Re

math-easy

用ida打开，发现有一个math-part函数。分析大概就是解方程了。\
化简脚本：

```
with open('C:\\Users\\a1516\\Desktop\\math_easy.txt', 'r') as f:
    str = f.readlines()
    #print(str)
str3=[]
str_box=''
for i in range(len(str)):
    str_box1=str[i]
    for j in range(len(str_box1)):
        if str_box1[j] == ' ':
            continue
        else:
            break
    str_box2=str_box1[j:]
    str3.append(str_box2)
    str_box2=''
#print(str3)
str4=[]
for i in range(len(str3)-1):
    str_box1=str3[i]
    str_box2=str3[i+1]
    if str_box2[0]=='+':
        str_box1=str_box1[:-1]
    str4.append(str_box1)
    str_box1=''
str4.append(str3[len(str3)-1])
str5=''
str5=''.join(str4)

str5=str5.replace('|| (','')
for i in range(len(str5)):
            if str5[i:i+4]=='<< 6':
                box=str5[i-8:i]
                str5=str5[:i-8]+ '64 * '+box+str5[i+5:]
str5=str5.replace('* *v34','* v34[0]')
str5=str5.replace('(','')
str5=str5.replace(')','')
str5=str5.replace(',\n',' + ')
str5=str5.replace('!=','==')
str5=str5.replace('if ','')
str5=str5.replace('goto LABEL_37;','')
str5=str5.replace('v34','flag')
str5=str5.replace(';\n',' + ')
#print(str5)
str7=[]
str7=str5.split('\n')
str7.remove('')
#print(str7)
for i in range(len(str7)):
    str6=str7[i]
    str6+='\n'
    str6=str6[5:]
    for j in range(len(str6)):
        if str6[j]=='v':
            str8=str6[:j]+str6[j+5:]
    str7[i]='solver.add('+str8[:-1]+')\n'

print(''.join(str7))
```

输出结果:

```
solver.add(76 * flag[21]+ 31 * flag[9]+ 87 * flag[28]+ 54 * flag[2]+ 74 * flag[5]+ 99 * flag[26]+ 94 * flag[3]+ 84 * flag[19]+ 32 * flag[15]+ 90 * flag[27]+ 16 * flag[14]+ 19 * flag[8]+
solver.add(55 * flag[6]+ 38 * flag[9]+ 39 * flag[18]+ 73 * flag[24]+ 86 * flag[13]+ 18 * flag[11]+ 40 * flag[21]+ 40 * flag[26]+ 54 * flag[14]+ 81 * flag[10]+ 71 * flag[27]+ 20 * flag[8]
solver.add(53 * flag[10]+ 82 * flag[14]+ 70 * flag[5]+ 84 * flag[2]+ 57 * flag[19]+ 92 * flag[27]+ 57 * flag[11]+ 77 * flag[4]+ 49 * flag[8]+ 62 * flag[29]+ 97 * flag[22]+ 47 * flag[1]+
solver.add(53 * flag[10]+ 82 * flag[14]+ 70 * flag[5]+ 84 * flag[2]+ 57 * flag[19]+ 92 * flag[27]+ 57 * flag[11]+ 77 * flag[4]+ 49 * flag[8]+ 62 * flag[29]+ 97 * flag[22]+ 47 * flag[1]+
solver.add(14 * flag[24]+ 46 * flag[6]+ 56 * flag[7]+ 13 * flag[2]+ 82 * flag[11]+ 49 * flag[30]+ 97 * flag[18]+ 50 * flag[14]+ 83 * flag[27]+ 38 * flag[13]+ 49 * flag[29]+ 9 * flag[4]+
solver.add(14 * flag[24]+ 46 * flag[6]+ 56 * flag[7]+ 13 * flag[2]+ 82 * flag[11]+ 49 * flag[30]+ 97 * flag[18]+ 50 * flag[14]+ 83 * flag[27]+ 38 * flag[13]+ 49 * flag[29]+ 9 * flag[4]+
solver.add(9 * flag[28]+ 63 * flag[5]+ 20 * flag[4]+ 96 * flag[8]+ 39 * flag[11]+ 91 * flag[1]+ 40 * flag[9]+ 85 * flag[14]+ 62 * flag[16]+ 95 * flag[19]+ 34 * flag[22]+ 67 * flag[31]+ 5
solver.add(9 * flag[28]+ 63 * flag[5]+ 20 * flag[4]+ 96 * flag[8]+ 39 * flag[11]+ 91 * flag[1]+ 40 * flag[9]+ 85 * flag[14]+ 62 * flag[16]+ 95 * flag[19]+ 34 * flag[22]+ 67 * flag[31]+ 5
solver.add(88 * flag[9]+ 48 * flag[4]+ 83 * flag[13]+ 66 * flag[7]+ 60 * flag[30]+ 57 * flag[6]+ 85 * flag[17]+ 71 * flag[28]+ 98 * flag[24]+ 83 * flag[10]+ 12 * flag[1]+ 72 * flag[31]+
solver.add(88 * flag[9]+ 48 * flag[4]+ 83 * flag[13]+ 66 * flag[7]+ 60 * flag[30]+ 57 * flag[6]+ 85 * flag[17]+ 71 * flag[28]+ 98 * flag[24]+ 83 * flag[10]+ 12 * flag[1]+ 72 * flag[31]+
solver.add(57 * flag[21]+ 63 * flag[12]+ 4 * flag[14]+ 59 * flag[31]+ 15 * flag[23]+ 12 * flag[25]+ 58 * flag[5]+ 40 * flag[4]+ 26 * flag[30]+ 8 * flag[15]+ 25 * flag[6]+ 97 * flag[10]+
solver.add(57 * flag[21]+ 63 * flag[12]+ 4 * flag[14]+ 59 * flag[31]+ 15 * flag[23]+ 12 * flag[25]+ 58 * flag[5]+ 40 * flag[4]+ 26 * flag[30]+ 8 * flag[15]+ 25 * flag[6]+ 97 * flag[10]+
solver.add(86 * flag[17]+ 74 * flag[0]+ 72 * flag[4]+ 27 * flag[20]+ 88 * flag[9]+ 64 * flag[21] + 52 * flag[15]+ 4 * flag[19]+ 8 * flag[1]+ 16 * flag[13]+ 54 * flag[25]+ 8 * flag[29]+ 5
solver.add(86 * flag[17]+ 74 * flag[0]+ 72 * flag[4]+ 27 * flag[20]+ 88 * flag[9]+ 64 * flag[21] + 52 * flag[15]+ 4 * flag[19]+ 8 * flag[1]+ 16 * flag[13]+ 54 * flag[25]+ 8 * flag[29]+ 5
solver.add(51 * flag[7]+ 42 * flag[4]+ 78 * flag[8]+ 45 * flag[25]+ 63 * flag[30]+ 85 * flag[26]+ 30 * flag[29]+ 83 * flag[14]+ 62 * flag[31]+ 71 * flag[22]+ 45 * flag[17]+ 64 * flag[6]
solver.add(51 * flag[7]+ 42 * flag[4]+ 78 * flag[8]+ 45 * flag[25]+ 63 * flag[30]+ 85 * flag[26]+ 30 * flag[29]+ 83 * flag[14]+ 62 * flag[31]+ 71 * flag[22]+ 45 * flag[17]+ 64 * flag[6]
solver.add( 13 * flag[29]+ 11 * flag[22]+ 41 * flag[5]+ 38 * flag[13]+ 90 * flag[31]+ 68 * flag[7]+ 56 * flag[14]+ 4 * flag[23]+ 66 * flag[28]+ 28 * flag[1]+ 6 * flag[12]+ 91 * flag[16]+
solver.add( 13 * flag[29]+ 11 * flag[22]+ 41 * flag[5]+ 38 * flag[13]+ 90 * flag[31]+ 68 * flag[7]+ 56 * flag[14]+ 4 * flag[23]+ 66 * flag[28]+ 28 * flag[1]+ 6 * flag[12]+ 91 * flag[16]+
solver.add( 23 * flag[25]+ 32 * flag[3]+ 72 * flag[15]+ 41 * flag[26]+ 33 * flag[30]+ 82 * flag[13]+ 20 * flag[0]+ 7 * flag[12]+ 25 * flag[29]+ 39 * flag[21]+ 57 * flag[14]+ 14 * flag[16
solver.add( 23 * flag[25]+ 32 * flag[3]+ 72 * flag[15]+ 41 * flag[26]+ 33 * flag[30]+ 82 * flag[13]+ 20 * flag[0]+ 7 * flag[12]+ 25 * flag[29]+ 39 * flag[21]+ 57 * flag[14]+ 14 * flag[16
solver.add( 41 * flag[24]+ 45 * flag[30]+ 82 * flag[20]+ 86 * flag[19]+ 99 * flag[9]+ 96 * flag[22]+ 85 * flag[28]+ 70 * flag[5]+ 77 * flag[23]+ 80 * flag[11]+ 40 * flag[31]+ 66 * flag[1
solver.add( 41 * flag[24]+ 45 * flag[30]+ 82 * flag[20]+ 86 * flag[19]+ 99 * flag[9]+ 96 * flag[22]+ 85 * flag[28]+ 70 * flag[5]+ 77 * flag[23]+ 80 * flag[11]+ 40 * flag[31]+ 66 * flag[1
solver.add( 42 * flag[31]+ 95 * flag[30]+ 58 * flag[8]+ 47 * flag[13]+ 65 * flag[15]+ 24 * flag[17]+ 97 * flag[10]+ 24 * flag[21]+ 28 * flag[0]+ 77 * flag[5]+ 97 * flag[6]+ 24 * flag[26]
solver.add( 42 * flag[31]+ 95 * flag[30]+ 58 * flag[8]+ 47 * flag[13]+ 65 * flag[15]+ 24 * flag[17]+ 97 * flag[10]+ 24 * flag[21]+ 28 * flag[0]+ 77 * flag[5]+ 97 * flag[6]+ 24 * flag[26]
solver.add( flag[1]+ 88 * flag[3]+ 90 * flag[0]+ 4 * flag[23]+ 46 * flag[7]+ 54 * flag[16]+ 16 * flag[6]+ 89 * flag[22]+ 76 * flag[27]+ 38 * flag[17]+ 3 * flag[5]+ 70 * flag[14]+ 3 * fla
solver.add( flag[1]+ 88 * flag[3]+ 90 * flag[0]+ 4 * flag[23]+ 46 * flag[7]+ 54 * flag[16]+ 16 * flag[6]+ 89 * flag[22]+ 76 * flag[27]+ 38 * flag[17]+ 3 * flag[5]+ 70 * flag[14]+ 3 * fla
solver.add( 27 * flag[18]+ 48 * flag[4]+ 13 * flag[20]+ 44 * flag[10]+ 70 * flag[12]+ 44 * flag[17]+ 22 * flag[23]+ 55 * flag[14]+ 73 * flag[26]+ 55 * flag[8]+ 58 * flag[11]+ 31 * flag[3
solver.add( 27 * flag[18]+ 48 * flag[4]+ 13 * flag[20]+ 44 * flag[10]+ 70 * flag[12]+ 44 * flag[17]+ 22 * flag[23]+ 55 * flag[14]+ 73 * flag[26]+ 55 * flag[8]+ 58 * flag[11]+ 31 * flag[3
solver.add( 31 * flag[12]+ 35 * flag[10]+ 54 * flag[20]+ 26 * flag[29]+ 29 * flag[3]+ 2 * flag[23]+ 46 * flag[0]+ 30 * flag[26]+ 56 * flag[27]+ 100 * flag[11]+ 43 * flag[1]+ 15 * flag[4]
solver.add( 31 * flag[12]+ 35 * flag[10]+ 54 * flag[20]+ 26 * flag[29]+ 29 * flag[3]+ 2 * flag[23]+ 46 * flag[0]+ 30 * flag[26]+ 56 * flag[27]+ 100 * flag[11]+ 43 * flag[1]+ 15 * flag[4]
solver.add( 92 * flag[20]+ 43 * flag[23]+ 16 * flag[19]+ 92 * flag[5]+ 49 * flag[26]+ 44 * flag[2]+ 26 * flag[29]+ 64 * flag[25] + 45 * flag[24]+ 99 * flag[11]+ 43 * flag[4]+ 75 * flag[2
solver.add( 92 * flag[20]+ 43 * flag[23]+ 16 * flag[19]+ 92 * flag[5]+ 49 * flag[26]+ 44 * flag[2]+ 26 * flag[29]+ 64 * flag[25] + 45 * flag[24]+ 99 * flag[11]+ 43 * flag[4]+ 75 * flag[2
solver.add( 68 * flag[2]+ 83 * flag[10]+ 47 * flag[5]+ 85 * flag[13]+ 22 * flag[8]+ 92 * flag[27]+ 75 * flag[28]+ 43 * flag[3]+ 29 * flag[22]+ 92 * flag[0]+ 54 * flag[16]+ 17 * flag[30]+
solver.add( 68 * flag[2]+ 83 * flag[10]+ 47 * flag[5]+ 85 * flag[13]+ 22 * flag[8]+ 92 * flag[27]+ 75 * flag[28]+ 43 * flag[3]+ 29 * flag[22]+ 92 * flag[0]+ 54 * flag[16]+ 17 * flag[30]+
solver.add( 77 * flag[9]+ 56 * flag[30]+ 79 * flag[2]+ 71 * flag[29]+ 95 * flag[28]+ 87 * flag[24]+ 62 * flag[16]+ 85 * flag[26]+ 43 * flag[20]+ 67 * flag[15]+ 97 * flag[8]+ 80 * flag[0]
```

```
solver.add( 77 * flag[9]+ 56 * flag[30]+ 79 * flag[2]+ 71 * flag[29]+ 95 * flag[28]+ 87 * flag[24]+ 62 * flag[16]+ 85 * flag[26]+ 43 * flag[20]+ 67 * flag[15]+ 97 * flag[8]+ 80 * flag[0]
solver.add( 81 * flag[30]+ 21 * flag[6]+ 72 * flag[11]+ 48 * flag[18]+ 2 * flag[19]+ 42 * flag[10]+ 22 * flag[24]+ 99 * flag[2]+ 78 * flag[22]+ 83 * flag[12]+ 60 * flag[9]+ 59 * flag[13]
solver.add( 81 * flag[30]+ 21 * flag[6]+ 72 * flag[11]+ 48 * flag[18]+ 2 * flag[19]+ 42 * flag[10]+ 22 * flag[24]+ 99 * flag[2]+ 78 * flag[22]+ 83 * flag[12]+ 60 * flag[9]+ 59 * flag[13]
solver.add( 53 * flag[27]+ 52 * flag[29]+ 70 * flag[22]+ 35 * flag[30]+ 50 * flag[16]+ 59 * flag[8]+ 75 * flag[10]+ 55 * flag[20]+ 23 * flag[0]+ 52 * flag[17]+ 47 * flag[3]+ 91 * flag[13
solver.add( 53 * flag[27]+ 52 * flag[29]+ 70 * flag[22]+ 35 * flag[30]+ 50 * flag[16]+ 59 * flag[8]+ 75 * flag[10]+ 55 * flag[20]+ 23 * flag[0]+ 52 * flag[17]+ 47 * flag[3]+ 91 * flag[13
solver.add( 80 * flag[21]+ 43 * flag[31]+ 67 * flag[16]+ 55 * flag[13]+ 95 * flag[24]+ 46 * flag[28]+ 93 * flag[5]+ 75 * flag[20]+ 14 * flag[25]+ 24 * flag[26]+ 50 * flag[29]+ 70 * flag[
solver.add( 80 * flag[21]+ 43 * flag[31]+ 67 * flag[16]+ 55 * flag[13]+ 95 * flag[24]+ 46 * flag[28]+ 93 * flag[5]+ 75 * flag[20]+ 14 * flag[25]+ 24 * flag[26]+ 50 * flag[29]+ 70 * flag[
solver.add( 27 * flag[11]+ 40 * flag[8]+ 53 * flag[15]+ 40 * flag[18]+ 56 * flag[3]+ 2 * flag[2]+ 32 * flag[4]+ 90 * flag[1]+ 54 * flag[16]+ 20 * flag[9]+ 86 * flag[17]+ 82 * flag[31]+ 4
solver.add( 27 * flag[11]+ 40 * flag[8]+ 53 * flag[15]+ 40 * flag[18]+ 56 * flag[3]+ 2 * flag[2]+ 32 * flag[4]+ 90 * flag[1]+ 54 * flag[16]+ 20 * flag[9]+ 86 * flag[17]+ 82 * flag[31]+ 4
solver.add( flag[27]+ 66 * flag[18]+ 40 * flag[17]+ 17 * flag[0]+ 27 * flag[19]+ 26 * flag[31]+ 57 * flag[24]+ 35 * flag[3]+ 80 * flag[1]+ 67 * flag[5]+ 85 * flag[6]+ 7 * flag[15]+ 93 *
solver.add( flag[27]+ 66 * flag[18]+ 40 * flag[17]+ 17 * flag[0]+ 27 * flag[19]+ 26 * flag[31]+ 57 * flag[24]+ 35 * flag[3]+ 80 * flag[1]+ 67 * flag[5]+ 85 * flag[6]+ 7 * flag[15]+ 93 *
solver.add( 71 * flag[16]+ 4 * flag[1]+ 77 * flag[31]+ 83 * flag[2]+ 11 * flag[30]+ 53 * flag[19]+ 85 * flag[12]+ 67 * flag[13]+ 39 * flag[8]+ 45 * flag[24]+ 84 * flag[22]+ 99 * flag[14]
solver.add( 87 * flag[2]+ 86 * flag[24]+ 76 * flag[14]+ 38 * flag[23]+ 85 * flag[3]+ 71 * flag[22]+ 42 * flag[29]+ 85 * flag[30]+ 14 * flag[10]+ 17 * flag[13]+ 42 * flag[25]+ 11 * flag[1
solver.add( 36 * flag[1]+ 60 * flag[3]+ 84 * flag[11]+ 19 * flag[26]+ 76 * flag[27]+ 86 * flag[16]+ 92 * flag[8]+ 96 * flag[14]+ 60 * flag[21]+ 23 * flag[4]+ 60 * flag[12]+ 50 * flag[23]
solver.add( 25 * flag[26]+ 98 * flag[24]+ 15 * flag[6]+ 50 * flag[18]+ 88 * flag[20]+ 74 * flag[11]+ 83 * flag[1]+ 86 * flag[21]+ 52 * flag[7]+ 39 * flag[10]+ 40 * flag[13]+ 82 * flag[28
solver.add( 68 * flag[23]+ 60 * flag[18]+ 93 * flag[20]+ 100 * flag[11]+ 98 * flag[14]+ 32 * flag[3]+ 15 * flag[21]+ 79 * flag[0]+ 6 * flag[24]+ 62 * flag[26]+ 96 * flag[6]+ 68 * flag[22
solver.add( 86 * flag[9]+ 20 * flag[7]+ 29 * flag[16]+ 31 * flag[14]+ 83 * flag[26]+ 11 * flag[4]+ 29 * flag[19]+ 82 * flag[13]+ 84 * flag[10]+ 70 * flag[1]+ 52 * flag[12]+ 40 * flag[6]+
solver.add( 12 * flag[11]+ 82 * flag[24]+ 100 * flag[8]+ 29 * flag[26]+ 97 * flag[12]+ 32 * flag[6]+ 26 * flag[27]+ 46 * flag[19]+ 8 * flag[25]+ 72 * flag[0]+ 16 * flag[17]+ 63 * flag[10
solver.add( 84 * flag[25]+ 91 * flag[10]+ 67 * flag[22]+ 77 * flag[15]+ 23 * flag[26]+ 38 * flag[4]+ 3 * flag[31]+ 76 * flag[13]+ 50 * flag[0]+ 74 * flag[11]+ 45 * flag[28]+ 58 * flag[29
```

放到学长给的代码里跑就完事儿了

Say-Muggle-Code a.k.a. SMC
就照着hint做呗。。。
发现有个flag长度39个字符，中间32个字符被分割成两部分\
前一部分比较简单

```
 >>> data1='0DEh, 0D1h, 0D8h, 8Ch, 8Fh, 0D9h, 0DFh, 0DEh, 0DFh, 8Ch, 0D8h, 0DAh, 8Ch, 0DCh, 0DDh, 0D8h'
>>> data1=data1.replace('0','')
>>> data1=data1.replace('h, ', ', 0x')
>>> data1='0x'+data1[:-1]
>>> data1=data1.split(',')
>>> data=[]
>>> for i in data1:
    data.append(eval(i) ^ 0xE9)


>>> data
[55, 56, 49, 101, 102, 48, 54, 55, 54, 101, 49, 51, 101, 53, 52, 49]
>>> str1=''
>>> for i in data:
    str1+= chr(i)


>>> str1
'781ef0676e13e541'
>>>
```

第二部分主要有一个modify函数，把代码进行了加密
看hint的wp，知道可以写idc脚本，对代码进行还原。
idc脚本：

```
#include <idc.idc>

static main()
{
    auto addr = 0x0000000000603000;   //这里填入要解密字节串的起始地址
    auto i = 0;
    auto v3 = 123;
    for (i = 0; addr + i < 0x0000000000603200; i++)   //循环结束的条件为字节串的结束地址
    {
        PatchByte(addr + i, Byte(addr + i) ^ v3++);   //异或的数字根据情况修改
    }
}
```

还原之后得到encrypt函数，经过学长的明示。。。知道了这个是TEA加密算法\
接下来的事情把我菜哭了，就是不知道啥叫类型。一个DWORD相当于4个字符，4个DWORD正好16个字符。\
心塞塞，菜到不想说话=。=

```
#include <stdio.h>
#include <stdint.h>
#include <string.h>
void decrypt(uint32_t* v, uint32_t* k) {
    uint32_t v0 = v[0], v1 = v[1], sum = 0xC6EF3720, i; /* set up */
    uint32_t delta = 0x9e3779b9; /* a key schedule constant */
    uint32_t k0 = k[0], k1 = k[1], k2 = k[2], k3 = k[3]; /* cache key */
    for (i = 0; i < 32; i++) { /* basic cycle start */
        v1 -= ((v0 << 4) + k2) ^ (v0 + sum) ^ ((v0 >> 5) + k3);
        v0 -= ((v1 << 4) + k0) ^ (v1 + sum) ^ ((v1 >> 5) + k1);
        sum -= delta;
    } /* end cycle */
    v[0] = v0; v[1] = v1;
}

//emmm。。。和上面的是一样的
void decrypt_1(uint32_t* v, uint32_t* k)
{
    uint32_t v0 = v[0], v1 = v[1];
    uint32_t delta = 0x9e3779b9;
    uint32_t k0 = k[0], k1 = k[1], k2 = k[2], k3 = k[3];
    int sum = -0x61C88647 * 32;
    for (int i = 0; i < 32; i++) {
        v1 -= ((v0 << 4) + k2) ^ (v0 + sum) ^ ((v0 >> 5) + k3);
        v0 -= ((v1 << 4) + k0) ^ (v1 + sum) ^ ((v1 >> 5) + k1);
        sum += 0x61C88647;
    }
    v[0] = v0;
```

```
    v[1] = v1;
}


int main()
{
    const char *passwd = "7\x0f\tT\x03V\x06\x01\x01ST\x02VP\x01\x05";
    uint32_t key[4];
    //uint32_t crpto[2] = { 0x1D99DB41 ,0x7449A1CB };
    //uint32_t crpto_1[2] = { 0x1D99DB41 ,0x7449A1CB };
    uint32_t crpto[2] = { 0x5408F12B ,0x49D728B4 };
    uint32_t crpto_1[2] = { 0x5408F12B ,0x49D728B4 };
    memcpy(key, passwd, sizeof(key));
    decrypt_1(crpto_1, key);
    decrypt(crpto, key);
    getchar();
    return 0;
}
```

解密出来是这样的

```
38 34 61 35 34 32 62 35      37 64 65 31 65 34 37 66
84a542b57de1e47f
```

加上前面的str1就是flag了。。。

感谢oyiadin学长的不杀之恩，看来w eek4的real，感觉生命到头了。