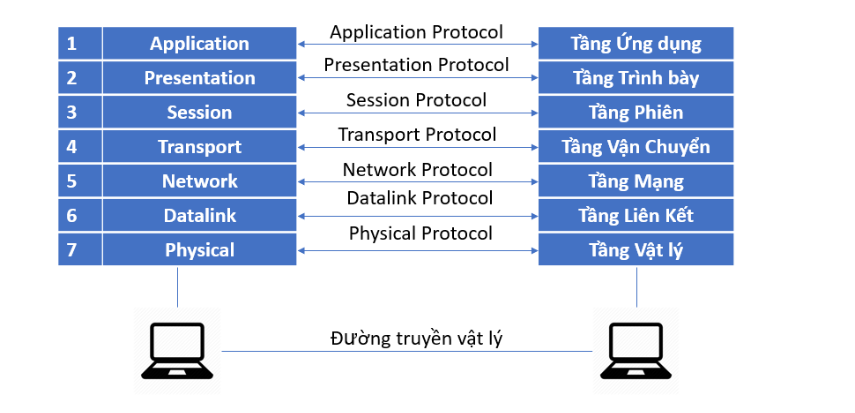
Task 4 : NETWORKING

1. Mô hình OSI:

\_Mô hình kết nối các hệ thống mở OSI là mô hình căn bản về các tiến trình truyền thông, thiết lập các tiêu chuẩn kiến trúc mạng ở mức Quốc tế, là cơ sở chung để các hệ thống khác nhau có thể liên kết và truyền thông được với nhau. Mô hình OSI tổ chức các giao thức truyền thông thành 7 tầng, mỗi một tầng giải quyết một phần hẹp của tiến trình truyền thông, chia tiến trình truyền thông thành nhiều tầng và trong mỗi tầng có thể có nhiều giao thức khác nhau thực hiện các nhu cầu truyền thông cụ thể.



\_Mô hình OSI gồm 7 tầng giao thức với các nguyên tắc sau :

+ Các tầng có tính độc lập tương đối với nhau thực hiện các chức năng riêng biệt

+ Cho phép thay đổi chức năng hoặc giao thức trong một tầng không làm ảnh hưởng đến các tầng khác.

+ Có thể chia một tầng thành các tầng con khi cần thiết.

+ Cho phép huỷ bỏ các tầng con nếu thấy không cần thiết.

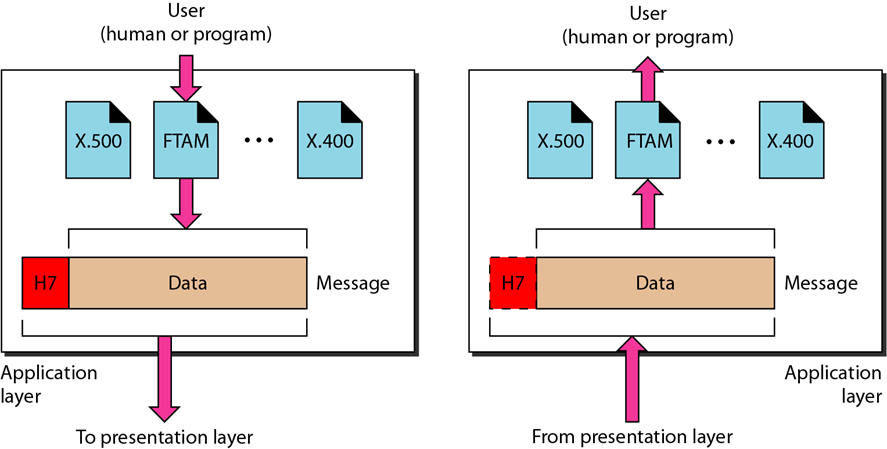
+ Bảo đảm liên kết cho nhiều hệ thống mạng khác nhau

+ Thích ứng với nhu cầu phát triển các công nghệ mới trong tương lai…

1. Tầng Application(7) :

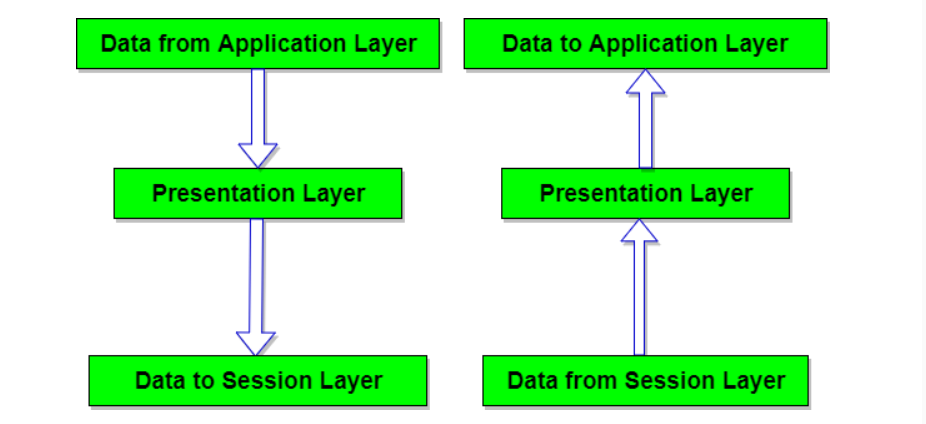
\_Tầng ứng dụng cho phép người sử dụng, phần mềm truy cập vào mạng.

\_Cung cấp giao diện NSD và hỗ trợ cho các dịch vụ như mail, truy cập/truyền file, chia sẻ CSDL và các dịch phân tán khác.



1. Tầng Presentation(6) :

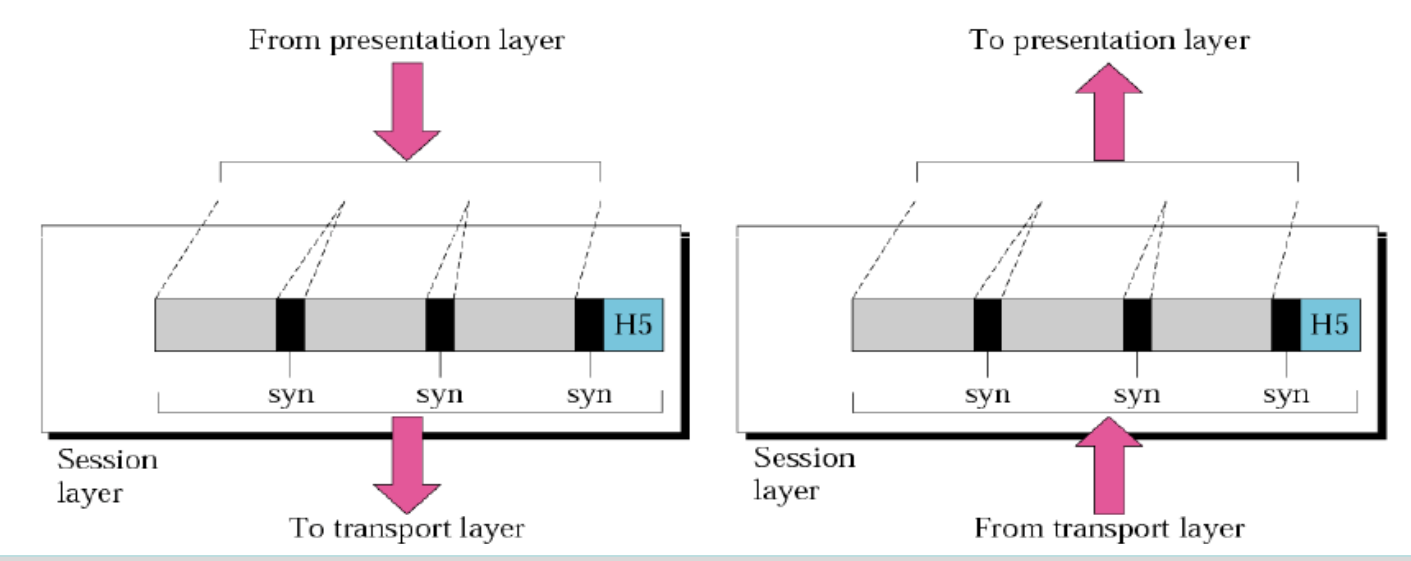
\_Tầng trình bày giải quyết các vấn đề liên quan đến các cú pháp và ngữ nghĩa của thông tin được truyền. Biểu diễn thông tin người sử dụng phù hợp với thông tin làm việc của mạng và ngược lại. Thông thường biểu diễn thông tin các ứng dụng nguồn và ứng dụng đích có thể khác nhau bởi các ứng dụng được chạy trên các hệ thống có thể khác nhau. Tầng trình bày phải chịu trách nhiệm chuyển đổi dữ liệu gửi đi trên mạng từ một loại biểu diễn này sang một loại biểu diễn khác. Để đạt được điều đó nó cung cấp một dạng biểu diễn truyền thông chung cho phép chuyển đổi từ dạng biểu diễn cục bộ sang biểu diễn chung và ngược lại.



1. Tầng Session(5) :

\_Tầng phiên cho phép người sử dụng trên các máy khác nhau thiết lập, duy trì và đồng bộ phiên truyền thông giữa họ với nhau. Nói cách khác tầng phiên thiết lập “các giao dịch” giữa các thực thể đầu cuối.

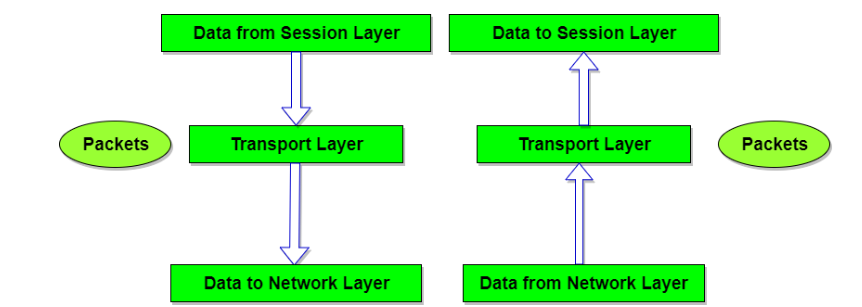
\_Dịch vụ phiên cung cấp một liên kết giữa 2 đầu cuối sử dụng dịch vụ phiên sao cho trao đổi dữ liệu một cách đồng bộ và khi kết thúc thì giải phóng liên kết. Sử dụng thẻ bài (Token) để thực hiện truyền dữ liệu, đồng bộ hóa và hủy bỏ liên kết trong các phương thức truyền đồng thời hay luân phiên. Thiết lập các điểm đồng bộ hóa trong hội thoại. Khi xảy ra sự cố có thể khôi phục hội thoại bắt đầu từ một điểm đồng bộ hóa đã thỏa thuận.



1. Tầng Transport(4) :

\_Là tầng cao nhất liên có liên quan đến các giao thức trao đổi dữ liệu giữa các hệ thống mở, kiểm soát việc truyền dữ liệu từ nút tới nút (End-to-End). Thủ tục trong 3 tầng dưới (vật lý, liên kết dữ liệu và mạng) chỉ phục vụ việc truyền dữ liệu giữa các tầng kề nhau trong từng hệ thống. Các thực thể đồng tầng hội thoại, thương lượng với nhau trong quá trình truyền dữ liệu.

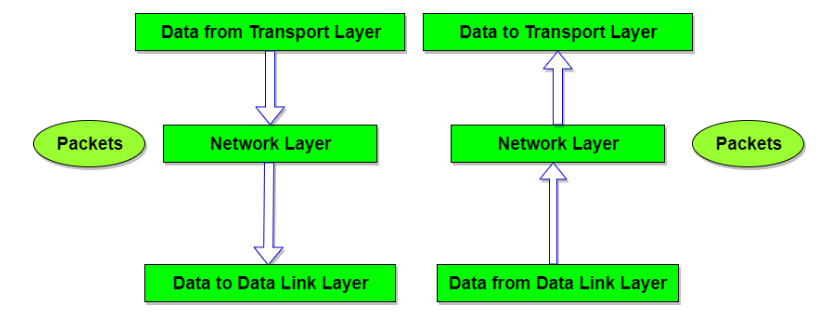
\_Tầng vận chuyển thực hiện việc chia các gói tin lớn thành các gói tin nhỏ hơn trước khi gửi đi và đánh số các gói tin và đảm bảo chúng chuyển theo đúng thứ tự. Là tầng cuối cùng chịu trách nhiệm về mức độ an toàn trong truyền dữ liệu nên giao thức tầng vận chuyển phụ thuộc nhiều vào bản chất của tầng mạng. Tầng vận chuyển có thể thực hiện việc ghép kênh (multiplex) một vài liên kết vào cùng một liên kết nối để giảm giá thành.



1. Tầng Network(3) :

\_Tầng mạng thực hiện các chức năng chọn đường đi (routing) cho các gói tin nguồn tới đích có thể trong cùng một mạng hoặc khác mạng nhau. Đường có thể được cố định, cũng có thể được định nghĩa khi bắt đầu hội thoại và có thể đường đi là động (Dynamic) có thể thay đổi với từng gói tin tùy theo trạng thái tải tức thời của mạng. Trong mạng kiểu quảng bá (Broadcast) routing rất đơn giản.

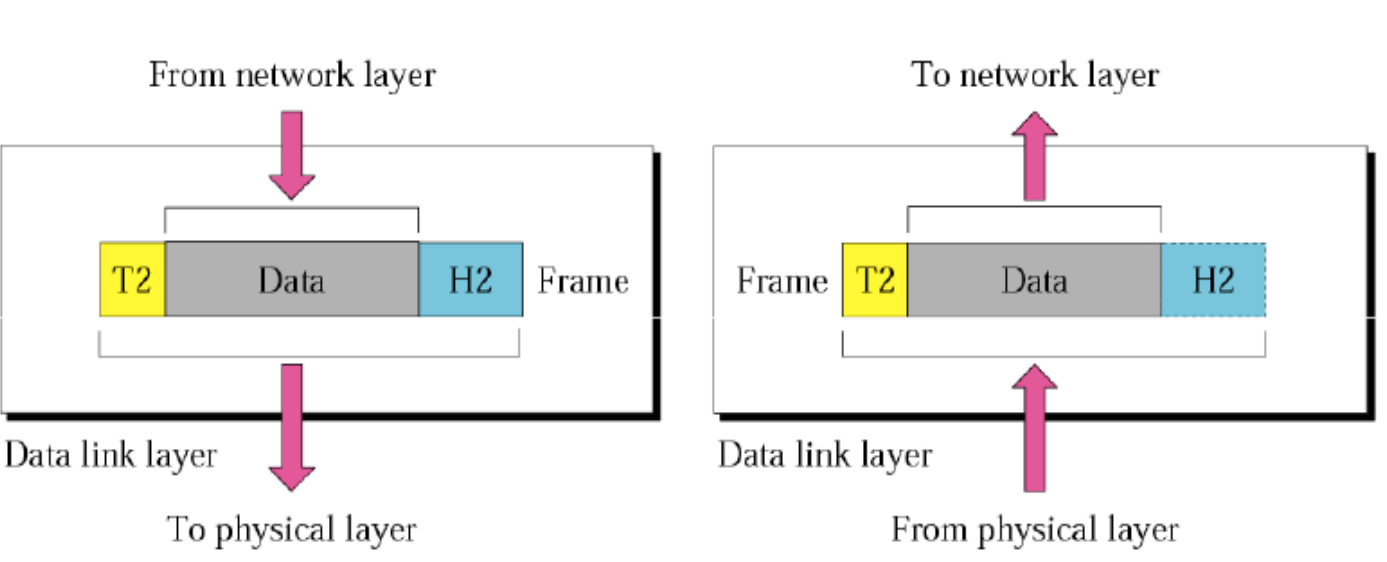
\_Một chức năng quan trọng khác của tầng mạng là chức năng điều khiển tắc nghẽn (Congestion Control). Nếu có quá nhiều gói tin cùng lưu chuyển trên cùng một đường thì có thể xảy ra tình trạng tắc nghẽn. Thực hiện chức năng giao tiếp giữa các mạng khi các gói tin đi từ mạng này sang mạng khác để tới đích.



1. Tầng Data Link(2) :

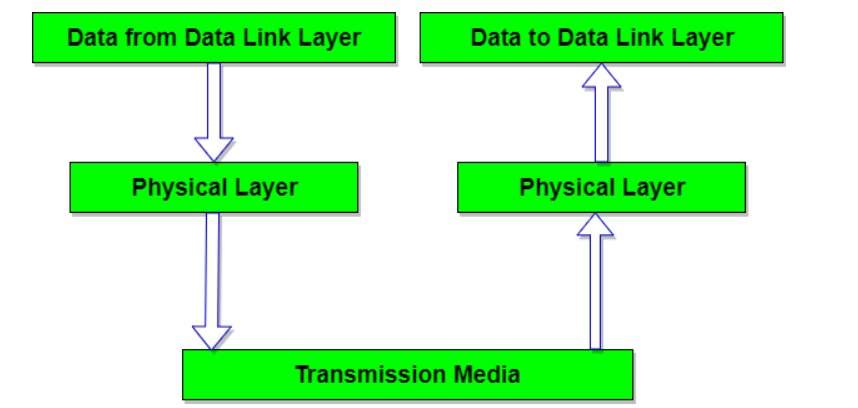
\_Chức năng chủ yếu của tầng liên kết dữ liệu là thực hiện thiết lập các liên kết, duy trì và hủy bỏ các liên kết dữ liệu. Kiểm soát lỗi (CRC) và kiểm soát lưu lượng.

\_Chia thông tin thành các khung thông tin (Frame), truyền các khung tuần tự và xử lý các thông điệp xác nhận (Acknowledgement Frame) từ bên máy thu gửi về. Tháo gỡ các khung thành chuỗi bit không cấu trúc chuyển xuống tầng vật lý. Tầng 2 bên thu, tái tạo chuỗi bit thành các khung thông tin. Đường truyền vật lý có thể gây ra lỗi, nên tầng liên kết dữ liệu phải giải quyết vấn đề kiểm soát lỗi, kiểm soát luồng, kiểm soát lưu lượng, ngăn không để nút nguồn gây “ngập lụt” dữ liệu cho ben thu có tốc độ thấp hơn. Trong các mạng quảng bá, tầng con MAC (Medium Access Sublayer) điều khiển việc duy trì nhập đường truyền.



1. Tầng Physical(1) :

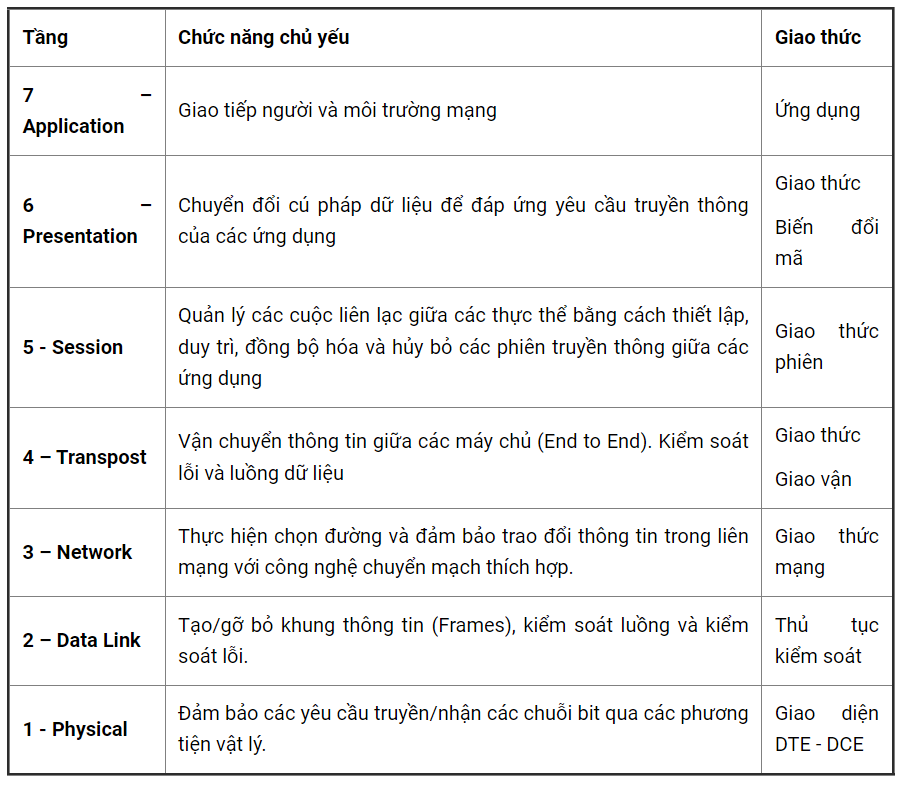
\_Tầng vật lý là tầng thấp nhất trong mô hình 7 lớp OSI. Các thực thể tầng giao tiếp với nhau qua một đường truyền vật lý. Tầng vật lý xác định các chức năng, thủ tục về điện, cơ, quang để kích hoạt, duy trì và giải phóng các kết nối vật lý giữa các hệ thống mạng. Cung cấp các cơ chế về điện, hàm, thủ tục, … nhằm thực hiện việc kết nối các phần tử của mạng thành một hệ thống bằng các phương pháp vật lý. Đảm bảo cho các yêu cầu về chuyển mạch hoạt động nhằm tạo ra các đường truyền thực cho các chuỗi bit thông tin. Các chuẩn trong tầng vật lý là các chuẩn xác định giao diện người sử dụng và môi trường mạng. Các giao thức tầng vật lý có hai loại: truyền dị bộ (Asynchronous) và truyền đồng bộ (Synchronous).



\* Độ dài Các Trường Dữ Liệu :

|  |  |
| --- | --- |
| **1.Physical** | Độ dài trường dữ liệu không giới hạn nhưng phụ thuộc vào phương tiện truyền thông sử dụng như cáp đồng trục, cáp xoắn đôi hoặc sóng vô tuyến. |
| **2.Data Link** | Độ dài trường dữ liệu của tầng này là từ 46 đến 1500 byte. |
| **3.Network** | Độ dài trường dữ liệu của tầng này là từ 20 đến 60 byte. |
| **4.Transport** | Độ dài trường dữ liệu của tầng này là từ 20 đến 60 byte. |
| **5.Seesion** | Độ dài trường dữ liệu của tầng này là không giới hạn. |
| **6.Presentation** | Độ dài trường dữ liệu của tầng này là không giới hạn. |
| **7.Application** | Độ dài trường dữ liệu của tầng này là không giới hạn. |

\* Tóm Tắt Chức Năng Của Các Tầng Trong Mô Hình OSI :



\* Cấu Trúc Thành Phần các gói tin trong mô hình OSI :

\_Lớp Vật lý (Physical Layer): Các thành phần của gói tin ở lớp này bao gồm bit (0 hoặc 1), đường truyền, tần số sóng, độ dài sóng, điện áp...

\_Lớp Liên kết dữ liệu (Data Link Layer): Các thành phần của gói tin ở lớp này bao gồm khung (frame), địa chỉ MAC (Media Access Control), kiểm soát lỗi, chuyển tiếp gói tin, đồng bộ hóa.

\_Lớp Mạng (Network Layer): Các thành phần của gói tin ở lớp này bao gồm giao thức IP (Internet Protocol), địa chỉ IP, bảng định tuyến, giao thức truyền tải...

\_Lớp Giao vận (Transport Layer): Các thành phần của gói tin ở lớp này bao gồm các giao thức như TCP (Transmission Control Protocol), UDP (User Datagram Protocol), cổng (port)...

\_Lớp Phiên (Session Layer): Lớp này thiết lập, quản lý và giải phóng các kết nối phiên làm việc giữa các ứng dụng trên các thiết bị khác nhau. Các thành phần của gói tin ở lớp này bao gồm các yêu cầu bắt đầu phiên, yêu cầu kết thúc phiên, yêu cầu đồng bộ hóa phiên, đồng hồ phiên.

\_Lớp trình diễn (Presentation layer): Ở lớp này, thành phần của gói tin bao gồm dữ liệu được mã hóa và định dạng để có thể hiển thị trên các thiết bị khác nhau.

\_Lớp ứng dụng (Application layer): Tại lớp này, thành phần của gói tin bao gồm các dữ liệu và thông tin điều khiển của ứng dụng.

\*Các giao thức TCP/IP, UDP, HTTP, HTTPS và SMB là các giao thức mạng được sử dụng để truyền tải dữ liệu giữa các thiết bị trong mạng máy tính. Dưới đây là mô tả ngắn gọn về mỗi giao thức này:

\_TCP/IP(TransmissionControlProtocol/Internet Protocol): Là một bộ giao thức được sử dụng để kết nối các thiết bị trên Internet hoặc mạng cục bộ. Giao thức TCP được sử dụng để đảm bảo rằng dữ liệu được truyền tải đến đúng địa chỉ và đảm bảo tính toàn vẹn của dữ liệu. Giao thức IP được sử dụng để định tuyến dữ liệu từ nguồn đến đích.

\_UDP(User Datagram Protocol): Là một giao thức truyền tải dữ liệu dạng gói (packet) được sử dụng để truyền tải dữ liệu nhanh và hiệu quả hơn TCP. Tuy nhiên, UDP không đảm bảo tính toàn vẹn và độ tin cậy của dữ liệu.

\_HTTP(Hypertext Transfer Protocol): Là giao thức được sử dụng để truyền tải các trang web và dữ liệu khác trên Internet. HTTP sử dụng phương thức yêu cầu/đáp ứng (request/response) giữa máy khách (client) và máy chủ (server) để truyền tải dữ liệu.

\_HTTPS(Hypertext Transfer Protocol Secure): Là phiên bản an toàn của HTTP, sử dụng mã hóa SSL/TLS để bảo vệ dữ liệu khi truyền tải giữa máy khách và máy chủ.

\_SMB(Server Message Block): Là giao thức được sử dụng để chia sẻ tài nguyên và truyền tải tập tin giữa các thiết bị trong mạng Windows. SMB được sử dụng để chia sẻ tập tin, máy in, thư mục và các tài nguyên khác trên mạng.

\_FTP(File Transfer Protocol): Là giao thức được sử dụng để truyền tải các tệp từ máy tính này sang máy tính khác trong mạng. Nó cho phép người dùng tải lên và tải xuống các tệp từ máy chủ FTP.

\_DNS(Domain Name System): Là giao thức được sử dụng để chuyển đổi tên miền thành địa chỉ IP của máy chủ đó. Đây là giao thức quan trọng trong việc liên kết các tên miền với các địa chỉ IP của các máy chủ.

\_DHCP(Dynamic Host Configuration Protocol): Là giao thức được sử dụng để cấu hình các thiết bị mạng, như máy tính, để tự động nhận địa chỉ IP và các thông số mạng khác từ máy chủ DHCP.

\_SNMP(Simple Network Management Protocol): Là giao thức được sử dụng để quản lý các thiết bị mạng, như định vị và giám sát các thiết bị trong mạng.

\_ICMP(Internet Control Message Protocol): Là giao thức được sử dụng để gửi các thông báo lỗi và thông báo kiểm tra mạng giữa các thiết bị trong mạng.

\_OSPF(Open Shortest Path First): Là giao thức định tuyến được sử dụng để tính toán đường đi ngắn nhất giữa các thiết bị trong mạng. Nó được sử dụng chủ yếu trong các mạng lớn và phức tạp.

\_BGP(Border Gateway Protocol): Là giao thức định tuyến được sử dụng để kết nối các mạng lớn với nhau và truyền tải thông tin định tuyến giữa các nhà cung cấp dịch vụ Internet.

\_SSL/TLS(Secure Sockets Layer/Transport Layer Security): Là giao thức được sử dụng để bảo mật các thông tin nhạy cảm như mật khẩu, thông tin tài khoản ngân hàng và thẻ tín dụng khi truyền tải trên Internet. Các trình duyệt web hiện đại đều hỗ trợ SSL/TLS và nhiều trang web sử dụng SSL/TLS để đảm bảo an toàn cho người dùng.

1. Wireshark :
2. Wireshark là gì ?

\_Wireshark là ứng dụng phân tích mạng (network packet analyzer). Công dụng của ứng dụng này là dùng để bắt, phân tích và xác định các vấn đề có liên quan đến network bao gồm: kết nối chậm, rớt gói tin hoặc các truy cập bất thường.

\_Thông qua Wireshark, quản trị viên có thể hiểu hơn về các Network Packets đang chạy trên hệ thống. Như vậy, việc xác định nguyên nhân gây ra lỗi cũng sẽ dễ dàng hơn.

1. Ứng Dụng Wireshark dùng để làm gì ?

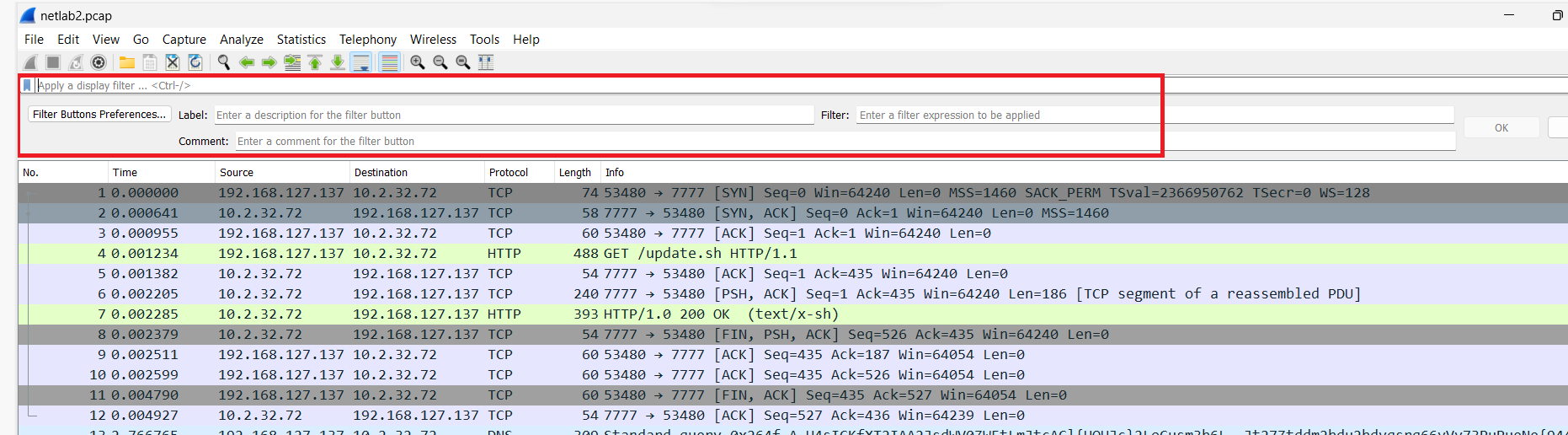
* Trước hết, Wireshark được Network administrators sử dụng trong việc khắc phục sự cố về mạng.
* Bên cạnh đó, Wireshark còn được các kỹ sư Network security dùng để kiểm tra các vấn đề liên quan đến bảo mật.
* Trong khi đó thì Wireshark lại được các kỹ sư QA sử dụng để xác minh các network applications.
* Và các developers dùng Wireshark trong việc gỡ lỗi triển khai giao thức.
* Còn đối với người dùng mạng máy tính bình thường thì Wireshark giúp chúng ta học internals giao thức mạng.
* Ngoài ra, Wireshark còn được sử dụng trong rất nhiều tình huống thực tế khác nữa mà chỉ những người trong giới chuyên môn mới biết câu trả lời.

1. Tính năng của Wireshark :

* Ứng dụng này giúp người dùng có thể chụp dữ liệu gói trực tiếp từ giao diện mạng.
* Thực hiện mở các tệp có chứa dữ liệu gói bằng tcpdump/ WinDump, Wireshark cũng như một số chương trình packet capture khác.
* Nhập các gói từ các tệp văn bản có chứa các hex dumps của packet data.
* Hiển thị các gói thông tin một cách vô cùng chi tiết.
* Tiến hành việc lưu trữ tất cả các dữ liệu gói đã bị bắt.
* Xuất một số hoặc tất cả các gói thông qua định dạng capture file.
* Dựa vào các tiêu chí khác nhau để lọc các gói tin.
* Dựa trên nhiều tiêu chí để tìm kiếm các gói.
* Colorize là gói hiển thị dựa trên bộ lọc.
* Wireshark còn giúp tạo các số liệu thống kê khác nhau.

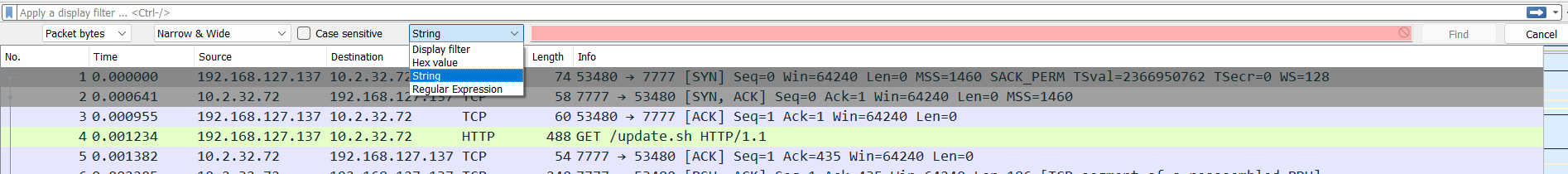
1. Phân Tích Bằng Wireshark :
2. Tìm kiếm gói tin (Find Packet)

\_Để tìm kiếm gói tin, chúng ta có thể sử dụng thanh công cụ “Find Packet”



****\* Chúng ta có thể tìm kiếm packet dựa vào(Ctrl+F):****

* ****Display Filter:**** nhập vào một biểu thức filter (expression-based filter), Wireshark sẽ tìm kiếm các gói tin khớp với biểu thức này.
* ****Hex value:**** Tìm kiếm dựa trên giá trị Hex.****String:**** tìm kiếm dựa trên chuỗi dữ liệu.
* ****Regular Expression:**** Tìm kiếm dựa trên biểu thức Regex.



| **Options** | **Ví dụ** |
| --- | --- |
| Display filter | tcp.src port==80 hoặc ip.src==192.168.1.1 |
| Hex | 010108ffff |
| String | Flag hoặc GET / |
| Regular Expression | GET .\* HTTP |
|  |  |

1. **Wireshark Capture Filter sử dụng cú pháp của Berkeley Packet Filter (BPF):**

* Mỗi filter gọi là một expression.
* Mỗi expression chứa một hoặc nhiều primitives. Các primitives được kết hợp với nhau bằng các “Logical Operator” như AND (&&), OR (||) và NOT (!) .
* Mỗi ****primitives****chứa một hoặc nhiều ****qualifiers****, theo sau là một ID name hoặc number. Các BPF Qualifiers bao gồm:

| **Qualifiers** | **Mô tả** | **Ví dụ** |
| --- | --- | --- |
| ****Type**** | Chỉ định ID name hoặc number ta sẽ tham chiếu | host, net, port |
| ****Dir**** | Chỉ định hướng của dữ liệu (transfer direction) | src, dst |
| ****Proto**** | Protocol | ether, ip, tcp, udp, http, ftp |

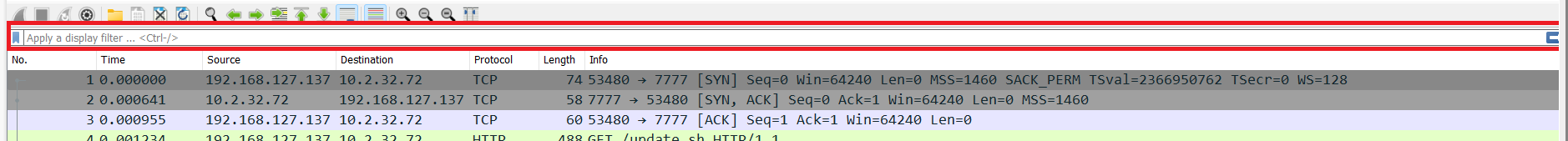
****\*Một vài Wireshark Expression tham khảo cho phần Capture Filter:****

| **Expression** | **Ý nghĩa** |
| --- | --- |
| host 172.18.5.4 | Wireshark filter by IP: Bắt gói tin liên quan đến IP 172.18.5.4 |
| src 192.168.0.10 | Wireshark filter source IP: Bắt các gói tin có source IP là 192.168.0.10 |
| dst 192.168.0.1 | Wireshark filter destination IP: Bắt các gói tin có destination IP là 192.168.0.1 |
| net 192.168.0.0/24 hoặc: net 192.168.0.0 mask 255.255.255.0 | Bắt gói tin liên quan đến subnet 192.168.0.0/24 |
| src net 192.168.0.0/24 hoặc: src net 192.168.0.0 mask 255.255.255.0 | Bắt các gói tin có source IP thuộc subnet 192.168.0.0/24 |
| dst net 192.168.0.0/24 hoặc: dst net 192.168.0.0 mask 255.255.255.0 | Bắt các gói tin có destination IP thuộc subnet 192.168.0.0/24 |
| port 53 | Wireshark port filter: Bắt gói tin DNS |
| port 67 or port 68 | Bắt gói tin DHCP |
| host 192.168.1.1 and not (port 80 or 443) hoặc: host 192.168.1.1 and not port 80 and not port 443 | Capture tất cả traffic liên quan đến IP 192.168.1.1 nhưng không phải traffic HTTP/HTTPS |
| (tcp[0:2] > 1500 and tcp[0:2] < 1550) or (tcp[2:2] > 1500 and tcp[2:2] < 1550) hoặc: tcp portrange 1501-1549 | Capture các packet nằm trong range port từ 1501-1549 |
| ip | Wireshark IPv4 filter |
| ip6 | Wireshark IPv6 filter |
| tcp | Bắt gói tin TCP |
| udp | Bắt gói tin UDP |
| icmp | Bắt gói tin ICMP |
| http | Wireshark HTTP filter |
| https | Wireshark HTTPS filter |
| tcp[13] & 32 == 32 | TCP packets với cờ URG được bật |
| tcp[13] & 16 == 16 | TCP packets với cờ ACK được bật |
| tcp[13] & 8 == 8 | TCP packets với cờ PSH được bật |
| tcp[13] & 4 == 4 | TCP packets với cờ RST được bật |
| tcp[13] & 2 == 2 | TCP packets với cờ SYN được bật |
| tcp[13] & 1 == 1 | TCP packets với cờ FIN được bật |
| icmp[0:2] == 0x0301 | ICMP destination unreachable, host unreachable |

### C.**Display Filter**

\_Display Filter giúp lọc ra những packet thỏa điều kiện trong file capture để thể hiện lên cho người dùng. Display filter chỉ lọc và thể hiện packet thỏa điều kiện chứ không xóa bỏ những packet không thỏa điều kiện, dữ liệu trong file capture hoàn toàn không bị ảnh hưởng.

\_Sử dụng Display Filter bằng cách nhập biểu thức (expression) vào Filter textbox phía trên phần Packet List. Bạn cũng có thể nhấp vào phần “Expression” để lựa chọn các pre-defined filters có sẵn ứng với từng giao thức.



****\* Cú pháp của Wireshark Display Filter phần lớn tuân theo cú pháp:****

protocol.feature.subfeature COMPARISION\_OPERATOR value LOGICAL\_OPERATOR protocol.feature.subfeature COMPARISION\_OPERATOR value

\* Trong đó, ****Comparison Operators**** bao gồm:

| **Operator** | **Ý nghĩa** |
| --- | --- |
| == | Bằng (equal to) |
| != | Không bằng (not equal to) |
| > | Lớn hơn (greater than) |
| < | Nhỏ hơn |
| >= | Lớn hơn hoặc bằng (greater than or equal) |
| <= | Nhỏ hơn hoặc bằng (less than or equal) |

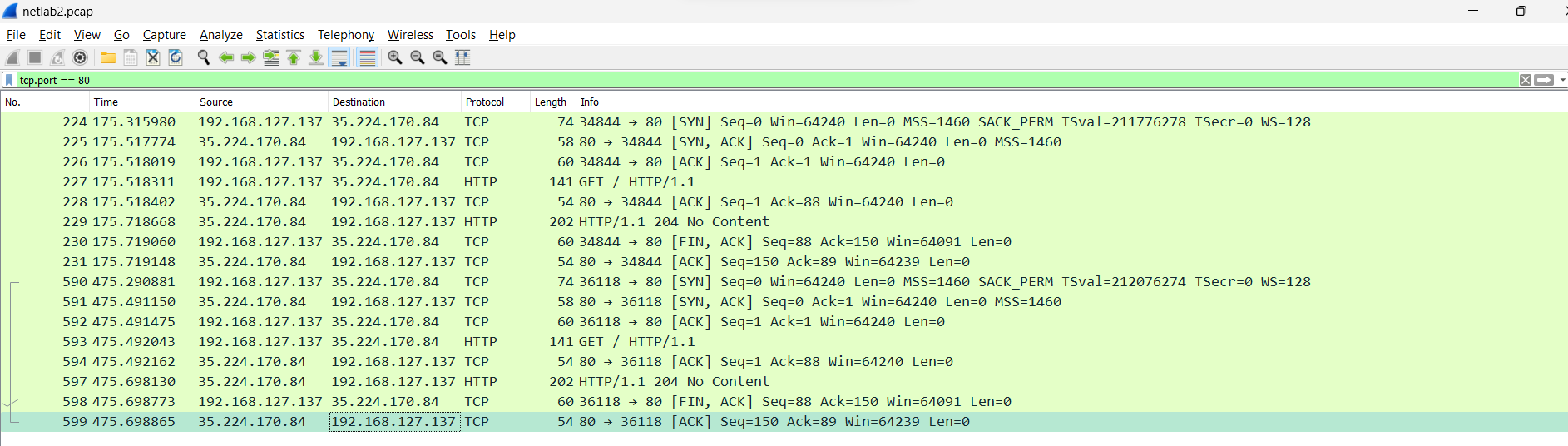
****\* Logical Operators****bao gồm:

| **Operator** | **Ý nghĩa** |
| --- | --- |
| and | tất cả các điều kiện phải được thỏa mãn |
| or | một trong các điều kiện được thoả mãn |
| xor | một và chỉ một điều kiện được thỏa mãn |
| not | không điều kiện nào được phép thảo mãn |

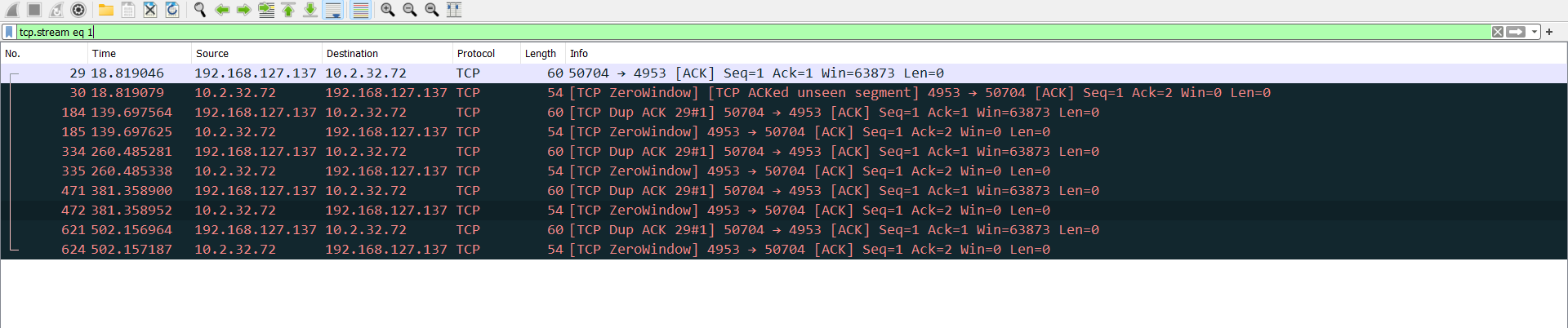
#EX :

+ Hiển thị các gói được gửi qua cổng TCP 80, bao gồm lưu lượng HTTP.

|  |  |
| --- | --- |
| 1 | tcp.port == 80 |

+ Hiển thị các gói thuộc luồng TCP 1.

|  |  |
| --- | --- |
| 2 | Tcp.stream eq 1 |

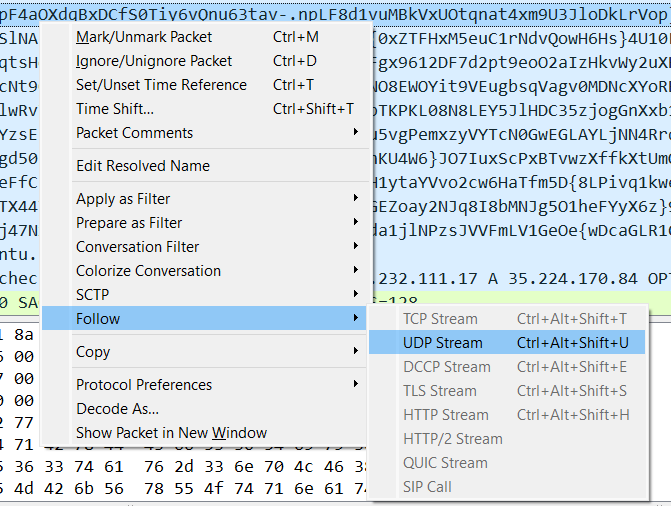
+ ****Một vài Wireshark Expression tham khảo cho phần Display Filter:****

| **Expression** | **Ý nghĩa** |
| --- | --- |
| tcp.port eq 25 or icmp | Lọc gói tin TCP liên quan port 25 hoặc sử dụng giao thức ICMP |
| ip.src==192.168.0.0/16 and ip.dst==192.168.0.0/16 | Lọc traffic trao đổi trong mạng LAN của subnet 192.168.0.0/16 |
| tcp.window\_size == 0 && tcp.flags.reset != 1 | TCP buffer full và source kết nối báo hiệu cho Destination ngừng gửi dữ liệu |
| udp contains 81:60:03 | UDP packet chứa 3 bytes 81:60:03 ở vị trí bất kỳ trong header hoặc payload |
| http.request.uri matches “gl=se$” | HTTP request có URL tận cùng bằng chuỗi “gl=se” |
| ip.addr == 192.168.0.1 hoặc: ip.src == 192.168.0.1 or ip.dst == 192.168.0.1 | Wireshark filter by ip: Lọc tất cả traffic liên quan đến IP 192.168.0.1 |
| ! ( ip.addr == 192.168.0.1 ) hoặc: ! (ip.src == 192.168.0.1 or ip.dst == 192.168.0.1) | Lọc tất cả traffic KHÔNG liên quan đến IP 192.168.0.1 |
| tcp.flags.syn == 1 | Các gói tin TCP có cờ SYN được bật |
| tcp.flags.syn == 1 && tcp.flags.ack == 1 | Các gói tin TCP có cờ SYN/ACK được bật |
| http.host == “quantrilinux.vn” | HTTP request có Host header là “quantrilinux.vn” |
| http.response.code == 404 | Các HTTP request có response status code là 404 |
| smtp || imap || pop | Traffic liên quan đến email (SMTP, IMAP, POP) |
| ! tcp.port == 22 | Loại bỏ traffic SSH |
| ! arp | Loại bỏ traffic ARP |
| ip.version == 4 | Wireshark IPv4 filter: Lọc tất cả các gói tin IP version 4 |
| tcp.srcport == 80 | Wireshark port filter: Lọc tất cả gói tin TCP có source port là 80 |
| tcp.port == 80 | Lọc tất cả các gói tin có liên quan đến port 80 |
| udp.port == 67 or udp.port == 68 | Traffic DHCP |
| dns | Filter traffic liên quan DNS |
| http | Wireshark http filter |
| https | Wireshark https filter |
| ip.src == 192.168.0.1 | Wireshark filter source ip |
| ip.dst == 192.168.0.1 | Wireshark filter destination ip |

\*Một tính năng hữu ích khác là ****“Follow TCP stream”****, chọn một packet rồi click chuột phải vào packet chọn****“Follow > TCP Stream”****. Sau đó, một hộp thoại sẽ hiện ra cho thấy dữ liệu trao đổi giữa Client và Server trong luồng tương ứng và các packet liên quan. Ta có thể click vào các giao thức khác trong menu Follow để xem đầy đủ các đoạn hội thoại, nếu có thể.



\* Ngoài ra còn có thể Follow nhiều giao thức khác :



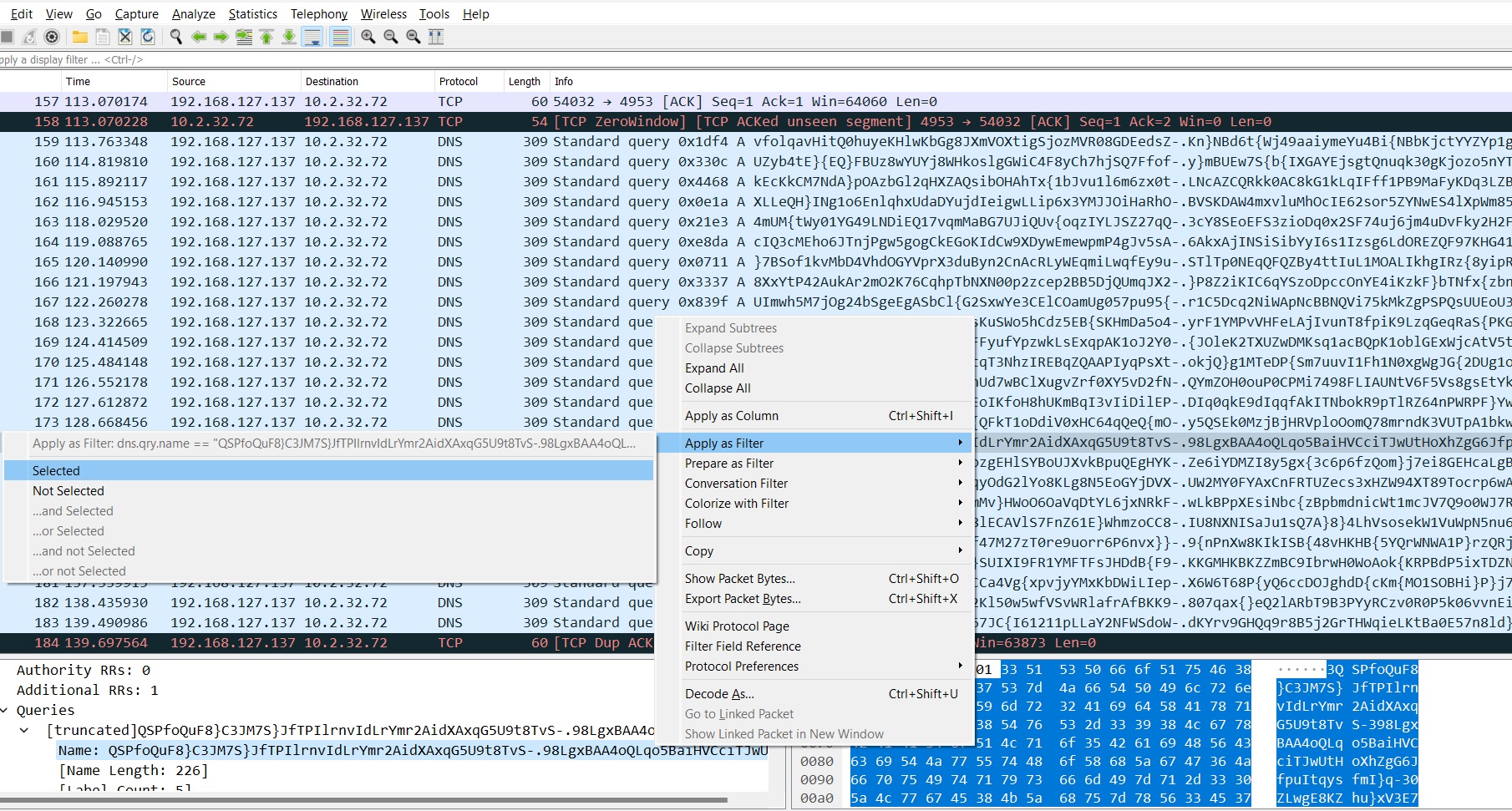
\* Ở khung cửa sổ Packet List sẽ cung cấp cho chúng ta các thông tin như:

* No: Số thứ tự của gói tin trong file capture hiện tại.
* Time: Thời gian tương đối mà gói tin này được bắt, tính từ lúc bắt đầu quá trình bắt gói tin.
* Source: địa chỉ source IP của kết nối.
* Destination: địa chỉ destination IP của kết nối.
* Length: chiều dài của gói tin.
* Protocol: giao thức của gói tin
* Info: các thông tin tổng quan liên quan đến gói tin.

\* Ở khung cửa sổ của Packet Details sẽ cho ta thông tin chi tiết từng Layer của packet như:

* Frame: Interface
* Ethernet: Destination, Source, Mac Address
* Internet: Source IP, Destination IP, TTL, Protocol, Flags, Checksum, Identification, Total Length…
* TCP/UDP/ICMP: Source Port, Destination Port, Sequence Number, ACK Number, Flags, TCP Options …
* Application Layer: HTTP, DNS, SMTP…

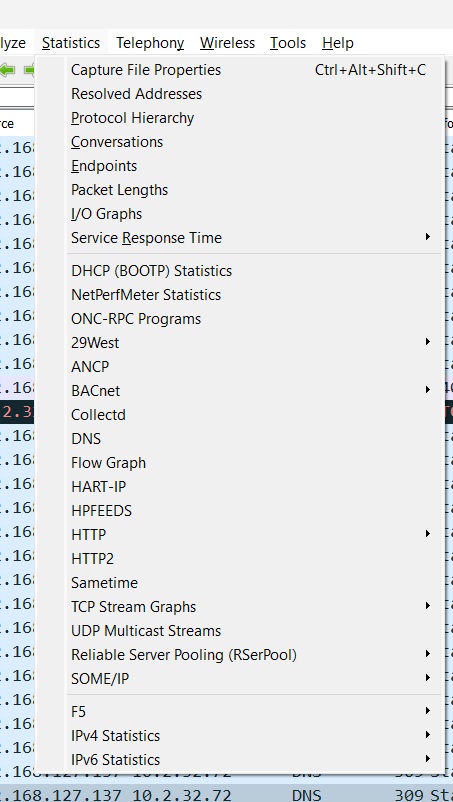
\* Ta cũng có thể tạo thêm các filter ở đây. Chỉ cần click chuột phải vào một trong số các chi tiết, rồi chọn ****Apply as Filter**** để tạo một filter dựa theo đó.



#### \***Wireshark Statistics**

\_Phần menu Statistic cung cấp những thông tin thống kê có giá trị liên quan đến file capture hiện tại như:

* Capture File Properties: Các thông số tổng quan của file capture
* Protocol Hierarchy: Tổng quan về protocol
* Conversation: thông tin về các luồng trao đổi giữa client và server
* Endpoints: Danh sách những IP tham gia kết nối, số lượng packets và bytes tương ứng.
* Packet Lengths: Thống kê về chiều dài của các gói tin tham gia kết nối.
* I/0 Graph: biểu đồ kết nối.
* Thống kê liên quan đến các giao thức như: HTTP, HTTP2, DNS, DHCP…
* IPv4/IPv6 Statistics: thống kê về danh sách IP, số lượng packet, tần số kết nối.



1. Tshark là gì ?

\_TShark là công cụ phân tích giao thức mạng, một phiên bản terminal của Wireshark được sử dụng khi giao diện đồ họa không có sẵn hoặc không cần thiết. Với TShark bạn có thể bắt gói tin, đọc/phân tích gói tin từ file đã lưu…

\* Một số lệnh trong wireshark :

1.**Để xem danh sách network interfaces trên máy tính hiện tại ta sử dụng lệnh:**

Mã:

tshark -D

2.**Bắt và lưu gói tin**  
  
Sau khi xem danh sách network interfaces ở bước trên chúng ta sẽ tiến hành bắt và lưu gói tin với tham số -i và -w. File sẽ được lưu dưới dạng .pcap

Mã:

tshark -i eth0 -w vidu.pcap

3.Với lệnh trên sẽ bắt tất cả gói tin từ network interface eth0 và ghi ra file vidu.pcap cho đến khi nhấn Ctrl-C để dừng. Để bắt một số lượng gói tin nhất định ta có thể dùng tham số -c

Mã:

tshark -i eth0 -c 69 -w vidu.pcap

4.**Đọc file pcap**  
Với tham số -r ta có thể xem nội dung file .pcap đã lưu.

Mã:

tshark -r vidu.pcap

5.**Phân tích HTTP request**  
Trong ví dụ sau chúng ta sẽ bắt gói tin từ network interface eth0 và lọc ra HTTP requests bằng tham số -Y http.request kết hợp tham số -T và -e để lọc dữ liệu là các trường http.host và http.user\_agent

Mã:

tshark -i eth0 -Y http.request -T fields -e http.host -e http.user\_agent

6.**Kết hợp Tshark với lệnh shell để thống kê, phân tích dữ liệu**  
  
Trong ví dụ sau ta sử dụng tham số -Y bên trên để lọc dữ liệu từ một file .pcap đã kết xuất trước đó kết hợp với các lệnh shell cơ bản để nhanh chóng đếm tần suất xuất hiện của các user agent.  
Từ kết quả có được chúng ta có thể dùng để phân tích lưu lượng mạng để phát hiện những truy cập bất thường, mã độc...

Mã:

tshark -r vidu.pcap -Y http.request -T fields -e http.host -e http.user\_agent | sort | uniq -c | sort -n

6.**Phân tích truy vấn DNS**  
Để bắt gói tin và lọc ra các truy vấn DNS và địa chỉ phản hồi ta dùng lệnh sau

Mã:

tshark -i eth0 -f "src port 53" -n -T fields -e dns.qry.name -e dns.resp.addr

7.**Dùng Tshark để... lấy mật khẩu**  
Bằng cách sử dụng Tshark để bắt gói tin, kết hợp tùy chọn lọc**tcp contains "password"** và lệnh **grep** trong Linux ta sẽ có một vài thứ hay ho như mật khẩu...

Mã:

tshark -i eth0 -Y 'http.request.method == POST and tcp contains "password"' | grep password

8.**Trích xuất dữ liệu (files)**  
Với tùy chọn --export-objects ta có thể dể dàng trích xuất dữ liệu dạng file vào một thư mục được chỉ định.

Mã:

tshark -nr vidu.pcap --export-objects http,thumucluu

9.**Các tùy chọn khác của Tshark hỗ trợ phân tích gói tin**  
Trong quá trình phân tích ta có thể sử dụng tùy chọn -V để hiển thị chi tiết thông tin từng gói tin như số frame, giao thức...

Mã:

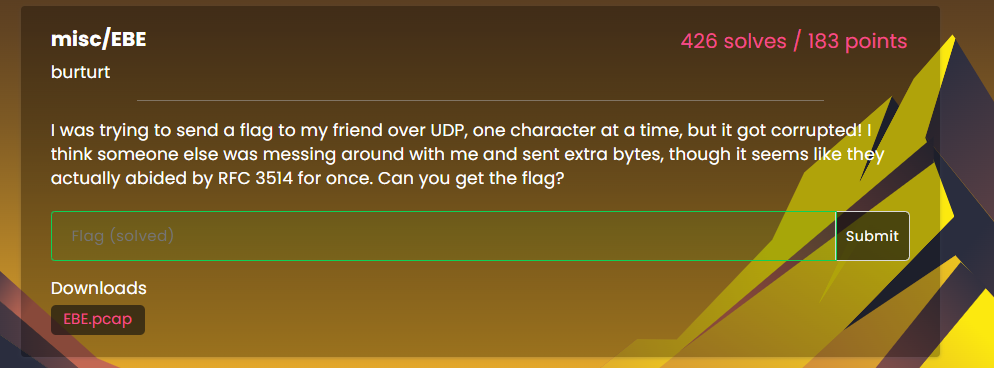
tshark -i eth0 -V

10.**Tùy chọn -O tương tự -V nhưng chỉ hiện thị chi tiết gói tin sử dụng giao thức mà chúng ta chỉ định.**

Mã:

tshark -i eth0 -O ftp

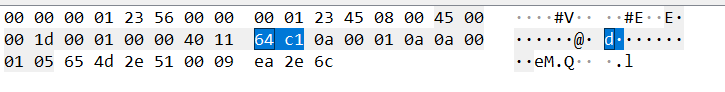
#EX : Một challenge trong giải lactf 2023 vừa qua

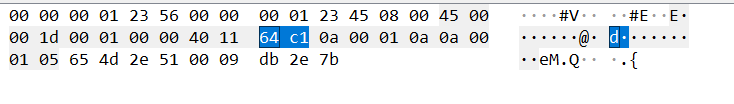
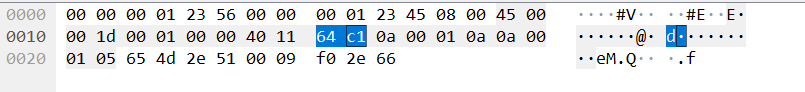
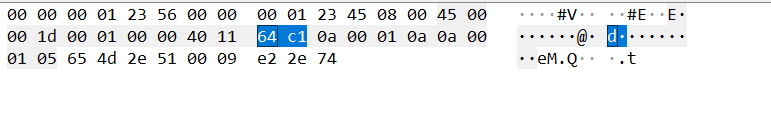
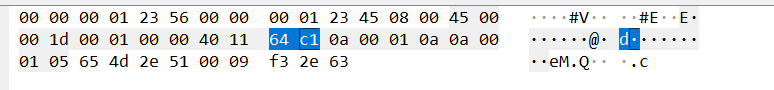
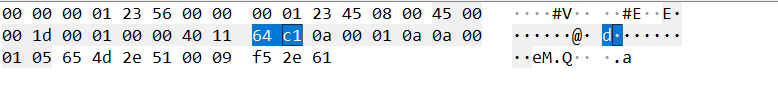


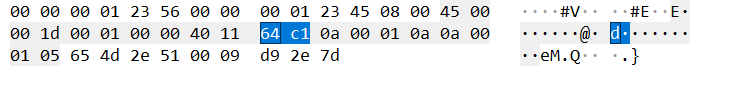
\* Đầu tiên ta dùng wireshark để phân tích dữ liệu của nó :

+ Dựa vào form flag là lactf{} e phát hiện điều bất thường :

- E dùng strings để tìm phát hiện :

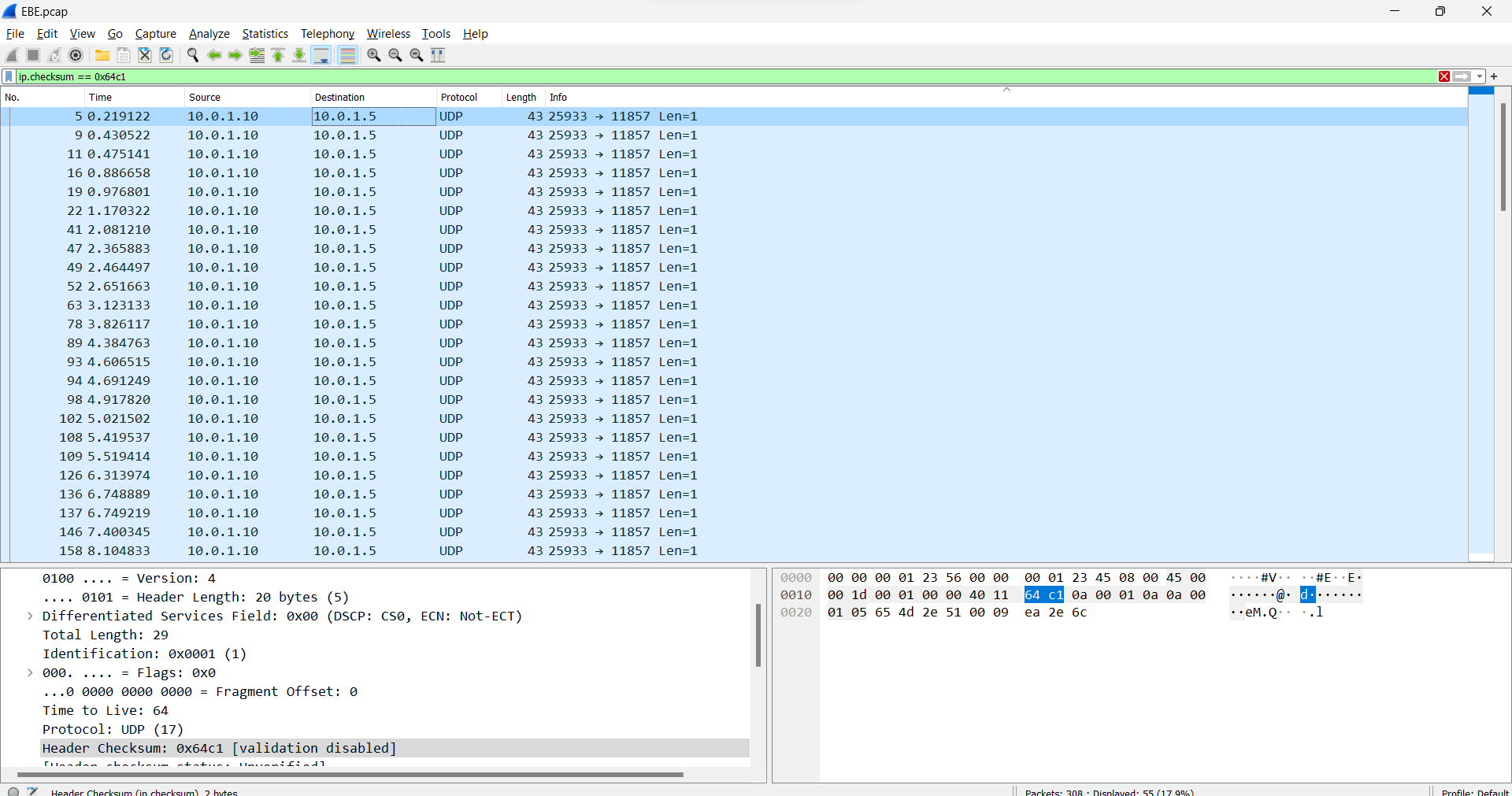






=> e chắc chắn được rằng 0x64c1 có liên quan đến flag nên quyết định lấy hết các dữ liệu liên quan đến nó

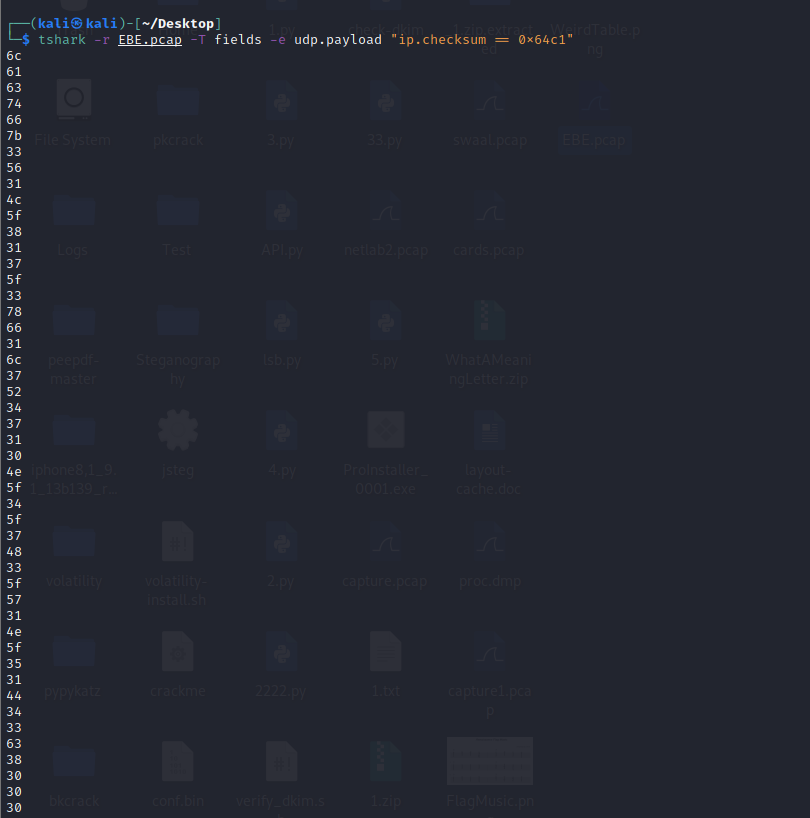
\_Dùng ****Apply as Filter**** để tạo một filter dựa theo 0x64c1 :



=> Lướt hết các packet e thấy được mỗi packet chứa các ký tự của flag nằm ở phần udp.payload

=> và e dùng tshark để lấy hết nó ra bằng lệnh :

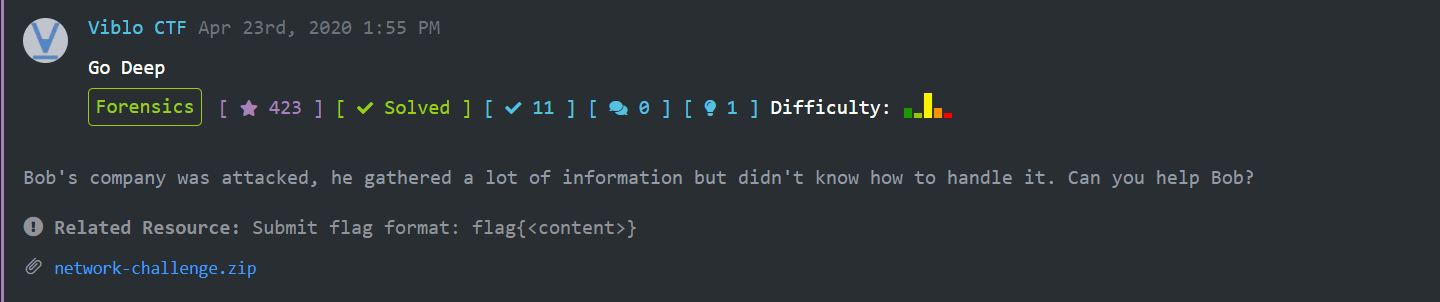
**$tshark -r EBE.pcap -T fields -e udp.payload "ip.checksum == 0x64c1"**



=> cuối cùng decode hex bằng cyberchef.org và nhận được flag :



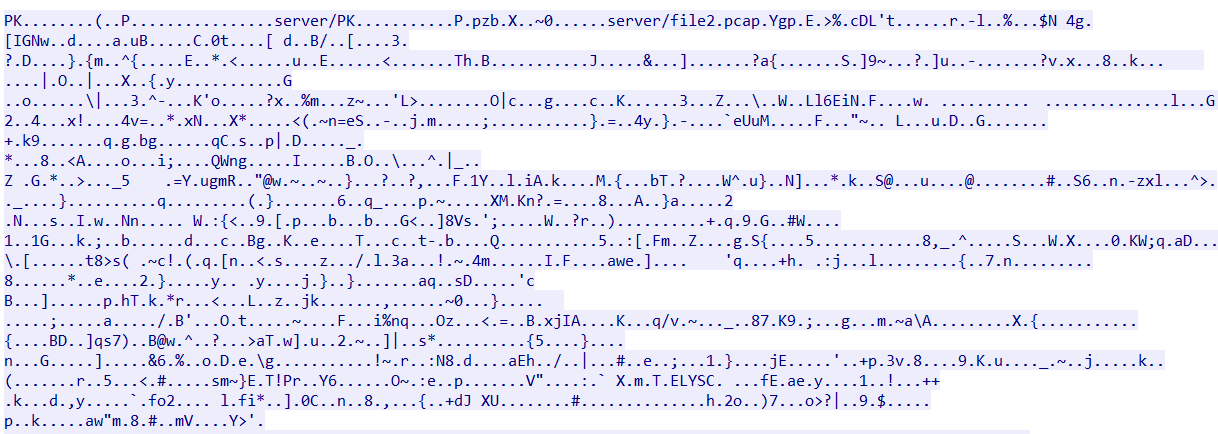
#EX2: 1 challenge của viblo CTF :



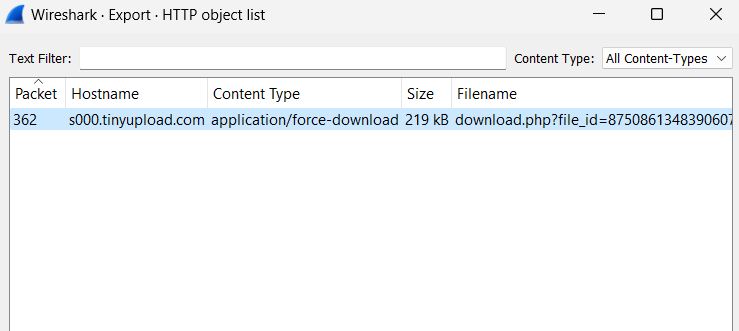
\_Dùng wireshark để phân tích :

\_Nhìn sơ lượt 1 lúc thì e thấy chủ yếu toàn là giao thức tcp nên đã Follow TCP Stream xem sao

=> phát hiện :



=> extract file zip này ra bằng export HTTP :

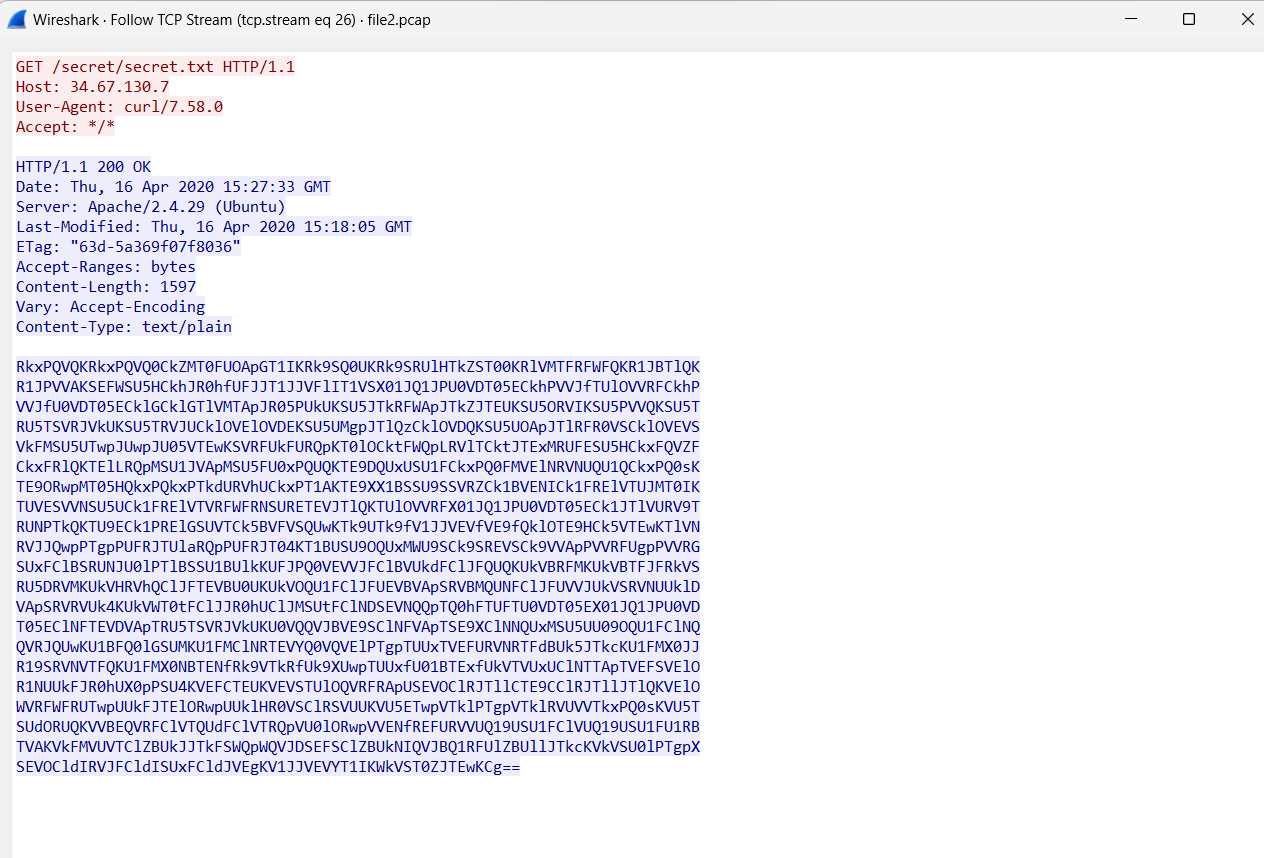


=> save file và lưu với đuôi .zip

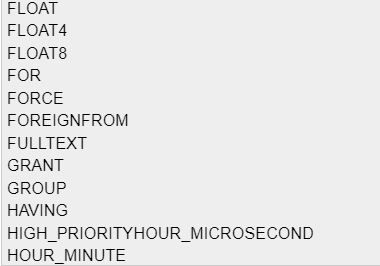
=> thì nhận được 1 file pcap nữa và bắt đầu phân tích nó

=> cũng khá giống với file pcap đầu tiên protocol TCP chiếm phần lớn nên e tiếp tục follow tcp stream

=> Và hiện hiện ở luồng 26 có mã base64 đáng nghi ngờ và quyết định decode nó ra :

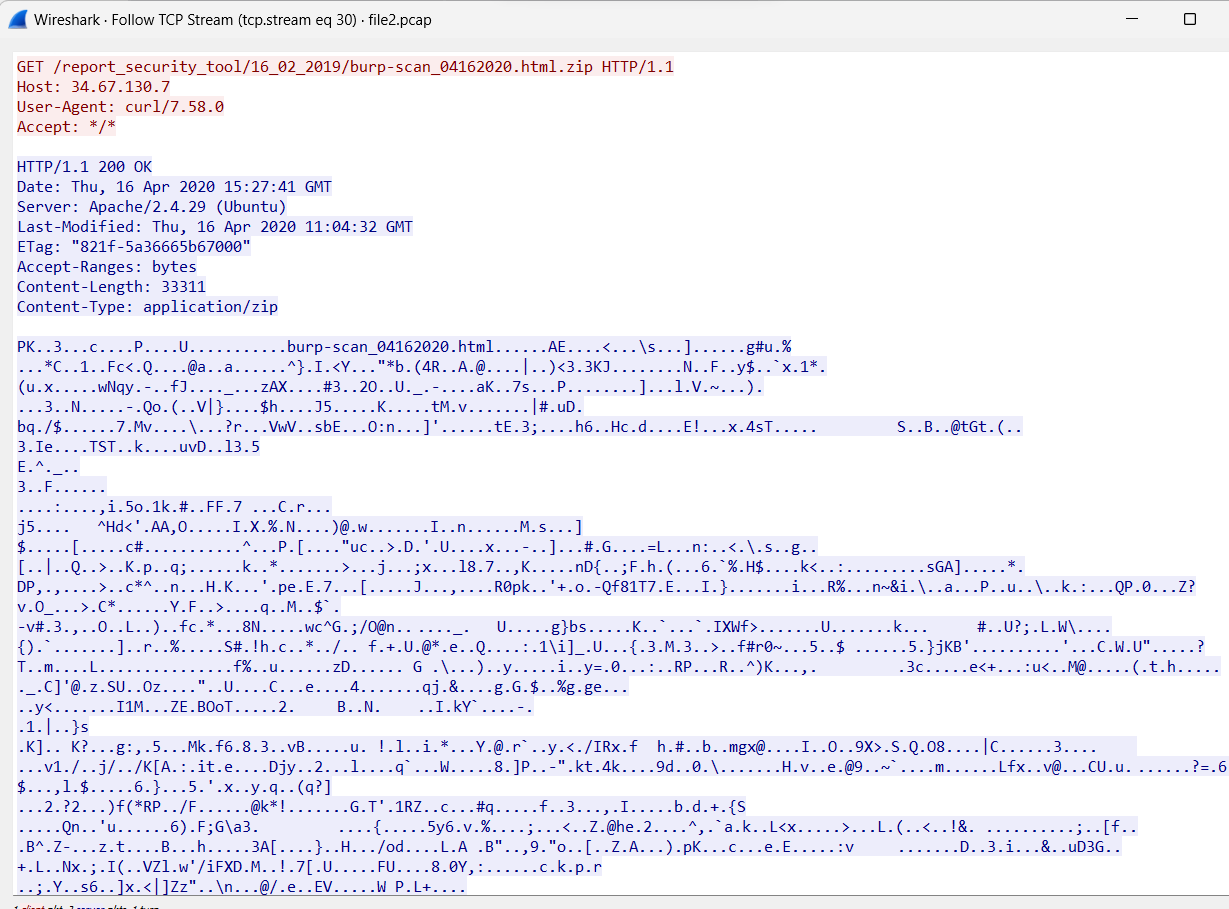


=> sau khi decode e nhận được :

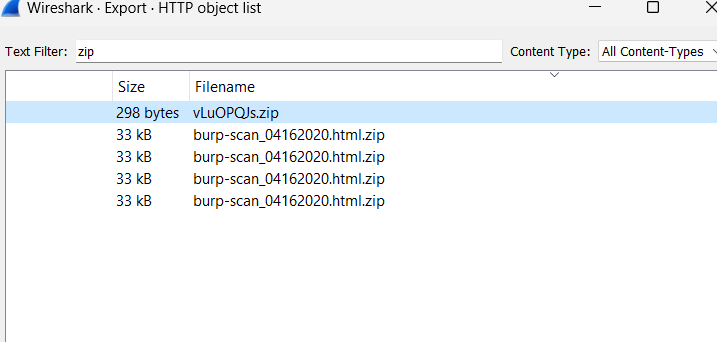


=> e nhận thấy nó giống như danh sách mật khẩu để crack file zip nào đó :vv nên đã quyết định copy nó vào file 1.txt và để đó

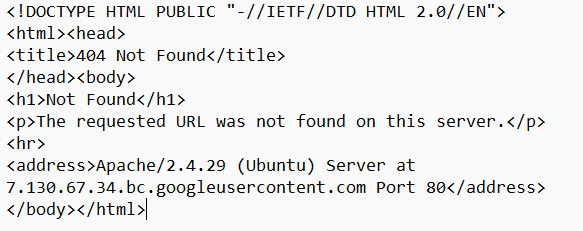
\_Và ở luồng 30 e phát hiện :



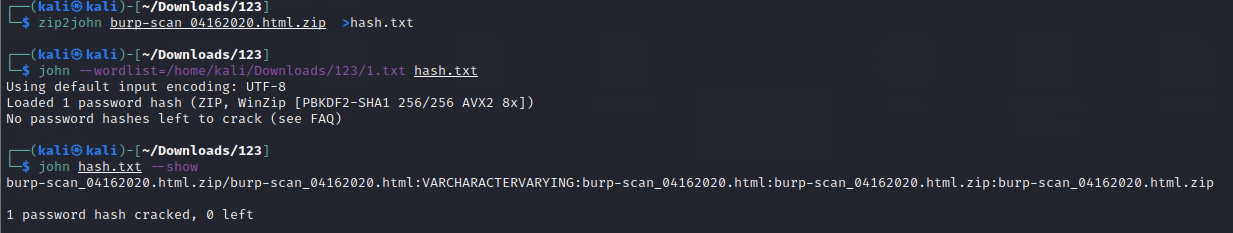
\_Y như lúc đầu , e Export HTTP và save file zip lại :



=> nhận thấy file đầu tiên thực chất ko phải là file zip mà chỉ là mã trang web lỗi "404 Not Found"



=> e extract file burp-scan\_04162020.html.zip và xem nó thì bắt nhập pass và e liền nhớ đến danh sách lúc này để có thể crack file zip này bằng johntheripper



=> Do e crack nó 1 lần rồi nên lần này chỉ cần show thôi :vv và pass là **VARCHARACTERVARYING**

=> Nhập pass vào và mở file lên, e đọc sơ 1 lượt thì cũng không có gì đặc biệt lắm chỉ là giới thiệu về Burp Suite là trình quét bảo mật về web phổ biến được sử dụng rộng rãi nhất hiện nay.

=> sau đó e hỏi a long thì dùng với strings | grep để lấy flag :



1. Thự Viện Scapy :

\_ Scapy là một thư viện Python mạnh mẽ được sử dụng để tạo, gửi và nhận các gói tin mạng. Nó được sử dụng để phân tích gói tin mạng, kiểm tra và thử nghiệm tính bảo mật hệ thống mạng, tạo các công cụ mạng tùy chỉnh và phát triển các ứng dụng mạng.

\_ Scapy có thể tạo, gửi và nhận các gói tin TCP, UDP, ICMP, DNS, DHCP, ARP và nhiều giao thức khác. Nó cũng cho phép bạn thực hiện các hoạt động như quét mạng, giả mạo địa chỉ MAC, thu thập thông tin từ các gói tin mạng và nhiều hơn nữa.

\* Một số tính năng của Scapy bao gồm:

+ Tạo và tùy chỉnh các gói tin mạng

+ Tương tác với các giao thức mạng khác nhau

+ Phân tích gói tin mạng

+ Thực hiện các hoạt động kiểm tra bảo mật mạng

+ Phát hiện và ngăn chặn các tấn công mạng

+ Ghi lại và phân tích các hoạt động mạng

+ Scapy cung cấp một giao diện dòng lệnh và một API Python cho phép bạn tích hợp nó vào các ứng dụng mạng của mình. \_Scapy là một công cụ rất mạnh mẽ và phổ biến trong cộng đồng an ninh mạng và phát triển ứng dụng mạng.

1. Logs Web :

\_Logs web là các tập tin ghi lại các thông tin về hoạt động của một trang web, bao gồm các yêu cầu HTTP, địa chỉ IP của người dùng, thông tin về trình duyệt và các trang được truy cập. Để đọc hiểu thông tin ghi trong log web, bạn cần hiểu các thuật ngữ và khái niệm sau:

1.Yêu cầu HTTP: là các yêu cầu được gửi đến máy chủ web bởi trình duyệt của người dùng, bao gồm các phương thức như GET, POST, PUT, DELETE, vv.

2.Địa chỉ IP: là địa chỉ số dạng nhị phân hoặc thập phân được sử dụng để xác định máy tính của người dùng truy cập vào trang web.

3.User agent: là thông tin về trình duyệt hoặc phần mềm mà người dùng sử dụng để truy cập vào trang web.

4.Referrer: là trang web hoặc tài nguyên trên trang web mà người dùng đã truy cập trước đó và dẫn đến trang web hiện tại.

5.Status code: là mã trả về từ máy chủ web để chỉ ra kết quả của yêu cầu HTTP, ví dụ như 200 OK (yêu cầu thành công) hoặc 404 Not Found (không tìm thấy trang).

6.Thời gian truy cập: là thời điểm mà người dùng đã truy cập vào trang web.

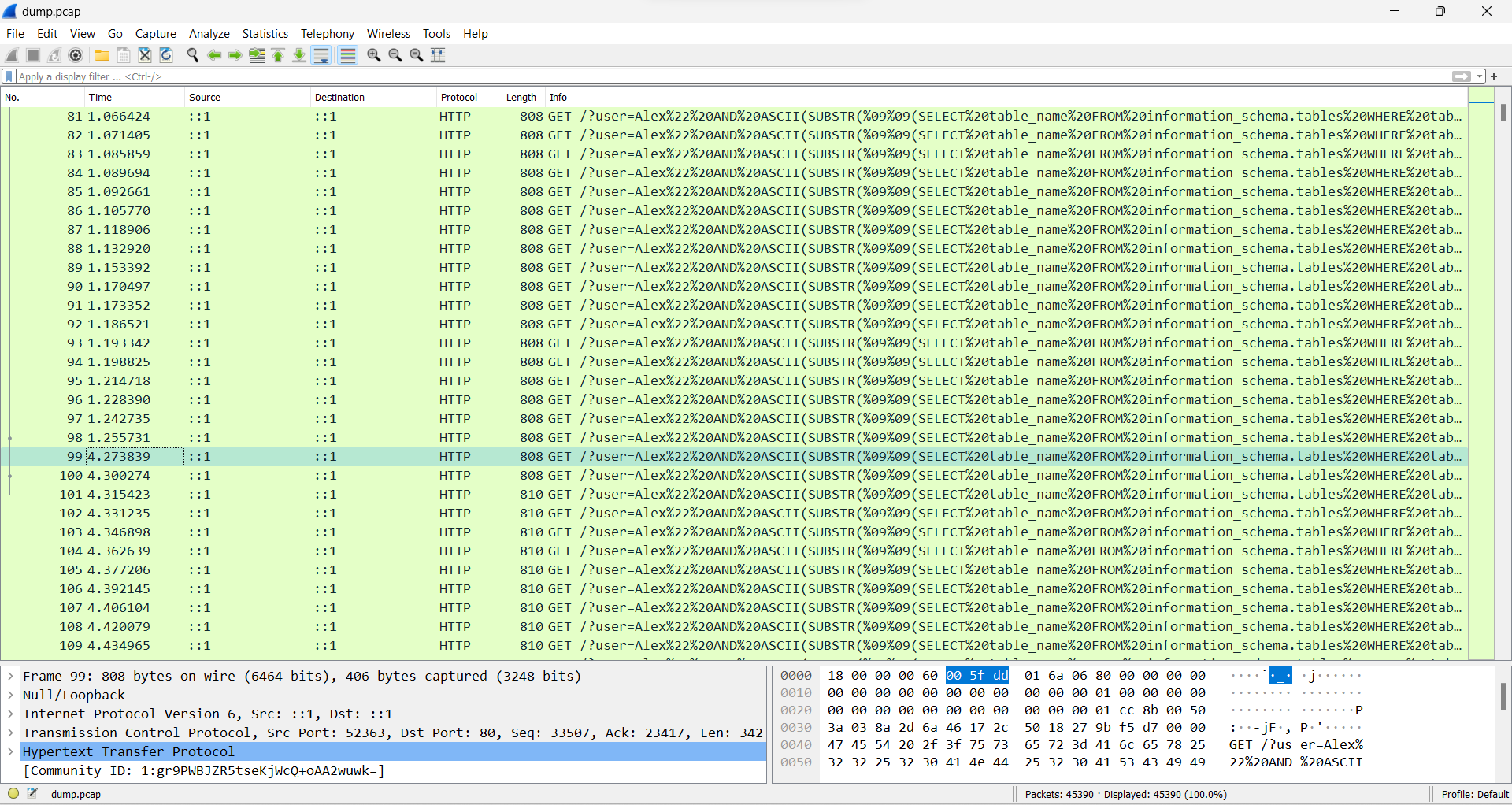
7.URI: là địa chỉ của trang web hoặc tài nguyên được truy cập trên trang web.

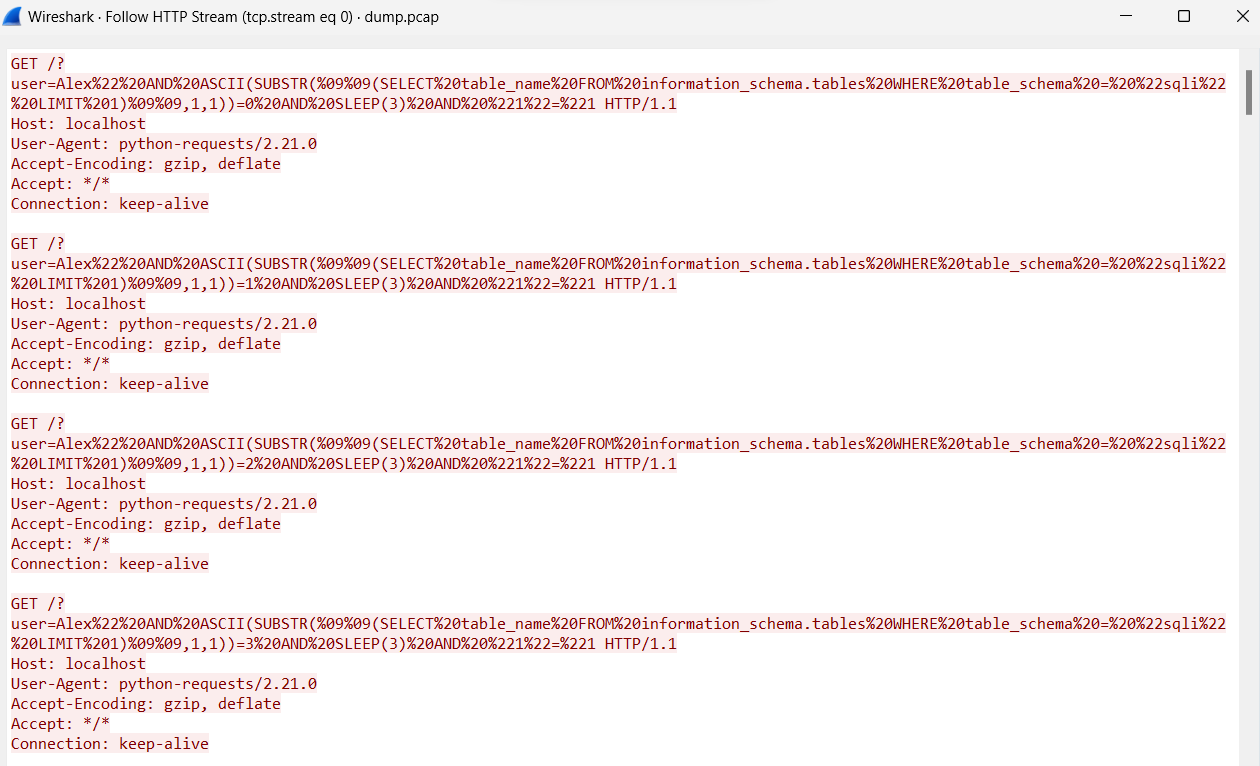
\*Để đọc hiểu thông tin ghi trong log web, bạn có thể sử dụng các công cụ đọc log web như Apache Logs Viewer, GoAccess hoặc các phần mềm đọc log web khác. Các công cụ này cho phép bạn lọc và phân tích các thông tin trong log web để hiểu hành vi của người dùng trên trang web của bạn, tìm kiếm các vấn đề bảo mật hoặc tìm kiếm các cơ hội để cải thiện trang web của bạn.

\_Phân tích logs web :

+ Đây là 1 file pcap e đã phân tích có liên quan đến logs web

+ Dùng wireshark để đọc dữ liệu thì e phát hiện các yêu cầu HTTP có chung 1 mẫu :





=> Nhận thấy phần user có mã URL, nên e đã decode nó ra và nhận được :

“Alex" AND ASCII(SUBSTR((SELECT table\_name FROM information\_schema.tables WHERE table\_schema = "sqli" LIMIT 1),1,1))=0 AND SLEEP(3) AND "1"="1 HTTP/1.1”

=> Điều này có nghĩa là : kẻ tấn công đang thưc hiện thao tác tiêm SQL dựa trên thời gian.ASCII() trả về giá trị ascii của một chữ cái. SUBSTR() trả về một chữ cái cho một vị trí nhất định.Khi một kết quả được đánh giá là đúng (giá trị ascii là chính xác) thì phản hồi sẽ mất 3 giây để đến nơi. Nói cách khác, chúng ta phải nhận mọi yêu cầu chính xác với độ trễ 3 giây trước các yêu cầu tiếp theo.

Ví dụ: yêu cầu số 98 là đúng vì yêu cầu tiếp theo 99 có độ trễ hơn 3 giây:

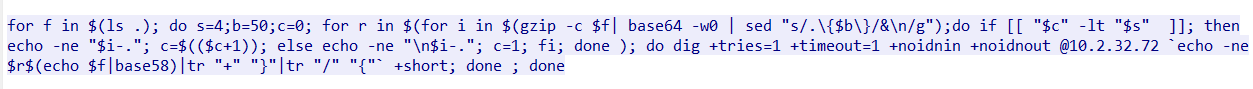


1. WU netlab2 and netlab5 :
2. Netlab2 :

\_ Dùng wireshark để xem file thì e thấy :

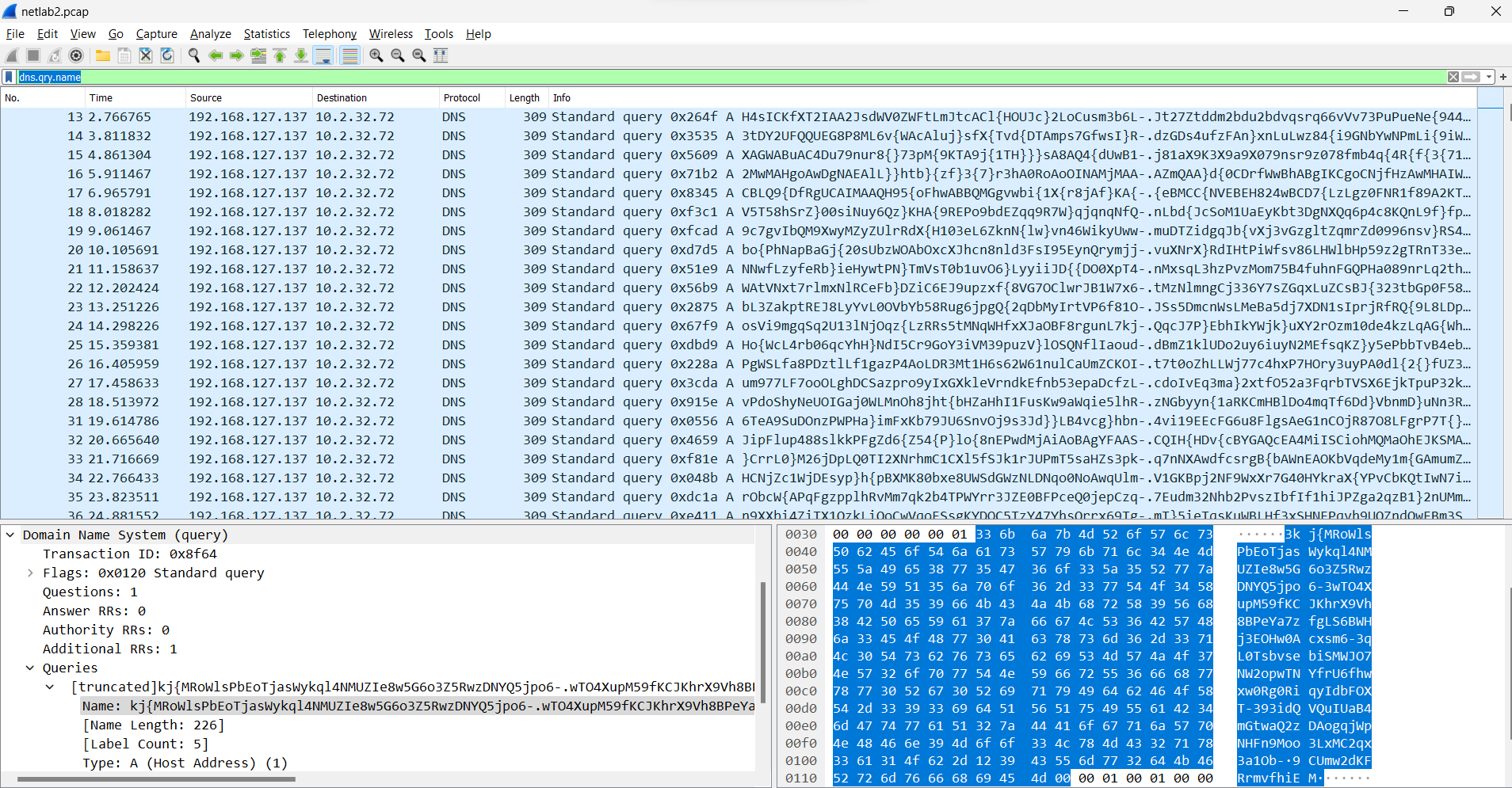


=> Follow nó và e thấy một file bash là :

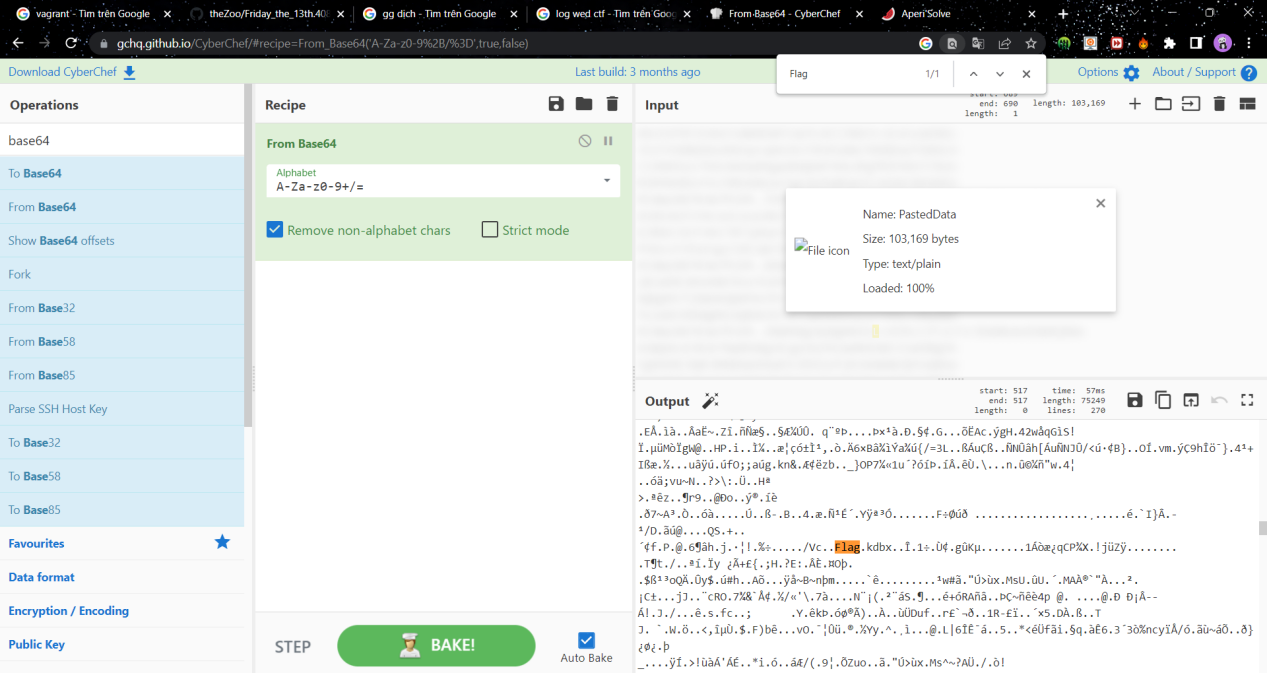


=> Đọc hiểu nó thì e hiểu được 1 chút là đầu ra của lệnh trước sau đó được sử dụng để xây dựng các truy vấn DNS bằng lệnh dig. Tập lệnh gửi truy vấn DNS đến địa chỉ IP 10.2.32.72 bằng cách sử dụng chuỗi truy vấn được tạo bằng cách nối đoạn dữ liệu được mã hóa base64 và tên tệp $f, được mã hóa ở định dạng base58. Dấu cộng (+) trong chuỗi truy vấn được thay thế bằng dấu ngoặc nhọn đóng (}) và dấu gạch chéo lên (/) được thay thế bằng dấu ngoặc nhọn mở ({). Tùy chọn plusshort được sử dụng để chỉ in các địa chỉ IP do máy chủ DNS trả về.Dấu (.) sẽ thay thế được thay thế bằng ( ) .

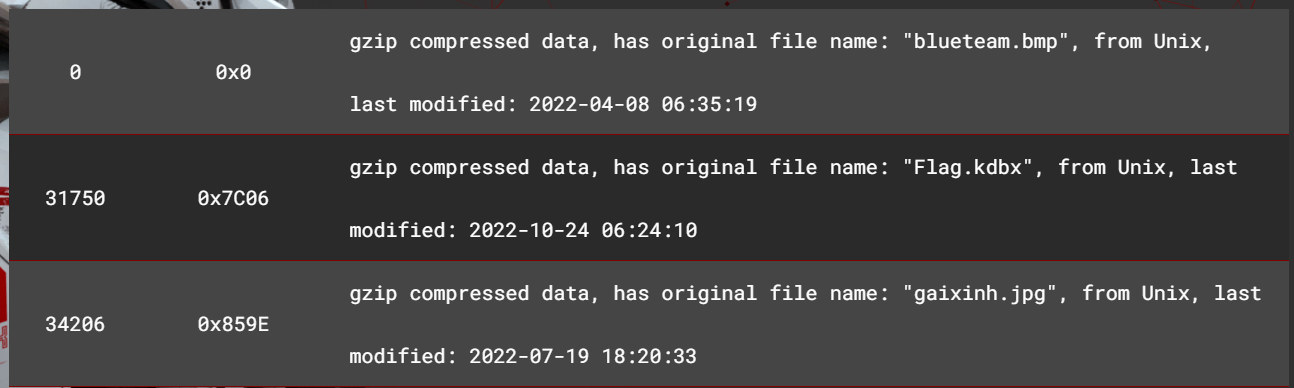
\_Cộng thêm việc quan sát giao thức DNS(nằm ở dns.qry.name) thì có khác nhiều mã rất giống với kiểu base58 và base64



\_Lúc đầu e dùng tshark để extract nó ra và chuyển đổi dấu (.) thành ( ) ; chuyển ({) thành (/) và chuyển (}) thành (+) ; decode base64 thì phát hiện có 1 file Flag.kdbx:



\_ E quyết định binwalk nó ra :



\_Nhưng khi mở file của nó thì bị lỗi ko mở được nên e dựa vào scapy để lấy file Flag.kdbx (có thể e bị sai bước nào đó, e nghĩ là về gzip)

\_Dựa vào thư viện scapy để giải mã nó ra file 1.zip :

from scapy.all import \*

import base64

import base58

pcap = rdpcap(r"C:\Users\ASUS\Downloads\netlab2.pcap")

def cleanse(strings):

    if '.' in strings:

        return strings.replace('.', '')

    return strings

out = ""

for pkt in pcap:

    if pkt.haslayer(DNSQR):

        strings = str(pkt[DNSQR].qname)[2:-2]

        if not 'local' in strings:

            tmp = strings.split("-")

            specify = base58.b58decode(cleanse(tmp[-1]))

            if specify == b'Flag.kdbx\n':

                tmp.pop(-1)

                for i in tmp:

                    out += cleanse(i)

out = out.replace('{', '/')

out = out.replace('}', '+')

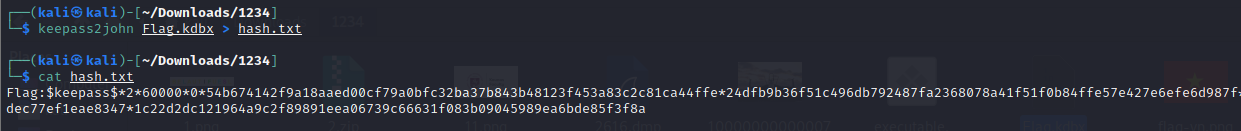
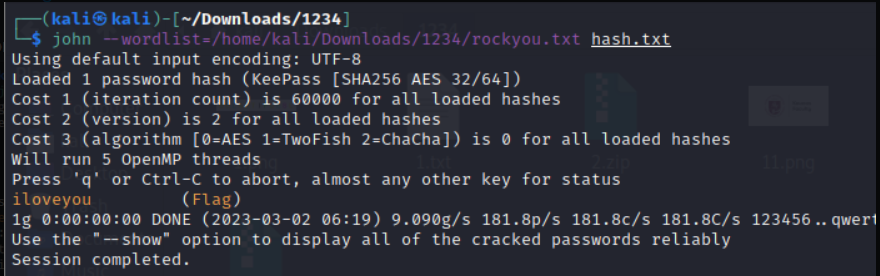
zoo = base64.b64decode(out)

with open ("1.zip", "wb") as f:

    f.write(bytearray(zoo))

=> e nhận được file 1.zip và extract nó ra nhận được file Flag.kdbx nhưng lần này đã mở được file

Nhưng nó bắt nhập pass nên e dùng johntheripper để crack file :

=> pass : iloveyou

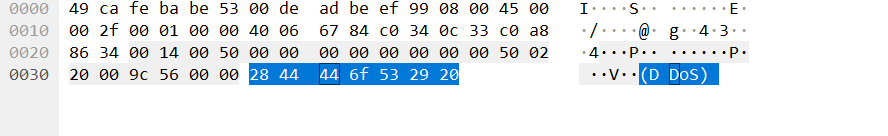
=> Mở file và copy password và e có được cờ :

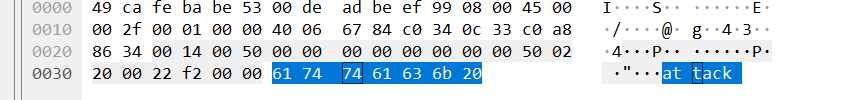
#Flag{NetLab2\_DNS\_3xf1ltr4t10n\_15\_5t3al7hy}

1. Netlab5 :

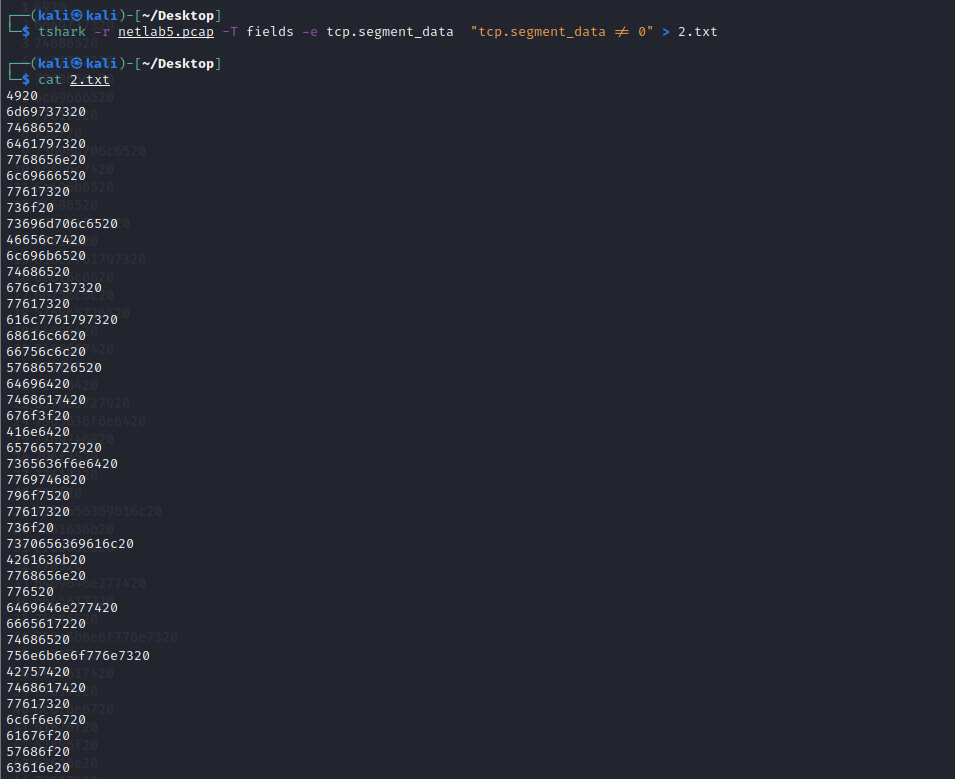
\_Đầu tiên dùng wireshark để xem dữ liệu của nó

\_Oh theo mô tả của đề bài thì nó có liên quan đến DDOS Attack nên e sử dụng strings để xem nó có đề cập ko và e đã thấy được thứ mình cần

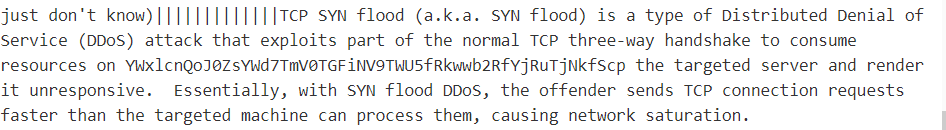




=> Nó nằm ở tcp.segment\_data nên đã sử dụng tshark để lấy nó ra :



=> decode hex thì e nhận được 1 mã base64



=> cuối cùng ta chỉ cần giải mã nó ra và nhận được cờ :

