2022网刃杯 个人向WP ICS/Reverse easyiec,freestyle,ez_algorithm writeup

原创

练习两年半的吹头发练习生 已于 2022-04-26 15:27:30 修改 169 收藏

分类专栏： ctf逆向 文章标签： python c语言 软件工程

于 2022-04-25 13:16:48 首次发布

ICS

easyiec

虚假的签到题与真正的签到题.jpg

打开方式切到记事本 Ctrl+F 直接搜flag梭哈

REVERSE

freestyle

简单逆向算法题 f5主函数 一个fun1一个fun2 先跟进fun1

输入一个值 经过处理后得到4400 atoi()这个函数百度了一下跟int()差不多 数学题

4 * (3 * int(s)/9-9) == 4400 得到s = 3327 记一下到fun2

提示里给到The code value is the smallest divisible 意思是下面的取余逆向取最小的被除数

2 * (int(s) %56) == 98 得到s = 105

```
#4 * (3 * int(s)/9-9 ) == 4400
s = ((4400/4)+9)*3
print(s)
#2 * (int(s) %56) == 98
s = (98/2) + 56
print(s) //应该没人看这个吧...
```

主函数提示flag为输入值的md5 拼起来3327105转一下包上flag{}提交

ez_algorithm

解压是个exe 直接丢ida64里

Shift+F12查字符串

前三条明显是加密解密用，直接到"GJ!YOU WIN!"这条 双击跳到ida-view
Ctrl+X 查看交叉引用 跳转到目标函数直接F5反编译

很明显是将输入与他的一个目标字符串进行比较，相同就输出You Win(这里的BRUF是我自定义的名字 环境可能不一样)
目标字符串点一下就能发现是固定不变的，那就看encry函数对输入做了怎么样的处理了 附注释

```
char *__fastcall encryption(char *input_1)
{
  int v1; // eax
  char v2; // al
  char v3; // al
  __int64 v4; // kr00_8
  char v5; // al
  char v6; // al
  int v7; // eax
  char v8; // al
  char v9; // al
  char v10; // al
  char v11; // al
  char v12; // al
  size_t v13; // rbx
  char v15[1012]; // [rsp+20h] [rbp-60h] BYREF
  int v16; // [rsp+414h] [rbp+394h]
  const char *xyp3; // [rsp+418h] [rbp+398h]
  const char *xyp2; // [rsp+420h] [rbp+3A0h]
  int v19; // [rsp+42Ch] [rbp+3ACh]
  char *v20; // [rsp+430h] [rbp+3B0h]
  char *v21; // [rsp+438h] [rbp+3B8h]

  xyp2 = ::xyp2();
  xyp3 = ::xyp3();
  v21 = v15;
  v20 = input_1;
  v19 = 0;
  v16 = 1;
  while ( 1 )
  {
    v13 = v19;
    if ( v13 >= strlen(input_1) )
      return v15;
    if ( *v20 <= 64 || *v20 > 90 )              // 取非大写字母
    {
      if ( *v20 <= 96 || *v20 > 122 )           // 取非小写字母(其实就是数字和符号)
      {
        if ( *v20 == 95 )                       // 当字符为下划线时
        {
          switch ( v16 + rand() % 7 )           // 替换为随机字符
          {
            case 0:
              *v21 = 58;
              break;
            case 1:
              *v21 = 38;
              break;
            case 2:
              *v21 = 43;
              break;
            case 3:
              *v21 = 42;
```

```
            *v21 = 42;
            break;
          case 4:
            *v21 = 92;
            break;
          case 5:
            *v21 = 63;
            break;
          case 6:
            *v21 = 36;
            break;
          case 7:
            *v21 = 35;
            break;
          default:
            break;
        }
      }
      else if ( *v20 <= 47 || *v20 > 57 )
      {
        *v21 = *v20;                            // 特殊符号不做处理直接保留
      }
      else
      {
        v12 = encryption2(*v20);                // 数字进入encry2这个函数处理
        *v21 = v12;
      }
    }
    else                                        // 小写字母处理的开始
    {
      v7 = v19 % 4;                             // 根据取余结果选择一个算法转换
      if ( v19 % 4 == 1 )
      {
        v9 = encryption2(xyp2[(*v20 - 97) * (v19 % 4)]);
        *v21 = v9;
      }
      else if ( v7 > 1 )
      {
        if ( v7 == 2 )
        {
          v10 = encryption2(xyp2[(*v20 - 97) ^ (v19 % 4)]);// v20减去97 相当于把a-z的字母打成0~26
          *v21 = v10;
        }
        else if ( v7 == 3 )
        {
          v11 = encryption2(xyp2[*v20 - 97 + v19 % 4]);
          *v21 = v11;
        }
      }
      else if ( !v7 )
      {
        v8 = encryption2(xyp2[*v20 - 97 - v19 % 4]);
        *v21 = v8;
      }
    }
  }
  else                                          // 大写字母处理
  {
    v1 = v19 % 4;
    if ( v19 % 4 == 1 )
```

```
      {
        v3 = encryption2(xyp3[*v20 - 65 + v19 % 4]); //逻辑大部分同小写
        *v21 = v3;
      }
      else if ( v1 > 1 )
      {
        if ( v1 == 2 )
        {
          v4 = v19 * (*v20 - 65);
          v5 = encryption2(xyp3[(((HIDWORD(v4) >> 30) + (unsigned __int8)v19 * (*v20 - 65)) & 3) - (HIDWORD(v4)
>> 30)]);
          *v21 = v5;
        }
        else if ( v1 == 3 )
        {
          v6 = encryption2(xyp3[(*v20 - 65) ^ (v19 % 4)]);
          *v21 = v6;
        }
      }
      else if ( !v1 )
      {
        v2 = encryption2(xyp3[*v20 - 65 - v19 % 4]);
        *v21 = v2;
      }
    }
    ++v19;
    ++v20;   //计数器
    ++v21;
  }
}
```

按注释一点一点分析
先看第一条v13=v19 v19初始是个0 直接翻到最下面可以看到v19的自加
且判断v13>=输入字符串的长度时 return v15给主函数 那么他就是个计数器了

然后的判断对着ASCII表画个区间图 就能看出来指向的分别是下划线 符号 数字 小写字母 大写字母
下划线直接rand函数随机给值了 看了一眼asc 这些也都是符号
再往下 //保留注释符号 这一条有一点麻烦 我跟上面的区间 画图并了一下
发现其实就是除了下划线的特殊符号集 处理方法是直接保留不动

然后是数字的处理 进到了enc2函数里 那就先进去分析一下 附注释

这里挺简单的 转ASC看一下就能知道作者思路了
enc2把大写转小写 小写转大写 数字符号不处理 丢尽enc3里
enc3把字符不论大小写 往前或往后移位 T与G单独替换 数字反向
这个enc2+3也可以理解为一个一一替换的表

然后往下看就是小写字母的处理 先将计数器 v19%4 然后根据取余选算法
例如^ 就是 将输入结果减97 而a的asc就是97 相当于把$a_{z转为了0}$26 然后对计数器根据取余的结果选择±^/四个算法得到一个数字先叫他i 再去用xyp2这个字符串找这个数字位也就是xyp2[i] 最后再对这个值enc转换

再往下是大写字母的处理 这个地方有一个跟小写不一样 就是计数器取余得到2的时候是一条很长的式子 我没搞懂 但是这一条用不到 可以跳过

原理有了，开始写逆向脚本 enc2+3只是个转换 直接照搬

```
def enc(v):
    asc = ord(v)
    if(asc>64 and asc<=90):
        return encry(chr(asc+32))
    if(asc>96 and asc<=122):
        return encry(chr(asc-32))
    if(asc>47 and asc<=57):
        return encry(chr(asc))
    return encry(chr(asc))

def encry(v):
    asc = ord(v)
    if (asc > 64 and asc <= 70 or asc > 94 and asc <= 102):
        return chr(asc+20)
    if (asc > 84 and asc <= 90 or asc > 116 and asc <= 122):
        return chr(asc - 20)
    if (asc > 71 and asc <= 77 or asc > 103 and asc <= 109):
        return chr(asc + 6)
    if (asc > 77 and asc <= 83 or asc > 109 and asc <= 115):
        return chr(asc - 6)
    if (asc == 71 or asc == 103):
        return chr(asc+13)
    if (asc == 84 or asc == 116):
        return chr(asc-13)
    if(asc>47 and asc <=57):
        asc = chr(105 - asc)
    return asc


xyp1 = 'BRUF{E6oU9Ci#J9+6nWAhwMR9n:}'
xyp2 = 'ckagevdxizblqnwtmsrpufyhoj'
xyp3 = 'TMQZWKGOIAGLBYHPCRJSUXEVND'
daxie =   'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
xiaoxie = 'abcdefghijklmnopqrstuvwxyz'
shuzi = '0123456789'
fuhao = """!"#$%&'()*+,-./:;<=>?@[\\]^_`{|}~"""
flag = ''

for i in range(len(xyp1)):
    if(xyp1[i] in xiaoxie):
        if (i % 4 == 0):
            flag +=chr(int(xyp3.find(enc(xyp1[i])) + (i % 4)) + 65)
        elif (i % 4 == 1):
            flag +=chr(int(xyp3.find(enc(xyp1[i])) - (i % 4)) + 65)
        elif (i % 4 == 3):
            flag +=chr(int(xyp3.find(enc(xyp1[i])) ^ (i % 4)) + 65)
    elif(xyp1[i] in daxie):
        if(i%4==0):
            flag +=chr(int(xyp2.find(enc(xyp1[i])) + (i % 4)) + 97)
        elif(i%4==1):
            flag +=chr(int(xyp2.find(enc(xyp1[i])) / (i % 4)) + 97)
        elif(i%4==2):
            flag +=chr(int(xyp2.find(enc(xyp1[i])) ^ (i % 4)) + 97)
        elif(i%4==3):
            flag +=chr(int(xyp2.find(enc(xyp1[i])) - (i % 4)) + 97)
    elif(xyp1[i] in shuzi):
        flag += enc(xyp1[i])
    elif(xyp1[i] in fuhao):
```

```
    elif(xyp1[i] in fuhao):
        flag += xyp1[i]
print(flag)
```

按着原逻辑反着写 加减对换 乘换成除 异或照写 数字一段加密 符号暂时不变
得到 flag{w3Lc0mE#t0+3NcrYPti0N:} 交一下 错了 往前看符号的转换逻辑
往符号不变的上面看 _下划线 会被替换为随机符号 flag里的三个符号都符合下划线转换后的格式 那怎么知道他本来就是这个符号还是下划线呢？我反正不知道 三个符号3+3+1七种组合爆一下就行了

```
flag{w3Lc0mE_t0+3NcrYPti0N:}
flag{w3Lc0mE#t0_3NcrYPti0N:}
flag{w3Lc0mE#t0+3NcrYPti0N_}
flag{w3Lc0mE_t0+3NcrYPti0N_}
flag{w3Lc0mE#t0_3NcrYPti0N_}
flag{w3Lc0mE_t0_3NcrYPti0N:}
flag{w3Lc0mE_t0_3NcrYPti0N_} //总有一款适合你
```