vm

写个正向的代码

```
#include<stdio.h>
unsigned char key[1008] = {
    0x00, 0x03, 0x02, 0x00, 0x03, 0x00, 0x02, 0x03, 0x00, 0x00, 0x00, 0x00,
0x00, 0x02, 0x01, 0x00,
    0x00, 0x03, 0x02, 0x32, 0x03, 0x00, 0x02, 0x03, 0x00, 0x00, 0x00, 0x00,
0x03, 0x00, 0x01, 0x00,
    0x00, 0x03, 0x02, 0x64, 0x03, 0x00, 0x02, 0x03, 0x00, 0x00, 0x00, 0x00,
0x03, 0x03, 0x01, 0x00,
    0x00, 0x03, 0x00, 0x08, 0x00, 0x02, 0x02, 0x01, 0x03, 0x04, 0x01, 0x00,
0x03, 0x05, 0x02, 0x00,
    0x03, 0x00, 0x01, 0x02, 0x00, 0x02, 0x00, 0x01, 0x01, 0x00, 0x00, 0x03,
0x00, 0x01, 0x03, 0x00,
    0x03, 0x00, 0x00, 0x02, 0x00, 0x03, 0x00, 0x03, 0x01, 0x28, 0x04, 0x06,
0x5F, 0x05, 0x00, 0x00,
    0x03, 0x03, 0x00, 0x02, 0x01, 0x00, 0x03, 0x02, 0x96, 0x03, 0x00, 0x02,
0x03, 0x00, 0x00, 0x00,
    0x00, 0x04, 0x07, 0x88, 0x00, 0x03, 0x00, 0x01, 0x03, 0x00, 0x03, 0x00,
0x00, 0x02, 0x00, 0x03,
    0x00, 0x03, 0x01, 0x28, 0x04, 0x07, 0x63, 0xFF, 0xFF};
unsigned int input[200] = {
    0x00000030, 0x00000031, 0x00000032, 0x00000033, 0x00000034, 0x00000035,
0x00000036, 0x00000037,
    0x00000038, 0x00000039, 0x00000030, 0x00000031, 0x00000032, 0x00000033,
0x00000034, 0x00000035,
    0x00000036, 0x00000037, 0x00000038, 0x00000039, 0x00000030, 0x00000031,
0x00000032, 0x00000033,
    0x00000034, 0x00000035, 0x00000036, 0x00000037, 0x00000038, 0x00000039,
0x00000030, 0x00000031,
    0x00000032, 0x00000033, 0x00000034, 0x00000035, 0x00000036, 0x00000037,
0x00000038, 0x00000039,
    0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000,
    0x00000000, 0x00000000, 0x0000009B, 0x000000A8, 0x00000002, 0x000000BC,
0x000000AC, 0x0000009C,
    0x000000CE, 0x000000FA, 0x00000002, 0x000000B9, 0x000000FF, 0x0000003A,
0x00000074, 0x00000048,
    0x00000019, 0x00000069, 0x000000E8, 0x00000003, 0x000000CB, 0x000000C9,
0x000000FF, 0x000000FC,
    0x00000080, 0x000000D6, 0x0000008D, 0x000000D7, 0x00000072, 0x00000000,
0x000000A7, 0x0000001D,
    0x0000003D, 0x00000099, 0x00000088, 0x00000099, 0x000000BF, 0x000000E8,
0x00000096, 0x0000002E,
    0x0000005D, 0x00000057, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000,
    0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x000000C9, 0x000000A9,
0x000000BD, 0x0000008B,
    0x00000017, 0x000000C2, 0x0000006E, 0x000000F8, 0x000000F5, 0x0000006E,
0x00000063, 0x00000063,
```

```
    0x000000D5, 0x00000046, 0x0000005D, 0x00000016, 0x00000098, 0x00000038,
0x00000030, 0x00000073,
    0x00000038, 0x000000C1, 0x0000005E, 0x000000ED, 0x000000B0, 0x00000029,
0x0000005A, 0x00000018,
    0x00000040, 0x000000A7, 0x000000FD, 0x0000000A, 0x0000001E, 0x00000078,
0x0000008B, 0x00000062,
    0x000000DB, 0x0000000F, 0x0000008F, 0x0000009C, 0x00000000, 0x00000000,
0x00000000, 0x00000000,
    0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00004800, 0x0000F100,
    0x00004000, 0x00002100, 0x00003501, 0x00006400, 0x00007801, 0x0000F900,
0x00001801, 0x00005200,
    0x00002500, 0x00005D01, 0x00004700, 0x0000FD00, 0x00006901, 0x00005C00,
0x0000AF01, 0x0000B200,
    0x0000EC01, 0x00005201, 0x00004F01, 0x00001A01, 0x00005000, 0x00008501,
0x0000CD00, 0x00002300,
    0x0000F800, 0x00000C00, 0x0000CF00, 0x00003D01, 0x00004501, 0x00008200,
0x0000D201, 0x00002901,
    0x0000D501, 0x00000601, 0x0000A201, 0x0000DE00, 0x0000A601, 0x0000CA01,
0x00000000, 0x00000000,
    0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000, 0x00000000,
0x00000000, 0x00000000
};
int a1[9] = { 0 };
int reg[80] = { 0 };
int main() {
    int count = 1,count_func5 = 1;
    while (key[a1[6]] != 255) {
        printf("count:%d, call func%d, ",count++, key[a1[6]]);
        switch (key[a1[6]])
        {
        case 0u:
            printf("run opcode: %d %d %d %d", key[a1[6]], key[a1[6] + 1],
key[a1[6] + 2], key[a1[6] + 3]);
            if (key[a1[6] + 1])
            {
                switch (key[a1[6] + 1])
                {
                case 1u:
                    printf(",input[%d] = a1[0]", a1[2]);
                    input[a1[2]] = a1[0];
                    break;
                case 2u:
                    printf(", a1[%d] = a1[%d]", key[a1[6] + 2],key[a1[6] + 3]);
                    a1[key[a1[6] + 2]] = a1[key[a1[6] + 3]];
                    break;
                case 3u:
                    printf(", a1[%d] = %d", key[a1[6] + 2], key[a1[6] + 3]);
                    a1[key[a1[6] + 2]] = key[a1[6] + 3];
                    break;
                }
            }
            else
            {
                printf(", a1[0] = input[%d]", a1[2]);
```

```
                a1[0] = input[a1[2]];
            }
            a1[6] = a1[6] + 4;
            printf(", a1[0~3] = %d, %d, %d, %d", a1[0], a1[1], a1[2], a1[3]);
            break;
        case 1u:
            printf("run opcode: %d %d", key[a1[6]], key[a1[6] + 1]);
            if (key[a1[6] + 1])
            {
                switch (key[a1[6] + 1])
                {
                case 1u:
                    reg[++a1[7]] = a1[0];
                    break;
                case 2u:
                    reg[++a1[7]] = a1[2];
                    break;
                case 3u:
                    reg[++a1[7]] = a1[3];
                    break;
                }
            }
            else
            {
                reg[++a1[7]] = a1[0];
            }
            a1[6] = a1[6] + 2;
            break;
        case 2u:
            printf("run opcode: %d %d", key[a1[6]], key[a1[6] + 1]);
            if (key[a1[6] + 1])
            {
                switch (key[a1[6] + 1])
                {
                case 1u:
                    a1[1] = reg[a1[7]--];
                    break;
                case 2u:
                    a1[2] = reg[a1[7]--];
                    break;
                case 3u:
                    a1[3] = reg[a1[7]--];
                    break;
                }
            }
            else
            {
                a1[0] = reg[a1[7]--];
            }
            a1[6] = a1[6] + 2;
            break;
        case 3u:
            printf("run opcode: %d %d %d %d", key[a1[6]], key[a1[6] + 1],
key[a1[6] + 2], key[a1[6] + 3]);
            switch (key[a1[6] + 1])
```

```
            {
            case 0u:
                printf(", a1[%d] = a1[%d] + a1[%d]", key[a1[6] + 2], key[a1[6] +
2], key[a1[6] + 3]);
                a1[key[a1[6] + 2]] += a1[key[a1[6] + 3]];
                break;
            case 1u:
                printf(", a1[%d] = a1[%d] - a1[%d]", key[a1[6] + 2], key[a1[6] +
2], key[a1[6] + 3]);
                a1[key[a1[6] + 2]] -= a1[key[a1[6] + 3]];
                break;
            case 2u:
                printf(", a1[%d] = a1[%d] * a1[%d]", key[a1[6] + 2], key[a1[6] +
2], key[a1[6] + 3]);
                a1[key[a1[6] + 2]] *= a1[key[a1[6] + 3]];
                break;
            case 3u:
                printf(", a1[%d] = a1[%d] ^ a1[%d]", key[a1[6] + 2], key[a1[6] +
2], key[a1[6] + 3]);
                a1[key[a1[6] + 2]] ^= a1[key[a1[6] + 3]];
                break;
            case 4u:
                printf(", a1[%d] = (a1[%d] << a1[%d])&0xFF00", key[a1[6] + 2],
key[a1[6] + 2], key[a1[6] + 3]);
                a1[key[a1[6] + 2]] <<= a1[key[a1[6] + 3]];
                a1[key[a1[6] + 2]] &= 0xFF00u;
                break;
            case 5u:
                printf(", a1[%d] = a1[%d] >> a1[%d]", key[a1[6] + 2], key[a1[6]
+ 2], key[a1[6] + 3]);
                a1[key[a1[6] + 2]] >>= a1[key[a1[6] + 3]];
                break;
            default:
                break;
            }
            a1[6] = a1[6] + 4;
            printf(", a1[0~3] = %d, %d, %d, %d", a1[0], a1[1], a1[2], a1[3]);
            break;
        case 4u:
            printf("run opcode: %d", key[a1[6]]);
            if (a1[0] == a1[1])
                a1[8] = 0;
            if (a1[0] != a1[1])
                a1[8] = 1;
            a1[6] = a1[6] + 1;
            break;
        case 5u:
            printf("run opcode: %d, ", key[a1[6]]);
            printf("change the opcode position from %d to %d, func5 count: %d",
a1[6], key[a1[6] + 1], count_func5++);
            a1[6] = key[a1[6] + 1];
            break;
        case 6u:
            printf("run opcode: %d, ", key[a1[6]]);
            if (a1[8]) {
```

```
                printf("change the opcode position from %d to %d", a1[6], a1[6]
+ 2);
                a1[6] = (a1[6] + 2);
            }
            else {
                printf("change the opcode position from %d to %d", a1[6],
key[a1[6] + 1]);
                a1[6] = key[a1[6] + 1];
            }

            break;
        case 7u:
            printf("run opcode: %d, ", key[a1[6]]);
            if (a1[8]) {
                printf("change the opcode position from %d to %d", a1[6],
key[a1[6] + 1]);
                a1[6] = key[a1[6] + 1];
            }
            else {
                printf("change the opcode position from %d to %d", a1[6], a1[6]
+ 2);
                a1[6] = (a1[6] + 2);
            }
            break;
        default:
            break;
        }
        printf("\n");
    }
}
```

部分输出如下

```
count:1, call func0, run opcode: 0 3 2 0, a1[2] = 0, a1[0~3] = 0, 0, 0, 0
count:2, call func3, run opcode: 3 0 2 3, a1[2] = a1[2] + a1[3], a1[0~3] = 0, 0,
0, 0
count:3, call func0, run opcode: 0 0 0 0, a1[0] = input[0], a1[0~3] = 48, 0, 0,
0
count:4, call func0, run opcode: 0 2 1 0, a1[1] = a1[0], a1[0~3] = 48, 48, 0, 0
count:5, call func0, run opcode: 0 3 2 50, a1[2] = 50, a1[0~3] = 48, 48, 50, 0
count:6, call func3, run opcode: 3 0 2 3, a1[2] = a1[2] + a1[3], a1[0~3] = 48,
48, 50, 0
count:7, call func0, run opcode: 0 0 0 0, a1[0] = input[50], a1[0~3] = 155, 48,
50, 0
count:8, call func3, run opcode: 3 0 1 0, a1[1] = a1[1] + a1[0], a1[0~3] = 155,
203, 50, 0
count:9, call func0, run opcode: 0 3 2 100, a1[2] = 100, a1[0~3] = 155, 203, 100,
0
count:10, call func3, run opcode: 3 0 2 3, a1[2] = a1[2] + a1[3], a1[0~3] = 155,
203, 100, 0
count:11, call func0, run opcode: 0 0 0 0, a1[0] = input[100], a1[0~3] = 201,
203, 100, 0
count:12, call func3, run opcode: 3 3 1 0, a1[1] = a1[1] ^ a1[0], a1[0~3] = 201,
2, 100, 0
count:13, call func0, run opcode: 0 3 0 8, a1[0] = 8, a1[0~3] = 8, 2, 100, 0
```

```
count:14, call func0, run opcode: 0 2 2 1, a1[2] = a1[1], a1[0~3] = 8, 2, 2, 0
count:15, call func3, run opcode: 3 4 1 0, a1[1] = (a1[1] << a1[0])&0xFF00,
a1[0~3] = 8, 512, 2, 0
count:16, call func3, run opcode: 3 5 2 0, a1[2] = a1[2] >> a1[0], a1[0~3] = 8,
512, 0, 0
count:17, call func3, run opcode: 3 0 1 2, a1[1] = a1[1] + a1[2], a1[0~3] = 8,
512, 0, 0
count:18, call func0, run opcode: 0 2 0 1, a1[0] = a1[1], a1[0~3] = 512, 512, 0,
0
count:19, call func1, run opcode: 1 0
count:20, call func0, run opcode: 0 3 0 1, a1[0] = 1, a1[0~3] = 1, 512, 0, 0
count:21, call func3, run opcode: 3 0 3 0, a1[3] = a1[3] + a1[0], a1[0~3] = 1,
512, 0, 1
count:22, call func0, run opcode: 0 2 0 3, a1[0] = a1[3], a1[0~3] = 1, 512, 0, 1
count:23, call func0, run opcode: 0 3 1 40, a1[1] = 40, a1[0~3] = 1, 40, 0, 1
count:24, call func4, run opcode: 4
count:25, call func6, run opcode: 6, change the opcode position from 91 to 93
count:26, call func5, run opcode: 5, change the opcode position from 93 to 0,
func5 count: 1
count:27, call func0, run opcode: 0 3 2 0, a1[2] = 0, a1[0~3] = 1, 40, 0, 1
count:28, call func3, run opcode: 3 0 2 3, a1[2] = a1[2] + a1[3], a1[0~3] = 1,
40, 1, 1
count:29, call func0, run opcode: 0 0 0 0, a1[0] = input[1], a1[0~3] = 49, 40, 1,
1
count:30, call func0, run opcode: 0 2 1 0, a1[1] = a1[0], a1[0~3] = 49, 49, 1, 1
count:31, call func0, run opcode: 0 3 2 50, a1[2] = 50, a1[0~3] = 49, 49, 50, 1
count:32, call func3, run opcode: 3 0 2 3, a1[2] = a1[2] + a1[3], a1[0~3] = 49,
49, 51, 1
count:33, call func0, run opcode: 0 0 0 0, a1[0] = input[51], a1[0~3] = 168, 49,
51, 1
count:34, call func3, run opcode: 3 0 1 0, a1[1] = a1[1] + a1[0], a1[0~3] = 168,
217, 51, 1
count:35, call func0, run opcode: 0 3 2 100, a1[2] = 100, a1[0~3] = 168, 217,
100, 1
count:36, call func3, run opcode: 3 0 2 3, a1[2] = a1[2] + a1[3], a1[0~3] = 168,
217, 101, 1
count:37, call func0, run opcode: 0 0 0 0, a1[0] = input[101], a1[0~3] = 169,
217, 101, 1
count:38, call func3, run opcode: 3 3 1 0, a1[1] = a1[1] ^ a1[0], a1[0~3] = 169,
112, 101, 1
count:39, call func0, run opcode: 0 3 0 8, a1[0] = 8, a1[0~3] = 8, 112, 101, 1
count:40, call func0, run opcode: 0 2 2 1, a1[2] = a1[1], a1[0~3] = 8, 112, 112,
1
count:41, call func3, run opcode: 3 4 1 0, a1[1] = (a1[1] << a1[0])&0xFF00,
a1[0~3] = 8, 28672, 112, 1
count:42, call func3, run opcode: 3 5 2 0, a1[2] = a1[2] >> a1[0], a1[0~3] = 8,
28672, 0, 1
count:43, call func3, run opcode: 3 0 1 2, a1[1] = a1[1] + a1[2], a1[0~3] = 8,
28672, 0, 1
count:44, call func0, run opcode: 0 2 0 1, a1[0] = a1[1], a1[0~3] = 28672, 28672,
0, 1
count:45, call func1, run opcode: 1 0
count:46, call func0, run opcode: 0 3 0 1, a1[0] = 1, a1[0~3] = 1, 28672, 0, 1
count:47, call func3, run opcode: 3 0 3 0, a1[3] = a1[3] + a1[0], a1[0~3] = 1,
28672, 0, 2
```

```
count:48, call func0, run opcode: 0 2 0 3, a1[0] = a1[3], a1[0~3] = 2, 28672, 0,
2
count:49, call func0, run opcode: 0 3 1 40, a1[1] = 40, a1[0~3] = 2, 40, 0, 2
count:50, call func4, run opcode: 4
count:51, call func6, run opcode: 6, change the opcode position from 91 to 93
count:52, call func5, run opcode: 5, change the opcode position from 93 to 0,
func5 count: 2
......
.........
count:1015, call func0, run opcode: 0 3 2 0, a1[2] = 0, a1[0~3] = 39, 40, 0, 39
count:1016, call func3, run opcode: 3 0 2 3, a1[2] = a1[2] + a1[3], a1[0~3] = 39,
40, 39, 39
count:1017, call func0, run opcode: 0 0 0 0, a1[0] = input[39], a1[0~3] = 57, 40,
39, 39
count:1018, call func0, run opcode: 0 2 1 0, a1[1] = a1[0], a1[0~3] = 57, 57, 39,
39
count:1019, call func0, run opcode: 0 3 2 50, a1[2] = 50, a1[0~3] = 57, 57, 50,
39
count:1020, call func3, run opcode: 3 0 2 3, a1[2] = a1[2] + a1[3], a1[0~3] = 57,
57, 89, 39
count:1021, call func0, run opcode: 0 0 0 0, a1[0] = input[89], a1[0~3] = 87, 57,
89, 39
count:1022, call func3, run opcode: 3 0 1 0, a1[1] = a1[1] + a1[0], a1[0~3] = 87,
144, 89, 39
count:1023, call func0, run opcode: 0 3 2 100, a1[2] = 100, a1[0~3] = 87, 144,
100, 39
count:1024, call func3, run opcode: 3 0 2 3, a1[2] = a1[2] + a1[3], a1[0~3] = 87,
144, 139, 39
count:1025, call func0, run opcode: 0 0 0 0, a1[0] = input[139], a1[0~3] = 156,
144, 139, 39
count:1026, call func3, run opcode: 3 3 1 0, a1[1] = a1[1] ^ a1[0], a1[0~3] =
156, 12, 139, 39
count:1027, call func0, run opcode: 0 3 0 8, a1[0] = 8, a1[0~3] = 8, 12, 139, 39
count:1028, call func0, run opcode: 0 2 2 1, a1[2] = a1[1], a1[0~3] = 8, 12, 12,
39
count:1029, call func3, run opcode: 3 4 1 0, a1[1] = (a1[1] << a1[0])&0xFF00,
a1[0~3] = 8, 3072, 12, 39
count:1030, call func3, run opcode: 3 5 2 0, a1[2] = a1[2] >> a1[0], a1[0~3] = 8,
3072, 0, 39
count:1031, call func3, run opcode: 3 0 1 2, a1[1] = a1[1] + a1[2], a1[0~3] = 8,
3072, 0, 39
count:1032, call func0, run opcode: 0 2 0 1, a1[0] = a1[1], a1[0~3] = 3072, 3072,
0, 39
count:1033, call func1, run opcode: 1 0
count:1034, call func0, run opcode: 0 3 0 1, a1[0] = 1, a1[0~3] = 1, 3072, 0, 39
count:1035, call func3, run opcode: 3 0 3 0, a1[3] = a1[3] + a1[0], a1[0~3] = 1,
3072, 0, 40
count:1036, call func0, run opcode: 0 2 0 3, a1[0] = a1[3], a1[0~3] = 40, 3072,
0, 40
count:1037, call func0, run opcode: 0 3 1 40, a1[1] = 40, a1[0~3] = 40, 40, 0,
40
count:1038, call func4, run opcode: 4
count:1039, call func6, run opcode: 6, change the opcode position from 91 to 95
count:1040, call func0, run opcode: 0 3 3 0, a1[3] = 0, a1[0~3] = 40, 40, 0, 0
count:1041, call func2, run opcode: 2 1
```

```
count:1042, call func0, run opcode: 0 3 2 150, a1[2] = 150, a1[0~3] = 40, 3072,
150, 0
count:1043, call func3, run opcode: 3 0 2 3, a1[2] = a1[2] + a1[3], a1[0~3] = 40,
3072, 150, 0
count:1044, call func0, run opcode: 0 0 0 0, a1[0] = input[150], a1[0~3] = 18432,
3072, 150, 0
count:1045, call func4, run opcode: 4
count:1046, call func7, run opcode: 7, change the opcode position from 114 to
136
```

用python简化一下

```
a = [0x9b, 0xa8, 0x2, 0xbc, 0xac, 0x9c, 0xce, 0xfa, 0x2, 0xb9, 0xff, 0x3a, 0x74,
0x48, 0x19, 0x69, 0xe8, 0x3, 0xcb,
     0xc9, 0xff, 0xfc, 0x80, 0xd6, 0x8d, 0xd7, 0x72, 0x0, 0xa7, 0x1d, 0x3d,
0x99, 0x88, 0x99, 0xbf, 0xe8, 0x96, 0x2e,
     0x5d, 0x57, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0]
b = [0xc9, 0xa9, 0xbd, 0x8b, 0x17, 0xc2, 0x6e, 0xf8, 0xf5, 0x6e, 0x63, 0x63,
0xd5, 0x46, 0x5d, 0x16, 0x98, 0x38, 0x30,
     0x73, 0x38, 0xc1, 0x5e, 0xed, 0xb0, 0x29, 0x5a, 0x18, 0x40, 0xa7, 0xfd,
0xa, 0x1e, 0x78, 0x8b, 0x62, 0xdb, 0xf,
     0x8f, 0x9c, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0]
c = [0x4800, 0xf100, 0x4000, 0x2100, 0x3501, 0x6400, 0x7801, 0xf900, 0x1801,
0x5200, 0x2500, 0x5d01, 0x4700, 0xfd00,
     0x6901, 0x5c00, 0xaf01, 0xb200, 0xec01, 0x5201, 0x4f01, 0x1a01, 0x5000,
0x8501, 0xcd00, 0x2300, 0xf800, 0xc00,
     0xcf00, 0x3d01, 0x4501, 0x8200, 0xd201, 0x2901, 0xd501, 0x601, 0xa201,
0xde00, 0xa601, 0xca01, 0x0, 0x0, 0x0, 0x0,
     0x0, 0x0, 0x0, 0x0, 0x0, 0x0]
flag = '01234567890123456789012345678901234567890123456789'
res = [0 for i in range(40)]
for i in range(40):
    x = a[i] + ord(flag[i])
    y = x ^ b[i]
    z = (y << 8) & 0xff00
    m = y >> 8
    n = z + m
```

写个exp

```
a = [0x9b, 0xa8, 0x2, 0xbc, 0xac, 0x9c, 0xce, 0xfa, 0x2, 0xb9, 0xff, 0x3a, 0x74,
0x48, 0x19, 0x69, 0xe8, 0x3, 0xcb,
     0xc9, 0xff, 0xfc, 0x80, 0xd6, 0x8d, 0xd7, 0x72, 0x0, 0xa7, 0x1d, 0x3d,
0x99, 0x88, 0x99, 0xbf, 0xe8, 0x96, 0x2e,
     0x5d, 0x57]
b = [0xc9, 0xa9, 0xbd, 0x8b, 0x17, 0xc2, 0x6e, 0xf8, 0xf5, 0x6e, 0x63, 0x63,
0xd5, 0x46, 0x5d, 0x16, 0x98, 0x38, 0x30,
     0x73, 0x38, 0xc1, 0x5e, 0xed, 0xb0, 0x29, 0x5a, 0x18, 0x40, 0xa7, 0xfd,
0xa, 0x1e, 0x78, 0x8b, 0x62, 0xdb, 0xf,
     0x8f, 0x9c]
c = [0x4800, 0xf100, 0x4000, 0x2100, 0x3501, 0x6400, 0x7801, 0xf900, 0x1801,
0x5200, 0x2500, 0x5d01, 0x4700, 0xfd00,
     0x6901, 0x5c00, 0xaf01, 0xb200, 0xec01, 0x5201, 0x4f01, 0x1a01, 0x5000,
0x8501, 0xcd00, 0x2300, 0xf800, 0xc00,
```

```
    0xcf00, 0x3d01, 0x4501, 0x8200, 0xd201, 0x2901, 0xd501, 0x601, 0xa201,
0xde00, 0xa601, 0xca01]
c = [i for i in reversed(c)]
flag=''
for i in range(40):
    m = c[i] >> 8
    n = (c[i] & 0xff) << 8
    y = m + n
    y = y ^ b[i]
    y = y - a[i]
    flag += chr(y)
print(flag)# hgame{y0ur_rever5e_sk1ll_i5_very_g0od!!}
```

shellcode

把base64字符串解码

```
import base64
with open('data','wb') as f:

 f.write(base64.b64decode('VUiD7FBIjWwkIEiJTUBIi0VAiwCJRQC4BAAAAEgDRUCLAIlFBMdFC
AAAAADHRQwj782rx0UQFgAAAMdFFCEAAADHRRgsAAAAx0UcNwAAAMdFIAAAAACLRSCD+CBzWotFDANFC
IlFCItFBMHgBANFEItVCANVBDPCi1UEweoFA1UUM8IDRQCJRQCLRQDB4AQDRRiLVQgDVQAzwotVAMHqB
QNVHDPCA0UEiUUEuAEAAAADRSCJRSDrnkiLRUCLVQCJELgEAAAASANFQItVBIkQSI1lMF3D'))
```

然后用ida打开`data`文件

```
_DWORD *__fastcall sub_0(unsigned int *a1)
{
  _DWORD *result; // rax
  unsigned int v2; // [rsp+20h] [rbp+0h]
  unsigned int v3; // [rsp+24h] [rbp+4h]
  int v4; // [rsp+28h] [rbp+8h]
  unsigned int i; // [rsp+40h] [rbp+20h]

  v2 = *a1;
  v3 = a1[1];
  v4 = 0;
  for ( i = 0; i < 0x20; ++i )
  {
    v4 -= 0x54321ODD;
    v2 += ((v3 >> 5) + 33) ^ (v3 + v4) ^ (16 * v3 + 22);
    v3 += ((v2 >> 5) + 55) ^ (v2 + v4) ^ (16 * v2 + 44);
  }
  *a1 = v2;
  result = a1 + 1;
  a1[1] = v3;
  return result;
}
```

看来是普通的tea加密

```python
from ctypes import *


def encrypt(v, k):
    v0, v1 = c_uint32(v[0]), c_uint32(v[1])
    delta = 0x54321ODD
    k0, k1, k2, k3 = k[0], k[1], k[2], k[3]

    total = c_uint32(0)
    for i in range(32):
        total.value -= delta
        v0.value += ((v1.value << 4) + k0) ^ (v1.value + total.value) ^
((v1.value >> 5) + k1)
        v1.value += ((v0.value << 4) + k2) ^ (v0.value + total.value) ^
((v0.value >> 5) + k3)

    return v0.value, v1.value


def decrypt(v, k):
    v0, v1 = c_uint32(v[0]), c_uint32(v[1])
    delta = 0x54321ODD
    k0, k1, k2, k3 = k[0], k[1], k[2], k[3]

    total = c_uint32(-delta * 32)
    for i in range(32):
        v1.value -= ((v0.value << 4) + k2) ^ (v0.value + total.value) ^
((v0.value >> 5) + k3)
        v0.value -= ((v1.value << 4) + k0) ^ (v1.value + total.value) ^
((v1.value >> 5) + k1)
        total.value += delta

    return v0.value, v1.value


# test
if __name__ == "__main__":
    # 待加密的明文，两个32位整型，即64bit的明文数据
    value = [0, 0]
    # 四个key，每个是32bit，即密钥长度为128bit
    key = [22, 33, 44, 55]
    with open('flag.enc', 'rb') as f:
        s = f.read()
    for i in range(0, len(s), 8):
        value[0] = (s[i + 3] << 24) + (s[i + 2] << 16) + (s[i + 1] << 8) + s[i]
        value[1] = (s[i + 7] << 24) + (s[i + 6] << 16) + (s[i + 5] << 8) + s[i +
4]
        res = decrypt(value, key)
        bytearray.fromhex(hex(res[0])[2::]).decode()
        print(bytearray.fromhex(hex(res[0])[2::]).decode()[::-1],sep='', end='')
        print(bytearray.fromhex(hex(res[1])[2::]).decode()[::-1],sep='', end='')
# hgame{th1s_1s_th3_tutu's_hOmewOrk}
```