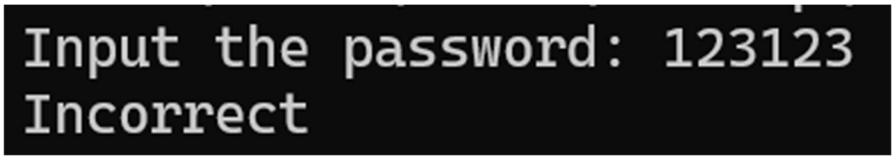
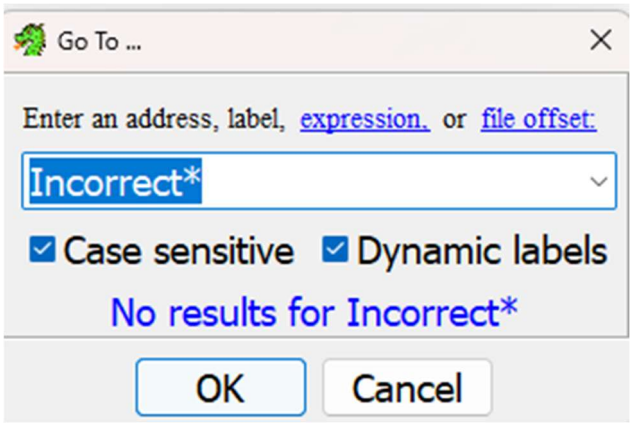
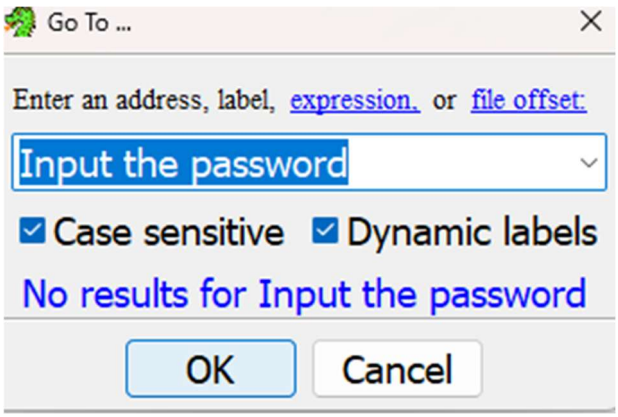
Запускаем программу, вводим пароль (конечно же) неправильно



Идем смотреть в гидре. Попробуем сразу найти место проверки пароля





Но получаем увы.

Однако в процессе просмотра секций файла я заметил следующее

```
undefined4 is_it_main(void)

{
  int iVar1;
  int *piVar2;
  int *piVar3;
  int local_c8 [20];
  int aiStack_78 [17];
  char acStack_33 [20];
  char local_1f [7];
  int local_18;
  int local_14;

  FUN_00401b10();
  local_18 = 0;
  piVar2 = &DAT_00404020;
  piVar3 = local_c8;
  for (iVar1 = 0x2a; iVar1 != 0; iVar1 = iVar1 + -1) {
    *piVar3 = *piVar2;
    piVar2 = piVar2 + 1;
    piVar3 = piVar3 + 1;
  }
  for (local_14 = 0; local_14 < 0x14; local_14 = local_14 + 1) {
    putchar(local_c8[local_14] / (local_14 + 1));
  }
  scanf("%s",local_1f);
  for (local_14 = 0x14; local_14 < 0x1a; local_14 = local_14 + 1) {
    if ((int)acStack_33[local_14] == local_c8[local_14] / (local_14 + 1)) {
      local_18 = local_18 + 1;
    }
  }
  if (local_18 == 6) {
    for (local_14 = 0x1a; local_14 < 0x21; local_14 = local_14 + 1) {
      putchar(local_c8[local_14] / (local_14 + 1));
    }
  }
  else {
    for (local_14 = 0x21; local_14 < 0x2a; local_14 = local_14 + 1) {
      putchar(local_c8[local_14] / (local_14 + 1));
    }
  }
  return 0;
}
```

Проведя ряд небольших манипуляций по переименованию очевидного, получилось следующее

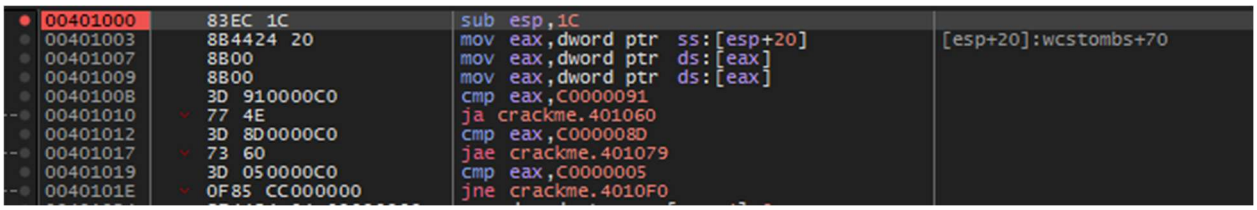
```
undefined4 is_it_main(void)

{
  int iVar1;
  int *piVar2;
  int *piVar3;
  int local_c8 [20];
  int aiStack_78 [17];
  char acStack_33 [20];
  char maybe_input [7];
  int local_18;
  int i;

  FUN_00401b10();
  local_18 = 0;
  piVar2 = &DAT_00404020;
  piVar3 = local_c8;
  for (iVar1 = 0x2a; iVar1 != 0; iVar1 = iVar1 + -1) {
    *piVar3 = *piVar2;
    piVar2 = piVar2 + 1;
    piVar3 = piVar3 + 1;
  }
  for (i = 0; i < 20; i = i + 1) {
    putchar(local_c8[i] / (i + 1));
  }
  scanf("%s",maybe_input);
  for (i = 20; i < 26; i = i + 1) {
    if ((int)acStack_33[i] == local_c8[i] / (i + 1)) {
      local_18 = local_18 + 1;
    }
  }
  if (local_18 == 6) {
    for (i = 26; i < 33; i = i + 1) {
      putchar(local_c8[i] / (i + 1));
    }
  }
  else {
    for (i = 33; i < 42; i = i + 1) {
      putchar(local_c8[i] / (i + 1));
    }
  }
  return 0;
}
```

Но я пока не вижу (или не понимаю) того, что может помочь в решении задания. Пока оставим надежду найти что-то ещё в гидре, попробую найти что-нибудь через x64dbg

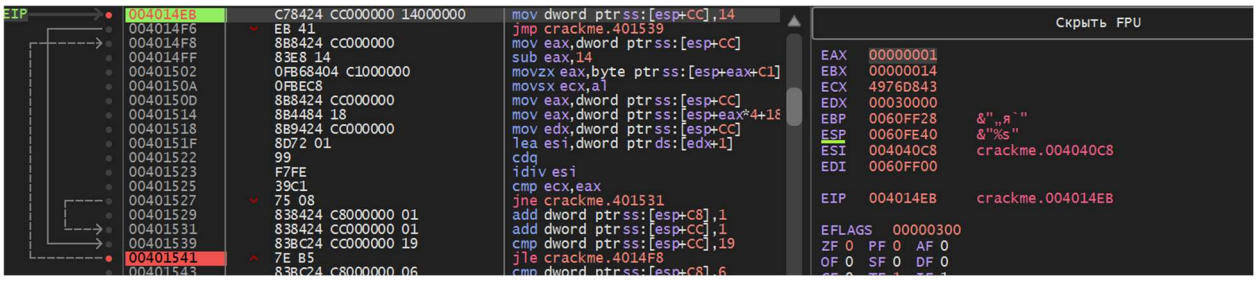
После запуска программы пойдем и поставим точку останова в начале модуля .exe



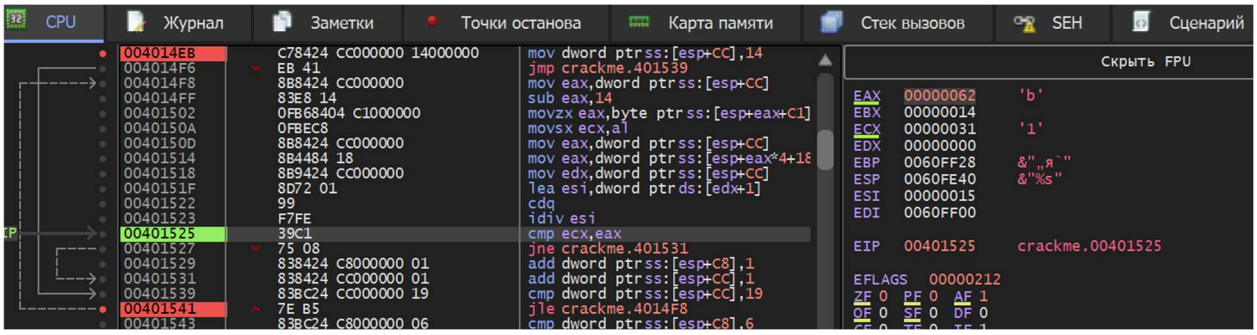
Апдейт:

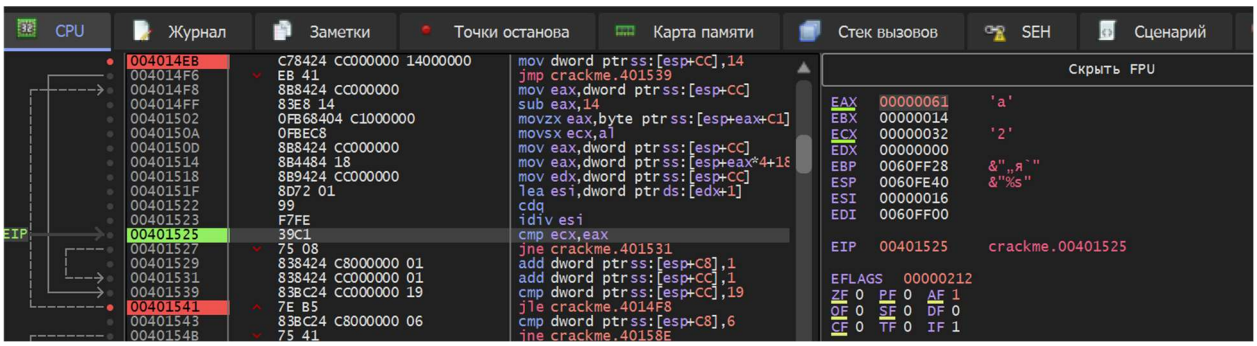
Надо было внимательнее рассмотреть вызовы scanf в этом месте

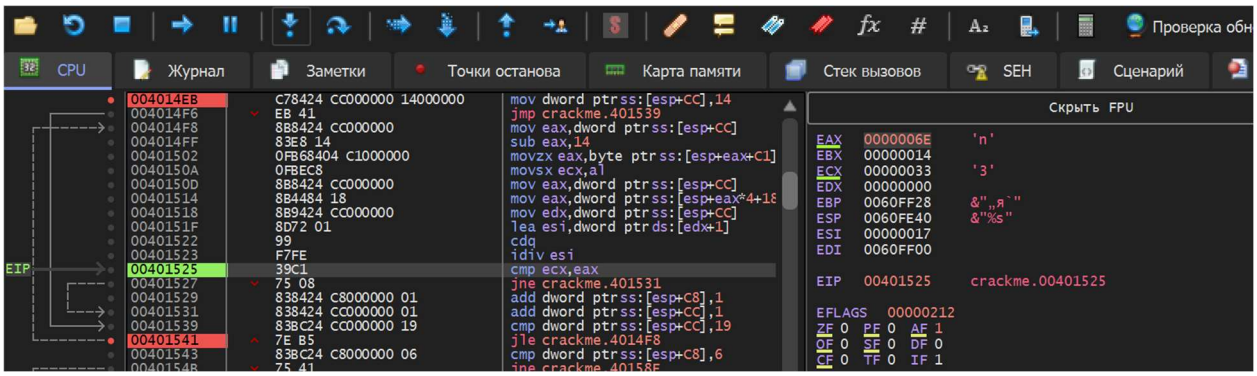
Если поставить точку останова в данном месте в отладчике, то можно увидеть следующее:

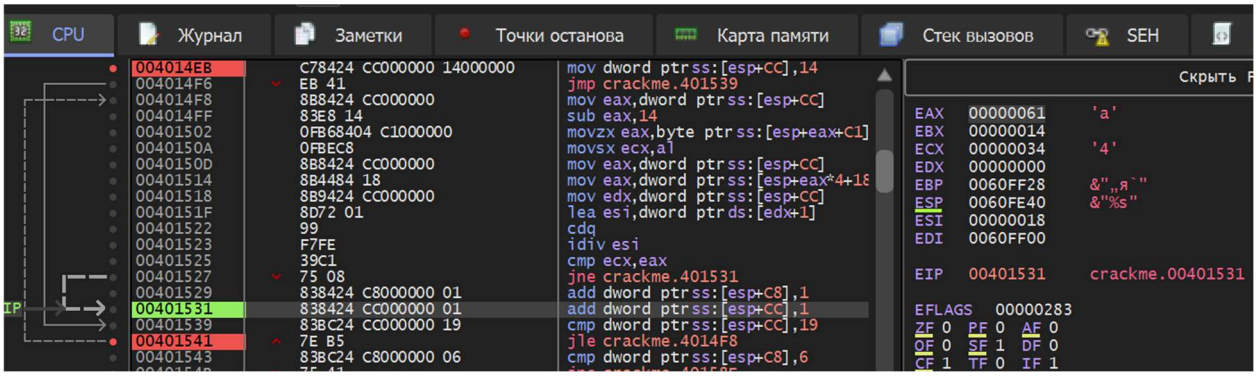


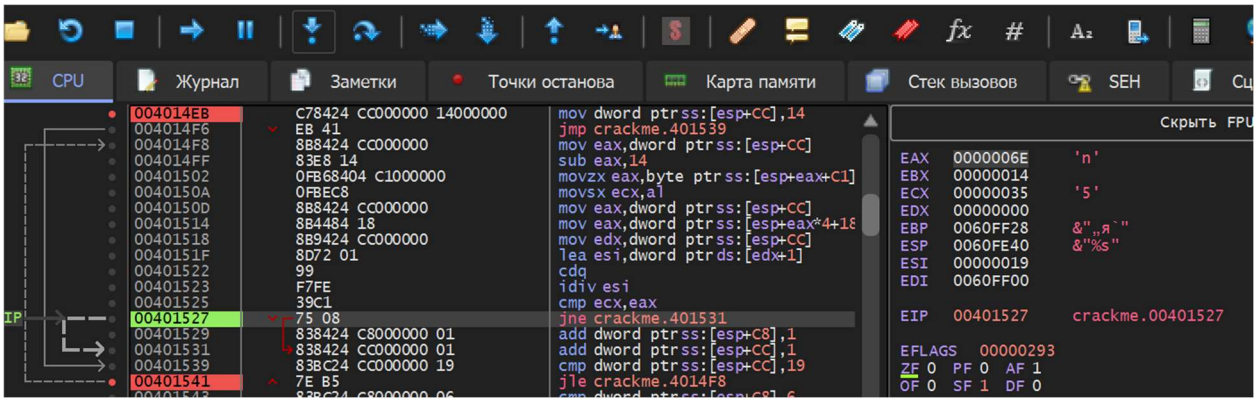
Также увидим цикл, поставим точки останова на вызове первоначальной функции и конце цикла, смотрим регистры при вводе пароля

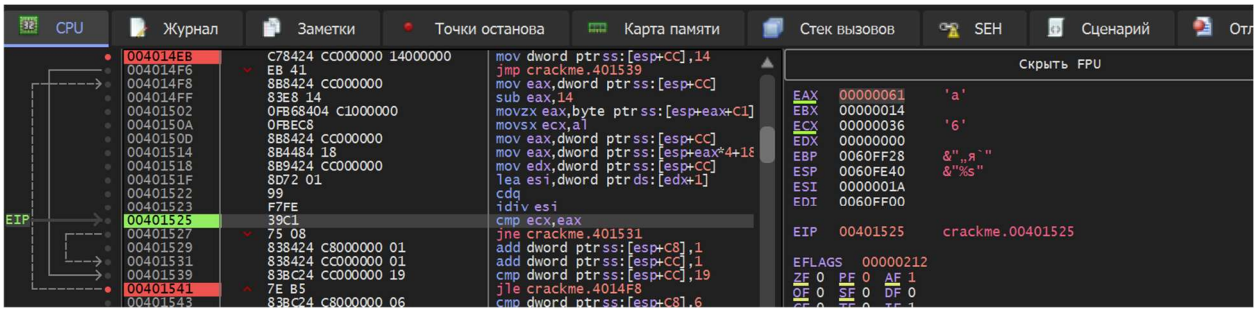










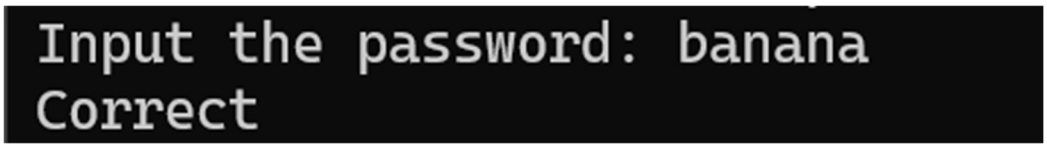


Введенная строка посимвольно загружается в регистр EAX, затем копируется из него в ECX. После этого в регистр EAX загружается эталонное значение проверяемого символа и по адресу 0x00401525 выполняется инструкция сравнения содержимого регистров

Получается, в регистре EAX содержатся символы пароля

Собираем их в строку, получаем слово «banana»

Проверяем



РЕЗЮМЕ:

Надо внимательнее смотреть вызовы функций чтения\записи – что записывают, куда возвращают значение, что потом с ним происходит. В начале я не придал значения инструкции scanf, что привело к бесполезному поиску инструкций валидации пароля в гидре. Надо было изучить внимательнее в дебаггере что происходит по адресу инструкции записи и куда пойдет выполнение программы далее.