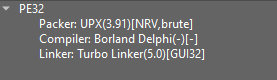


Object: unpack it and find serial

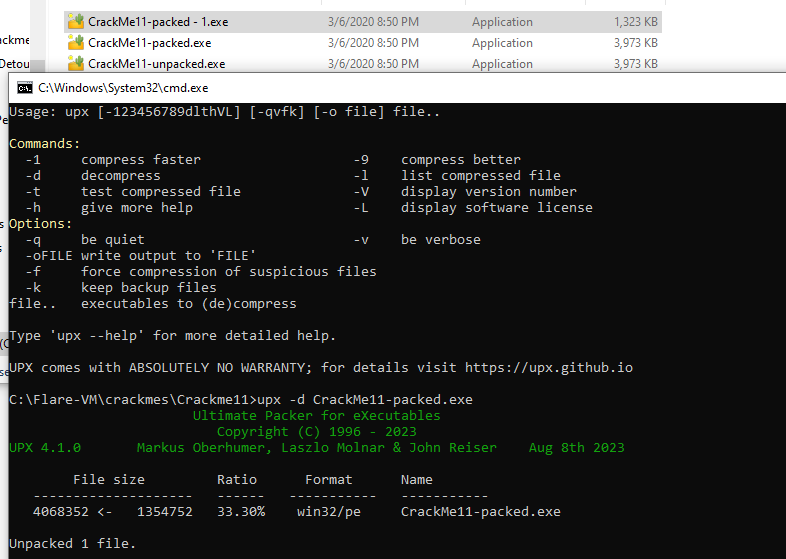
DiE: 32bit PE file



I. Unpack

1. Using upx

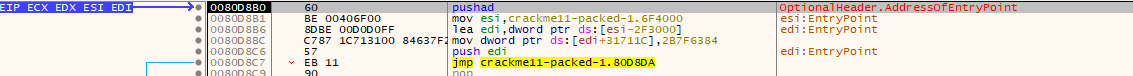
* According to DiE, we know that the crackme was packed by upx, so to unpack it, we can use upx

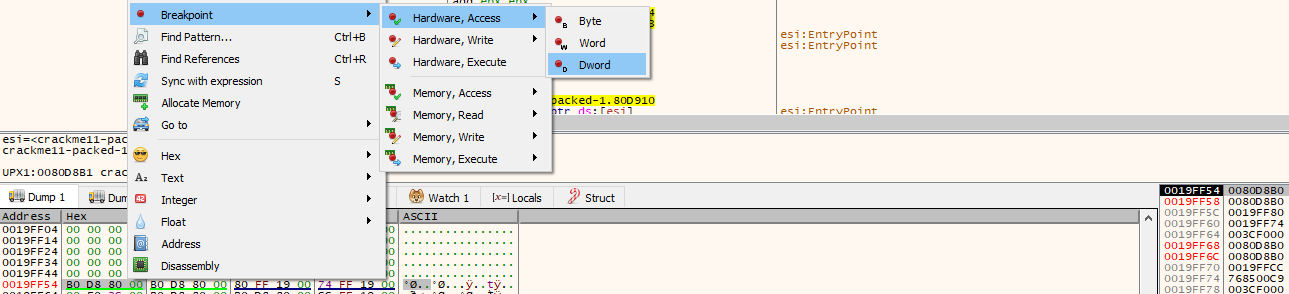
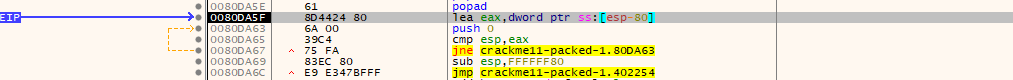


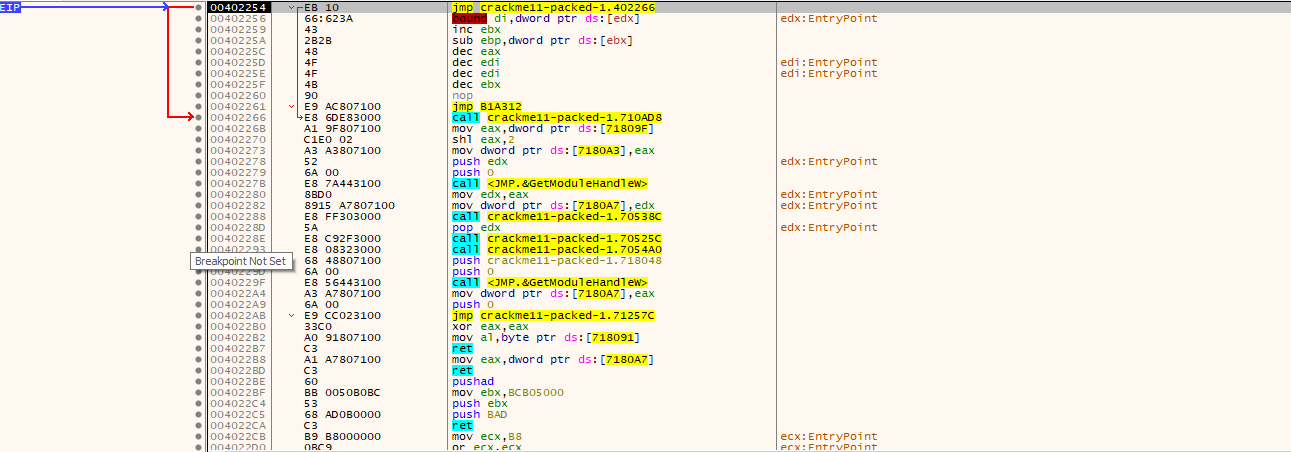
Can see that, file after unpack, it has size exactly with the unpacked crackme author give us

1. Manual

* Throw crackme into x32dbg
* At the Entry Point of the packed file, we see the first instruction is pushad. This is a popular technique used when packing files. This technique first store all register by using pushad, after the program run itself to unpack, before jump to OEP - Origin Entry Point which is the Entry Point of origin file before it is packed, it use a popad instruction to restore the information that pushad has push into stack at very first time.



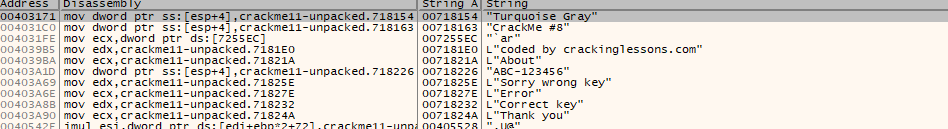
* So to manually unpack the crackme, we just need set a hardware breakpoint on information that pushad push into stack, and until the popad restore it information, the breakpoint stop program for us to find the EOP
* Continue program, and breakpoint stop program
* step a few step more, we reach the OEP



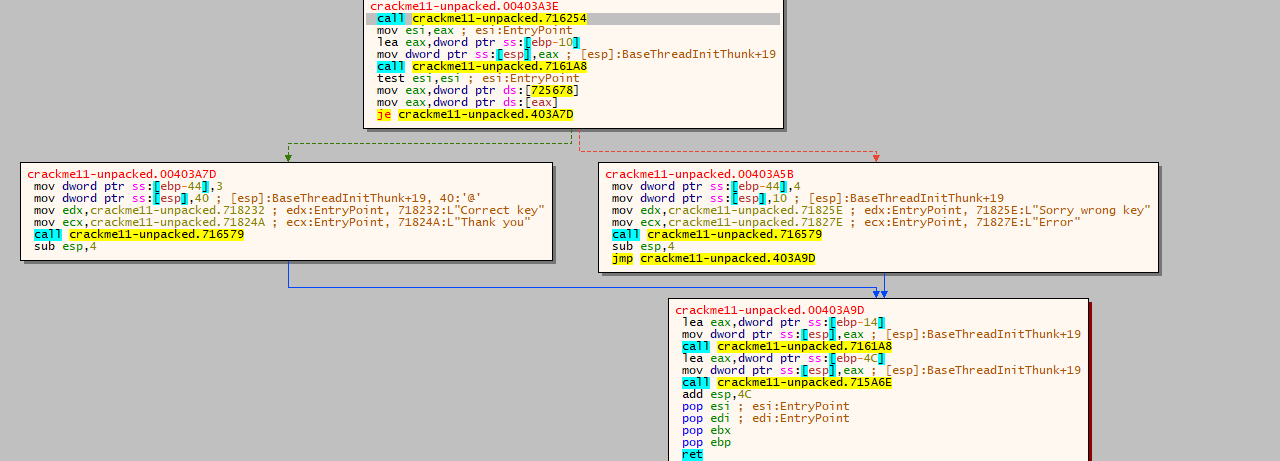
Now, we jumped to OEP., using Scylla to fix IAT, and we get unpacked file same as unpack using upx

II. Register the program

First, we search all reference strings

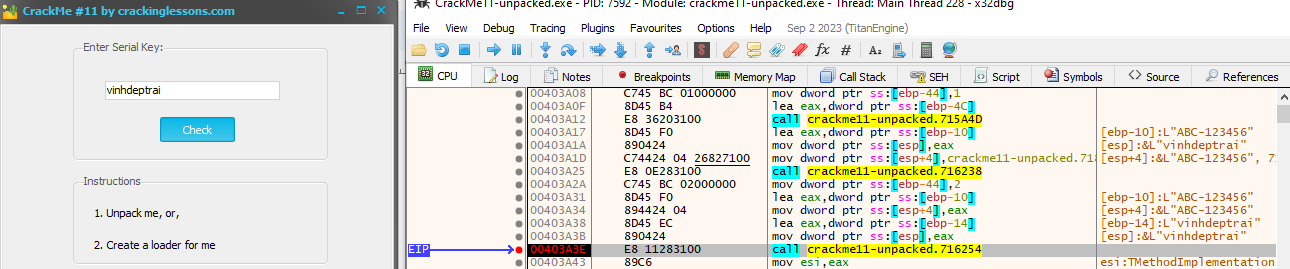


Go to “Correct Key” to view code there



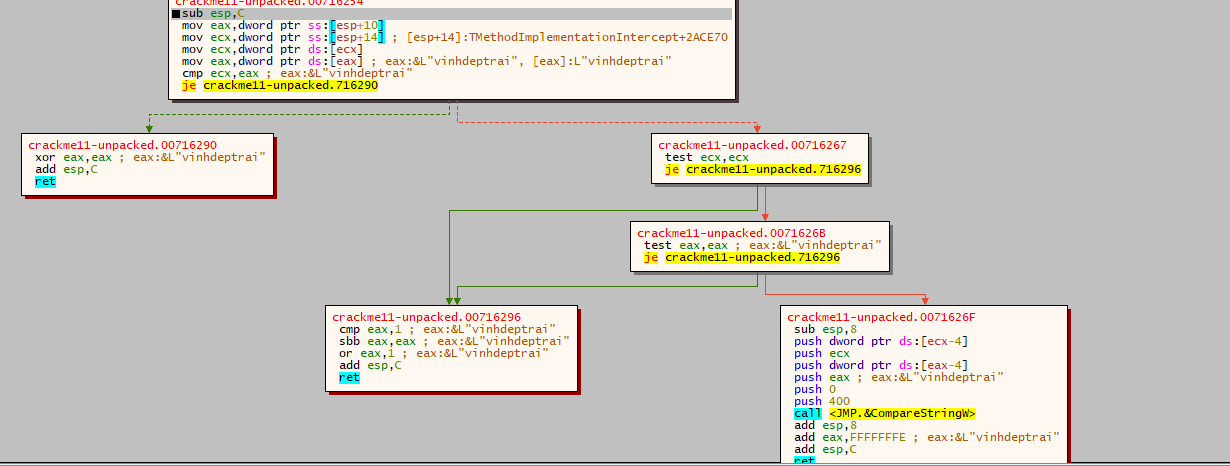
From the code, we can determine that value at ESI decides whether we jump to bad boy or goog boy.

And value at ESI was set by eax at very first, eax is the result of the first call, so set a breakpoint at this call to see what are passed into it and what it does.



So values of parameters passed into this function is input and a default string “ABC-123456”

Let’s dig into this function to see what happen



This function compares two parameters passed into through the CompareStringW function.

So, we can conclude that the valid input for this crackme is **ABC-123456**

Check:

