Level 1

char находятся в пределах от -128 до 127 (либо от 0 до 255 на некоторых платформах). Когда я передаю строку большой длины (128 символов) произойдет переполнение – в первый бит попадет 1 и значение c будет отрицательным.

Level 2

Максимальное значение unsigned int в бинарной 11111111 11111111 11111111 11111111. Т.к. результат выражения (ui+us\*uc) будет unsigned int, нужно, чтобы это число переполнилось ровно на 1\*переполнить его на еще один разряд не получится, т.к. 65535 \* 255 = 11111110 11111111 00000001. (получим 1 00000000 00000000 00000000 00000000).

Можно передать 4294967292 2 2, 4294967292 1 4, 4294967289 2 3 и т.д.

Level 3

Достаточно передать отрицательное число. atoi вернет знаковую величину, которая будет меньше 200, но когда мы присвоим это значение беззнаковой переменной i, бит со знаком отрежется и i будет больше 200.

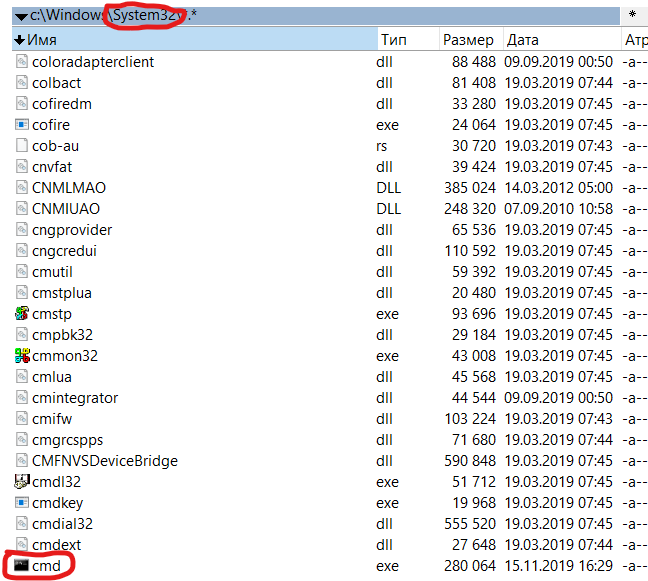
Level 4

Та же самая штука, что во втором уровне. Unsigned int s в !s вернет true, если s равно 0. S = a + 2, т.к. отрицательное а передать не можем, передаем 4294967294 (максимум – 1), получаем переполнение и s = 0.

Level 5

Просто передаю больше символов, чем 15. Переменная a будет находиться сразу за массивом buf16 и это значение будет записано в a.

Level 6



Level 7

Нужно передать 16 любых символов. Тогда 17-й (нулевой байт в любом случае передастся) функция strcpy скопирует в b (т.к. она выше в стеке). Если передать больше, то изменится b.

Level 8

☹

Level 9

Передаю 11111 11111111111111111111111111111111

Размер key = 32+5. key= (4 символа + конец строки) и пароль из 32 символов (=конец строки). Итого 36 символов +2 для окончания строки, strcat получает 38, в I записывается последний символ пароля

Level 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.

Level 11

Передаю любые 84 символа и DCBA.

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

01000001 01000010 01000011 01000100. На стеке saved\_eip, saved\_ebp (4 байта) и buf (80 байт). Если передать строчку из 88 байт, то первые 80 скопируются в buf, следующие 4 байта в saved\_ebp, а последние 4 байта в saved\_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.

Level 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).