of that file:



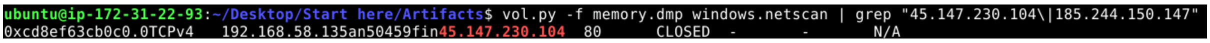
```
185.244.150.147
94.140.112.73
45.147.230.104
```

If you continue to explore the output of the netscan plugin we can find a closed connection to 45.147.230.104 (one of the IPs found in the strings output, and on VirusTotal):



```
0xcd8ef5e259a0  TCPv4   192.168.58.135  50064   104.26.3.70     443     ESTABLISHED     -       -       N/A
0xcd8ef63a01e0  TCPv4   192.168.58.135  50222   146.75.53.192   443     ESTABLISHED     -       -       N/A
0xcd8ef63cb0c0  TCPv4   192.168.58.135  50459   45.147.230.104  80      CLOSED  -       -       N/A
```

Alternatively, you can grep for the two other IPs we found within the file:



```
ubuntu@ip-172-31-22-93:~/Desktop/Start here/Artifacts$ vol.py -f memory.dmp windows.netscan | grep "45.147.230.104\|185.244.150.147"
0xcd8ef63cb0c0.0TCPv4   192.168.58.135an50459fin45.147.230.104  80      CLOSED  -       -       N/A
```

Answer: 45.147.230.104

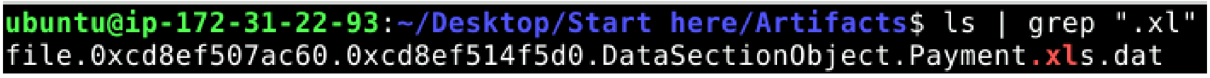
Identifying the process responsible for this suspicious behavior helps reconstruct the sequence of events leading to the execution of the malware and its source. What is the name of the process that initiated the malware?

We determined in our investigation conducted in question one that the process responsible for these network connections is EXCEL.EXE.

Answer: EXCEL.EXE

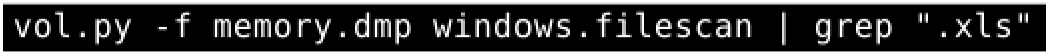
The malware's file name is crucial for further forensic analysis and extracting the malware. Can you provide its file name?

Recall earlier when we used the dumpfiles plugin to dump all the files associated with the EXCEL.EXE process, we observed an xls file called Payment.xls:

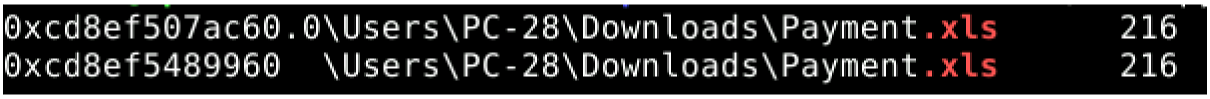


```
ubuntu@ip-172-31-22-93:~/Desktop/Start here/Artifacts$ ls | grep ".xl"
file.0xcd8ef507ac60.0xcd8ef514f5d0.DataSectionObject.Payment.xls.dat
```

Alternatively, you can use the filescan plugin which looks for files within the memory dump and pipe the output to grep so we can hunt for .xls files:



```
vol.py -f memory.dmp windows.filescan | grep ".xls"
```



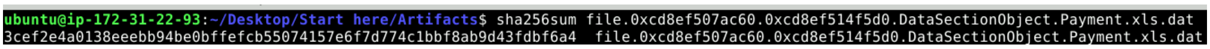
```
0xcd8ef507ac60.0\Users\PC-28\Downloads\Payment.xls     216
0xcd8ef5489960  \Users\PC-28\Downloads\Payment.xls     216
```

The .xls file being present within the Downloads directory makes it even more suspicious, as it likely suggests that the user fell for a phishing attack.

Answer: Payment.xls

Hashes are like digital fingerprints for files. Once the hash is known, it can be used to scan other systems within the network to identify if the same malicious file exists elsewhere. What is the SHA256 hash of the malware?

You can use the sha256sum command to generate the SHA256 hash of the Payment.xls file we determined to be qakbot in question one:

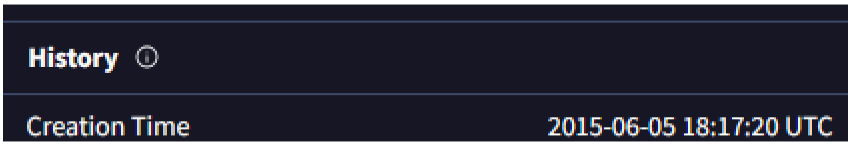


```
ubuntu@ip-172-31-22-93:~/Desktop/Start here/Artifacts$ sha256sum file.0xcd8ef507ac60.0xcd8ef514f5d0.DataSectionObject.Payment.xls.dat
3cef2e4a0138eeebb94be0bffefcb55074157e6f7d774c1bbf8ab9d43fdbf6a4  file.0xcd8ef507ac60.0xcd8ef514f5d0.DataSectionObject.Payment.xls.dat
```

Answer: 3cef2e4a0138eeebb94be0bffefcb55074157e6f7d774c1bbf8ab9d43fdbf6a4

To trace the origin of the malware and understand its development timeline, can you provide the UTC creation time of the malware file?

If you submit the SHA256 hash of the Payment.xls file from the previous question and navigate to the Details tab > History section, we can find the creation timestamp:



History

Creation Time 2015-06-05 18:17:20 UTC

Answer: 2015-06-05 18:17