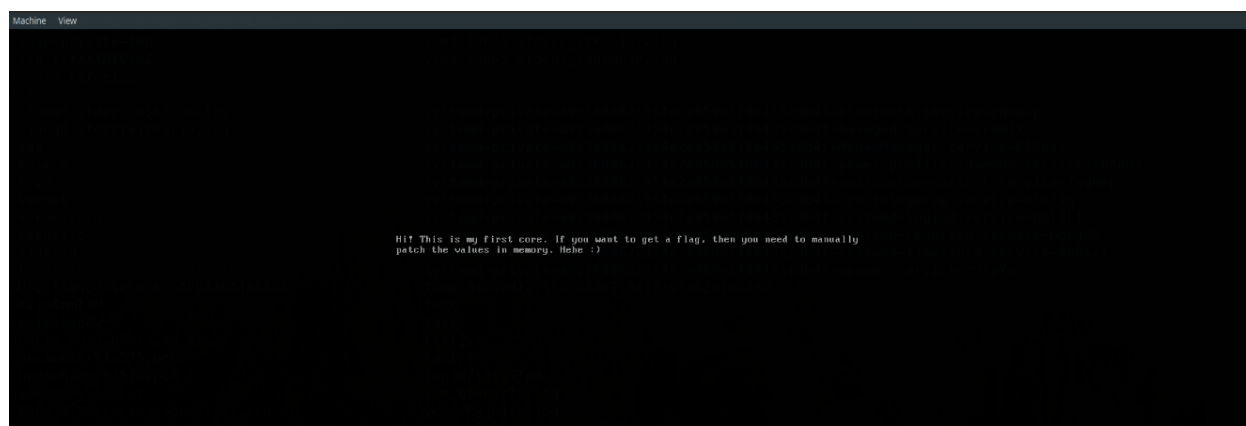




Название:	Моё первое ядро
Категория:	Реверс-инжиниринг
Уровень:	Средний
Очки:	400
Описание:	Попробуй запустить моё первое ядро. Там есть вывод текста :)
Теги:	Ядро, QEMU, патчинг
Автор:	ROP

Прохождение:

Открываем архив, видим файл `kernel`. Его можно запустить через `qemu-system-i386 -kernel kernel`.



Идём тогда в IDA. Изучаем функцию `kmain`. Можем восстановить исходник.

```

void kmain(void)
{
    const char *str = "Hi! This is my first core. If you want
to get a flag, then you need to manually patch the values in
memory. Hehe :)";
    const char *good = "Great! Look in memory before the loop
how they will look as a string, and then add them between cur
ly brackets in CODEBY{
}.";
    char *vidptr = (char*)0xb8000;
    unsigned int i = 0;
    unsigned int j = 0;

    while(j < 80 * 50 * 2) {
        vidptr[j] = ' ';
        vidptr[j+1] = 0x07;
        j = j + 2;
    }

    j = 0;

    while(str[j] != '\\0') {
        vidptr[i] = str[j];
        vidptr[i+1] = 0x07;
        ++j;
        i = i + 2;
    }

    unsigned int dec1 = 0x1;
    unsigned int dec2 = 0x1;
    unsigned int dec3 = 0x1;
    unsigned int dec4 = 0x1;
    unsigned int dec5 = 0;

    for (i = 0; i < dec1; i++) {

```

```

        dec2 ^= dec3;
        dec3 += dec4^1337;
        dec4 -= dec3^dec2;
        dec2 ^= dec4+dec3;
        dec3 ^= dec2-dec4;
        dec4 ^= dec2+dec3;
    }

    if (

        dec1 == 0x1337 &&
        dec2 == 0x82b9f885 &&
        dec3 == 0x5afcb5c &&
        dec4 == 0x77e64835
        ) {
        while(j < 80 * 50 * 2) {
            vidptr[j] = ' ';
            vidptr[j+1] = 0x07;
            j = j + 2;
        }

        j = 0;
        i = 0;

        while(good[j] != '\\0') {
            vidptr[i] = good[j];
            vidptr[i+1] = 0x02;
            ++j;
            i = i + 2;
        }

    }

    return;
}

```

Нам нужно менять от `dec1` до `dec4`. Можем написать программу, что выдаст нам нужные значения.

```
#include <stdio.h>

int main() {

    unsigned int dec1 = 0x1337;
    unsigned int dec2 = 0x82b9f885;
    unsigned int dec3 = 0x5afcb5c;
    unsigned int dec4 = 0x77e64835;
    unsigned int dec5 = 0;

    unsigned int i;

    for (i = 0; i < dec1; i++) {
        dec4 ^= dec2+dec3;
        dec3 ^= dec2-dec4;
        dec2 ^= dec4+dec3;
        dec4 += dec3^dec2;
        dec3 -= dec4^1337;
        dec2 ^= dec3;
    }

    printf("%#x\\n", dec1);
    printf("%#x\\n", dec2);
    printf("%#x\\n", dec3);
    printf("%#x\\n", dec4);

}
```

Запускаем.

```
0x1337
0x53723166
```

0x6c654e72

The screenshot shows the Immunity Debugger interface with the following details:

- Menu Bar:** File, Edit, Jump, Search, View, Debugger, Options, Windows, Help.
- Toolbar:** Standard debugger icons for file operations, search, and execution.
- Left Panel:**
 - Functions:** Lists 'start' and 'kmain'.
 - Function name:** 'start'.
 - File:** 'x86_get_cpu_thunk_x86'.
- Main Window (Hex View-1):**
 - Disassembly:**

```

.text:001000B3 0F B6 00 movzx eax, ds:(_GLOBAL_OFFSET_TABLE_ - 100310h)[eax]
.text:001000B6 84 C0 test al, al
.text:001000B8 75 C6 jnz short loc_1000B0
.text:001000BA C7 45 F8 37 13 00 00 mov [ebp+var_8], 1337h
.text:001000C1 C7 45 E0 66 31 72 53 mov [ebp+var_20], 53723166h
.text:001000C8 C7 45 E4 74 5F 68 32 mov [ebp+var_1C], 32685F74h
.text:001000CF C7 45 E8 72 4E 65 6C mov [ebp+var_18], 6C654E72h
.text:001000D6 C7 45 FC 00 00 00 00 mov [ebp+var_4], 0
.text:001000DD C7 45 D8 00 00 00 00 mov [ebp+var_28], 0
.text:001000E4 EB 3D jmp short loc_100123
;
loc_1000E6:
.text:001000E6
.text:001000E6
.text:001000E6 8B 45 E4 mov eax, [ebp+var_1C]
.text:001000E9 31 45 E0 xor [ebp+var_20], eax
.text:001000EC 8B 45 E8 mov eax, [ebp+var_18]
.text:001000EF 35 39 05 00 00 xor eax, 539h
.text:001000F4 01 45 E4 add [ebp+var_1C], eax
.text:001000F7 8B 45 E4 mov eax, [ebp+var_1C]
.text:001000FA 33 45 E0 xor eax, [ebp+var_20]
.text:001000FD 29 45 E8 sub [ebp+var_18], eax
.text:00100100 8B 55 E8 mov edx, [ebp+var_18]
.text:00100103 8B 45 E4 mov eax, [ebp+var_1C]
.text:00100106 01 D0 add eax, edx
.text:00100108 31 45 E0 xor [ebp+var_20], eax
.text:0010010B 8B 45 E0 mov eax, [ebp+var_20]
.text:0010010E 2B 45 E8 sub eax, [ebp+var_18]

```
 - Comment:** 'CODE XREF: kmain+111:j'.
 - Status Bar:** '000100C1: 001000C1: kmain+85 (Synchronized with Hex View-1)'.
- Bottom Panel:**
 - Output:**

The decompilation process is 75%.

Please check the Edit/Plugins menu for more information.

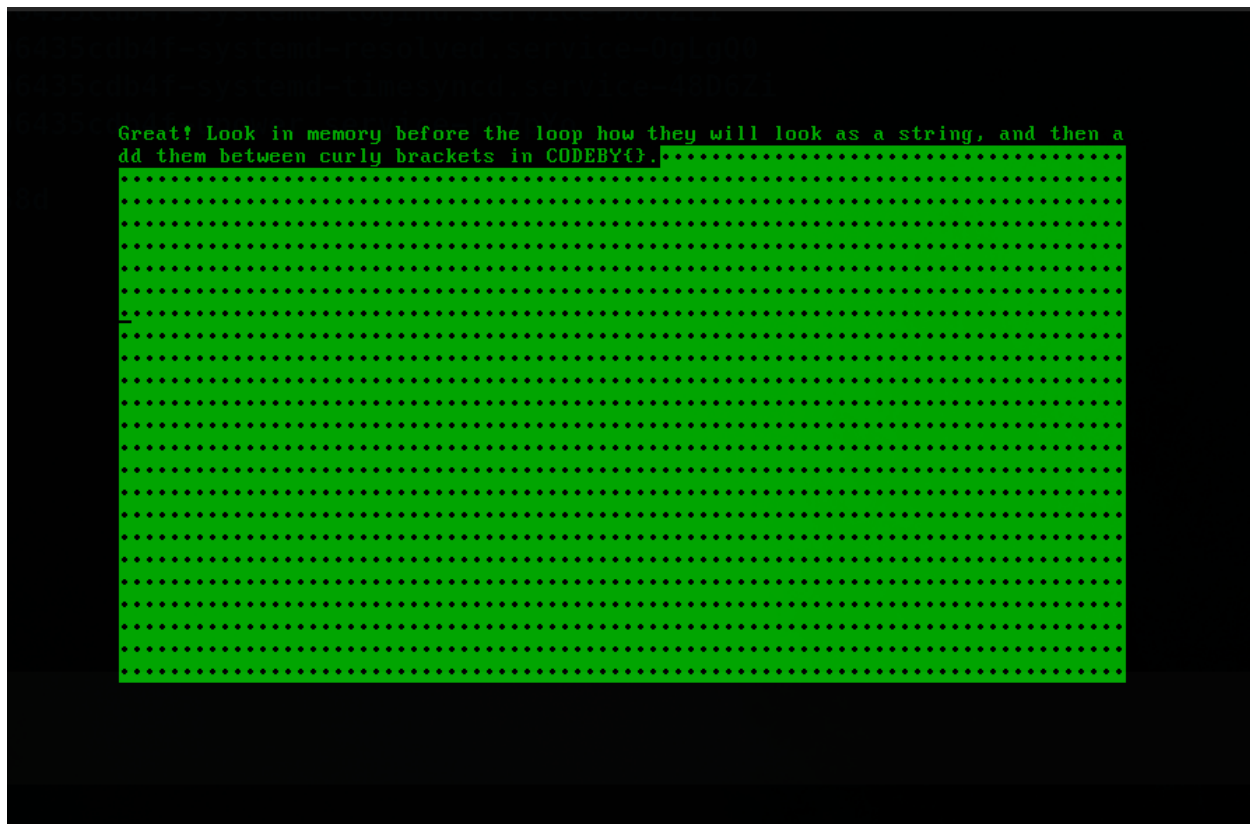
Propagating type information...

Function argument information has been propagated

The initial autoanalysis has been finished.

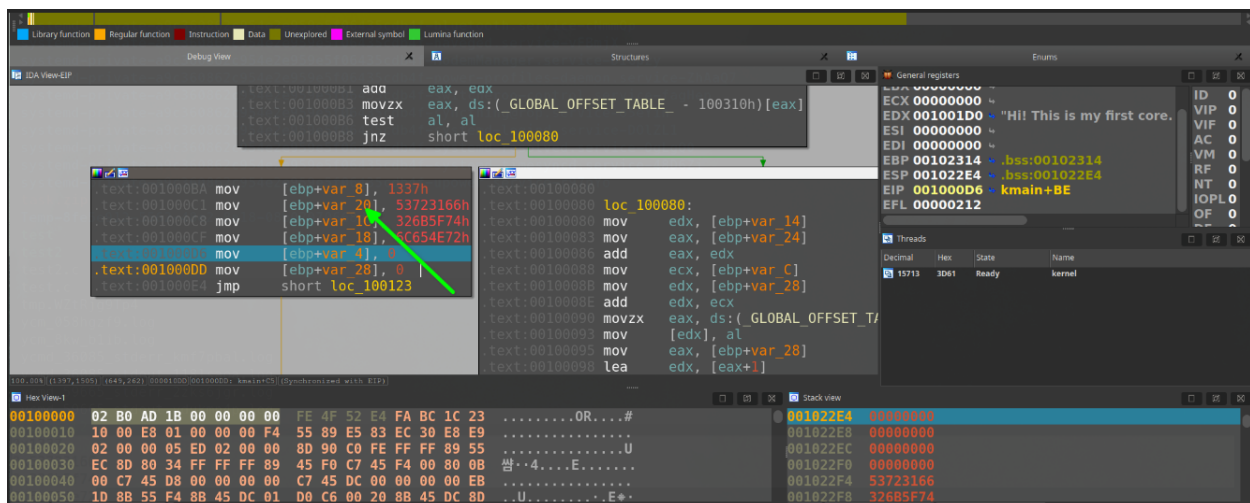
Applied 14/14 patch(es)
 - Register:** 'r0c 0x6c 65 4e 72'.

5



Успех!

Поменяв немного EIP в IDA мы можем посмотреть содержимое памяти и без QEMU.



Нажимаем 2 раза ЛКМ на `var_20`. Там будет флаг.