上周交作业逆向区

原创

Wolffy007 于 2021-12-03 09:29:03 发布 141 收藏
文章标签： c语言

本文链接：https://blog.csdn.net/Wolffy007/article/details/121691475

逆向区作业

1.Hello, CTF

引用`[https://www.jianshu.com/p/8272ed14cfde]`里的python解16进制编码的方法
437261636b4d654a757374466f7246756e
CrackMeJustForFun

> **FLAG:**
>
> flag
> CrackMeJustForFun
>
> **看C语言伪代码**:
>
> **分析：**

```
 ...
strcpy(&v13, "437261636b4d654a757374466f7246756e");// v13存了东西
  while ( 1 )
  {
    memset(&v10, 0, 0x20u);
    v11 = 0;
    v12 = 0;
    sub_40134B((int)aPleaseInputYou, v6);       // sub_40134B输出字符
    scanf(aS, v9);                              // v9接收
    if ( strlen(v9) > 0x11 )                    // 长度大于17 break
      break;
    v3 = 0;
    do
    {
      v4 = v9[v3];
      if ( !v4 )                                // 这个取反就可以理解为
                                                // 当字符数组某位存在时 返回int 1
                                                // 当某位不存在时 返回int 0
                                                // 这里是取反的结果
        break;
      sprintf(&v8, asc_408044, v4);             // sprintf 格式化输出转存字符串
                                                // asc_408044变量中存储占位符 %x
                                                // 将V4结果转换为16进制
                                                // 也就是进行16进制编码
                                                // 结果输出到V8变量中
      strcat(&v10, &v8);
      ++v3;
    }
    while ( v3 < 17 );                          // 也就对应了最大输入 17 位
      ...
```

2.insanity

DO:

IDA打开查看伪代码

DO2:

发现有一个明显的提示：`This_is_a_flag`

FLAG:

`9447{This_is_a_flag}`

3.python-trade

DO

- 反编译 pyc---->py

代码：

题目加密过程：

```
Python:
# coding=utf-8

import base64
###加密
def encode(message):
 s = ''
 for i in message:
  x = ord(i) ^ 32     #取数组中的数字 计算ASCII值 与32进行位运算
  x = x + 16          #偏移 +16
  s += chr(x)         #将ASCII转字符 并保存
  print(s)
 return base64.b64encode(s)
###
```

- **解码：**

```
###解密   反过来
a=base64.b64decode('XlNkVmtUI1MgXWBZXCFeKY+AaXNt')
x=''
for i in a:
 x += chr((ord(i)-16)^32)
# 字符转 ASCII -16
#下一步：这里进行的是^位运算     我们通过结果与 32 进行位运算，是可以得到原结果的。(相同取假，不同取真)
#结果  nctf{d3c0mpil1n9_PyC}
print x
###
```

FLAG： *flag:nctf{d3c0mpil1n9_PyC}*

4.re1

1.操作：

> 使用R键转换为字符
> 得到答案：
> .rdata:00413E34 xmmword_413E34 xmmword ‘0tem0c1eW{FTCTUD’
> .rdata:00413E34 ; DATA XREF: _main+10↑r
> .rdata:00413E44 qword_413E44 dq ‘}FTCTUD’ ; DATA XREF: _main+27↑r

- 按照：
  小序:倒着读
  得到答案：

> DUTCTF{Welc0met0DUTCTF}

- 或者直接一点：按下a键转换为字符串

> DUTCTF{Welc0met0DUTCTF}

2.代码分析：

```
正题：
  _mm_storeu_si128((__m128i *)&v5, _mm_loadu_si128((const __m128i *)&xmmword_413E34));
            // _mm_storeu_si128 将后空间内的值放入前面的空间
                                                 // _mm_loadu_si128加载128位值。返回值代表寄存器的变量中的相同值，地址p
不需要16字节对齐。
  v7 = 0;
  v6 = qword_413E44;
  v8 = 0;
  printf("欢迎来到DUTCTF呦\n");
  printf("这是一道很可爱很简单的逆向题呦\n");
  printf("输入flag吧:");
  scanf("%s", &v9);
  v3 = strcmp((const char *)&v5, &v9);
  if ( v3 )
    v3 = -(v3 < 0) | 1;
  if ( v3 )
    printf(aFlag_0);  //错误
  else
    printf((const char *)&unk_413E90); //成功
  system("pause");
  return 0;
}
```

跳转到 `unk_413E90` 变量 直接按a得到字符串。

关于这部分：

```
if ( v3 )
    v3 = -(v3 < 0) | 1;
  if ( v3 )
    printf(aFlag_0);  //错误
  else
    printf((const char *)&unk_413E90); //成功
```

`v3 = -(v3 < 0) | 1` 位运算后 v3 只能是-1 0 1

5.game

题目描述： *菜鸡最近迷上了玩游戏，但它总是赢不了，你可以帮他获胜吗*

游戏规则：

```
"Play a game\n"
"The n is the serial number of the lamp,and m is the state of the lamp\n"
"If m of the Nth lamp is 1,it's on ,if not it's off\n"
"At first all the lights were closed\n"
"Now you can input n to change its state\n"
"But you should pay attention to one thing,if you change the state of the Nth lamp,the state of (N-1)th and (N+1
)th will be changed too\n"
"When all lamps are on,flag will appear\n"
"Now,input n \n"
```

Writeup：

1.xctf-wp

```
    if ( byte_532E28[0] == 1
      && byte_532E28[1] == 1
      && byte_532E28[2] == 1
      && byte_532E28[3] == 1
      && byte_532E28[4] == 1
      && byte_532E28[5] == 1
      && byte_532E28[6] == 1
      && byte_532E28[7] == 1 )
    {
      sub_457AB4();    //跳转
    }
```

> 或者：
> 查找字符串，找到flag，直接定位到这句话所在的函数。

> `sub_457AB4();` 进入：

```
sub_45A7BE((int)"done!!! the flag is ", v1);
  v60 = 18;
  v61 = 64;
  v62 = 98;
  v63 = 5;
  v64 = 2;
  v65 = 4;
  v66 = 6;
  v67 = 3;
  v68 = 6;
  v69 = 48;
  v70 = 49;
  v71 = 65;
  v72 = 32;
  v73 = 12;
  v74 = 48;
  v75 = 65;
  v76 = 31;
  v77 = 78;
  v78 = 62;
  v79 = 32;
  v80 = 49;
  v81 = 32;
  v82 = 1;
  v83 = 57;
  v84 = 96;
  v85 = 3;
  v86 = 21;
  v87 = 9;
  v88 = 4;
  v89 = 62;
  v90 = 3;
  v91 = 5;
  v92 = 4;
  v93 = 1;
  v94 = 2;
  v95 = 3;
  v96 = 44;
  v97 = 65;
  v98 = 78;
  v99 = 32;
  v100 = 16;
  v101 = 97;
  v102 = 54;
  v103 = 16;
  v104 = 44;
  v105 = 52;
  v106 = 32;
  v107 = 64;
  v108 = 89;
  v109 = 45;
  v110 = 32;
  v111 = 65;
  v112 = 15;
  v113 = 34;
  v114 = 18;
  v115 = 16;
  v116 = 0;
  v3 = 123;
  v4 = 32;
  v5 = 18;
  v6 = 98;
  v7 = 119;
  v8 = 108;
  v9 = 65;
  v10 = 41;
  v11 = 124;
  v12 = 80;
  v13 = 125;
  v14 = 38;
  v15 = 124;
  v16 = 111;
  v17 = 74;
  v18 = 49;
  v19 = 83;
  v20 = 108;
  v21 = 94;
  v22 = 108;
  v23 = 84;
  v24 = 6;
  v25 = 96;
  v26 = 83;
  v27 = 44;
  v28 = 121;
  v29 = 104;
  v30 = 110;
  v31 = 32;
  v32 = 95;
  v33 = 117;
  v34 = 101;
  v35 = 99;
  v36 = 123;
  v37 = 127;
  v38 = 119;
  v39 = 96;
  v40 = 48;
  v41 = 107;
  v42 = 71;
  v43 = 92;
  v44 = 29;
  v45 = 81;
  v46 = 107;
  v47 = 90;
  v48 = 85;
  v49 = 64;
  v50 = 12;
  v51 = 43;
  v52 = 76;
  v53 = 86;
  v54 = 13;
  v55 = 114;
  v56 = 1;
  v57 = 117;
  v58 = 126;
  v59 = 0;
  for ( i = 0; i < 56; ++i )
  {
    *(&v3 + i) ^= *(&v60 + i);   //v60-v115
    *(&v3 + i) ^= 0x13u;   //0x13u=0x13,u是无符号数，0x是十六进制，0x13=19
  }
  return sub_45A7BE((int)"%s\n", (unsigned int)&v3);
}
```

模拟重现：

```
a=[123,32,18,98,119,108,65,41,124,80,125,38,124,111,74,49,83,108,94,108,84,6,96,83,44,121,104,110,32,95,117,101,
99,123,127,119,96,48,107,71,92,29,81,107,90,85,64,12,43,76,86,13,114,1,117,126]
b=[18,64,98,5,2,4,6,3,6,48,49,65,32,12,48,65,31,78,62,32,49,32,1,57,96,3,21,9,4,62,3,5,4,1,2,3,44,65,78,32,16,97
,54,16,44,52,32,64,89,45,32,65,15,34,18,16]
for i in range(56):
    a[i]=a[i]^b[i]
    a[i]=a[i]^19
    print(chr(a[i]),end='')
```

> **flag：** zsctf{T9is_tOpic_1s_v5ry_int7resting_b6t_others_are_n0t}

法二：OD

- 查找 flag

[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-Z8gZxcPy-1638494859438)(C:\Users\YuDong\Pictures\笔记\11.JPG)]

- 记录内存地址
- 我是加在了cls上

[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-VoY7TAJH-1638494859439)(C:\Users\YuDong\Pictures\笔记\13.JPG)]

[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-UCmUaxu9-1638494859440)(C:\Users\YuDong\Pictures\笔记\14.JPG)]

6.open-source

> 题目描述：菜鸡学逆向学得头皮发麻，终于它拿到了一段源代码

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
int main(int argc, char *argv[]) {
    if (argc != 4) {
        printf("what?\n");
        exit(1);
    }
    unsigned int first = atoi(argv[1]); //将字符串转换为整数
    if (first != 0xcafe) {
        printf("you are wrong, sorry.\n");
        exit(2);
    }
    unsigned int second = atoi(argv[2]);
    if (second % 5 == 3 || second % 17 != 8) {
        printf("ha, you won't get it!\n");
        exit(3);
    }
    if (strcmp("h4cky0u", argv[3])) {
        printf("so close, dude!\n");
        exit(4);
    }
    printf("Brr wrrr grr\n");
    unsigned int hash = first * 31337 + (second % 17) * 11 + strlen(argv[3]) - 1615810207;
    printf("Get your key: ");
    printf("%x\n", hash);
    return 0;
}
```

解决问题：

先生成一个符合条件的数字，再代入计算hash

再按照要求16进制输出

```
for second in range(100):
    if not(second % 5 == 3 or second % 17 != 8):
        print (second)
//我这里用25为例
hash = 0xcafe * 31337 + (25 % 17) * 11 + 7 - 1615810207
print(hex(hash))
```

7.simple-unpack

> 题目描述：菜鸡拿到了一个被加壳的二进制文件

- **dpx -d “二进制文件” 脱壳**

[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-Co0tUNqT-1638494859441)(C:\Users\YuDong\Pictures\笔记\15.JPG)]

- **伪代码：**

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char s1; // [rsp+0h] [rbp-70h]
  unsigned __int64 v5; // [rsp+68h] [rbp-8h]

  v5 = __readfsqword(0x28u);
  _isoc99_scanf((unsigned __int64)"%96s");
  if ( !strcmp(&s1, flag) )
    puts("Congratulations!", flag);
  else
    puts("Try again!", flag);
  return 0;
}
```

- **跳转到flag变量**

> flag{Upx_1s_n0t_a_d3liv3r_c0mp4ny}

8.logmein

> 题目描述：菜鸡开始接触一些基本的算法逆向了

```
  v9 = 0;
  strcpy(v8, ":\"AL_RT^L*.?+6/46");
  v7 = 28537194573619560LL;
  v6 = 7;
  printf("Welcome to the RC3 secure password guesser.\n", a2, a3);
  printf("To continue, you must enter the correct password.\n");
  printf("Enter your guess: ");
  __isoc99_scanf("%32s", s);
  v3 = strlen(s);
  if ( v3 < strlen(v8) )
    sub_4007C0(v8);
  for ( i = 0; i < strlen(s); ++i )
  {
    if ( i >= strlen(v8) )
      ((void (*)(void))sub_4007C0)();
    if ( s[i] != (char)(*((_BYTE *)&v7 + i % v6) ^ v8[i]) )
      ((void (*)(void))sub_4007C0)();
  }
  sub_4007F0();
}
```

分析：

首先v8拷贝到v3中 输入字符长度不能小于8

加密过程：`(char)(*((_BYTE *)&v7 + i % v6) ^ v8[i])`：

循环执行：

v7 长整形转换为字符数组 取 v7[i%v6] 然后进行位运算

引用：

> https://adworld.xctf.org.cn/task/writeup?type=reverse&id=5078&number=4&grade=0&page=1 官方write up
> https://blog.csdn.net/qq_43656475/article/details/103069606 v7的理解方式
> https://www.52pojie.cn/thread-1055089-1-1.html BYTE操作的理解

模拟解密过程：

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define BYTE unsigned char
int main()
{
    char v8[20]=":\"AL_RT^L*.?+6/46";
    long long int  v7 = 28537194573619560LL;
    printf("v7:%lld\n",v7);
    int v6 = 7;
    char result[20];
    int i=0;
    for (i = 0; i < 18; ++i )
    {
      result[i]= (char)(*((BYTE*)&v7 + i % v6) ^ v8[i]);//理解该加密过程
    }
    result[i]='\0';
    printf("%s\n",result);
    return 0;
}
```

笔记部分：

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define BYTE unsigned char
int main()
{
    char v8[20]=":\"AL_RT^L*.?+6/46";
    printf("长度:%d\n",strlen(v8));//长度17
    long long int  v7 = 28537194573619560LL;//尾部LL是声明长整形 __int 64也可以用来定义长整形
    char v77[]="harambe";
    printf("v7:%lld\n",v7);
    int v6 = 7;
    char result[20];
    char result2[20];
    int i=0;
    for (i = 0; i < 18; ++i )
    {
        printf("%s\n",(BYTE *)&v7);//将64位长整形转换为字符型char 输出：harambe  BYTE等价于char
        //引用：https://blog.csdn.net/qq_43656475/article/details/103069606
      result[i]= (char)(*((char*)&v7 + i % v6) ^ v8[i]);//理解该加密过程
      result2[i]= (char)(v77[i % v6] ^ v8[i]);//理解该加密过程
    }
    result[i]='\0';result2[i]='\0';
    printf("%s\n",result);
    printf("%s\n",result2);
    return 0;
}
```

运行结果：

RC3-2016-XORISGUDa

9. no-strings-attached

> 想使用gdb动态调试 但是程序在linux上如何不被当成bash脚本去运行呢?

先找到了authenticate函数：

```
void authenticate()
{
  wchar_t ws[8192]; // [esp+1Ch] [ebp-800Ch]
  wchar_t *s2; // [esp+801Ch] [ebp-Ch]

  s2 = (wchar_t *)decrypt(&s, &dword_8048A90);
  if ( fgetws(ws, 0x2000, stdin) )
  {
    ws[wcslen(ws) - 1] = 0;
    if ( !wcscmp(ws, s2) )
      wprintf(&unk_8048B44);
    else
      wprintf(&unk_8048BA4);
  }
  free(s2);
}
```

分析:

1.定义了宽字节ws s2内放入（使用decrypt函数进行运算的结果）

2. fgets内容到ws
3. 末尾加0截停
4. 字符比对ws和s2
5. 分析`decrypt`函数
6. 找到字符串s

```
.rodata:08048AA8 s               dd 143Ah                ; DATA XREF: authenticate+11↑o
.rodata:08048AAC                 db  36h ; 6
.rodata:08048AAD                 db  14h
.rodata:08048AAE                 db    0
.rodata:08048AAF                 db    0
.rodata:08048AB0                 db  37h ; 7
.rodata:08048AB1                 db  14h
.rodata:08048AB2                 db    0
.rodata:08048AB3                 db    0
.rodata:08048AB4                 db  3Bh ; ;
.rodata:08048AB5                 db  14h
```

2. fgets内容到ws
3. 末尾加0截停
4. 字符比对ws和s2
5. 分析`decrypt`函数
6. 找到字符串s

```
.rodata:08048AA8 s               dd 143Ah                ; DATA XREF: authenticate+11↑o
.rodata:08048AAC                 db  36h ; 6
.rodata:08048AAD                 db  14h
.rodata:08048AAE                 db    0
.rodata:08048AAF                 db    0
.rodata:08048AB0                 db  37h ; 7
.rodata:08048AB1                 db  14h
.rodata:08048AB2                 db    0
.rodata:08048AB3                 db    0
.rodata:08048AB4                 db  3Bh ; ;
.rodata:08048AB5                 db  14h
```