

Task1

1.1 Part 1

1.在题目中有一个函数是**加密**相关的函数，请找出这个函数的**地址**

- 首先使用 `readelf` 命令发现程序的入口点是0x1443(start函数的地址)，因此从start开始分析
- 观察到start中有四个类似加密函数的函数，便依次查看，可得加密函数为 `sub_12CD`
- Hex地址: 0x12CD

2.当你找到了这个加密函数，请找出程序在加密过程中所使用到的密钥

- 首先可得该函数传入的参数为 `v3,v2,v1`
- 进入 `sub_10F0`，`a1,a2` 对应的是参数 `v3,v2`，可得该函数把 `a1` 作为密钥，`a2` 为密钥长度进行加密过程一，并将结果存入 `a3` 中。可得 `v3` 为密钥

```
o for ( i = 0; i <= 255; ++i )
{
    result = &a3[i];
    *result = i;
} //将a3数组的元素初始化为0到255的连续整数（初始化S盒）（a3即为密钥）
v4 = 0;
v5 = 0;
for ( j = 0; j <= 255; ++j ) //随机搅乱S盒，a2为密钥长度
{
    v5 += a3[j] + a1[v4];
    v6 = a3[j];
    a3[j] = a3[v5];
    a3[v5] = v6;
    result = (_BYTE *) (unsigned int) ((v4 + 1) % a2);
    v4 = (v4 + 1) % a2;
}
return result;
}
```

- 进入 `sub_11DF`

```

for ( i = 0; ; ++i )//将S-box和明文进行xor运算,得到密文
{
    result = (unsigned int)i;
    if ( i >= a3 )
        break;
    v5 += a1[++v4];
    v6 = a1[v4];
    a1[v4] = a1[v5];
    a1[v5] = v6;
    a2[i] ^= a1[(unsigned __int8)(a1[v4] + a1[v5])];
}
return result;
}

```

由上述分析可得，密钥为 `v3` 即aaa@niwu，长度为8

3.在这个题目中，程序简单封装了**短字符串**类型，请在IDA 中恢复它的结构体

- 可将十六进制表示恢复为短字符串类型

```

int v0; // [rsp+0h] [rbp-D0h] BYREF
int v1; // [rsp+4h] [rbp-CCh] BYREF
unsigned int v2; // [rsp+20h] [rbp-B0h]
__int64 v3[3]; // [rsp+24h] [rbp-ACh] BYREF
int v4; // [rsp+40h] [rbp-90h] BYREF
__int64 v5; // [rsp+44h] [rbp-8Ch]
__int64 v6; // [rsp+4Ch] [rbp-84h]
__int64 v7; // [rsp+54h] [rbp-7Ch]
int v8; // [rsp+60h] [rbp-70h] BYREF
__int64 v9; // [rsp+64h] [rbp-6Ch]
__int64 v10; // [rsp+6Ch] [rbp-64h]
__int64 v11; // [rsp+74h] [rbp-5Ch]
int v12; // [rsp+80h] [rbp-50h] BYREF
__int64 v13; // [rsp+84h] [rbp-4Ch]
__int64 v14; // [rsp+8Ch] [rbp-44h]
__int64 v15; // [rsp+94h] [rbp-3Ch]
int v16; // [rsp+A0h] [rbp-30h] BYREF
__int64 v17; // [rsp+A4h] [rbp-2Ch]
__int64 v18; // [rsp+ACH] [rbp-24h]
__int64 v19; // [rsp+B4h] [rbp-1Ch]
unsigned __int64 v20; // [rsp+C8h] [rbp-8h]

v20 = __readfsqword(0x28u);
v2 = 8;
v3[0] = 'aaa@niwu';
v3[1] = 0LL;
v3[2] = 0LL;
v4 = 11;
v5 = 'alf ruoY';
v6 = 2112103LL;
v7 = 0LL;
v8 = 8;
v9 = '!tcerroC';
v10 = 0LL;
v11 = 0LL;
v12 = 6;
v13 = '!gnorW';
v14 = 0LL;
v15 = 0LL;

```

C语言描述：

```
typedef struct{
    int v0;
    int v1;
    unsigned int v2;
    _int64 v3[3];
    int v4;
    _int64 v5;
    _int64 v6;
    _int64 v7;
    int v8;
    _int64 v9;
    _int64 v10;
    _int64 v11;
    int v12;
    _int64 v13;
    _int64 v14;
    _int64 v15;
    int v16;
    _int64 v17;
    _int64 v18;
    _int64 v19;
    unsigned _int64 v20;
}struct;
```

4. 解答的flag内容

- 即通过rc4的解密过程可得flag
- 但是要注意的点!
 - 密钥是反过来的! (instead of aaa@niwu ,it is uwin@aaa!!!!)
 - 加密后的密文也要按行倒序! (呜呜呜因为没注意到这两个卡了好久)
- 在得知这两点注意事项后, 就很容易得到解密后的原文, 即flag: AAA{y0u_c4tch_Me!}

1.2 Part2

- 首先执行命令 `readelf -h chall` 得到程序入口点, 为 `start` 函数, 进入该函数进行分析, 发现 `_libc_start_main` 函数, 故进入 `main` 函数分析
- 依次进入 `sub_149F` `sub_1432` `sub_13A6` 等函数进行查看
-

```

int64 __fastcall sub_12B8(int64 a1, int64 a2, int a3)
{
    int64 result; // rax
    unsigned __int8 v4; // [rsp+1Ch] [rbp-8h]
    unsigned __int8 v5; // [rsp+1Dh] [rbp-7h]
    char v6; // [rsp+1Eh] [rbp-6h]
    unsigned int i; // [rsp+20h] [rbp-4h]

    v4 = 0;
    v5 = 0;
    for ( i = 0; ; ++i )
    {
        result = i;
        if ( (int)i >= a3 )
            break;
        v5 += *(_BYTE *) (i+v4 + a1);
        v6 = *(_BYTE *) (v4 + a1);
        *(_BYTE *) (v4 + a1) = *(_BYTE *) (v5 + a1);
        *(_BYTE *) (a1 + v5) = v6;
        *(_BYTE *) ((int)i + a2) ^= *(_BYTE *) ((unsigned __int8) (*(_BYTE *) (v4 + a1) + *(_BYTE *) (v5 + a1)) + a1);
    }
    return result;
}

```

由sub13A6函数的12B8可知该函数实现的是RC4加密算法（与上题类似），所以key作为13A6传入的参数， dword_4010作为密钥的长度，可知密钥即为aUwinAaa,查找该地址的具体内容得到密钥，uwin@aaa

```

sub_13A6(aUwinAaa, (unsigned int)dword_4010, &v5, 24LL);
if ( (unsigned int)sub_14D2(&v4, &dword_4030) )
    sub_149F(&v14, &dword_4030);

```

再由下面的14D2函数中的strncmp操作可知该操作作用key对dword_4030进行处理（检验输入的flag正确与否），可知4030即是处理之前的密文。

- 再进行观察，可得到151c对密文和key进行了异或的操作
-

```

unsigned __int64 sub_151C()
{
    int i; // [rsp+0h] [rbp-30h]
    int j; // [rsp+4h] [rbp-2Ch]
    __int64 v3; // [rsp+8h] [rbp-28h]
    __int64 v4[3]; // [rsp+10h] [rbp-20h]
    unsigned __int64 v5; // [rsp+28h] [rbp-8h]

    v5 = __readfsqword(0x28u);
    v3 = 0xDFDB9F8A81D7DAABLL;
    v4[0] = 0x672B12E36C9410ECLL;
    v4[1] = 0xE6EE307E2B906862LL;
    v4[2] = 0xCA787CF352A4EC49LL;
    for ( i = 0; i < dword_4010; ++i )
        aUwinAaa[i] ^= *(_BYTE *) &v4[-1] + i);
    for ( j = 0; j < dword_4030; ++j )
        byte_4034[j] ^= *(_BYTE *) &v4 + j);
    return v5 - __readfsqword(0x28u);
}

```

所以用该字符串将key和密文分别进行异或操作，再通过rc4解密，即可得到flag

其中byte_4034的内容为

```

byte_4034 db 0FBh, 61h, 7, 84h, 0E2h, 52h, 0B0h, 2 dup(0F7h), 70h
; DATA XREF: sub_151C+A6+o
; sub_151C+C4+o
db 5Eh, 69h, 0BEh, 0B5h, 0E8h, 0A5h, 58h, 0CBh, 0FBh, 0F2h
db 8Dh, 2Eh, 85h, 0Eh

```

然后v4的内容就在上面给出，但是要注意的是要反过来！

对密钥的异或操作同理，其中v4[-1]就是v3，同样注意反过来

```

a = 0xfb610784e252b0f7f7705e69beb5e8a558cbfbf28d2e850e
b = 0xec10946ce3122b676268902b7e30eee649eca452f37c78ca
c = a^b
print("{:x}".format(c))
d = 0x7577696e40616161
e = 0xabdad7818a9fdbdf
f = d^e
print("{:x}".format(f))

```

```

177193e801409b909518ce42c085064311275fa07e52fdc4
deadbeefcafebabe

```

可得用于rc4加密的密钥和密文，这时只需解密即可得到flag！

The screenshot shows an online RC4 decryption tool. The interface includes a green header with the text "RC4" and a close/pause icon. Below the header, there are three input fields: "Passphrase" containing "deadbeefcaf...", "Input format" set to "Hex", and "Output format" set to "UTF8". To the right of these fields, the ciphertext "177193e801409b909518ce42c085064311275fa07e52fdc4" is entered. Below the input fields, there is a section labeled "Output" which displays the decrypted result: "AAA{amAz1ng_y0u_F1nd_M3}".

flag即为 AAA{amAz1ng_y0u_F1nd_M3}

Task 2

首先是readelf -h 查看程序入口点，得知从start函数开始分析，进入start函数，点进1268函数，得到该程序进行加密的主要函数。

```

__isoc99_scanf("%30s", v9);
v9[30] = 0;
if ( !(unsigned int)sub_157B(v9) )
{
    puts("Wrong Length.");
    exit(0);
}
if ( !(unsigned int)sub_15AB(v9) )
{
    puts("Wrong Format.");
    exit(0);
}
for ( i = 0; i <= 4; ++i )
    v7[i] = rand() % 128;
for ( j = 0; j <= 29; ++j )
    v9[j + 32] = v9[v8[j]];
v9[62] = 0;
for ( k = 0; k <= 29; ++k )
    v9[k + 32] ^= LOBYTE(v7[k % 5]);
printf("Do u know pseudo? Give u the message: ");
for ( m = 0; m <= 29; ++m )
    printf("%02X", (unsigned int)(char)v9[m + 32]);
putchar(10);

```

分析该程序可知，首先随机产生长度为5的v7数组，其次用v8打乱v9数组，然后使用v7对v9进行异或运算的加密，最后输出结果。由该过程可知v7为密钥，v9为加密的文章。现在的关键就在于找到密钥

注意到附件output文本，又flag的内容固定含有AAA{}，因此考虑通过推算其在output中的内容反推出密钥的内容

首先由v8的序列得到v8[10]=0,v8[6]=1,v8[17]=2,v8[28]=3,v8[19]=29，分别对应flag中的AAA{}，将序号加32可得它们在output中的位置，又注意到10, 6, 17, 28, 19模5分别为0, 1, 2, 3, 4所以解出来的key

v8[10]=0	A		v9[42]	41	74	35
v8[6]=1	A		v9[38]	41	44	05
v8[17]=2	A	⇒	v9[44]	41	66	2A
v8[28]=3	{		v9[60]	7b	02	79
v8[19]=29	}		v9[51]	7d	2d	50

其中最右边三列，第一列为AAA{}的16进制表示，第三列为output中对应位置的hex值，中间一列为进行异或运算后的密钥hex值

即为密钥key = 0x744446b022d。得到密钥后依次与output中异或运算并且按v8的序号重新排列之后即可得到原文即flag：AAA{UpxEz_4nd_Ps3udo_1s_Fvn!!}