ToanNgo
Date:2/7/2024

Lab3

For this lab report we will be talking about what is metadata, what are hash value and how are they change if something were done on the file, we will also explore what can you do in the metadata hexadecimal value, and also we will be looking into how people able to store a secret message in a image, and how are this are connect to the metadata. But first we need to understand what is metadata.

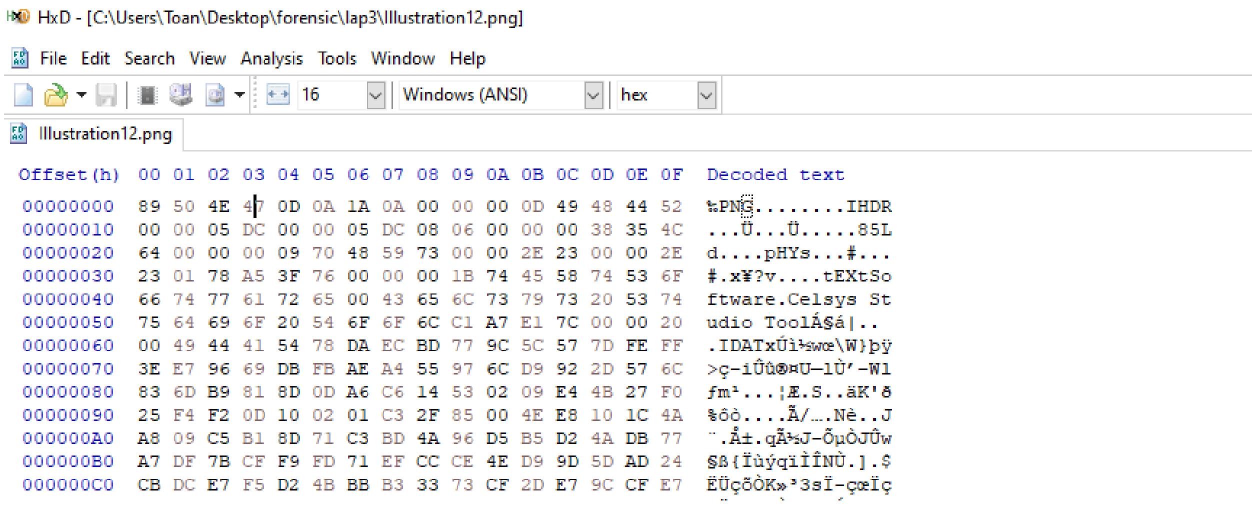
What is metadata? Metadata is a string of hexadecimal numbers that are stored along with the files when they initially create and save, their job is to store everything in relation to the file type that includes the date it was created,the system that created,files type ect.. Just to simplify this concept, just think of metadata as a container that stores all information that relates to the file in a hex value. Now that we have some understanding on what is metadata, let us go over what is hash value. A hash value is a mapping method that maps a target data in a container of fixed size bits like SHA512(this generates a hash but of 512 bits), and sometimes A hash value can be treated as an authenticator for the files because of how the algorithm generates its value. The unique thing about a hash value is that every file will have a unique hash value even if the change is miniscule. The only time that the hash values are the same is when you compare the same files with no modification done to it. And this is why checking hashing value can be considered a form of authentication.

Why are metadata important ? Metadata helps digital forensic to identify what type of files they are looking at without directly executing/ running the files themself risking whatever on that files files to run. This is also one of the ways to check if the files are clear or not. Metadata also provides digital forensics some information about what type of machine the files are coming from because as we talk before metadata jobs is to store that information, this will help the digital forensics team alot when they need to track back to the files origin. Next, why are hash values important? Hash value provides a way for a file to encrypt themself when it gets sent over the internet, and it also serves as an authentication for the files as well when it reaches its destination.Because as we talk before no two files hash value are the same unless they are the same files.

Now that we have some kinda understanding of how importance metadata is to forensic investigator and how hash value help with authentication, lets us move on to some more technical side of how do you look in to metadata, how to add secret message in the metadata, how it hash value change when the message are added, and finally how do you add a secret message in a image files not metadata file but it on the image themself.

First in order to see metadata we will have to download this tool HxD - Freeware Hex Editor and Disk Editor. This will allow us to see the hexadecimal value that was stored along with the files when they were being created.

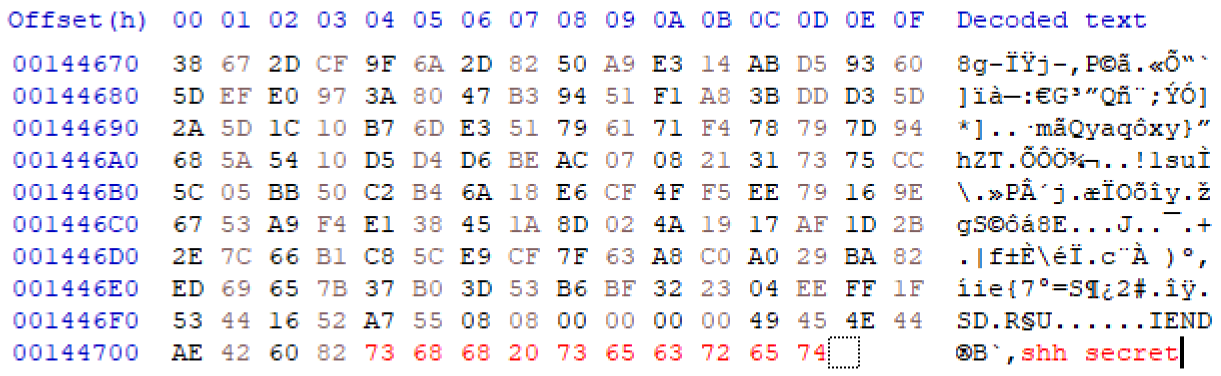
ToanNgo
Date:2/7/2024


HxD - [C:\Users\Toan\Desktop\forensic\lap3\Illustration12.png]

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text

00000000  89 50 4E 47 0D 0A 1A 0A 00 00 00 0D 49 48 44 52  ‰PNG........IHDR
00000010  00 00 05 DC 00 00 05 DC 08 06 00 00 00 38 35 4C  ...Ü...Ü.....85L
00000020  64 00 00 00 09 70 48 59 73 00 00 2E 23 00 00 2E  d....pHYs...#...
00000030  23 01 78 A5 3F 76 00 00 00 1B 74 45 58 74 53 6F  #.x¥?v....tEXtSo
00000040  66 74 77 61 72 65 00 43 65 6C 73 79 73 20 53 74  ftware.Celsys St
00000050  75 64 69 6F 20 54 6F 6F 6C C1 A7 E1 7C 00 00 20  udio ToolÁ§á|..
00000060  00 49 44 41 54 78 DA EC BD 77 9C 5C 57 7D FE FF  .IDATxÚì½wœ\W}þÿ
00000070  3E E7 96 69 DB FB AE A4 55 97 6C D9 92 2D 57 6C  >ç–iÛû®¤U—lÙ’-Wl
00000080  83 6D B9 81 8D 0D A6 C6 14 53 02 09 E4 4B 27 F0  ƒm¹...¦Æ.S..äK'ð
00000090  25 F4 F2 0D 10 02 01 C3 2F 85 00 4E E8 10 1C 4A  %ôò....Ã/….Nè..J
000000A0  A8 09 C5 B1 8D 71 C3 BD 4A 96 D5 B5 D2 4A DB 77  ¨.Å±.qÃ½J–ÕµÒJÛw
000000B0  A7 DF 7B CF F9 FD 71 EF CC CE 4E D9 9D 5D AD 24  §ß{ÏùýqïÌÎNÙ.].$
000000C0  CB DC E7 F5 D2 4B BB B3 33 73 CF 2D E7 9C CF E7  ËÜçõÒK»³3sÏ-çœÏç
```

(this is how the hex of the files look, and if you look closely in the beginning the hexadecimal serves as a signature of what type of files this hexadecimal belongs to. And in my case it contains a PNG signature. And just a little notes for the decoded text all the dot on it are symbolize that the bits of small data that exist within the files, and because of how small it is, they cannot decode it and in it place they use the dot to replace it.)

Next I will go to the end of the hex files and add a little secret message into the image without making any change to the actual image. However when we compare the file size between the modified hex one with the original files, the size of both are different.



```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text

00144670  38 67 2D CF 9F 6A 2D 82 50 A9 E3 14 AB D5 93 60  8g-ÏŸj-‚P©ã.«Õ“`
00144680  5D EF E0 97 3A 80 47 B3 94 51 F1 A8 3B DD D3 5D  ]ïà—:€G³”Qñ¨;ÝÓ]
00144690  2A 5D 1C 10 B7 6D E3 51 79 61 71 F4 78 79 7D 94  *]..·mãQyaqôxy}”
001446A0  68 5A 54 10 D5 D4 D6 BE AC 07 08 21 31 73 75 CC  hZT.ÕÔÖ¾¬..!1suÌ
001446B0  5C 05 BB 50 C2 B4 6A 18 E6 CF 4F F5 EE 79 16 9E  \.»PÂ´j.æÏOõîy.ž
001446C0  67 53 A9 F4 E1 38 45 1A 8D 02 4A 19 17 AF 1D 2B  gS©ôá8E...J..¯.+
001446D0  2E 7C 66 B1 C8 5C E9 CF 7F 63 A8 C0 A0 29 BA 82  .|f±È\éÏ.c¨À )º‚
001446E0  ED 69 65 7B 37 B0 3D 53 B6 BF 32 23 04 EE FF 1F  íie{7°=S¶¿2#.îÿ.
001446F0  53 44 16 52 A7 55 08 08 00 00 00 00 49 45 4E 44  SD.R§U......IEND
00144700  AE 42 60 82 73 68 68 20 73 65 63 72 65 74        ®B`‚shh secret
```

(in this example i added the secret message in the bottom of the hex files without actuary making change to the files contents, also extra note the last 4 bits of the of the hex files serve as a signature for most files, and the first 8 or 12 bits in the beginning usually serve as a signature for what files it is.)

ToanNgo
Date:2/7/2024

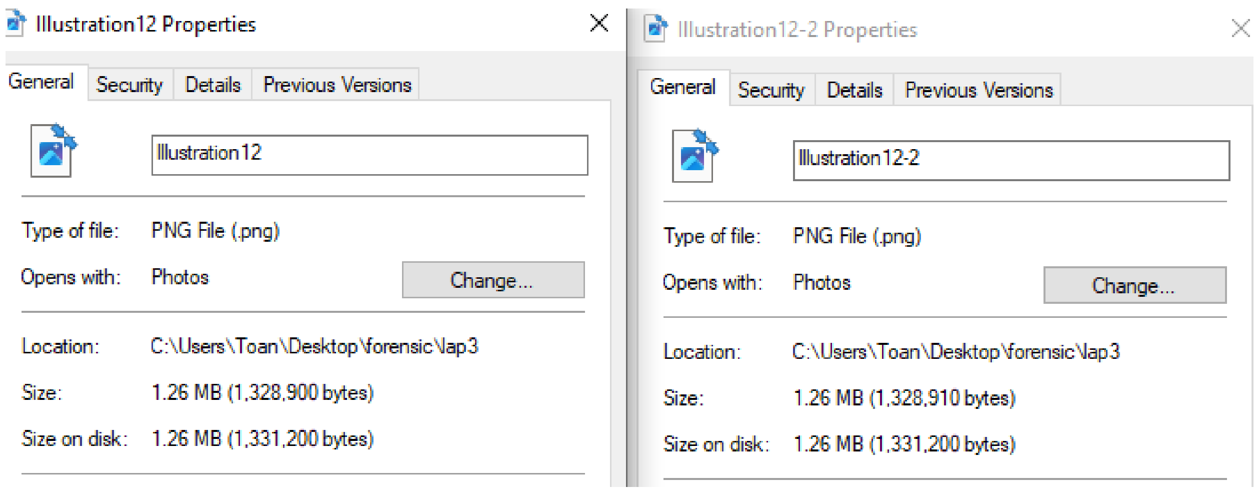


Illustration12 Properties

General | Security | Details | Previous Versions

Illustration12

Type of file: PNG File (.png)
Opens with: Photos
Location: C:\Users\Toan\Desktop\forensic\lap3
Size: 1.26 MB (1,328,900 bytes)
Size on disk: 1.26 MB (1,331,200 bytes)

Illustration12-2 Properties

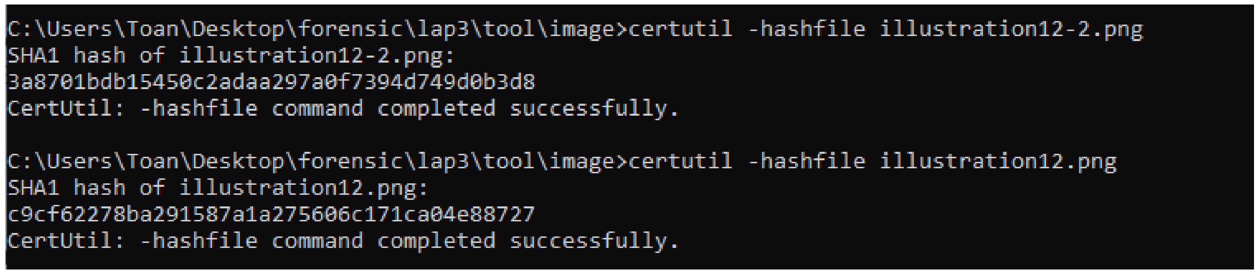
General | Security | Details | Previous Versions

Illustration12-2

Type of file: PNG File (.png)
Opens with: Photos
Location: C:\Users\Toan\Desktop\forensic\lap3
Size: 1.26 MB (1,328,910 bytes)
Size on disk: 1.26 MB (1,331,200 bytes)

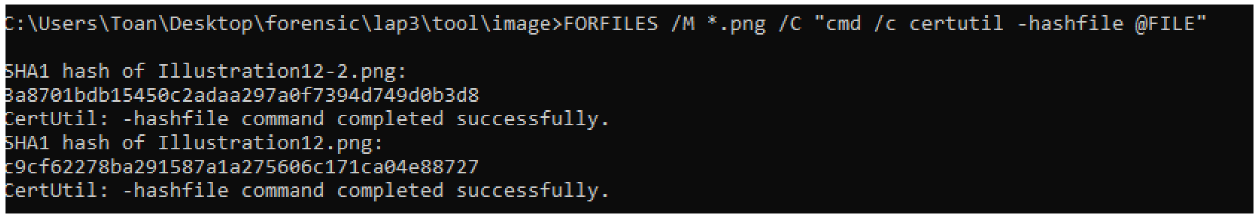
(This shows that modifying the hex code does change the actual files size, even if the size that it changes is small, it does change it. This does show that you can indeed hide a secret message in any files and pass it along without anyone noticing, unless someone actually checks the files size and looks at the hex code in those files.)

Now we move on hashing files using cmd. In window you can call certutil -hashfile (input your file here)* to create a hash for your files. and because of how unique the hashing algorithm is, each and every file that it hash will not contain the same hashing number, unless it is the same file. This is also the reason that checking the hash number between two files can serve as a form of authentication.



```
C:\Users\Toan\Desktop\forensic\lap3\tool\image>certutil -hashfile illustration12-2.png
SHA1 hash of illustration12-2.png:
3a8701bdb15450c2adaa297a0f7394d749d0b3d8
CertUtil: -hashfile command completed successfully.

C:\Users\Toan\Desktop\forensic\lap3\tool\image>certutil -hashfile illustration12.png
SHA1 hash of illustration12.png:
c9cf62278ba291587a1a275606c171ca04e88727
CertUtil: -hashfile command completed successfully.
```

(In this example I have two similar files, one with the secret message and the original one. If we compare the hash number between two files you can clearly see that they both contain a very different hash value. This shows us that they are not the same files, there are some modifications being done to one of the files.)



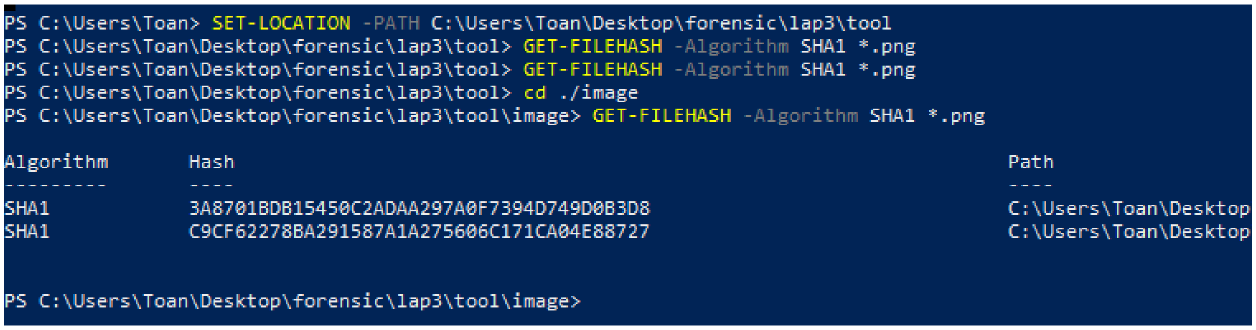
```
:\Users\Toan\Desktop\forensic\lap3\tool\image>FORFILES /M *.png /C "cmd /c certutil -hashfile @FILE"

SHA1 hash of Illustration12-2.png:
3a8701bdb15450c2adaa297a0f7394d749d0b3d8
CertUtil: -hashfile command completed successfully.
SHA1 hash of Illustration12.png:
c9cf62278ba291587a1a275606c171ca04e88727
CertUtil: -hashfile command completed successfully.
```

(In this example we test out hashing all the files in relation to .png in one loop. Code rundown: FORFILES(run a command for multiple files, you can think of it like a

ToanNgo
Date:2/7/2024

loop that runs through multiple files.) /m(is the search mask, ie what it should search for.)/c(this is the command, what the files should do following the next input) "cmd /c certutil -hashfile @FILE" (this tell the cmd to run certutil hashfile for all files). What you get from running this is, all the .png files in the directory will turn into hash and display it on the screen.

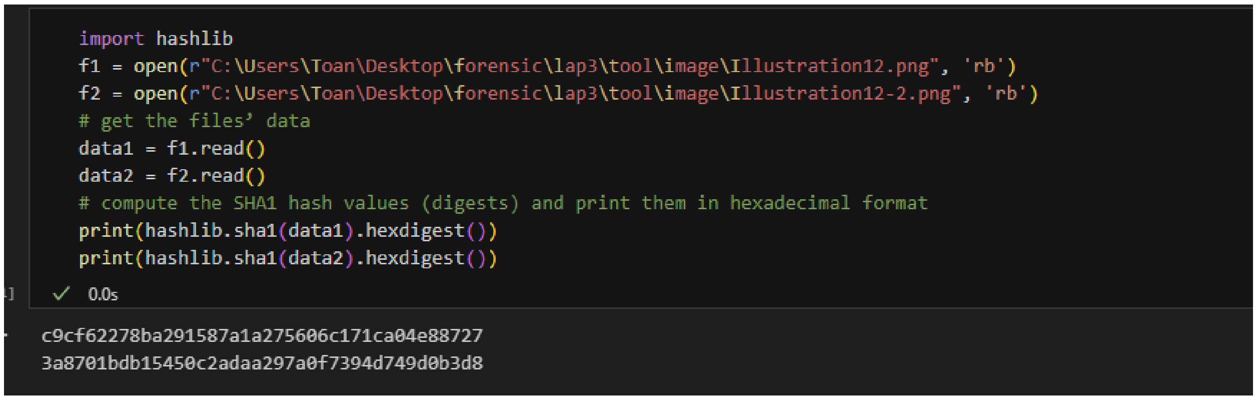
This is how you hash the files, if you use a windows powershell.



```
PS C:\Users\Toan> SET-LOCATION -PATH C:\Users\Toan\Desktop\forensic\lap3\tool
PS C:\Users\Toan\Desktop\forensic\lap3\tool> GET-FILEHASH -Algorithm SHA1 *.png
PS C:\Users\Toan\Desktop\forensic\lap3\tool> GET-FILEHASH -Algorithm SHA1 *.png
PS C:\Users\Toan\Desktop\forensic\lap3\tool> cd ./image
PS C:\Users\Toan\Desktop\forensic\lap3\tool\image> GET-FILEHASH -Algorithm SHA1 *.png

Algorithm       Hash                                                                   Path
---------       ----                                                                   ----
SHA1            3A8701BDB15450C2ADAA297A0F7394D749D0B3D8                               C:\Users\Toan\Desktop
SHA1            C9CF62278BA291587A1A275606C171CA04E88727                               C:\Users\Toan\Desktop


PS C:\Users\Toan\Desktop\forensic\lap3\tool\image>
```

(the set-location acts the same way as cd in the cmd, it changes your current window pointer directory location. The get-filehash(this will set up a file file for hashing) -Algorithm(this mean that the following line will be applying as a algorithm for hashing) SHA1 *.png(this means that you are applying a group SHA1 algorithm(there are four main groups of hashing SHA0,SHA1,SHA2,SHA3. Each of these serve a purpose for hashing , however some of these are already depreciated and shouldn't be used anymore.) for hashing all the png files).

Now let's move on to hashing in python and hidden messages in the image. First, how do you make a hash in python?



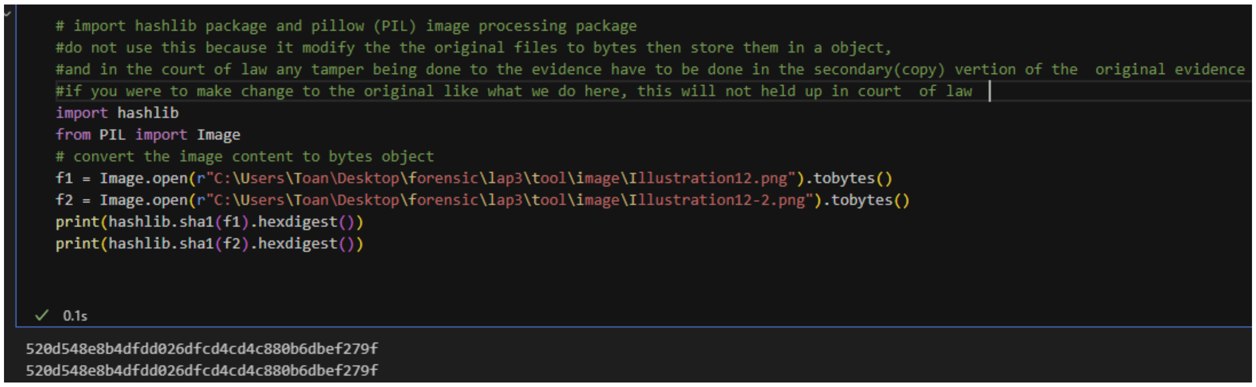
```python
import hashlib
f1 = open(r"C:\Users\Toan\Desktop\forensic\lap3\tool\image\Illustration12.png", 'rb')
f2 = open(r"C:\Users\Toan\Desktop\forensic\lap3\tool\image\Illustration12-2.png", 'rb')
# get the files' data
data1 = f1.read()
data2 = f2.read()
# compute the SHA1 hash values (digests) and print them in hexadecimal format
print(hashlib.sha1(data1).hexdigest())
print(hashlib.sha1(data2).hexdigest())
```

```
✓ 0.0s

c9cf62278ba291587a1a275606c171ca04e88727
3a8701bdb15450c2adaa297a0f7394d749d0b3d8
```

(code run down: you import a hashlib library, open the two file that you want to hash in "read binary mode('rb')" store the read binary in to two object data1 and data2, the you call the hashlib library running the SHA1 algorithms for the hash then print it out. This should give you a simple hashing method for the items that you want to hash. And also as you can see the two values are completely different and that is because one file has a secret message added in the metadata hex value.

Now this is something you shouldn't do as a digital forensic investigator.

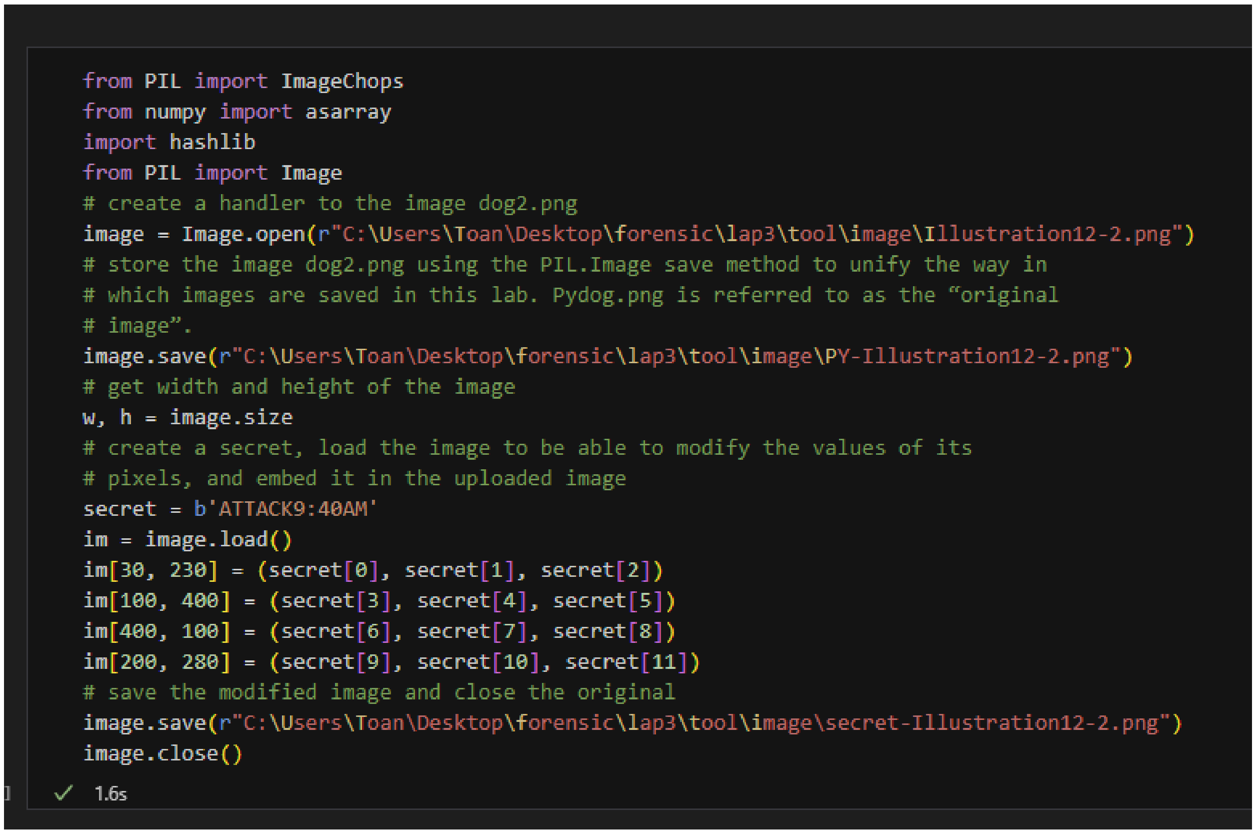
ToanNgo
Date:2/7/2024



```
# import hashlib package and pillow (PIL) image processing package
#do not use this because it modify the the original files to bytes then store them in a object,
#and in the court of law any tamper being done to the evidence have to be done in the secondary(copy) vertion of the  original evidence
#if you were to make change to the original like what we do here, this will not held up in court  of law
import hashlib
from PIL import Image
# convert the image content to bytes object
f1 = Image.open(r"C:\Users\Toan\Desktop\forensic\lap3\tool\image\Illustration12.png").tobytes()
f2 = Image.open(r"C:\Users\Toan\Desktop\forensic\lap3\tool\image\Illustration12-2.png").tobytes()
print(hashlib.sha1(f1).hexdigest())
print(hashlib.sha1(f2).hexdigest())

✓ 0.1s

520d548e8b4dfdd026dfcd4cd4c880b6dbef279f
520d548e8b4dfdd026dfcd4cd4c880b6dbef279f
```

(what is happening here is that you open the files, so far so good, however the thing that make this not good is at the end of that statement, the “.tobytes" convert what you open to bytes and that is not good because as a digital forensic investigator you cannot make any change to the original files at all, if you want to change or experiment on piece of evidence you will have to make a copy of that evidence and change that, leaving the original clean as it can be. And the logic behind it is that you can easily fabricate a digital evidence in comparison to a physical one, therefore you can not make any change to the original at all.)

Next adding a secret message to an image.



```
from PIL import ImageChops
from numpy import asarray
import hashlib
from PIL import Image
# create a handler to the image dog2.png
image = Image.open(r"C:\Users\Toan\Desktop\forensic\lap3\tool\image\Illustration12-2.png")
# store the image dog2.png using the PIL.Image save method to unify the way in
# which images are saved in this lab. Pydog.png is referred to as the “original
# image”.
image.save(r"C:\Users\Toan\Desktop\forensic\lap3\tool\image\PY-Illustration12-2.png")
# get width and height of the image
w, h = image.size
# create a secret, load the image to be able to modify the values of its
# pixels, and embed it in the uploaded image
secret = b'ATTACK9:40AM'
im = image.load()
im[30, 230] = (secret[0], secret[1], secret[2])
im[100, 400] = (secret[3], secret[4], secret[5])
im[400, 100] = (secret[6], secret[7], secret[8])
im[200, 280] = (secret[9], secret[10], secret[11])
# save the modified image and close the original
image.save(r"C:\Users\Toan\Desktop\forensic\lap3\tool\image\secret-Illustration12-2.png")
image.close()

✓ 1.6s
```

ToanNgo
Date:2/7/2024

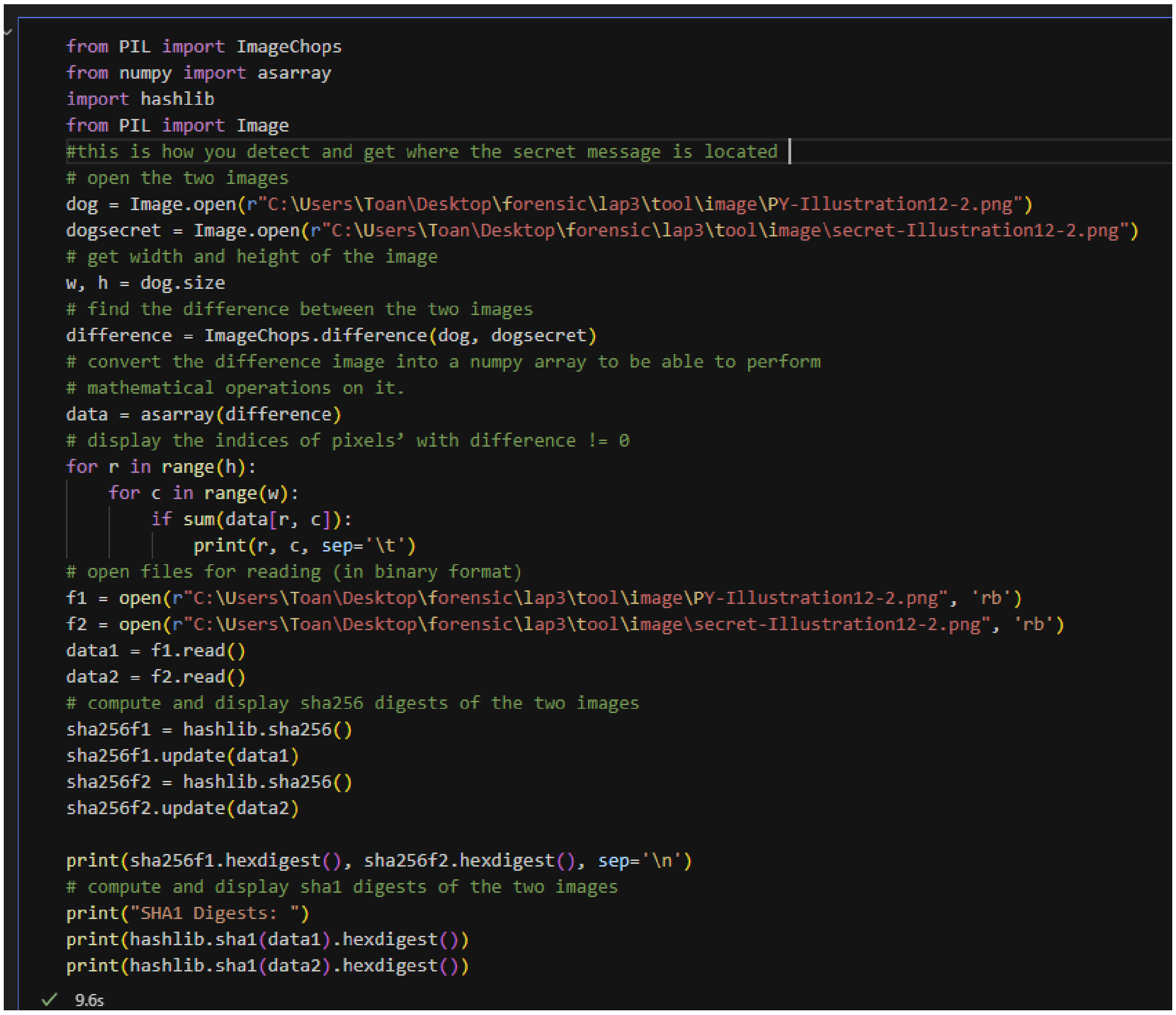
(code run down: you open a control image, save the control image as a duplicate(this is the image we will be attempting to add a message in), get the height and width of the image by calling ".size", create a message and wrap it in 'b'(this will convert the string to byte), load the image, base on the height and width of the image that we just get in the previous steps set a coordinate that each letter gonna reside in(for this experiment the letter A T T are reside in height:30 and width: 230),then save the image. If you were to open the image up you will not notice the difference at all.)

(the original image)

(image with added secret message)

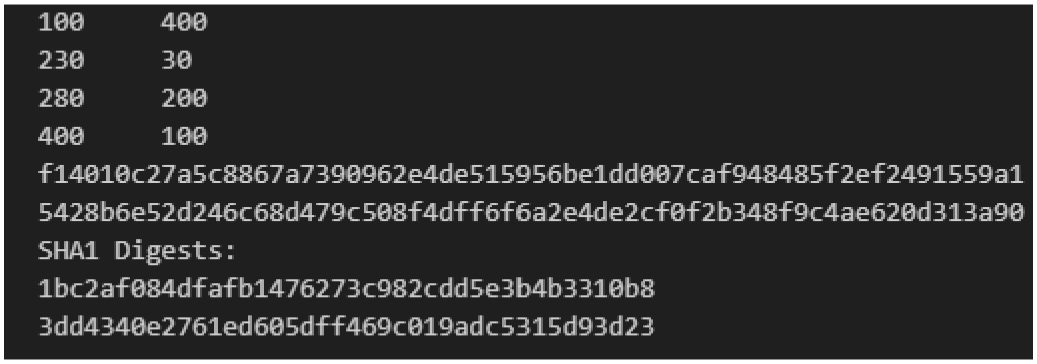
ToanNgo
Date:2/7/2024

Now if we try and find the difference in python however it will show you where the message is located in.



```
from PIL import ImageChops
from numpy import asarray
import hashlib
from PIL import Image
#this is how you detect and get where the secret message is located
# open the two images
dog = Image.open(r"C:\Users\Toan\Desktop\forensic\lap3\tool\image\PY-Illustration12-2.png")
dogsecret = Image.open(r"C:\Users\Toan\Desktop\forensic\lap3\tool\image\secret-Illustration12-2.png")
# get width and height of the image
w, h = dog.size
# find the difference between the two images
difference = ImageChops.difference(dog, dogsecret)
# convert the difference image into a numpy array to be able to perform
# mathematical operations on it.
data = asarray(difference)
# display the indices of pixels' with difference != 0
for r in range(h):
    for c in range(w):
        if sum(data[r, c]):
            print(r, c, sep='\t')
# open files for reading (in binary format)
f1 = open(r"C:\Users\Toan\Desktop\forensic\lap3\tool\image\PY-Illustration12-2.png", 'rb')
f2 = open(r"C:\Users\Toan\Desktop\forensic\lap3\tool\image\secret-Illustration12-2.png", 'rb')
data1 = f1.read()
data2 = f2.read()
# compute and display sha256 digests of the two images
sha256f1 = hashlib.sha256()
sha256f1.update(data1)
sha256f2 = hashlib.sha256()
sha256f2.update(data2)

print(sha256f1.hexdigest(), sha256f2.hexdigest(), sep='\n')
# compute and display sha1 digests of the two images
print("SHA1 Digests: ")
print(hashlib.sha1(data1).hexdigest())
print(hashlib.sha1(data2).hexdigest())
```

✓ 9.6s

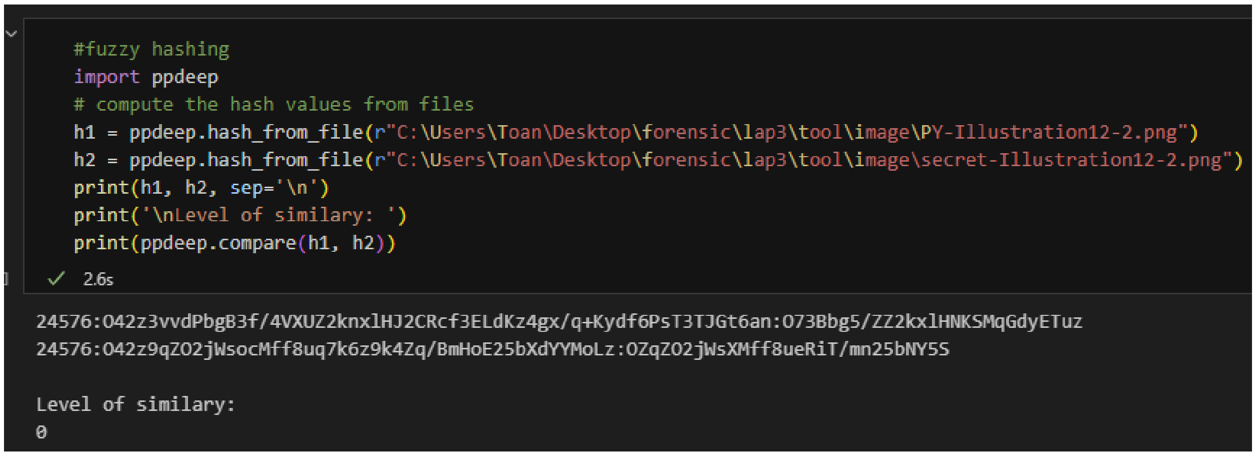


```
100     400
230     30
280     200
400     100
f14010c27a5c8867a7390962e4de515956be1dd007caf948485f2ef2491559a1
5428b6e52d246c68d479c508f4dff6f6a2e4de2cf0f2b348f9c4ae620d313a90
SHA1 Digests:
1bc2af084dfafb1476273c982cdd5e3b4b3310b8
3dd4340e2761ed605dff469c019adc5315d93d23
```

(the 100 400 and so on is the location of the message in the image based on the above code. If we were to run a hash encryption on this, the value will be different between the two files, this proves to us that the two files may look the same but they are indeed not the same files.)

ToanNgo
Date:2/7/2024

For the last thing, fuzzy hashing. Is a hashing method that hashes and compares the similar hash values, in theory if the two files are the same then it should have similar hash values.



```
#fuzzy hashing
import ppdeep
# compute the hash values from files
h1 = ppdeep.hash_from_file(r"C:\Users\Toan\Desktop\forensic\lap3\tool\image\PY-Illustration12-2.png")
h2 = ppdeep.hash_from_file(r"C:\Users\Toan\Desktop\forensic\lap3\tool\image\secret-Illustration12-2.png")
print(h1, h2, sep='\n')
print('\nLevel of similary: ')
print(ppdeep.compare(h1, h2))
```

✓ 2.6s

```
24576:O42z3vvdPbgB3f/4VXUZ2knxlHJ2CRcf3ELdKz4gx/q+Kydf6PsT3TJGt6an:O73Bbg5/ZZ2kxlHNKSMqGdyETuz
24576:O42z9qZO2jWsocMff8uq7k6z9k4Zq/BmHoE25bXdYYMoLz:OZqZO2jWsXMff8ueRiT/mn25bNY5S

Level of similary:
0
```

(in my case the files have being modify by adding the secret message so the similarly between two files are 0 (this i'm not totally sure about))

So to sum it all up. Hashing is very good for authenticating the legitimacy of the files, and metadata can contain the files data and you could hide a message there as well. So my takeaway from this lab is that even if the files that look similar on the surface, it doesn't mean that it's a similar file, and metadata can be used to help track where the files are coming from based on the hexadecimal value. And just a little extra note in the metadata the first few bits in the hexadecimal value is the signature of the files, and in my case the beginning 8 bits stand for a png file.

ToanNgo
Date:2/7/2024

Resource page

Metadata source -
https://www.techtarget.com/whatis/definition/metadata
Hash value in metadata -
https://www.hklaw.com/en/insights/publications/2022/03/hash-functions-their-utility-for-both-clients-and-lawyers

Tools
https://mh-nexus.de/en/hxd/