# Ramnit Lab

## Scenario

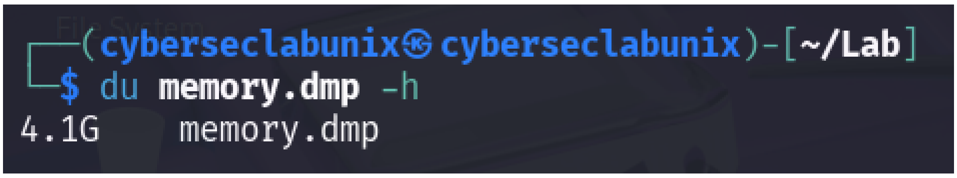
Our intrusion detection system has alerted us to suspicious behavior on a workstation, pointing to a likely malware intrusion. A memory dump of this system has been taken for analysis. Your task is to analyze this dump, trace the malware's actions, and report key findings.

## Questions

1. What is the name of the process responsible for the suspicious activity?
2. What is the exact path of the executable for the malicious process?
3. Identifying network connections is crucial for understanding the malware's communication strategy. What IP address did the malware attempt to connect to?
4. To determine the specific geographical origin of the attack, Which city is associated with the IP address the malware communicated with?
5. Hashes serve as unique identifiers for files, assisting in the detection of similar threats across different machines. What is the SHA1 hash of the malware executable?
6. Examining the malware's development timeline can provide insights into its deployment. What is the compilation timestamp for the malware?
7. Identifying the domains associated with this malware is crucial for blocking future malicious communications and detecting any ongoing interactions with those domains within our network. Can you provide the domain connected to the malware?

## Analysis

We are provided with a `memory.dmp` file, with the size of 4.1G.




By quickly running a `file` command on the file, we can get some basic information about the dump file.


```
memory.dmp:
    MS Windows 64bit crash dump,
    version 15.19041,
    4 processors,
    DumpType (0x1),
    1048576 pages
```

We can also try to peak inside the file to check what it's made of. I chose to use a `hexdump` and then `strings` to get some more information about the file.

```
┌──(cyberseclabunix㉿cyberseclabunix)-[~/Lab]
└─$ hexdump -C 16 memory.dmp | head -n 256
hexdump: 16: No such file or directory
00000000  50 41 47 45 44 55 36 34  0f 00 00 00 61 4a 00 00  |PAGEDU64....aJ..|
00000010  02 d0 1a 00 00 00 00 00  00 00 00 00 00 a2 ff ff  |................|
00000020  90 a2 02 4a 00 f8 ff ff  60 df 01 4a 00 f8 ff ff  |...J....`..J....|
00000030  64 86 00 00 04 00 00 00  80 00 00 00 50 41 47 45  |d...........PAGE|
00000040  54 44 4f 00 00 00 00 00  00 00 00 00 00 00 00 00  |TDO.............|
00000050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000060  50 41 47 45 50 41 47 45  50 41 47 45 50 41 47 45  |PAGEPAGEPAGEPAGE|
*
00000080  20 0b 00 4a 00 f8 ff ff  02 00 00 00 00 00 00 00  | ..J............|
00000090  00 00 10 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000000a0  00 00 0c 00 00 00 00 00  00 00 10 00 00 00 00 00  |................|
000000b0  00 00 04 00 00 00 00 00  50 41 47 45 50 41 47 45  |........PAGEPAGE|
000000c0  50 41 47 45 50 41 47 45  50 41 47 45 50 41 47 45  |PAGEPAGEPAGEPAGE|
*
00000370  50 41 47 45 50 41 47 45  0f 00 10 00 80 1f 00 00  |PAGEPAGE........|
00000380  10 00 2b 00 2b 00 53 00  2b 00 18 00 86 02 04 00  |..+.+.S.+.......|
00000390  50 41 47 45 50 41 47 45  50 41 47 45 50 41 47 45  |PAGEPAGEPAGEPAGE|
```

We can safely consider this a 64-bit Windows pagefile. Next what we are going to do is to use a popular digital forensic tool called **Volatility**.

```
┌──(venv)─(cyberseclabunix㉿cyberseclabunix)-[~/volatility3]
└─$ vol
Volatility 3 Framework 2.26.2
usage: vol [-h] [-c CONFIG] [--parallelism [{processes,threads,off}]] [-e EXTEND] [-p PLUGIN_DIRS] [-s SYMBOL_DIRS] [-v] [-l LOG] [-o OUTPUT_DIR] [-q] [-r RENDERER] [-f FILE]
           [--write-config] [--save-config SAVE_CONFIG] [--clear-cache] [--cache-path CACHE_PATH] [--offline | -u URL] [--filters FILTERS] [--hide-columns [HIDE_COLUMNS ...]]
           [--single-location SINGLE_LOCATION] [--stackers [STACKERS ...]] [--single-swap-locations [SINGLE_SWAP_LOCATIONS ...]]
           PLUGIN ...
vol: error: Please select a plugin to run (see 'vol --help' for options
```

First, what want to do is get basic information of this Windows pagefile. We will use `windows.info` plugin.

```
$ vol -f memory.dmp windows.info

┌──(venv)─(cyberseclabunix㉿cyberseclabunix)-[~/volatility3]
└─$ vol -f memory.dmp windows.info
Volatility 3 Framework 2.26.2
Progress:  100.00               PDB scanning finished
Variable        Value

Kernel Base     0xf80049400000
DTB     0x1ad000
Symbols
file:///home/cyberseclabunix/volatility3/volatility3/symbols/windows/ntkrnlmp.p
db/68A17FAF3012B7846079AEECDBE0A583-1.json.xz
Is64Bit True
IsPAE   False
layer_name      0 WindowsIntel32e
memory_layer    1 WindowsCrashDump64Layer
base_layer      2 FileLayer
KdDebuggerDataBlock     0xf8004a000b20
NTBuildLab      19041.1.amd64fre.vb_release.1912
CSDVersion      0
KdVersionBlock  0xf8004a00f398
Major/Minor     15.19041
MachineType     34404
KeNumberProcessors      4
SystemTime      2024-02-01 19:54:11+00:00
NtSystemRoot    C:\Windows
NtProductType   NtProductWinNt
```

```
NtMajorVersion  10
NtMinorVersion  0
PE MajorOperatingSystemVersion  10
PE MinorOperatingSystemVersion  0
PE Machine      34404
PE TimeDateStamp        Wed Jun 28 04:14:26 1995
```

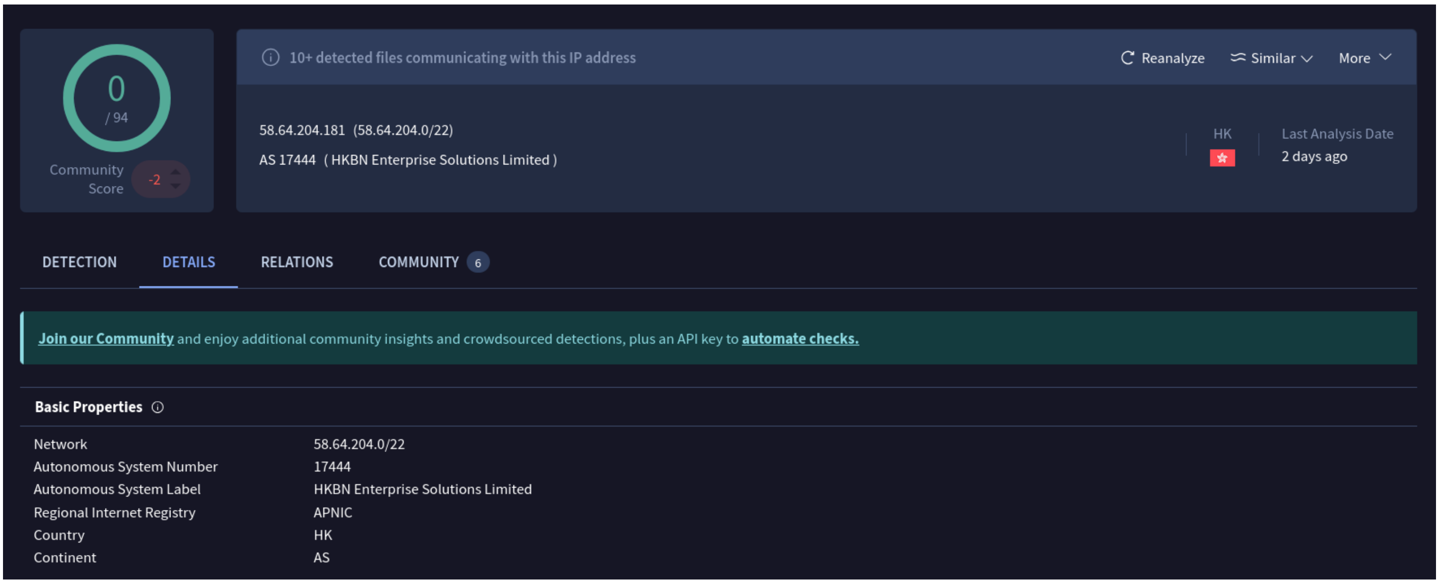
This dump file is from a Windows 10 (x64) desktop/server.

What I like to do first when analyzing Windows 10 is to look at the active connections. We can achive this by using `vol -f memory.dmp windows.netstat`. I would recommend to always save the output to a file, for convenience.

I'm looking for something that stands out (i.e. uncommon port). And we already got something interesting.

```
0xca82b38b0730,TCPv4,192.168.19.133,49765,52.179.219.14,443,ESTABLISHED,2500,svchost.exe,2024-02-01 19:52:58.000
0xca82b78cba20,TCPv4,192.168.19.133,49694,95.100.200.202,443,CLOSE_WAIT,5912,WWAHost.exe,2024-02-01 19:49:20.000
0xca82b7e5a700,TCPv4,192.168.19.133,49700,95.100.200.202,443,CLOSE_WAIT,5912,WWAHost.exe,2024-02-01 19:49:20.000
0xca82b8bc2b30,TCPv4,192.168.19.133,49682,58.64.204.181,5202,SYN_SENT,4628,ChromeSetup.ex,2024-02-01 19:48:51.00
0xca82b8baea20,TCPv4,192.168.19.133,49696,95.100.200.202,443,CLOSE_WAIT,5912,WWAHost.exe,2024-02-01 19:49:20.000
0xca82b1c2ed30,TCPv4,0.0.0.0,135,0.0.0.0,0,LISTENING,928,svchost.exe,2024-02-01 19:48:24.000000 UTC
```

A ChromeSetup.exe (4628) is trying to connect to 58.64.204.181 via 5202/tcp. Let's check the reputation of this address.




.

I expected this address to belong to Google's address space, but Virustotal says otherwise. We have the answer to our first question.

> What is the name of the process responsible for the suspicious activity? → **ChromeSetup.exe**

Now we can check the process tree of this suspicious process by using `vol -f input/memory.dmp windows.pstree --pid 4628`.

```
Volatility 3 Framework 2.26.2

PID     PPID    ImageFileName   Offset(V)       Threads Handles SessionId       Wow64   CreateTime      ExitTime        Audit   Cmd
Path

624     516     winlogon.exe    0xca82b28cb080  4       -       1       False   2024-02-01 19:48:23.000000 UTC  N/A
\Device\HarddiskVolume3\Windows\System32\winlogon.exe   winlogon.exe    C:\Windows\system32\winlogon.exe
* 4508  624     userinit.exe    0xca82b7426340  0       -       1       False   2024-02-01 19:48:26.000000 UTC  2024-02-01
19:48:52.000000 UTC     \Device\HarddiskVolume3\Windows\System32\userinit.exe   -       -
** 4568 4508    explorer.exe    0xca82b7440340  55      -       1       False   2024-02-01 19:48:26.000000 UTC  N/A
\Device\HarddiskVolume3\Windows\explorer.exe    C:\Windows\Explorer.EXE C:\Windows\Explorer.EXE
*** 4628        4568    ChromeSetup.ex  0xca82b830a300  4       -       1       True    2024-02-01 19:48:50.000000 UTC  N/A
\Device\HarddiskVolume3\Users\alex\Downloads\ChromeSetup.exe    "C:\Users\alex\Downloads\ChromeSetup.exe"       C:
\Users\alex\Downloads\ChromeSetup.exe

```

We can also achieve this by using `windows.cmdline` plugin.

```
┌──(venv)─(cyberseclabunix㉿cyberseclabunix)-[~/Lab]
└─$ vol -f input/memory.dmp windows.cmdline --pid 4628
Volatility 3 Framework 2.26.2
Progress:  100.00               PDB scanning finished
PID     Process Args

4628    ChromeSetup.ex  "C:\Users\alex\Downloads\ChromeSetup.exe"
```

We've got the answer to our second question!

> What is the exact path of the executable for the malicious process? → **C:\Users\alex\Downloads\ChromeSetup.exe**

We can also answer the third and the fourth question.

> Identifying network connections is crucial for understanding the malware's communication strategy. What IP address did the malware attempt to connect to? → **58.64.204.181**

> To determine the specific geographical origin of the attack, Which city is associated with the IP address the malware communicated with? → **Hong Kong**

Now we would like to get the hash of the malicious file. First we need to fetch the handles of the process. We will use `windows.handles` and `windows.dumpfiles` Volatility plugins.

```
┌──(venv)─(cyberseclabunix㉿cyberseclabunix)-[~/Lab]
└─$ vol -f input/memory.dmp windows.handles --pid 4628 | grep -e "File"
4628ressChromeSetup.ex  0xca82b8217680an0x40 finFile    0x100020        \Device\HarddiskVolume3\Windows
4628    ChromeSetup.ex  0xca82b82171d0  0x8c    File    0x100020        \Device\HarddiskVolume3\Users\alex\Downloads
4628    ChromeSetup.ex  0xca82b8219110  0x23c   File    0x120089        \Device\DeviceApi\CMApi
4628    ChromeSetup.ex  0xca82b821a0b0  0x258   File    0x100003        \Device\KsecDD
4628    ChromeSetup.ex  0xca82b821a6f0  0x294   File    0x100020        \Device\HarddiskVolume3\Windows\WinSxS\x86_microsoft.windows.common-controls_6595b641
44ccf1df_6.0.19041.1110_none_a8625c1886757984
4628    ChromeSetup.ex  0xca82b8538680  0x2f4   File    0x100001        \Device\CNG
```

Executing `vol -f input/memory.dmp windows.dumpfiles --pid 4628` will carve out all files related to this process. We are looking for out `ChromeSetup.exe` files

```
┌──(venv)─(cyberseclabunix㉿cyberseclabunix)-[~/Lab]
└─$ ls -lh | grep Chrome
-rw------- 1 cyberseclabunix cyberseclabunix 980K Jun  8 18:00 file.0xca82b85325a0.0xca82b7e06c80.ImageSectionObject.ChromeSetup.exe.img
-rw------- 1 cyberseclabunix cyberseclabunix 980K Jun  8 18:00 file.0xca82b85325a0.0xca82b83c7770.DataSectionObject.ChromeSetup.exe.dat
```

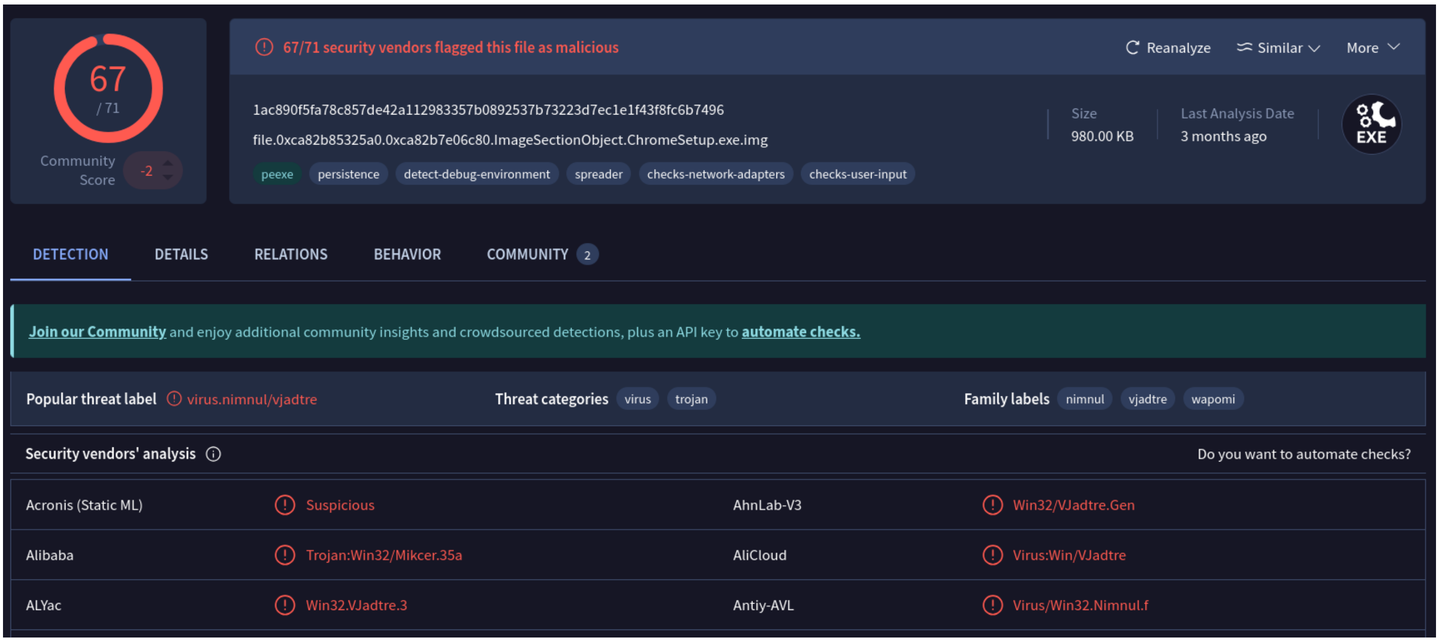
I've renamed these files to `ChromeSetup.exe.img` and `ChromeSetup.exe.dat`. What we are interested in is the file with `.img` extension.

```
┌──(venv)─(cyberseclabunix㉿cyberseclabunix)-[~/Lab]
└─$ sha1sum Chrome*
b9921cc2bfe3b43e457cdbc7d82b849c66f119cb  ChromeSetup.exe.dat
280c9d36039f9432433893dee6126d72b9112ad2  ChromeSetup.exe.img
```

The hash we are looking for is the second one: `280c9d36039f9432433893dee6126d72b9112ad2`. This is our answer to the fifth question

> Hashes serve as unique identifiers for files, assisting in the detection of similar threats across different machines. What is the SHA1 hash of the malware executable? → **280c9d36039f9432433893dee6126d72b9112ad2**

We can look it up in VirusTotal to verify this is it.




I always like to look at other reports, which you can find in **Behavior** tab > **Full reports** and select whichever you'd like. I went with Zenbox.

The sixth question ask us to find out the compilation timestamp of the malicious binary file. We will use `objdump` for this task.

```
objdump -p ChromeSetup.exe.img | grep "Time/Date"

Time/Date               Sun Dec  1 09:36:04 2019
```

You can also achieve this using `rabin2`.

```
rabin2 -I ChromeSetup.exe.img | grep "timestamp"
```

Now since my lab is in +0100 timezone, I have to substract one hour to get the UTC time.

> Examining the malware's development timeline can provide insights into its deployment. What is the compilation timestamp for the malware? → **2019-12-01 08:36**

Getting the answer to the last question, we can simply look for domains in one of the Virustotal full reports I mentioned above. Looking at the [Zenbox report](), we can find one domain.

| Domain | IP Resolutions | Signatures |
| --- | --- | --- |
| ddos.dnsnb8.net active | 34.174.61.199 | • Downloads files from webservers via HTTP<br>• Performs DNS lookups<br>• URLs found in memory or binary data<br>• Detected TCP or UDP traffic on non-standard ports<br>• Uses a known web browser user agent for HTTP communication |

> Identifying the domains associated with this malware is crucial for blocking future malicious communications and detecting any ongoing interactions with those domains within our network. Can you provide the domain connected to the malware? → **dnsnb8.net**

# Answers

1. What is the name of the process responsible for the suspicious activity?

   > ChromeSetup.exe

2. What is the exact path of the executable for the malicious process?

   > C:\Users\alex\Downloads\ChromeSetup.exe

3. Identifying network connections is crucial for understanding the malware's communication strategy. What IP address did the malware attempt to connect to?

   > 58.64.204.181

4. To determine the specific geographical origin of the attack, Which city is associated with the IP address the malware communicated with?

   > Hong Kong

5. Hashes serve as unique identifiers for files, assisting in the detection of similar threats across different machines. What is the SHA1 hash of the malware executable?

   > 280c9d36039f9432433893dee6126d72b9112ad2

6. Examining the malware's development timeline can provide insights into its deployment. What is the compilation timestamp for the malware?

   > 2019-12-01 08:36

7. Identifying the domains associated with this malware is crucial for blocking future malicious communications and detecting any ongoing interactions with those domains within our network. Can you provide the domain connected to the malware?

   > dnsnb8.net

# Resources used

- https://www.speedguide.net/port.php?port=5202
- https://www.youtube.com/watch?v=Uk3DEgY5Ue8
- https://www.google.com/search?client=firefox-b-e&channel=entpr&q="58.64.204.181"
- https://www.virustotal.com/gui/ip-address/58.64.204.181
- https://www.virustotal.com/gui/file/1ac890f5fa78c857de42a112983357b0892537b73223d7ec1e1f43f8fc6b7496/behavior

- https://vtbehaviour.commondatastorage.googleapis.com/1ac890f5fa78c857de42a112983357b0892537b73223d7ec1e1f43f8fc6b7496_CAPE Sandbox.html?GoogleAccessId=758681729565-rc7fgq07icj8c9dm2gi34a4cckv235v1@developer.gserviceaccount.com&Expires=1749399369&Signature=GbFtNl8dtx2Ur9onw3wBZNKbx59QTWuVHiwtuLLUiu9so%2FPmeJgSC8AlUmCh9cvdcUen%2FYodq5lIN0djnsBZvfZZioudKV50xVOMrLwMYh%2B6CYt3I8GSOEsJ6dTGIb8Ic8T%2F13O9nkDshNdAgswqoxFWtmloh6CSOcRjCFyQbfcJ2pPUDzVMEeD7ROsFx3WqFayGYobXanUoKbRn87hyPzDN04L2t71je%2FBbSE772UO9LZoaoPTkJnsPP%2BLub431NMZkvZrwHC0dtmKhqrVILSgktxTjaicP5RlcTGKDDGYMLXiSMuB10nXYPrgyneWuOly43H6%2B1xSgXgcuoaBtzA%3D%3D&response-content-type=text%2Fhtml;#behavior
- https://vtbehaviour.commondatastorage.googleapis.com/1ac890f5fa78c857de42a112983357b0892537b73223d7ec1e1f43f8fc6b7496_Zenbox.html?GoogleAccessId=758681729565-rc7fgq07icj8c9dm2gi34a4cckv235v1@developer.gserviceaccount.com&Expires=1749399179&Signature=iRWWvTrwx2A9Plgs8sKGDAciR%2FBJIexakDdV5wjjH7mwLHUNqQ5Sq9fFwnmACmts8TBEQxspzIrjTvcXUCgRh3LQ6jtB77tgol56tTGohDYHE0gZrxf%2BeqZqKL%2FpWpDZLkDpgj2iFXPkRMBh5QhXDwA3yC%2FY1B2xkb%2FSBmfd3fKmfgD%2B71blMfcgrXPTeMFOBXk49P2uhjJsvLqebHzlnmgnsQUGbNoxA%2FOYiBjNlkY4Xkz7My2UwKmPLbTTRdKefxHtiito0dlyXecyuFjlW38Ke0hOHwYFXdLZEUh%2Fai1L4MxAiCbhc0Q6VTcUQaeXhE4TJuSAW5YTNeUudBeTSQ%3D%3D&response-content-type=text%2Fhtml;#overview