Иванова Мария ПИ 15 группа

**Level 1**

Char: представляет один символ в кодировке ASCII. Занимает в памяти 1 байт (8 бит). Может хранить любое значение из диапазона от -128 до 127, либо от 0 до 255. Если передать строку, которая по длине будет больше 128, то произойдёт переполнение, переменная с будет меньше 0. И мы перейдём на новый уровень.

**Level 2**

11111111111111111111111111111111 – двоичное представление числа 4 294 967 295, которое является максимальным значением для unsigned int.

Т.к. (ui+us\*uc) – unsigned int, то надо переполнить его на 1. То есть нам достаточно получить число 4 294 967 296, которое в двоичном представлении будет 100000000000000000000000000000000. Но из-за переполнения в памяти компьютера останется только 00000000000000000000000000000000, что нам и надо для перехода на новый уровень.

Например, можно ввести 4294967293 1 3 или 4294967291 1 5.

**Level 3**

Нужно ввести не очень большое отрицательное число. Atoi покажет значение со знаком, которое является меньше 200. А когда мы будем записывать отрицательное значение в unsigned int, то в переменной i получим i = UINT\_MAX+x+1, что будет больше 200.

**Level 4**

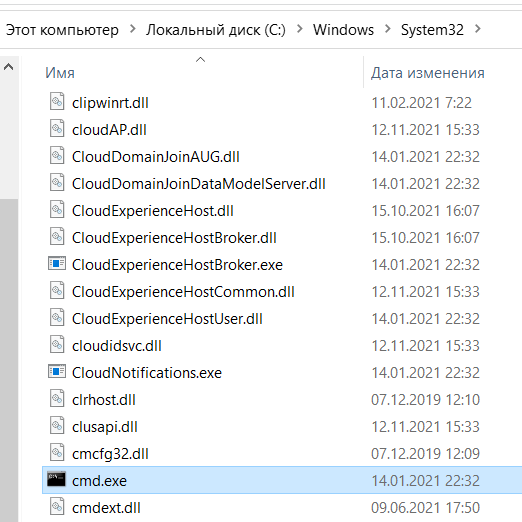
Делаем то же самое, что и на втором уровне. Нужно ввести 4294967294.

**Level 5**

Нужно передать в buf значение, которое по длине больше 15. Функция gets считается небезопасной. Проблема функции в том, что она считывает данные в буфер до тех пор, пока не встретится символ конца строки, не учитывая размер буфера. Т.е. если мы выделили память под 16 символов, а ввели строку из большего числа символов, то следующие за строкой байты будут перезаписаны. Тем самым изменится значение переменной a.

**Level 6**

Нужно ввести System32, потому что именно там лежит cmd.exe.



**Level 7**

Нужно ввести значение, длина которого равна 16.

В таком случае в buf запишется вся наша строка за исключением нуль-символа (символа конца строки). Этот нуль-символ пойдёт в младший байт переменной b. Переменная b изначально была равна 11111111111111111111111111111111. После того, как туда в конец запишется нуль-символ, который является типом char (то есть занимает 8 бит), переменная b станет равна 11111111111111111111111100000000, что нам и надо. А значение переменной a останется прежним.

**Level 8**

Передаём «%x %x %x %x %x %x %x %x %x %x %x %x %x %x %x %x %x %x %x %x».

В printf передаётся такая строка-формат. Printf подставляет в неё аргументы, которые лежат в стеке, и выводит их в шестнадцатеричной форме. Первым будет идти адрес возврата, потом адрес строки формата и т.д. Последним аргументом будет идти наша искомая величина.

**Level 9**

Передаём 12345 12345678901234567890123456789012.

Пароль передаём, состоящий из 32 символов или больше. Размер key выставили MAXPASS + 5 = 37. Записываем в key «key=» и пароль. Получается, что записываем в key 38 символов (4+32+2). Происходит переполнение и в переменную i записывается символ пароля.

**Level 10**

Не передаю никакие параметры в программу. В argv[0] лежит полное имя запущенной программы, длина которого больше 16. Значит, после цикла переменная i будет равна 16. 0 запишется в buf[16]. А поскольку память под него уже не выделялась, то 0 запишется в переменную i.

**Level 11**

Передаём строку, состоящую из 88 символов, последние четыре из которых должны быть равны DCBA (остальные любые).

Нам надо получить saved\_eip = 0x41424344, то есть в бинарном виде saved\_eip = 01000001 01000010 01000011 01000100 или в ASCII saved\_eip = ABCD. Поэтому мы вводим 88 символов. Первые 80 идут в buf, следующие 4 идут в saved\_ebp (т.к. unsigned int занимает 4 байта), а последние 4 (DCBA) идут в saved\_eip, как нам и надо. А вводим в перевёрнутом порядке, так как в стеке число лежит от младшего к старшему байту.

**Level 12**

Передаём строку, состоящую из 88 символов, последние четыре из которых должны быть равны ctrl+O ctrl+P @ ctrl+@ (остальные любые).

Нам надо вызвать функцию, адрес которой 0x40100f (он прописывается после каждого запуска программы). В двоичном виде этот адрес: 00000000 01000000 00010000 00001111. Если перевести эти блоки в десятичные числа, то получим: 0, 64, 16, 15. Переведём эти десятичные числа в символы ASCII: NUL (ctrl+@), @, DLE (ctrl+P), SI (ctrl+O). Записывать нам надо их в обратном порядке.

Следовательно, 80 символов пойдёт в buf, 4 символа (4 байта) в аргументы main и 4 последних символа (ctrl+O ctrl+P @ ctrl+@) в адрес возврата.