

Crypto日记之利用CyberChef解png图片xor难题

原创

Sm0ry 已于 2022-03-10 00:25:51 修改 1349 收藏 2

分类专栏: [Crypto日记](#) 文章标签: [安全](#) [算法](#)

于 2022-03-10 00:24:25 首次发布

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: <https://blog.csdn.net/zgzhzywzd/article/details/123390548>

版权



[Crypto日记](#) 专栏收录该内容

9 篇文章 2 订阅

订阅专栏

今天晚上在刷题时发现了两道png图片加密的题目, 都是xor, 发现用CyberChef都能秒出, 下面把解题过程整理下。

第一题: 攻防世界Crypto sleeping-guard

题目来源: csaw-ctf-2016-quals

题目描述: 只有真正的hacker才能看到这张图片。(据说原是有个py的代码, 包含key的长度。)

题目场景给了ip和端口, 直接用nc连接发现一长串编码, 这么长串的编码, 先from base64一波:

output是一段乱码, 不过好像看到了RKEAey连续的字符, 就像是%PNG这样的字符, 不管怎样先保存成png后缀的图片, 就保存为RKEAey.png吧。010打开看一下, 并对比一下png的文件头:

```

/Users/smoryxu/Downloads/
RKEAey.png x
0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF
0000h: DE 3F 0F 2F 52 4B 45 41 65 79 21 32 1E 27 05 3A p?./RKEAey!2.'.:
0010h: 5F 41 5E B7 65 79 21 97 5F 69 41 68 5F 44 65 17 _A^ey!-_iAh_De.
0020h: 23 79 21 3F 53 08 00 25 1E 41 5F FA EA 72 DD 5E #y! ?S.%.A_üêrY^
0030h: 52 6F 41 68 7F 22 17 19 28 79 21 45 71 6F 41 E8 RoAh."..(y!EqoAè
0040h: DB 41 5F B1 65 79 21 BF BF 6F 41 1D 6F 41 5F A1 ÛA_ey!¿¿oA.oA_i
0050h: 05 79 21 05 CF 6F 41 7F 2F DD E5 1A 59 79 21 3F .y!.IoA./Yâ.Yy!?
0060h: 5E 1F 09 31 2C 41 5F 59 11 79 21 2D 23 6E 9F 0E ^..1,A_Y.y!-#nY.
0070h: 40 39 5F 4B 64 20 48 6B 0F 1B 19 25 13 7B 3C 24 @9_Kd Hk...%.{<$
0080h: 08 57 40 5B 38 0D 24 46 27 2C 2F 4B 65 79 21 3F .w@[8.$F',/Key!?
0090h: 6B 17 7B 10 32 31 32 2E 11 18 01 47 3A 03 2F 1B k.{.212...G:./
00A0h: 65 39 62 69 04 1D 4E 5D 32 55 2F 1B 65 2C 3A 3F e9bi..N]2U/.e,:?
00B0h: 04 56 03 1F 2F 55 39 05 2F 35 34 76 47 21 6C 6F .V../U9./54vG!lo
00C0h: 77 2C 2E 1A 3A 61 6A 65 51 57 11 1D 69 65 61 48 w,..:ajeQW..ieaH
00D0h: 7F 7D 2D 2F 03 43 73 7B 11 4F 39 05 33 2F 2C 71 .}-/..Cs{.09.3/,q
00E0h: 17 1D 47 02 75 07 35 1C 2F 7B 70 64 12 0E 56 11 ..G.u.5./{pd..V.
00F0h: 20 5C 6F 07 2D 26 70 7A 5C 40 18 10 67 5D 6E 5A \o.-&pz\@..g]nZ
0100h: 6D 6C 2D 2F 03 54 52 46 39 1B 20 10 72 2F 2C 68 m1-/.TRF9..r/,h
0110h: 47 47 2B 1F 77 4F 61 48 7F 7D 2D 2F 03 43 65 5A GG+.wOaH.}-/.CeZ
0120h: 24 0C 33 01 2F 35 36 24 0B 59 53 5B 31 55 20 0A $.3./56$.YS[1U .
0130h: 30 34 2B 76 47 5B 2B 1F 77 4F 61 48 7F 61 7F 6B 04+vG[+.wOaH.a.k
0140h: 45 59 01 47 3A 03 2F 1B 65 35 36 2D 03 44 03 57 EY.G:./..e56-.D.W
CSDN @SmOry

```

```

RKEAey.png Before.png x
0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF
0000h: 89 50 4E 47 0D 0A 1A 0A 00 00 00 0D 49 48 44 52 %PNG.....IHDR
0010h: 00 00 00 D7 00 00 00 D7 08 06 00 00 00 89 7D C4 ...x...x...%}A
0020h: B5 00 00 00 19 74 45 58 74 53 6F 66 74 77 61 72 µ...tEXtSoftwar
0030h: 65 00 41 64 6F 62 65 20 49 6D 61 67 65 52 65 61 e.Adobe ImageRea
0040h: 64 79 71 C9 65 3C 00 00 03 B5 69 54 58 74 58 4D dyqEes...µiTXtXM
0050h: 4C 3A 63 6F 6D 2E 61 64 6F 62 65 2E 78 6D 70 00 L:com.adobe@smory

```

misc的题解多了应该对png头很熟悉，具体的头包含哪些内容这里就不详细解释了，感兴趣的可以搜一搜。

这里要说明的是png文件头的前16位是固定的 **89 50 4E 47 0D 0A 1A 0A 00 00 00 0D 49 48 44 52**，那么可以根据png加密后的前16位来异或出key的值，如果key是一段有意思的字符串就很容易辨别出来。

本题png加密后的前16位为：**DE 3F 0F 2F 52 4B 45 41 65 79 21 32 1E 27 05 3A**

接下来就轮到神器CyberChef了，将固定长度的原文和cypher异或，就能得到key，这里我们把前16位全部放进去，input先From Hex，再XOR Cipher，就可以得到key，因为是循环xor，所以key应该是WoAh_A_Key!?!，刚好12位：

The screenshot shows the CyberChef interface with the following configuration:

- Recipe:** From Hex, XOR
- Input:** 89 50 4E 47 0D 0A 1A 0A 00 00 00 0D 49 48 44
- From Hex:** Delimiter: Auto
- XOR:** Key: DE 3F 0F 2F 52 4B 45 41 65 79 ... (HEX)
- Scheme:** Standard, Null preserving (unchecked)
- Output:** WoAh_A_Key!?!WoA

解到这里得出了key，再利用CyberChef对整个加密的png文件作xor，得到原文件：

Recipe 📄 📁 🗑️ **Input** Length: 126,909 total: 2 Loaded: 2 + 📄 🔄 🗑️ 📄

XOR 🔇 ⏸️

Key: UTF8 ▾

Scheme: Null preserving

Output 🔍 time: 21ms length: 126909 lines: 544 + 📄 🔄 🗑️ 📄

Input < 1: 89 50 4E 47 0D 0A 1A 0A 00 00 00 0... × **2: RKEAey.png** × > ...

 Name: RKEAey.png
Size: 126,909 bytes
Type: image/png
Loaded: 100%

Output < 1: 89 50 4E 47 0D 0A 1A 0A 00 00 00 0D 49 ... **2: PNG IHDR ü.....\F....gAMA±.üa....** > ...

```
.PNG
.
...
IHDR...ü.....\F....gAMA±.üa....
CHRM..z&.....ú...è..u0..ê`.....p.ºQ<...
pHYs...t...t.Pf.X...YiTXtXML:com.adobe.xmp.....<x:xmpmeta xmlns:x="adobe:ns:meta/"
x:xmptk="XMP Core 5.4.0">
  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <rdf:Description rdf:about=""
      xmlns:tiff="http://ns.adobe.com/tiff/1.0/">
      <tiff:Orientation>1</tiff:Orientation>
    </rdf:Description>
```

STEP 👨‍🍳 **BAKE!** Auto Bake

CSDN @Sm0ry

惊喜，看到png的文件头了吗？保存下来就得到了flag：



第二题：攻防世界Crypto beginners-luck

题目来源：bitsctf-2017

这道题跟上面这道题差不多，并给了加密png的脚本：

```
#!/usr/bin/env python

def supa_encryption(s1, s2):
    res = [chr(0)]*24
    for i in range(len(res)):
        q = ord(s1[i])
        d = ord(s2[i])
        k = q ^ d
        res[i] = chr(k)
    res = ''.join(res)
    return res

def add_pad(msg):
    L = 24 - len(msg)%24
    msg += chr(L)*L
    return msg

with open('fullhd.png','rb') as f:
    data = f.read()

data = add_pad(data)

with open('key.txt') as f:
    key = f.read()

enc_data = ''
for i in range(0, len(data), 24):
    enc = supa_encryption(data[i:i+24], key)
    enc_data += enc

with open('BITSCTFfullhd.png', 'wb') as f:
    f.write(enc_data)
```

看了脚本大概明白这题是利用一个长度为 24 的 key 循环异或原图，得到一个一堆乱码的png。上面那题的key长度是12位，这题的key是24位。

而png文件头前16位是固定的，再后面8位就是图片的长和宽，根据信息可以得到图片的尺寸是1920x1080，那么就可以得到原图的前24位固定值：**89 50 4E 47 0D 0A 1A 0A 00 00 00 0D 49 48 44 52 00 00 07 80 00 00 04 38;**

BITSCTFfullhd.png也就是加密后的png前24位为：**FB 3B 26 62 5C 5A 2E 6D 30 26 33 6A 7D 7E 04 66 2A 25 61 A8 55 4E 27 64;**

Last build: 6 months ago Options About / Support

Recipe	Input
<p>From Hex </p> <p>Delimiter Auto</p>	<p>length: 48 lines: 1 </p> <p>89504E470D0A1A0A0000000D494844520000078000000438</p>
<p>XOR </p> <p>Key FB3B26625C5A2E6D3026336A7D7E04... HEX ▾</p> <p>Scheme Standard <input type="checkbox"/> Null preserving</p>	<p>time: 1ms length: 24 lines: 1 </p> <p>Output</p> <p>rkh%QP4g0&3g46@4*%f(UN#)</p>

CSDN @Sm0ry

利用 CyberChef很快得到key为: **rkh%QP4g0&3g46@4*%f(UN#)**

再利用key还原图片:

Recipe length: 44,904 total: 2 loaded: 2

Recipe	Input
<p>XOR </p> <p>Key rkh%QP4g0&3g46@4*%f(UN#) UTF8 ▾</p> <p>Scheme Standard <input type="checkbox"/> Null preserving</p>	<p>< 1: 89504E470D0A1A0A0000000D4948... X 2: BITSCTFfullhd.png X > ...</p> <div style="border: 1px solid gray; padding: 5px; width: fit-content;"> <p> Name: BITSCTFfullhd.png Size: 44,904 bytes Type: image/png Loaded: 100%</p> </div> <p>time: 15ms length: 44904 lines: 107 </p> <p>< 1: 89504E470D0A1A0A0000000D49484452... 2: PNG IHDR 8 g z V s RGB 0 ! ... > ...</p> <p>.PNG ... IHDR.....8.....g z V.....sRGB.0! é...gAMA..±.üa... pHYs...t...t.Pf.x...@EIDATx^iy+t^!Ä..èI: ws0ÄYEE~u...EZG.ds.ä.H\$....DFFFF".E}k0Ys>.<.. yø>5U..i@@ÇOu0`y.....A..@.....p..ð.....4.....G! ...ÄQ.@.....p..ð.....4.....G! ...ÄQ.@.....p..ð.....4.....G! ...ÄQ.@.....p..ð.....4.....G! ...ÄQ.@.....p..ð.....4.....G! ...ÄQ.@.....p..ð.....4.....G! ...ÄQ.@.....p..ð.....4.....G!</p>

STEP **BAKE!** Auto Bake CSDN @Sm0ry

保存下来就是flag了:

BITSCTF
{p-en-ge}

CSDN @Sm0ry