

MemLabs Lab 0 - Never Too Late Mister

I. Mô tả thử thách:

- John – là một "nhà hoạt động vì môi trường" và là một người nhân đạo. Anh ta ghét hệ tư tưởng của Thanos trong phim Avengers: Infinity War. John lập trình rất tệ – anh ta dùng quá nhiều biến trong chương trình của mình. Một ngày nọ, John đưa cho mình một bản memory dump và nhờ mình tìm hiểu xem anh ta đã làm gì khi chụp bộ nhớ. Bạn có thể tìm ra điều đó giúp mình không?

File thử thách: (được cung cấp qua Google Drive)

Challenge file: [Google Drive](#)

☞ Những manh mối có thể rút ra từ phần mô tả:

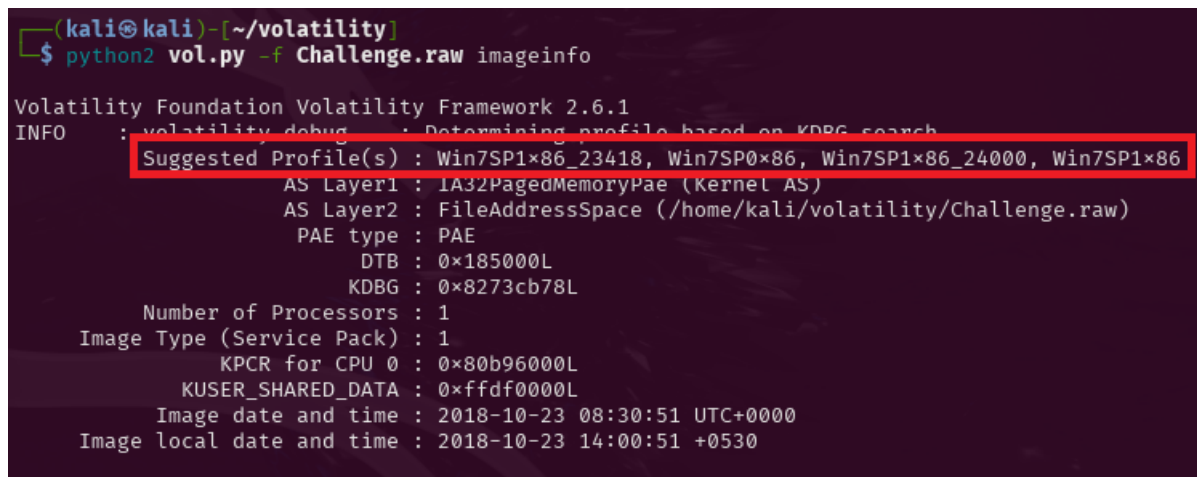
- Nhà hoạt động vì môi trường – có dấu ngoặc kép => có thể là một ám chỉ quan trọng.
- John ghét Thanos – nghe có vẻ vô dụng, nhưng biết đâu sẽ cần.
- John lập trình rất tệ và sử dụng quá nhiều biến – điều này có thể liên quan đến cách viết script hoặc chương trình mà anh ấy đã chạy trong RAM.

II. Memory Dump Analysis

1. Xác định profile của file **Challenge.raw**:

- Việc đầu tiên khi forensic analysis một **memory dump** là xác định **profile**.
- Profile cho chúng ta biết hệ điều hành của hệ thống hoặc máy tính từ **memory dump**. Volatility (tool) có một **plugin** tích hợp để giúp chúng ta xác định profile của **memory dump**:
- Sử dụng plugin `imageinfo`:

```
$ python2 vol.py -f Challenge.raw imageinfo
```



```
(kali㉿kali)-[~/volatility]
$ python2 vol.py -f Challenge.raw imageinfo

Volatility Foundation Volatility Framework 2.6.1
INFO : volatility.debug : Determining profile based on KDBG search
Suggested Profile(s) : Win7SP1x86_23418, Win7SP0x86, Win7SP1x86_24000, Win7SP1x86
AS Layer1 : IA32PagedMemoryPae (Kernel AS)
AS Layer2 : FileAddressSpace (/home/kali/volatility/Challenge.raw)
PAE type : PAE
DTB : 0x185000L
KDBG : 0x8273cb78L
Number of Processors : 1
Image Type (Service Pack) : 1
KPCR for CPU 0 : 0x80b96000L
KUSER_SHARED_DATA : 0xffdf0000L
Image date and time : 2018-10-23 08:30:51 UTC+0000
Image local date and time : 2018-10-23 14:00:51 +0530
```

2. Phân tích hệ thống mục tiêu thông qua **memory dump** từ file **Challenge.raw**:

Một trong những điều quan trọng nhất mà chúng ta cần biết từ một hệ thống trong quá trình phân tích là:

- Các tiến trình hoạt động
- Các commands được thực thi trong shell / terminal / Command prompt
- Tiến trình ẩn (nếu có) hoặc tiến trình đã thoát
- Lịch sử trình duyệt (Điều này phụ thuộc rất nhiều đối với tình huống liên quan)

Bây giờ, để liệt kê các quy trình đang hoạt động hoặc đang chạy, chúng ta sử dụng sự trợ giúp của plugin `pslist`:

```
$ python2 vol.py -f Challenge.raw --profile=Win7SP1x86 pslist
```

```
(kali@kali)~[/volatility]
$ python2 vol.py -f Challenge.raw --profile=Win7SP1x86 pslist
```

Offset(V)	Name	PID	PPID	Thds	Hnds	Sess	Wow64	Start	Exit
0x83d09c58	System	4	0	85	483		0	2018-10-23 08:29:16 UTC+0000	
0x8437db18	smss.exe	260	4	2	29		0	2018-10-23 08:29:16 UTC+0000	
0x84d69030	csrss.exe	340	332	8	347	0	0	2018-10-23 08:29:21 UTC+0000	
0x84d8d030	csrss.exe	380	372	9	188	1	0	2018-10-23 08:29:23 UTC+0000	
0x84d93c68	wininit.exe	388	332	3	79	0	0	2018-10-23 08:29:23 UTC+0000	
0x84dcbd20	winlogon.exe	424	372	6	117	1	0	2018-10-23 08:29:23 UTC+0000	
0x84deb2d0	services.exe	484	388	10	191	0	0	2018-10-23 08:29:25 UTC+0000	
0x84def3d8	lsass.exe	492	388	7	480	0	0	2018-10-23 08:29:25 UTC+0000	
0x84df2378	lsass.exe	500	388	10	146	0	0	2018-10-23 08:29:25 UTC+0000	
0x84e23030	svchost.exe	592	484	12	358	0	0	2018-10-23 08:29:30 UTC+0000	
0x84e41708	VBoxService.exe	652	484	12	116	0	0	2018-10-23 08:29:31 UTC+0000	
0x84e54030	svchost.exe	716	484	9	243	0	0	2018-10-23 08:29:32 UTC+0000	
0x84e7ad20	svchost.exe	804	484	19	378	0	0	2018-10-23 08:29:32 UTC+0000	
0x84e84898	svchost.exe	848	484	20	400	0	0	2018-10-23 08:29:33 UTC+0000	
0x84e89c68	svchost.exe	872	484	19	342	0	0	2018-10-23 08:29:33 UTC+0000	
0x84e8c648	svchost.exe	896	484	30	809	0	0	2018-10-23 08:29:33 UTC+0000	
0x84ea7d20	audiodg.exe	988	804	6	127	0	0	2018-10-23 08:29:35 UTC+0000	
0x84f033c8	svchost.exe	1192	484	15	365	0	0	2018-10-23 08:29:40 UTC+0000	
0x84f323f8	spoolsv.exe	1336	484	16	295	0	0	2018-10-23 08:29:43 UTC+0000	
0x84f4dca0	svchost.exe	1364	484	19	307	0	0	2018-10-23 08:29:43 UTC+0000	
0x84f7d578	svchost.exe	1460	484	11	148	0	0	2018-10-23 08:29:44 UTC+0000	
0x84f828f8	svchost.exe	1488	484	8	170	0	0	2018-10-23 08:29:44 UTC+0000	
0x850b2538	taskhost.exe	308	484	8	151	1	0	2018-10-23 08:29:55 UTC+0000	
0x850d0030	sppsvc.exe	1164	484	6	154	0	0	2018-10-23 08:29:57 UTC+0000	
0x85109030	dwm.exe	1992	848	5	132	1	0	2018-10-23 08:30:04 UTC+0000	
0x85097870	explorer.exe	324	1876	33	827	1	0	2018-10-23 08:30:04 UTC+0000	
0x85135af8	VBoxTray.exe	1000	324	14	159	1	0	2018-10-23 08:30:08 UTC+0000	
0x85164030	SearchIndexer.exe	2032	484	14	614	0	0	2018-10-23 08:30:14 UTC+0000	
0x8515ad20	SearchProtocolHost.exe	284	2032	7	235	0	0	2018-10-23 08:30:16 UTC+0000	
0x8515cd20	SearchFilterHost.exe	1292	2032	5	80	0	0	2018-10-23 08:30:17 UTC+0000	
0x851a6610	cmd.exe	2096	324	1	22	1	0	2018-10-23 08:30:18 UTC+0000	
0x851a5cd8	conhost.exe	2104	380	2	52	1	0	2018-10-23 08:30:18 UTC+0000	
0x845a8d20	DumpIt.exe	2412	324	2	38	1	0	2018-10-23 08:30:48 UTC+0000	
0x84d83d20	conhost.exe	2424	380	2	51	1	0	2018-10-23 08:30:48 UTC+0000	

- Thực hiện lệnh này cung cấp cho chúng ta một danh sách các quá trình đang chạy khi **memory dump** được thực hiện. Đầu ra của lệnh cung cấp các thông tin gồm tên, PID, PPID, Luồng, Xử lý, thời gian bắt đầu, v.v.

Quan sát kỹ, chúng tôi nhận thấy một số tiến trình cần chú ý:

- cmd.exe
- DumpIt.exe
- explorer.exe

0x85097870	explorer.exe	324	1876	33	827	1	0	2018-10-23 08:30:04 UTC+0000
0x85135af8	VBoxTray.exe	1000	324	14	159	1	0	2018-10-23 08:30:08 UTC+0000
0x85164030	SearchIndexer.exe	2032	484	14	614	0	0	2018-10-23 08:30:14 UTC+0000
0x8515ad20	SearchProtocolHost.exe	284	2032	7	235	0	0	2018-10-23 08:30:16 UTC+0000
0x8515cd20	SearchFilterHost.exe	1292	2032	5	80	0	0	2018-10-23 08:30:17 UTC+0000
0x851a6610	cmd.exe	2096	324	1	22	1	0	2018-10-23 08:30:18 UTC+0000
0x851a5cd8	conhost.exe	2104	380	2	52	1	0	2018-10-23 08:30:18 UTC+0000
0x845a8d20	DumpIt.exe	2412	324	2	38	1	0	2018-10-23 08:30:48 UTC+0000
0x84d83d20	conhost.exe	2424	380	2	51	1	0	2018-10-23 08:30:48 UTC+0000

- cmd.exe:** Đây là tiến trình chịu trách nhiệm cho command prompt. Việc trích xuất nội dung từ tiến trình này có thể cung cấp cho chúng ta thông tin chi tiết về những lệnh đã được thực thi trong hệ thống.
- DumpIt.exe:** Tiến trình này đã được ta sử dụng để có kết xuất được bộ nhớ của hệ thống.
- Explorer.exe:** Tiến trình này xử lý File Explorer.

Ta đã thấy rằng cmd.exe đang chạy, hãy thử xem liệu có bất kỳ lệnh nào được thực thi và nội dung được in ra trong shell/terminal hay không.

- Đối với điều này, ta sử dụng plugin `consoles`

```
$ python2 vol.py -f Challenge.raw --profile=Win7SP1x86 consoles
```

```

(kali@kali)-[~/volatility]
$ python2 vol.py -f Challenge.raw --profile=Win7SP1x86 consoles
Volatility Foundation Volatility Framework 2.6.1
*****
ConsoleProcess: conhost.exe Pid: 2104
Console: 0xe981c0 CommandHistorySize: 50
HistoryBufferCount: 2 HistoryBufferMax: 4
OriginalTitle: %SystemRoot%\system32\cmd.exe
Title: C:\Windows\system32\cmd.exe
AttachedProcess: cmd.exe Pid: 2096 Handle: 0x5c

CommandHistory: 0x300690 Application: python.exe Flags:
CommandCount: 0 LastAdded: -1 LastDisplayed: -1
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0x0

CommandHistory: 0x300498 Application: cmd.exe Flags: Allocated, Reset
CommandCount: 1 LastAdded: 0 LastDisplayed: 0
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0x5c
Cmd #0 at 0x2f43c0: C:\Python27\python.exe C:\Users\hello\Desktop\demon.py.txt

Screen 0x2e6368 X:80 Y:300
Dump:
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\hello>C:\Python27\python.exe C:\Users\hello\Desktop\demon.py.txt
335d366f5d6031767631707f

```

- Từ hình trên, một tệp python đã được thực thi. Lệnh được thực hiện là C:\Python27\python.exe C:\Users\hello\Desktop\demon.py.txt và nội dung trả về là một chuỗi nhất định đã được viết ra stdout. Bây giờ ta có thể thấy đây là một chuỗi được mã hóa hex 335d366f5d6031767631707f. Một khi chúng tôi cố gắng decode hex, ta nhận được một văn bản vô nghĩa. 3[60]`1vv1p\x7f`

```

data = bytes.fromhex("335d366f5d6031767631707f")
print(data)

```

```
output: b'3[60]`1vv1p\x7f`'
```

- Bây giờ ta hãy nhớ lại một số manh mối từ mô tả thử thách. Cái đầu tiên là từ **Environmental**. Có một số biến do hệ thống xác định được gọi là **biến môi trường**.
- Để xem các biến môi trường trong một hệ thống, hãy sử dụng plugin `envvars`. Kéo xuống ở đầu ra, chúng ta thấy một biến lạ có tên **Thanos** (Ah! vì vậy có lẽ đó là lý do tại sao nó được cung cấp trong mô tả.), giá trị của biến là **xor and password**

```
$ python2 vol.py -f Challenge.raw --profile=Win7SP1x86 envvars
```

```

(kali@kali)-[~/volatility]
$ python2 vol.py -f Challenge.raw --profile=Win7SP1x86 envvars
Volatility Foundation Volatility Framework 2.6.1

```

Pid	Process	Block	Variable	Value
260	smss.exe	0x001707f0	Path	C:\Windows\System32
260	smss.exe	0x001707f0	SystemDrive	C:
260	smss.exe	0x001707f0	SystemRoot	C:\Windows
340	csrss.exe	0x003807f0	ComSpec	C:\Windows\system32\cmd.exe
340	csrss.exe	0x003807f0	FP_NO_HOST_CHECK	NO
340	csrss.exe	0x003807f0	NUMBER_OF_PROCESSORS	1
340	csrss.exe	0x003807f0	OS	Windows_NT
340	csrss.exe	0x003807f0	Path	C:\Windows\system32;C:\Windows;C:\Windows\System32\Wb
340	csrss.exe	0x003807f0	PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
340	csrss.exe	0x003807f0	PROCESSOR_ARCHITECTURE	x86
340	csrss.exe	0x003807f0	PROCESSOR_IDENTIFIER	x86 Family 6 Model 142 Stepping 9, GenuineIntel
340	csrss.exe	0x003807f0	PROCESSOR_LEVEL	6
340	csrss.exe	0x003807f0	PROCESSOR_REVISION	8e09
340	csrss.exe	0x003807f0	PSModulePath	C:\Windows\system32\WindowsPowerShell\v1.0\Modules\
340	csrss.exe	0x003807f0	SystemDrive	C:
340	csrss.exe	0x003807f0	SystemRoot	C:\Windows
340	csrss.exe	0x003807f0	TEMP	C:\Windows\TEMP
340	csrss.exe	0x003807f0	Thanos	xor and password
340	csrss.exe	0x003807f0	TMP	C:\Windows\TEMP
340	csrss.exe	0x003807f0	USERNAME	SYSTEM
340	csrss.exe	0x003807f0	windir	C:\Windows
340	csrss.exe	0x003807f0	windows_tracing_flags	3
340	csrss.exe	0x003807f0	windows_tracing_logfile	C:\BVTBin\Tests\installpackage\csilogfile.log
380	csrss.exe	0x002307f0	ComSpec	C:\Windows\system32\cmd.exe
380	csrss.exe	0x002307f0	FP_NO_HOST_CHECK	NO
380	csrss.exe	0x002307f0	NUMBER_OF_PROCESSORS	1
380	csrss.exe	0x002307f0	OS	Windows_NT
380	csrss.exe	0x002307f0	Path	C:\Windows\system32;C:\Windows;C:\Windows\System32\Wb

Bây giờ, chúng ta có tổng cộng 3 điều:

- Văn bản vô nghĩa là kết quả của việc decode bằng hex
- xor
- password

Hãy thử thử giải mã xor trên văn bản vô nghĩa 335d366f5d6031767631707f

```
data = bytes.fromhex("335d366f5d6031767631707f")

for k in range(1, 256):
    s = ''.join(chr(b ^ k) for b in data)
    if s.isprintable():
        print(f"[Key {k:3}] => {s}")
```

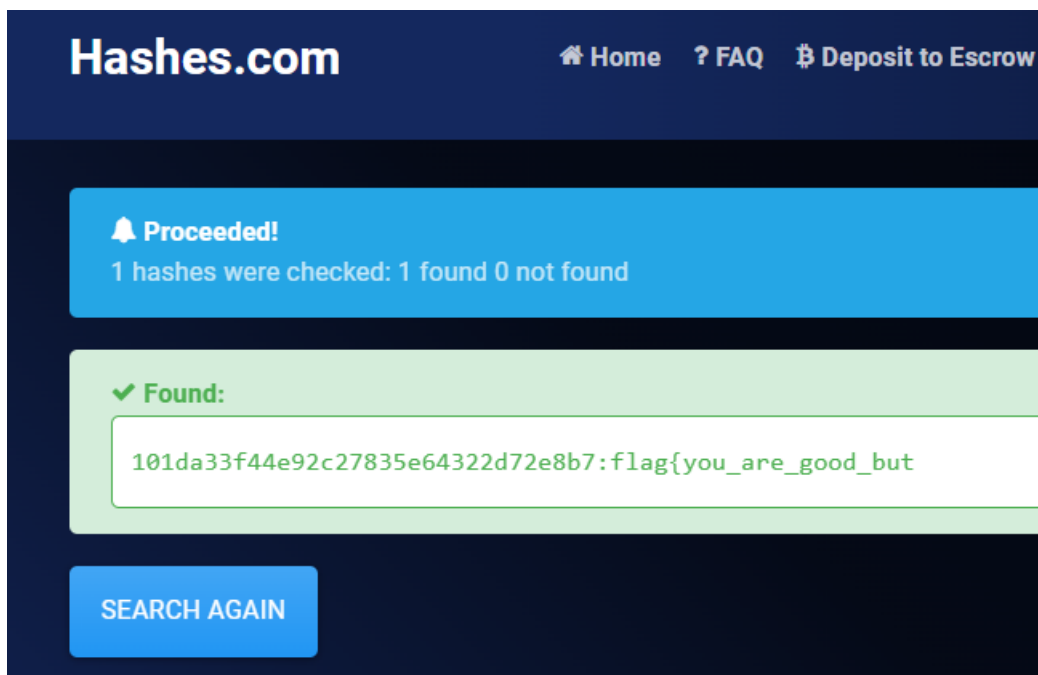
output: [Key 2] => 1_4m_b3tt3r}

- Có tổng cộng 255 khả năng (tương ứng với 255 giá trị khóa XOR khác nhau), và nếu để ý, kết quả thứ 2 hiển thị một chuỗi đáng ngờ: **1_4m_b3tt3r}** – Trông giống như phần sau của flag.
- Tiếp theo là "password" (mật khẩu). Sử dụng Volatility, chúng ta có thể trích xuất các hash mật khẩu NTLM (Windows) bằng plugin hashdump:

```
$ python2 vol.py -f Challenge.raw --profile=Win7SP1x86 hashdump
```

```
(kali㉿kali)-[~/volatility]
$ python2 vol.py -f Challenge.raw --profile=Win7SP1x86 hashdump mimikatz
Volatility Foundation Volatility Framework 2.6.1
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
hello:1000:aad3b435b51404eeaad3b435b51404ee:101da33f44e92c27835e64322d72e8b7:::
```

- Lệnh này sẽ hiển thị danh sách các username và password hash có trong bộ nhớ RAM lúc dump. Trong số đó, hash mà chúng ta quan tâm là: 101da33f44e92c27835e64322d72e8b7
- Giờ bạn có thể dùng các website bẻ khóa hash NTLM online (như Hashes.com...) để giải mã hash này thành chuỗi văn bản gốc (mật khẩu rõ ràng).
- Link: hashes.com



Kết quả sau khi giải mã:

- Hash đó khi giải được cho ra phần đầu của flag: **flag{you_are_good_but**
- Ghép hai phần lại:
 - Phần đầu: **flag{you_are_good_but**
 - Phần sau (từ đoạn XOR hex): **1_4m_b3tt3r}**

☞ Kết quả hoàn chỉnh:

☞ FLAG: **flag{you_are_good_but1_4m_b3tt3r}**

☞ Tài nguyên tham khảo: [link](#)