Case Presentation

Info 4120
06/11/19

Ryan Binley 100220465
Rekha Sihota 100068192
Abdul Rehman 100327994
Brian Nadjiwon 100251108

Case Overview

Our case begins on November 11th, 2019. A local man is arrested for solicitation of a minor after an online messaging exchange and subsequent meetup with an undercover officer. An examination of the computer hard drives seized from his residence reveals nothing immediately. However, one file directory catches the eye of an investigator. The folder is filled with innocuous pictures that appear to have no common link to each other. Utilizing forensic technology, the investigator examines the files and finds a number of jpeg images embedded inside them. After extracting the files and reviewing the images, they are revealed to be of child pornography. A reverse image search is performed on the images the files were embedded in, leading to an obscure message forum. The images are being shared by the users of this forum, which seems to be serving as an exchange site. A subpoena is issued to the company hosting the site, which turns over the server logs. The user accounts and associated IP addresses are located all over the world, but two of them are within the jurisdiction of our investigators. The first was our initial suspect, but the second is a new lead. A warrant is issued, and the second suspect's home is searched. The investigators seize his computer, and on reviewing his hard drives they find none of the images. Using forensic technology, they examine the disk again, and discover several deleted files and directories. The files are recovered, and an md5 checksum reveals them to be the same ones that were being exchanged on the forum. Charges are brought against both suspects for possession of child pornography, and the information regarding the IP addresses outside our investigators jurisdiction is shared with their international partners. Furthermore, the exif data of the illicit images is analyzed, and locational data and camera metadata are used to rescue a number of children from abusive homes and brings charges against their abusers.

Digital Evidence

Disk one:

```
root@kali:~# genisoimage -o evildisk.iso /root/evildisk
I: -input-charset not specified, using utf-8 (detected in locale settings)
 78.94% done, estimate finish Thu Nov 21 17:26:44 2019
Total translation table size: 0
Total rockridge attributes bytes: 0
Total directory bytes: 2048
Path table size(bytes): 22
Max brk space used 0
6335 extents written (12 MB)
root@kali:~#
```

```
root@kali:~# fls -r evildisk.iso
d/d 1:  DIR1
+ r/r 2:        COVER.JPG
+ r/r 3:        FILEHEAD.JPG
d/d 4:  $OrphanFiles
Segmentation fault
root@kali:~#
```

Disk two:

```
root@kali:~# genisoimage -o evildisk2.iso /root/Evildisk2
I: -input-charset not specified, using utf-8 (detected in locale settings)
Total translation table size: 0
Total rockridge attributes bytes: 0
Total directory bytes: 8192
Path table size(bytes): 66
Max brk space used 1b000
193 extents written (0 MB)
```

```
root@kali:~# fls -r evildisk2.iso
d/d 1:  ASMITH
+ d/d 2:        DIR
++ r/r 5:       REPORT1.DOC
++ r/r 6:       REPORT2.DOC
++ r/r 7:       REPORT3.DOC
+ d/d 3:        _COVER
+ d/d 4:        _FOREST
V/V 8:  $OrphanFiles
root@kali:~#
```

Forensic Tools Used

Binwalk

Binwalk is a tool capable of searching a disk image for embedded files and code. It accomplishes this by identifying file signatures or executable opcode signatures. It also has extraction and data carving utility.

Sleuthkit

Sleuthkit is a suite of tools for analyzing disk images and recovering deleted files. It allows us to list allocated and unallocated file names and their associated inodes, and to extract the file to standard output.

Steghide

Steghide is a steganography utility that allows for the embedding of a file within a cover file. Any type of file may be embedded, and a number of file types are compatible for use as a cover file. The program supports a number of cryptographic methods to secure the file. Steghide works by using a graph-theoretic approach, where after encrypting and compressing the data, a map of pixel locations in the cover file are generated, and parts of the encrypted data are embedded in these locations.

Testdisk

Testdisk is a file recovery utility that can recover lost partitions and boot non-booting disks damaged by viruses, software errors, user errors, or malicious actions.

Photorec

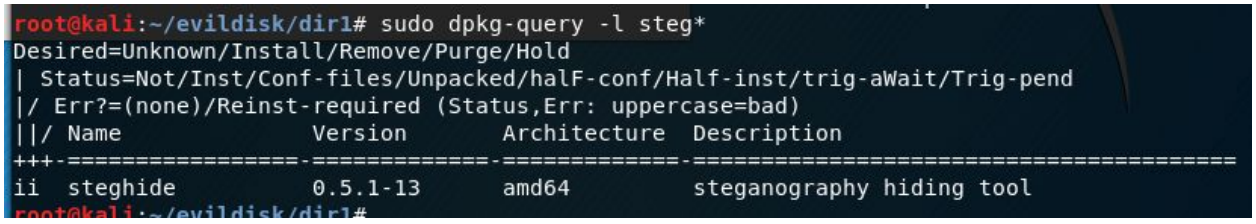
Photorec is a file recovery software that can retrieve lost files from disk images. Photorec is not concerned with the file system and instead searches for the underlying data, making retrieval of files possible even when the file system has been damaged or reformatted.

Image Discovery and Recovery

Background

Steganography is the art of concealing a message or communication. Unlike encryption, steganography seeks to hide not just the contents of a message but the exchange of the message itself. In the digital world, steganography is typically accomplished by embedding files within other files, typically by using a tool or command line trick. The file will appear to an unaware user to simply be a picture or document, but any file that is sufficiently smaller can be hidden within the slack space or appended to the file. Steganography has some legitimate uses, such as embedding watermarks or data that can guarantee ownership or authenticity of a file. It also has some less than legitimate uses, such as the files Facebook embeds in photos to track users. Not surprisingly, this technique can be abused for criminal activity. Steganography has become a particular concern for law enforcement in the area of child pornography. Let's assume a group wants to exchange illicit images. They agree beforehand on a time and place to exchange the images. They agree on a method to embed and extract a file, and a password if required. Theoretically, any site that allows for file uploads could then be used as an exchange site, and the owners of the site and any other users would be none the wiser. Sites don't check uploaded files for signs of embedding, and depending on the sophistication of the software used to embed the files, they wouldn't find anything if they did.

Steganalysis is the process of detecting steganography. It is a full spectrum approach that can include anything from analyzing images for artifacts to using neural networks to identify signs at a bit by bit level. Generally, we would start by searching for indications of steganographic programs on the suspect machine. This could include checking for installed programs or checking command history. In the example below, we use the dpkg-query to search for installed programs matching a certain file pattern:



Recent website activity could also indicate the suspect has recently downloaded steganography software. Context can also provide clues; if the subject is suspected of possessing child pornography, and there are a large number of sufficiently large but otherwise innocuous files on his computer, this may be an indication that these are being used to embed hidden files.

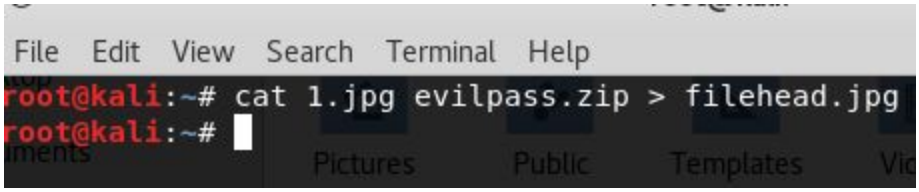
Once we have suspicion that the suspect is using steganography software or possesses steganographic files, we need to search the files for indicators of steganography. Most files have headers that indicate the type of a file and where it starts and ends. The presence of unexpected headers suggests steganography. There are a number of tools that can scan for these headers, such as Sleuthkit and Binwalk. However, if the steganography was performed using software with encryption and masking algorithms, these tools will be unable to detect these headers. For instance, here is output from Binwalk run against a jpg that had another jpg embedded using Steghide:

```
root@kali:~/evildisk/dir1# binwalk -B cover.jpg

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             JPEG image data, JFIF standard 1.01

root@kali:~/evildisk/dir1#
```

While Binwalk can successfully find something simple like a concatenated file, it has less success against files hidden using steghide. This is due to steghide using an encryption process to mask the file header and body when embedding the file.

The Crime

There are a number of methods to hide files within other files. Tools like Steghide and Stegosuite exist to facilitate the process. It can also be done manually by using the cat command in linux. In this instance, the criminals make use of both methods. The sharing of these files involves a “header” picture, that contains a password. This password is then used to unencrypt the other files. The “header” picture is created by using the cat command in linux. There are two files: 1.jpg, the cover file, and evilpass.txt, which contains the password. The criminals compress the evilpass.txt and send it to a zip file. They then concatenate this zip file with the cover file using the following command:



This creates a single filehead.jpg from the two files. The illicit materials are hidden using Steghide. Steghide supports a number of different formats and file structures. The main requirement is that the embedded file will be able to fit into the cover file.

In the example above, the criminals use -cf to indicate the cover file, and -ef to indicate the embed file. Steghide will then prompt them to enter a passphrase. Once it has a valid passphrase it will embed the file.

The Investigation

A preliminary review of the suspect's computer reveals that the suspect has recently used the program steghide:

The investigator attempts to use steghide to extract a file from the cover.jpg, but it is password protected. The investigator could try running a cracking program to brute-force the password, but that may take a while depending on the password complexity. The investigator decides to search the other files for the password. Using sleuthkit, we run fls with the recursive -r switch command to see the files in the disk and the associated inodes:

```
root@kali:~# fls -r evildisk.iso
d/d 1:  DIR1
+ r/r 2:        COVER.JPG
+ r/r 3:        FILEHEAD.JPG
d/d 4:  $OrphanFiles
Segmentation fault
root@kali:~#
```

We see two jpg files at inode 2 and 3. The investigator runs icat against inode 3 and pipes it to xxd to see if there is any information regarding embedded files:

```
root@kali:~# icat evildisk.iso 2 | xxd
00000000: ffd8 ffe0 0010 4a46 4946 0001 0100 0001  ......JFIF......
00000010: 0001 0000 ffdb 0043 0001 0101 0101 0101  .......C........
00000020: 0101 0101 0101 0101 0101 0101 0101 0101  ................
```

At the beginning of the file is the hexadecimal header “ffd8 ffe0” and the ascii equivalent JFIF, which indicates this is a jpg file. The investigator does a search through the output to see if there are any other file headers in the file:

```
root@kali:~# icat evildisk.iso 3 | xxd | grep "50 4b"
00007db0: 4201 1b71 52bb 0e30 ab33 a953 7150 4b12  B..qR..0.3.SqPK.
root@kali:~#
```

The investigator finds a file header fragment and associated ascii equivalent that seems to indicate the presence of a zip file within the jpg. It is possible that the zip file could be reassembled bit by bit using the output.

However, we have another tool that should be able to extract the zip for us. Binwalk is a program that iterates through files searching for common file headers. The investigator runs binwalk against the file:

```
root@kali:~/evildisk/dir1# binwalk -B filehead.jpg

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             JPEG image data, EXIF standard
12            0xC             TIFF image data, big-endian, offset of first image directory: 8
6413          0x190D          Unix path: /www.w3.org/1999/02/22-rdf-syntax-ns#"> <rdf:Descriptio
n rdf:about="" xmlns:xmp="http://ns.adobe.com/xap/1.0/" xmlns:xmpMM="http
147941        0x241E5         Zip archive data, at least v2.0 to extract, uncompressed size: 8,
name: evilpass
148123        0x2429B         End of Zip archive
```

The output confirms that there is a hidden zip file in the jpg. We can now use Binwalk's extraction capability to reassemble the zip file:

```
root@kali:~# binwalk -e filehead.jpg

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             JPEG image data, EXIF standard
12            0xC             TIFF image data, big-endian, offset of first i
directory: 8
6413          0x190D          Unix path: /www.w3.org/1999/02/22-rdf-syntax-n
<rdf:Description rdf:about="" xmlns:xmp="http://ns.adobe.com/xap/1.0/" xmln
pMM="http
147941        0x241E5         Zip archive data, at least v2.0 to extract, un
ressed size: 8, name: evilpass
148123        0x2429B         End of Zip archive

root@kali:~# unzip evilpass.zip
Archive:  evilpass.zip
replace evilpass? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
  inflating: evilpass
```

The investigator suspects that the text within the file may be the password for steghide. They attempt to extract the file:

```
oot@kali:~/cover# steghide extract -sf cover.jpg
Enter passphrase:
wrote extracted data to "embed.jpg".
```

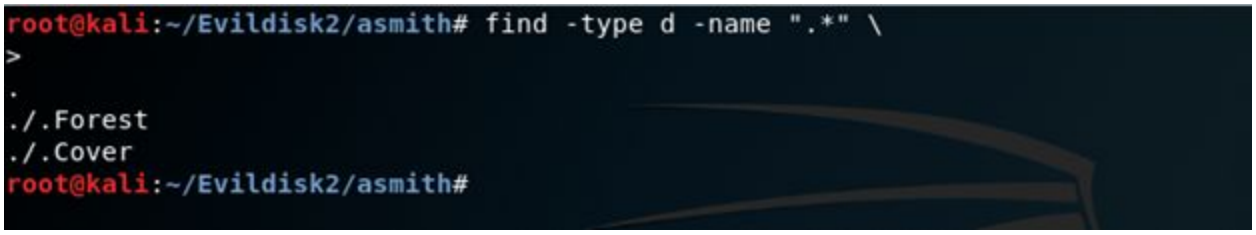
This successfully extracts the file. The investigator runs a md5 check against the file for future use:

```
root@kali:~# md5sum embed.jpg
e77e40cfce2dd8aaaed68525a2acd5cc  embed.jp
root@kali:~#
```

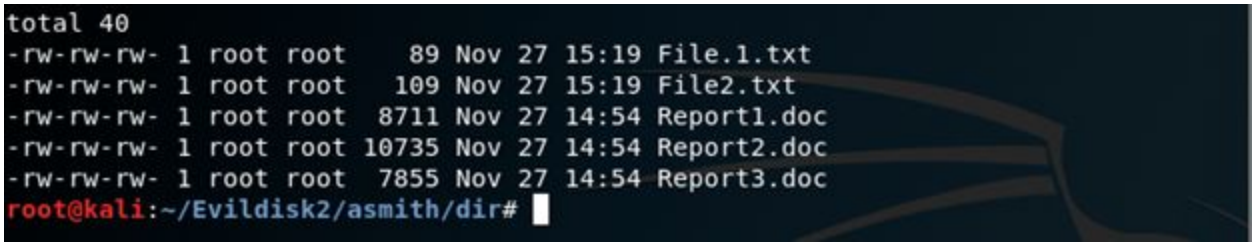
File Recovery

There are many different methods to hide data so it can't be easily found. First one being, giving the directory an erroneous name so it can potentially be overlooked on the file listing. In most cases the actual data is not disguised in any way, but instead it's trying to hide the directory itself. There are a couple of directory names that are used again and again by users who want to conceal the data. They are three dots and two dots and a space. Secondly, creating a directory in a location where it has a minimal chance of being found. Thirdly, would be to delete the file containing the data. However, users who want to hide large amounts of data might not use this method, as there may be potential difficulties in recovering all the data in one piece. Lastly, changing the file names, so they blend in amongst all the other files and appear to be valid. For example, a user trying to download pornography image files under there account by giving the files innocuous names and changing the file extension to something like ".docx".

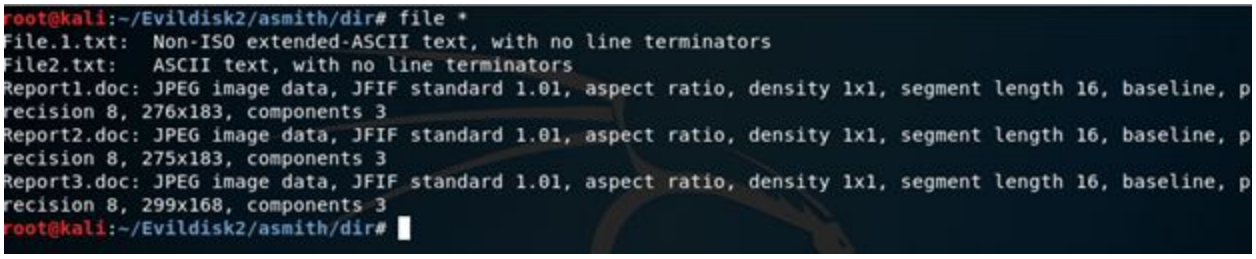
We were able to see the two hidden directory under the "asmith" folder. Directories can be hidden by putting "." (one dot) before the file name. They cannot be viewed using the ls command.



The word files below that all start with the word report appear to be strange. These files seem to be obscured amongst the other files.

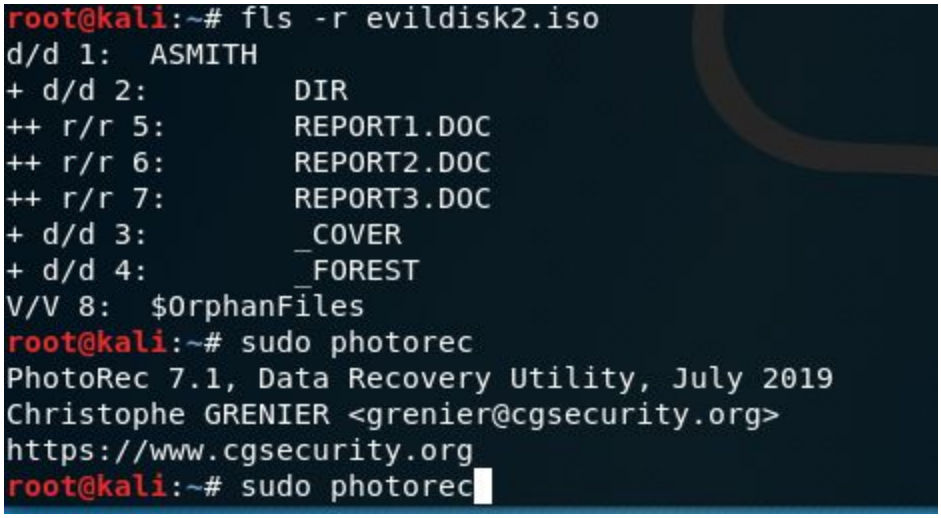


It appears the file extensions was changed on these files. As shown below, they are actually .jpeg files. We were able to get the evidence we needed from these files.

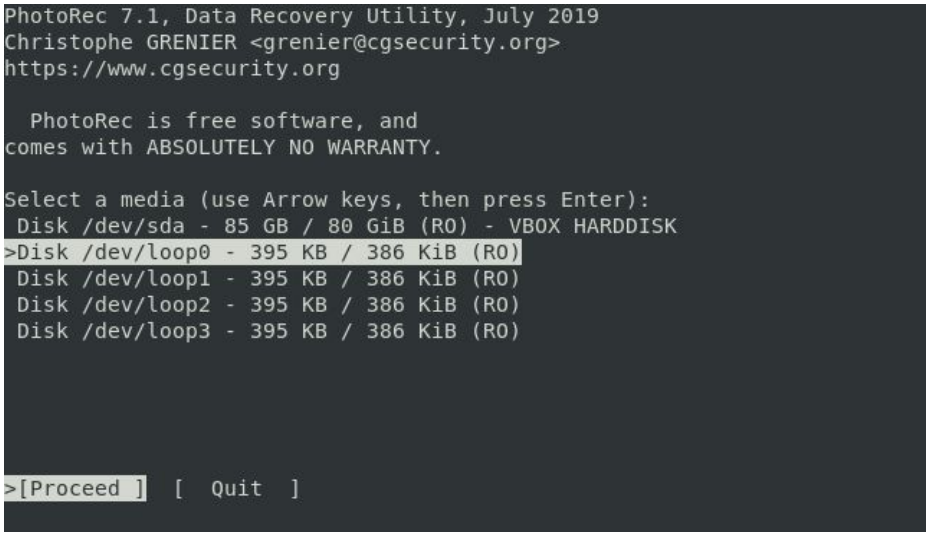


Picture file recovery using Photorec.

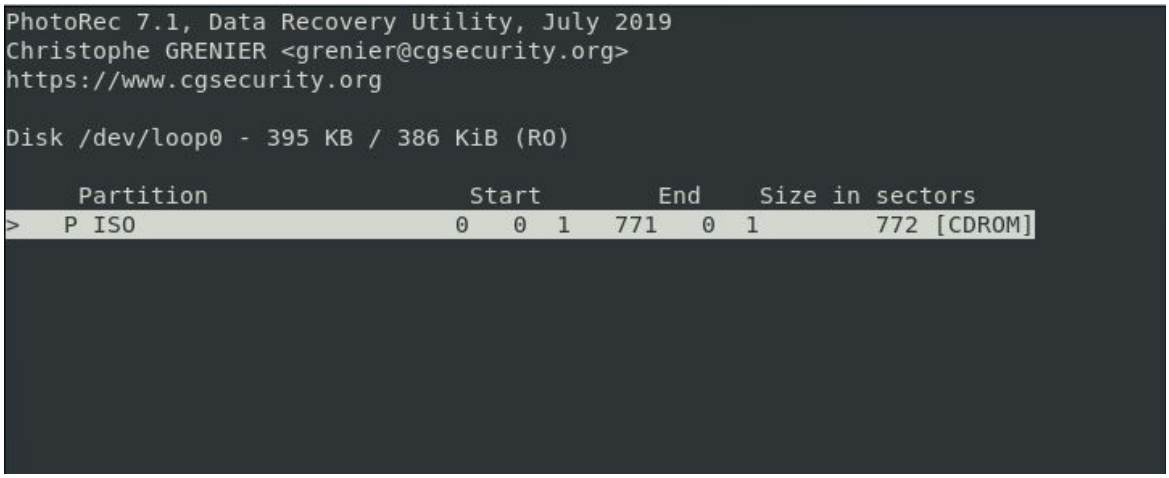
The Photorec tool is used to recover deleted picture files. This tool is only used for picture files.



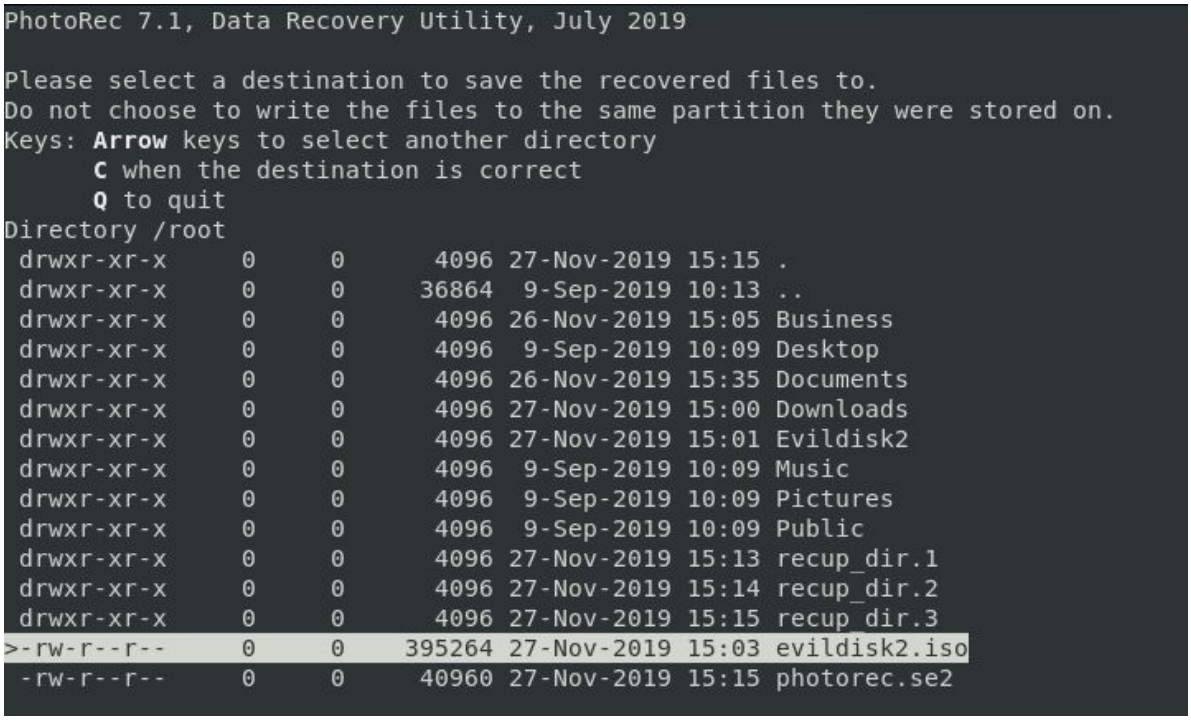
I loaded the iso file on my CD-ROM disk drive and selected the disk here.



This shows the details of the disk CD-ROM drive.



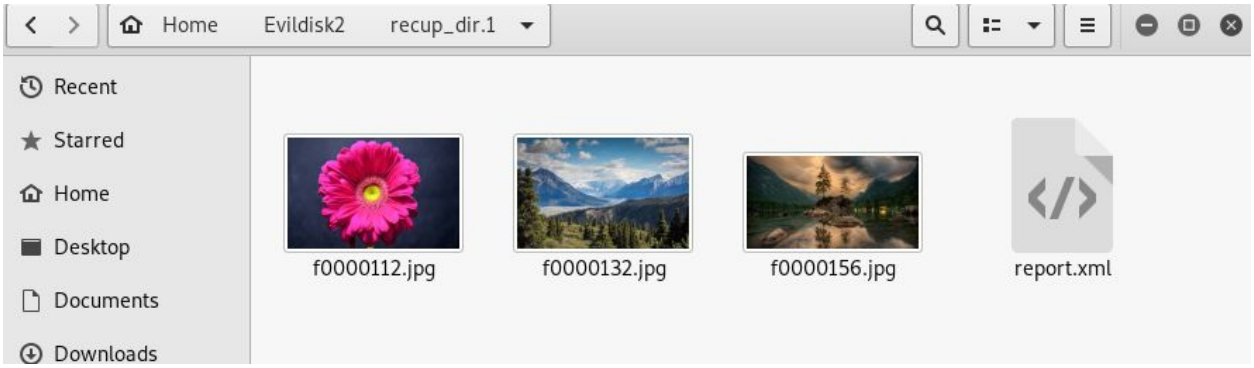
I selected the evildisk2 directory to save the recovered files to.



We were able to recover the three picture files using this tool. The recovered files are written in recup_dir directory as per the location I selected.

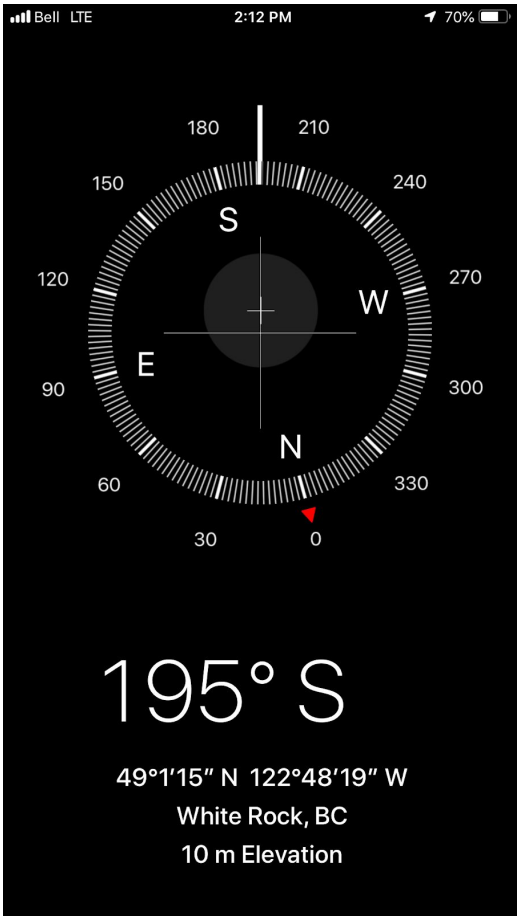
```
PhotoRec 7.1, Data Recovery Utility, July 2019
Christophe GRENIER <grenier@cgsecurity.org>
https://www.cgsecurity.org

Disk /dev/loop0 - 395 KB / 386 KiB (RO)
     Partition                  Start        End    Size in sectors
   P ISO                    0   0  1   771   0  1        772 [CDROM]


3 files saved in /root/recup_dir directory.
Recovery completed.

You are welcome to donate to support and encourage further development
https://www.cgsecurity.org/wiki/Donation
```

Here are the files that were deleted and we were successful in retrieving them.



Exif Data Recovery

A photo's EXIF data contains a lot of information about your camera. It also possibly show where the picture was taken by providing the GPS coordinates. There are a lot of details others can see when you are sharing your pictures. Every time you take a picture using you digital device, it saves a file to its storage. It records the metadata, such as the date, time, camera settings. If your camera comes with GPS capabilities, it can record the geolocation metadata. This allows you and others to see the location of where the picture was taken.



With the use of forensic tools, the above image was discovered and recovered using Exif Data Recovery. Picture Info from the site https://exifinfo.org/

Exif Info.org;/'

Online tool to analyze and display the meta-data in images and other media files.

The location of interest was verified using GPS coordinates.The photo was uploaded to Exif Info.org and the metadata analysis revealed a location. (see below)

The Exif info contains all the metadata extracted from the imported image that is uploaded to the ExitInfo.org website. The image metadata contains information about the image, such as color, resolution and image type. The image metadata also contains information about the camera settings such the camera type, ISO, ISO setting, exposure length and lens settings. When the image is taken is taken with a cellular phone that is GPS capable, the location of where the picture was taken is also stored in metadata of the image.

Exif Info: IMG_1767[2].JPG

File

Filename
IMG_1767[2].JPG
File Size
2.0 MB
File Type
JPEG
File Type Extension
jpg
MIME Type
image/jpeg

Exif Byte Order
Big-endian (Motorola, MM)
Image Width
4032
Image Height
3024
Encoding Process
Baseline DCT, Huffman coding
Bits Per Sample
8

Color Components
3
Y Cb Cr Sub Sampling
YCbCr4:2:0 (2 2)

EXIF

Make

Apple
Camera Model Name
iPhone 6s Plus
Orientation
Rotate 90 CW
X Resolution
72
Y Resolution
72
Resolution Unit
inches
Software
12.1.4
Modify Date
2019:11:16 14:11:56
Y Cb Cr Positioning
Centered
Exposure Time
1/220
F Number
2.2
Exposure Program
Program AE
ISO
25
Exif Version
0221
Date/Time Original
2019:11:16 14:11:56
Create Date
2019:11:16 14:11:56
Components
Configuration
Y, Cb, Cr, -
Shutter Speed Value
1/220
Aperture Value
2.2
Brightness Value
7.638587105
Exposure Compensation
0
Metering Mode
Multi-segment
Flash
Auto, Did not fire
Focal Length
4.2 mm
Subject Area
2015 1511 2217 1330
Sub Sec Time Original
787
Sub Sec Time Digitized
787
Flashpix Version
0100
Color Space
sRGB
Exif Image Width
4032
Exif Image Height
3024
Sensing Method
One-chip color area
Scene Type
Directly photographed
Exposure Mode
Auto
White Balance
Auto
Focal Length In 35mm
Format
29 mm
Scene Capture Type
Standard
Lens Info
4.15mm f/2.2
Lens Make
Apple
Lens Model
iPhone 6s Plus back
camera 4.15mm f/2.2
GPS Latitude Ref
North
GPS Latitude
49 deg 1' 16.60"
GPS Longitude Ref
West
GPS Longitude
122 deg 48' 20.42"
GPS Altitude Ref
Above Sea Level
GPS Altitude
10.90522265 m
GPS Time Stamp
22:11:56
GPS Speed Ref
km/h
GPS Speed
0
GPS Img Direction Ref
True North
GPS Img Direction
165.3394623
GPS Dest Bearing Ref
True North
GPS Dest Bearing
165.3394623
GPS Date Stamp
2019:11:16
GPS Horizontal
Positioning Error
5 m
Compression
JPEG (old-style)
Thumbnail Offset
2158

Thumbnail Length
7228

MakerNotes

Run Time Flags
Valid
Run Time Value
138950374011375
Run Time Scale
1000000000
Run Time Epoch
0

Composite

Aperture
2.2
GPS Altitude
10.9 m Above Sea Level
GPS Date/Time
2019:11:16 22:11:56Z
GPS Latitude
49 deg 1' 16.60" N
GPS Longitude
122 deg 48' 20.42" W
GPS Position
49 deg 1' 16.60" N, 122 deg 48' 20.42" W
Image Size
4032x3024
Megapixels
12.2
Run Time Since Power Up
1 days 14:35:50
Scale Factor To 35 mm Equivalent
7.0
Shutter Speed
1/220
Create Date
2019:11:16 14:11:56.787
Date/Time Original
2019:11:16 14:11:56.787
Thumbnail Image
[binary data]
Circle Of Confusion
0.004 mm
Field Of View
63.7 deg
Focal Length
4.2 mm (35 mm equivalent: 29.0 mm)
Hyperfocal Distance
1.82 m
Light Value
12.1

Conclusion

Despite the efforts of the suspects to hide their participation in this exchange ring, we were able to recover evidence that we can use to prosecute them. Using investigative techniques based on contextual clues and probable cause, we determined that suspect one used steganography software, in particular a program called steghide, to hide illicit images that linked them to a criminal activity. Through analysis of file signatures and headers, we were able to extract a file using binwalk. Using that password, we extracted the embedded file. We were able to extract the data from the file.

Using the information gained from the evidence found on the first suspects machine, we were able to find the presence of these pictures on an internet forum. After receiving a warrant we gained access to suspect two's computer, only to find no presence of the files locally. Using investigative techniques, we were able to find two hidden directories under the "asmith" folder. Subsequently, we were able to extract these files using Testdisk. We were also able to recover pictures files from the suspect computer using the Photorec utility. Finally, we investigated the exif data in the files, and using the gps coordinates and other info, were able to identify more suspects that may have been involved in the exchange.

References

http://www.ncdsv.org/images/NDAA_Steganography_Update_v1_no1_2004.pdf

https://tools.kali.org/forensics/binwalk

http://www.sleuthkit.org/sleuthkit/man/icat.html

http://www.sleuthkit.org/sleuthkit/man/fls.html

http://steghide.sourceforge.net/documentation/manpage.php

https://linux.die.net/man/1/testdisk

https://linux.die.net/man/1/photorec