

SEA

主函数逻辑是假的，输入8个数字会触发除0异常跳转到真正的流程中

将输入分为两部分，前10位进行异或加密，后16位进行魔改S盒的aes加密

```
m = [0x70, 0x3B, 0x6C, 0x5B, 0x42, 0x6A, 0x7A, 0x5C, 0x43, 0x65]

m[0] = (m[0] ^ 0x16) - 33
m[1] = (m[1] + 18) ^ 0x19
m[3] = (m[3] + 18) ^ 0x19
m[2] = (m[2] ^ 0x16) - 33
m[4] = (m[4] ^ 0x16) - 33

m[5] = (m[5] + 18) ^ 0x19
m[7] = (m[7] + 18) ^ 0x19
m[6] = (m[6] ^ 0x16) - 33
m[8] = (m[8] ^ 0x16) - 33
m[9] = (m[9] + 18) ^ 0x19
print(bytes(m))
# b'ETYt3ekw4n'
```

```
#include <iostream>
#include <cstdlib>
#include <stdio.h>
using namespace std;

unsigned char S[256] = { // S盒
    0x18, 0x79, 0x28, 0xF9, 0x55, 0x99, 0x71, 0xD5, 0x1B, 0xEC,
    0xBB, 0xB0, 0x95, 0x6F, 0x94, 0x70, 0xA3, 0x53, 0x63, 0xAD,
    0x54, 0x7B, 0x37, 0x6E, 0xC1, 0xDB, 0xB1, 0xD7, 0x3D, 0x92,
    0x4D, 0xD0, 0x14, 0x4C, 0xB7, 0x78, 0x62, 0xA0, 0x6A, 0x1A,
    0xCD, 0x00, 0xF1, 0x7D, 0x5E, 0x1E, 0xF5, 0x8D, 0x11, 0x65,
    0xA9, 0xE2, 0xF2, 0xB9, 0xCA, 0x8C, 0xA1, 0xD2, 0x47, 0xAB,
    0x7C, 0x66, 0x52, 0xE4, 0x06, 0x77, 0x89, 0xC6, 0x7E, 0xB3,
    0xAE, 0xE6, 0xB4, 0x8B, 0xDF, 0x1D, 0x23, 0x17, 0xEA, 0x3C,
    0x90, 0xDC, 0x81, 0x32, 0xA5, 0xAF, 0x50, 0x20, 0x5D, 0x2D,
    0x96, 0x42, 0x35, 0x2E, 0x0A, 0xBF, 0xED, 0x8E, 0x38, 0xBA,
    0x61, 0x0B, 0x85, 0x5B, 0x24, 0x6B, 0xF0, 0x21, 0x3F, 0xCE,
    0x2B, 0x22, 0xA8, 0xC5, 0xE1, 0x4A, 0x30, 0x74, 0xEF, 0xCF,
    0xA4, 0xD3, 0xC8, 0xD9, 0xEB, 0xFB, 0xA7, 0xBE, 0x3E, 0x41,
    0xE0, 0xB5, 0x9F, 0xC0, 0xAC, 0x93, 0x9E, 0xF8, 0xF7, 0x7F,
    0xDE, 0x3B, 0xDA, 0x72, 0x88, 0x0D, 0x56, 0xE8, 0xE7, 0x8A,
    0xF4, 0x91, 0x5A, 0x64, 0x19, 0x67, 0x57, 0xD8, 0x84, 0xFA,
    0x0C, 0x25, 0x9B, 0xA2, 0x07, 0x15, 0x04, 0xC4, 0x87, 0x43,
    0x97, 0xB8, 0x60, 0xE3, 0x45, 0xAA, 0x8F, 0x13, 0xFD, 0xCB,
    0x2C, 0xA6, 0x1C, 0x3A, 0xEE, 0x36, 0x7A, 0xE9, 0xD1, 0x09,
    0x39, 0x4E, 0x33, 0xFE, 0x9A, 0x5C, 0x86, 0x6D, 0x16, 0x2F,
    0xD4, 0xB2, 0x48, 0x82, 0x5F, 0x68, 0x29, 0x03, 0xC9, 0x02,
    0x80, 0x44, 0x26, 0xBC, 0xFF, 0x75, 0x9C, 0x46, 0x2A, 0x27,
    0x4F, 0xC2, 0x9D, 0xF6, 0x01, 0x0F, 0x98, 0x40, 0x83, 0xF3,
    0x31, 0xBD, 0x58, 0x4B, 0x05, 0xB6, 0xD6, 0x08, 0xC3, 0x49,
    0x1F, 0x59, 0x10, 0xC7, 0xFC, 0x12, 0xE5, 0xCC, 0x51, 0xDD,
    0x0E, 0x76, 0x69, 0x73, 0x6C, 0x34
```

```

};
unsigned char IS[256] = {                                     // S盒的逆
    41, 224, 209, 207, 166, 234, 64, 164, 237, 189, 94, 101, 160, 145, 250, 225,
    242, 48, 245, 177, 32, 165, 198, 77, 0, 154, 39, 8, 182, 75, 45, 240, 87, 107,
    111, 76, 104, 161, 212, 219, 2, 206, 218, 110, 180, 89, 93, 199, 116, 230, 83,
    192, 255, 92, 185, 22, 98, 190, 183, 141, 79, 28, 128, 108, 227, 129, 91, 169,
    211, 174, 217, 58, 202, 239, 115, 233, 33, 30, 191, 220, 86, 248, 62, 17, 20, 4,
    146, 156, 232, 241, 152, 103, 195, 88, 44, 204, 172, 100, 36, 18, 153, 49, 61,
    155, 205, 252, 38, 105, 254, 197, 23, 13, 15, 6, 143, 253, 117, 215, 251, 65,
    35, 1, 186, 21, 60, 43, 68, 139, 210, 82, 203, 228, 158, 102, 196, 168, 144, 66,
    149, 73, 55, 47, 97, 176, 80, 151, 29, 135, 14, 12, 90, 170, 226, 5, 194, 162,
    216, 222, 136, 132, 37, 56, 163, 16, 120, 84, 181, 126, 112, 50, 175, 59, 134,
    19, 70, 85, 11, 26, 201, 69, 72, 131, 235, 34, 171, 53, 99, 10, 213, 231, 127,
    95, 133, 24, 221, 238, 167, 113, 67, 243, 122, 208, 54, 179, 247, 40, 109, 119,
    31, 188, 57, 121, 200, 7, 236, 27, 157, 123, 142, 25, 81, 249, 140, 74, 130,
    114, 51, 173, 63, 246, 71, 148, 147, 187, 78, 124, 9, 96, 184, 118, 106, 42, 52,
    229, 150, 46, 223, 138, 137, 3, 159, 125, 244, 178, 193, 214
};

unsigned char Mul(unsigned char a, unsigned char b)
{
    unsigned char i;
    unsigned char temp;
    unsigned char result[8];
    unsigned char sum=0;
    result[0] = a;
    for(i=1;i<8;i++)          //x乘法
    {
        temp=result[i-1];    //>>右移, <<左移, ^异或, &与操作 (同1为1)
        temp=temp>>7;        //temp为最高位, 判断 最高位 是0还是1
        if(temp==1)
        {
            result[i]=result[i-1]<<1^0x1b;    // 0x1b=0b 00011011
        }
        else
        {
            result[i]=result[i-1]<<1;
        }
    }
    for(i=0;i<8;i++){        //左移之后补零
        temp=b<<i&0x80;      //判断 b二进制的哪几位为1
        if(temp==0x80)
        {
            sum^=result[7-i];
        }
    }
    return sum;
}

void KeyExpansion(unsigned char K[16], unsigned char k[11][16])
{
    unsigned char RC[10];
    RC[0]=1;
    int i;
    for(i=1;i<10;i++)
        RC[i]=Mul(0x02, RC[i-1]);
    for(i=0;i<16;i++)
        k[0][i]=K[i];
}

```

```

for(i=1;i<11;i++)
{
    k[i][0]=k[i-1][0]^S[k[i-1][13]]^RC[i-1];
    k[i][1]=k[i-1][1]^S[k[i-1][14]];
    k[i][2]=k[i-1][2]^S[k[i-1][15]];
    k[i][3]=k[i-1][3]^S[k[i-1][12]];
    k[i][4]=k[i-1][4]^k[i][0];
    k[i][5]=k[i-1][5]^k[i][1];
    k[i][6]=k[i-1][6]^k[i][2];
    k[i][7]=k[i-1][7]^k[i][3];
    k[i][8]=k[i-1][8]^k[i][4];
    k[i][9]=k[i-1][9]^k[i][5];
    k[i][10]=k[i-1][10]^k[i][6];
    k[i][11]=k[i-1][11]^k[i][7];
    k[i][12]=k[i-1][12]^k[i][8];
    k[i][13]=k[i-1][13]^k[i][9];
    k[i][14]=k[i-1][14]^k[i][10];
    k[i][15]=k[i-1][15]^k[i][11];
}
}

void AddRoundKey( unsigned char *a , unsigned char *key ) { // 轮密钥加
    for( int i = 0 ; i < 16 ; i ++ )
        a[i] ^= key[i] ;
}

void SubBytes( unsigned char *input ) { // S盒字节代换
    for( int i = 0 ; i < 16 ; i ++ )
        input[i] = S[input[i]] ;
}

void InvSubBytes( unsigned char *input ) { // S盒字节代换逆变换
    for( int i = 0 ; i < 16 ; i ++ )
        input[i] = IS[input[i]] ;
}

void ShiftRows( unsigned char *a ) { // 行移位-矩阵按列展开
    unsigned char b[16] ;
    b[ 0] = a[ 0] ; b[ 4] = a[ 4] ; b[ 8] = a[ 8] ; b[12] = a[12] ;
    b[ 1] = a[ 5] ; b[ 5] = a[ 9] ; b[ 9] = a[13] ; b[13] = a[ 1] ;
    b[ 2] = a[10] ; b[ 6] = a[14] ; b[10] = a[ 2] ; b[14] = a[ 6] ;
    b[ 3] = a[15] ; b[ 7] = a[ 3] ; b[11] = a[ 7] ; b[15] = a[11] ;
    for( int i = 0 ; i < 16 ; i ++ )
        a[i] = b[i] ;
}

void InvShiftRows( unsigned char *a ) { // 行移位逆变换
    unsigned char b[16] ;
    b[ 0] = a[ 0] ; b[ 4] = a[ 4] ; b[ 8] = a[ 8] ; b[12] = a[12] ;
    b[ 1] = a[13] ; b[ 5] = a[ 1] ; b[ 9] = a[ 5] ; b[13] = a[ 9] ;
    b[ 2] = a[10] ; b[ 6] = a[14] ; b[10] = a[ 2] ; b[14] = a[ 6] ;
    b[ 3] = a[ 7] ; b[ 7] = a[11] ; b[11] = a[15] ; b[15] = a[ 3] ;
    for( int i = 0 ; i < 16 ; i ++ )
        a[i] = b[i] ;
}

// 列混合
void MixColumns( unsigned char *a ) {
    unsigned char b[16] ;
    b[ 0] = Mul(0x02,a[0]) ^ Mul(0x03,a[1]) ^ a[2] ^ a[3];

```

```

    b[ 1] = Mul(0x02,a[1]) ^ Mul(0x03,a[2]) ^ a[3] ^ a[0];
    b[ 2] = Mul(0x02,a[2]) ^ Mul(0x03,a[3]) ^ a[0] ^ a[1];
    b[ 3] = Mul(0x02,a[3]) ^ Mul(0x03,a[0]) ^ a[1] ^ a[2];
    b[ 4] = Mul(0x02,a[4]) ^ Mul(0x03,a[5]) ^ a[6] ^ a[7];
    b[ 5] = Mul(0x02,a[5]) ^ Mul(0x03,a[6]) ^ a[7] ^ a[4];
    b[ 6] = Mul(0x02,a[6]) ^ Mul(0x03,a[7]) ^ a[4] ^ a[5];
    b[ 7] = Mul(0x02,a[7]) ^ Mul(0x03,a[4]) ^ a[5] ^ a[6];
    b[ 8] = Mul(0x02,a[8]) ^ Mul(0x03,a[9]) ^ a[10] ^ a[11];
    b[ 9] = Mul(0x02,a[9]) ^ Mul(0x03,a[10]) ^ a[11] ^ a[8];
    b[10] = Mul(0x02,a[10]) ^ Mul(0x03,a[11]) ^ a[8] ^ a[9];
    b[11] = Mul(0x02,a[11]) ^ Mul(0x03,a[8]) ^ a[9] ^ a[10];
    b[12] = Mul(0x02,a[12]) ^ Mul(0x03,a[13]) ^ a[14] ^ a[15];
    b[13] = Mul(0x02,a[13]) ^ Mul(0x03,a[14]) ^ a[15] ^ a[12];
    b[14] = Mul(0x02,a[14]) ^ Mul(0x03,a[15]) ^ a[12] ^ a[13];
    b[15] = Mul(0x02,a[15]) ^ Mul(0x03,a[12]) ^ a[13] ^ a[14];
    for( int i = 0 ; i < 16 ; i ++ )
        a[i] = b[i] ;
}

//列混合的逆
void InvMixColumns(unsigned char *a)
{
    unsigned char b[16];
    b[0] = Mul(0x0E, a[0]) ^ Mul(0x0B, a[1]) ^ Mul(0x0D, a[2]) ^ Mul(0x09,
a[3]);
    b[1] = Mul(0x0E, a[1]) ^ Mul(0x0B, a[2]) ^ Mul(0x0D, a[3]) ^ Mul(0x09,
a[0]);
    b[2] = Mul(0x0E, a[2]) ^ Mul(0x0B, a[3]) ^ Mul(0x0D, a[0]) ^ Mul(0x09,
a[1]);
    b[3] = Mul(0x0E, a[3]) ^ Mul(0x0B, a[0]) ^ Mul(0x0D, a[1]) ^ Mul(0x09,
a[2]);
    b[4] = Mul(0x0E, a[4]) ^ Mul(0x0B, a[5]) ^ Mul(0x0D, a[6]) ^ Mul(0x09,
a[7]);
    b[5] = Mul(0x0E, a[5]) ^ Mul(0x0B, a[6]) ^ Mul(0x0D, a[7]) ^ Mul(0x09,
a[4]);
    b[6] = Mul(0x0E, a[6]) ^ Mul(0x0B, a[7]) ^ Mul(0x0D, a[4]) ^ Mul(0x09,
a[5]);
    b[7] = Mul(0x0E, a[7]) ^ Mul(0x0B, a[4]) ^ Mul(0x0D, a[5]) ^ Mul(0x09,
a[6]);
    b[8] = Mul(0x0E, a[8]) ^ Mul(0x0B, a[9]) ^ Mul(0x0D, a[10]) ^ Mul(0x09,
a[11]);
    b[9] = Mul(0x0E, a[9]) ^ Mul(0x0B, a[10]) ^ Mul(0x0D, a[11]) ^ Mul(0x09,
a[8]);
    b[10] = Mul(0x0E, a[10]) ^ Mul(0x0B, a[11]) ^ Mul(0x0D, a[8]) ^ Mul(0x09,
a[9]);
    b[11] = Mul(0x0E, a[11]) ^ Mul(0x0B, a[8]) ^ Mul(0x0D, a[9]) ^ Mul(0x09,
a[10]);
    b[12] = Mul(0x0E, a[12]) ^ Mul(0x0B, a[13]) ^ Mul(0x0D, a[14]) ^ Mul(0x09,
a[15]);
    b[13] = Mul(0x0E, a[13]) ^ Mul(0x0B, a[14]) ^ Mul(0x0D, a[15]) ^ Mul(0x09,
a[12]);
    b[14] = Mul(0x0E, a[14]) ^ Mul(0x0B, a[15]) ^ Mul(0x0D, a[12]) ^ Mul(0x09,
a[13]);
    b[15] = Mul(0x0E, a[15]) ^ Mul(0x0B, a[12]) ^ Mul(0x0D, a[13]) ^ Mul(0x09,
a[14]);
    for (int i = 0; i < 16; i++)
        a[i] = b[i];
}

```

```

//AES算法加密函数
void AES(unsigned char plaintext[16], unsigned char ciphertext[16], unsigned
char k[11][16],int Round)
{
    int i,round;
    for(i=0;i<16;i++){
        ciphertext[i]=plaintext[i];} //此处注意代码思想
    AddRoundKey(ciphertext,k[0]);
    for(round=1;round<Round;round++){
        SubBytes(ciphertext);
        ShiftRows(ciphertext);
        MixColumns(ciphertext);
        AddRoundKey(ciphertext,k[round]);
    }
    SubBytes(ciphertext);
    ShiftRows(ciphertext);
    AddRoundKey(ciphertext,k[Round]);
}

//AES解密常规模式
void RAES(unsigned char plaintext[16], unsigned char ciphertext[16], unsigned
char k[11][16], int Round)
{
    int i,round;
    for(i=0;i<16;i++){
        plaintext[i]=ciphertext[i];}
    AddRoundKey(plaintext,k[Round]);
    InvShiftRows(plaintext);
    InvSubBytes(plaintext);
    for(round=9;round>0;round--){
        AddRoundKey(plaintext,k[round]);
        InvMixColumns(plaintext);
        InvShiftRows(plaintext);
        InvSubBytes(plaintext);
    }
    AddRoundKey(plaintext,k[0]);
}

int main()
{
    int i;
    unsigned char k[11][16]; //扩展密钥
    unsigned char ciphertext[16]; //密文
    unsigned char plaintext[16]={0x15, 0xAE, 0x9F, 0xEE, 0x9E, 0xAC, 0xEF, 0x05,
0x28, 0xC2, 0x2C, 0xD1, 0xA0, 0x03, 0xEE, 0xCD
    };
    unsigned char Key[16]={
        0x2B, 0x7E, 0x15, 0x16, 0x19, 0xAE , 0xD2 , 0xA6, 0xAB, 0xF7, 0x15,
0x88, 0x26, 0xCF, 0x4F, 0x3C
    };
    printf("plainText:\n");
    for(i=0;i<16;i++){
        printf("%02x ",plaintext[i]);} //明文
    printf("\n");

    KeyExpansion(Key,k);
    // //加密
    // AES(plaintext,ciphertext,k,10);
    // printf("cipherText:\n");

```

```

// for(i=0;i<16;i++){
//     printf("%02x ",ciphertext[i]);
// }
// printf("\n");
//解密
// RAES(plaintext,ciphertext,k,10);
RAES(ciphertext,plaintext,k,10);
printf("decipherText:\n");
for(i=0;i<16;i++){
    printf("%02x ",ciphertext[i]);
}
printf("\n");
for(i=0;i<16;i++){
    printf("%c",ciphertext[i]);
}
printf("\n");
return 0;
}
// plainText:
// 15 ae 9f ee 9e ac ef 05 28 c2 2c d1 a0 03 ee cd
// decipherText:
// 6d 4d 50 66 42 79 6a 41 51 69 71 68 43 6c 44 54
// mMPfByjAQiqhClDT

```

ncar

360加固，利用frida-dexdump脱壳，找到登录逻辑

```

    });
    ((Button)root.findViewById(0x7f080057)).setOnClickListener(new View.OnClickListener() {
        @Override // android.view.View.OnClickListener
        public void onClick(View view) {
            EditText ipt_username = (EditText)root.findViewById(0x7f080090);
            EditText ipt_password = (EditText)root.findViewById(0x7f08008f);
            if(("Admin".equals(ipt_username.getText().toString()) && ("u203p2v2f3y2937383n2q2p223v2n2r263p2r2z2n2w2p2a3t2n2u29323h3".equals(NotificationsFragment.trans(ipt_password.getText().
            NotificationsFragment.this.isLogin = true;
            System.out.println("Login ok");
            Toast.makeText(NotificationsFragment.this.getContext(), "Login ok", 1).show();
            textView.setText("Login OK<=");
            return;
        }

        NotificationsFragment.this.isLogin = false;
        Toast.makeText(NotificationsFragment.this.getContext(), "fail", 1).show();
        System.out.println("Login failllllllllllllllllllll");
        textView.setText("Login failllllllllllllllllllll");
    }
    });
    return root;
}

@Override // androidx.fragment.app.Fragment
public void onDestroyView() {
    super.onDestroyView();
    this.binding = null;
}

public static String trans(String _ipt) {
    byte[] ipt = _ipt.getBytes();
    StringBuffer buff = new StringBuffer();
    int i;
    for(i = 0; i < ipt.length / 3; ++i) {
        int sum = 0;
        int j;
        for(j = 0; j < 3; ++j) {
            sum = (sum + (ipt[i * 3 + j] - 0x30)) * 10;
        }
        int v3_1 = sum / 10;
        buff.insert(i * 2, String.valueOf(((char)NotificationsFragment.seed.charAt(v3_1 % 36))));
        buff.insert(i * 2 + 1, String.valueOf(((char)NotificationsFragment.seed.charAt(v3_1 / 36))));
    }
    return buff.toString();
}
}

```

```

m = 'u203p2v2f3y2937383n2q2p223v2n2r263p2r2z2n2w2p2a3t2n2u29323h3'
table = "0123456789abcdefghijklmnopqrstuvwxyz"
for i in range(0,len(m),2):
    tmp = table.index(m[i])+table.index(m[i+1])*36
    # print(str(tmp).rjust(3,'0'),end='')
    print(chr(tmp),end='')
# flag{just_bang_crack_have_fun}

```

