Flexlm Encryption seed recovery technique

1. Flexlm version 7.x-9.x:

- Create a fake license & name it “dummy.dat”
  - SERVER **Computer_Name** ANY
  - VENDOR **Vendor_Name**
  - USE_SERVER
  - INCREMENT test **Vendor_Name Version_Number dd-mmm-yyyy** 1 0123456789AB

- Load vendor daemon into ollydbg (with arguments: -t **computer_name** 4 –c dummy.dat

- Find _l_sg: by finding the seed value (use the “*search for constant*” command)
  - **7648B98E** for flexlm v7.x to v8.C
  - **6F7330B8** for Flexlm v8.D and onwards
  - you will find two references, and only the first one, which looks similar to: C745 F4 B8307>MOV DWORD PTR SS:[EBP-C],6F7330B8, is _l_sg and counts. (The 2nd one: C745 F8 B8307>MOV DWORD PTR SS:[EBP-8],6F7330B8 is _l_vk)

- Locate the call to _l_n36_buff (inside _l_sg)& set breakpoint #1.
  - (This call which is a dword pointer call, can be found @ instruction FF15????????)
  - (FF15 D4794B00 CALL DWORD PTR DS:[4B79D4])

- Set a breakpoint # 2 at the ret of _l_n36_buff

- Run the program & let it break. (@ 1st breakpoint)

- Single step into the _l_n36_buff call (one step only!)

- Locate the **EB05** (v7.x to v8.C) or **EB09** (v8.D & ↑) jmp. (You will find this one just above the vendor name loop inside _l_n36_buff, at the end of multiple calls to _time)
  - (EB 09 JMP SHORT callmd.0040C227)

- Set breakpoint #3, and Run the program & let it break. (at BP#3)

- Check the memory address inside ecx or edx.(follow in dump).One of them will contain the location of the job structure.

- Delete the 16 random bytes inside the job structure, (starting @ job+04 and ending @ job+13), and replace with “00”

- Run the program & let it break at BP#2 (“Break on RET” after returning from the call to _l_n36_buff)

- Now Look at the following stack locations: (follow in dump)
  - ESP+04: Pointer to vendor name (name of vendor daemon)
  - ESP+08: Pointer to vendor code (which now will contain the clean seed 1 and 2)
  - **VC+04 = Seed1**
  - **VC+08 = Seed2**

2. Flexlm version 10.x-11.4:

- Create a fake license & name it “dummy.dat”
  - SERVER **Computer_Name** ANY
  - VENDOR **Vendor_Name**
  - USE_SERVER
  - INCREMENT test **Vendor_Name Version_Number dd-mmm-yyyy** 1 0123456789AB

- Load vendor daemon into ollydbg (with arguments: -t **computer_name** 4 –c dummy.dat

- Find _l_sg: (by finding the seed value 6F7330B8)
  - you will find two references, and only the first one, which looks similar to:
    C745 F4 B8307>MOV DWORD PTR SS:[EBP-C],6F7330B8, is _l_sg and does count.
    (The 2nd one is: C745 F8 B8307>MOV DWORD PTR SS:[EBP-8],6F7330B8, & is _l_vk)

- Locate call to _l_n36_buff (inside _l_sg)& set breakpoint #1.
  - This dword pointer call, can be found @ instruction FF90???????? call dword ptr [EAX+524])
  - (FF90 24050000 CALL DWORD PTR DS:[EAX+524])

- Set a breakpoint # 2 at the ret of _l_n36_buff

- Run the program & let it break. (@ 1st breakpoint)

- Single step into the _l_n36_buff call (one step only!)

- Locate the **EB09** jmp
  (You will find this one just above the vendor name loop inside _l_n36_buff, at the end of multiple calls to _time)

- Set breakpoint #3

- Run the program & let it break. (at BP#3)

- Check the memory address inside ecx or edx.(follow in dump).One of them will contain the location of the job structure. (note that this new Job structure starts with **00** 00 00 00 instead of **66** 00 00 00)

- Delete the 16 random bytes inside the job structure, (starting @ job+04 and ending @ job+13), and replace with “00”

- Run the program & let it break at BP#2 (“Break on RET”, after returning from the call to _l_n36_buff)

- Now Look at the following stack locations: (follow in dump)
  - ESP+04: Pointer to vendor name (name of vendor daemon)
  - ESP+08: Pointer to vendor code (which now will contain the clean seed 1 and 2)
  - **VC+04 = Seed1**
  - **VC+08 = Seed2**