常见文件文件头和隐写术总结 CTF中Misc必备

原创

思源湖的鱼 于 2020-11-24 22:48:25 发布 3400 收藏 48

分类专栏： cyber security 文章标签： misc ctf 文件头 图片隐写 音频隐写

本文链接：https://blog.csdn.net/weixin_44604541/article/details/110082054

前言

对常见文件文件头和隐写术做个归纳总结

一、文件头文件尾

1、图片

- JPEG
  文件头：`FF D8 FF`
  文件尾：`FF D9`
- TGA
  未压缩的前4字节 `00 00 02 00`
  RLE压缩的前5字节 `00 00 10 00 00`
- PNG
  文件头：`89 50 4E 47 0D 0A 1A 0A`
  文件尾：`AE 42 60 82`
- GIF
  文件头：`47 49 46 38 39(37) 61`
  文件尾：`00 3B`
- BMP
  文件头：`42 4D`
  文件头标识(2 bytes) 42(B) 4D(M)
- TIFF (tif)
  文件头：`49 49 2A 00`
- ico
  文件头：`00 00 01 00`
- Adobe Photoshop (psd)
  文件头：`38 42 50 53`

2、office文件

MS Word/Excel (xls.or.doc)
文件头：`D0 CF 11 E0`

MS Access (mdb)
文件头：`53 74 61 6E 64 61 72 64 20 4A`

WordPerfect (wpd)
文件头：`FF 57 50 43`

Adobe Acrobat (pdf)
文件头：`25 50 44 46 2D 31 2E`

application/vnd.visio(vsd)
文件头：`D0 CF 11 E0 A1 B1 1A E1`

Email [thorough only] (eml)
文件头：`44 65 6C 69 76 65 72 79 2D 64 61 74 65 3A`

Outlook Express (dbx)
文件头：`CF AD 12 FE C5 FD 74 6F`

Outlook (pst)
文件头：`21 42 44 4E`

Rich Text Format (rtf)
文件头：`7B 5C 72 74 66`

txt 文件(txt)
文件头：Unicode：`FE FF` / Unicode big endian：`FF FE` / UTF-8：`EF BB BF` /ANSI编码是没有文件头的

3、压缩包文件

- ZIP Archive (zip)
  文件头：`50 4B 03 04`
  文件尾：`50 4B`
- RAR Archive (rar)
  文件头：`52 61 72 21`

4、音频文件

- Wave (wav)
  文件头：`57 41 56 45`
- audio(Audio)
  文件头：`4D 54 68 64`
- audio/x-aac（aac）
- 文件头：`FF F1(9)`

5、视频文件

- AVI (avi)
  文件头：`41 56 49 20`
- Real Audio (ram)
  文件头：`2E 72 61 FD`
- Real Media (rm)
  文件头：`2E 52 4D 46`
- MPEG (mpg)
  文件头：`00 00 01 BA(3)`
- Quicktime (mov)
  文件头：`6D 6F 6F 76`
- Windows Media (asf)
  文件头：`30 26 B2 75 8E 66 CF 11`
- MIDI (mid)
  文件头：`4D 54 68 64`

6、代码文件

XML (xml)
文件头：`3C 3F 78 6D 6C`

HTML (html)
文件头：`68 74 6D 6C 3E`

Quicken (qdf)
文件头：`AC 9E BD 8F`

Windows Password (pwl)
文件头：`E3 82 85 96`

7、其他类型

- windows证书文件(der)
  文件头：`30 82 03 C9`
- CAD (dwg)
  文件头：`41 43 31 30`
- Windows Shortcut (lnk)
  文件头：`4C 00 00 00`
- Windows reg(reg)
  文件头：`52 45 47 45 44 49 54 34`

二、图片隐写

1、附加式的图片隐写

操作系统识别，从文件头标志，到文件的结束标志位
当系统识别到图片的结束标志位后，默认是不再继续识别的
所以可以在文件尾后面加东西

（1）附加字符串

最简单的是附加字符串

附加方法

- winhex直接附加再保存
- `copy /b a.jpg+b.txt c.jpg`，在a图片里加b的内容，得到c图片

识别方法

- winhex直接看
- notepad也可以看
- linux的strings指令

应用

- 制作图片马，即把木马放到图片的最后

实例

- Aesop_secret
- Training-Stegano-1

（2）隐藏压缩文件

可以把压缩文件藏在图片文件尾后
看起来还是图片

附加方法

- winhex直接附加再保存

识别方法

- 有些直接改扩展名就可以用
- linux的binwalk指令
- stegsolve分离
- winhex复制压缩文件内容重新保存

实例

- a_good_idea
- 攻防世界 Misc高手进阶区 2分题 Ditf
- 攻防世界 Misc高手进阶区 2分题 再见李华
- 攻防世界 Misc高手进阶区 3分题 miscmisc
- 攻防世界 Misc高手进阶区 3分题 3-11

2、基于文件结构的图片隐写

主要是针对PNG图片

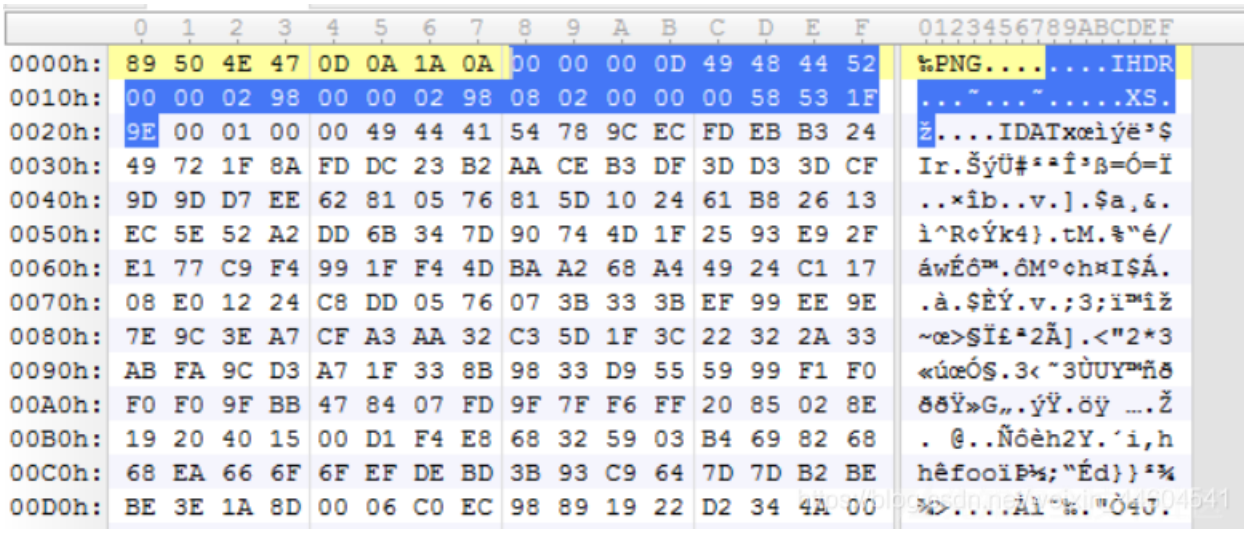
标准的PNG文件结构应包括：

- PNG文件标志
- PNG数据块：关键数据块和辅助数据块，其中正常的关键数据块有长度、数据块类型码、数据块数据和CRC这4种

（1）png图片文件头数据块（IHDR）

PNG图片的第一个数据块

- 一张PNG图片仅有一个IHDR数据块
- 包括了图片的宽，高，图像深度，颜色类型，压缩方法等信息



蓝色部分就是IHDR

可以修改高度值或宽度值对部分信息进行隐藏

- 如果图片原本是800(宽)*600(高)，然后图片的高度从600变成500
- 这样下面800×100区域的信息就无法从图片中显示出来，我们可见的只有上方800*500的区域，这样就达成了图片隐写的目的
- 同理可知图片的宽度也可以进行类似的修改以达到隐藏信息的目的

识别方法

- 用winhex或者010Editor等编辑器打开图片
- 修改长度或宽度值
- 在修改文件后，需要利用CRC Calculator对CRC校验码进行重新计算赋值，以防图片被修改后，自身的CRC校验报错，导致图片不能正常打开

实例

- 攻防世界 Misc高手进阶区 2分题 Ditf
- 攻防世界 Misc高手进阶区 3分题 2-1

（2）IDAT 数据块

- 存储实际的数据
- 在数据流中可包含多个连续顺序的图像数据块
- 写入一个多余的IDAT也不会多大影响肉眼对图片的观察

识别方法

- 用pngcheck对图片进行检测 `pngcheck -v hidden.png`

```
Type: IHDR
Size: 13
CRC :  5412913F

Pos :  33
Type: IDAT
Size: 10980
CRC :  98F96EEB

Pos :  11025
Type: IEND
Size: 0
CRC :  AE426082
```

可能会出现一个size为0的异常块

提取内容的脚本

```
#!/usr/bin/python

from struct import unpack
from binascii import hexlify, unhexlify
import sys, zlib

# Returns [Position, Chunk Size, Chunk Type, Chunk Data, Chunk CRC]
def getChunk(buf, pos):
    a = []
    a.append(pos)
    size = unpack('!I', buf[pos:pos+4])[0]
    # Chunk Size
    a.append(buf[pos:pos+4])
    # Chunk Type
    a.append(buf[pos+4:pos+8])
    # Chunk Data
    a.append(buf[pos+8:pos+8+size])
    # Chunk CRC
    a.append(buf[pos+8+size:pos+12+size])
    return a

def printChunk(buf, pos):
    print 'Pos : '+str(pos)+''
    print 'Type: ' + str(buf[pos+4:pos+8])
    size = unpack('!I', buf[pos:pos+4])[0]
    print 'Size: ' + str(size)
    #print 'Cont: ' + str(hexlify(buf[pos+8:pos+8+size]))
    print 'CRC : ' + str(hexlify(buf[pos+size+8:pos+size+12]).upper())
    print
```

```
if len(sys.argv)!=2:
    print 'Usage: ./this Stegano_PNG'
    sys.exit(2)

buf = open(sys.argv[1]).read()
pos=0

print "PNG Signature: " + str(unpack('cccccccc', buf[pos:pos+8]))
pos+=8

chunks = []
for i in range(3):
    chunks.append(getChunk(buf, pos))
    printChunk(buf, pos)
    pos+=unpack('!I',chunks[i][1])[0]+12


decompressed = zlib.decompress(chunks[1][3])
# Decompressed data length = height x (width * 3 + 1)
print "Data length in PNG file : ", len(chunks[1][3])
print "Decompressed data length: ", len(decompressed)

height = unpack('!I',(chunks[0][3][4:8]))[0]
width = unpack('!I',(chunks[0][3][:4]))[0]
blocksize = width * 3 + 1
filterbits = ''
for i in range(0,len(decompressed),blocksize):
    bit = unpack('2401c', decompressed[i:i+blocksize])[0]
    if bit == '\x00': filterbits+='0'
    elif bit == '\x01': filterbits+='1'
    else:
        print 'Bit is not 0 or 1... Default is 0 - MAGIC!'
        sys.exit(3)

s = filterbits
endianess_filterbits = [filterbits[i:i+8][::-1] for i in xrange(0, len(filterbits), 8)]

flag = ''
for x in endianess_filterbits:
    if x=='00000000': break
    flag += unhexlify('%x' % int('0b'+str(x), 2))

print 'Flag: ' + flag
```

3、LSB隐写

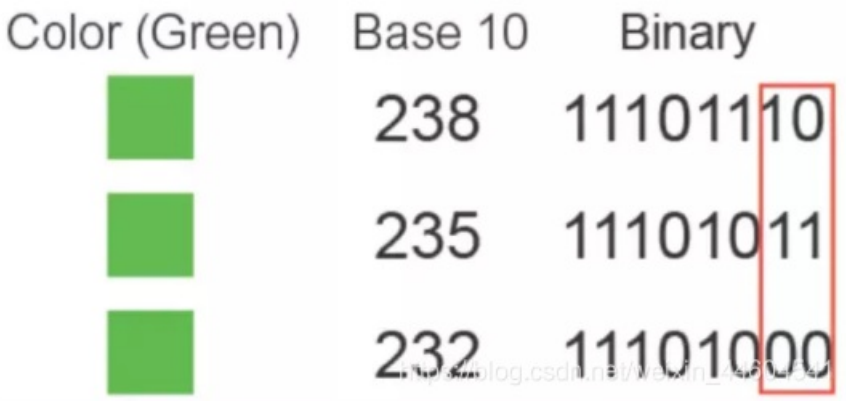
LSB，最低有效位，英文是Least Significant Bit

- 容量大、嵌入速度快、对载体图像质量影响小
- 在PNG和BMP上可以实现

原理

- 图片中的像素一般是由三种颜色组成，即三原色(红绿蓝)，由这三种原色可以组成其他各种颜色
- 在png图片的存储中,每个颜色占有8bit,即有256种颜色，一共包含256的三次方颜色，即16777216种颜色
- 人类的眼睛可以区分约1,000万种不同的颜色，剩下无法区分的颜色就有6777216
- LSB隐写就是修改了像素中的最低位，把一些信息隐藏起来

给个直观例子



这人眼看不出颜色区别，但最低位不一样

嵌入脚本

```
from PIL import Image
import math

class LSB:
 def __init__(self):
  self.im=None

 def load_bmp(self,bmp_file):
  self.im=Image.open(bmp_file)
  self.w,self.h=self.im.size
  self.available_info_len=self.w*self.h # 不是绝对可靠的
  print ("Load>> 可嵌入",self.available_info_len,"bits的信息")

 def write(self,info):
  """先嵌入信息的长度，然后嵌入信息"""
  info=self._set_info_len(info)
  info_len=len(info)
  info_index=0
  im_index=0
  while True:
   if info_index>=info_len:
    break
   data=info[info_index]
   x,y=self._get_xy(im_index)
   self._write(x,y,data)
   info_index+=1
   im_index+=1

 def save(self,filename):
  self.im.save(filename)

 def read(self):
  """先读出信息的长度，然后读出信息"""
  _len,im_index=self._get_info_len()
```

```
 info=[]
 for i in range(im_index,im_index+_len):
  x,y=self._get_xy(i)
  data=self._read(x,y)
  info.append(data)
 return info

#=============================================================#
def _get_xy(self,l):
 return l%self.w,int(l/self.w)

def _set_info_len(self,info):
 l=int(math.log(self.available_info_len,2))+1
 info_len=[0]*l
 _len=len(info)
 info_len[-len(bin(_len))+2:]=[int(i) for i in bin(_len)[2:]]
 return info_len+info

def _get_info_len(self):
 l=int(math.log(self.w*self.h,2))+1
 len_list=[]
 for i in range(l):
  x,y=self._get_xy(i)
  _d=self._read(x,y)
  len_list.append(str(_d))
 _len=''.join(len_list)
 _len=int(_len,2)
 return _len,l

def _write(self,x,y,data):
 origin=self.im.getpixel((x,y))
 lower_bit=origin%2
 if lower_bit==data:
  pass
 elif (lower_bit,data) == (0,1):
  self.im.putpixel((x,y),origin+1)
 elif (lower_bit,data) == (1,0):
  self.im.putpixel((x,y),origin-1)

def _read(self,x,y):
 data=self.im.getpixel((x,y))
 return data%2

if __name__=="__main__":
 lsb=LSB()
 # 写
 lsb.load_bmp('test.bmp')
 info1=[0,1,0,1,1,0,1,0]
 lsb.write(info1)
 lsb.save('lsb.bmp')
 # 读
 lsb.load_bmp('lsb.bmp')
 info2=lsb.read()
 print (info2)
```

识别方法

- stegsolve，调通道
- zsteg，神一样的工具

提取脚本

```
from PIL import Image

im = Image.open("extracted.bmp")
pix = im.load()
width, height = im.size

extracted_bits = []
for y in range(height):
    for x in range(width):
        r, g, b = pix[(x,y)]
        extracted_bits.append(r & 1)
        extracted_bits.append(g & 1)
        extracted_bits.append(b & 1)

extracted_byte_bits = [extracted_bits[i:i+8] for i in range(0, len(extracted_bits), 8)]
with open("extracted2.bmp", "wb") as out:
    for byte_bits in extracted_byte_bits:
                byte_str = ''.join(str(x) for x in byte_bits)
        byte = chr(int(byte_str, 2))
        out.write(byte)
```

实例

- pure_color
- 攻防世界 Misc高手进阶区 2分题 stage1
- 攻防世界 Misc高手进阶区 2分题 打野
- 攻防世界 Misc高手进阶区 2分题 倒立屋
- 攻防世界 Misc高手进阶区 2分题 Erik-Baleog-and-Olaf
- 攻防世界 Misc高手进阶区 3分题 flag_universe
- 攻防世界 Misc高手进阶区 3分题 Excaliflag

4、基于DCT域的JPG图片隐写

JPEG图像格式使用离散余弦变换（Discrete Cosine Transform，DCT）函数来压缩图像

- 通过识别每个8×8像素块中相邻像素中的重复像素来减少显示图像所需的位数
- 使用近似估算法降低其冗余度
- 有损压缩（Loss Compression）技术
- 常见的隐写方法有JSteg、JPHide、Outguess、F5

Jsteg隐写

- 将秘密信息嵌入在量化后的DCT系数的LSB上
- 原始值为-1,0，+1的DCT系数除外
- 量化后的DCT系数中有负数

实现

```
import math
import cv2
import numpy as np

def dct(m):
    m = np.float32(m)/255.0
    return cv2.dct(m)*255

class Jsteg:
 def __init__(self):
  self.sequence_after_dct=None

 def set_sequence_after_dct(self,sequence_after_dct):
  self.sequence_after_dct=sequence_after_dct
  self.available_info_len=len([i for i in self.sequence_after_dct if i not in (-1,1,0)]) # 不是绝对可靠的
  print ("Load>> 可嵌入",self.available_info_len,'bits')

 def get_sequence_after_dct(self):
  return self.sequence_after_dct

 def write(self,info):
  """先嵌入信息的长度，然后嵌入信息"""
  info=self._set_info_len(info)
  info_len=len(info)
  info_index=0
  im_index=0
  while True:
   if info_index>=info_len:
    break
   data=info[info_index]
   if self._write(im_index,data):
    info_index+=1
   im_index+=1


 def read(self):
  """先读出信息的长度，然后读出信息"""
  _len,sequence_index=self._get_info_len()
  info=[]
  info_index=0

  while True:
   if info_index>=_len:
    break
   data=self._read(sequence_index)
   if data!=None:
    info.append(data)
    info_index+=1
   sequence_index+=1

  return info

 #=============================================================#

 def _set_info_len(self,info):
  l=int(math.log(self.available_info_len,2))+1
  info_len=[0]*l
  len=len(info)
```

```
    _len=len(info)
    info_len[-len(bin(_len))+2:]=[int(i) for i in bin(_len)[2:]]
    return info_len+info

  def _get_info_len(self):
    l=int(math.log(self.available_info_len,2))+1
    len_list=[]
    _l_index=0
    _seq_index=0
    while True:
      if _l_index>=l:
        break
      _d=self._read(_seq_index)
      if _d!=None:
        len_list.append(str(_d))
        _l_index+=1
      _seq_index+=1
    _len=''.join(len_list)
    _len=int(_len,2)
    return _len,_seq_index

  def _write(self,index,data):
    origin=self.sequence_after_dct[index]
    if origin in (-1,1,0):
      return False

    lower_bit=origin%2
    if lower_bit==data:
      pass
    elif origin>0:
      if (lower_bit,data) == (0,1):
        self.sequence_after_dct[index]=origin+1
      elif (lower_bit,data) == (1,0):
        self.sequence_after_dct[index]=origin-1
    elif origin<0:
      if (lower_bit,data) == (0,1):
        self.sequence_after_dct[index]=origin-1
      elif (lower_bit,data) == (1,0):
        self.sequence_after_dct[index]=origin+1

    return True

  def _read(self,index):
    if self.sequence_after_dct[index] not in (-1,1,0):
      return self.sequence_after_dct[index]%2
    else:
      return None

if __name__=="__main__":
  jsteg=Jsteg()
  # 写
  sequence_after_dct=[-1,0,1]*100+[i for i in range(-7,500)]
  jsteg.set_sequence_after_dct(sequence_after_dct)
  info1=[0,1,0,1,1,0,1,0]
  jsteg.write(info1)
  sequence_after_dct2=jsteg.get_sequence_after_dct()
  # 读
  jsteg.set_sequence_after_dct(sequence_after_dct2)
  info2=jsteg.read()
  print (info2)
```

Outgusee算法

- 针对Jsteg算法的缺陷提出的一种方法
- 嵌入过程不修改ECT系数值为0，1的DCT系数
- 利用为随机数发生器产生间隔以决定下一个要嵌入的DCT系数的位置
- 纠正过程消除对效应的出现

识别方法

- Stegdetect：检测到通过JSteg、JPHide、OutGuess、Invisible Secrets、F5、appendX和Camouflage等这些隐写工具隐藏的信息
- JPHS：针对JPHide
- Outguess：针对OutGuess

5、数字水印隐写

数字水印（digital watermark）

- 在数字化的数据内容中嵌入不明显的记号
- 被嵌入的记号通常是不可见或不可察的
- 可以通过计算操作检测或者提取

盲水印

- 对图像进行傅里叶变换，起始是一个二维离散傅里叶变换，图像的频率是指图像灰度变换的强烈程度
- 将二维图像由空间域变为频域后，图像上的每个点的值都变成了复数，也就是所谓的复频域，通过复数的实部和虚部，可以计算出幅值和相位，计算幅值即对复数取模值，将取模值后的矩阵显示出来，即为其频谱图
- 对模值再取对数，在在0~255的范围内进行归一化，这样才能够准确的反映到图像上，发现数据之间的差别，区分高频和低频分量

识别方法

- bwm

实例

- 攻防世界 Misc高手进阶区 3分题 warmup
- 攻防世界 Misc高手进阶区 3分题 信号不好先挂了

6、图片容差隐写

容差

- 在选取颜色时所设置的选取范围
- 容差越大，选取的范围也越大
- 其数值是在0-255之间

容差比较的隐写

若是有两张图片，则对两张图片的每一个像素点进行对比，设置一个容差的阈值α，超出这个阈值的像素点RGB值设置为(255,255,255),若是没超过阈值，则设置该像素点的RGB值为(0,0,0)。因此，通过调整不同的α值，可以使对比生成的图片呈现不同的画面。比如两张图完全一样，设置阈值α为任何值，最后得到的对比图都只会是全黑。若两张图每一个像素点都不同，阈值α设置为1，则对比图将是全白。如果将隐藏信息附加到某些像素点上，这时调整阈值α即可看到隐藏信息。

如果是一张图片，则根据每一像素点周围像素的值进行判断，同样设置一个阈值，若当前像素点超过周围像素点的均值，或者其它的某种规则，则将该像素点RGB值置为(255,255,255)，反之则不进行处理，或者设置为全0.这样也可以获得隐藏的信息。

识别方法

- beyond compare比较图片

7、打乱进制

比如把整个二进制都逆序
得到一堆乱码

识别方法

- winhex，看文件头尾是不是互换且逆序

实例

- 攻防世界 Misc高手进阶区 2分题 Reverse-it

8、GIF的组合

gif每帧是某个图的一部分
提取每帧再拼接

工具

- ps

实例

- 攻防世界 Misc高手进阶区 2分题 glance-50
- 攻防世界 Misc高手进阶区 3分题 双色块

三、音频隐写

简单提一下

- 频谱图藏信息
- 高低位二进制
- 波形藏摩斯密码
- MP3Stego
- 音频中也有LSB

本来想自己整理下
看到国光大佬的很全面
就直接放个链接

CTF中音频隐写的一些整理总结

实例

- 攻防世界 Misc高手进阶区 2分题 Hear-with-your-Eyes
- 攻防世界 Misc高手进阶区 3分题 很普通的Disco
- 攻防世界 Misc高手进阶区 4分题 intoU

四、电子文档隐写

1、隐藏文字

看图说话

2、文件隐藏

类似图片隐藏文件
直接看例子吧
攻防世界 Misc高手进阶区 3分题 小小的PDF

结语

对常见文件文件头和图片音频文档隐写术做了个总结