

DEFCON 20 CTF 磁盘取证分析题目

原创

合天网安实验室 于 2021-04-08 17:29:46 发布 726 收藏 1

分类专栏: [蚁景网安学院](#) 文章标签: [unctf](#)

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/qq_38154820/article/details/115526560

版权



[蚁景网安学院 专栏收录该内容](#)

57 篇文章 18 订阅

订阅专栏

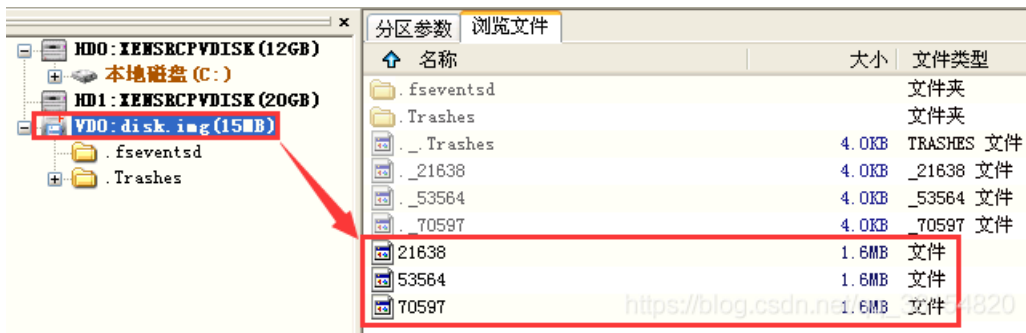
这是一道取证分析题目, 主要考察取证分析能力, 包括磁盘文件恢复、图片文件修复、数据分析、图片隐写信息提取等。

本次实验题目地址: 《[DEFCON 20 CTF Quals Forensic 200](#)》。

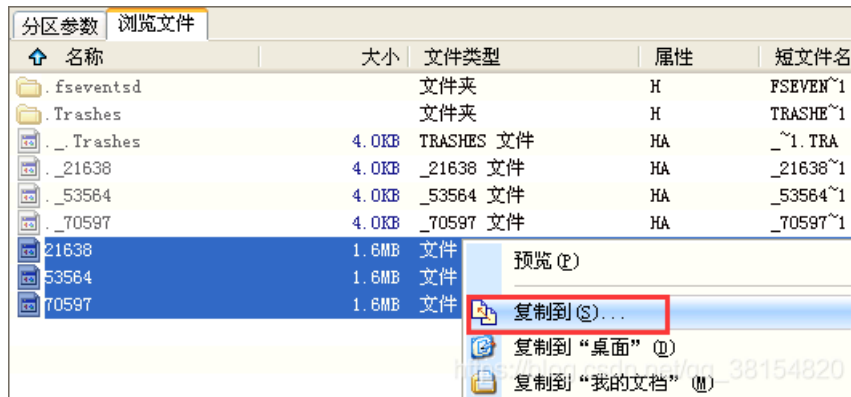
题目提供了一个disk.img文件, 我们首先可以尝试使用DiskGenius来查看其中的文件。打开DiskGenius_4.3.exe, 依次选择“硬盘”、“打开虚拟硬盘文件”菜单项, 如下图所示:



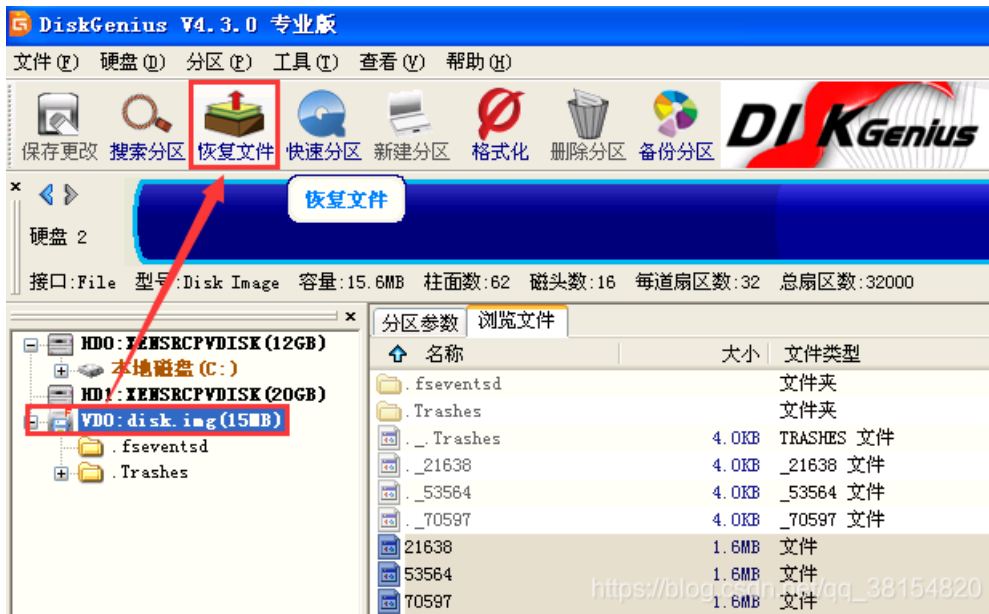
使用DiskGenius打开C:\CTF\DiskForensics\4\disk.img文件之后, 可以看到在磁盘的根目录下存在有三个1.6MB的文件, 文件名分别为21638、53564、70597, 如下图所示:



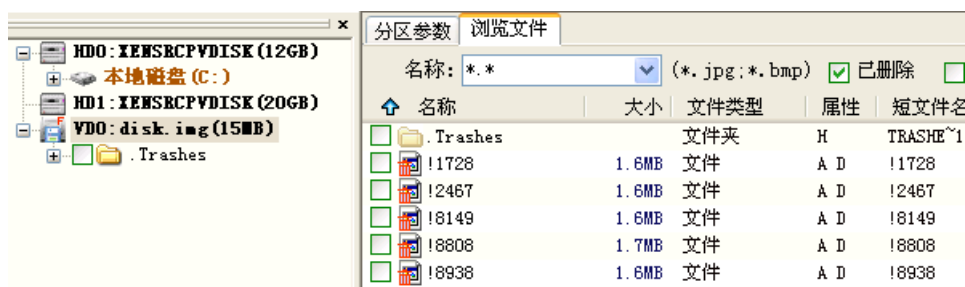
我们选中这三个1.6MB的文件后单击右键菜单，在弹出的菜单中选择“复制到(S)...”，将其复制到C:\CTF\DiskForensics\4\Files目录下，如下图所示：



既然是取证那我们来检查一下从磁盘镜像中是否可以恢复出已删除的文件。在DiskGenius主界面左侧的树形控件中选中VD0:disk.img(15MB)之后，点击工具栏上的“恢复文件”按钮，如下图所示：

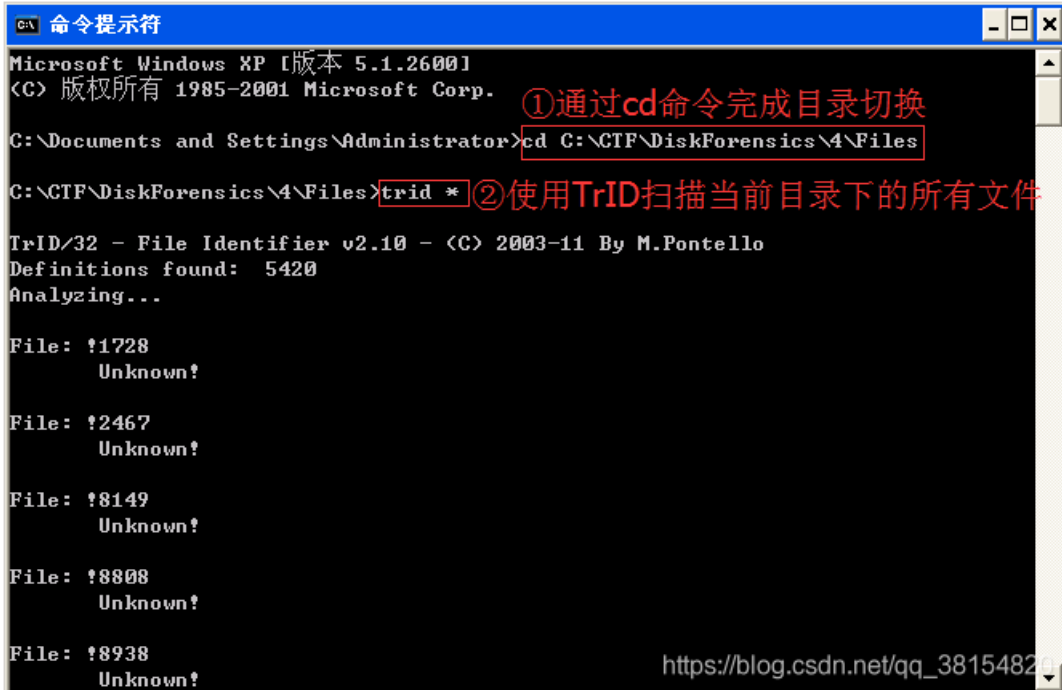


等待操作完成之后，我们可以看到一共恢复出了5个新文件，其中有4个1.6MB的文件以及1个1.7MB的文件，如下图所示：



同样选中这5个文件之后单击鼠标右键，在弹出的右键菜单中选择“复制到(S)...”，将这5个文件恢复到C:\CTF\DiskForensics\4\Files目录下。

现在我们已经从disk.img文件中提取出了八个文件，但是这八个文件都没有扩展名，所以我们可以考虑使用TrID工具来识别一下。打开CMD命令提示符，切换到C:\CTF\DiskForensics\4\Files目录，输入trid *即可扫描该目录下的所有文件，但是很遗憾的是TrID并没有识别出任何一个文件的类型，如下图所示：



```
Microsoft Windows XP [版本 5.1.2600]
(C) 版权所有 1985-2001 Microsoft Corp.
C:\Documents and Settings\Administrator>cd C:\CTF\DiskForensics\4\Files
C:\CTF\DiskForensics\4\Files>trid *
TrID/32 - File Identifier v2.10 - (C) 2003-11 By M.Pontello
Definitions found: 5420
Analyzing...

File: !1728
      Unknown!

File: !2467
      Unknown!

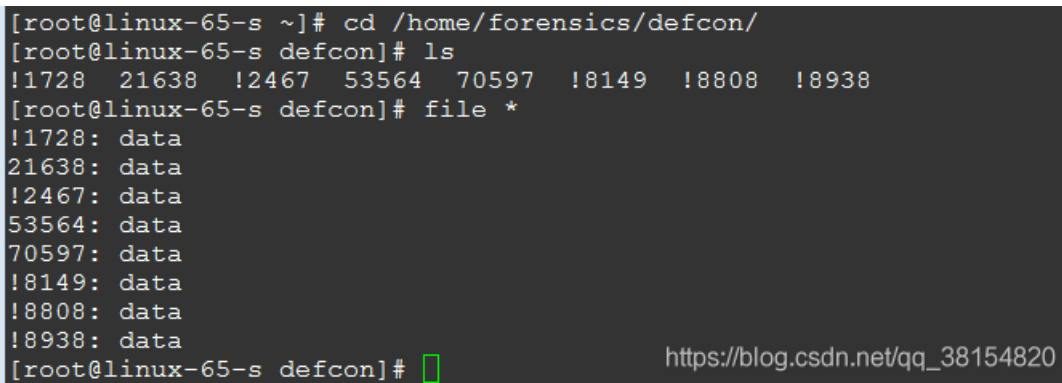
File: !8149
      Unknown!

File: !8808
      Unknown!

File: !8938
      Unknown!
```

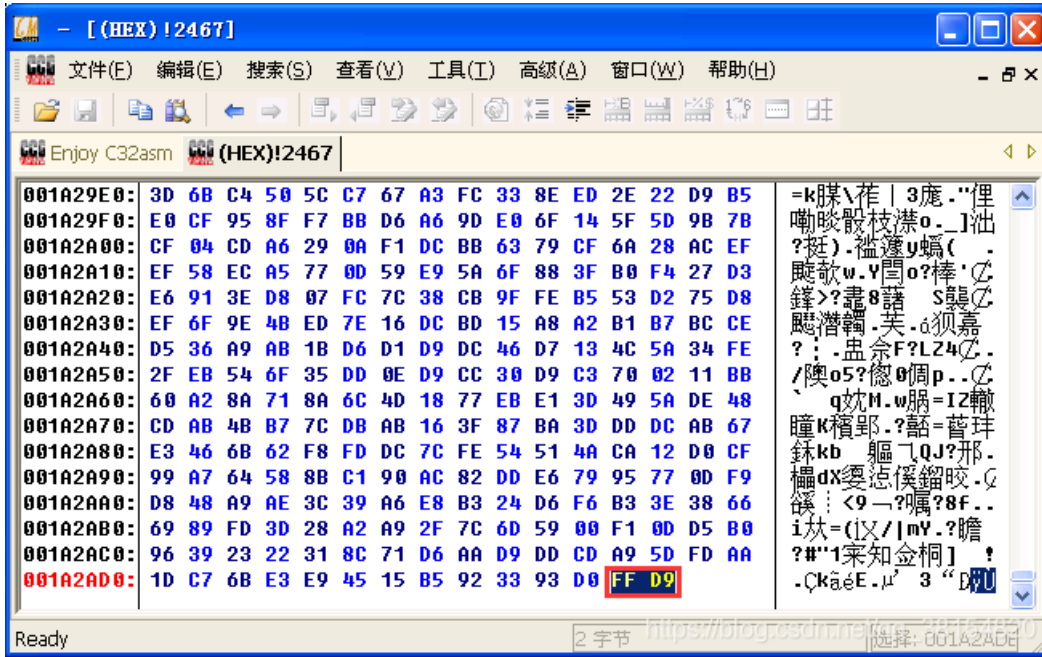
识别不出呢，咋办呢。

别慌我们还有linux的file命令，我们已经把提取出来的八个文件放到Linux实验机器的/home/forensics/defcon目录下了。现在切换到Linux实验主机，使用cd命令切换到/home/forensics/defcon目录之后，执行file *来对文件进行扫描，跟TrID一样，file命令也识别不出任何结果，如下图所示：



```
[root@linux-65-s ~]# cd /home/forensics/defcon/
[root@linux-65-s defcon]# ls
!1728 21638 !2467 53564 70597 !8149 !8808 !8938
[root@linux-65-s defcon]# file *
!1728: data
21638: data
!2467: data
53564: data
70597: data
!8149: data
!8808: data
!8938: data
[root@linux-65-s defcon]#
```

莫慌，这肯定是数据被破坏了，我们还可以手动识别 打开十六进制编辑器C32asm（位于C:\Tools\c32asm\C32asm.exe），使用C32asm打开!2467文件，可以看到文件的前面两个字节为00 00，显然文件头部字节被抹掉了，而如果来到文件末尾，可以看到最后的两个字节是FF D9，如下图所示：



3. 因为最后面两个字节是FF D9，所以有可能是一个JPG文件，因为JPG文件的头部两个字节是FF D8，而末尾两个字节是FF D9，所以我们可以把最前面的两个字节填充为FF D8，然后按下Ctrl+S保存对文件的修改；

4. 给文件!2467添加.jpg扩展名，打开发现可以正常显示，说明这就是一个JPG文件；

5. 经过同样的操作，我们可以发现!8808、!8938、21638、53564、70597这五个文件也是JPG文件，而!1728、!8149则无法直接看出是什么文件；

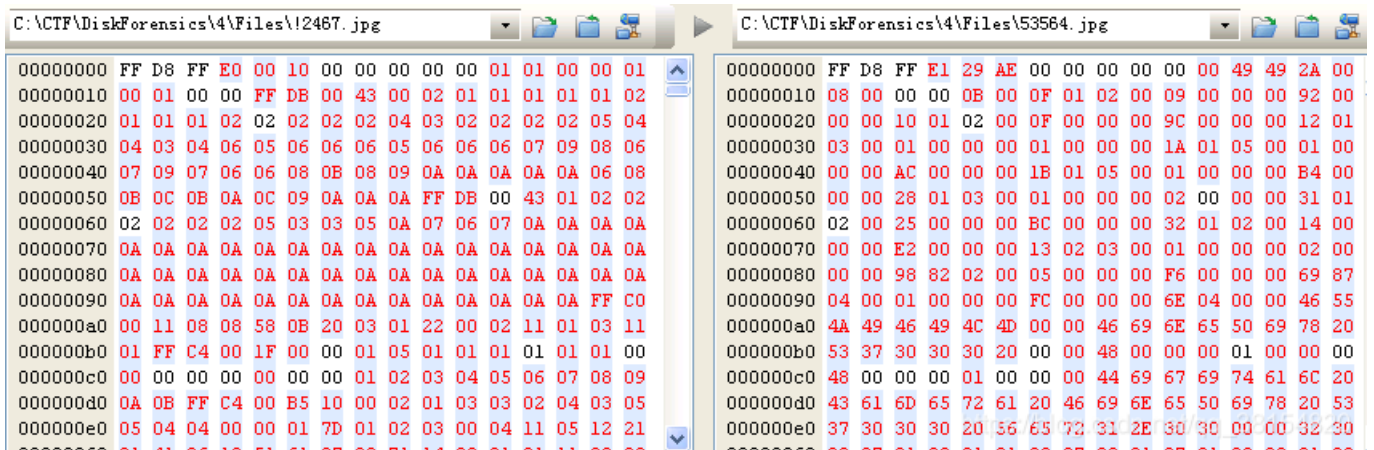
这里我们发挥一下想象，这六个图片文件中有两个文件显示的图像是一样的，经过对比发现两个文件的大小不一样，其中前者为1.63 MB (1,714,910 字节)，后者为1.59 MB (1,670,111 字节)。

此事必有蹊跷，对比一下两个文件看下

1. 打开UltraCompare（位于C:\Tools\UltraCompare\uc.exe）；

2. 依次点击“模式”、“二进制（快速）模式”菜单项；

3. 单击文件夹图标选中两个要比较的文件，单击绿色箭头图标开始比较，如下图所示；



4. 比较之后可以发现，两个文件的二进制数据存在大量差异之处，如下图所示；

```
[root@linux-65-s ~]# cd /home/forensics/defcon2/
[root@linux-65-s defcon2]# ls
!2467.jpg 53564.jpg
[root@linux-65-s defcon2]# stegdetect \!2467.jpg
!2467.jpg : negative
[root@linux-65-s defcon2]# stegdetect 53564.jpg
53564.jpg : negative
[root@linux-65-s defcon2]#
```

经过上面的分析，发现两个图片文件大部分的二进制内容是不一样的，可以知道这里不是简单的在图片末尾附加数据。

别慌我可以使用stegdetect工具来检查一下。现在切换到Linux实验机器来进行操作，具体的实验步骤如下：

\1. 通过cd命令切换到/home/forensics/defcon2目录，我们已经把上面的两个JPG文件复制到该目录下了；

\2. 使用stegdetect检测两个图片文件，发现都提示negative，即并没有检测出隐写信息，如下图所示：

```
[root@linux-65-s defcon2]# stegdetect -s 2.0 \!2467.jpg
!2467.jpg : outguess(oid) (*)
[root@linux-65-s defcon2]# stegdetect -s 2.0 53564.jpg
53564.jpg : negative
```

\3. 调整stegdetect的敏感度（通过-s参数指定），设定敏感度为2.0，再次检测两个文件，发现文件!2467.jpg存在outguess隐写信息，如下图所示：

```
[root@linux-65-s defcon2]# outguess -r \!2467.jpg data.txt
Reading !2467.jpg...
Extracting usable bits: 1793659 bits
Steg retrieve: seed: 8113, len: 24297
[root@linux-65-s defcon2]# file data.txt
data.txt: data
[root@linux-65-s defcon2]# hexdump -C -n 64 data.txt
00000000 5e de d0 57 a6 31 c0 39 d0 8b 5d f0 a7 cd 80 84 |^..W.1.9..].....|
00000010 0c bc 02 ec 75 dc 21 ca 2c b2 c1 31 53 10 b3 42 |...u.!.,.1S..B|
00000020 39 66 6c e9 0d aa ff 32 78 1f 26 b7 99 00 a6 ce |9fl....2x.&.....|
00000030 61 e4 85 fb bc 37 18 d5 f4 57 58 1b 9d 88 8e db |a....7...WX.....|
```

到现在为止，我们基本推测出了文件!2467使用了outguess来隐藏了隐写信息，现在我们可以使用outguess来提取其中的隐写信息，在Linux中执行outguess -r !2467.jpg data.txt即可，如下图所示：

```
[root@linux-65-s defcon2]# outguess -r \!2467.jpg -k "ddtek" data.txt -e
Initialize encoding/decoding tables
Reading !2467.jpg...
Extracting usable bits: 1793659 bits
Decode: 12 data after ECC: 4
Steg retrieve: seed: 297, len: 55614
Decode: 55614 data after ECC: 29013
[root@linux-65-s defcon2]# file data.txt
data.txt: Zip archive data, at least v2.0 to extract
[root@linux-65-s defcon2]#
```

看来是的很有可能是outguess提取隐写信息的时候需要指定一个密码，这时候可以编写一个脚本来破解这个密码，由于不知道密码的构词规则，所以可以使用暴力破解或者是字典破解的方法（可以暴力破解5个字母的密码，或者使用字典进行破解）。

最终破解出的密码是ddtek（曾经组织过DEFCON CTF的一个队伍名称），同时在使用outguess提取隐写信息的时候还要指定-e参数，表示需要使用错误纠正编码，完整的命令为outguess -r !2467.jpg -k "ddtek" data.txt -e。待提取完毕后，执行file data.txt可以知道这是一个ZIP文件，如下图所示：



这个在linux服务器上面呢，服务器的IP地址为10.1.1.47，我们在这里执行nohup python -m SimpleHTTPServer 8888 &即可在服务器上监听8888端口，在XP下的Firefox浏览器中下载<http://10.1.1.47:8888/data.txt>即可，下载之后将其重命名为data.zip并解压出其中的文件，打开解压出来的PDF文件即可看到Flag。