Уровень 1

Передаю любую комбинацию символов, по длине большую, чем 255, т.к. значения в типе char находятся в пределах от -128 до 127 (либо от 0 до 255 на некоторых платформах). Так же длина комбинации символов должна быть такой, чтобы при переполнении интерпретация получившихся битов в памяти дала отрицательное число. Искомая длина подбирается перебором.

Уровень 2

Передаю 4294967292 2 2,

т. к. 4294967295 является максимальным значением unsigned int; это число +1 приводит к переполнению и обращает условие в 0.

strtoul преобразует строку в числовое значение. Соответственно, у нас получится три числа типов unsigned int, unsigned short и unsigned char. Все они беззнаковые, значит выражение ui+us\*uc тоже будет являться таковым. Максимально значение unsigned int в бинарной системе выглядит как 11111111 11111111 11111111 11111111. Число на единицу большее будет равно 1 00000000 00000000 00000000 00000000. Единица уже не будет учитываться, так как она выйдет за пределы числа, и значит получится просто 0. Тогда условие !(ui+us\*uc) станет правдой.

Уровень 3

Передаю -1

Число I имеет тип unsigned int. Если я передам туда -1, которое в двоичной системе (2-complement) выглядит как 11111111 11111111 11111111 11111111, число будет воспринято как максимальное значение unsigned int, которое точно больше чем 200.

Уровень 4

Передаю (4294967295 - 1)

Максимальное значение unsigned int +1 даст 0. Работает аналогично с уровнем 2.

Поэтому для того, чтобы получить unsigned int s = i + 2 равное нулю, передаю максимальное значение unsigned int -1.

Уровень 5

Передаю больше 15 символов, код которых отличен от 5.

Т.к. на стеке переменная а хранится прямо под элементами массива, gets запишет значение buf[16] в переменную а.

Уровень 6

Передаю System32, т.к. именно в этой папке лежит cmd.exe.

Уровень 7

Передаю 16 любых символов.

17 символ argv[1] (нулевой байт) не помещается в массив buf и записывается в младший байт числа b, т. к. в стеке его младший байт находится под buf[15].

Уровень 8

Передаю "%x %x %x"

Printf получает такую строку-формат, подставляет в нее три следующих своих аргумента (точнее их hex) и выводит. Аргументы он ищет на стеке, но, т.к. ему ничего не передали, там будут адрес возврата и адрес строки формата, а третьей по счету будет идти искомая переменная I.

?Уровень 9

Передаю 11111 11111111111111111111111111111111

Размер key = 32+5. Туда копируется key= и пароль из 32 символов. Итого 36 символов +2 для окончания строки. Получается 38, в I записывается последний символ пароля

Уровень 10

Запускаю level10.exe, ничего не передавая. В argv[0] лежит путь к исполняемому файлу. В нём отсутствуют нулевые байты, и его длина больше 16. После цикла значение I становится равно 16. Следовательно 0 запишется в buf[16], который на самом деле представляет собой число в стеке под массивом, а это и есть I.

Уровень 11

Передаю любые 84 символа и DCBA.

Необходимо получить в eip 0x41424344. В бинарном виде это число равно

01000001 01000010 01000011 01000100. На стеке у нас имеется: eip (4 байта) ? ebp (4 байта) ? buf (80 байт). Соответственно, если передать строчку из 88 байт, то первые 80 скопируются в buf, следующие 4 байта в ebp, а последние 4 байта в eip.

Если перевести каждый блок бинарного числа в символы, то получится ABCD. Но так как в стеке число лежит от младшего байта к старшему, то нужно вводить в обратном порядке.

Уровень 12

Передаю любые 84 символа и ctrl+O ctrl+P @ ctrl+@.

При запуске уровня каждый раз выводится одно и то же число, которое является адресом функции unreachable. Если перевести его в двоичную систему, потом разбить на блоки по 8 бит и каждый блок перевести в десятичное число, то получатся коды символов (0, 64, 16, 15). По таблице ASCII эти символы равны null, @, ^P, ^O соответственно. Эти байты в обратном порядке нужно записать в адрес возврата main, чтобы после выхода из неё запустилась функция unreachable.

На стеке у нас имеется: buf (80 байт) ? аргументы main (4 байта) ? адрес возврата (4 байта). Поэтому передадим 84 символа и наши 4.