Requirements: OllyDbg

Open the crackme with OllyDbg, hit F8 until you reach the address 0x00401145, which is a anti-debug trick.

The anti-debug trick is based on the TEB (Thread Environment Block), the first instruction will get the address of the PEB, the second will get the address of the PEB (Process Environment Block), and the third will get the property BeingDebugged which tells the crackme that we debugging it and the fourth instruction will compare eax with 1 and that's it, at the address 0x0040114C you need to patch the jump which is a JE, to a JMP or JNE or JZ …

```
lkd> dt _teb
nt!_TEB
   +0x000 NtTib             : _NT_TIB
   +0x01c EnvironmentPointer : Ptr32 Void
   +0x020 ClientId          : _CLIENT_ID
   +0x028 ActiveRpcHandle   : Ptr32 Void
   +0x02c ThreadLocalStoragePointer : Ptr32 Void
   +0x030 ProcessEnvironmentBlock : Ptr32 _PEB


lkd> dt _PEB
nt!_PEB
   +0x000 InheritedAddressSpace : UChar
   +0x001 ReadImageFileExecOptions : UChar
   +0x002 BeingDebugged     : UChar
```

So set a breakpoint at the address 0x004011D4 which is the crackme “Run DBP” button handler and press F9, to enter your username and serial. As usual use “abcde” as your user! :)

“Step out” until you reach the address 0x00401260, this is our partial serial generation procedure, between the addresses 0x00401B7A and 0x00401C12, the crackme will retrieve the first character of our username and then will multiply by it many times as the username length defines, plus adding the current index of the iteration. This partial serial will be stored at the address 0x0040B378.

So in python is something like this...

```
username = "abcde"

i = 2
serial = ord(username[0])

while i < (len(username) + 1):
    serial += i
    serial *= ord(username[0])
    i += 1

serial = serial & 0xFFFFFFFFFFFFFFFF
```

"Step out" until you reach the address 0x00401C18, this is the second part of the algorithm which uses the FPU (Floating Point Unit), MMX, which is composed by 8 registers of 64 bits each, in this case the crackme will only use mm0 and mm1.

```
00401C18  >  0FEFC0        PXOR MM0,MM0
00401C1B  .  68 0000F642   PUSH 42F60000
00401C20  .  E8 AB100000   CALL 00402CD0
00401C25  .  DB5C24 04     FISTP DWORD PTR SS:[LOCAL.1]
00401C29  .  0F6F05 78B34( MOVQ MM0,QWORD PTR DS:[40B378]
00401C30  .  D905 F8A3400( FLD DWORD PTR DS:[40A3F8]
00401C36  .  D9FF          FCOS
00401C38  .  DB5C24 04     FISTP DWORD PTR SS:[LOCAL.1]
00401C3C  .  0F6F0D 78B34( MOVQ MM1,QWORD PTR DS:[40B378]
00401C43  .  D905 FCA3400( FLD DWORD PTR DS:[40A3FC]
00401C49  .  D9FA          FSQRT
00401C4B  .  DB5C24 04     FISTP DWORD PTR SS:[LOCAL.1]
00401C4F  .  0F71D0 08     PSRLW MM0,8
00401C53  .  D905 00A4400( FLD DWORD PTR DS:[40A400]
00401C59  .  D9FA          FSQRT
00401C5B  .  DB5C24 04     FISTP DWORD PTR SS:[LOCAL.1]
00401C5F  .  0F73F0 0C     PSLLQ MM0,0C
00401C63  .  D905 FCA3400( FLD DWORD PTR DS:[40A3FC]
00401C69  .  D9FA          FSQRT
00401C6B  .  DC05 1CA4400( FADD QWORD PTR DS:[40A41C]
00401C71  .  DB5C24 04     FISTP DWORD PTR SS:[LOCAL.1]
00401C75  .  0FF8C1        PSUBB MM0,MM1
00401C78  .  D905 04A4400( FLD DWORD PTR DS:[40A404]
00401C7E  .  D9FA          FSQRT
00401C80  .  DC05 1CA4400( FADD QWORD PTR DS:[40A41C]
00401C86  .  DB5C24 04     FISTP DWORD PTR SS:[LOCAL.1]
00401C8A  .  0F7F05 78B34( MOVQ QWORD PTR DS:[40B378],MM0
```

So, in the further enumeration will explain the basic process, of the second part of the algorithm.

1. Has I said before, the partial serial is stored at the 0x0040B378, and will be moved to the registers mm0 and mm1, at the addresses 0x00401C29 and 0x00401C3C respectively. If your username is "abcde" 0x0040B378 stores the value "0x000000020A90792F", and before you reach the address 0x00401C43, mm0 and mm1 will have the same value too.

2. At the address 0x00401C4F, you will find the following instruction "PSRLW" which stands for "Packed Shift Right Logical" and it ends with a W which means a "word", so the 4 words that constitute a 64 bits value will be shifted to the right by a factor of 8.

   0x0000 0002 0A90 792F → 0x0000 0000 000A 0079 : mm0

3. At the address 0x00401C4F, you will find the following instruction "PSLLQ" which stands for "Packed Shift Left Logical" and it ends with a Q which means a "Quadword", so all the 64 bits will be shifted to the left by a factor of 0x0C.

   0x00000000000A0079 → 0x00000000A0079000 : mm0

4. To finalize the mm0 will subtracted by mm1 at the address 0x00401C75, "PSUBB" subtracted by byte which means...

```
MM1=00,00,00,02,0A,90,79,2F
MM0=00,00,00,00,A0,07,90,00
```

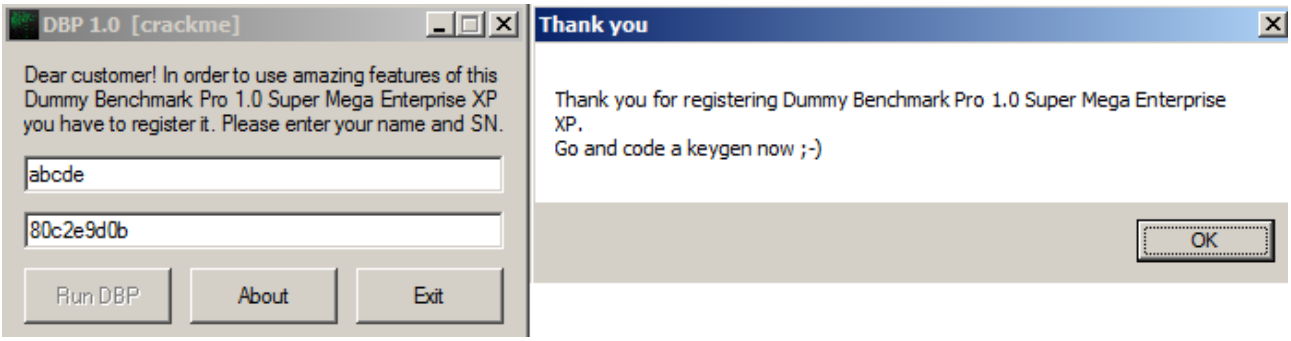
   0x00000000A0079000 → 0x000000FE967717D1 : mm0

So, this is the first part of the serial generation, lets go to the second and final algorithm. So return to the main procedure at the address 0x00401278 out serial will be converted to a string and stored at the address 0x0040B3A0. "Step out" until you reach the address 0x004012C2 our serial "FE967717D1" will be hashed by a SHA1, "8da6de5b07bb575332e4dd3c33423f380c2e9d0b".

And to finalize at the address 0x004012D4, the crackme will retrive the last 9 characters from our serial.

"8da6de5b07bb575332e4dd3c33423f380c2e9d0b" → "80c2e9d0b"

That's all!



Go see the keygen source!