Pandora's box

Un site utilisant Blogs UP

ASIS 2014 quals # Crypto – Random Image

Posté le **10 mai 2014** - 1 374 vues

URL: http://asis-ctf.ir/challenges/
Type: XOR + grayscale threshold
Solution: ASIS_af4e8dcbbcdcef44fd3ecdbc6e9695d4

Description
Find the flag
file

Archive contains 2 files:

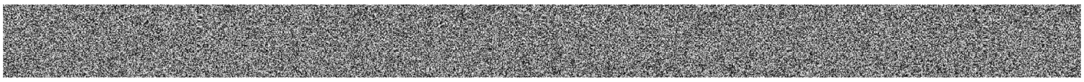
```
$ tar xvfJ crypto_150_8f3fd5d2bacd408904b8406c19183c23
x color_crypto.py
x enc.png
```

`color_crypto.py` is the Python source code of the algorithm used to encrypt the PNG file:

```
#!/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')
```

`enc.png` is a 8-bit grayscale PNG image:



ALGORITHM ANALYSIS

The coding scheme rely on grayscale threshold. Basicly, in original image:

- if pixel `gray level ≥ 250` then it is considered as *bright* pixel
- if pixel `gray level < 250` then it is considered as *dark* pixel

In encoded image:

- *bright* pixel is substitued with a random value:

  `random.randint(1, pow(2, 256)) % 250`
- *dark* pixel is computed in `get_color(x, y, r)` function, from its coordinates and a constant big random value `r = random.randint(1, pow(2, 256))`:

  `n = (pow(x, 3) + pow(y, 3)) ^ r`

  then it returns the 8 less significant bits (i.e. the less significant byte)

  `n ^ ((n >> 8) << 8 )`

  which somehow is equivalent to `n & 0xff`

WEAKNESS

The weakness here is that for each pixel the coordinates `x` and `y` are known, and `r` is a constant value. Knowing XOR associative property (`a^b=c` ⟺ `a^c=b` ⟺ `b^c=a`):

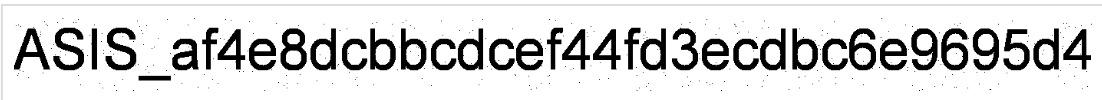
1. for each pixel of the encoded image, we XOR its value with $(x^3 + y^3)$
2. thus, the less significant byte of `r` is the most frequently occurring value
3. finally:
   - if a pixel of the encoded image XOR $(x^3 + y^3)$ = the less significant byte of `r`, then pixel of the original image was *dark* (let's say black)
   - else pixel of the original image was *bright* (let's say white)

   There could be some false positive (i.e. pixels that we considered *dark* when they were *bright*) due to randomness, but statistically they will be few.

PYTHON DECODER

```python
import Image

enc_img = Image.open('enc.png')
enc_pix = enc_img.load()

# 1. XOR all pixel values with corresponding (x**3 + y**3)
pow_pix = [(enc_pix[x,y] ^ (x**3 + y**3)) & 0xff
           for x in range(enc_img.size[0])
           for y in range(enc_img.size[1])]

# 2. r_LSbyte = most frequently occurring byte
r_LSbyte = max([b for b in range(0x100)], key = pow_pix.count)

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

for x in range(enc_img.size[0]):
  for y in range(enc_img.size[1]):
    if ((enc_pix[x, y] ^ (x**3 + y**3)) & 0xff == r_LSbyte):
      flag_pix[x, y] = 0              # original pixel was dark
    else:
      flag_pix[x, y] = 255            # original pixel was bright

flag_img.save('flag.png')
```

Running on `enc.png`, we obtain the following decoded PNG flag file (with less significant byte of `r = 0x3d`):



RELATED POSTS:

DEF CON 21 CTF Quals

DC21 quals # 3dub 03 - hypeman

ASIS 2014 final # Crypto - PTOT

Posté dans **ASIS**, **crypto**, **writeup** par **phoenix1204**.