QCTF2018 Misc - picture writeup

转载

xuchen16 于 2018-07-17 14:26:44 发布 780 收藏
分类专栏： ctf 文章标签： QCTF misc picture writeup 杂项

ctf 专栏收录该内容
66 篇文章 6 订阅
订阅专栏
转载自：https://www.jianshu.com/p/55c5477567fe

看到提示（我承认我一开始没想到。。。。）解密图片，

之后拿到加密逻辑脚本，发现是DES的实现脚本，写出解密脚本

```
#_*_ coding:utf-8 _*_

import re

import sys



ip=  (58, 50, 42, 34, 26, 18, 10, 2,

      60, 52, 44, 36, 28, 20, 12, 4,

      62, 54, 46, 38, 30, 22, 14, 6,

      64, 56, 48, 40, 32, 24, 16, 8,

      57, 49, 41, 33, 25, 17, 9 , 1,

      59, 51, 43, 35, 27, 19, 11, 3,

      61, 53, 45, 37, 29, 21, 13, 5,

      63, 55, 47, 39, 31, 23, 15, 7)



ip_1=(40, 8, 48, 16, 56, 24, 64, 32,

      39, 7, 47, 15, 55, 23, 63, 31,

      38, 6, 46, 14, 54, 22, 62, 30,

      37, 5, 45, 13, 53, 21, 61, 29,

      36, 4, 44, 12, 52, 20, 60, 28,
```

```
      35, 3, 43, 11, 51, 19, 59, 27,

      34, 2, 42, 10, 50, 18, 58, 26,

      33, 1, 41,  9, 49, 17, 57, 25)


e  =(32, 1,  2,  3,  4,  5,  4,  5,

       6, 7,  8,  9,  8,  9, 10, 11,

      12,13, 12, 13, 14, 15, 16, 17,

      16,17, 18, 19, 20, 21, 20, 21,

      22, 23, 24, 25,24, 25, 26, 27,

      28, 29,28, 29, 30, 31, 32,  1)


p=(16,  7, 20, 21, 29, 12, 28, 17,

     1, 15, 23, 26,  5, 18, 31, 10,

     2,  8, 24, 14, 32, 27,  3,  9,

     19, 13, 30, 6, 22, 11,  4,  25)


s=[ [[14, 4, 13,  1,  2, 15, 11,  8,  3, 10,  6, 12,  5,  9,  0,  7],

     [0, 15,  7,  4, 14,  2, 13,  1, 10,  6, 12, 11,  9,  5,  3,  8],

     [4,  1, 14,  8, 13,  6,  2, 11, 15, 12,  9,  7,  3, 10,  5,  0],

     [15, 12,  8,  2,  4,  9,  1,  7,  5, 11,  3, 14, 10,  0,  6, 13]],


     [[15,  1,  8, 14,  6, 11,  3,  4,  9,  7,  2, 13, 12,  0,  5, 10],

     [3, 13,  4,  7, 15,  2,  8, 14, 12,  0,  1, 10,  6,  9, 11,  5],

     [0, 14,  7, 11, 10,  4, 13,  1,  5,  8, 12,  6,  9,  3,  2, 15],

     [13,  8, 10,  1,  3, 15,  4,  2, 11,  6,  7, 12,  0,  5, 14,  9]],


     [[10,  0,  9, 14,  6,  3, 15,  5,  1, 13, 12,  7, 11,  4,  2,  8],

     [13,  7,  0,  9,  3,  4,  6, 10,  2,  8,  5, 14, 12, 11, 15,  1],

     [13,  6,  4,  9,  8, 15,  3,  0, 11,  1,  2, 12,  5, 10, 14,  7],

     [1, 10, 13,  0,  6,  9,  8,  7,  4, 15, 14,  3, 11,  5,  2, 12]],
```

```
   [[7, 13, 14,  3,  0,  6,  9, 10,  1,  2,  8,  5, 11,  12,  4, 15],
    [13,  8, 11,  5,  6, 15,  0,  3,  4,  7,  2, 12,  1, 10, 14,9],
    [10,  6,  9,  0, 12, 11,  7, 13, 15,  1,  3, 14,  5,  2,  8,  4],
    [3, 15,  0,  6, 10,  1, 13,  8,  9,  4,  5, 11, 12,  7,  2, 14]],


   [[2, 12,  4,  1,  7, 10, 11,  6,  8,  5,  3, 15, 13,  0, 14,  9],
    [14, 11,  2, 12,  4,  7, 13,  1,  5,  0, 15, 10,  3,  9,  8,  6],
    [4,  2,  1, 11, 10, 13,  7,  8, 15,  9, 12,  5,  6,  3,  0, 14],
    [11,  8, 12,  7,  1, 14,  2, 13,  6, 15,  0,  9, 10,  4,  5,  3]],

   [[12,  1, 10, 15,  9,  2,  6,  8,  0, 13,  3,  4, 14,  7,  5, 11],
    [10, 15,  4,  2,  7, 12,  9,  5,  6,  1, 13, 14,  0, 11,  3,  8],
    [9, 14, 15,  5,  2,  8, 12,  3,  7,  0,  4, 10,  1, 13, 11,  6],
    [4,  3,  2, 12,  9,  5, 15, 10, 11, 14,  1,  7,  6,  0,  8, 13]],

   [[4, 11,  2, 14, 15,  0,  8, 13,  3, 12,  9,  7,  5, 10,  6,  1],
    [13,  0, 11,  7,  4,  9,  1, 10, 14,  3,  5, 12,  2, 15,  8,  6],
    [1,  4, 11, 13, 12,  3,  7, 14, 10, 15,  6,  8,  0,  5,  9,  2],
    [6, 11, 13,  8,  1,  4, 10,  7,  9,  5,  0, 15, 14,  2,  3, 12]],

  [[13,  2,  8,  4,  6, 15, 11,  1, 10,  9,  3, 14,  5,  0, 12,  7],
    [1, 15, 13,  8, 10,  3,  7,  4, 12,  5,  6, 11,  0, 14,  9,  2],
    [7, 11,  4,  1,  9, 12, 14,  2,  0,  6, 10, 13, 15,  3,  5,  8],
    [2,  1, 14,  7,  4, 10,  8, 13, 15, 12,  9,  0,  3,  5,  6, 11]]]


pc1=(57, 49, 41, 33, 25, 17,  9,
       1, 58, 50, 42, 34, 26, 18,
      10,  2, 59, 51, 43, 35, 27,
```

```
      19, 11,  3, 60, 52, 44, 36,

      63, 55, 47, 39, 31, 23, 15,

       7, 62, 54, 46, 38, 30, 22,

      14,  6, 61, 53, 45, 37, 29,

      21, 13,  5, 28, 20, 12, 4);



pc2= (14, 17, 11, 24,  1,  5,  3, 28,

      15,  6, 21, 10, 23, 19, 12,  4,

      26,  8, 16,  7, 27, 20, 13,  2,

      41, 52, 31, 37, 47, 55, 30, 40,

      51, 45, 33, 48, 44, 49, 39, 56,

      34, 53, 46, 42, 50, 36, 29, 32)



d = (  1,  1,  2,  2,  2,  2,  2,  2, 1, 2, 2, 2, 2, 2, 2, 1)



__all__=['desencode']

class DES():



    def __init__(self):

        pass



    def code(self,from_code,key,code_len,key_len):

        output=""

        trun_len=0







        code_string=self._functionCharToA(from_code,code_len)

        code_key=self._functionCharToA(key,key_len)
```

```
if code_len%16!=0:
    real_len=(code_len/16)*16+16
else:
    real_len=code_len

if key_len%16!=0:
    key_len=(key_len/16)*16+16
key_len*=4



trun_len=4*real_len

for i in range(0,trun_len,64):
    run_code=code_string[i:i+64]
    l=i%key_len
    run_key=code_key[l:l+64]



    run_code= self._codefirstchange(run_code)
    run_key= self._keyfirstchange(run_key)



    for j in range(16):



        code_r=run_code[32:64]
        code_l=run_code[0:32]
```

```
            run_code=code_r



            code_r= self._functionE(code_r)



            key_l=run_key[0:28]
            key_r=run_key[28:56]
            key_l=key_l[d[j]:28]+key_l[0:d[j]]
            key_r=key_r[d[j]:28]+key_r[0:d[j]]
            run_key=key_l+key_r
            key_y= self._functionKeySecondChange(run_key)



            code_r= self._codeyihuo(code_r,key_y)



            code_r= self._functionS(code_r)



            code_r= self._functionP(code_r)



            code_r= self._codeyihuo(code_l,code_r)
            run_code+=code_r

        code_r=run_code[32:64]
        code_l=run_code[0:32]
        run_code=code_r+code_l
```

```
        output+=self._functionCodeChange(run_code)
    return output



def _codeyihuo(self,code,key):
    code_len=len(key)
    return_list=''
    for i in range(code_len):
        if code[i]==key[i]:
            return_list+='0'
        else:
            return_list+='1'
    return return_list



def _codefirstchange(self,code):
    changed_code=''
    for i in range(64):
        changed_code+=code[ip[i]-1]
    return changed_code



def _keyfirstchange (self,key):
    changed_key=''
    for i in range(56):
        changed_key+=key[pc1[i]-1]
    return changed_key
```

```
def _functionCodeChange(self, code):
    lens=len(code)/4
    return_list=''
    for i in range(lens):
        list=''
        for j in range(4):
            list+=code[ip_1[i*4+j]-1]
        return_list+="%x" %int(list,2)
    return return_list



def _functionE(self,code):
    return_list=''
    for i in range(48):
        return_list+=code[e[i]-1]
    return return_list



def _functionP(self,code):
    return_list=''
    for i in range(32):
        return_list+=code[p[i]-1]
    return return_list



def _functionS(self, key):
    return_list=''
    for i in range(8):
        row=int( str(key[i*6])+str(key[i*6+5]),2)
```

```
            raw=int(str( key[i*6+1])+str(key[i*6+2])+str(key[i*6+3])+str(key[i*6+4]),2)
            return_list+=self._functionTos(s[i][row][raw],4)

        return return_list


    def _functionKeySecondChange(self,key):
        return_list=''
        for i in range(48):
            return_list+=key[pc2[i]-1]
        return return_list


    def _functionCharToA(self,code,lens):
        return_code=''
        lens=lens%16
        for key in code:
            code_ord=int(key,16)
            return_code+=self._functionTos(code_ord,4)
        if lens!=0:
            return_code+='0'*(16-lens)*4
        return return_code


    def _functionTos(self,o,lens):
        return_code=''
        for i in range(lens):
            return_code=str(o>>i &1)+return_code
        return return_code
```

```
def tohex(string):

    return_string=''

    for i in string:

        return_string+="%02x"%ord(i)

    return return_string



def tounicode(string):

    return_string=''

    string_len=len(string)

    for i in range(0,string_len,2):

        return_string+=chr(int(string[i:i+2],16))

    return return_string




def desencode(from_code,key):
    from_code=tohex(from_code)
    key=tohex(key)
    des=DES()
    key_len=len(key)
    string_len=len(from_code)
    if string_len<1 or key_len<1:

        print 'error input'

        return False

    key_code= des.code(from_code,key,string_len,key_len)

    return key_code

# if __name__  == '__main__':

#   if(desencode(sys.argv[1],'mtqVwD4JNRjw3bkT9sQ0RYcZaKShU4sf')=='e3fab29a43a70ca72162a132df6ab53253527883

#       print 'correct.'

#   else:

#       print 'try again.'

__all__=['desdecode']
class DES():
    '''解密函数，DES加密与解密的方法相差不大
```

```
        只是在解密的时候所用的子密钥与加密的子密钥相反
    '''
    def __init__(self):
        pass

    def decode(self,string,key,key_len,string_len):
        output=""
        trun_len=0
        num=0

        #将密文转换为二进制
        code_string=self._functionCharToA(string,string_len)
        #获取字密钥
        code_key=self._getkey(key,key_len)


        #如果密钥长度不是16的整数倍则以增加0的方式变为16的整数倍
        real_len=(key_len/16)+1 if key_len%16!=0 else key_len/16
        trun_len=string_len*4
        #对每64位进行一次加密
        for i in range(0,trun_len,64):
            run_code=code_string[i:i+64]
            run_key=code_key[num%real_len]

            #64位明文初始置换
            run_code= self._codefirstchange(run_code)

            #16次迭代
            for j in range(16):

                code_r=run_code[32:64]
                code_l=run_code[0:32]

                #64左右交换
                run_code=code_r

                #右边32位扩展置换
                code_r= self._functionE(code_r)

                #获取本轮子密钥
                key_y=run_key[15-j]

                #异或
                code_r= self._codeyihuo(code_r,key_y)

                #S盒代替/选择
                code_r= self._functionS(code_r)

                #P转换
                code_r= self._functionP(code_r)

                #异或
                code_r= self._codeyihuo(code_l,code_r)

                run_code+=code_r
            num+=1

            #32互换
            code_r=run_code[32:64]
            code_l=run_code[0:32]
```

```
            code_l=run_code[0:32]
            run_code=code_r+code_l

            #将二进制转换为16进制、逆初始置换
            output+=self._functionCodeChange(run_code)
        return output

    #获取子密钥
    def _getkey(self,key,key_len):

        #将密钥转换为二进制
        code_key=self._functionCharToA(key,key_len)

        a=['']*16
        real_len=(key_len/16)*16+16 if key_len%16!=0 else key_len

        b=['']*(real_len/16)
        for i in range(real_len/16):
            b[i]=a[:]
        num=0
        trun_len=4*key_len
        for i in range(0,trun_len,64):
            run_key=code_key[i:i+64]
            run_key= self._keyfirstchange(run_key)
            for j in range(16):
                key_l=run_key[0:28]
                key_r=run_key[28:56]
                key_l=key_l[d[j]:28]+key_l[0:d[j]]
                key_r=key_r[d[j]:28]+key_r[0:d[j]]
                run_key=key_l+key_r
                key_y= self._functionKeySecondChange(run_key)
                b[num][j]=key_y[:]
            num+=1

        return b

    #异或
    def _codeyihuo(self,code,key):
        code_len=len(key)
        return_list=''
        for i in range(code_len):
            if code[i]==key[i]:
                return_list+='0'
            else:
                return_list+='1'
        return return_list

    #密文或明文初始置换
    def _codefirstchange(self,code):
        changed_code=''
        for i in range(64):
            changed_code+=code[ip[i]-1]
        return changed_code

    #密钥初始置换
    def _keyfirstchange (self,key):
        changed_key=''
        for i in range(56):
            changed_key+=key[pc1[i]-1]
        return changed_key
```

```
#逆初始置换
def _functionCodeChange(self, code):
    return_list=''
    for i in range(16):
        list=''
        for j in range(4):
            list+=code[ip_1[i*4+j]-1]
        return_list+="%x" %int(list,2)
    return return_list

#扩展置换
def _functionE(self,code):
    return_list=''
    for i in range(48):
        return_list+=code[e[i]-1]
    return return_list

#置换P
def _functionP(self,code):
    return_list=''
    for i in range(32):
        return_list+=code[p[i]-1]
    return return_list

#S盒代替选择置换
def _functionS(self, key):
    return_list=''
    for i in range(8):
        row=int( str(key[i*6])+str(key[i*6+5]),2)
        raw=int(str( key[i*6+1])+str(key[i*6+2])+str(key[i*6+3])+str(key[i*6+4]),2)
        return_list+=self._functionTos(s[i][row][raw],4)

    return return_list

#密钥置换选择2
def _functionKeySecondChange(self,key):
    return_list=''
    for i in range(48):
        return_list+=key[pc2[i]-1]
    return return_list

#将十六进制转换为二进制字符串
def _functionCharToA(self,code,lens):
    return_code=''
    lens=lens%16
    for key in code:
        code_ord=int(key,16)
        return_code+=self._functionTos(code_ord,4)

    if lens!=0:
        return_code+='0'*(16-lens)*4
    return return_code

#二进制转换
def _functionTos(self,o,lens):
    return_code=''
    for i in range(lens):
        return_code=str(o>>i &1)+return_code
```

```
        return return_code

#将unicode字符转换为16进制
def tohex(string):
    return_string=''
    for i in string:
        return_string+="%02x"%ord(i)
    return return_string

def tounicode(string):
    return_string=''
    string_len=len(string)
    for i in range(0,string_len,2):
        return_string+=chr(int(string[i:i+2],16))
    return return_string

#入口函数
def desdecode(from_code,key):
    key=tohex(key)

    des=DES()

    key_len=len(key)
    string_len=len(from_code)
    if string_len%16!=0:
        return False
    if string_len<1 or key_len<1:
        return False

    key_code= des.decode(from_code,key,key_len,string_len)
    return tounicode(key_code)

#测试
if __name__  == '__main__':
    print desdecode('e3fab29a43a70ca72162a132df6ab532535278834e11e6706c61a1a7cefc402c8ecaf601d00eee72','mtq
```

运行，拿到结果