# Rik

Wednesday, 15 January 2014

## [HACKYOU 2014] crypto300 - Matrix

So for this crypto challenge, you're given a file, encrypter.py, and another file, flag.wmv.out. Looking at the encrypter file, you can tell that the original file is broken up into blocks of 16 bytes each which are then transformed into 4x4 matrices. Each of these matrices is then multiplied by a key that was generated earlier and the resulting matrix is turned back into a string of bytes and written to the output file.

```
#!/usr/bin/python
import random
from struct import pack
from struct import unpack
from scipy import linalg


def Str2matrix(s):
        #convert string to 4x4 matrix
        return [map(lambda x : ord(x), list(s[i:i+4])) for i in xrange(0, len(s), 4)]

def Matrix2str(m):
        #convert matrix to string
        return ''.join(map(lambda x : ''.join(map(lambda y : pack('!H', y), x)), m))

def mMatrix2str(m):
        return ''.join(map(lambda x : ''.join(map(lambda y : pack('!B', y), x)), m))

def Generate(password):
        #generate key matrix
        random.seed(password)
        return [[random.randint(0,64) for i in xrange(4)] for j in xrange(4)]

def Multiply(A,B):
        #multiply two 4x4 matrix
        C = [[0 for i in xrange(4)] for j in xrange(4)]
        for i in xrange(4):
                for j in xrange(4):
                        for k in xrange(4):
                                C[i][j] += (A[i][k] * B[k][j])
        return C

def Encrypt(fname):
        #encrypt file
        key = Generate('')
        data = open(fname, 'rb').read()
        length = pack('!I', len(data))
        while len(data) % 16 != 0:
                data += '\x00'
        out = open(fname + '.out', 'wb')
        out.write(length)
        for i in xrange(0, len(data), 16):
                print Str2matrix(data[i:i+16])
                cipher = Multiply(Str2matrix(data[i:i+16]), key)
                out.write(Matrix2str(cipher))
        out.close()

Encrypt('sample.wmv')

```

encrypter.py hosted with by GitHub view raw

The key to this challenge was to take a look at the specification for the file format.

> A *WMV file* is in most circumstances encapsulated in the Advanced Systems Format (ASF) container format.

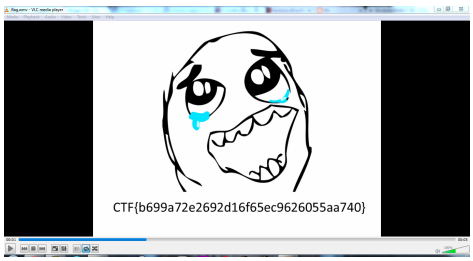
Looking at the ASF specification, these types of file usually start with a 16 byte GUID that identifies the file type. This hints at a known-plaintext attack. Using some basic linear algebra, given the plaintext and the ciphertext for the first 16 bytes of the file, it is possible to recover the key matrix. Once this key matrix is recovered, the rest of the file can be decrypted and the original wmv file can be recovered. The details of the steps involving the calculations are explained in comments in the code below.

```
1
2  from scipy import linalg
3  import numpy as np
4  from struct import pack,unpack
5  import sys
6
7  filename = 'flag.wmv' if len(sys.argv)==1 else sys.argv[1]
8
9  m_transform = np.frompyfunc(lambda x: int(round(x)),1,1)
10
11 header_byte_seq  = [0x30,0x26,0xB2,0x75,0x8E,0x66,0xCF,0x11,0xA6,0xD9,0x00,0xAA,0x00,0x62,0xCE,0x6C]
12 #turn header_byte_seq into a 4x4 matrix
13 hbs_matrix = np.array( header_byte_seq ).reshape(4,4)
14
15
16
17 #get ciphertext
18 ciphertext_file = open(filename+'.out','rb')
19 ciphertext = ciphertext_file.read()
20 ciphertext_file.close()
21
22 #ciphertext was packed as a series of shorts
23
24 #get length
25 length = unpack('!I',ciphertext[0:4])[0]
26 ciphertext = ciphertext[4:]
27
28 #convert into list integers
29 ciphertext = [ unpack('!H',ciphertext[i*2:i*2+2])[0] for i in range(0,length) ]
30
31 #first 16 bytes hold the header guid
32 hbs_ciphertext = ciphertext[0:16]
33
34
35
36 #RECOVER KEY USED FOR ENCRYPTION
37
38 # Let hbs_matrix = B
39 # Let key = K
40 # Let hbs_ciphertext = C
41 # BK = C
42 # (B^(-1)B)K = B^(-1)C
43 # K = B^(-1)C
44
45 B = hbs_matrix
46 C = np.array(hbs_ciphertext).reshape(4,4)
47 B_inverse = linalg.inv(B)
48 K = m_transform(B_inverse.dot(C))
49
50
51 #RECOVER ORIGINAL DATA GIVEN KEY AND CIPHERTEXT
52 # BK = C
53 # BK.K^(1) = C.K^(-1)
54 # B = C.K^(-1)
55
56 K_inverse = linalg.inv(K)
```

```
57
58  plaintext = []
59
60  for i in range(0,length/16):
61          C = ciphertext[i*16:i*16+16]
62          C = np.array(C).reshape(4,4)
63          B = m_transform(C.dot(K_inverse))
64          plaintext += [x for x in B.reshape(1,16)[0]]
65
66
67  #WRITE DECRYPTED DATA TO FILE
68
69  decrypted_file = open(filename,'wb')
70  data = ''.join( pack('!B',x) for x in plaintext )
71  decrypted_file.write(data)
72  decrypted_file.close()
```

hackyou_crypto300.py hosted with by GitHub view raw

Once the file has been decrypted, the wmv file is playable and it reveals the flag.




Posted by Jarred at 12:58

## No comments:

## Post a Comment

Enter your comment...

Comment as: Select profile...

Publish Preview

Newer Post Home Older Post

**Blog Archive**