

CTF—图片隐写+数据隐写

原创

小常吃不了了 于 2021-07-28 17:27:29 发布 549 收藏 1

分类专栏: [CTF](#) 文章标签: [安全 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/weixin_52620919/article/details/119184050

版权



[CTF 专栏收录该内容](#)

32 篇文章 1 订阅

订阅专栏

一、【图片隐写】

题目描述:

在实验主机上的C:\Stegano\3目录下提供了一个名为stego的文件, 找到一个形式为flag{word_word_word}的字符串。



实验任务



stego
文件
177 KB

预备知识

【TrID】

TrID是一款根据文件二进制数据特征进行判断的文件类型识别工具。

虽然也有类似的文件类型识别工具, 但是大多数都是使用硬编码的识别规则, 而TrID则没有固定的匹配规则,

TrID具有灵活的可扩展性, 可以通过训练来进行文件类型的快速识别。

TrID通过附加的文件类型指纹数据库来进行匹配, 可用于 取证分析、未知文件 识别等用途。

【BinWalk】

BinWalk是一个固件的分析工具, 旨在协助研究人员对固件进行分析, 提取及逆向工程用处。

简单易用, 完全自动化脚本, 并通过自定义签名, 提取规则和插件模块,

还有重要一点的是可以轻松地扩展。最简单的使用方法很直接, 提供文件路径和文件名即可。

【StegHide】

Steghide是一个隐写程序, 其可以将数据隐藏在各种图片文件以及音频文件之中。

Steghide可以对隐写的数据进行加密和压缩操作。

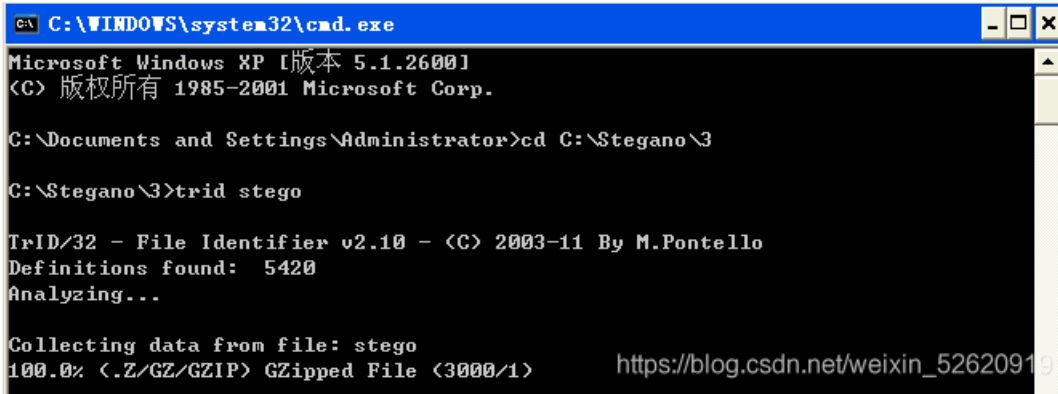
考察意图:

考察选手的文件隐写取证分析能力，包括对工具TrID、BinWalk、StegHide等的了解。

分析文件类型

题目提供的文件没有文件扩展名，因此我们需要先确定文件的真实文件类型，使用【TrID】工具可以对未知文件类型进行有效的识别。

1、打开cmd命令提示符，切换到相关路径下，然后使用TrID对文件进行识别，如图所示：



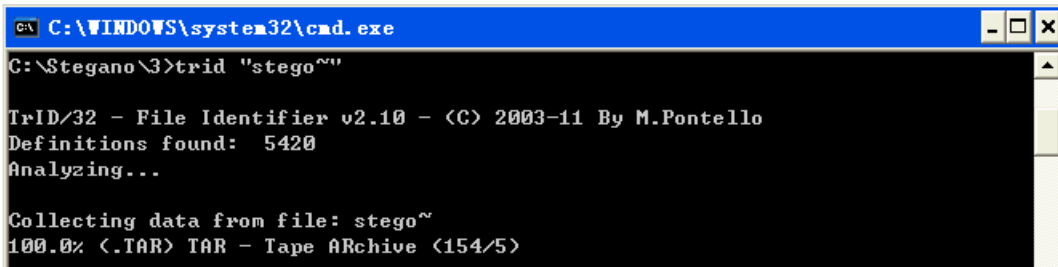
```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [版本 5.1.2600]
(C) 版权所有 1985-2001 Microsoft Corp.

C:\Documents and Settings\Administrator>cd C:\Stegano\3
C:\Stegano\3>trid stego

TrID/32 - File Identifier v2.10 - (C) 2003-11 By M.Pontello
Definitions found: 5420
Analyzing...

Collecting data from file: stego
100.0% (.Z/GZ/GZIP) GZipped File (3000/1) https://blog.csdn.net/weixin_52620919
```

根据TrID的识别结果，这是一个GZip压缩文件，我们使用7Zip对其进行解压得到另一个没有扩展名的文件，再次使用TrID对其进行识别，提示为TAR压缩文件，如图所示：



```
C:\WINDOWS\system32\cmd.exe
C:\Stegano\3>trid "stego~"

TrID/32 - File Identifier v2.10 - (C) 2003-11 By M.Pontello
Definitions found: 5420
Analyzing...

Collecting data from file: stego~
100.0% (.TAR) TAR - Tape ARchive (154/5)
```

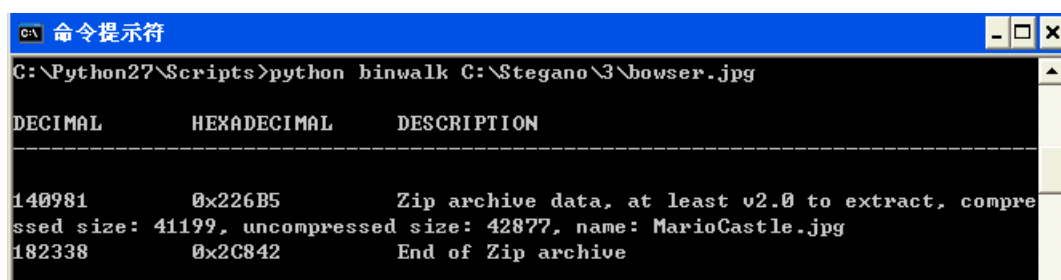
同样使用7Zip对其进行解压，得到 browser.jpg 文件，是一张图片。



内嵌文件数据分析

在CTF竞赛中，很多情况下会把一个文件的二进制数据嵌入到另一个文件之中，对于[嵌入ZIP]之类的数据，我们可以尝试直接将文件名的后缀改为zip，然后使用WinRAR之类的解压缩工具打开即可。但是对于未知的文件类型，这样就显得力不从心了，这里介绍一种通用的文件识别方法，就是使用BinWalk工具。

打开cmd命令提示符，切换到相关路径下。执行python binwalk命令来对browser.jpg文件进行处理，如图所示：

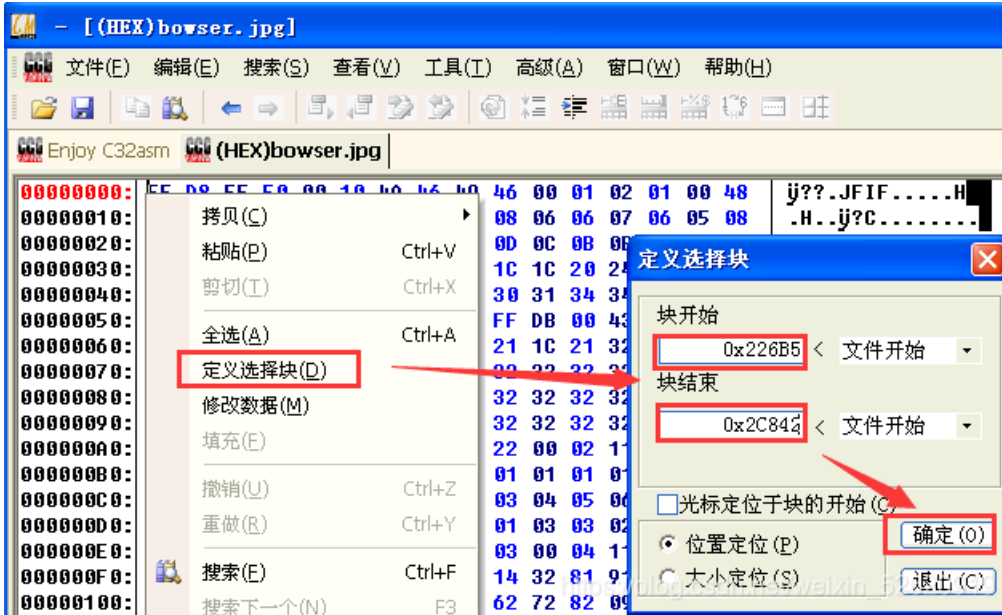


```
C:\Python27\Scripts>python binwalk C:\Stegano\3\browser.jpg

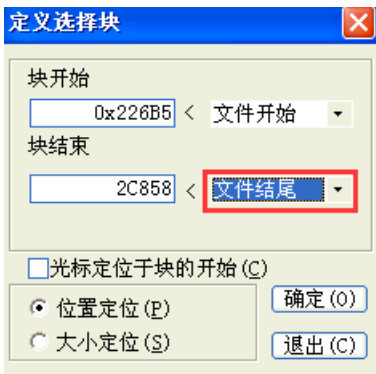
DECIMAL      HEXADECIMAL    DESCRIPTION
-----
140981       0x226B5        Zip archive data, at least v2.0 to extract, compressed size: 41199, uncompressed size: 42877, name: MarioCastle.jpg
182338       0x2C842        End of Zip archive
```

从BinWalk的分析结果可以看出，JPG文件中有一个Zip压缩包，其文件偏移地址范围是0x226B5~0x2C842，这里我们使用C32Asm将Zip数据提取出来。

打开桌面上的【C32Asm工具】，选择“文件”、“打开十六进制文件”载入C:\Stegano\3\browser.jpg文件，然后右键选择“定义选择块”填入数据块的起始地址为 0x226B5，结束地址为 0x2C842，单击确定就选中数据块了
右键复制数据，然后新建一个十六进制文件，将原有的数据替换为复制的数据
保存即可得到压缩包文件。操作过程如图所示：



但是实际测试时发现压缩包不能正常解压，这是因为BinWalk对ZIP的识别机制还不完善，漏掉了ZIP文件末尾的一段数据，所以我们需要重新复制数据，文件的开始位置仍然是0x226B5，结束位置直到文件的末尾，如图所示：



这样提取出来的压缩包就可以正常解压了，得到图片文件MarioCastle.jpg



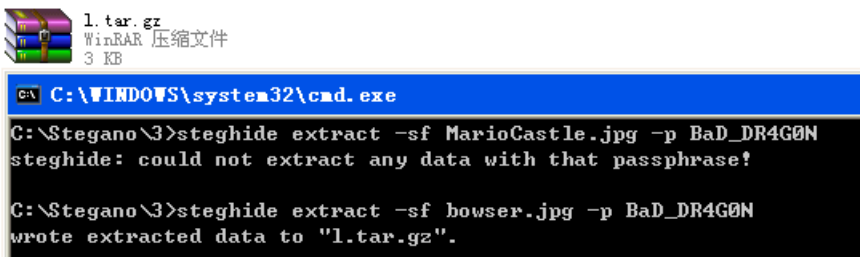
提取隐写数据

从实验步骤二中得到了MarioCastle.jpg文件，这个图片里面显示有一个Passphrase字段，值为BaD_DR4G0N，看到 [Passphrase]我们就可以猜测使用了【steghide】进行数据隐写。因此我们尝试使用steghide对MarioCastle.jpg进行隐写数据提取



打开cmd命令提示符，输入steghide extract -sf MarioCastle.jpg -p BaD_DR4G0N命令，提示没有任何可以提取的数据。

那么如果尝试对原来的 **bowser.jpg** 进行处理呢？实际测试表明**bowser.jpg**里面隐写了数据，通过**steghide**我们提取出了**1.tar.gz**文件，如图所示：



解压1.tar.gz文件，得到flaga.jpg文件，打开即可看到flag为flag{You_F0und_M3}。