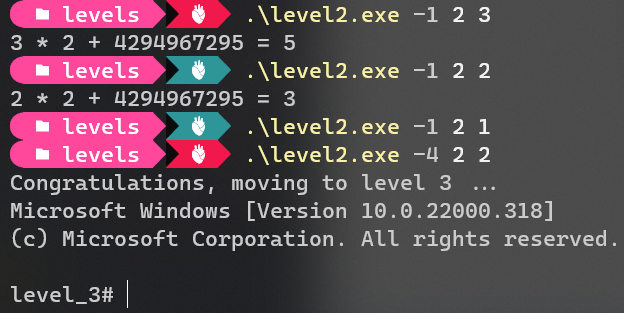
Безопасное программирование СП Лабораторная 4:

Level 1:

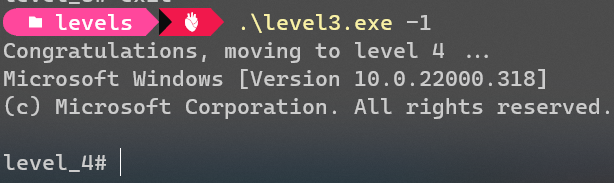
Ввести строку длиной больше 127 и меньшей 256 символов, т.к. char – знаковый тип и занимает интервал от -127 до 127, size\_t от strlen вернёт число больше чем 127 и получится переполнение у чара, если ввести строку длиной 256 символов но меньше чем 384 символа, то при переполнении число будет лежать снова в интервале от 0 до 127.

Level 2:



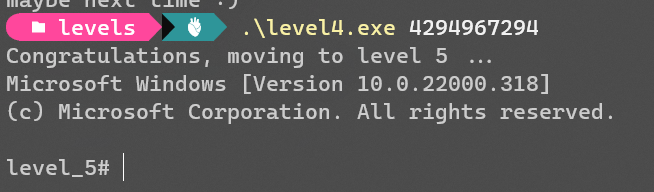
Введя -4 мы вызовем переполнение у unsigned int и значение переменной ui станет равной UINT\_MAX – 4, двумя другими переменныи добавляем 4 и получается, что у нас выйдет за unsigned int по верхней границе и результат суммы станет равным 0.

Level 3:



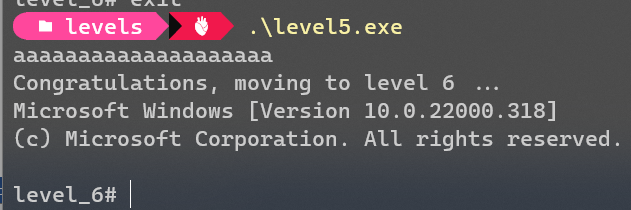
Введя -1 мы не вызовем if для atoi(argv[1] > 200), т.к. оно конвертируется в -1, но потом присвоив -1 переменной i, тип которой unsigned int мы получим переполнение и значение переменной станет UINT\_MAX.

Level 4:



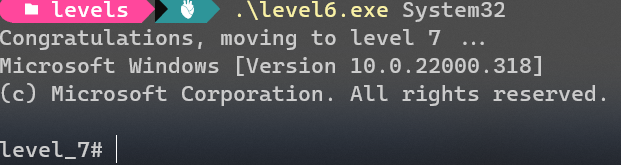
Ввести UINT\_MAX – 2, переполнение, s становится равной 0.

Level 5:



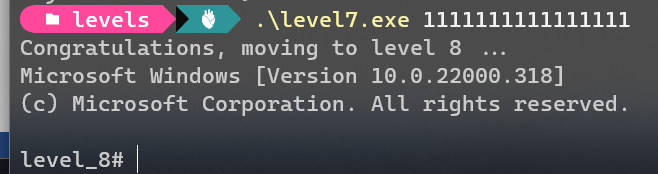
Написать больше 16 символов, gets не проверит длину строки, а в стеке переменная а хранится сразу за массивом buf, значение a поменяется.

Level 6:



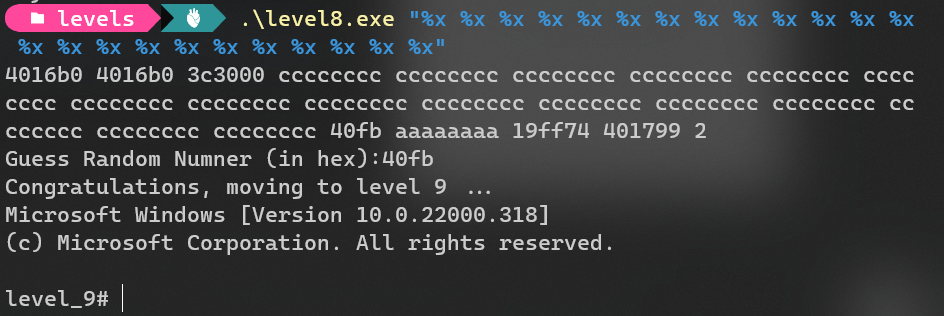
Открывает консоль, находится по пути …/System32/…

Level 7:



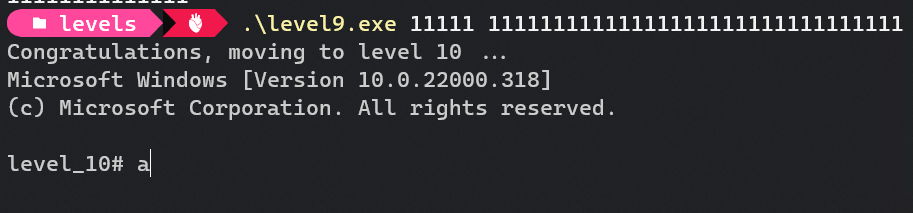
То же самое что и на 5 уровне, но надо ровно 16 символов

Level 8:



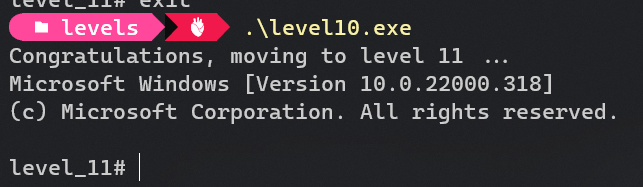
Прочитал стек и нашел данные которые нужны были.

Level 9:



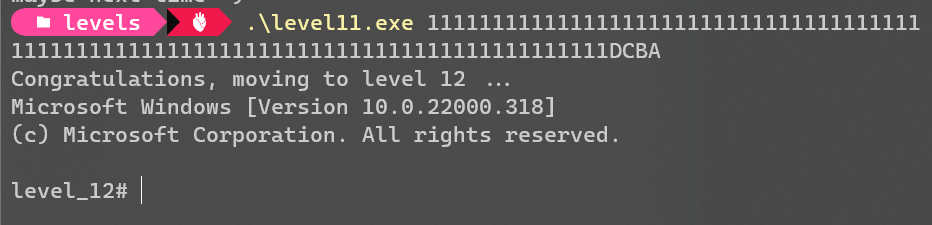
Больше 5 символов для первого аргумента чтобы в функции check\_user() поменялся i при переполнении пароля

Level 10:



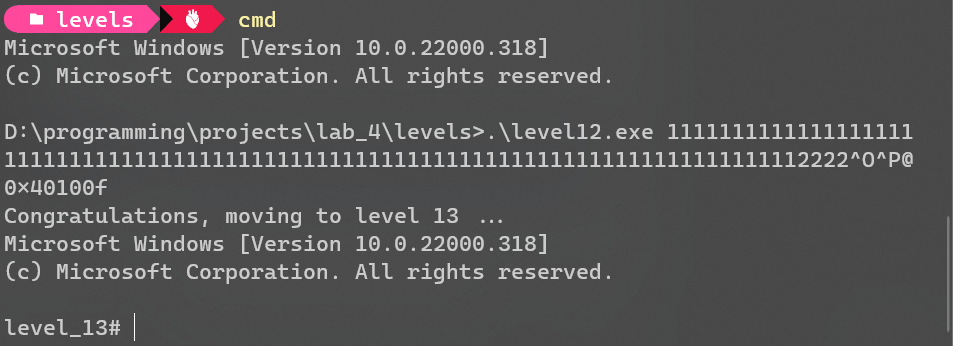
Просто запустил программу…

Level 11:



Переполнил buf, переполнил saved\_ebp, добрался до saved\_eip и поменял в нём значение пользуясь свойством LIFO.

Level 12:



Переполнили buf, 4 символа для printf и вызвали функцию через символы, соответствующие адресу функции в стеке(LIFO)