

asis-ctf的writeup收集

原创

beyondkmp 于 2014-05-15 09:05:36 发布 4081 收藏 1

分类专栏： 网络攻防

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/u011500307/article/details/25838075>

版权



[网络攻防 专栏收录该内容](#)

3篇文章 0订阅

订阅专栏

1. Random Image

官方给出的加密代码如下：

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

import Image
import random

def get_color(x, y, r):
    n = (pow(x, 3) + pow(y, 3)) ^ r
    return (n ^ ((n >> 8) << 8))

flag_img = Image.open("flag.png")
im = flag_img.load()
r = random.randint(1, pow(2, 256))
print flag_img.size

enc_img = Image.new(flag_img.mode, flag_img.size)
enpix = enc_img.load()

for x in range(flag_img.size[0]):
    for y in range(flag_img.size[1]):
        t = random.randint(1, pow(2, 256)) % 250
        enpix[x,y] = t

for x in range(flag_img.size[0]):
    for y in range(flag_img.size[1]):
        if im[x,y] < 250 :
            s = get_color(x, y, r)
            enpix[x,y] = s

enc_img.save('enc' + '.png')
```

加密后的图片如下（enc.png）：



分析加密代码主要就是当原来的像素小于250的时候就经过get_color()来获得相应的像素值。

get_color()这个函数就是取 $(x^{**}3+y^{**}3)^r$ 的最后8位，虽然不能直接求出 r ,但是只要知道其最后八位就可以了。可以用暴力从0-255。初始化enc.png的像素的时候都是取250的模的所以应该都是小于250,对于那些大于250的值一定是从get_color()中获得。

(1) 先获得像素点大于250的点

```
import Image

enc_img=Image.open('enc.png')
im=enc_img.load()
count=0

for x in range(enc_img.size[0]):
    for y in range(enc_img.size[1]):
        if im[x,y]>=250:
            print x,y,im[x,y]
            count+=1

print count
```

一部分的结果如下：

```
1317 64 251
1323 29 254
1325 27 254
1326 61 251
1330 43 253
1334 44 254
1335 45 250
1344 37 251
1345 54 255
1347 32 253
1347 40 253
1347 48 253
1347 56 253
1350 32 254
1350 56 254
1353 59 250
1366 29 251
1366 59 253
1368 37 251
1372 54 254
```

在其中随便选中一点来实现暴力求解 r

```
def get_color(x, y, r):
    n = (pow(x, 3) + pow(y, 3)) ^ r
    return (n ^ ((n >> 8) << 8))

for i in range(256):
    if get_color(1372, 54, i) == 254:
        print i
        break
```

```
beyond@beyond ~/下载 $ python dec3.py
38
```

r的结果是38,则可以反解出相应的图片

```
import Image

enc_img=Image.open('enc.png')
im=enc_img.load()
count=0

dec_img=Image.new(enc_img.mode,enc_img.size)
print enc_img.mode
depix=dec_img.load()

for x in range (enc_img.size[0]):
    for y in range(enc_img.size[1]):
        if im[x,y]>=250:
            depix[x,y]=0
        elif (x**3+y**3)^38==im[x,y]:
            depix[x,y]=0
            print x,y
        else:
            depix[x,y]=255
print enc_img.size
dec_img.save('flag1'+''.png')
```

结果如下：

ASIS_05df5fedfc700926df42fcd591b791ec

2. Hidden flag

直接用burp suit手抓包如下：

```
HTTP/1.1 200 OK
Server: nginx
Date: Thu, 15 May 2014 00:43:11 GMT
Content-Type: text/html; charset=utf-8
Connection: keep-alive
Vary: Cookie, Accept-Language
X-Frame-Options: SAMEORIGIN
x-flag: ASIS_b6b?244608c2?c2e869cb56?67b64?b1
Content-Language: en-us
Set-Cookie: csrftoken=MnyNAPazKJLrEELqOTWPRlt7ODCNm3Yz; expires=Thu, 14-May-2015 00:42:56 GMT; Max-Age=3144
X-XSS-Protection: 1; mode = block
X-Content-Type-Options: nosniff
X-Hacker: Don't Be A Jerk
X-Powered-By: ASIS
Content-Length: 3731
```

可以看到x-flag有四个问号，再看下提交flag的网页的代码：

```
$(document).on('hidden.bs.modal', function (e) {
    e.preventDefault();
    $(e.target).removeData('bs.modal');

});

var i=0;

var result=['Please try again!', 'Try harder!', 'Your answer is not correct!', 'The submitted flag is n
var final_result="Do you want to hack me?";

$('#flag_submission').submit(function(e){
    e.preventDefault();
    var shaObj = new jsSHA(document.forms["flag_submission"]["id_flag"].value, "TEXT");
    var hash = shaObj.getHash("SHA-256", "HEX");
    var shaObj2 = new jsSHA(hash, "TEXT");
    var hash2 = shaObj2.getHash("SHA-256", "HEX");
    if (document.forms["flag_submission"]["check"].value !== hash2) {
        if ($("#id_flag").next().length == 0){
            $('<div class="alert alert-danger" id="answer" />').insertAfter('#id_flag');
        }
        if (i++>6){
            $('#answer').removeClass('alert-danger').addClass('alert');
            $('#answer').text(final_result);
        }
        else $('#answer').text(result[Math.floor(Math.random() * 7)]);
        return false;
    }
})
```

可以看出上面是两次sha-256,再和check value的值比较，check value也可以从网页中获得是：

```
<input id="id_check" name="check" type="hidden" value="61e18627ead3caaf56c89140e11533491ea3cc7b405d3e4d95bb
```

现在可以直接暴力求解这四个字符了。代码如下：

```
import hashlib

hexs="0123456789abcdef"

for a in hexs:
    for b in hexs:
        for c in hexs:
            for d in hexs:
                flag='ASIS_b6b%sc244608c2%sc2e869cb56%s67b64%sb1' %(a,b,c,d)
                flaghash=hashlib.sha256(flag).hexdigest()
                flaghash=hashlib.sha256(flaghash).hexdigest()
                if flaghash=='61e18627ead3caaf56c89140e11533491ea3cc7b405d3e4d95bba333860c0acc':
                    print a,b,c,d
                    print flag
                    break
```

代码运行如下：

```
beyond@beyond ~/code/code-python $ python hiddenflag.py
9 f 0 b
ASIS_b6b9244608c2fc2e869cb56067b64bb1
```

3. Prying ears

89504e470d0a1a

打开文件是一个pcap文件，在wireshark里面观察可以看到很多dns，这里第一个dns的地址是89504e470d0a1a.asis.io，这个89504e470d0a1a是png的开始的几个数据，叫做magic numbers.所以可以猜测应该是所有的dns请求的第一部分组合成了一个Png图片。

先是用dns.flags == 0x8180 and dnsqry.name matches "[0-9a-f]{14}.asis.io" 来过滤出所有的dns，如下图

forensic_175_d78a42edc01c9104653776f16813d9e5 [Wireshark 1.10.6 (Git Rev Unknown from unknown)]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: dns.flags == 0x8180 and dns.qry.name matches "[0-9a-f]{14}" Expression... Clear Apply 保存

No.	Time	Source	Destination	Protocol	Length	Info
196	34.127735000	192.168.11.4	192.168.110.2	DNS	98	Standard query response 0x5ca8 A 87.107.124.13
200	38.234670000	192.168.11.4	192.168.110.2	DNS	98	Standard query response 0x4a88 A 87.107.124.13
204	40.391913000	192.168.11.4	192.168.110.2	DNS	98	Standard query response 0xe45a A 87.107.124.13
208	44.505796000	192.168.11.4	192.168.110.2	DNS	98	Standard query response 0xe2bf A 87.107.124.13
212	45.626635000	192.168.11.4	192.168.110.2	DNS	98	Standard query response 0x991e A 87.107.124.13
216	47.735131000	192.168.11.4	192.168.110.2	DNS	98	Standard query response 0x2d30 A 87.107.124.13
224	51.846797000	192.168.11.4	192.168.110.2	DNS	98	Standard query response 0x28f2 A 87.107.124.13
231	54.949051000	192.168.11.4	192.168.110.2	DNS	98	Standard query response 0xa7c2 A 87.107.124.13
235	55.044936000	192.168.11.4	192.168.110.2	DNS	98	Standard query response 0x99ac A 87.107.124.13
245	58.324089000	192.168.11.4	192.168.110.2	DNS	98	Standard query response 0xfb4 A 87.107.124.13
249	58.427065000	192.168.11.4	192.168.110.2	DNS	98	Standard query response 0xf967 A 87.107.124.13
254	58.531237000	192.168.11.4	192.168.110.2	DNS	98	Standard query response 0xcdab A 87.107.124.13
258	58.638083000	192.168.11.4	192.168.110.2	DNS	98	Standard query response 0x856e A 87.107.124.13
273	59.755200000	192.168.11.4	192.168.110.2	DNS	98	Standard query response 0x8cd1 A 87.107.124.13
283	61.868361000	192.168.11.4	192.168.110.2	DNS	98	Standard query response 0xa7a6 A 87.107.124.13
290	62.979300000	192.168.11.4	192.168.110.2	DNS	98	Standard query response 0xaeb4 A 87.107.124.13
294	63.088279000	192.168.11.4	192.168.110.2	DNS	98	Standard query response 0xaea7c A 87.107.124.13
329	66.278105000	192.168.11.4	192.168.110.2	DNS	98	Standard query response 0xa6a1 A 87.107.124.13
333	66.565223000	192.168.11.4	192.168.110.2	DNS	98	Standard query response 0x1eff A 87.107.124.13

> Frame 196: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0

> Ethernet II, Src: D-LinkIn_73:d3:0e (78:54:2e:73:d3:0e), Dst: Apple_08:0a:b0 (60:33:4b:08:0a:b0)

> Internet Protocol Version 4, Src: 192.168.11.4 (192.168.11.4), Dst: 192.168.110.2 (192.168.110.2)

> User Datagram Protocol, Src Port: domain (53), Dst Port: 54600 (54600)

0000 60 33 4b 08 0a b0 78 54 2e 73 d3 0e 08 00 45 00 `3K...xT .s....E.

0010 00 54 13 e3 00 00 7c 11 30 5f c0 a8 0b 04 c0 a8 .T....|. 0.....

0020 6e 02 00 35 e8 d0 00 40 1a 2a 5c a8 81 80 00 01 n...5...@ ..*\....

0030 00 01 00 00 00 00 0e 38 39 35 30 34 65 34 37 308 9504e470

Ready to load or capture Packets: 6867 Display... Profile: Default

再就是选择file---> export specified packets

forensic_1 Wireshark: Export Specified Packets forensic_175_d78a42edc01c9104653776f16813d9e5 [Wireshark 1.10.6 (Git Rev Unknown from unknown)]

File Edit View Go Capt

名称(N):

保存于文件夹(E): beyond 下载 创建文件夹(L)

Filter: dns.flags == 0x8180

No.	Time	Source	Destination	Protocol	Length	Info
196	34.127735000	192.168.11.4	192.168.110.2	DNS	98	Standard query response 0x5ca8 A 87.107.124.13
200	38.234670000	192.168.11.4	192.168.110.2	DNS	98	Standard query response 0x4a88 A 87.107.124.13
204	40.391913000	192.168.11.4	192.168.110.2	DNS	98	Standard query response 0xe45a A 87.107.124.13
208	44.505796000	192.168.11.4	192.168.110.2	DNS	98	Standard query response 0xe2bf A 87.107.124.13
212	45.626635000	192.168.11.4	192.168.110.2	DNS	98	Standard query response 0x991e A 87.107.124.13
216	47.735131000	192.168.11.4	192.168.110.2	DNS	98	Standard query response 0x2d30 A 87.107.124.13
224	51.846797000	192.168.11.4	192.168.110.2	DNS	98	Standard query response 0x28f2 A 87.107.124.13
231	54.949051000	192.168.11.4	192.168.110.2	DNS	98	Standard query response 0xa7c2 A 87.107.124.13
235	55.044936000	192.168.11.4	192.168.110.2	DNS	98	Standard query response 0x99ac A 87.107.124.13
245	58.324089000	192.168.11.4	192.168.110.2	DNS	98	Standard query response 0xfb4 A 87.107.124.13
249	58.427065000	192.168.11.4	192.168.110.2	DNS	98	Standard query response 0xf967 A 87.107.124.13
254	58.531237000	192.168.11.4	192.168.110.2	DNS	98	Standard query response 0xcdab A 87.107.124.13
258	58.638083000	192.168.11.4	192.168.110.2	DNS	98	Standard query response 0x856e A 87.107.124.13
273	59.755200000	192.168.11.4	192.168.110.2	DNS	98	Standard query response 0x8cd1 A 87.107.124.13
283	61.868361000	192.168.11.4	192.168.110.2	DNS	98	Standard query response 0xa7a6 A 87.107.124.13
290	62.979300000	192.168.11.4	192.168.110.2	DNS	98	Standard query response 0xaeb4 A 87.107.124.13
294	63.088279000	192.168.11.4	192.168.110.2	DNS	98	Standard query response 0xaea7c A 87.107.124.13
329	66.278105000	192.168.11.4	192.168.110.2	DNS	98	Standard query response 0xa6a1 A 87.107.124.13
333	66.565223000	192.168.11.4	192.168.110.2	DNS	98	Standard query response 0x1eff A 87.107.124.13

> Frame 196: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0

> Ethernet II, Src: D-LinkIn_73:d3:0e (78:54:2e:73:d3:0e), Dst: Apple_08:0a:b0 (60:33:4b:08:0a:b0)

> Internet Protocol Version 4, Src: 192.168.11.4 (192.168.11.4), Dst: 192.168.110.2 (192.168.110.2)

> User Datagram Protocol, Src Port: domain (53), Dst Port: 54600 (54600)

0000 60 33 4b 08 0a b0 78 54 2e 73 d3 0e 08 00 45 00 `3K...xT .s....E.

0010 00 54 13 e3 00 00 7c 11 30 5f c0 a8 0b 04 c0 a8 .T....|. 0.....

0020 6e 02 00 35 e8 d0 00 40 1a 2a 5c a8 81 80 00 01 n...5...@ ..*\....

0030 00 01 00 00 00 00 0e 38 39 35 30 34 65 34 37 308 9504e470

Ready to load or capture Packets: 6867 Display... Profile: Default

保存好后可以直接用strings来提取此文件中的字符串，用grep过滤出所有的十六进制的字符串，再用tr命令去除换行符号。

```
beyond@beyond ~/下载 $ strings qweqwe.pcapng | grep '^[\x21-\x7e]\{14\}$' | tr -d '\n' > test.txt
```

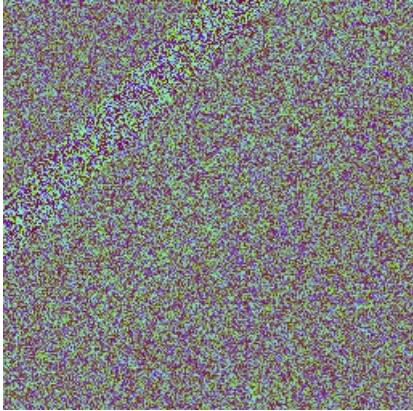
再将test.txt转化成相应的二进制png就可以了。

```
beyond@beyond ~/下载 $ xxd -r -p test.txt out.png
```

最后的图片如下：

ASIS_da8733c9d73930ea9e4d012e0e9c1d48

4. White noise



这个图片先是直接rgb展开看下具体内容：

```
from PIL import Image

im = Image.open("foto.png")
rgb_im = im.convert('RGB')
#size is 256 by 256
for x in xrange(0,255):
    for y in xrange(0,255):

        r, g, b = rgb_im.getpixel((x, y))
        print str(r) + " " + str(g) + " " + str(b)
```

结果如下：

```
R   G   B
128 80 239
128 171 83
128 165 100
128 136 219
128 165 161
128 68 224
128 119 60
...
```

可以猜测结果与r无关,只与gb有关。这个g,b的值可能就是坐标的值, 取前256*256/2个直接涂点就可以得到 (128可能是告诉我们取一半的点就可以了)

```
from PIL import Image

im=Image.open("foto.png")
rgb_im=im.convert('RGB')
i=0
c=[]
d=[]

dec_img=Image.new('L',(256,256))
depix=dec_img.load()

for x in range(256):
    for y in range(256):
        depix[x,y]=255

for x in xrange(0,255):
    for y in xrange(0,255):
        r,g,b=rgb_im.getpixel((x,y))
        print str(r)+" "+str(g)+" "+str(b)
        depix[int(g),int(b)]=0
        c.append(int(g))
        d.append(int(b))
        i=i+1
    if i >= 32768:
        break

dec_img.save('steg'+'.png')

print max(c)
print max(d)
print i
```

输出的图片如下：

