**№1**

Char: Занимает в памяти 1 байт (8 бит). Может хранить любой значение из диапазона от -128 до 127, соответственно нужно ввести строку длиной 128 символов, чтобы произошло переполнение и с было < 0.

D:\Женя\Projects\levels\levels>level1.exe "0123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789"

Congratulations, moving to level 2 ...

**№2**

unsigned int: представляет положительное целое число, от 0 до 4 294 967 295 (для 4 байт).

Результат выражение ui+us\*uc приведется к максимальному типу к unsigned int, чтобы его переполнить и получить 0, нужно чтобы результат выражения стал равен 4294967296, это верно например для следующих аргументов.

D:\Женя\Projects\levels\levels>level2.exe "4294967290" "2" "3"

Congratulations, moving to level 3 ...

**№3**

Функция atoi() конвертирует строку, на которую указывает параметр str, в величину типа int.

int: представляет целое число в диапазоне от −2 147 483 648 до 2 147 483 647. Занимает в памяти 4 байта (32 бита).

Соответственно введя число больше 2147483647, например 2147483648 получим переполнение и пройдем первую проверку, однако i имеет тип unsigned int (представляет положительное целое число, от 0 до 4 294 967 295) который без переполнения вмещает данное число.

D:\Женя\Projects\levels\levels>level3.exe "2147483648"

Congratulations, moving to level 4 ...

"cmd.exe" не является внутренней или внешней

командой, исполняемой программой или пакетным файлом.

№4

Передаю число 4294967294(макс значение unsigned int - 1), первую проверку проходим, так как там происходит переполнение, а в случае пере­полнения функция strtol возвращает LONG\_MAX.

Затем strtoul успешно парсит данное значение и прибавив к i 2 получаем снова переполнение(0 в переменной s), что позволяет пройти вторую проверку.

D:\Женя\Projects\levels\levels>level4.exe "4294967294"

Congratulations, moving to level 5 ...

"cmd.exe" не является внутренней или внешней

командой, исполняемой программой или пакетным файлом.

№5

Передаем в gets больше 15 символов(больше размера массива без учета символа конца строки), так же код 16-ого символа не должен быть кодом пятерки. Он запишется в переменную a.

D:\Женя\Projects\levels\levels>level5.exe

0123456789123451

Congratulations, moving to level 6 ...

№6

Передаем System32, чтобы получить корректный путь к файлу cmd.exe.

D:\Женя\Projects\levels\levels>level6.exe "System32"

Congratulations, moving to level 7 ...

"cmd.exe" не является внутренней или внешней

командой, исполняемой программой или пакетным файлом.

№7

b = 4294967295, в десятичной системе счисления, для проверки нам нужно чтобы b = 4294967040, достигнем мы этого превысив размер буфера на 1 символ, введя 16 символов, младший байт числа b заполнится, из-за переполнения.

в b.

D:\Женя\Projects\levels\levels>level7.exe "0123456789123123"

Congratulations, moving to level 8 ...

"cmd.exe" не является внутренней или внешней

командой, исполняемой программой или пакетным файлом.

№8

Передаем "%x %x %x", %x — целое число типа int со знаком в шестнадцатеричной системе счисления, тем самым printf выводит нам 3 аргумента которые мы не передали, поэтому он выведет переменные на стеке, там будут адрес возврата и адрес строки формата, а третьей по счету будет идти искомая переменная.

№9

Передаем пароль из 32 символов, массив key имеет размер 36(без учета конца строки), туда мы записываем 4 символа + символ конца строки, затем присоединяем цепочкой пароль(32 символа), того 4 + 1 + 32 = 37, переполнение из-за чего i уже != 0.

№ 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.

№ 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.

№ 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.