



## BÁO CÁO THỰC HÀNH LAB 3

Môn học: Pháp chứng kỹ thuật số

Nhóm: Pha Pha

### THÀNH VIÊN THỰC HIỆN:

STT	Họ và tên	MSSV
1	Nguyễn Đoàn Xuân Bình	19521265
2	Trần Hoàng Khang	19521671
3	Nguyễn Mỹ Quynh	19520241

# BÁO CÁO CHI TIẾT

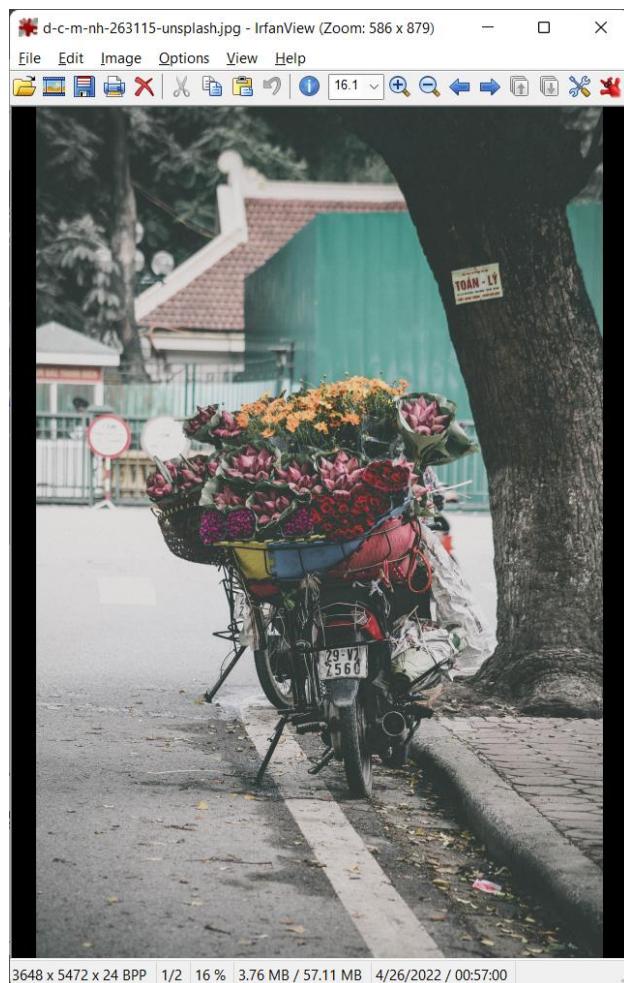
**Lưu ý:** Trong bài có sử dụng một số tool mà trên các repo chính thống, hay các trang chủ và các nguồn download chính thức bị lỗi/gặp vấn đề/dead link thì mình có thể dùng [Wayback Machine](#) để “lùi lại” và xem những version trước của trang web download đó (theo mình làm thì mình lấy phiên bản cách đây 6-9 năm) thì mình download và dùng được tool như bình thường



## Kịch bản 01-a. Thực hiện phân tích thông tin tập tin ảnh

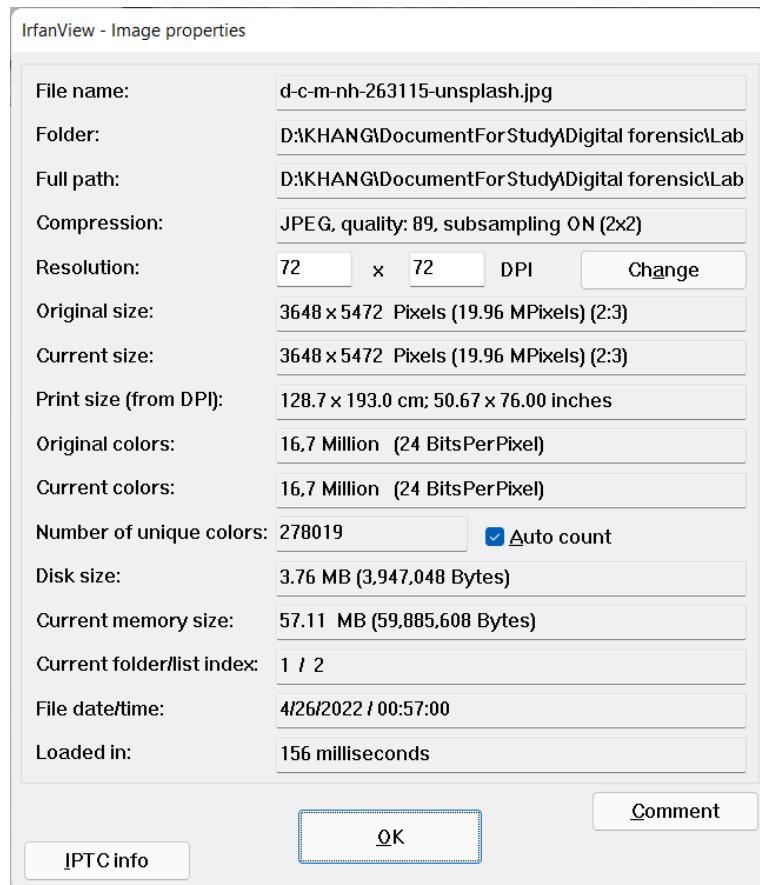
- Tài nguyên thực hiện, nằm trong thư mục kb-01-a
- Yêu cầu: Cung cấp các thông tin chi tiết liên quan tới các bức ảnh trên bằng phần mềm IrfanView

Import file **d-c-m-nh-263115-unsplash.jpg**



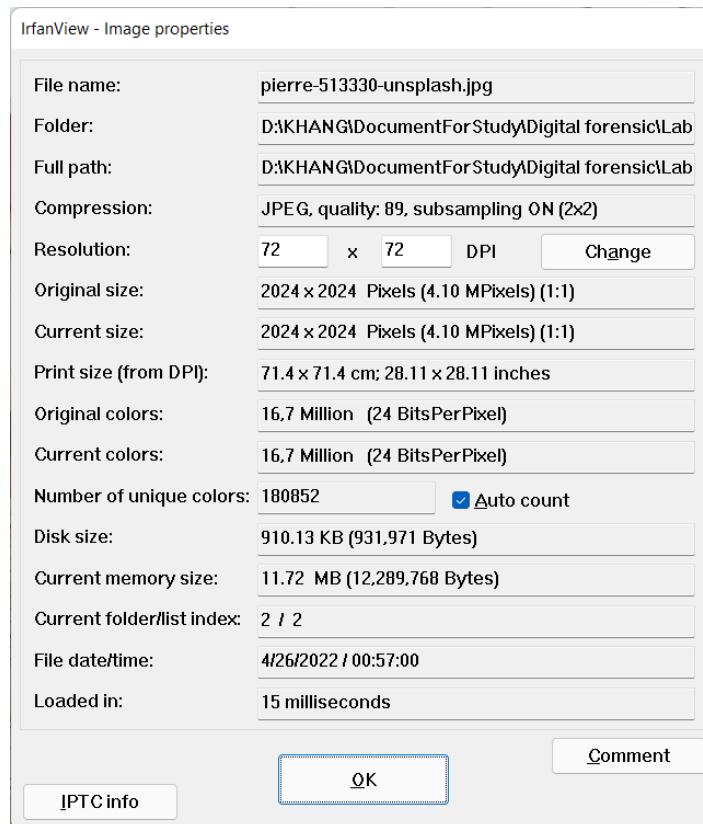
Chọn Image -> Information:

## Thi thực hành cuối kì



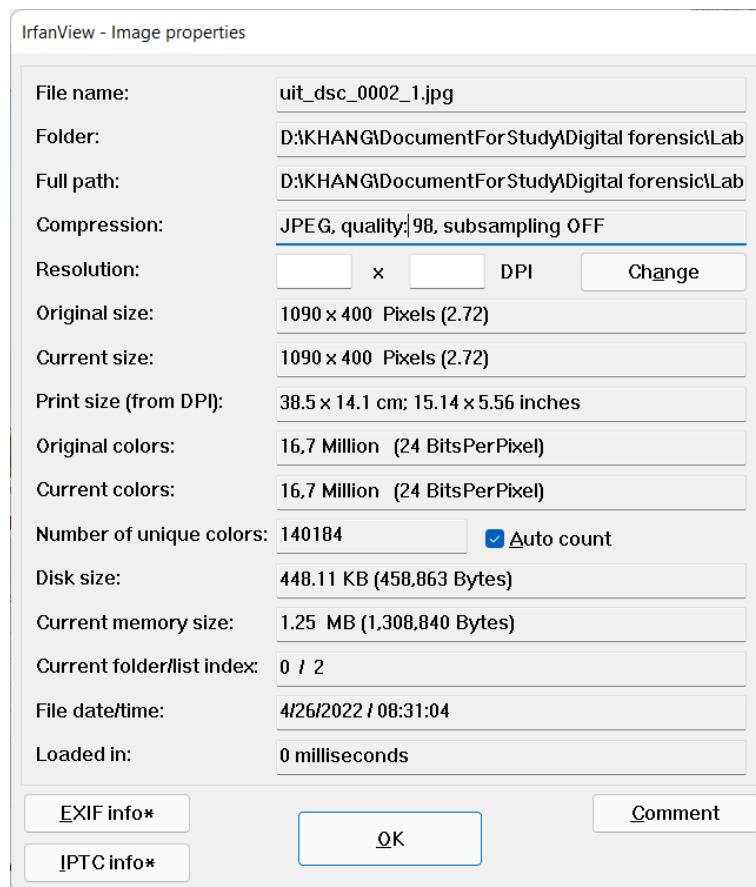
Ảnh này không có thông tin EXIF để chiết xuất.

Tương tự với ảnh **pierre-513330-unsplash.jpg**. Thông tin ảnh :



## Thi thực hành cuối kì

Ảnh cũng không có mục EXIF luôn. Examine ảnh uit\_dsc\_0002\_1.jpg



Có trường thông tin EXIF được trích ra. Vào xem thử:



Lấy được thông tin giá trị 2 trường FileName và Copyright(bản quyền) là HuNGo

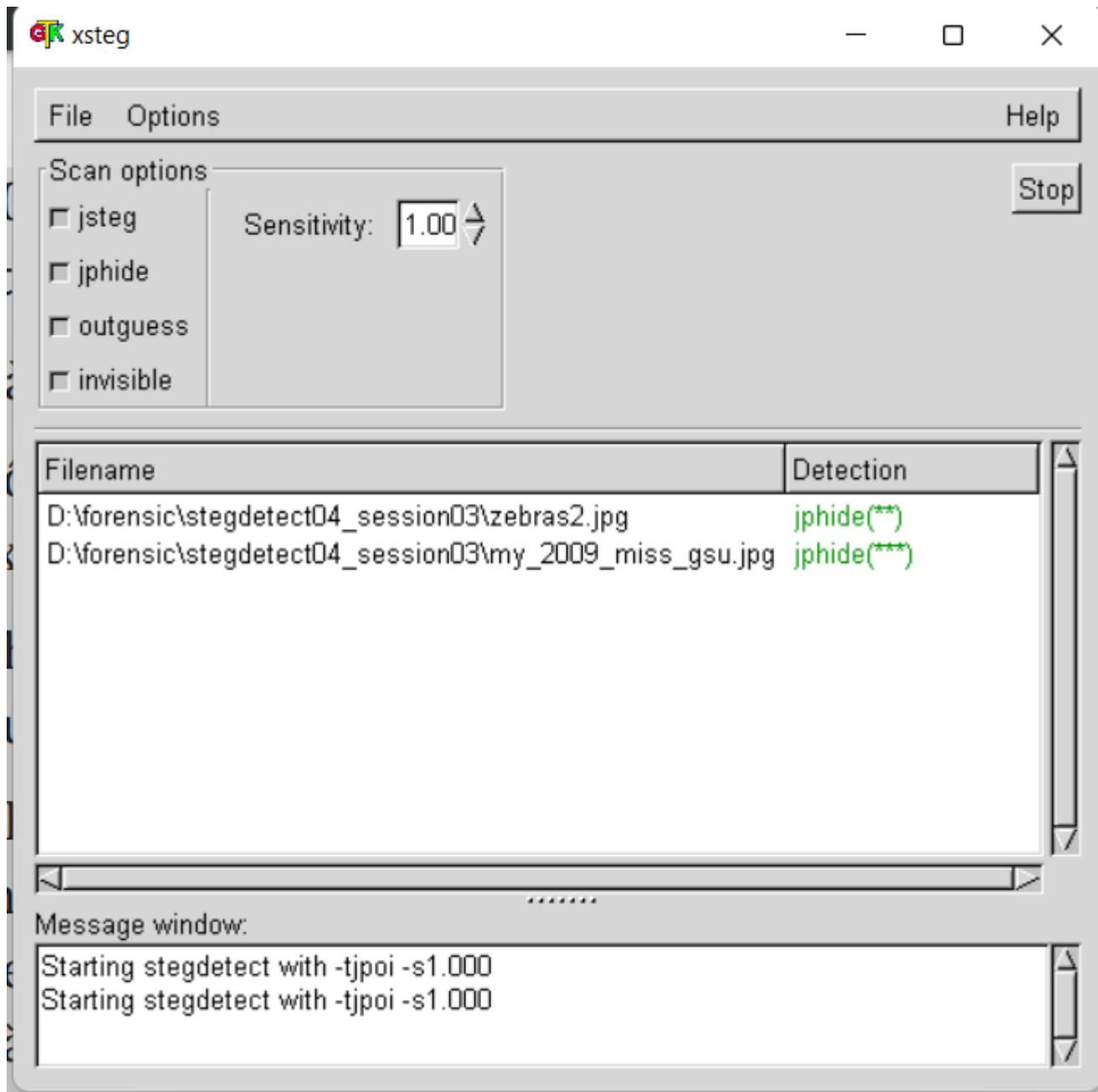
## Thi thực hành cuối kì

### Kịch bản 01-c. Phát hiện dữ liệu được giấu trong ảnh JPEG sử dụng StegDetect.

- Tài nguyên: image\_session03.zip
- Công cụ: stegdetect04\_session03.zip. Thực hiện giải nén và chạy file “xsteg.exe”
- Chọn thư mục chứa ảnh cần phân tích. Thực hiện quét và đưa ra kết quả, nhận xét.
- Thực hiện bẻ khóa mật khẩu trong quá trình giấu tin. (Chuẩn bị: my\_2009\_miss\_gsu.jpg - ảnh đã giấu thông tin ở kịch bản 02 bên trên, Zebras2.jpg, Stegbreak.exe). Mật khẩu tìm thấy là gì? Nhận xét về khả năng tìm thấy của công cụ?
- Giải nén thông tin chứa trong file ảnh có phát hiện ẩn giấu thông tin bằng mật khẩu tìm được.

Khi quét thử mục chứa các file ảnh my\_2009\_miss\_gsu.jpg và zebras2.jpg thì được kết quả.

## Thi thực hành cuối kì



Ở kịch bản 1-b thì em có cài đặt mật khẩu để giấu file là UIT nên tool xsteg.exe đã phát hiện ra file được che giấu.

Tương tự nó cũng đoán ra được file zebras2.jpg cũng chứa file ẩn

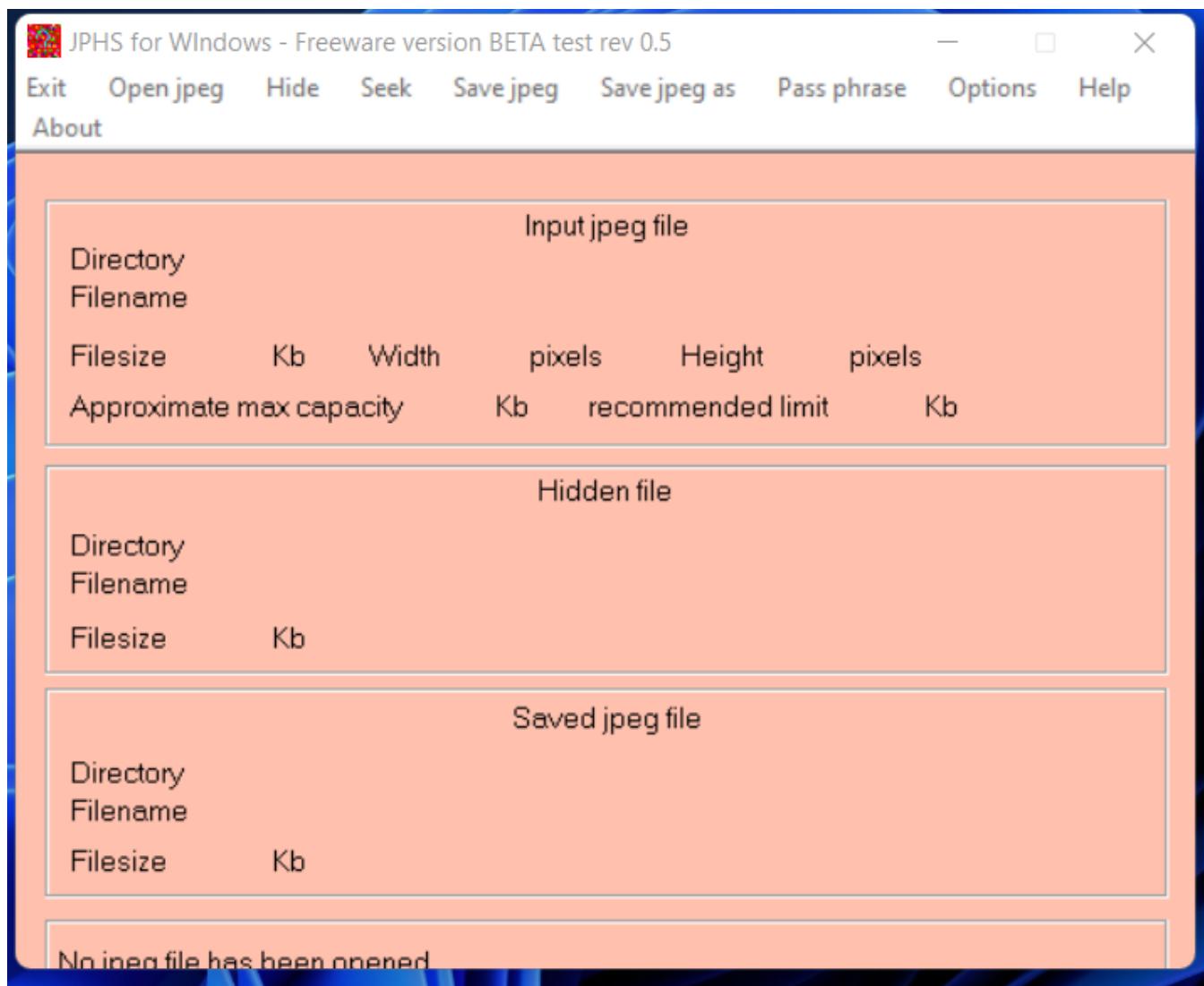
Sau đó em dùng tool stegbreak.exe để brute force mật khẩu bằng file MedDict.DIC.

```
PS D:\forensic\stegdetect04_session03> ./stegbreak -r rules.ini -f MedDict.DIC zebras2.jpg
Loaded 1 files
zebras2.jpg : jphide[v5](together)
Processed 1 files, found 1 embeddings.
Time: 3 seconds: Cracks: 68608, 22869.3 c/s
PS D:\forensic\stegdetect04_session03> ^0
```

Em đã tìm được mật khẩu giấu file là “together”.

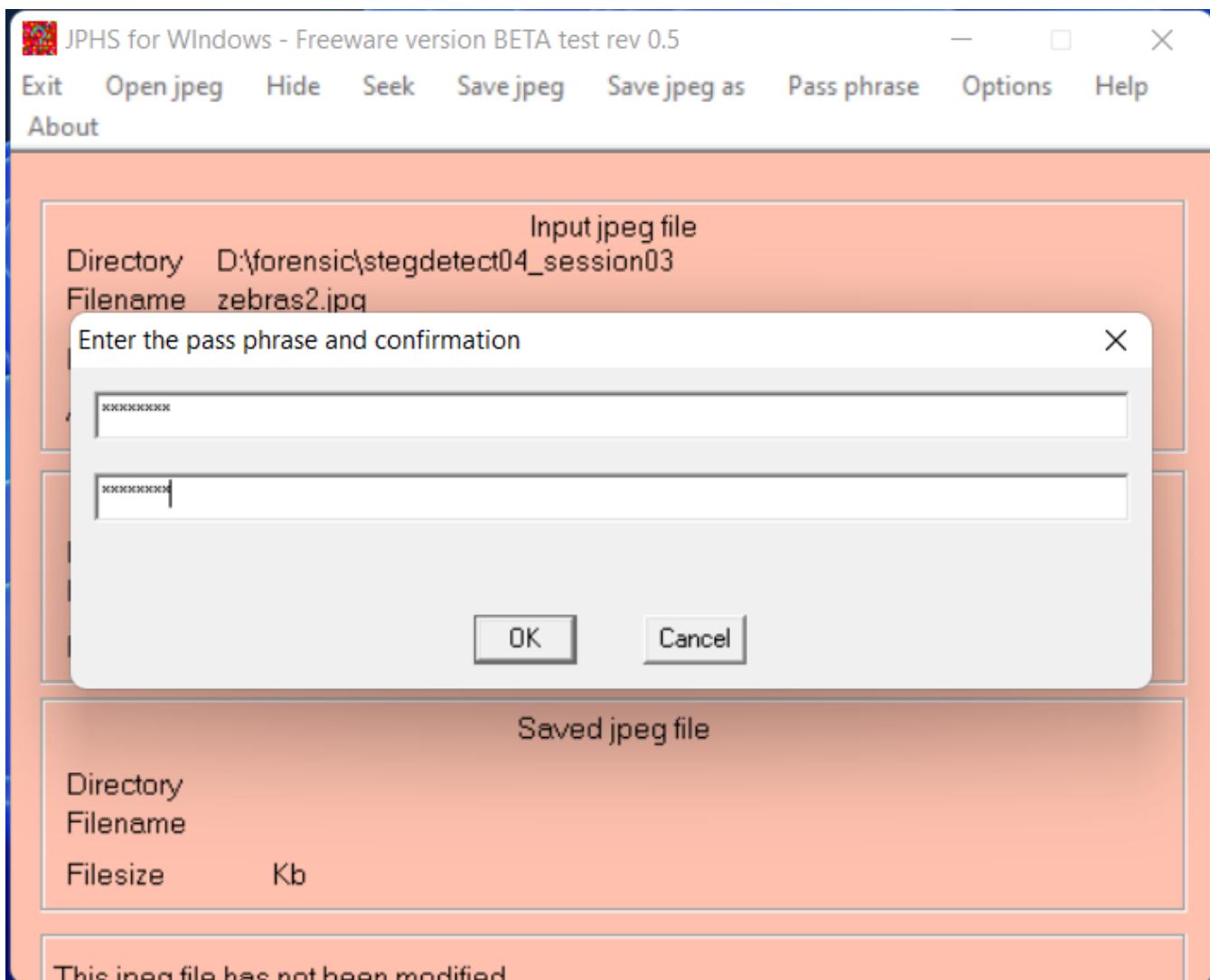
Sau đó lại mở tool Jshswin.exe để xem file được ẩn giấu

## Thi thực hành cuối kì



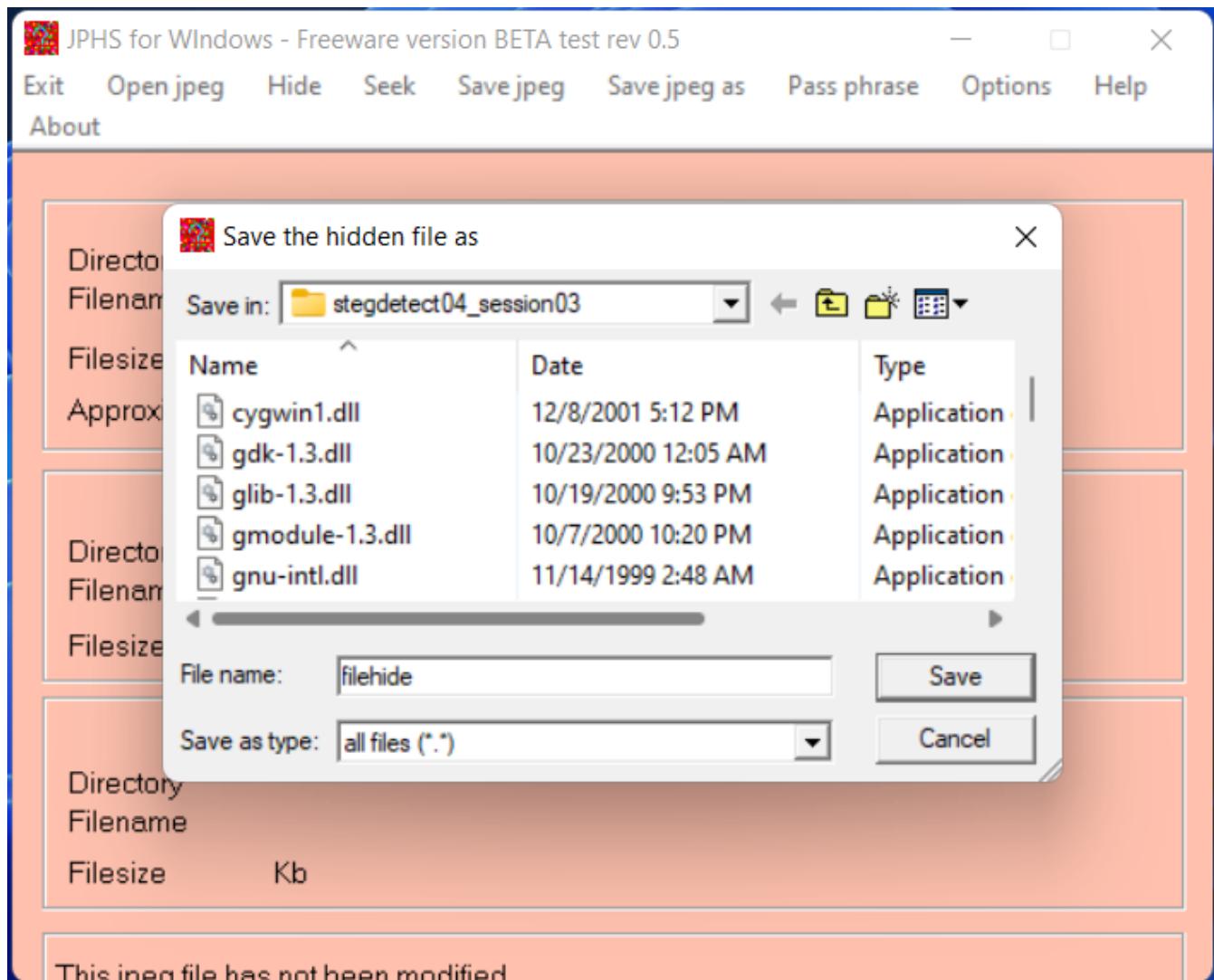
Vào seek nhập mật khẩu là together.

## Thi thực hành cuối kì



Khi nhập mật khẩu thành công thì chương trình yêu cầu chúng ta lưu file ẩn giấu lại.

## Thi thực hành cuối kì



Vì em không biết nó là đuôi gì nên cứ lưu đại là filehide thử. Lần đầu lưu file thì nó sẽ bị lỗi vì thiếu cái tool GhostPDL này.

## Ghostscript/GhostPDL 9.56.1

Latest

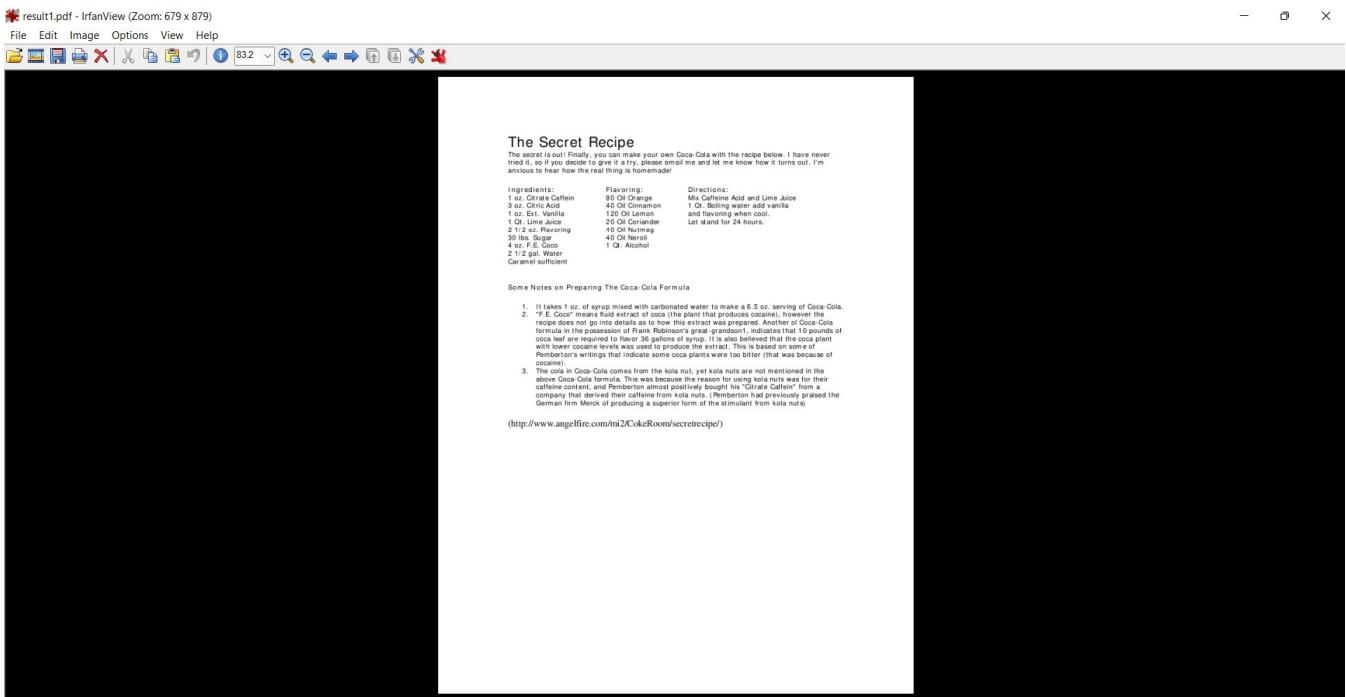
Important: This release includes the new PDF interpreter (implemented in C rather than PostScript). It is both integrated into Ghostscript (now ENABLED by default), and available as a standalone, PDF only, binary. See <https://ghostscript.com/pdfi.html> for more details.

This also bundles the latest zlib (1.2.12) which addresses a security issue ([CVE-2018-25032](https://ghostscript.com/cve-2018-25032))

(2022/04/12) The Windows installers have been updated to also install the (now required) VisualC++ Redistributable package. *No changes to the Ghostscript sources, etc.*

Sau khi cài tool này thì em đã xem được file ẩn giấu như thế này

## Thi thực hành cuối kì



Nó có dạng giống như file pdf và took GhostPDL này cũng là tool để biên dịch file pdf.

### Kịch bản 02.Ẩn giấu dữ liệu bằng công cụ Our Secret

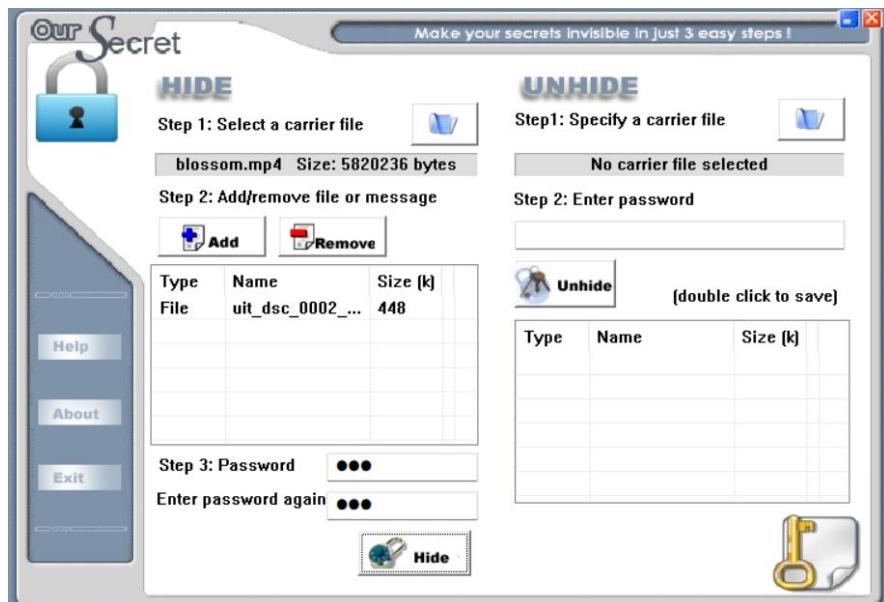
- Tài nguyên: uit\_dsc\_0002\_1.jpg, blossom.mp4
- Phần mềm Our Secret: có thể tải tại liên kết sau:

<http://steganography.findmysoft.com/>

- Cài đặt phần mềm, sau đó giấu ảnh uit\_dsc\_0002\_1.jpg vào tập tin mp4. Đặt mật khẩu trong quá trình giấu tin là "E81". Nhận xét về sự thay đổi của video (thời gian, dung lượng, chất lượng) khi thêm ảnh vào đoạn phim blossom.mp4.
- Giải mã thông tin giấu trong đoạn phim blossom.mp4. Nhận xét về nội dung file giải mã được với file ban đầu (file/thông tin được chọn để giấu)

Mở Our Secret, “nhét” file uit\_dsc\_0002\_1.jpg vào video blossom.mp4.

## Thi thực hành cuối kì



Đặt mật khẩu là “E81” -> Hide -> Save tên file **blossomsecret.mp4**. Ta thấy 2 file có kích thước khác nhau, file sau khi được chèn vào có kích thước lớn hơn file gốc . Nhưng kích thước lại bé hơn một chút so với 2 file cộng lại ( $6130 < 5864 + 449 = 6133$ )

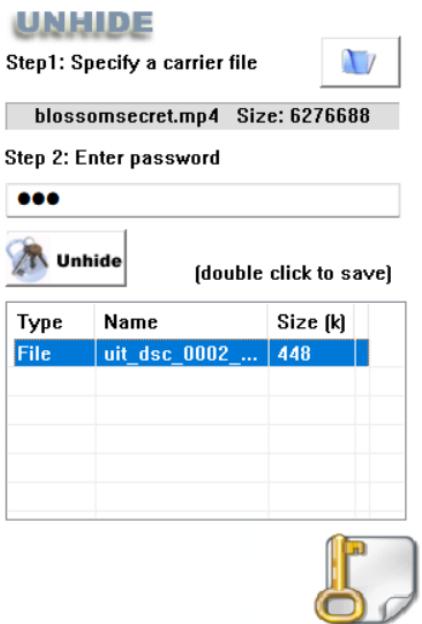
blossom.mp4	4/26/2022 1:45 AM	MP4 File	5,684 KB
blossomsecret.mp4	4/26/2022 9:01 AM	MP4 File	6,130 KB
uit_dsc_0002_1.jpg	4/26/2022 1:45 AM	JPG File	449 KB

Xét về độ dài video và chất lượng (độ phân giải) dường như giống nhau hoàn toàn

<b>General</b>	<b>General</b>
<b>Video</b>	<b>Video</b>
Length 00:02:09	Length 00:02:09
Frame width 852	Frame width 852
Frame height 480	Frame height 480
Data rate 305kbps	Data rate 305kbps
Total bitrate 353kbps	Total bitrate 353kbps
Frame rate 25.00 frames/second	Frame rate 25.00 frames/second
<b>Audio</b>	<b>Audio</b>
Bit rate 47kbps	Bit rate 47kbps
Channels 2 (stereo)	Channels 2 (stereo)
Audio sample rate 22.050 kHz	Audio sample rate 22.050 kHz
<b>Media</b>	<b>Media</b>
Contributing artists	Contributing artists
Year	Year
Genre	Genre
<b>Origin</b>	<b>Origin</b>
Directors	Directors
Producers	Producers
<a href="#">Remove Properties and Personal Information</a>	
<a href="#">OK</a>	<a href="#">OK</a>
<a href="#">Cancel</a>	<a href="#">Cancel</a>
<a href="#">Apply</a>	<a href="#">Apply</a>

## Thi thực hành cuối kì

Chiết xuất file được chèn vào bằng “Unhide”



Double click vào file được tô xanh. Lưu lại, so với file gốc thấy cũng không có khác biệt gì:

blossom.mp4	4/26/2022 1:45 AM	MP4 File	5,684 KB
blossomsecret.mp4	4/26/2022 9:01 AM	MP4 File	6,130 KB
uit_dsc_0002_1.jpg	4/26/2022 1:45 AM	JPG File	449 KB
uit_dsc_0002_1_reveal.jpg	4/25/2022 6:45 PM	JPG File	449 KB

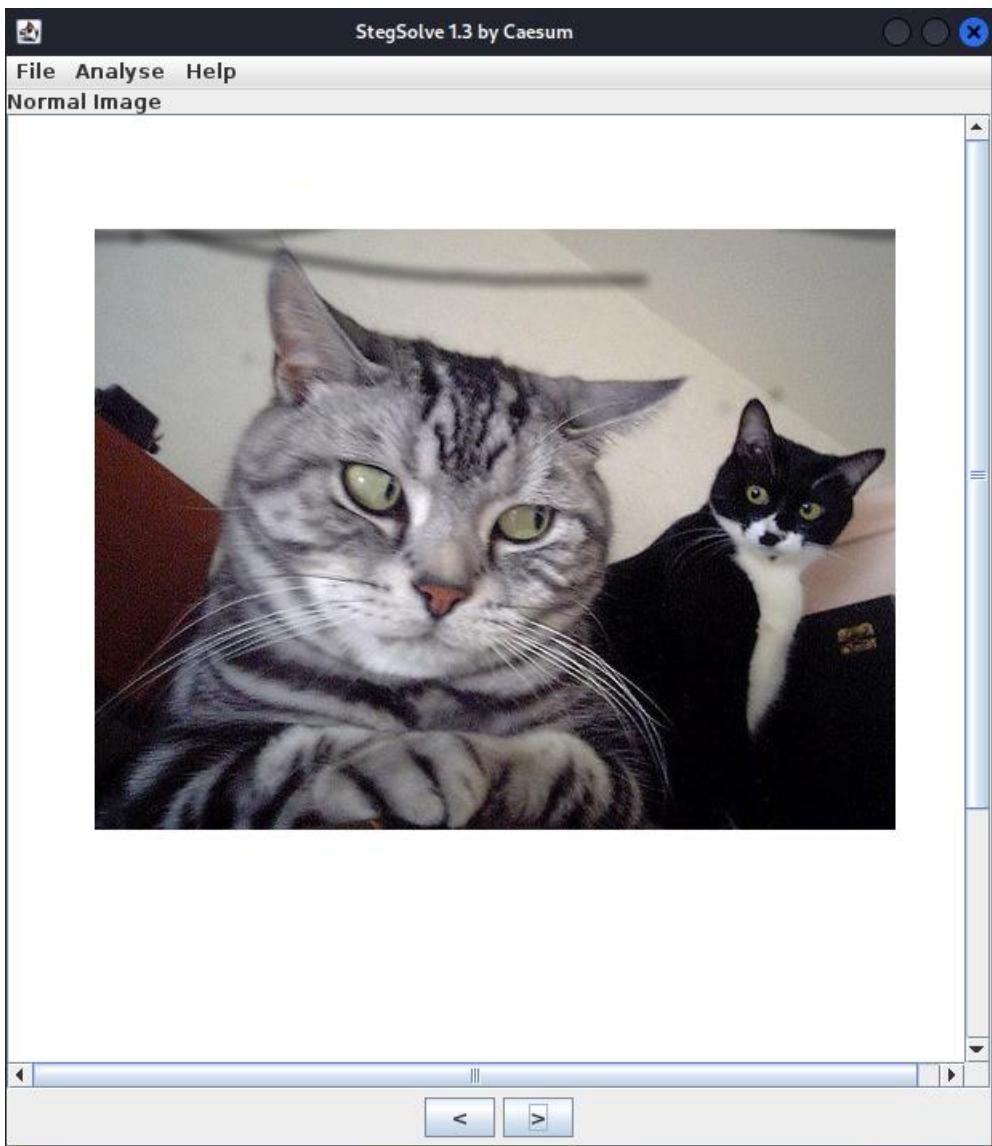
### Kịch bản 03. Điều tra thông tin được ẩn giấu

- Tài nguyên: kb03-suspicion.png
- Mô tả: Bức ảnh scan này đã được phục hồi từ các tập tin của một cựu nhân viên của Hiệp hội Ngờ ngǎn Miêu Quốc. Nhân viên điều tra cần phải tìm ra số sê-ri của máy in này để có thể xác định vị trí của thiết bị. Tìm số sê-ri của máy in.

Đáp án:

Mở stegsolve và load ảnh lên:

## Thi thực hành cuối kì

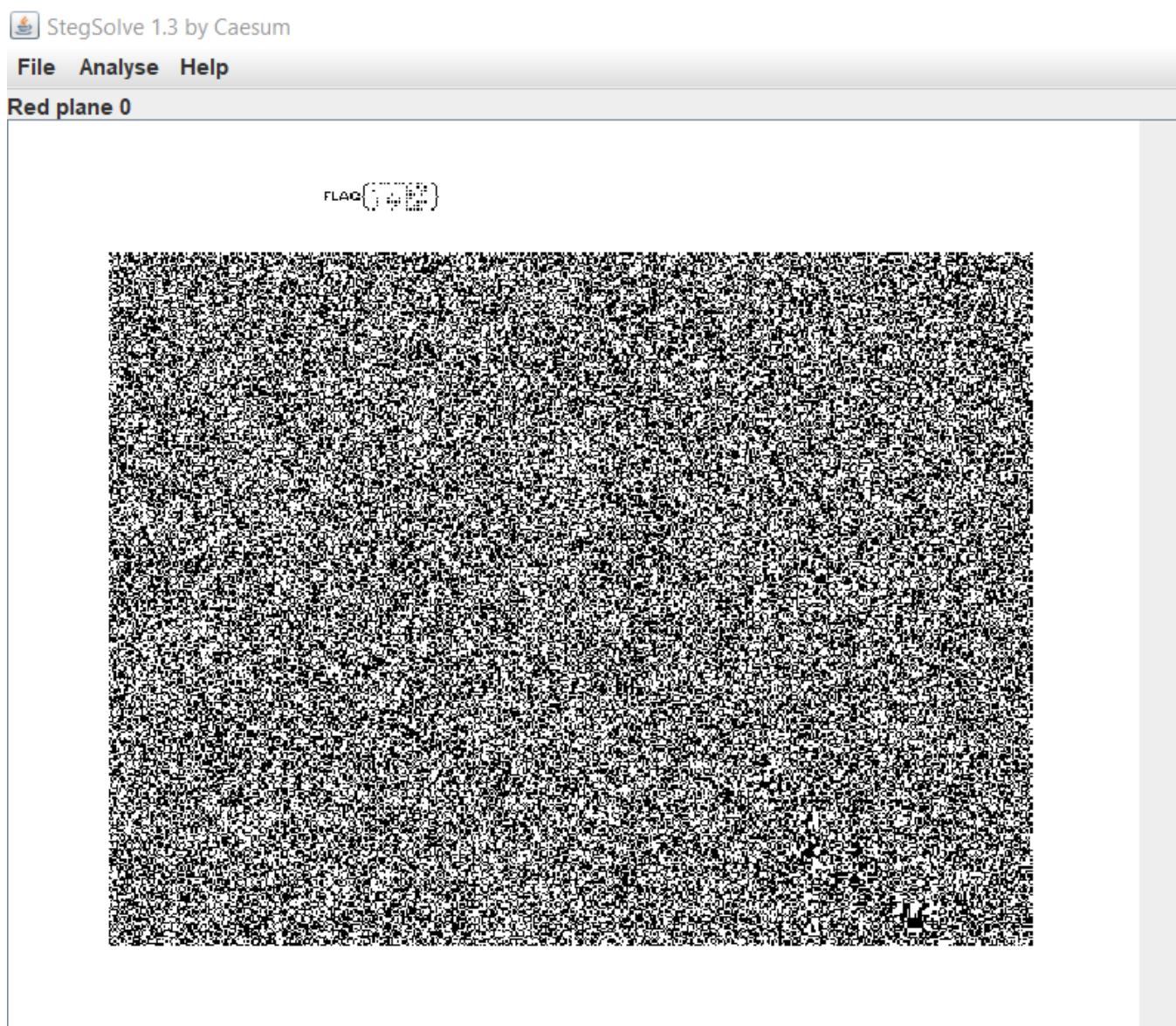


Xem dưới dạng gam màu plane khác:



Cho đến “Red plane 0” thì thấy có flag:

## Thi thực hành cuối kì



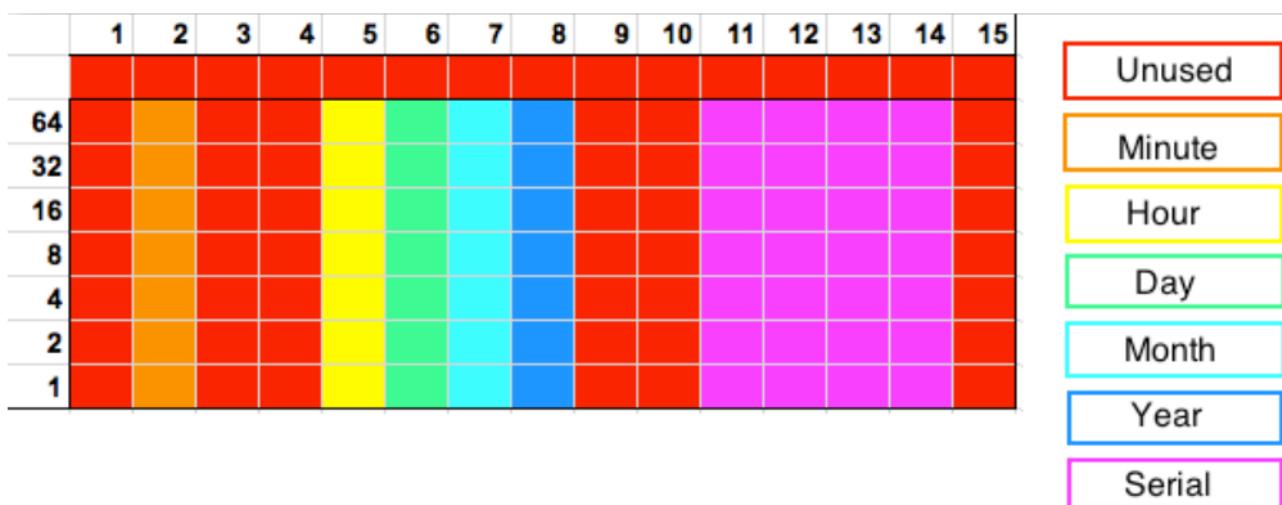
Mở file ảnh kb03-suspicion.png bằng tool StegSolve.jar và chuyển sang chế độ strings thì được flag như hình.

FLAG{0x48454C4F}

## Thi thực hành cuối kì

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
64															
32															
16															
8															
4															
2															
1															

Mình đã điền đầy đủ các chấm trên hình vào bảng theo format của máy in.



Theo form của máy in thì chỉ có các cột ở dòng 11 12 13 14 là số seri của máy in.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
64															
32															
16															
8															
4															
2															
1															
	33	14	0	0	12	21	6	12	0	127	57	19	71	53	100

Như vậy số seri là **53711957**

## Kịch bản 04. Điều tra thông tin được ẩn giấu

- Tài nguyên: star-wars.jpg
- Yêu cầu - Gợi ý: Bức ảnh được nhân viên điều tra tìm thấy trong một máy tính của một nghi phạm có sở thích xem ảnh của họa sĩ John Bramblitt. Tìm thông điệp được ẩn giấu, biết rằng thông điệp bắt đầu bằng "become".

Đáp án:

Search các tool cho CTF steganography thì ra repo này mình thấy cũng khá hay [DominicBreuker/stego-toolkit: Collection of steganography tools - helps with CTF challenges \(github.com\)](https://DominicBreuker/stego-toolkit: Collection of steganography tools - helps with CTF challenges (github.com))

Với việc điều tra về mảng steganography, repo có gợi ý mình dùng các tool cho bước đầu . Mình sử dụng tûm lum tool foremost, file, binwalk, exiftool, ... để kiểm tra sơ bộ thông tin file và metadata nhưng không có gì đặc sắc.

Trong repo trên có đề xuất dùng steghide, mình thử dùng tool thì bị bắt nhập mật khẩu. Vậy có thể mật khẩu được lưu trong file này chăng?

Mình xem thử nội dung file dưới dạng hex để điều tra dữ liệu ẩn xem thử như thế nào, dùng bless để kiểm tra:

Address	Hex	ASCII
00004dd6	14 51 45 00 14 51 45 00 2D 28 A6 D2 E6 80 16 94 1A 6E	.QE..QE.- (.....n
00004de8	68 A7 71 58 7E 68 CD 37 34 66 9D C5 61 D9 A2 9B 40 34	h.qX~h.74f..a...@4
00004dfa	EE 16 1B 45 14 56 65 85 14 51 40 05 14 51 40 05 14 52	...E.Ve..Q@..Q@..R
00004e0c	50 02 D2 51 45 00 19 A2 92 8A 06 14 94 51 48 02 8A 4A	P..QE.....QH..J
00004e1e	28 00 A2 8A 4A 00 28 A2 8A 00 28 14 94 50 03 B3 4E 14	(...J. ....(..P..N.
00004e30	D1 4B 9A 00 75 2E 29 05 2E 69 88 4A 4A 53 4D 34 0C 43	.K..u.)...i.JJSM4.C
00004e42	49 4B 49 48 05 CD 14 94 50 03 A9 28 CD 19 A0 02 92 8A	IKIH....P...(.....
00004e54	28 00 A2 8A 4A 00 28 A2 8A 00 28 A2 8A 00 28 A2 8A 00	(...J. ....(....(...
00004e66	28 A2 8A 00 28 A2 8A 00 28 A2 8A 00 29 73 49 45 00 2D	(...(...(..)sIE.-
00004e78	25 14 50 01 45 14 50 01 4B 49 45 00 2D 28 A4 A5 A0 05	%..P.E.P.KIE.-(...
00004e8a	A7 0A 6D 2D 00 3A 8A 4C D1 4C 91 68 A2 8A 06 14 51 45	..m-..L.L.h...QE
00004e9c	00 14 51 45 00 14 51 45 00 14 51 45 00 14 51 45 00 14	..QE..QE..QE..QE..
00004eae	51 45 00 14 51 45 30 0A 5C D2 51 40 1F FF D9 31 30 30	QE..QE0.\.Q@...100
00004ec0	31 31 30 31 30 31 30 31 30 31 30 31 30 31 31 31 31	110101010101010111
00004ed2	30 31 30 31 30 30 31 31 30 31 30 31 30 31 30 31 31	010100110101010101
00004ee4	31 31 30 31 30 31 30 31 30 31 31 31 31 30 0A	110101010011110.

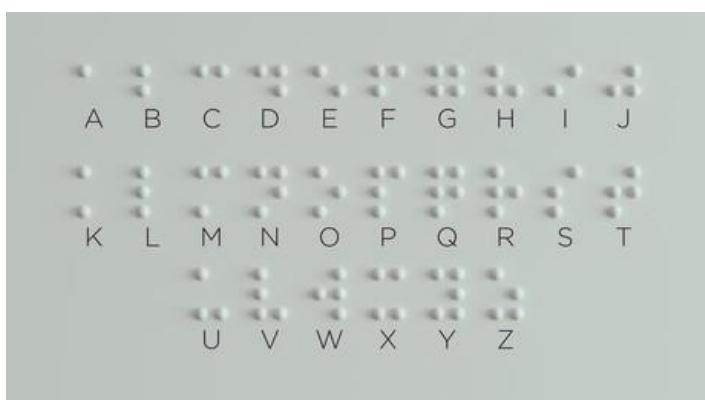
Ta thấy các byte cuối của file có gì đó bất thường, nó là một mã nhị phân. Trích xuất ra và phân tích:

## Thi thực hành cuối kì

```
(virus@virus)@[~]
$ python3
Python 3.9.12 (main, Mar 24 2022, 13:02:21)
[GCC 11.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information
.
>>> len("1001101010101010101110101001101010101110101010011110")
54
```

Length của chuỗi nhị phân này là 54. Ban đầu mình định decode theo ASCII nhưng mà ký tự ASCII là 8 bits ứng với 1 bytes/1 ký tự. Theo gợi ý, đề bài có cung cấp thông tin về một họa sĩ khuyết tật mắt (mù lòe) tên là **John Bramblitt**. Điều này cho hint rằng ông cũng tổng thông điệp vào bằng mã cho người khiếm thị. Độ dài 54 này chia hết cho 6 nên mình cũng nghĩ đến mã Braille dành cho người khiếm thị là hợp lý

Thử decode ra theo bảng mã minh tra được dưới đây:



Ký tự được tra từ mã nhị phân theo nguyên tắc một hình chữ nhật  $3 \times 2$ , từ trên xuống dưới và từ trái qua phải. Ví dụ .. được xem là điểm 1-4 tức là 100100 và là ký tự c. Ta có chuỗi trích ra được từ bên trên như sau:

10011010101010101110101001101010101110101010011110

Phân cách mỗi 6 bits cho đẹp:

100110 101010 101010 111010 100110 101010 101110 101010 011110

Decode Braille theo quy tắc và bảng bên trên ta có:

d o o r d o n o t

Mật khẩu là **doordonot**

Giờ có password rồi, quay lại việc chạy lệnh steghide:

```
steghide extract -sf star-wars.jpg -p doordonot -xf output.txt
wrote extracted data to "output.txt"
```

## Thi thực hành cuối kì

```
(virus㉿virus)-[~/Desktop]
$ steghide extract -sf star-wars.jpg -p doordonot -xf output.txt
wrote extracted data to "output.txt".
```

Output được lưu vào **output.txt**. Đọc file output:

```
(virus㉿virus)-[~/Desktop]
$ cat output.txt
YmVjb21lYWplZGltYXN0ZXJ5b3V3aWxs
```

Ta thấy nội dung bị mã hóa **base64**. Thực hiện decode:

```
(virus㉿virus)-[~/Desktop]
$ cat output.txt | base64 -d
becomeajedimasteryouwill
```

Thông điệp ẩn: **becomeajedimasteryouwill**

### Kịch bản 05. Thực hiện phân tích, tìm thông tin ẩn giấu trong ảnh

- Tài nguyên thực hiện: qn001.jpg
- Yêu cầu – Gợi ý: Tìm thông điệp (flag) được ẩn giấu. Thông tin flag liên quan đến Đội tuyển bóng đá nam Việt Nam.

*Gợi ý:*

Ta thấy file ảnh nhưng lại có dung lượng khá lớn, ≈ 7MB

qn001.jpg	4/29/2022 9:20 AM	JPG File	6,949 KB
-----------	-------------------	----------	----------

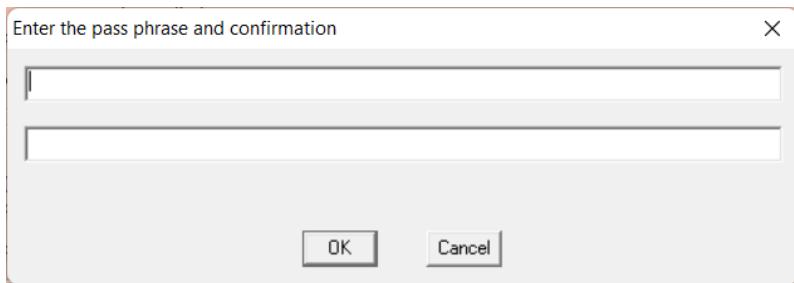
Vậy có thể trong file ảnh embedded một file khác. Ta có thể dùng tool stegbreak để tìm file được nhúng vào theo wordlist [-f] và rule [-r] mặc định được cung cấp tại [repo chính chủ của stegdetect](#)

Trong repo có sẵn link trang chủ để download nhưng đã chết: [s-fiebig/stegdetect-stegbreak: Stegdetect and stegbreak \(github.com\)](#) => [Wayback Machine](#)

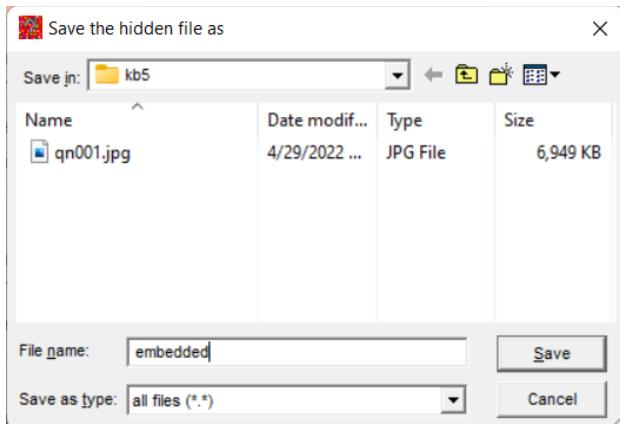
```
D:\KHANG\Setup\Stegdetect\stegdetect>stegbreak.exe -r rules.ini -l rockyou.txt qn001.jpg
stegbreak.exe: unknown option -- l
Usage: stegbreak.exe [-V] [-r <rules>] [-f <wordlist>] [-t <schemes>] file.jpg ...
D:\KHANG\Setup\Stegdetect\stegdetect>stegbreak.exe -r rules.ini -f rockyou.txt qn001.jpg
Corrupt JPEG data: bad Huffman code
Loaded 1 files ...
qn001.jpg : jphide[v5]()
Processed 1 files, found 1 embeddings.
Time: 1 seconds: Cracks: 4752,    4752.0 c/s
```

Ta thấy có một file được ẩn trong nội dung file ảnh. Tuy nhiên tool này không thể extract được file ra. Tiếp đến ta dùng tool **JPHS** (download [tại đây](#)) để trích xuất file đó ra. Chọn *Seek* -> Phần *passphrase* ta để trống:

## Thi thực hành cuối kì



Sau đó chọn nơi lưu file được extract:



Ta kiểm tra định dạng file trên Linux bằng command “**file**”:

```
(virus㉿virus) - [~/Desktop]
$ file embedded
embedded: Microsoft Word 2007+
```

Như ảnh trên, đây là file của **Microsoft Word 2007**, ta đổi đuôi thành **.doc** và mở lên để xem nội dung.

**Tuyển Việt Nam luôn biết làm chủ thế trận**

Trang bóng đá *Okezone Bola* miêu tả chi tiết về trận đấu diễn ra trên sân Hàng Đẫy: "Đội tuyển Việt Nam nhập cuộc khá tốt với chủ đích tấn công trực diện. Kết quả là đội chủ nhà hoàn toàn nắm giữ thế trận khi Campuchia phụ thuộc nhiều vào những pha phản công".

"Mặc dù làm chủ trận đấu, trên thực tế, đoàn quân của HLV Park Hang-seo lại gặp khó khăn khi không thể khoan thủng hàng phòng ngự đối phương ở đầu hiệp 1. Phải đến phút thứ 39, nỗ lực của Việt Nam mới được đền đáp khi Tiến Linh tận dụng lối thế từ đường chuyền của Trọng Hoàng để xâm nhập vào vòng cấm và đánh đầu ghi bàn", tờ báo của Indonesia bình luận.

Những bàn thắng tiếp theo, đến như một lẽ tất nhiên khi đội chủ nhà cởi bỏ được áp lực ghi bàn và thoải mái dẫn dắt trận đấu. "Các học trò của ông Park chỉ mất thêm 3 phút để Quang Hải ghi tên mình lên bảng tỷ số sau khi tận dụng pha kiến thiết của Nguyễn Phong Hồng Duy", tờ *Bola Sport* viết.

Trang *Straits Times* của Singapore đặc biệt ấn tượng với pha làm bàn của Văn Đức khi miêu tả đây là "cú sút đầy lạnh lùng quyết đoán từ một pha xoay người rất nhanh, nồng ty sốt lên 3-0 để ấn định một chiến thắng không có gì phải bàn cãi cho Việt Nam".

Coi đây là màn trình diễn tốt nhất của tuyển Việt Nam từ đầu chiến dịch AFF Cup 2018 đến nay, *Fox Sports Asia* nhận định: "Thầy trò Park Hang-seo đã chạm đến vòng knockout bằng chiến thắng hủy diệt 3-0 trước Campuchia. Thắng lợi này không chỉ đảm bảo cho họ một vé vào bán kết, mà với thất bại của Myanmar trước Malaysia, nhà vô địch Đông Nam Á năm 2008 đã kết thúc vòng bảng với tư cách là đội đứng đầu bảng A".

Trang *FourFourTwo* lại đánh giá cao khả năng xoay tua nhân sự của thuyền trưởng người Hàn Quốc: "Đội tuyển Việt Nam dễ dàng vượt qua Campuchia để dẫn đầu bảng A. Trong bối cảnh Campuchia đã bị loại, HLV Park cắt những trụ cột như Xuân Trường, Anh Đức và Công Phượng để tạo điều kiện cho nhiều cầu thủ khác có cơ hội ra sân gây ấn tượng".

*Reuters* gọi đây là "thắng lợi như thường lệ của đội tuyển Việt Nam trước Campuchia. Malaysia cũng đã đánh bại Myanmar để cùng Việt Nam - ứng cử viên sáng giá cho chức vô địch - tiến vào bán kết ASEAN".

Ta thấy nội dung liên quan đến bóng đá chủ nhà VN (đúng hướng), tuy nhiên lại không liên quan gì đến flag. Thủ tiếp tục extract file ẩn trong file doc bằng **binwalk**

\*NOTE: Mình install bên Kali Linux cũ bị lỗi nên chuyển sang Ubuntu

Đẩy file bên Window sang máy Ubuntu:

```
| scp embedded.doc virus@192.168.153.128:/home/virus/Desktop
```

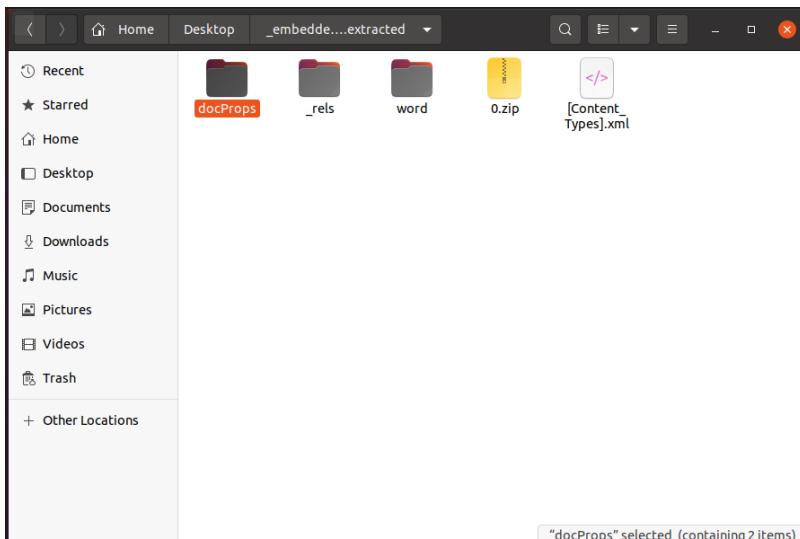
```
D:\KHANG\DocumentForStudy\Digital forensic\Lab\Lab3\resources\kb5>scp embedded.doc virus@192.168.153.128:/home/virus/Desktop
virus@192.168.153.128's password:
embedded.doc                                         100%  190KB   9.4MB/s   00:00
```

Dùng **binwalk**

```
| binwalk -e embedded.doc
```

DECIMAL	HEXADECIMAL	DESCRIPTION
0	0x0	Zip archive data, at least v2.0 to extract, compressed size: 359, uncompressed size: 1363, name: [Content_Types].xml
928	0x3A0	Zip archive data, at least v2.0 to extract, compressed size: 239, uncompressed size: 590, name: _rels/.rels
1728	0x6C0	Zip archive data, at least v2.0 to extract, compressed size: 3915, uncompressed size: 19670, name: word/document.xml
5690	0x163A	Zip archive data, at least v2.0 to extract, compressed size: 264, uncompressed size: 949, name: word/_rels/document.xml.rels
6276	0x1884	Zip archive data, at least v1.0 to extract, compressed size: 17893, uncompressed size: 178935, name: word/media/image1.jpg
185262	0x2D3AE	Zip archive data, at least v2.0 to extract, compressed size: 1538, uncompressed size: 7076, name: word/theme/theme1.xml
186851	0x2D9E3	Zip archive data, at least v2.0 to extract, compressed size: 1118, uncompressed size: 3160, name: word/settings.xml
188016	0x2DE70	Zip archive data, at least v2.0 to extract, compressed size: 3267, uncompressed size: 31584, name: word/styles.xml
191328	0x2EB60	Zip archive data, at least v2.0 to extract, compressed size: 471, uncompressed size: 2670, name: word/webSettings.xml
191849	0x2ED69	Zip archive data, at least v2.0 to extract, compressed size: 576, uncompressed size: 1968, name: word/fontTable.xml
192473	0x2EFD9	Zip archive data, at least v2.0 to extract, compressed size: 386, uncompressed size: 747, name: docProps/core.xml
193170	0x2F292	Zip archive data, at least v2.0 to extract, compressed size: 479, uncompressed size: 992, name: docProps/app.xml
194731	0x2F8AB	End of Zip archive, footer length: 22

Trích ra được “một đống” file nhìn có vẻ có giá trị



## **Thi thực hành cuối kì**

Nhưng mò một hồi thấy không có gì đặc biệt lắm, file nào cũng có định dạng và nội dung hợp lệ, ngoại trừ 1 file có đường dẫn `_embedded.doc.extracted/word/document.xml` có chứa một số ký tự lạ trong nội dung file

Mở bằng trình duyệt để *render định dạng XML*, cho ra output đẹp hơn.

Ta thấy ở đây có 2 chỗ chứa thông điệp lạ. Giờ ta cần tìm thuật toán để decode, để cho nhanh ta dùng [tool này](#) để detect nhanh loại cipher phổ biến hiện nay. Copy và paste một trong hai đoạn trên ta sẽ được.

## **Thi thực hành cuối kì**

Khả năng cao là ngôn ngữ ***Brainfuck***. Thủ đưa toàn bộ payload có chứa “brainfuck” vào và decode:

Flag: Forensics05@UIT{Vietnam-win-Cambodia}

## Thi thực hành cuối kì

### Kịch bản 06. Thực hiện phân tích:

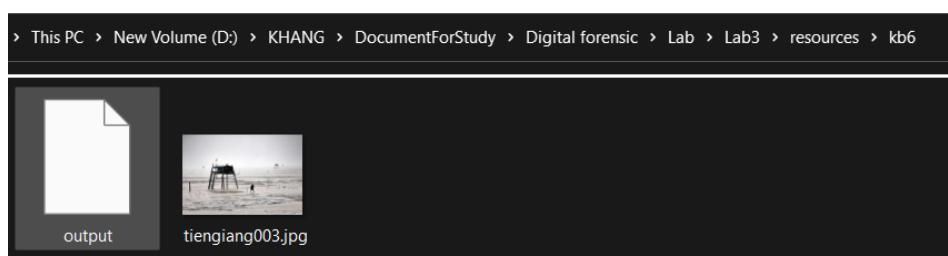
- Tài nguyên: tiengiang003.jpg
- Yêu cầu – Gợi ý: Tìm thông điệp (flag) được ẩn giấu. Thuật toán dùng tìm ra flag liên quan đến việc thay thế các kí tự trong chuỗi ban đầu thành chuỗi chỉ gồm 2 kí tự a và b.

*Đáp án:*

Thao tác điều tra giống câu trên, sử dụng stegbreak để kiểm tra :

```
D:\KHANG\Setup\Stegdetect\stegdetect>stegbreak.exe -r rules.ini -f rockyou.txt tiengiang003.jpg
Loaded 1 files ...
tiengiang003.jpg : jphide[v5]()
Processed 1 files, found 1 embeddings.
Time: 1 seconds: Cracks: 4752,    4752.0 c/s
```

Ta thấy có một file ẩn, dùng JHPS để trích xuất ra file và lưu là **output**:



Đưa qua máy ảo Ubuntu

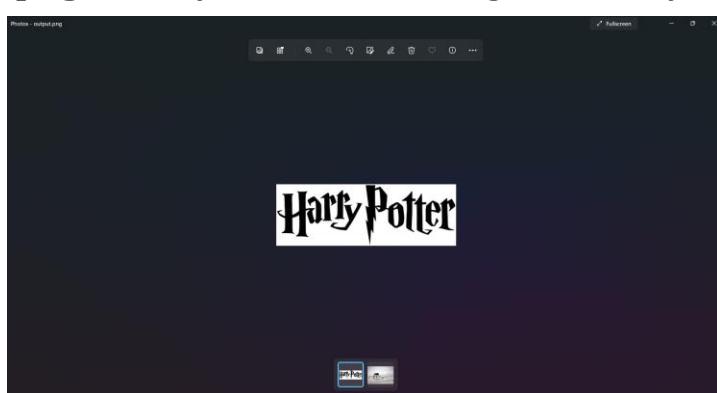
scp output virus@192.168.153.128:/home/virus/Desktop

```
D:\KHANG\DocumentForStudy\Digital forensic\Lab\Lab3\resources\kb6>scp output virus@192.168.153.128:/home/virus/Desktop
virus@192.168.153.128's password:
          100% 6157      2.9MB/s   00:00
```

Và kiểm tra định dạng file:

```
virus@ubuntu:~/Desktop$ file output
output: PNG image data, 385 x 131, 8-bit colormap, non-interlaced
```

Đặt lại extension là **.png**. Ta thấy ảnh có chứa dòng chữ “Harry Potter”:



Dùng lệnh strings để lấy ký tự đọc được

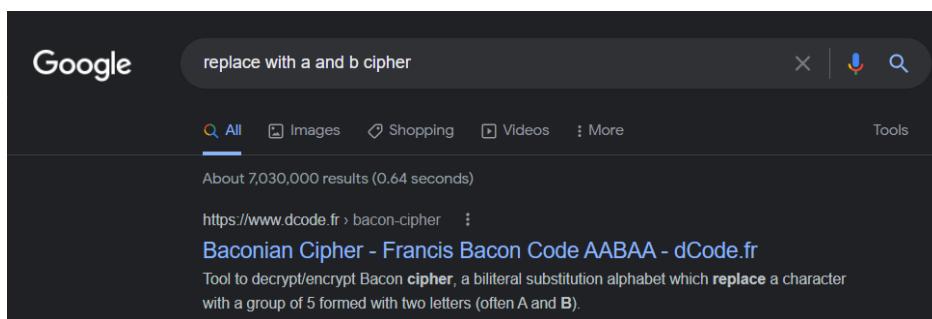
strings output.png

```
virus@ubuntu: ~/Desktop
7jqX
#-6x
Zc1,3
AWWE
@kjhj
uydy$Gm#Lm
i)7n
;DD
^ng L
P[]*s
2~jv
OCQDZ
- g{_PLP
u} Ic
jg3U
c)U>
IEND
where Should one Really look for this flag
virus@ubuntu:~/Desktop$
```

Ta thấy có chuỗi “*where Should one Really look for this flag*” (*Cái nào là nơi thật sự có thể tìm ra flag này*). Theo mình hiểu thì đây là flag, nhưng mình phải tìm “thứ” (thuật toán) làm sao đó để giải mã được message này.

Liên hệ với hint đầu bài, xem thử “việc thay thế các ký tự ban đầu thành 2 ký tự a và b” là loại cipher nào.

Sau một hồi thì từ khóa này cho kết quả khá thỏa đáng



Vào [link đầu tiên](#).

## Thi thực hành cuối kì

Ta thấy đoạn input cho việc mã hóa chỉ gồm các ký tự “A” và “B”. Đại khái **Baconian Cipher** là một loại mã hóa mỗi lần thao tác với 5 ký tự được tra theo bảng sau:

Letter	Code	Binary	Letter	Code	Binary
A	aaaaaa	00000	N	abbaa	01100
B	aaaab	00001	O	abbab	01101
C	aaaba	00010	P	abbba	01110
D	aaabb	00011	Q	abbbb	01111
E	aabaa	00100	R	baaaa	10000
F	aabab	00101	S	baaab	10001
G	aabba	00110	T	baaba	10010
H	aabbb	00111	U, V	baabb	10011
I, J	abaaa	01000	W	babaa	10100
K	abaab	01001	X	babab	10101
L	ababa	01010	Y	babba	10110
M	ababb	01011	Z	babbb	10111

Đồng thời thông điệp “*wherE ShOUld onE ReAllY lOoK fOr tHiS flag*” **chứa 35 ký tự** (chia hết cho 5 – phù hợp với mã hóa nói trên). Vậy quan trọng ở đây là từ chuỗi ban đầu, đưa kỹ thuật **Baconian Cipher** vào đoạn string trên như thế nào? Thì ở đây mình có link tham khảo một số cách ứng dụng loại *mã hóa* này vào steganography

[Steganography: Because Who Doesn't Love Bacon? | Ball in your Court \(craigball.net\)](#)

Theo link trên, ta nhận thấy vì chuỗi theo dạng mã hóa **Baconian Cipher** có dạng nhị phân “A” và “B” cũng như “0” và “1”, nên có nhiều cách để quy chuẩn chuỗi của mình thành dạng mã hóa. Ví dụ một số phương pháp được liệt kê trong link như

- Phân biệt chữ có móc và chữ đơn giản



## Thi thực hành cuối kì

- Hay chữ in đậm và in thường trong văn bản:

I hid my name in this sentence as a series of **bolded** letters.  
 I hid my name in this sentence as *italicized* characters.  
 I hid my name in this sentence as sans serif characters.

Ở đây dễ nhìn nhất thì ta nhận ra “**thuộc tính nhị phân để phân biệt**” trong đoạn chuỗi “*wherE ShOUld onE ReaLly lOoK fOr tHis flag*” là **chữ hoa** và **chữ thường**. Vậy ta giả định **chữ thường** là “A” và **chữ hoa** là “B” (Nếu không được thì đổi lại cho đến khi có decoded message có nghĩa thôi).

Vậy đoạn input message của ta từ thông điệp trên sẽ thành (message ngắn nên khỏi code :v) :

AAAAB BABBA AAB BAABA ABAB ABA ABAA AAAA

(Căn lại cho đẹp mỗi 5 ký tự :> )

Decode ra bằng [tool](#) và kết quả hợp nghĩa nhất là **BYDELTA**



**BACON CIPHER**  
Cryptography > Substitution Cipher > Bacon Cipher

**BACONIAN CIPHER DECODER**

★ BACON CIPHERTEXT  
AAAAB BABBA AAABB AABAA ABABA BAABA AAAAA

► DECRYPT BACON

See also: Uppercase Lowercase – Delastelle Trifid Cipher

**BACON ENCODER**

★ BACON PLAIN TEXT  
dCode Bacon

★ LETTER 1 A  
★ LETTER 2 B  
★ ADD A SPACE SEPARATOR EVERY 5 CHARACTERS

► ENCRYPT

See also: Delastelle Trifid Cipher

Answers to Questions (FAQ)

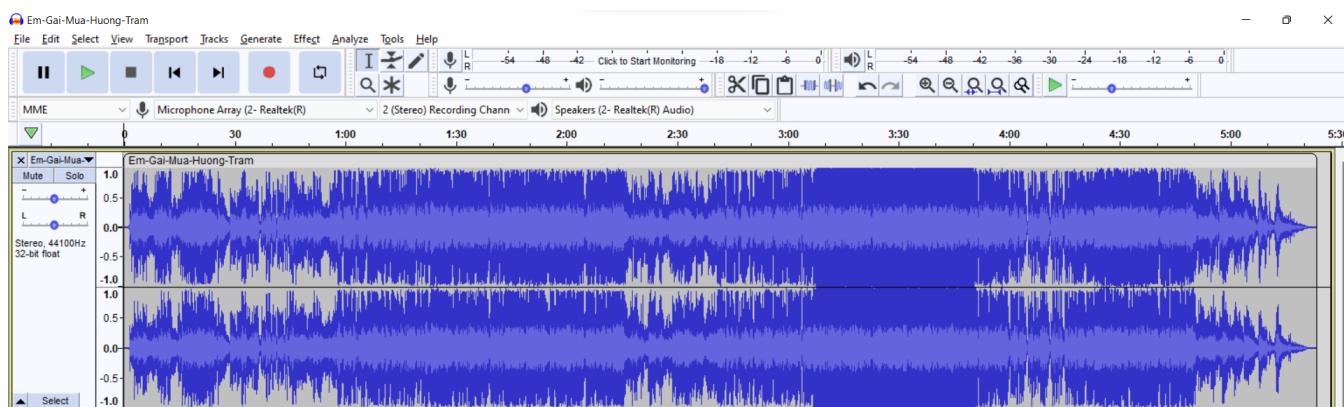
Flag: **BYDELTA**

### Kịch bản 07. Thực hiện phân tích, tìm thông tin ẩn giấu:

- Tài nguyên: kb07-res (Tìm thông tin ẩn giấu trong Em-Gai-Mua-Huong-Tram.mp3, capture-the-flag.png)

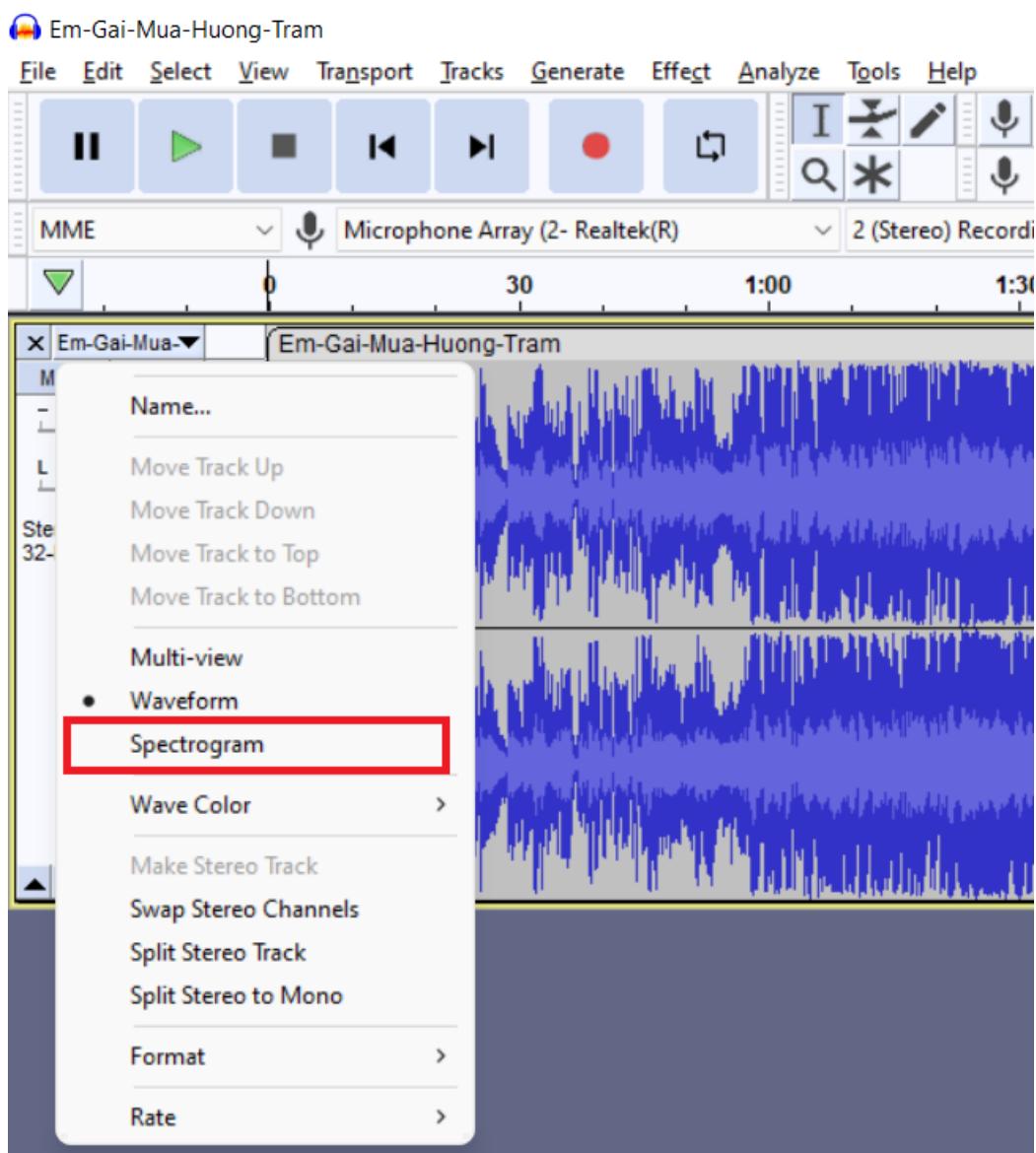
*Đáp án:*

Bài này thực sự khá stuck, hên sao nhỉ [link này](#) giúp mình cho hint xem file âm thanh dưới dạng [Spectrogram](#), cái tính năng này thì trên Audacity có sẵn. Mở file bằng ứng dụng này:

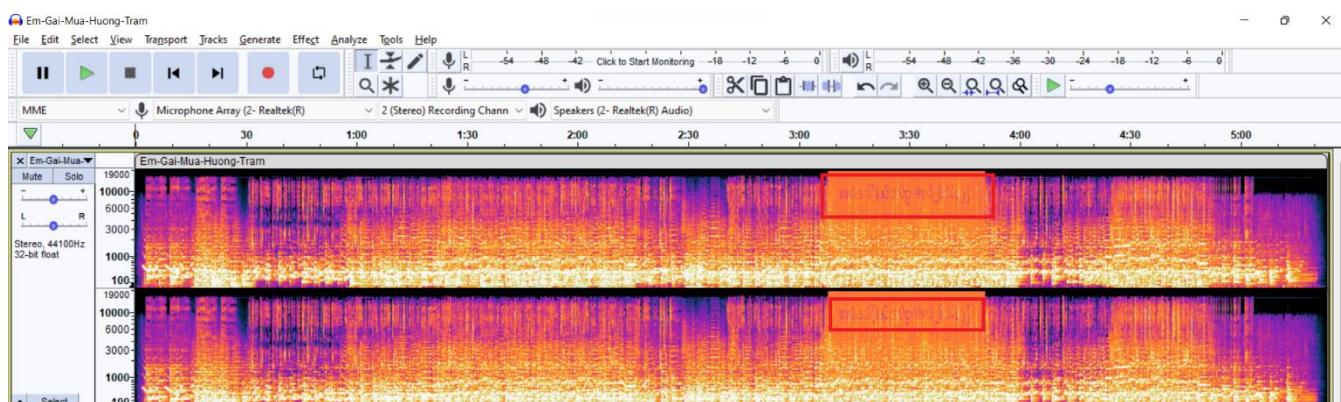


Mặc định file được import vào sẽ được hiển thị dưới dạng Wave Form. Chúng ta có thể chuyển View sang dạng Spectrogram và xem:

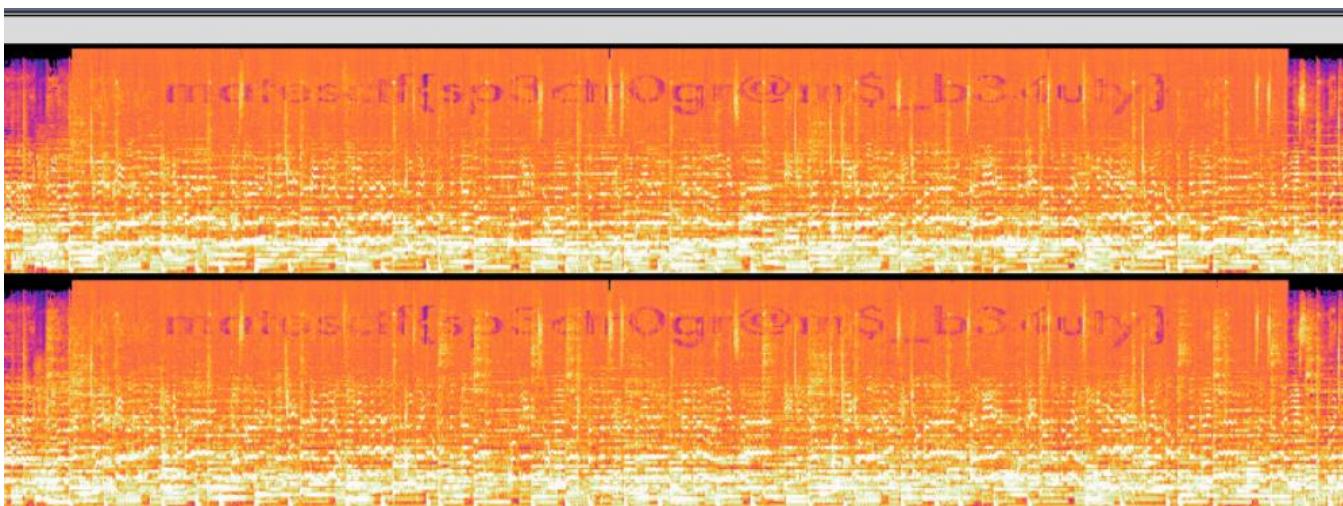
## Thi thực hành cuối kì



Spectrogram:



Ta thấy ở trên có dòng thông điệp gì đó, thử giàn video ra xem thử:



Chữ không rõ lắm, và không có hệ thống flag nên ta chỉ đoán.

Flag: **matesctf{sp3ctr0gr@m\$\_b34uty}**

#### Kịch bản 08. Thực hiện phân tích, tìm thông tin ẩn giấu:

- Tài nguyên: LoveLetter.txt
- Yêu cầu – Gợi ý: Có gì đó đáng ngờ trong bức thư tình mà bạn đang đọc. Nhân viên điều tra cũng nghĩ rằng bức thư tình này chứa một thông điệp bí mật nào đó. Hãy tìm thông điệp được ẩn giấu (flag). Flag có dạng “FLAG-\*”
- Link CTF: <https://ringzer0ctf.com/challenges/215>

Đối với những bài examine nội dung file, mình sẽ dùng HxD để xem raw:

Offset(h)	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	Decoded text
00000000	49 20 77 65 6E 74 A0 74 6F 20 74 68 65 20 70 61	I went to [] the pa
00000010	72 6B 20 74 6F 64 61 79 2C A0 73 61 77 A0 61 20	rk today, saw a
00000020	6C 6F 74 20 6F 66 A0 66 69 73 68 2E 20 46 69 73	lot of fish. Fis
00000030	68 20 61 72 65 A0 63 6F 6F 6C 2C A0 62 75 74 20	h are cool, but
00000040	74 68 65 79 20 61 72 65 6E 27 74 20 6D 79 A0 66	they aren't my f
00000050	61 76 6F 72 69 74 65 20 61 6E 69 6D 61 6C 21 21	avorite animal!!
00000060	20 54 68 65 20 6D 6F 6E 6B 65 79 20 69 73 20 61	The monkey is a
00000070	A0 67 6F 6F 64 20 61 6E 69 6D 61 6C 2C A0 73 6F	good animal, so
00000080	20 69 73 20 74 68 65 20 42 6C 75 65 2D 54 6F 75	is the Blue-Tou
00000090	6E 67 65 64 A0 53 6B 69 6E 6B 2C A0 62 75 74 A0	nged Skink, but
000000A0	49 20 72 61 72 65 6C 79 20 67 65 74 A0 74 6F 20	I rarely get to
000000B0	73 65 65 A0 74 68 6F 73 65 A0 61 74 20 74 68 65	see those at the
000000C0	A0 70 61 72 6B 21 20 41 6C 6C 20 6F 66 A0 74 68	park! All of th
000000D0	69 73 A0 6D 61 6B 65 73 20 6D 65 20 73 61 64 2C	is makes me sad,
000000E0	A0 62 75 74 A0 6A 75 73 74 20 65 6E 63 6F 75 72	but just encour
000000F0	61 67 65 73 A0 6D 65 A0 74 6F 20 74 72 61 76 65	ages me to trave
00000100	6C 20 6D 6F 72 65 2E 20 49 27 6C 6C A0 73 74 61	l more. I'll sta
00000110	72 74 20 72 65 73 65 61 72 63 68 69 6E 67 20 77	rt researching w
00000120	68 65 72 65 20 69 6E A0 74 68 65 A0 77 6F 72 6C	here in the worl
00000130	64 20 49 A0 63 61 6E A0 73 65 65 20 74 68 65 73	d I can see thes
00000140	65 20 61 6E 69 6D 61 6C 73 A0 69 6E A0 74 68 65	e animals in the

Ta để ý file có hơi “bịp” một chút, ký tự khoảng trắng thông thường có hex là 20, ở đây một số ký tự khoảng trắng lại có hex bằng A0.

## **Thi thực hành cuối kì**

Thật sự mình cũng không hiểu sự khác biệt này lắm, tại sao ascii lại để render như vậy. Mình hiểu theo [nguồn này](#). Đại khái 20(hex) là một space bình thường, còn A0(hex) là non-breaking space.

Sau đó mình tra thêm về technique này :

- <https://www.researchgate.net/publication/321937719> Boosting the Capacity of Web based Steganography by Utilizing Html Space Codes A blind Steganography Approach
  - <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.538.781&rep=rep1&type=pdf>

Thì đây là một kỹ thuật được đề xuất để ẩn thông tin từ trang HTML. Bên paper thứ 2 người ta dùng bảng dưới để xử lý theo một quy tắc:

Table 3: Fixed Space Codes

Sr no	Space codes	Bits
1	Space	0000
2	&#9	0001
3	&#XA0	0010
4	&#32	0011
5	&#8194	0100
6	&#160	0101
7	&nbsP	0110
8	&#XA0	0111
9	&#8194	1001
10	&#1v60	1010

Tuy nhiên trong bài này chỉ có 2 giá trị là **space thường** với **non-breaking space** và mình search thì không có kỹ thuật nào đúng chính xác như vậy. Đến đây thì mình đoán:

- space thường mình set là 0 , non-breaking space set 1
  - space thường set 1, non-breaking space set 0

Với trường hợp thứ 2 thì mình ra ngoài phạm vi biểu diễn ký tự ascii -> *không có chuỗi hợp lệ*.

Với trường hợp đầu thì OK

## Code solve python:

```
import binascii

binary= """
with open("LoveLetter.txt","r") as file:
    print(file)
```

## **Thi thực hành cuối kì**

```
for char in file.read():
    if char == ' ':
        binary = binary + "0"
    if ord(char) == 160:
        binary = binary + "1"

print("Binary result: " + binary)

bin_form = '0b' + binary # Convert to binary format
dec_form = int(bin_form, 2) # Prepare for unhexlify
decoded_result = binascii.unhexlify('%x' % dec_form)

print(decoded_result)
```

Ra kết quả:

Flag: FLAG-3b6f70fcf070009561f5276fe98fc9c6

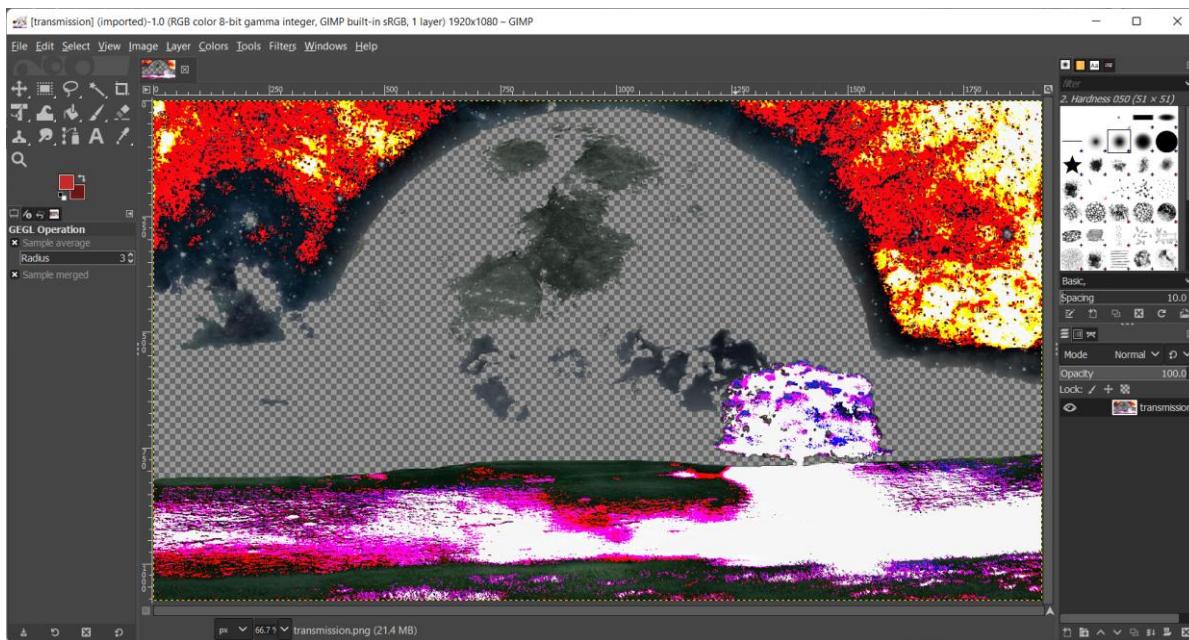
### Kích bản 09. Thực hiện phân tích, tìm thông tin ẩn giấu:

- Tài nguyên: transmission.png
  - Yêu cầu – Gợi ý: Tìm thông điệp được ẩn giấu bằng các công cụ đã học trong buổi này.

### Đáp án:

Mở image bằng Gimp hoặc Photoshop, ta thấy một phần của image bị transparent.

## Thi thực hành cuối kì



Xóa phần này đi, trong kỹ thuật được gọi là **Remove alpha channel/layer**. Dùng thư viện xử lý ảnh **Pillow** module và convert các điểm màu **transparency** thành **RGB**, còn các điểm màu các hiển nhiên đúng format RGB. Code python sau:

```
from PIL import Image
img = Image.open('transmission.png').convert('RGB') # Open file
'transmission.png' and convert to RGB

img.show()
img.save('flag.png')
```

Output:



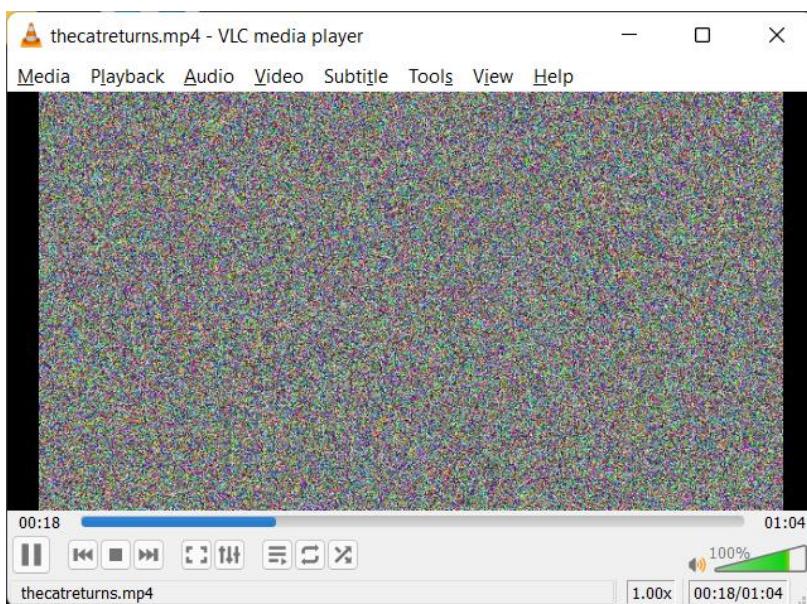
Flag: {Cooper\_Brand}

### Kịch bản 10. Thực hiện phân tích, tìm thông tin ẩn giấu:

- Tài nguyên: thecatreturns.mp4
- Yêu cầu – Gợi ý: Tìm sự khác biệt giữa các khung hình (frame) trong đoạn phim đã cho. Chuyển nội dung đoạn phim thành các khung hình để phân tích. Công cụ ffmpeg, ImageJ.

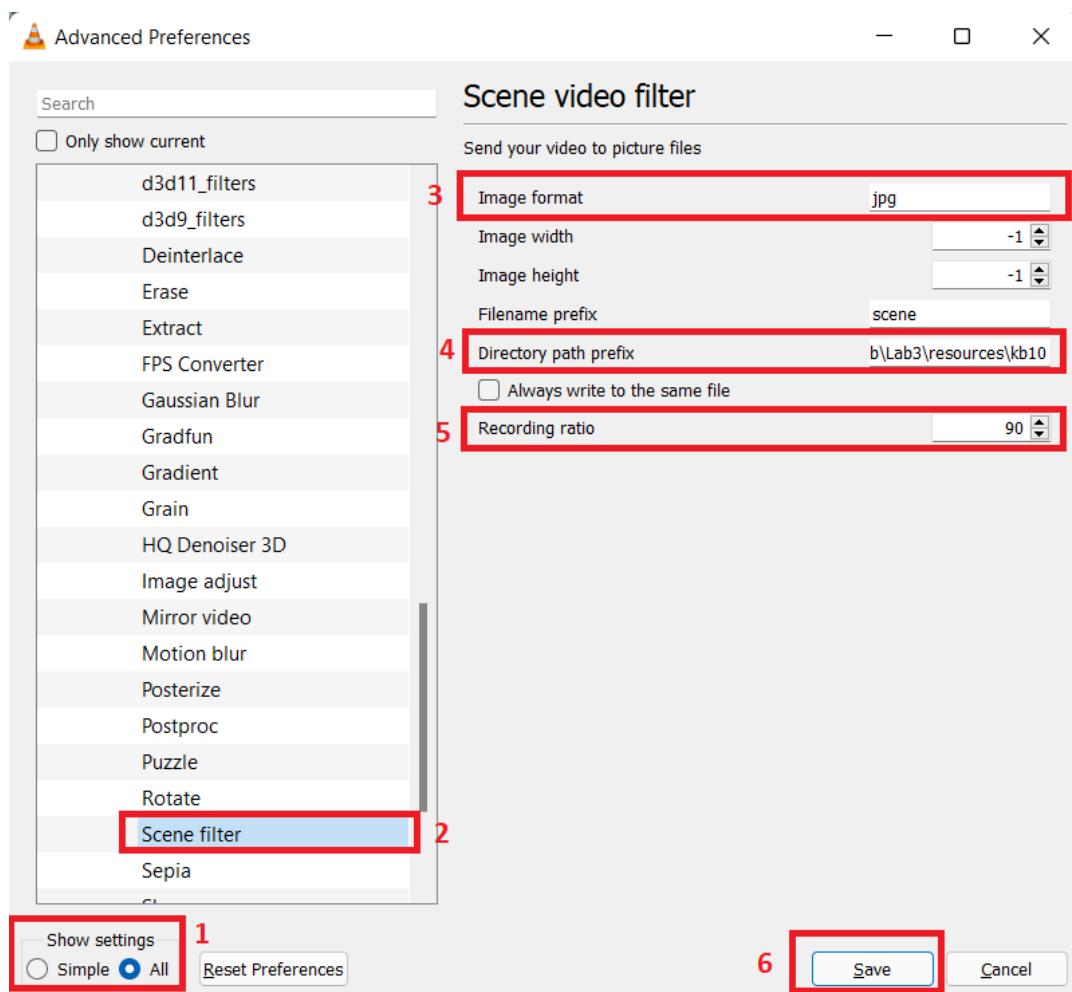
Đáp án:

Khi xem video, ta để ý thấy video có chuyển động của “thứ gì đó”. Có thể đây là một video có nội dung nhưng bị *lỗi nhiễu*. Định hướng của ta là thử chia video thành nhiều frame và so sánh nội dung thay đổi theo từng frame (vì mình không thể trực tiếp khử nhiễu để xem video gốc, hoặc là mình chưa biết :< )



Trong VLC Media Player có hỗ trợ tính năng để tách các frame thành các image. Cài đặt như sau:

## Thi thực hành cuối kì



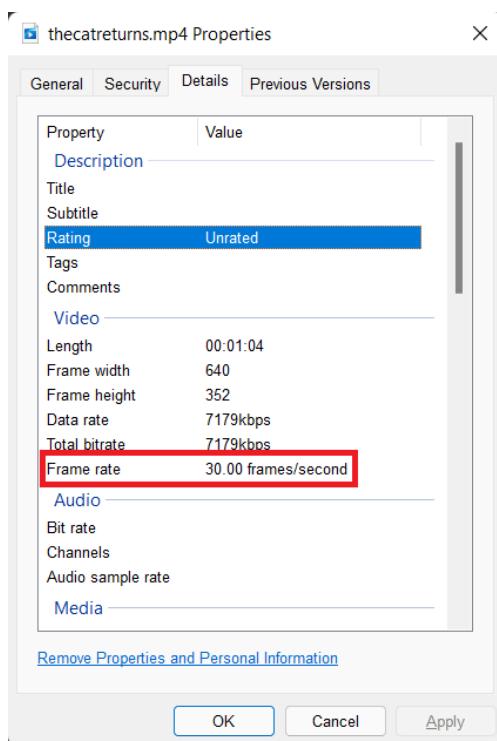
- Mở Preferences (*Ctrl + P*)

- Trong phần “Show settings” chọn **All**

- Trong khung diaglog box ở phía tay trái: *Video -> Filters -> Scene Filter*:

- Chọn **Image Format** là **jpg** (để cho xem trước ảnh khi view trong folder)
- **Directory path** prefix là nơi lưu trữ output
- **Recording ratio** là phần quan trọng. Nó sẽ lưu trữ mỗi ảnh theo mỗi **N frames**, ở đây mình set **N** là **90 frames** vì trong video gốc có tốc độ khung hình là **30 frames/s** nên cách khác là mình sẽ lưu ảnh mỗi **3s**

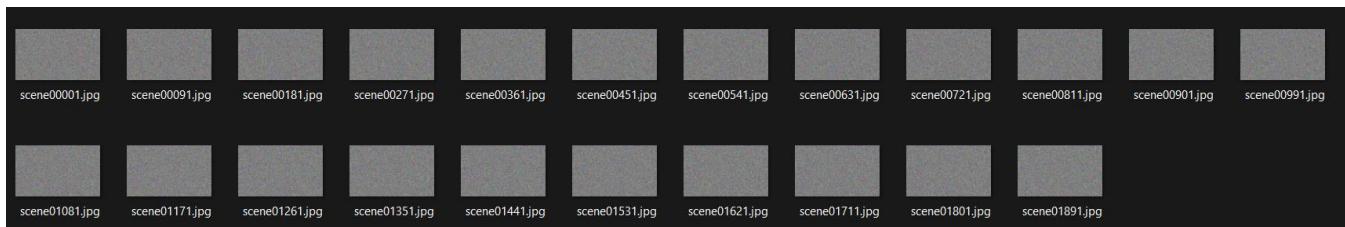
## Thi thực hành cuối kì



*NOTE:* Mình có thể giảm số này lại để lưu ảnh với độ chính xác cao hơn, chi tiết hơn, nhưng đối với bài này mình thấy ảnh trên video không thay đổi nhiều nên mình cảm nhận ra số này :v

- Nhấn "Save"

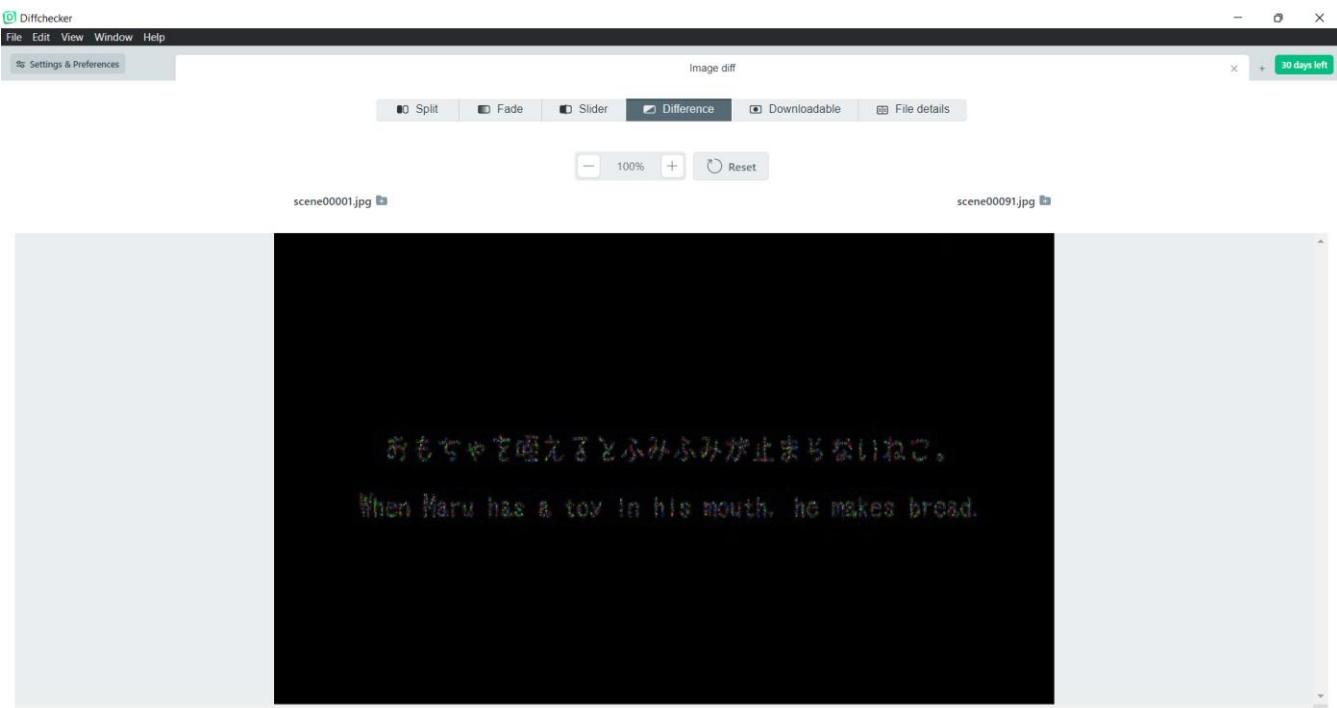
Tắt VLC đi và mở lại rồi chạy video. Sau mỗi 3s video sẽ generate ra một ảnh theo như ta cấu hình, để video (1:04 -> 64s) chạy hết ta sẽ có **64/3 = 21.(3) = 22** ảnh, như bên dưới:



Dùng tool [diffchecker](#) để show nội dung khác nhau giữa 2 ảnh và làm rõ nó lên

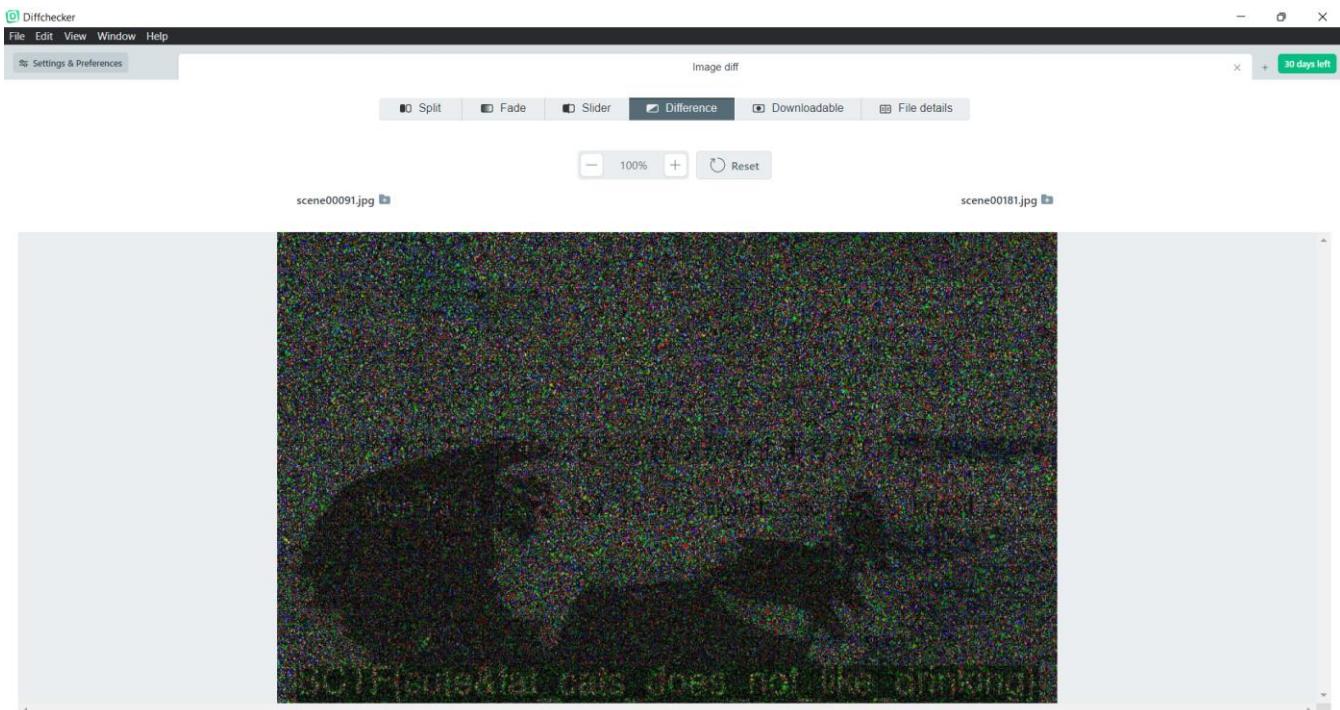
So sánh 2 đầu **scene00001.jpg** và **scene00091.jpg**

## Thi thực hành cuối kì



Ta thấy có thông điệp “*When Maru has a toy in his mouth, he makes bread*”. Có vẻ hướng đi ta đúng, vì câu thoại này ám chỉ con mèo, tức ám chỉ đến tiêu đề challenges.

So sánh 2 image **scene00091.jpg** và **scene00181.jpg**. Ta thấy có flag:



Flag: **BCTF{cute&fat\_cats\_does\_not\_like\_drinking}**