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**

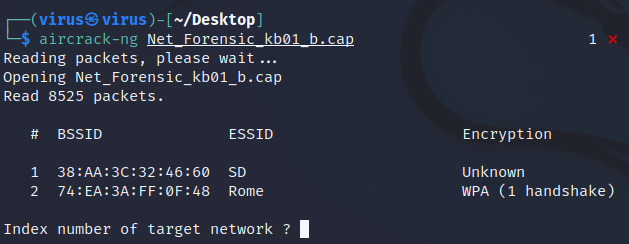
|  |
| --- |
| ***Lưu ý***: Trong bài có sử dụng một số tool mà trên các repo chính thống, hay các trang chủ và các nguồn download chính thức bị lỗi/gặp vấn đề/dead link thì mình có thể dùng [Wayback Machine](https://archive.org/web/) để “lùi lại” và xem những version trước của trang web download đó (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 tập tin dữ liệu mạng.** - Mô tả: Một máy tính trọng mạng nội bộ bị nghi ngờ tấn công từ bên ngoài, nhân viên quản trị mạng dùng những công cụ chuyên dụng bắt các kết nối đến máy nạn nhân trọng thời gian diễn ra cuộc tấn công. Sau đó lưu lượng mạng được trích xuất toàn bộ nội dung trọng tập tin pcap.- Tài nguyên thực hiện: traffic\_kb01\_a.pcap- Yêu cầu: Thực hiện phân tích tập tin dump từ dữ liệu mạng để tìm nguồn gốc và nguyên nhân vụ tấn công để có giải pháp khắc phục |

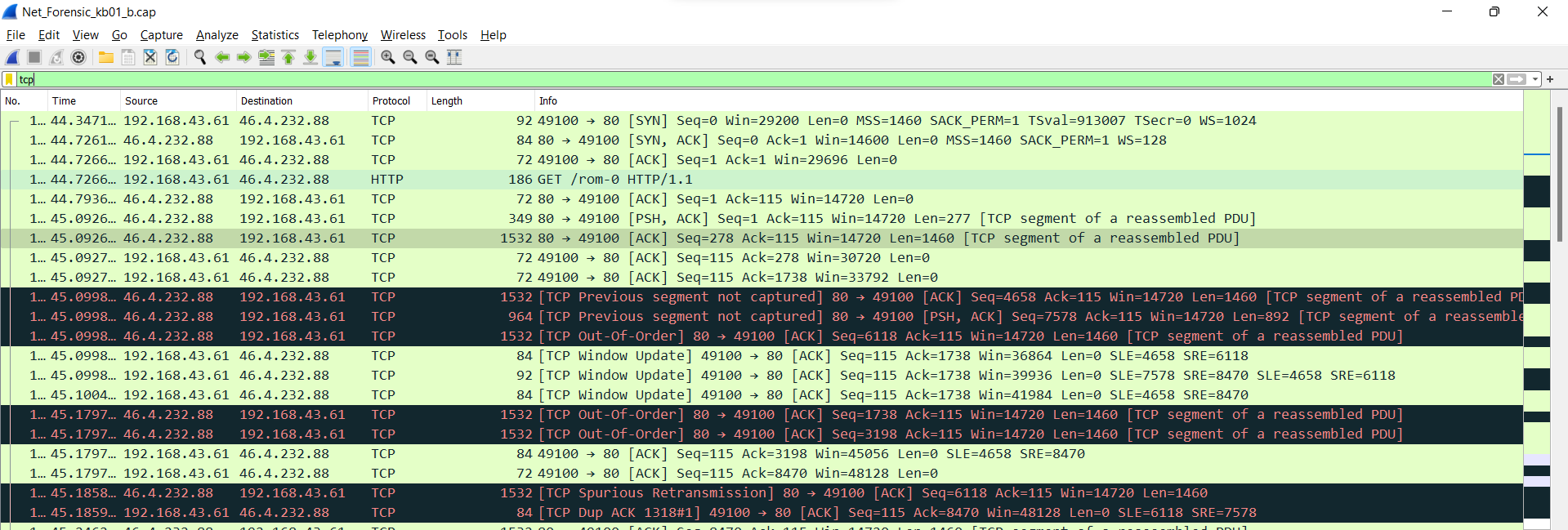
|  |
| --- |
| **Kịch bản 01-b. Thực hiện phân tích tập tin dữ liệu mạng thu được.**  - Mô tả: Tập tin pcap được chọ là dữ liệu mạng thu được từ một mạng không dây.  - Tài nguyên thực hiện: Network\_Forensic\_kb01\_b.pcap  - Yêu cầu: Thực hiện phân tích tập tin dump từ dữ liệu mạng để tìm SSID, mật khẩu giải mã stream TCP, sau đó phân tích stream đã giải mã để tìm flag.  Đáp án: Flag: be02d2a396482969e39d92b6e440f5e3 |

Lấy thông tin cơ bản traffic network bằng **aircrack-ng**, trích xuất thông tin ***Wifi Encryption (WPA) .***

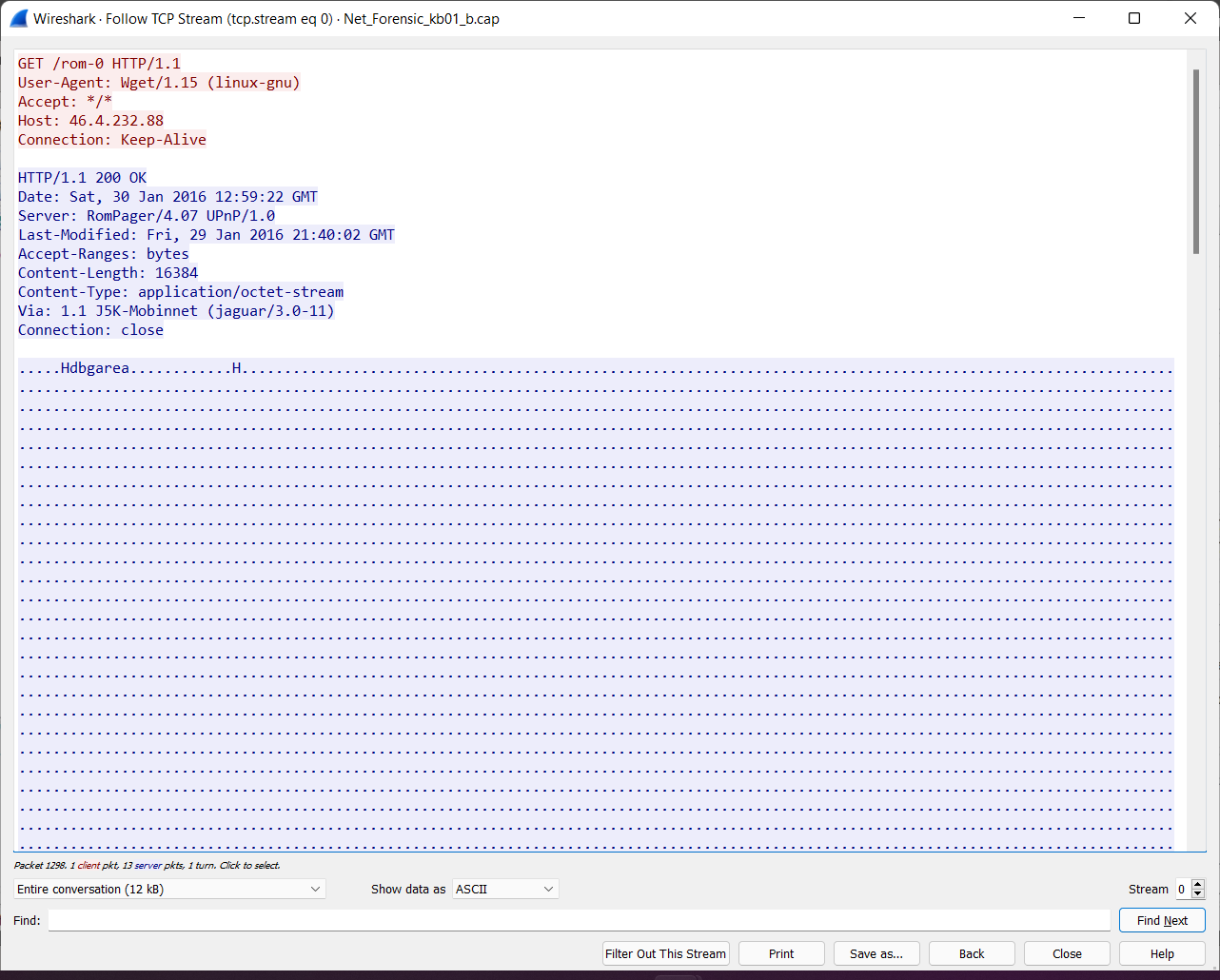


***\*Note****: Loại encryption này cũng khá cũ rồi, giờ mình thấy người ta hay xài WPA2*

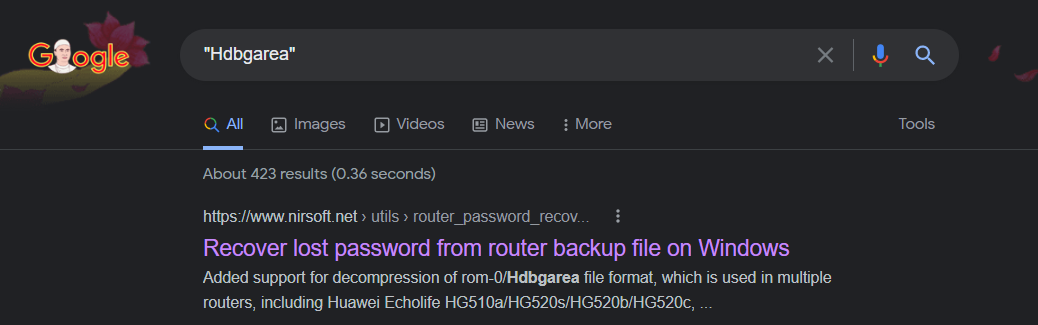
Dùng Wireshark để đào sâu thêm, lọc các gói tin **TCP**



*Chuột phải vào gói tin TCP bất kỳ -> TCP Stream hoặc (Ctrl + Alt + Shift + T)*. Ta sẽ thấy full nội dung được gửi đi.

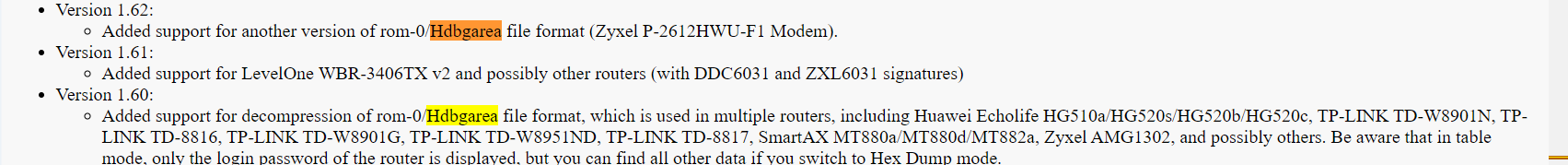


“Hdbgarea” là đoạn mở đầu trong file cấu hình của **RouterPassView**

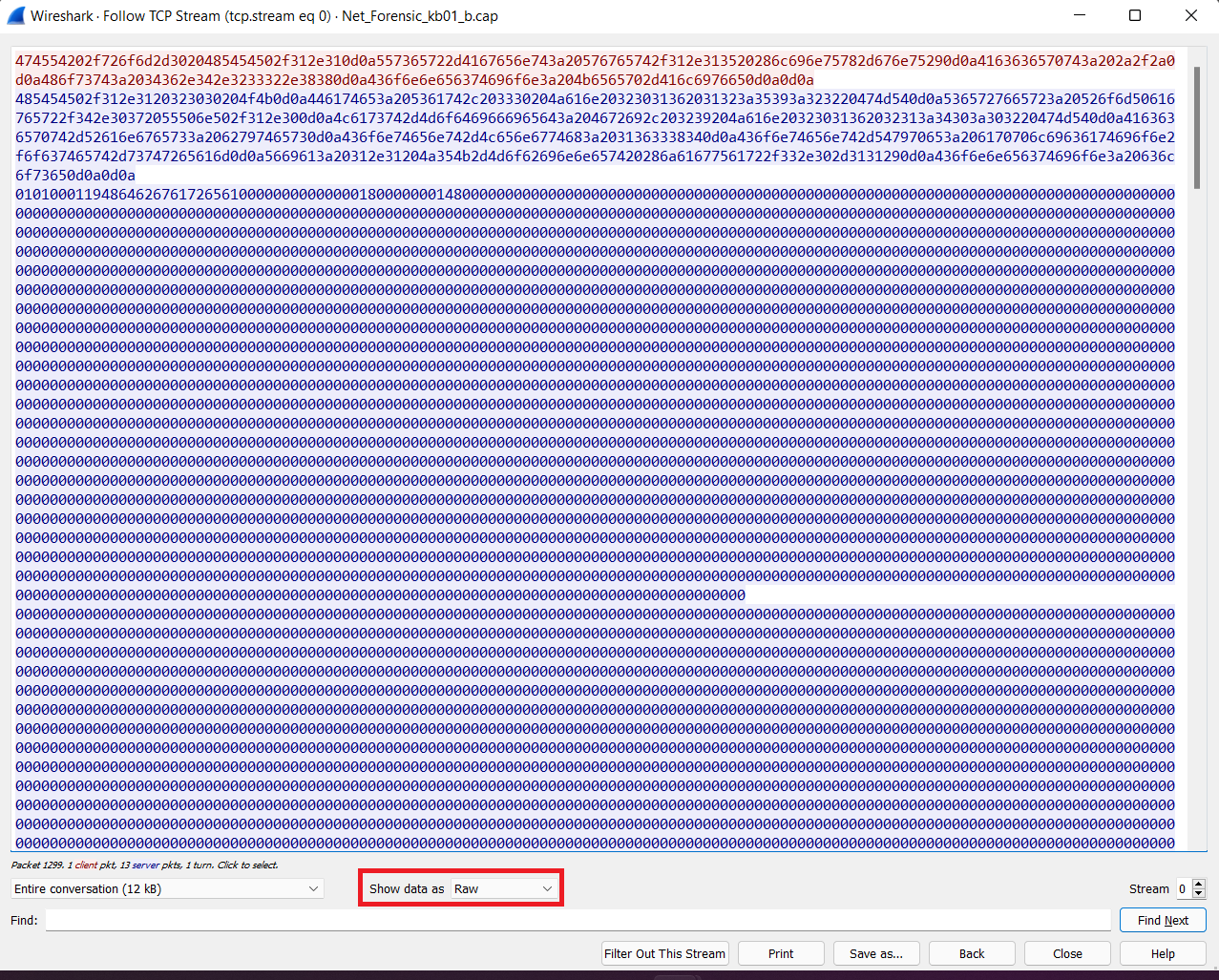


Kết quả dẫn ta đến link này (link đầu tiên ở trên):

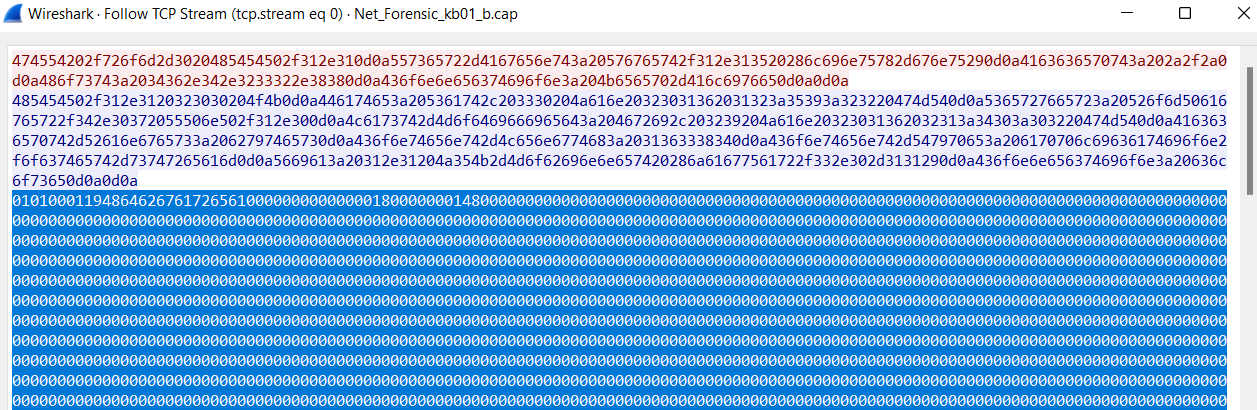
<https://www.nirsoft.net/utils/router_password_recovery.html>

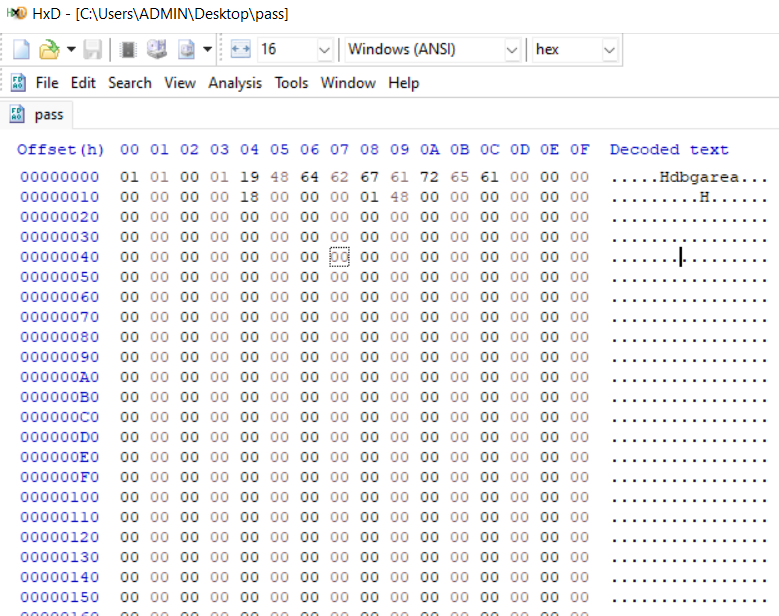


Tải phần mềm phục hồi về [RouterPassView](https://www.nirsoft.net/toolsdownload/routerpassview.zip) . Lưu nội dung request trên thành một file cấu hình hợp lệ. Xóa tất cả header trong HTTP request protocol đi. Để làm vậy mình sẽ để dưới dạng Raw cho tiện:

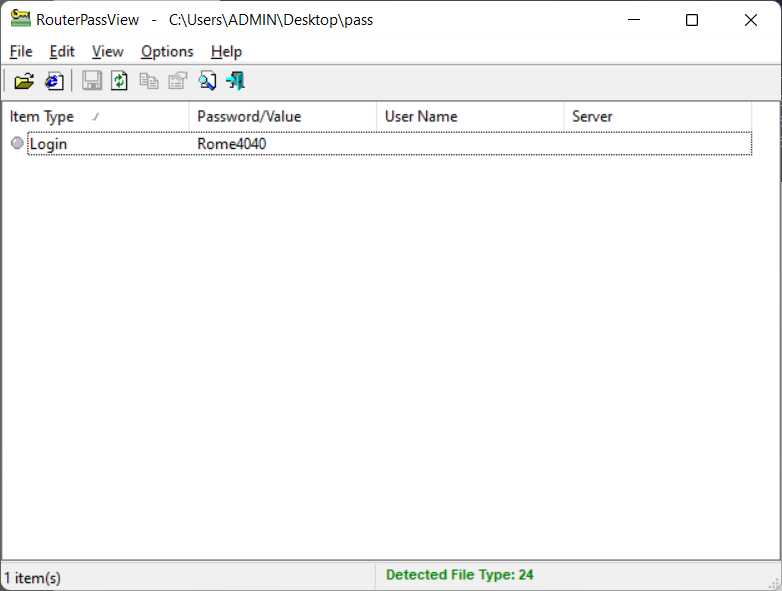


Sau đó mình copy đoạn hex từ 0101 đến hết xuống dưới (Phần đầu là header) và paste vào HxD (Hex Editor).



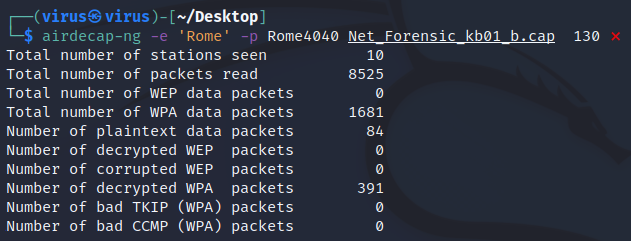


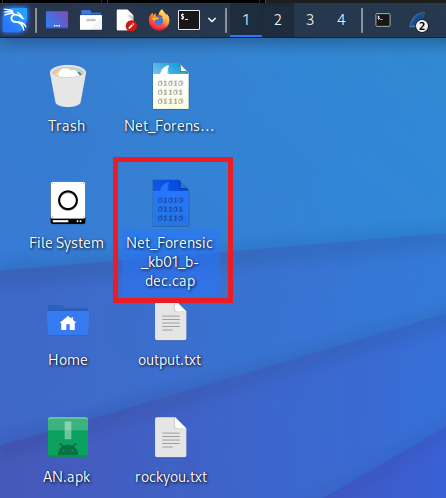
Lưu lại với tên file **config**. Sau đó bỏ vào phần mềm **RouterPassView**ở trên để recover lại password.



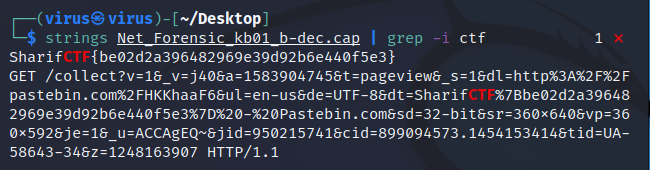
Mật khẩu ở đây là **Rome4040.** Sau đó dùng aircrack-ng để giải mã các packet với mật khẩu đã xác định với tùy chọn -e là **ESSID** và -p là **password** đã tìm

|  |
| --- |
| *airdecap-ng -e 'Rome' -p Rome4040 Net\_Forensic\_kb01\_b.cap* |



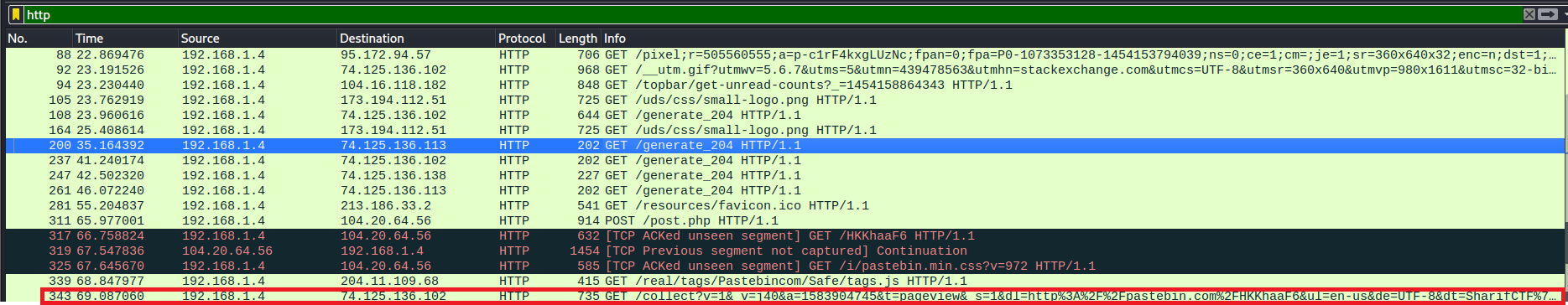


Kiểm tra có flag trên toàn bộ gói tin?

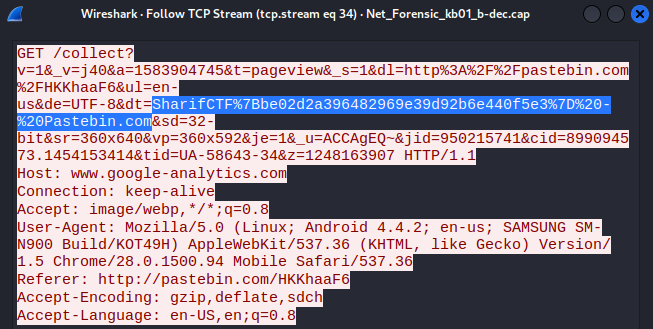
****

Như ta thấy thì dùng cách này có thể thấy được flag. Vì đây là một bài ctf nên đến đây có thể xong rồi. Tuy nhiên, mình nên làm challenge với góc độ phân tích packet nên mình sẽ examine lại file đã được decrypt trên.

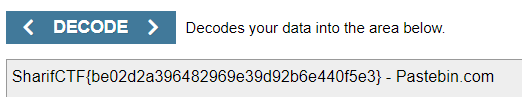
Ta thấy gói packet có timestamp 343 có định dạng flag:



Follow stream để xem toàn bộ flag:



Đưa vào [URL Decode](https://www.urldecoder.org/) để xem truy vấn gốc:



Flag: **SharifCTF{be02d2a396482969e39d92b6e440f5e3}**

|  |
| --- |
| **Kịch bản 02. Điều tra trên dữ liệu lưu lượng mạng thu được.**  - Tài nguyên: capture-output\_kb02.7z  - Yêu cầu: Thực hiện phân tích các request DNS, các truy cập HTTP đến các trang web nào. Người dùng đã gửi một số tập tin thông qua một trang web. Xác định dịch vụ mà người dùng sử dụng để chuyển tập tin, thông tin người nhận (email, thông điệp lời nhắn, tên file đã gửi). Trích xuất nội dung các file đã gửi.  *Gợi ý: Wireshark/tshark* |

* Thực hiện phân tích các request DNS, các truy cập HTTP đến các trang web nào

Để xác định được user truy cập trang web nào. Ta có combo payload sau:

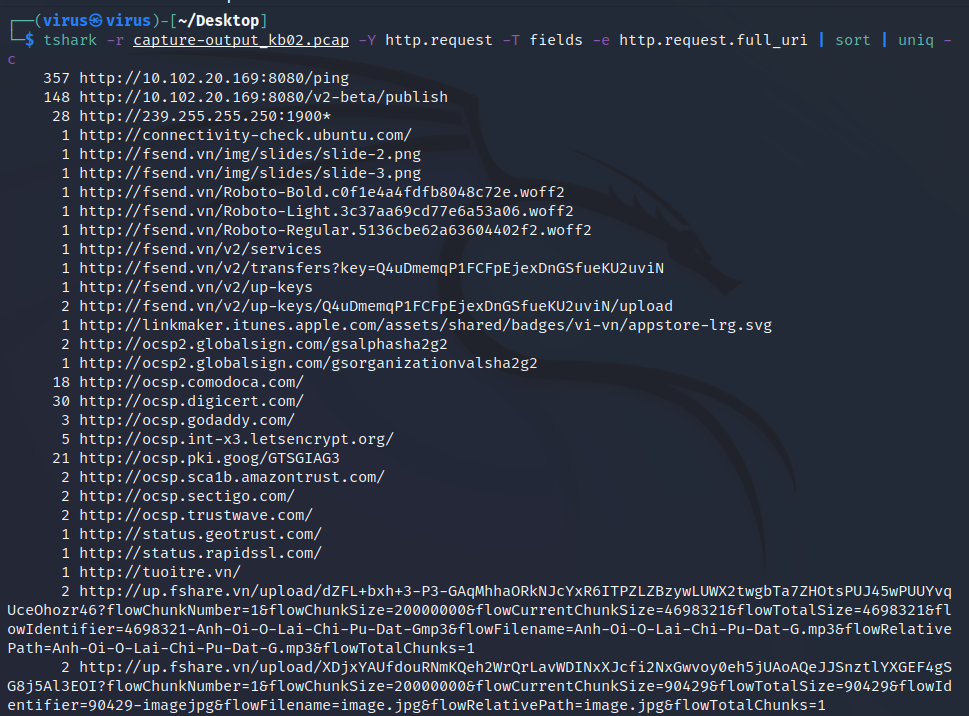
|  |
| --- |
| *tshark -r capture-output\_kb02.pcap -Y http.request -T fields -e http.request.full\_uri* |

Với -r là file pcap cần phân tích, -Y là filter (syntax như Wireshark), -T là dạng xuất ra (ở đây là fields) và -e là trường thông tin được lấy ra.

Các URL được lấy ra khá nhiều và bị trùng lặp. Để cho đẹp hơn thì mình nên **sort** lại (để các link giống nhau gần nhau) và sau đó **uniq** theo số dòng (tức là số lần xuất hiện)

|  |
| --- |
| *tshark -r capture-output\_kb02.pcap -Y http.request -T fields -e http.request.full\_uri | sort | uniq -c* |

Toàn bộ output:



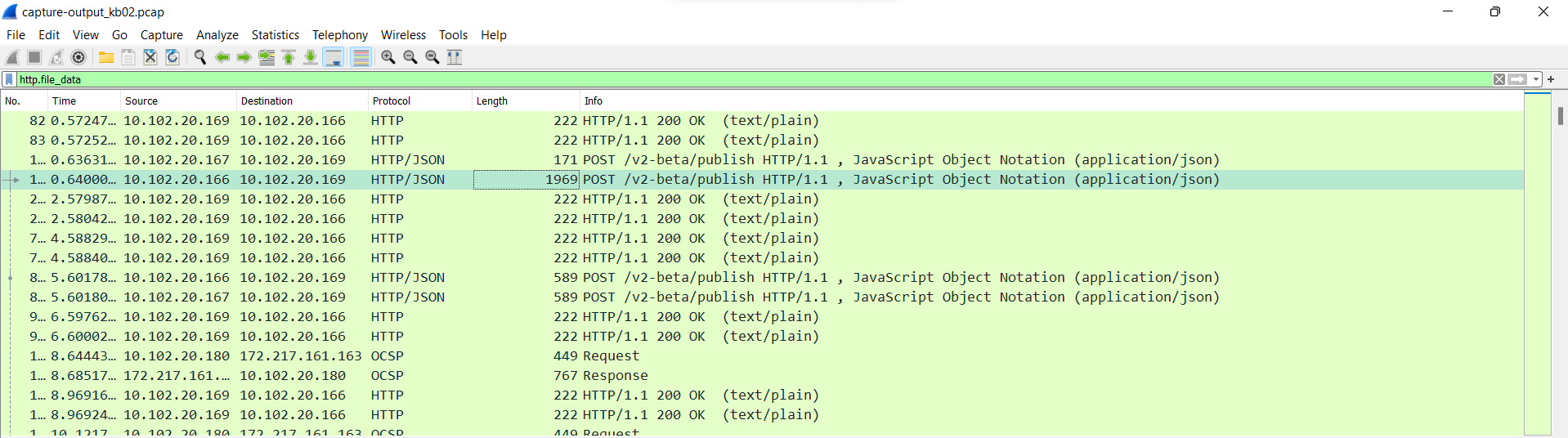
* Người dùng đã gửi một số tập tin thông qua một trang web. Xác định dịch vụ mà người dùng sử dụng để chuyển tập tin, thông tin người nhận (email, thông điệp lời nhắn, tên file đã gửi).

Ở đây, như hình trên, ta có thể search google với các domain trên và thấy user sử dụng 2 trang web chính để upload file:

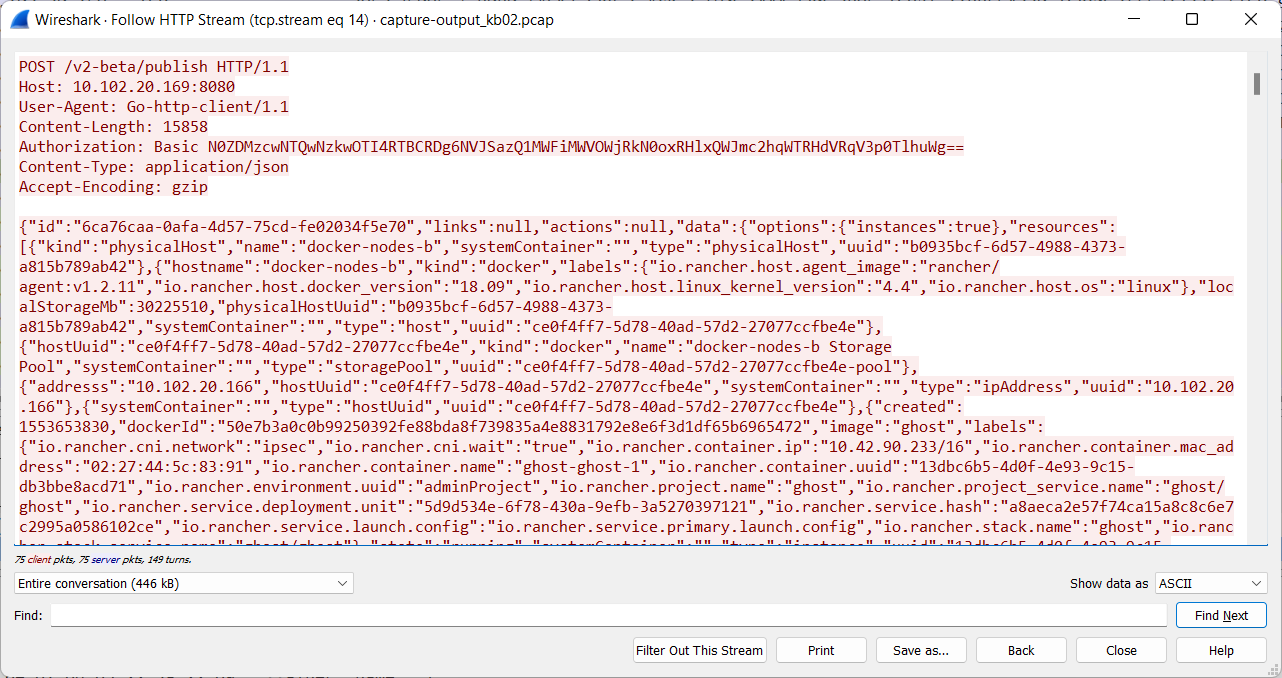
- <http://fsend.vn>

- <https://www.fshare.vn/>

Dùng Wireshark để xem thông tin các packet có request method là POST trên các URL này. Dùng bộ lọc http.file\_data :



Thực hiện xem một số POST packet:



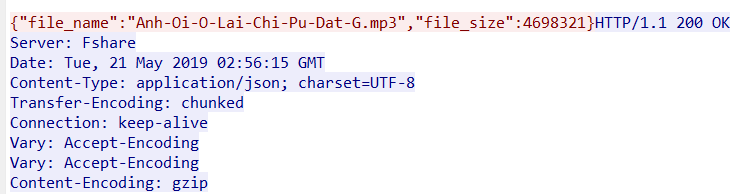
Ta thấy không có gì đặc biệt lắm. Thử tìm kiếm chặt chẽ hơn bằng cách thêm bộ lọc với các ***request POST*** và URL chứa ***“http://fsend.vn”.*** Filter Wireshark:

|  |
| --- |
| *http.file\_data && http.request.method == "POST" && http contains "http://fsend.vn"* |

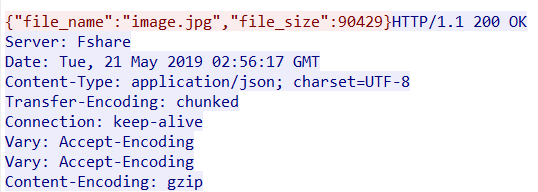


May mắn thay, **follow** packet đầu tiên ta thấy có chứa mọi thông tin ta cần. File được upload gồm 1 file mp3 và 1 file ảnh:

**Anh-Oi-O-Lai-Chi-Pu-Dat-G.mp3**



**image.jpg**



Thông tin gửi:



- Người nhận (recipient): [duypt@uit.edu.vn](mailto:duypt@uit.edu.vn)

- Thông điệp (message): “Khong o lai dau :v”

- Tiêu đề: bỏ trống (null)

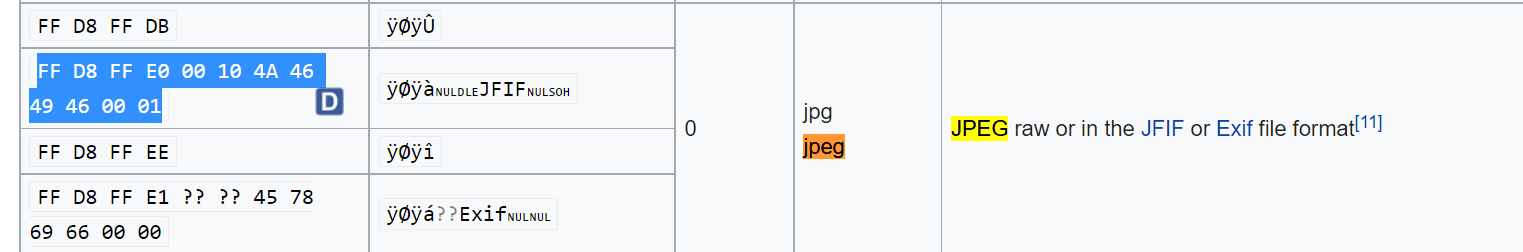
Trích xuất file:

- Xem packet đã gửi có chứa file:

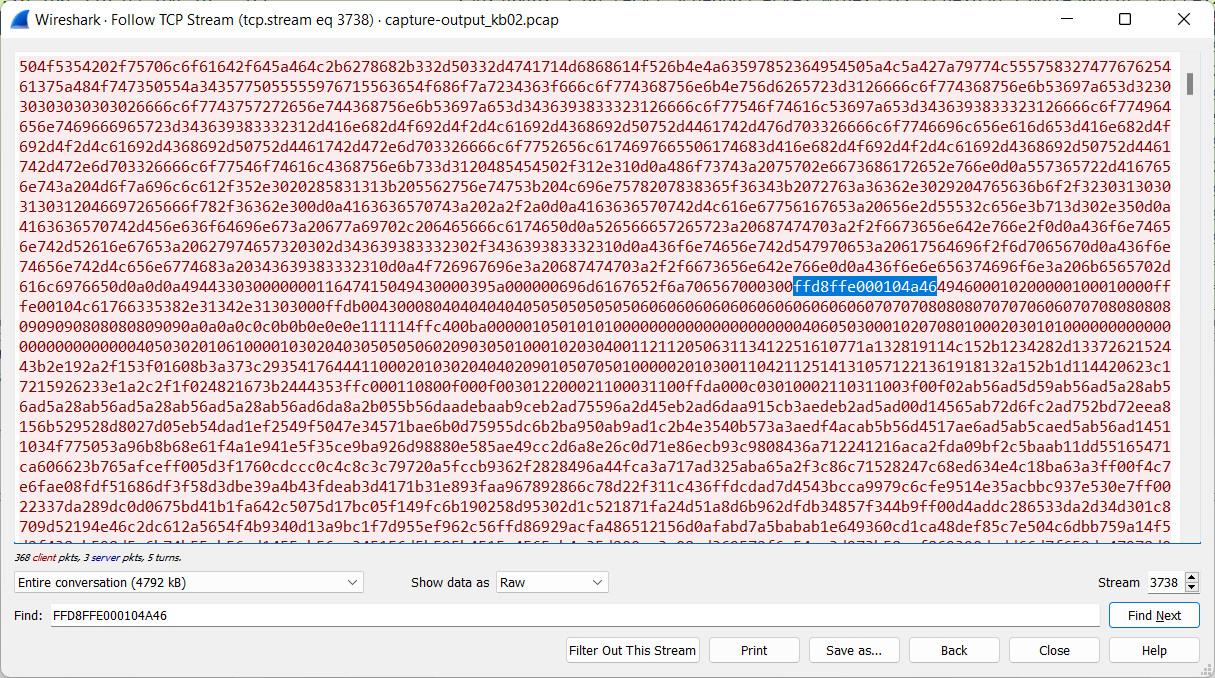
+ Packet thứ 2 trong 4 packet đã fileter ở trên -> *Follow Stream*



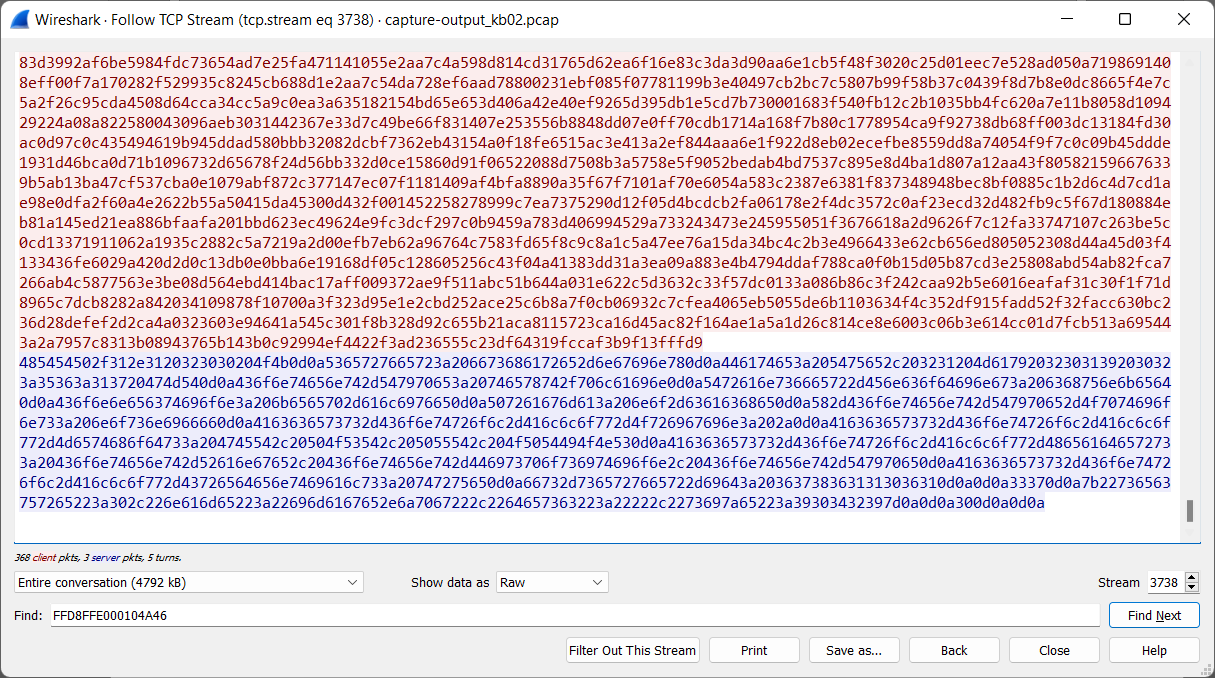
Ta đã có nội dung file, chuyển sang view dưới dạng Raw (***Show data as***) , và tìm chuỗi định dạng file (chữ ký file) hợp lệ [tại đây](https://en.wikipedia.org/wiki/List_of_file_signatures) .



Phần bắt đầu, (search với chuỗi chữ ký trên ***“FFD8FFE000104A46”***)



Phần kết thúc file (giới hạn đến phần đỏ, màu xanh là header của request):



Copy từ đầu đến cuối (như đã phân tích ở trên) vào một file mới trong phần hex data (bên trái) của HxD. Lưu file lại thành ***Anh-Oi-O-Lai-Chi-Pu-Dat-G*.jpg**

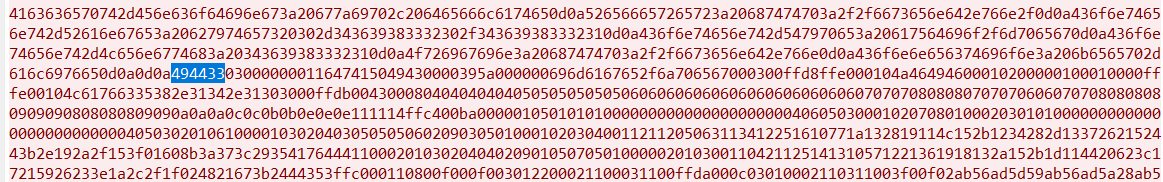
Image được trích xuất:



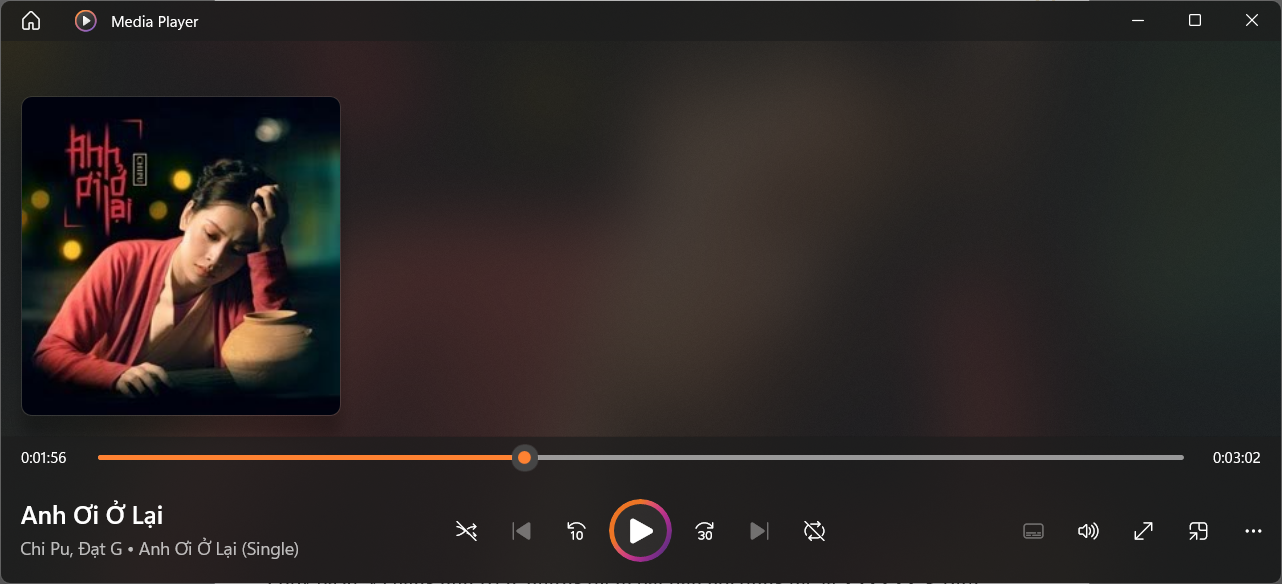
+ Thực hiện tương tự với stream gói tin trên, ta nhận ra file image này là một phần của file mp3. Signature của file mp3



Tìm kiếm trong stream:



Copy paste y chang như trên, nhưng điểm bắt đầu nội dung file là **494433 ->** *Đưa vào HxD -> Save as -> Anh-Oi-O-Lai-Chi-Pu-Dat-G.mp3*



+ Tương tự tiếp với luồng cũ, chữ ký **FFD8FFE000104A46** xuất hiện 2 lần, đồng nghĩa với việc có 2 file ảnh, thực hiện với chữ ký thứ 2 (nằm gần ở dưới).



Copy từ đoạn này đến phần đuôi như ở trên (không đổi) -> *Đưa vào HxD -> Save as -> image.jpg*

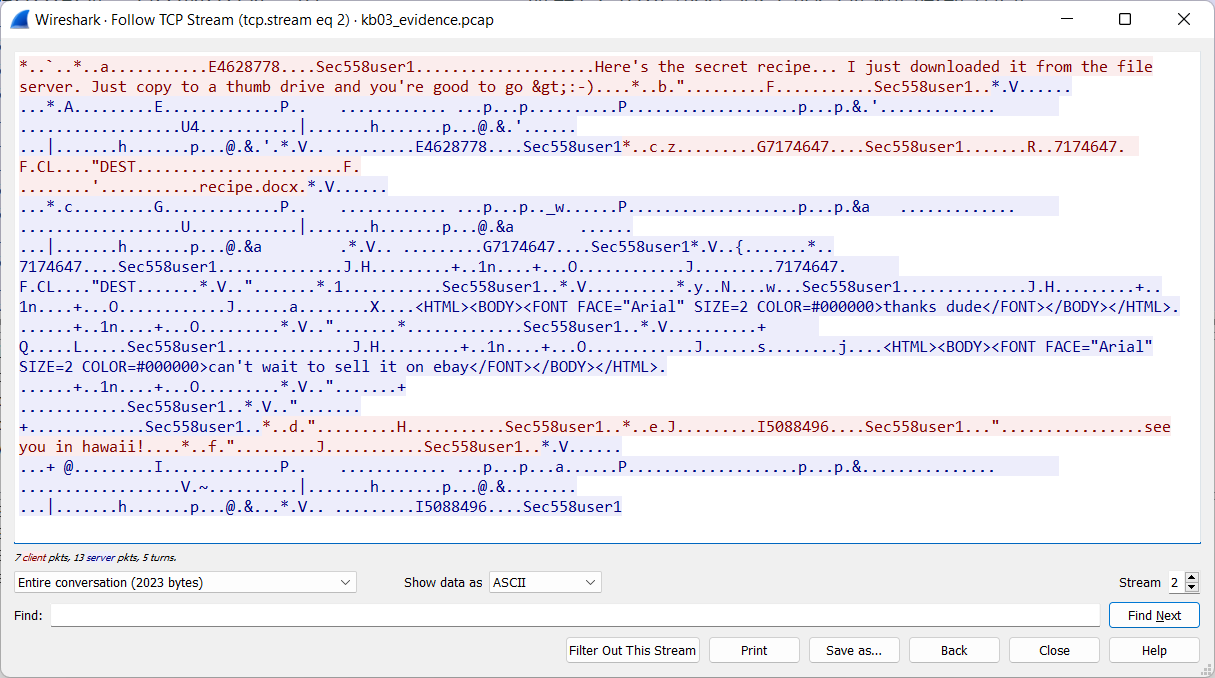


|  |
| --- |
| **Kịch bản 03. Điều tra trên dữ liệu lưu lượng mạng thu được.**  - Tài nguyên: kb03\_evidence.pcap  - Mô tả: Công ty Anarchy-R-Us, Inc. chọ rằng một trọng những nhân viên của họ,  Ann Dercover là một gián điệp thương mại làm việc chọ công ty đối thủ vì nhân viên này đã từng xâm nhập vàọ máy chủ chứa dữ liệu mật của công ty. Nhân viên an ninh của công ty nghi ngờ rằng Ann đã trộm công thức bí mật của công ty.  Nhân viên an ninh mạng đã theo dõi Ann một thời gian nhưng chưa phát hiên được gì. Hôm nay, có một laptop lạ đã kết nối vàọ mạng wireless của công ty. Máy tính của Ann (IP: 192.168.1.158) đã gởi một số tin nhắn tới máy tính đó, laptop lạ ngắt kết nối với mạng wireless ngay sau đó. Dữ liệu mạng của máy của phiên kết nối đã bị an ninh  mạng công ty lưu lại. Hãy giúp công ty điều tra xếm Ann có phải là gián điệp hay không, và công thức bí mật của công ty đã bị đánh cắp hay không?  ***Đáp án:*** |

Filter với ip source là hint như trên:

|  |
| --- |
| *ip.src == 192.168.1.158* |

Click vào và follow stream một gói tin SSL bất kỳ



Ta thấy mặc dù nội dung đã bị mã hóa nhưng vẫn tiết lộ được một phần nào các thông điệp đáng nghi như đoạn thông điệp:

“Here's the secret recipe... I just downloaded it from the file server. Just copy to a thumb drive and you're good to go”

Hay tên file :

recipe.docx

Đến đây kết luận thì cũng hơi vội 😶. Chưa chắc người ta đã làm tình làm tội gì vì chưa biết được file ở trên có thực sự chứa thông tin nhạy cảm của công ty hay không.

Ở đây, mình sẽ sử dụng Tool **NetworkMiner**. Tool này khá mạnh mẽ trong việc hỗ trợ “phân tích sẵn có”, nhận đầu vào là tập tin .*pcap, .dump, .dmp ,…..* và phân tích các trường thông tin tổng quan nhưng quan trọng như *hosts, files, images, …* xuất hiện trong file đã bắt được.

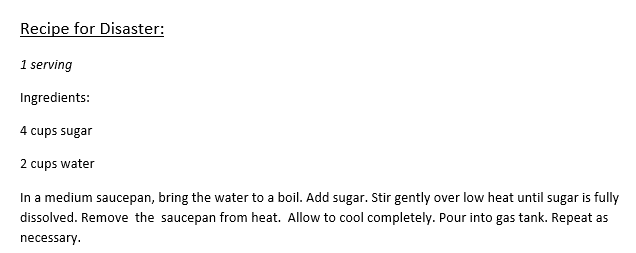
Import file đầu vào, xem thông tin với IP **192.168.1.158.**



Chuyển sang tab *Files ->* Thấy trích xuất được 3 files



Trong đó ta cần quan đến file **recipe.docx** *-> Chuột phải vào file -> Chọn Open File.* Ta thấy được nội dung file như sau:

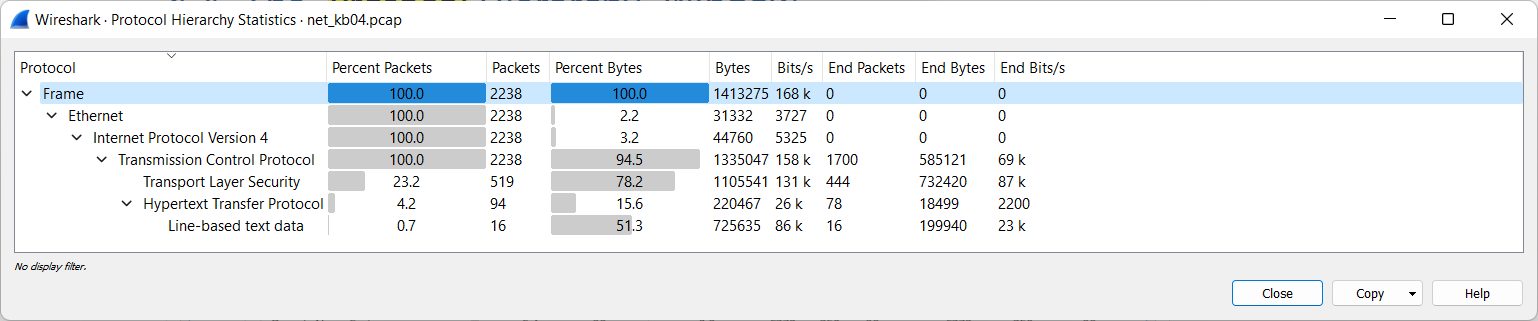


Vậy là đã đủ chứng minh thanh niên này bịp rồi 😡. Giờ thì kick ổng ra khỏi công ty thôi

|  |
| --- |
| **Kịch bản 04. Điều tra trên dữ liệu lưu lượng mạng thu được.**  - Tài nguyên: net\_kb04.pcap  - Yêu cầu – Gợi ý: Đây là dữ liệu mạng thu được khi bắt gói tin duyệt web trọng một khọảng thời gian. Tìm flag, biết flag có định dạng flag{…}  *Đáp án:*  [https://github.com/ctfs/write-ups-2015/tree/master/csaw-ctf-2015/forensics/transfer-100](https://github.com/ctfs/write-ups-2015/tree/master/csaw-ctf-) |

Chúng ta thử xem sơ lược các giao thức được sử dụng trong file capture.

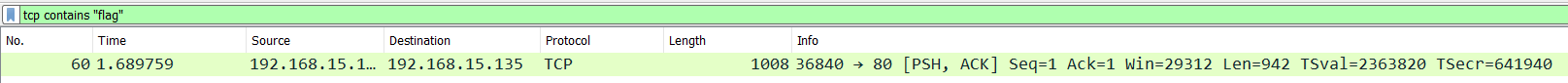
*Statistics -> Protocol Hierachy*



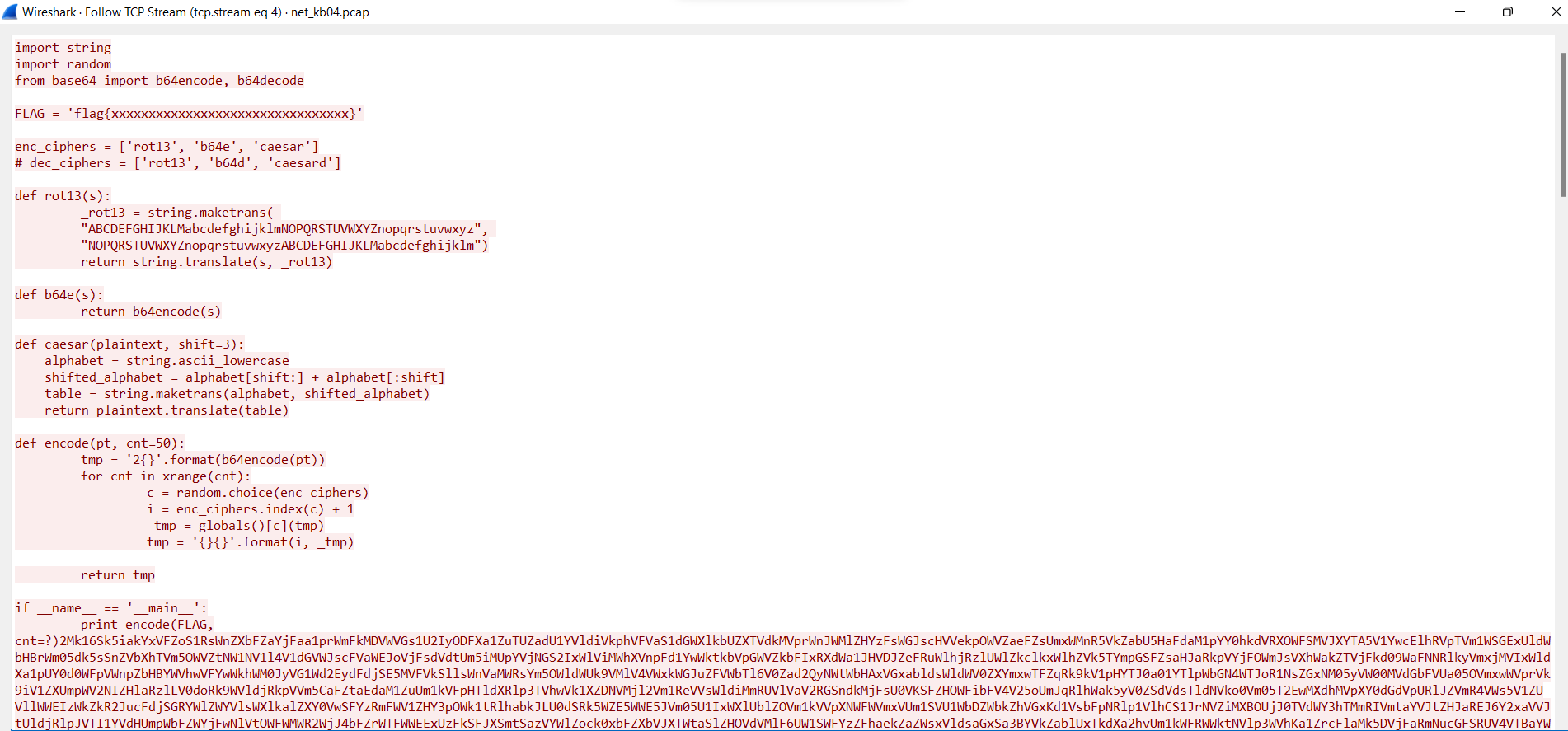
Các giao thức chúng ta cần quan tâm là TCP, HTTP, TLS. Nhưng cũng không cso mấy thông tin quan trọng .

Chúng ta được hint trước định dạng flag là **flag{…}**. Vậy cứ “chăm chăm” tìm flag này trước thôi 🚩

Tìm thông tin với giao thức TCP trước:



Chỉ có duy nhất một packet. Xem nội dung thử như thế lào:



Ta thấy nội dung file là một chương trình mã hóa một chuỗi plaintext thành ciphertext. Và cipher ta có là chuỗi dài loằn ngoằn ở phía sau. Bây giờ nhiệm vụ của chúng ta là phân tích file encode và tìm cách viết file decode.

Phân tích:

Hàm chính ***encode()***

def encode(*pt*, *cnt*=50):

    tmp = '2{}'.format(b64encode(*pt*))

    for *cnt* in xrange(*cnt*):

        c = random.choice(enc\_ciphers)

        i = enc\_ciphers.index(c) + 1

        \_tmp = globals()[c](tmp)

        tmp = '{}{}'.format(i, \_tmp)

    return tmp

Hàm này thực hiện đưa số 2 vào trước chuỗi plaintext, sau đó đưa vào vòng for và trong mỗi lần lặp thì chọn random một thuật toán trong mảng enc\_ciphers **['rot13', 'b64e', 'caesar']** . Thực hiện đến 50 lần (vì tham số mặc định là 50).

Các hàm như **rot13, b64e, caesar** có chức năng mã hóa nhất định, tuân theo quy luật của 3 loại mã hóa cơ bản như đã biết : *ROT13, base64, Caesar*

Hàm **rot13** không cần viết lại

Hàm **base64** dùng sẵn thư viện để decode

Hàm **Caesar** decode sẽ tương đương với việc truyền tham số trái dấu so với hàm encode

Build hàm decode, việc decode thực hiện tương tự nhưng nghịch đảo lại.

\***Note:** Vì số random trong thuật toán encode lỏng lẻo nên bản chất ciphertext vẫn chứa số random này

Lưu ciphertext vào một file khác ***ciphertext.txt.*** Chương trình decode hoàn chỉnh:

import string

import random

from base64 import b64encode, b64decode

FLAG = open("ciphertext.txt").read()

dec\_ciphers = ['rot13', 'b64d', 'caesard']

def rot13(*s*):

    \_rot13 = string.maketrans(

        "ABCDEFGHIJKLMabcdefghijklmNOPQRSTUVWXYZnopqrstuvwxyz",

        "NOPQRSTUVWXYZnopqrstuvwxyzABCDEFGHIJKLMabcdefghijklm")

    return string.translate(*s*, \_rot13)

def b64d(*s*):

    return b64decode(*s*)

def caesar(*plaintext*, *shift*=3):

    alphabet = string.ascii\_lowercase

    shifted\_alphabet = alphabet[*shift*:] + alphabet[:*shift*]

    table = string.maketrans(alphabet, shifted\_alphabet)

    return *plaintext*.translate(table)

def caesard(*ciphertext*, *shift*=3):

    return caesar(*ciphertext*, *shift*=-*shift*)

def decode(*ct*):

    while True:

        try:

            i = int(*ct*[0]) - 1

        except:

            print(*ct*)

            exit(0)

*ct* = *ct*[1:]

        c = dec\_ciphers[i]

        \_ct = globals()[c](*ct*)

*ct* = \_ct

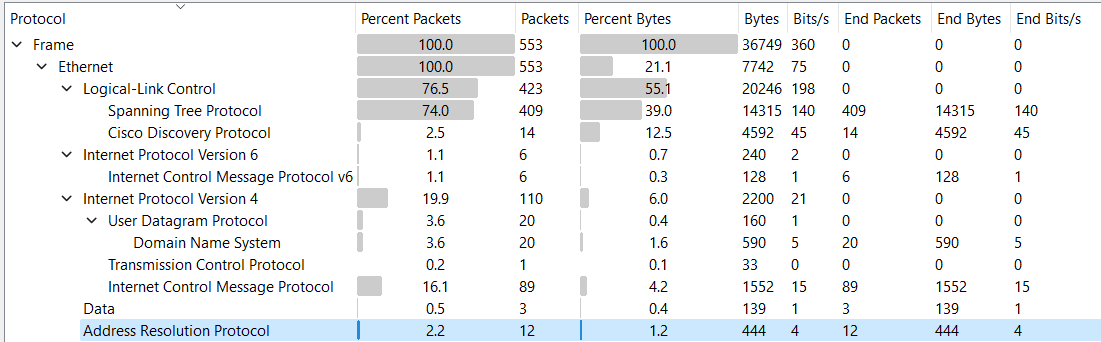
if \_\_name\_\_ == '\_\_main\_\_':

    decode(FLAG)

Flag: **flag{li0ns\_and\_tig3rs\_4nd\_b34rs\_0h\_mi}**

|  |
| --- |
| **Kịch bản 05. Điều tra trên dữ liệu lưu lượng mạng thu được.**  - Tài nguyên thực hiện: kb05.gz  - Yêu cầu – Gợi ý: Xác định các kết nối trọng dữ liệu thu được. Chú ý các  gói ICMP, trường giá trị Identifiers của các gói để tìm flag. Flag có định  dạng bắt đầu bằng chuỗi “S3”, với tổng chiều dài là 11 kí tự.  - Công cụ: Wireshark, tshark,…  **Gợi ý: https://github.com/ctfs/write-ups-2015/tree/master/nuit-du-hack-ctfquals-2015/forensic/private** |

Thực hiện khảo sát sơ lược có các protocols nào: *Statistics -> Protocol Hierachy*

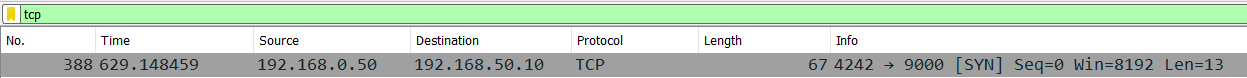


Ở đây ta thấy có khá nhiều giao thức: LLC, STP, Ipv6, ARP, … Đề có nói chú đến đến các gói **ICMP**, và flag có định dạng bắt đầu bằng chuỗi **“S3”**. Vậy ta có filter sau:

|  |
| --- |
| *icmp matches "^S3"* |

Kết quả không khả thi, không trả về gói tin nào. Tính mở bằng Network Miner mà thấy đuôi là pcapng (chỉ có bản Pro mới “chơi” được) nên thôi. 😭

Thử loay hoay tìm các giao thức khác (DNS, HTTP) thì thấy thằng TCP có duy nhất 1 gói tin



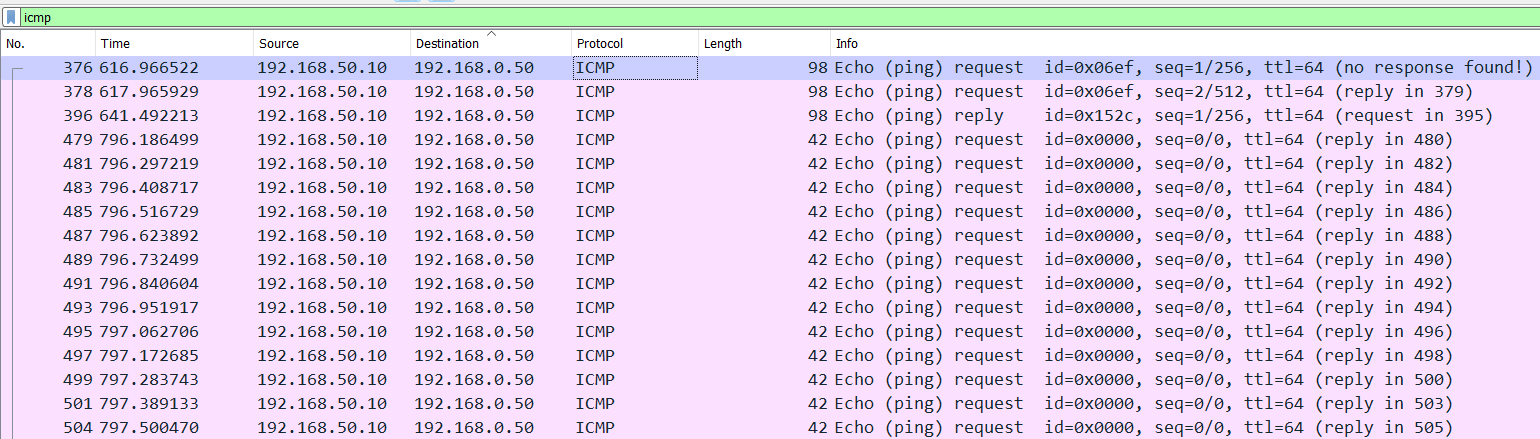
Ở đây ta thấy IP Source là **192.168.0.50** gửi gói tin đến **192.168.50.10**.

**\*Note:** Ở đây nếu ta xem nội dung gói tin TCP trên ta sẽ thấy có một dòng chữ

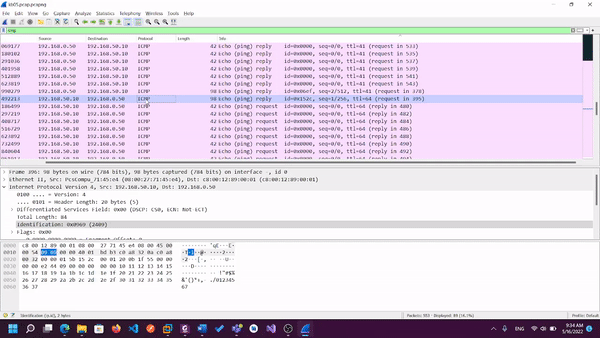
CDAISIWillWin . Search thông điệp này ra ta sẽ có mấy bài write-up :v

Không có manh mối gì khác, ta thử tập trung vào các gói **ICMP**

|  |
| --- |
| *icmp* |

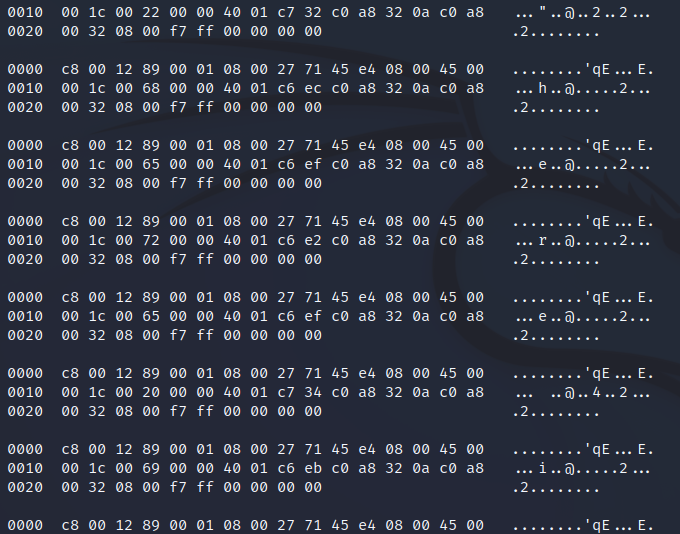


Ta để ý trường **Id** có gì đó lạ, giống như đang ẩn giấu thông tin gì đó - Ta lần lượt lướt các gói tin từ trên xuống dưới và để ý giá trị ở trường **Identifications** thay đổi liên tục và tạo thành message nào đó.



Dùng tshark để trích xuất thông tin & nhìn rõ hơn:

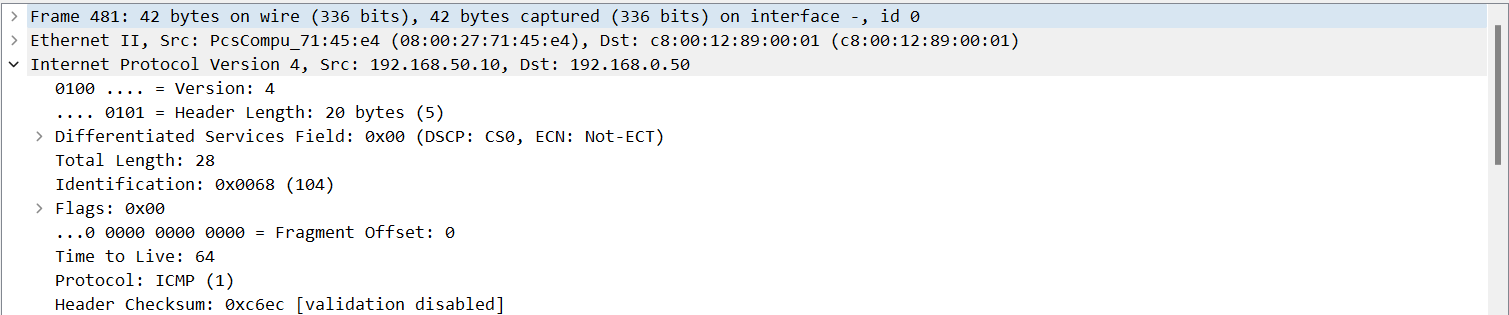
|  |
| --- |
| *tshark -r kb05.pcap.pcapng -x 'icmp and ip.src==192.168.50.10'* |



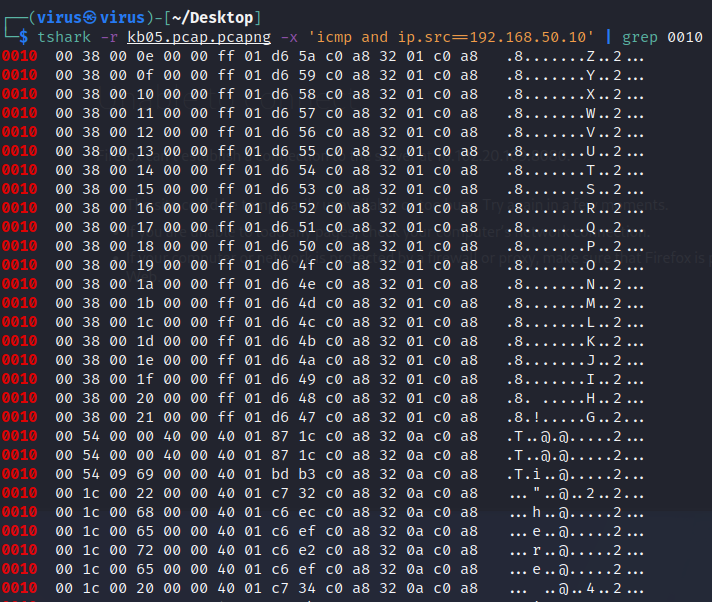
Ta để ý thấy các chữ cái “h” , “e”, “r” ,”e” ,”i” , … tạo thành một thông điệp và ta muốn lấy ở dòng có offset “0010” nên mình sẽ **grep** chuỗi này

|  |
| --- |
| *tshark -r kb05.pcap.pcapng -x 'icmp and ip.src==192.168.50.10' | grep 0010* |

* “0010” là giá trị version của IP (Ipv4) tại Tầng IP (Layer 3) trong gói tin



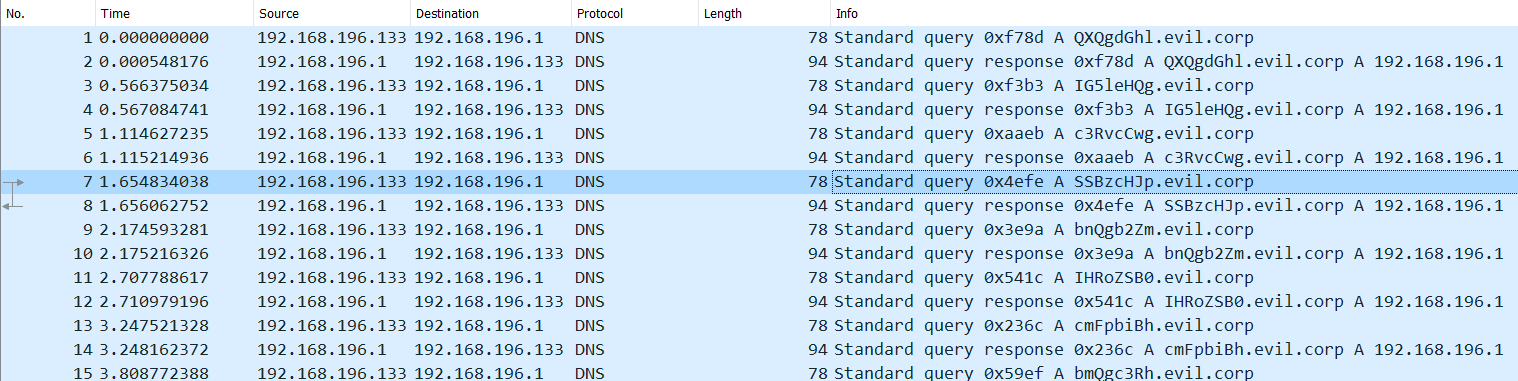
* Filter theo IP Source là **192.168.50.10** vì các gói tin này mang lại sự thay đổi và có thể tạo thành thông điệp như phân tích trên.



Flag: **S3cr3t4g3nt**

|  |
| --- |
| **Kịch bản 06. Điều tra trên dữ liệu lưu lượng mạng thu được.**  - Mô tả: Một trong các máy chủ của CoMix Wave Films bị xâm nhập vào tuần trước, tuy nhiên không có thiệt hại đáng kể nàọ được ghi nhận. Mặc dù hệ thống tường lửa của công ty rất mạnh nhưng nhóm bảọ mật của công ty phát hiện ra một số họạt động đáng ngờ, có thể bị tuồn dữ liệu ra bên ngọài. Hãy điều tra liệu kẻ tấn công đã lấy được những gì từ máy chủ của công ty, giao thức sử dụng? Tìm flag.  - Tài nguyên: Nandemonaiya\_kb06.pcap  ***Yêu cầu – Gợi ý:*** <https://bitbucket.org/kscrivs/netsec-0x325-> |

Mở file pcap bằng Wireshark, ta thấy trước mắt là rất nhiều gói tin DNS



Điều đáng ngờ ở đây là các query DNS trỏ đến tên miền **.evil.corp** (nghe như trong phim *Mr. Robot* @.@). Ta thấy có chuỗi trông giống base64 encode trên tên miền. Thử một chuỗi đầu xem sao:



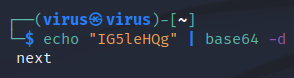
Vậy là một chuỗi có ý nghĩa, dùng tshark trích xuất ra toàn bộ rồi decode vài cái thử :

* Port 53 – DNS
* -2 filter qua 2 giai đoạn vào buffer
* dns.qry.name: lấy tên miền dns được query ra

|  |
| --- |
| *tshark -r Nandemonaiya\_kb06.pcapng -2 -R udp.dstport==53 -T fields -e "dns.qry.name" | grep "evil.corp"* |



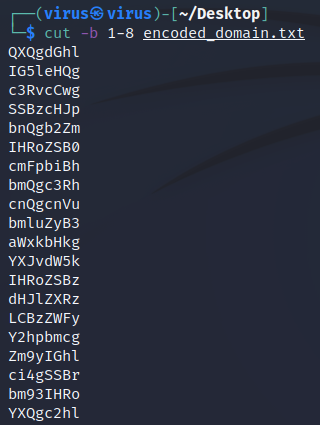
Thử decode base64 tiếp



OK, vậy giờ mình sẽ viết một payload command để decode hết những chuỗi trên một lúc. Đưa output tên domain vào 1 file riêng biệt (**encoded\_domain.txt**)

|  |
| --- |
| *tshark -r Nandemonaiya\_kb06.pcapng -2 -R udp.dstport==53 -T fields -e "dns.qry.name" | grep "evil.corp" > encoded\_domain.txt* |

Do output như mình thấy bên trên thì chuỗi base64 đều có length giống nhau trong các domain, vậy nên cũng tiện. Ta dùng command **cut**:

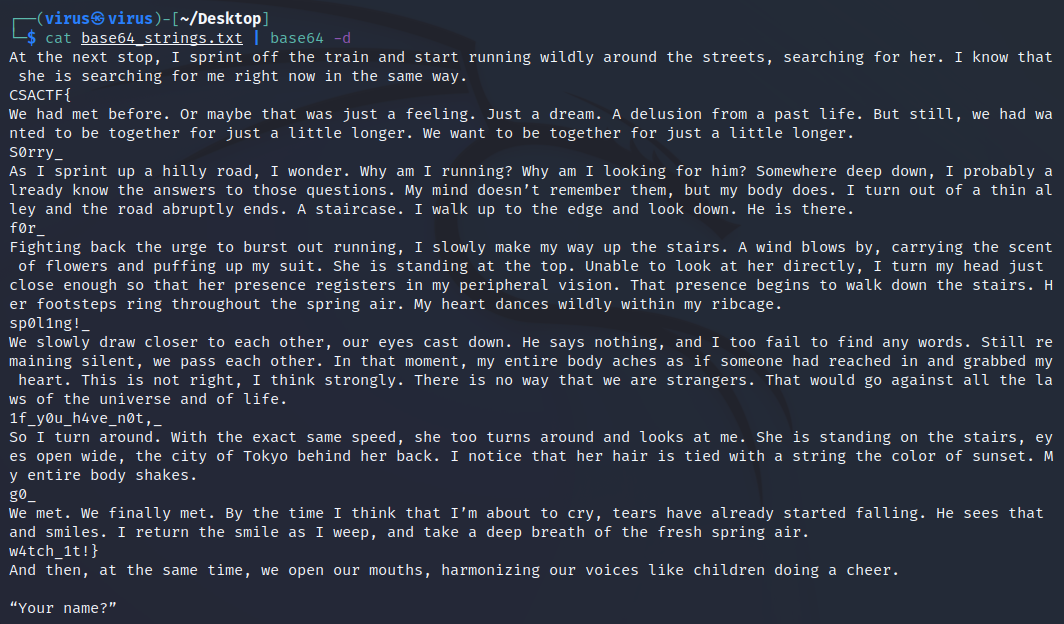


Ở đây mình sẽ đưa các giá trị này vào file *base64\_strings.txt*:

|  |
| --- |
| *cut -b 1-8 encoded\_domain.txt > base64\_strings.txt* |

Sau đó decode từng dòng trên file này:

|  |
| --- |
| *cat base64\_strings.txt | base64 -d* |



Flag là các segment rời rạc được chèn vào và đứng thành một dòng riêng lẻ, đó cũng là một lời xin lỗi muộn màng tương tư. Nhưng mà chúng ta đảm bảo rằng attacker sẽ bị vô tù 💖

Flag: **CSACTF{S0rry\_f0r\_sp0l1ng!\_ 1f\_y0u\_h4ve\_n0t,\_ g0\_w4tch\_1t!}**