Challenge: Amadey Lab

Platform: CyberDefenders

Category: Endpoint Forensics

Difficulty: Easy

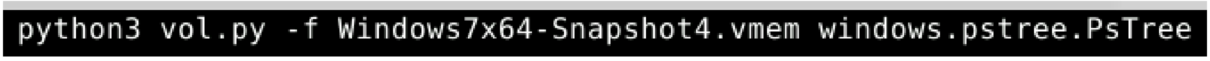
Tools Used: Volatility 3

Summary: This lab involved investigating a memory dump from a compromised Windows 7 machine. It required the use of Volatility 3 and a series of plugins. Personally, I absolutely love memory forensics, especially with Volatility 3 or 2. If you enjoy digital forensics, I highly recommend completing this lab.

Scenario: An after-hours alert from the Endpoint Detection and Response (EDR) system flags suspicious activity on a Windows workstation. The flagged malware aligns with the Amadey Trojan Stealer. Your job is to analyse the presented memory dump and create a detailed report for actions taken by the malware.

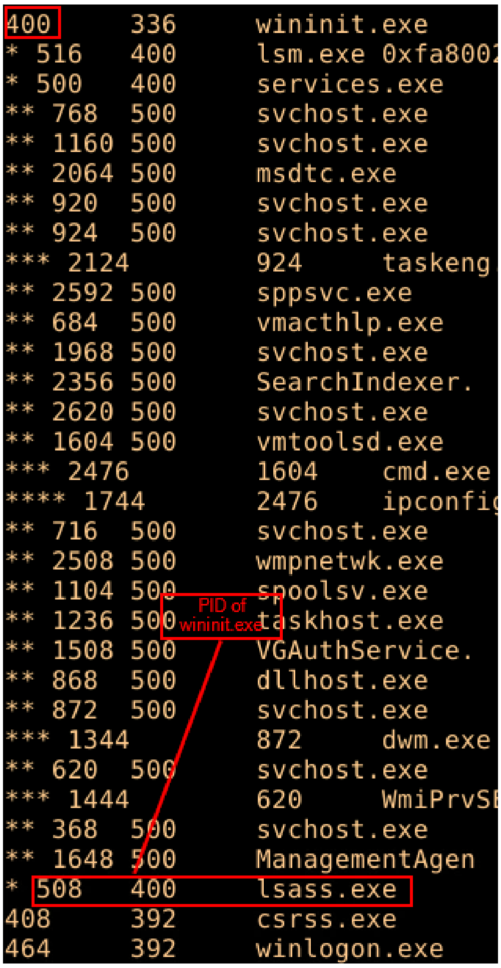
In the memory dump analysis, determining the root of the malicious activity is essential for comprehending the extent of the intrusion. What is the name of the parent process that triggered this malicious behavior?

A great starting point is to list all the running processes at the time of the memory capture, this can be achieved by using the pstree plugin:



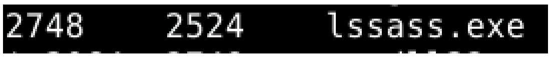
```
python3 vol.py -f Windows7x64-Snapshot4.vmem windows.pstree.PsTree
```

After looking through the output, I noticed a process for lssass.exe. The correct lsass process is not spelt with lssass, furthermore, the legitimate parent process of lsass.exe should be wininit.exe. In this case, the legitimate lsass process has a PPID (Parent Process ID) of 400, which is the PID of wininit.exe:



```
400      336      wininit.exe
* 516    400      lsm.exe 0xfa8002
* 500    400      services.exe
** 768   500      svchost.exe
** 1160  500      svchost.exe
** 2064  500      msdtc.exe
** 920   500      svchost.exe
** 924   500      svchost.exe
*** 2124          924      taskeng
** 2592  500      sppsvc.exe
** 684   500      vmacthlp.exe
** 1968  500      svchost.exe
** 2356  500      SearchIndexer.
** 2620  500      svchost.exe
** 1604  500      vmtoolsd.exe
*** 2476          1604     cmd.exe
**** 1744         2476     ipconfig
** 716   500      svchost.exe
** 2508  500      wmpnetwk.exe
** 1104  500      spoolsv.exe
** 1236  500      taskhost.exe
** 1508  500      VGAuthService.
** 868   500      dllhost.exe
** 872   500      svchost.exe
*** 1344          872      dwm.exe
** 620   500      svchost.exe
*** 1444          620      WmiPrvSE
** 368   500      svchost.exe
** 1648  500      ManagementAgen
* 508    400      lsass.exe
408      392      csrss.exe
464      392      winlogon.exe
```

The process attempting to spoof lsass.exe has a PPID of 2524, which is not wininit.exe.

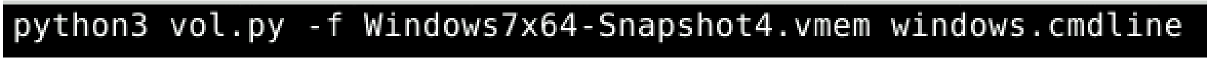


```
2748     2524     lssass.exe
```

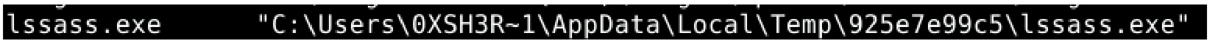
Answer: lssass.exe

Once the rogue process is identified, its exact location on the device can reveal more about its nature and source. Where is this process housed on the workstation?

In order to determine the source location of the lssass.exe process, we can use the cmdline plugin:



```
python3 vol.py -f Windows7x64-Snapshot4.vmem windows.cmdline
```



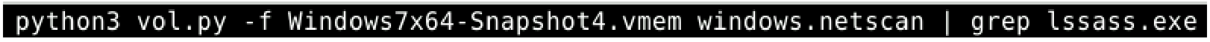
```
lssass.exe      "C:\Users\0XSH3R~1\AppData\Local\Temp\925e7e99c5\lssass.exe"
```

It being executed from the Temp directory is extremely suspicious as well.

Answer: C:\Users\0XSH3R~1\AppData\Local\Temp\925e7e99c5\lssass.exe

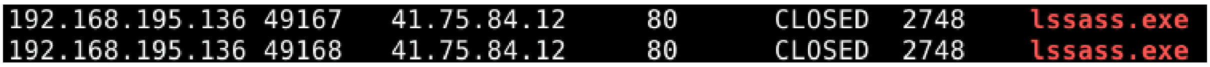
Persistent external communications suggest the malware's attempts to reach out C2C server. Can you identify the Command and Control (C2C) server IP that the process interacts with?

To identify network objects present in the memory dump, you can use the netscan plugin:



```
python3 vol.py -f Windows7x64-Snapshot4.vmem windows.netscan | grep lssass.exe
```

Here we can see that there were two closed connections made to 41.75.84.12 over port 80:

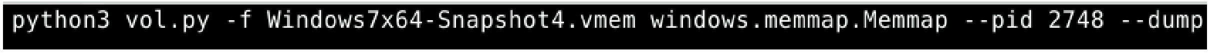


```
192.168.195.136 49167    41.75.84.12     80      CLOSED   2748     lssass.exe
192.168.195.136 49168    41.75.84.12     80      CLOSED   2748     lssass.exe
```

Answer: 41.75.84.12

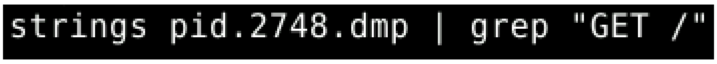
Following the malware link with the C2C, the malware is likely fetching additional tools or modules. How many distinct files is it trying to bring onto the compromised workstation?

To identify any requests made by the malicious process, we first need to dump the memory associated with the process, and look in the output for any GET requests:

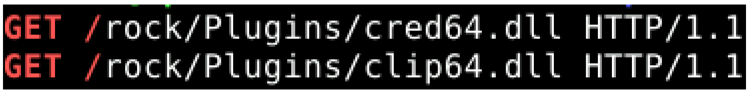


```
python3 vol.py -f Windows7x64-Snapshot4.vmem windows.memmap.Memmap --pid 2748 --dump
```

You can then run the strings command to look through the memory of the malicious process:



```
strings pid.2748.dmp | grep "GET /"
```



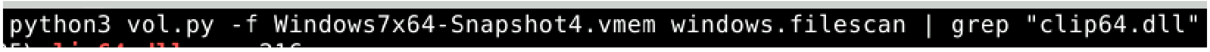
```
GET /rock/Plugins/cred64.dll HTTP/1.1
GET /rock/Plugins/clip64.dll HTTP/1.1
```

As you can see, the process made two GET requests to retrieve cred64.dll and clip64.dll.

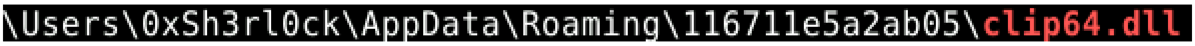
Answer: 2

Identifying the storage points of these additional components is critical for containment and cleanup. What is the full path of the file downloaded and used by the malware in its malicious activity?

To look for the path of the file downloaded and used by the malware, we can utilise the filescan plugin:



```
python3 vol.py -f Windows7x64-Snapshot4.vmem windows.filescan | grep "clip64.dll"
```



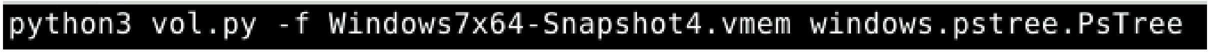
```
\Users\0xSh3rl0ck\AppData\Roaming\116711e5a2ab05\clip64.dll
```

Note! I tried to find the cred64.dll file, but it wasn't present on the file system at the time the memory dump occurred.

Answer: C:\Users\0xSh3rl0ck\AppData\Roaming\116711e5a2ab05\clip64.dll

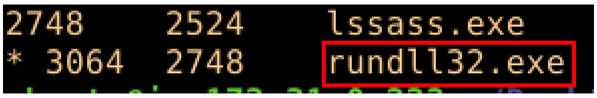
Once retrieved, the malware aims to activate its additional components. Which child process is initiated by the malware to execute these files?

Like in question 1, we can use the pstree command to list all running processes and their parent-child relationships.



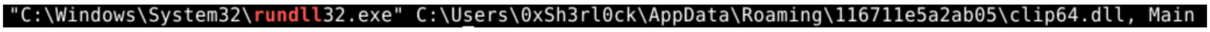
```
python3 vol.py -f Windows7x64-Snapshot4.vmem windows.pstree.PsTree
```

From the output, we can see that the malicious lssass.exe process spawned rundll32.exe.



```
2748    2524    lssass.exe
* 3064  2748    rundll32.exe
```

rundll32.exe is often used by malware to execute code. If you use the cmdline plugin, we can see that rundll32.exe was used to execute the Main entry point for clip64.dll.



```
"C:\Windows\System32\rundll32.exe" C:\Users\0xSh3rl0ck\AppData\Roaming\116711e5a2ab05\clip64.dll, Main
```

Read here for more information.

Answer: rundll32.exe

Understanding the full range of Amadey's persistence mechanisms can help in an effective mitigation. Apart from the locations already spotlighted, where else might the malware be ensuring its consistent presence?

Unfortunately the lab machine turned off before I could take screenshots, however, if you used the filescan plugin and piped the output to grep “lssass.exe”, you would see a location for a scheduled task that points to lssass.exe.

Answer: \Windows\System32\Tasks\lssass.exe