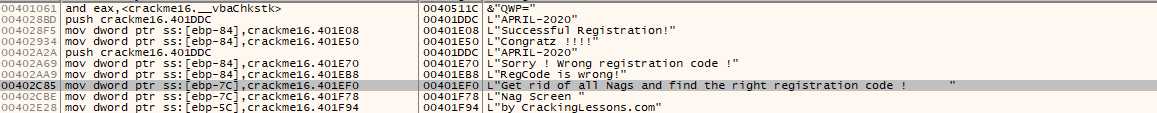
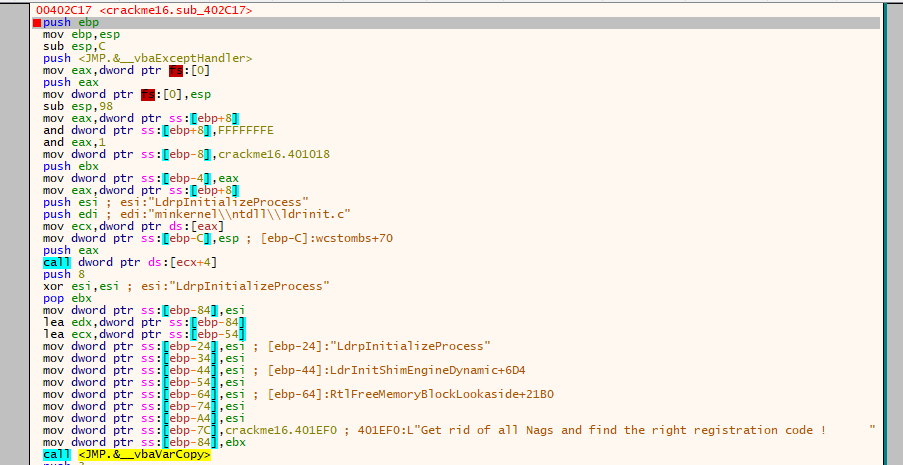


1. Remove First Nag

Find all reference strings



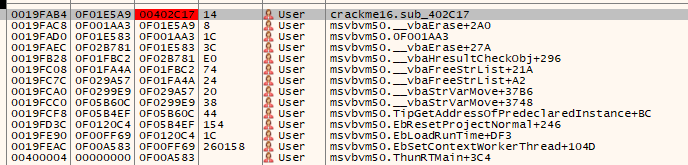
We found where Nag Screen located



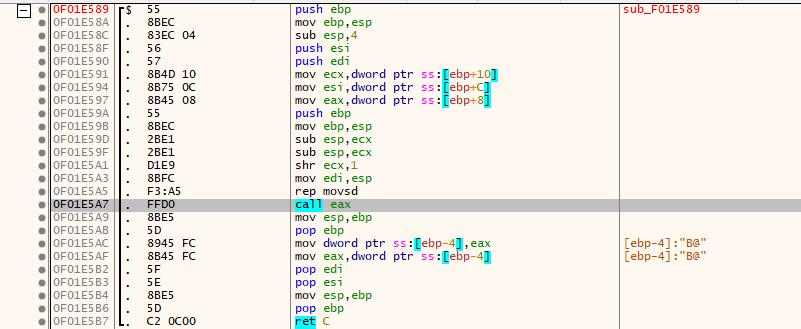
We found that there are no jump instructions to remove the nag screen. So we find where the function is called , and remove that call.

In CALL STACK window give use that information

Set a breakpoint at the function, and run program until breakpoint, In CALL STACK window, we see that



follow the nearest call, we jump to there

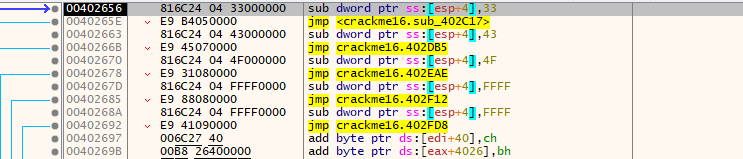


This call eax call out function, so set a breakpoint there. and what does this call eax actually do? Restart crackme, until it breaks the breakpoint.



we see that, eax is address crackme16.00402656

F7 to follow the call

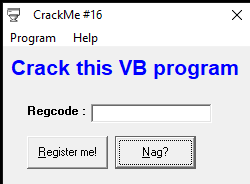


We see a list of jumps and the next instruction is the jump to the function sub\_402C17 which calls the nag screen. we cannot nop that jump, because it breaks the stack, and calling another function, breaks the following of the function, because, instruction before is called eax, so we can put a ret instruction to save the working flow of crackme.

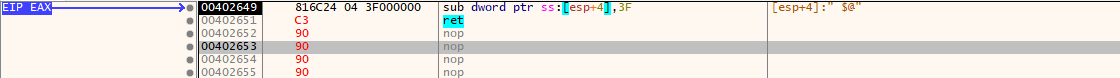
Test it, but a breakpoint at return address of call eax, continue program, no nag at first more.

1. Remove the last nag.

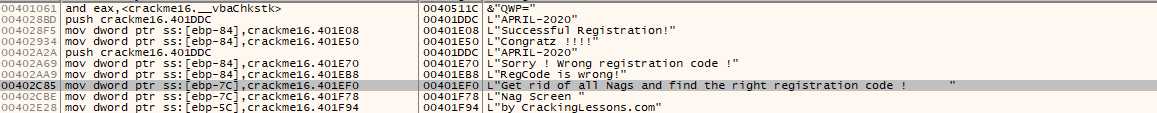
Set breakpoint again at the function which calls the nag (sub\_402C17 )



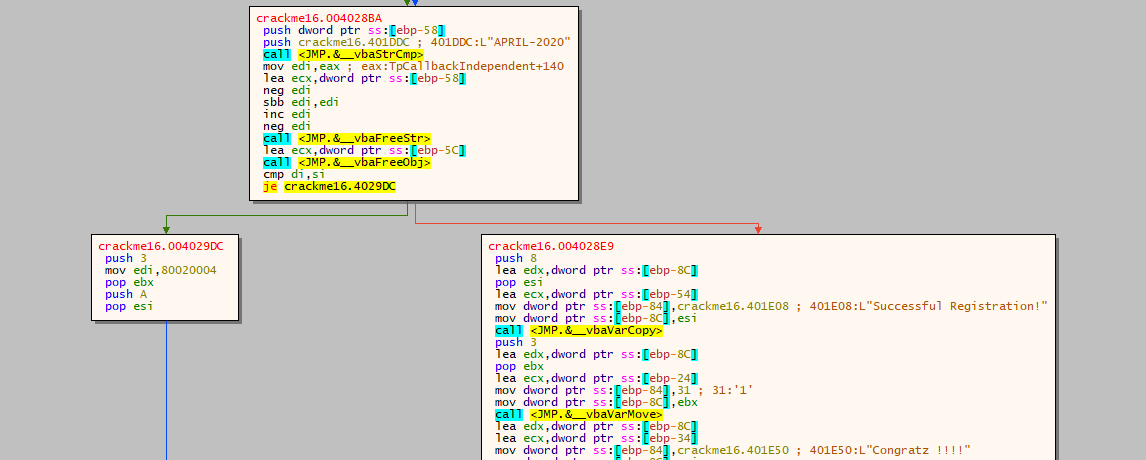
Press the Nag?

We found that, crackme jumps to call eax exactly as the first nag, but eax now is different but same logic as before, we also path the jump into ret.

Continue the program, No nag any more 🙂

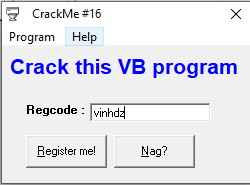
1. Register the crackme.

In reference strings, we found Congratz !!! So jump there



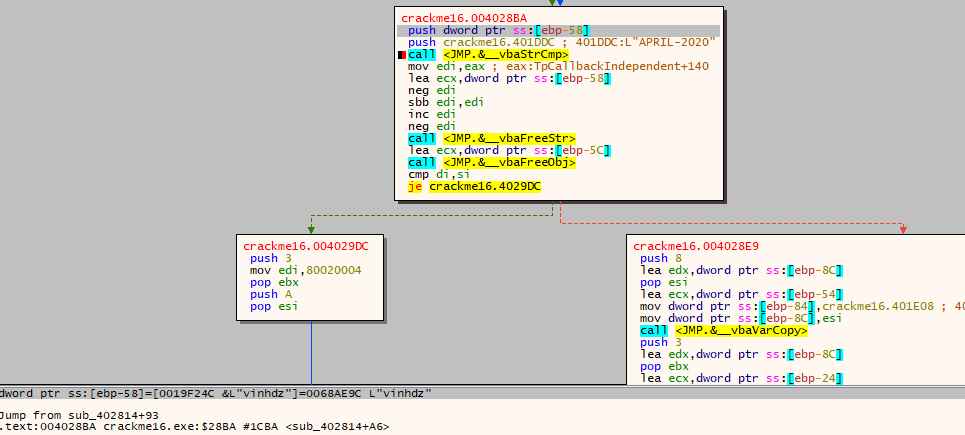
We see there is a je which decides whether to jump to goodboy or not.

And above, we see a function StrCmp which is used to compare two strings, one is stored at [ebp - 58] other one is "APRIL-2020". So we put a breakpoint at StrCmp function and see what is stored at [ebp - 58]



Press Register me!

And we jump to breakpoint



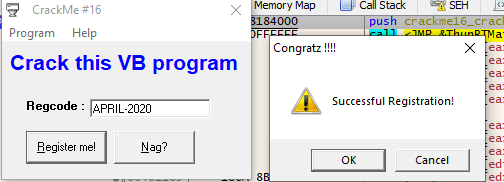
We found that, our input is stored at [ebp - 58]

Result of this function then stored in edi, edi was subtracted by one, then compared to esi which was set to 0 before, if they are equal we can not jump to good boy

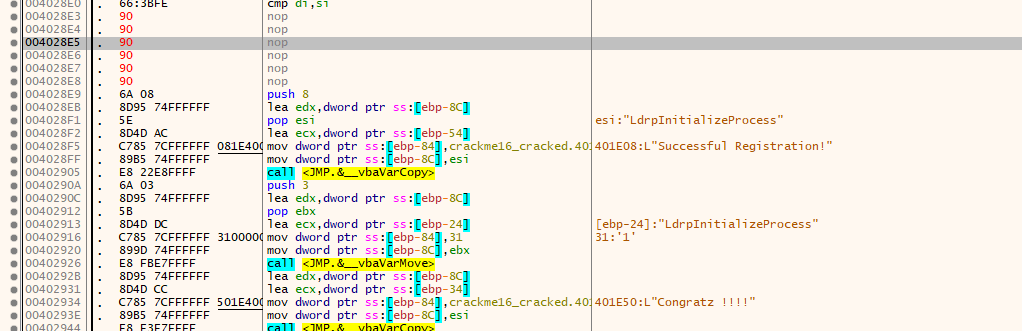
edi - 1 # 0 ⇒ edi # 1 which mean two strings of the StrCmp function must be equal

=> Key is: APRIL-2020

Test



We can patch to the je never jump by nop it

So it always registers us.

Check