FormBook - Hiding in plain sight

usualsuspect.re/article/formbook-hiding-in-plain-sight

- Articles
- About
- Legal Notice

<< back to Articles

published 2019-05-02

Intro

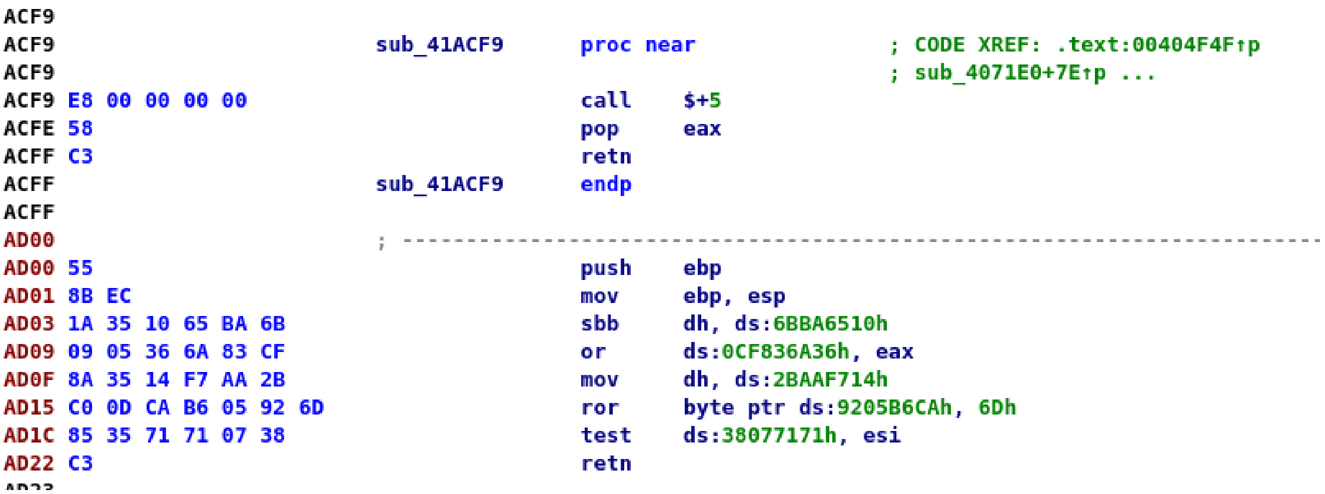
Today I came across the FormBook malware and had a quick look. As is tradition for malware, it uses encrypted strings to make it harder to detect. The way FormBook stores its encrypted strings however is new to me.

FormBook itself is not new and the decryption has been documented before and decryption scripts for its encrypted strings exist.

Arbor for example has an article here, and the decryption script which is basically a Python conversion of the code in FormBook itself can be found here.

Under the hood

So what does it look like in the binary?



```
ACF9
ACF9                        sub_41ACF9      proc near               ; CODE XREF: .text:00404F4F↑p
ACF9                                                                ; sub_4071E0+7E↑p ...
ACF9 E8 00 00 00 00                         call    $+5
ACFE 58                                     pop     eax
ACFF C3                                     retn
ACFF                        sub_41ACF9      endp
ACFF
AD00                        ; ---------------------------------------------------------------------------
AD00 55                                     push    ebp
AD01 8B EC                                  mov     ebp, esp
AD03 1A 35 10 65 BA 6B                      sbb     dh, ds:6BBA6510h
AD09 09 05 36 6A 83 CF                      or      ds:0CF836A36h, eax
AD0F 8A 35 14 F7 AA 2B                      mov     dh, ds:2BAAF714h
AD15 C0 0D CA B6 05 92 6D                   ror     byte ptr ds:9205B6CAh, 6Dh
AD1C 85 35 71 71 07 38                      test    ds:38077171h, esi
AD22 C3                                     retn
AD23
```

What you see here is an encrypted string blob. The lower part starting with `push ebp` is the actual string, while the top part is used to fetch its address.

Any string FormBook uses is stored in gibberish code blocks like above.

I've analyzed the function processing those code blocks but couldn't identify it at first. I expected some decompression/decryption but it looked like nothing I've seen before and it also seemed rather simple (just copying memory around basically). The existing script is also just a 1:1 translation of the malware's code into Python, so that doesn't solve the riddle either.

Then I had an idea - could it be that the processing code is actually a small disassembly engine? Turns out the answer is yes!

What FormBook *actually* is doing is disassembling these gibberish code blocks and extracts operands, which make up the "hidden" encrypted blob. So the weird transformation functions are part of the disassembly engine.

To confirm that, I ripped the above blob into a file and ran it through Arbor's script and printed the data before it is being decrypted using RC4, and the output was:

```
1065ba6b366a83cf14f7aa2bcab6059271710738
```

That is, these bytes are what are hidden in the code blob. And if you look closely, you can see the bytes in the above disassembly as operands but reversed (little-endian of DWORDs):

```
1065ba6b -> 0x6BBA6510
366a83cf -> 0xCF836A36
14f7aa2b -> 0x2BAAF714
cab60592 -> 0x9205B6CA
71710738 -> 0x38077171
```

So FormBook is literally hiding the encrypted strings in plain sight - as operands for nonsense code blobs and uses a small disassembly engine to retrieve them. Haven't seen this one before, that's a rather creative approach.

(I couldn't find anyone else noticing what's actually going on, so if I missed it, apologies, let me know)