[XCTF-Reverse] 13-18

原创

石氏是时试 于 2022-03-16 17:10:30 发布 154 收藏
分类专栏： CTF reverse 文章标签： reverse

13，tu-ctf-2016_re-for-50-plz-50

mips的题，但是机子上没安相应软件ida没法反编译成c，不过毕竟是2分题应该不很难，直接看汇编。

这里它弄了个串，然后把输入逐字节与0x37异或然后比较。

```
.text:004013C8 loc_4013C8:                             # CODE XREF: main+A4↓j
.text:004013C8                 lui     $v0, 0x4A
.text:004013CC                 addiu   $v1, $v0, (meow - 0x4A0000)  # "cbtcqLUBChERV[[Nh@_X^D]X_YPV[CJ"
.text:004013D0                 lw      $v0, 0x28+var_10($fp)
.text:004013D4                 addu    $v0, $v1, $v0
.text:004013D8                 lb      $v1, 0($v0)
.text:004013DC                 lw      $v0, 0x28+arg_4($fp)
.text:004013E0                 addiu   $v0, 4
.text:004013E4                 lw      $a0, 0($v0)
.text:004013E8                 lw      $v0, 0x28+var_10($fp)
.text:004013EC                 addu    $v0, $a0, $v0
.text:004013F0                 lb      $v0, 0($v0)
.text:004013F4                 xori    $v0, 0x37            <-------- 值异或0x37
.text:004013F8                 sll     $v0, 24
.text:004013FC                 sra     $v0, 24
.text:00401400                 beq     $v1, $v0, loc_401428
.text:00401404                 move    $at, $at
.text:00401408                 lui     $v0, 0x47
.text:0040140C                 addiu   $a0, $v0, (aNooooooooooooo - 0x470000)  # "NO000000000000000000\n"
.text:00401410                 jal     print
.text:00401414                 move    $at, $at
.text:00401418                 lw      $gp, 0x28+var_18($fp)
.text:0040141C                 jal     exit_funct
.text:00401420                 move    $at, $at
.text:00401424                 lw      $gp, 0x28+var_18($fp)
```

于是进行下解码：

```
a = b"cbtcqLUBChERV[[Nh@_X^D]X_YPV[CJ"
print("".join([chr(i^0x37) for i in a]))
#TUCTF{but_really_whoisjohngalt}
```

14，suctf-2016_dmd-50

题目对输入作md5然后与指定串比较

```
  std::operator<<<std::char_traits<char>>(&std::cout, "Enter the valid key!\n", envp);
  std::operator>><char,std::char_traits<char>>(&edata, &v42);
  std::allocator<char>::allocator(&v38);
  std::string::string(&v39, &v42, &v38);
  md5(&v40, &v39);                          //<----对输入值进行md5
  v41 = (_BYTE *)std::string::c_str((std::string *)&v40);
  std::string::~string((std::string *)&v40);
  std::string::~string((std::string *)&v39);
  std::allocator<char>::~allocator(&v38);
  if ( *v41 != 55
  ......
    || v41[31] != 48 )
```

找解密网站解得

```
md5(md5(xxx))  = grape
```

由于题目只做了一次md5,所以要md5(grape)

```
b781cbb29054db12f88f08c6e161c199
```

15，school-ctf-winter-2015_parallel-comparator-200

这个题直接给的c原码

```
#include <stdlib.h>
#include <stdio.h>
#include <pthread.h>

#define FLAG_LEN 20

void * checking(void *arg) {
    char *result = malloc(sizeof(char));
    char *argument = (char *)arg;
    *result = (argument[0]+argument[1]) ^ argument[2];
    return result;
}

int highly_optimized_parallel_comparsion(char *user_string)
{
    int initialization_number;
    int i;
    char generated_string[FLAG_LEN + 1];
    generated_string[FLAG_LEN] = '\0';

    while ((initialization_number = random()) >= 64);

    int first_letter;
    first_letter = (initialization_number % 26) + 97;

    pthread_t thread[FLAG_LEN];
    char differences[FLAG_LEN] = {0, 9, -9, -1, 13, -13, -4, -11, -9, -1, -7, 6, -13, 13, 3, 9, -13, -11, 6
    char *arguments[20];
```

```
    char *arguments[20];
    for (i = 0; i < FLAG_LEN; i++) {
        arguments[i] = (char *)malloc(3*sizeof(char));
        arguments[i][0] = first_letter;
        arguments[i][1] = differences[i];
        arguments[i][2] = user_string[i];

        pthread_create((pthread_t*)(thread+i), NULL, checking, arguments[i]);
    }

    void *result;
    int just_a_string[FLAG_LEN] = {115, 116, 114, 97, 110, 103, 101, 95, 115, 116, 114, 105, 110, 103, 95,
    for (i = 0; i < FLAG_LEN; i++) {
        pthread_join(*(thread+i), &result);
        generated_string[i] = *(char *)result + just_a_string[i];  #生成gen再与just比较，即result=0
        free(result);
        free(arguments[i]);
    }

    int is_ok = 1;
    for (i = 0; i < FLAG_LEN; i++) {
        if (generated_string[i] != just_a_string[i])
            return 0;
    }

    return 1;
}

int main()
{
    char *user_string = (char *)calloc(FLAG_LEN+1, sizeof(char));
    fgets(user_string, FLAG_LEN+1, stdin);
    int is_ok = highly_optimized_parallel_comparsion(user_string);
    if (is_ok)
        printf("You win!\n");
    else
        printf("Wrong!\n");
    return 0;
}
```

主要逻辑为

```
generated_string=result+just_a_string  == just_a_string
```

这个显然是要求result=0。再看生成result的check

```
result = (first_letter + differences)^ user_string
```

由于first_letter不确定，所以先作出26个解再找合适的解

```
a = [0, 9, -9, -1, 13, -13, -4, -11, -9, -1, -7, 6, -13, 13, 3, 9, -13, -11, 6, -7]
#b = [115, 116, 114, 97, 110, 103, 101, 95, 115, 116, 114, 105, 110, 103, 95, 105, 116, 95, 105, 115]
#generated_string[i] = *(char *)result + just_a_string[i];  #生成gen再与just比较，即result=0
# (?+a)^u ==0 ==> u==?+a
for i in range(26):
    print(bytes([i+97+a[j] for j in range(20)]))

#b'lucky_hacker_you_are'
```

16，school-ctf-winter-2015-secret-galaxy-300

初看没有问题，然后逐个看函数发现__libc_csu_gala有数据（linux下的程序编译进的函数比较少，没几个函数很容易找）

```
__int64 _libc_csu_gala()
{
  __int64 result; // rax

  sc = off_601288;
  *((_QWORD *)&sc + 2) = &byte_601384;
  *((_DWORD *)&sc + 2) = 31337;
  *((_DWORD *)&sc + 3) = 1;
  byte_601384 = off_601268[8];   //Andromeda   a
  byte_601385 = off_601280[7];   //Triangulum  l
  byte_601386 = off_601270[4];   //Messier     i
  byte_601387 = off_601268[6];   //            e
  byte_601388 = off_601268[1];   //            n
  byte_601389 = off_601270[2];   //            s
  byte_60138A = 95;              //            _
  byte_60138B = off_601268[8];   //            a
  byte_60138C = off_601268[3];   //            r
  byte_60138D = off_601278[5];   //Sombrero    e
  byte_60138E = 95;              //            _
  byte_60138F = off_601268[8];   //            a
  byte_601390 = off_601268[3];   //            r
  byte_601391 = off_601268[4];   //            o
  byte_601392 = off_601280[6];   //            u
  byte_601393 = off_601280[4];   //            n
  byte_601394 = off_601268[2];   //            d
  byte_601395 = 95;              //            _
  byte_601396 = off_601280[6];   //            u
  result = (unsigned __int8)off_601270[3];
  byte_601397 = off_601270[3];   //            s
  byte_601398 = 0;
  return result;
}
```

逐个点击找到对应的串的对应字母

```
//aliens_are_around_us
```

17，2019_西湖论剑_预选赛_testre

程序很乱没看大明白，发现对输入串进行编码后比较，这个编码用的是标准的base58

```
  while ( j < n )
  {
    v4 = v11;
    *((_BYTE *)s + v20) = byte_400EB0[v11[j]];// 123456789ABCDEFGHJKLMNPQRSTUVWXYZabcdefghijkmnopqrstuvwx
    *((_BYTE *)v26 + v20++) = byte_400EF0[v4[j++]];// ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxy
  }
  *((_BYTE *)s + v20) = 0;
  *v30 = v20 + 1;
  if ( !strncmp((const char *)s, "D9", 2uLL)  // s=D9cS9N9iHjMLTdA8YSMRMp
    && !strncmp((const char *)s + 20, "Mp", 2uLL)
    && !strncmp((const char *)s + 18, "MR", 2uLL)
    && !strncmp((const char *)s + 2, "cS9N", 4uLL)
    && !strncmp((const char *)s + 6, "9iHjM", 5uLL)
    && !strncmp((const char *)s + 11, "LTdA8YS", 7uLL) )
  {
    HIDWORD(v6) = puts("correct!");
  }
```

对对比串进行base58解码

```
//base58_is_boring
```

18，school-ctf-winter-2015_simple-check-100

这个逻辑有点复杂了，先与deadbeef异或后再与flag_data异或每4个一组倒着输出

```
  for ( i = 0; i <= 6; ++i )
  {
    v3 = *(_DWORD *)(4LL * i + v6) ^ 0xDEADBEEF;
    v1 = (int *)&v3;
    v7 = (int *)&v3;
    for ( j = 3; j >= 0; --j )
      LODWORD(v1) = putchar((char)(*((_BYTE *)v7 + j) ^ flag_data[4 * i + j]));
  }
```

解码程序

```
a = [84,-56,126,-29,100,-57,22,-102,-51,17,101,50,45,-29,-45,67,-110,-87,-99,-46,-26,109,44,-45,-74,-67,-2,
b = 0xDEADBEEF
b = [0xef,0xbe,0xad,0xde]
a = [(i+256)%256 for i in a]

flag_data =[0xDC,0x17,0xBF,0x5B,0xD4,0x0A,0xD2,0x1B,0x7D,0xDA,0xA7,0x95,0xB5,0x32,0x10,0xF6,0x1C,0x65,0x53,
for i in range(7):
    for j in range(3,-1,-1):
        c = a[i*4+j]^b[j]^flag_data[i*4+j]
        print(chr(c), end='')

#flag_is_you_know_cracking!!!
```

做饭了，明天见。