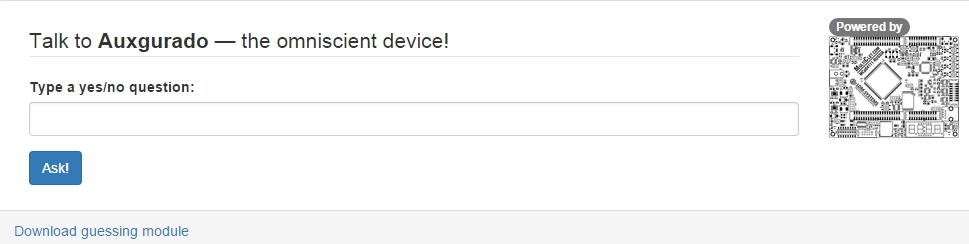
Заходим на сайт:



Сразу же жмём на «Download guessing module» и качаем исходники прошивки. 200 строк ассемблера под мультиклет не должны нас испугать. В конце листинга видно строки возможных ответов сервиса:

.data

M0: .asciz "Yes"

M1: .asciz "No"

M2: .asciz "Maybe"

M3: .asciz "I don't know"

M4: .asciz "Probably"

M5: .asciz "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

Вводя случайные вопросы увидим, что последний вариант (строка из звёздочек) никогда не появляется.

Попробуем понять, какую строку нужно ввести для получения этого ответа.

Посмотрим код функции do\_guess:

.global do\_guess

do\_guess:

jmp do\_guess.1

rdq Question

rdq Question + 8

rdq Question + 16

rdq Question + 24

xorq @4, @3

xorq @1, @3

xorq @1, @3

setq #0, @1

complete

В самом начале первые 32 байта строки вопроса читаются по 8 байт и полученные числа ксорятся. Результат попадает в регистр #0. Дальше по коду строка вопроса больше никак на используетсяВо втором параграфе функции do\_guess одна половина восьмибайтного числа из #0 вычитается из другой и результат сохраняется обратно в #0:

do\_guess.1:

jmp do\_guess.2

getq #0

getq 0

patch @1, @2

pack @2, @3

subl @1, @2

setq #0, @1

complete

В третьем параграфе значение #0 умножается на константу 0xB2D060ED и записывается обратно:

do\_guess.2:

jmp do\_guess.3

getl 0xB2D060ED

mull @1, #0

setq #0, @1

complete

В следующих параграфах значение #0 сравнивается с числами 0x33333333, 0x66666666, 0x99999999, 0xCCCCCCCC. В зависимости от того, между какими из них это значение окажется, выбирается один из первых 5 вариантов ответа. Если же оно оказалось строго равно 0xCCCCCCCC, выбирается интересующий нас шестой вариант.

Теперь осталось подобрать такой вопрос, чтобы значение #0 было равно 0xCCCCCCCC.

Проблему создаёт только ограничение на набор символов в строке вопроса. Это должны быть символы ASCII из диапазона 32 – 127.

Будем искать подходящие строки длины 16 (нам хватит, а остальные 16 байт будут нулями и никак на повлияют на результат).

До умножения на константу 0xB2D060ED число в #0 должно быть равно 0x940C027C. Будем перебирать пары четырёхбайтных чисел, сумма которых равна этому числу. И для каждого из чисел в паре будем в свою очередь искать пару четырёхбайтных печатных строк, ксором которых оно может быть получено. Если найти получилось – составляем найденных частей одну 16-байтную строку.

С помощью некоторых оптимизаций можно сделать этот алгоритм быстрым.

Вот одна из найденных строк: !!!!!!!!!!!M]#-!

Вводим её в качестве вопроса и получаем флаг:

