HỌC VIỆN KỸ THUẬT QUÂN SỰ

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO HỌC PHẦN:

THỰC TẬP CÔNG NGHỆ THÔNG TIN

Đề tài: Xây dựng công cụ chặn bắt và phân tích gói tin

Giáo viên hướng dẫn: Gv. Nguyễn Văn Quân

Thành viên nhóm:

Hoàng Văn Việt

Phan Thanh Tùng

Lỗ Trung Hiếu

Nguyễn Văn Dũng

HÀ NỘI, 2021

MỤC LỤC

DANH MỤC CHỮ VIẾT TẮT

Chữ viết tắt	Mô tả chữ viết tắt
OS	Operating System
ARP	Address Resolution Protocol
RARP	Reverse Address Resolution Protocol
IP	Internet Protocol
ICMP	Internet Control Message Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
FTP	File Transfer Protocol
DoD	Department of Defense
SMTP	Simple Mail Transfer Protocol
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
SNMP	Simple Network Management Protocol
HTTP	HyperText Transfer Protocol
IGMP	Internet Group Management Protocol
OSI	Open Systems Interconnection Model
MAC	Media Access Control
SOF	Start Frame Delimiter
DA	Destination Address
PM	Promiscuous Mode
SA	Source Address
FCS	Frame Check Sequence
BPS	Byte Per Second
PPS	Packet Per Second

DANH MỤC CÁC HÌNH

DANH MỤC CÁC BẢNG

DANH MỤC CÁC BIỂU ĐỒ

1. Mục đích xây dựng đề tài

Điều tra số ở tầng mạng là quá trình giám sát các gói tin và phân tích hoạt động trao đổi gói tin phục vụ cho việc phát hiện mã độc và tấn công trên không gian mạng. Bằng việc phát hiện ra vấn đề, thu thập và phân tích dữ liệu, sẽ đưa ra những giải pháp tốt nhất để xử lý và khắc phục sự cố.

Để phục vụ cho việc điều tra số trên tầng mạng, các chuyên gia xử lý sự cố sẽ sử dụng phương pháp theo dõi các gói tin đi qua đường truyền mạng, thu thập các gói tin nghi ngờ và sẽ phân tích gói tin này.

Ngoài việc phát hiện và xử lý các mối đe dọa từ mã độc, hoặc các cuộc tấn công thông qua đường mạng. Các vấn đề về hệ thống hạ tầng mạng, hiệu suất tổng thể và lưu lượng băng thông trên đường truyền mạng cũng phát sinh theo từng ngày, từng giờ.

Hằng năm, có rất nhiều tổ chức, doanh nghiệp bị tấn công trên không gian mạng, và vấn đề đặt ra là phải tập trung đưa ra những giải pháp về mặt công nghệ để phát hiện sớm, phòng thủ, để giảm thiểu các cuộc tấn công trên không gian mạng và việc mất dữ liệu người dùng trên máy tính thuộc sở hữu của cá nhân, tổ chức.

Xuất phát từ bài toán trên, các công cụ giúp chuyên gia theo dõi, phân tích các gói tin trên đường truyền mạng ra đời, với mục tiêu hỗ trợ chuyên gia trong việc điều tra số ở tầng mạng, đảm bảo một môi trường Internet “sạch” cho cá nhân, tổ chức, doanh nghiệp.

Với mục tiêu tìm hiểu và phân tích các gói tin trên đường truyền mạng, tiếp cận với những công nghệ mới hiện nay, chúng em quyết định lựa chọn đề tài “ Xây dựng công cụ chặn bắt và phân tích gói tin” dựa trên các tính nặng đang có của phần mềm Wireshark. Chúng em xây dựng phần mềm CyberShark được viết bằng ngôn ngữ .NET là phần mềm mô phỏng, bản rút gọn của phần mềm Wireshark với mục đích phục vụ cho việc học tập vào nghiên cứu.

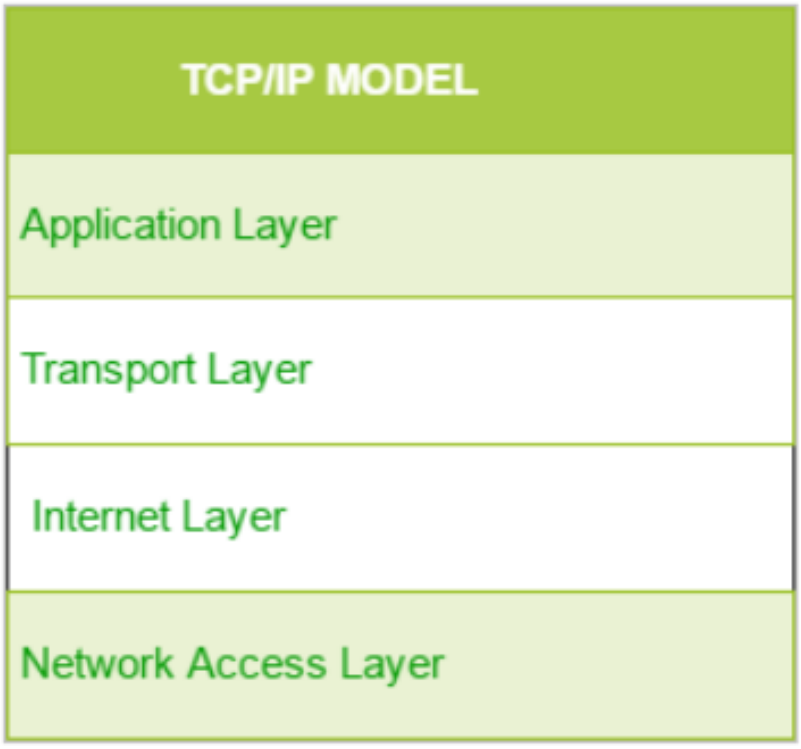
2. Nghiên cứu công nghệ cần sử dụng để thực hiện đề tài

2.1. Mô hình TCP/IP

2.1.1. Giới thiệu về giao thức

Trước tiên ta sẽ tìm hiểu về giao thức TCP/IP. Giao thức TCP/IP(Transmission Control Protocol/Internet Protocol) là một bộ gồm nhiều giao thức truyền thông khác nhau. Đây là bộ giao thức phổ biến nhất mà Internet và hầu hết các mạng máy tính thương mại hiện đang sử dụng. Bộ giao thức này được đặt tên theo hai giao thức chính của nó là TCP và IP. Bộ giao thức TCP/IP có thể được coi là một tập hợp các tầng. Mỗi tầng trong đó giải quyết một tập hợp các vấn đề có liên quan đến việc truyền, nhận dữ liệu.

Để cho phép các máy tính trao đổi dữ liệu với nhau TCP/IP] sử dụng mô hình truyền thông 4 tầng hay còn gọi là mô hình mạng DoD (Department of Defense):



Hình 2.1. Các tầng trong mô hình TCP/IP

- *Tầng ứng dụng (Application Layer).*

Tầng này bao gồm tất cả các chương trình ứng dụng sử dụng các dịch vụ sẵn có thông qua một chồng giao thức TCP/IP. Các chương trình ứng dụng tương tác với một trong các giao thức của tầng giao vận để truyền hoặc nhận dữ liệu.

- *Tầng giao vận (Transport Layer).*

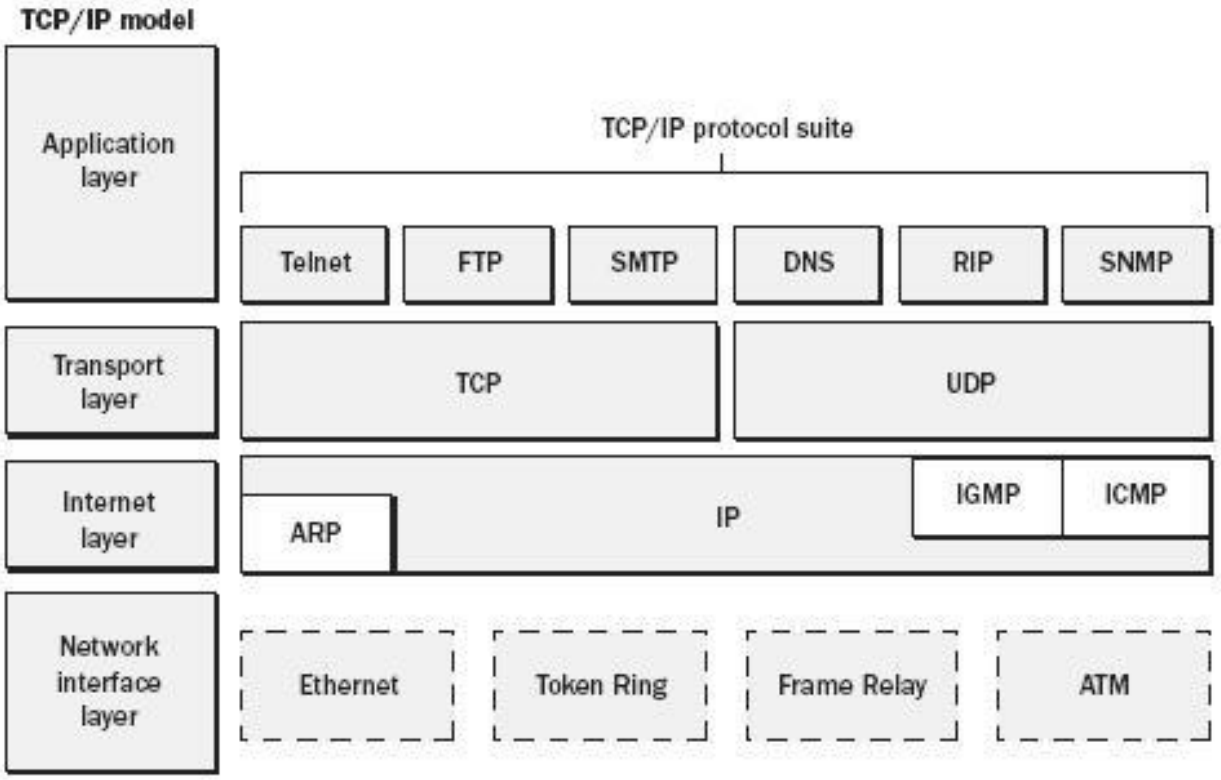
Nhiệm vụ trước tiên của tầng giao vận là cung cấp sự giao tiếp thông tin giữa các chương trình ứng dụng. Mỗi sự giao tiếp được gọi là end-to-end. Tầng giao vận cũng có thể điều chỉnh lưu lượng luồng thông tin. Tầng giao vận chia dòng dữ liệu ra thành những đơn vị dữ liệu nhỏ hơn (thường được gọi là các Packets) và tập hợp lại khi nhận đảm bảo dữ liệu toàn vẹn(không mất, không lỗi, không lặp, đúng thứ tự).

- *Tầng Internet (Internet Layer).*

Tầng mạng xử lý vấn đề dẫn các gói tin đi qua các mạng để đến đúng đích. Nó chấp nhận một yêu cầu để gửi một gói từ từ tầng giao vận cùng với một định danh của máy đích mà gói tin sẽ được gửi tới

- *Tầng truy cập (Access Layer).*

Là tầng thấp nhất của bộ giao thức TCP/IP, chịu trách nhiệm về việc chấp nhận các datagram của tầng trên (ví dụ IP datagram) và việc truyền phát chúng trên một mạng xác định. Theo quan điểm hiện nay mô hình TCP/IP không còn bao gồm các đặc tả vật lý, nói cách khác tầng truy cập cũng không còn bao gồm vấn đề về phần cứng hay việc truyền tín hiệu vật lý nữa.



Hình 2.2. Mô hình TCP/IP và các giao thức trong từng tầng.

Mỗi giao thức của TCP/IP đều thuộc một trong các tầng ở trên. Mỗi tầng có các giao thức để phục vụ tầng trên nó, và các giao thức để sử dụng dịch vụ của tầng dưới nó.

- Các giao thức chính hoạt động ở tầng Application:
    - *FTP* (File transfer Protocol): Giao thức truyền file, cho phép người dùng gửi hoặc nhận các file dữ liệu từ máy này tới máy khác thông qua mạng.
    - *Telnet*: Giao thức cho phép mô phỏng thiết bị đầu cuối giúp người dùng đăng nhập vào máy chủ từ một máy khác trên mạng.
    - *SMTP* (Simple Mail Transfer Protocol): Một giao thức để gửi và nhận thư.
    - *DHCP* (Dynamic Host Configuration Protocol): Giao thức cấu hình địa chỉ IP động cho các host.

- *DNS* (Domain Name System): Dịch vụ tên miền cho phép truy cập một máy tính bằng tên tên miền của nó thay vì địa chỉ IP khó nhớ.
- *SNMP* (Simple Network Management Protocol): Giao thức quản lý mạng đơn giản.
- *HTTP* (HyperText Transfer Protocol): là một trong năm giao thức chuẩn về mạng Internet, được dùng để liên hệ thông tin giữa Máy cung cấp dịch vụ (Web server) và Máy sử dụng dịch vụ (Web client) là giao thức Client/Server dùng cho WWW

- Các giao thức chính hoạt động ở tầng Transport:
  - *UDP* (User Datagram Protocol): Giao thức truyền không tin cậy nhưng có ưu điểm của nó là nhanh và tiết kiệm.
  - *TCP* (Transmission Control Protocol): Cung cấp một phương thức truyền tin cậy
- Các giao thức chính hoạt động ở tầng Internet:
  - *IP* (Internet Protocol): Giao thức Internet, cung cấp các thông tin để làm sao các gói tin có thể đến được đích.
  - *ARP* (Address Resolution Protocol): Giao thức này có chức năng biên dịch địa chỉ IP của máy đích thành địa chỉ MAC.
  - *ICMP* (Internet Control Message Protocol): Một giao thức thông báo lỗi xảy ra trên đường truyền.
  - *IGMP* (Internet Group Management Protocol): Giao thức cung cấp chức năng điều khiển truyền dẫn đa hướng (multicast).
- Các công nghệ thường gặp ở tầng Network access: Ethernet, Token Ring, Token Bus, Fiber.

2.1.2. Quá trình truyền nhận dữ liệu trong TCP/IP

Truyền thông trên mạng là một thủ tục phức tạp bao gồm một loạt các tiến trình. TCP/IP truyền dữ liệu qua mạng bằng cách chia nhỏ dữ liệu thành các khối gọi là các gói tin (packet). Tên của các gói này sẽ thay đổi theo từng giao thức. Truyền một lượng dữ liệu lớn tương đương với một gói lớn qua mạng sẽ mất khá nhiều thời gian, làm giảm đáng kể hiệu suất mạng

hoặc gây tắc nghẽn toàn mạng do vậy cần chia dữ liệu thành các gói nhỏ để chúng có thể dễ dàng di chuyển trên mạng điều này tránh được tắc nghẽn mạng và cho phép các máy khác cũng có thể truyền dữ liệu, nếu một gói bị hỏng thì chỉ cần truyền lại gói đó chứ không cần truyền lại toàn bộ dữ liệu.

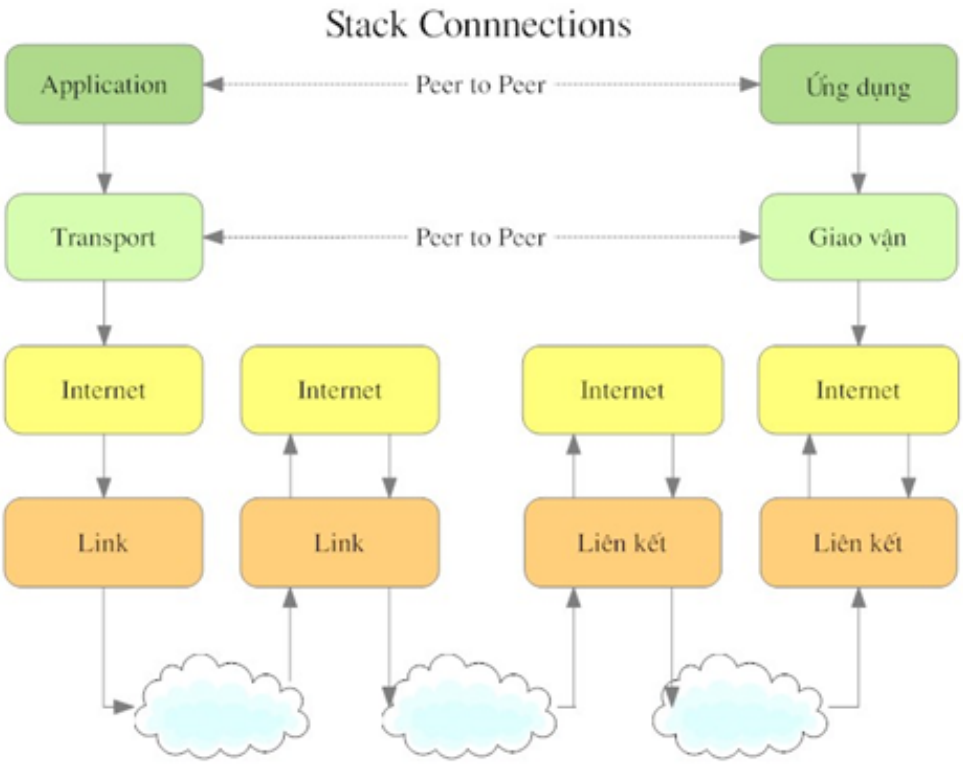
Khi gói được chuyển từ trên xuống qua các tầng thì giao thức tại mỗi tầng thêm vào các gói thông tin riêng, thông tin riêng này được gọi là các header.

Trong giao thức TCP đơn vị dữ liệu được gọi là các phân đoạn (segment), nó chứa header của giao thức TCP và dữ liệu ứng dụng.

Trong các giao thức không tin cậy như ICMP, UDP, IGMP, ARP đơn vị truyền được gọi là các thông báo (message), nó gồm phần header và dữ liệu ứng dụng hoặc dữ liệu giao thức.

Trong giao thức IP đơn vị truyền được gọi là gói dữ liệu (ip datagram), nó gồm phần header và dữ liệu từ tầng Transport.

Ở tầng Network access đơn vị dữ liệu được gọi là khung (frame), nó gồm header và dữ liệu từ tầng Internet.



Hình 2.3. Đường đi của các gói tin trong môi trường mạng

2.1.3. Một số giao thức thường gặp và khuôn dạng dữ liệu

1) Ethernet.

Là giao thức nằm trong tầng liên kết hay một chuẩn công nghệ dành cho mạng cục bộ (LAN) được quy định trong IEEE 802.3. Nó là một giao thức nằm trong tầng liên kết của bộ giao thức TCP/IP hay tương ứng là tầng liên kết dữ liệu trong mô hình OSI. Hiện nay nó đang được sử dụng rất rộng rãi so với các giao thức khác như FDDI, Token Ring…Ethernet được dùng để gửi những khối dữ liệu giữa điểm nguồn và điểm đích được xác định dựa vào địa chỉ MAC (Media Access Control).

Cấu trúc của một đơn vị dữ liệu trong giao thức Ethernet (gọi là Ethernet frame) như sau: (đơn vị tính theo byte).

7	1	6	6	2	46-1500	4
PRE	SOF	DA	SA	Length/Type	Data Payload	FCS

Bảng 2.1. Cấu trúc của Ethernet frame.

- Preamble (PRE): Phần mở đầu gồm 7 byte và không được tính vào kích thước của Ethernet. Tất cả các byte trong phần mở đầu này đều có giá trị 10101010 và nó được dùng để đồng bộ đồng hồ giữa nơi nhận và gửi frame.
- SOF (Start frame delimiter): gồm 1 byte và không được tính vào kích thước của Ethernet. Byte này có giá trị 101010111 và được sử dụng để đánh dấu bắt đầu của một frame. Đối với những hệ thống Ethernet hiện nay hoạt động ở tốc độ 100Mbps hoặc 1000Mbps không còn cần tới PRE và SOF.
- DA (Destination Address): có độ dài 6 byte là địa chỉ nơi MAC của Ethernet card nơi đến. Ở chế độ hoạt động bình thường Ethernet chỉ tiếp nhận những frame có địa chỉ nơi đến trùng với địa chỉ (duy nhất) của nó hoặc địa chỉ nơi đến thể hiện một thông điệp quảng bá. Tuy nhiên hầu hết các Ethernet card hiện nay đều có thể được đặt ở chế độ

đa hỗn tạp (promiscuous mode) và khi đó nó sẽ nhận tất cả các frame xuất hiện trong mạng LAN.

- SA (Source Address): có độ dài 6 byte là địa chỉ MAC của card nguồn.
- Length/Type (Độ dài/Loại): 2 byte chỉ ra độ dài (đối với IEEE 802.3 MAC frame) và loại của Ethernet frame chỉ giao thức của tầng cao hơn (đối với DIX Ethernet.(DEC- Intel – Xerox) – phổ biến hơn). Ví dụ như với DIX Ethernet frame có giao thức tầng trên là IP thì 2 byte này sẽ có giá trị là 0800h và ARP là 0806h.
- Data Payload: Phần thông tin dữ liệu có độ dài từ 46 tới 1500 byte.
- Trailer (FCS - Frame Check Sequence): 32 bit sửa lỗi CRC.

Ethernet sử dụng phương thức truy nhập đường truyền CSMA/CD, do vậy những frame lỗi do xảy ra xung đột (collision) trên đường truyền là không thể tránh khỏi. Tuy nhiên, nếu như tỉ lệ những frame lỗi vượt quá một mức nào đó (ví dụ như 1% tổng số frame) có nghĩa là hệ thống mạng đã có vấn đề. Những Ethernet frame lỗi bao gồm:

- Frame có độ lớn nhỏ hơn 64 byte. (normal collision – xảy ra khá phổ biến).
- Frame có độ lớn lớn hơn 1518 byte.
- Frame có độ lớn phù hợp nhưng có phần CRC bị sai lệch (late collision – nếu có nhiều frame dạng này tức là hệ thống mạng đang gặp vấn đề nghiêm trọng).

2) ARP (*Address Resolution Protocol*).

Giao thức phân giải địa chỉ ARP là phương pháp tìm địa chỉ tầng liên kết (hay địa chỉ vật lý) khi biết địa chỉ tầng Internet (IP) hoặc một vài kiểu địa chỉ tầng mạng khác. ARP được sử dụng không chỉ để chuyển đổi địa chỉ đối với IP và Ethernet mà nó được cài đặt để làm việc với nhiều loại địa chỉ của các tầng các loại mạng khác nhau. Tuy nhiên, do sự phổ biến của IPv4 và Ethernet nên ARP chủ yếu được dùng để chuyển đổi từ địa chỉ IP thành địa chỉ MAC. Nó cũng được sử dụng đối với IP dựa trên các công nghệ LAN khác Ethernet như FDDI, Token Ring, IEEE 802.11 hay ATM.

Trong thực tế, khi truyền thông với máy chủ thay vì truy vấn địa chỉ vật lý của máy chủ, giao thức ARP sẽ sử dụng bộ đệm ARP (ARP cache). Bộ đệm lưu trữ các địa chỉ IP gần nhất đã được phân giải. Nếu địa chỉ MAC của địa chỉ IP đích được tìm thấy trong bộ đệm thì địa chỉ này sẽ được sử dụng để truyền thông.

Cấu trúc của một đơn vị dữ liệu giao thức ARP như sau:

bit offset	0 - 7	8 - 15	16 - 31
0	Hardware type (HTYPE)		Protocol type (PTYPE)
32	Hardware length (HLEN)	Protocol length (PLEN)	Operation (OPER)
64	Sender hardware address (SHA) (first 32 bits)		
96	Sender hardware address (SHA) (last 16 bits)		Sender protocol address (SPA) (first 16 bits)
128	Sender protocol address (SPA) (last 16 bits)		Target hardware address (THA) (first 16 bits)
160	Target hardware address (THA) (last 32 bits)		
192	Target protocol address (TPA)		

Bảng 2.2. Cấu trúc tổng quát của gói tin ARP.

- Hardware type (2 bytes): Quy định kiểu phần cứng của mạng.
- Protocol type (2 bytes): Quy định giao thức được dùng ở tầng Network.
- Hardware length (1 byte): Độ dài của địa chỉ vật lý.
- Protocol length (1 byte): Độ dài của địa chỉ logic.
- Operation (2 bytes): cho biết bản tin ARP là yêu cầu (=1) hoặc trả lời (=2).
- Sender hardware address (6 bytes): Địa chỉ MAC máy gửi.
- Sender protocol address (4 bytes): Địa chỉ IP máy gửi.
- Target hardware address (6 bytes): Địa chỉ MAC máy gửi.
- Target protocol address (4 bytes): Địa chỉ IP máy nhận.

3) RARP (*Reserve Address Resolution Protocol*).

Là giao thức ngược lại so với ARP, tìm địa chỉ logic khi biết địa chỉ vật lý. Cấu trúc của một đơn vị dữ liệu của giao thức RARP hoàn toàn tương tự như ARP, ngoại trừ trường Operation. Đối với gói dữ liệu ARP thì Operation có giá trị 1 nếu là request, 2 nếu reply. Đối với gói dữ liệu RARP thì Operation có giá trị 3 nếu là request và 4 nếu là reply.

4) IP (*Internet Protocol*).

Giao thức IP là một giao thức trung tâm của bộ giao thức TCP/IP. IP là một giao thức hướng dữ liệu được sử dụng trong mạng chuyển mạch. IP hoạt động theo phương thức connectionless và truyền dữ liệu không đảm bảo tin cậy (không có sự trao đổi thông tin điều khiển). Vai trò của IP tương tự như vài trò của giao thức tầng mạng trong mô hình OSI với các chức năng như sau:

- Xác định lược đồ địa chỉ Internet.
- Di chuyển dữ liệu giữa tầng giao vận và tầng liên kết.
- Dẫn đường cho các đơn vị dữ liệu tới các trạm ở xa.
- Thực hiện việc cắt và hợp các đơn vị dữ liệu.

Giao thức IP sẽ bổ sung phần header vào trước segment được gửi từ tầng giao vận xuống và gói tin trong giao thức IP được gọi là IP Datagram.

Cấu trúc của IP Datagram như sau:

<table>
<tr><th>0 - 3</th><th>4- 7</th><th>8 - 15</th><th>16 - 18</th><th>19 - 23</th><th>24 - 31</th></tr>
<tr><td>VER</td><td>HLEN</td><td>Service Type</td><td colspan="3">Total length</td></tr>
<tr><td colspan="3">Identification</td><td>Flags</td><td colspan="2">Fragment Offset</td></tr>
<tr><td colspan="2">Time to live</td><td>Protocol</td><td colspan="3">Header Checksum</td></tr>
<tr><td colspan="6">Source IP Address</td></tr>
</table>

Destination IP Address	
IP Option (if any)	Padding
Data	
…	

Bảng 2.3. Cấu trúc gói tin IP Datagram.

- VER (4 bits): Cho biết version hiện hành của giao thức IP.
- HLEN (4 bits): Cho biết kích thước của header tính bằng đơn vị word (4 bytes).
- Service of type (8 bits): mô tả thứ tự ưu tiên của các gói tin. Thứ tự ưu tiên đó như sau:

0 1 2 3 4 5 6 7

Precedence	D	T	R	Unused

+ Precedence (3 bits đầu): xác định 8 mức độ ưu tiên.

+ Bit D (bit 3): yêu cầu chuyển gấp.

+ Bit T (bit 4): yêu cầu truyền ở đường truyền có thông lượng tốt.

+ Bit R (bit 5): yêu cầu truyền bảo đảm.

+ Bit 6 và 7: không sử dụng.

- Total length (16 bits): xác định kích thước tối đa của Datagram, tính bằng bytes.
- Identification (16 bits): mã hiệu định danh cho IP Datagram.
- Flags (3 bits):

Bit 1	Bit 2	Bit 3

+ Bit 1: không dùng.

+ Bit 2: Datagram hoàn chỉnh = 0, là Fragment = 1.

+ Bit 3: Fragment thường = 0, là Fragment cuối cùng = 1.

- Fragment Offset (13 bits): Xác định vị trí tương đối theo bội số 8 của phần dữ liệu của Fragment trong IP Datagram nguyên thủy ban đầu, giúp ghép các Fragment lại thành gói tin hoàn chỉnh. có nghĩa là phần dữ liệu mỗi gói tin (trừ gói tin cuối cùng) phải chứa một vùng dữ liệu có độ dài là bội số của 8 bytes.
- Time to live (8 bits): Xác định thời gian tồn tại của IP Datagram ở trên mạng, tính bằng số lần đi qua các router.
- Protocol (8 bits): Xác định giao thức ở tầng bên trên.
- Header Checksum (16 bits): Là phần mã kiểm tra gói tin IP Datagram có bị lỗi hay không.
- Source IP Address (32 bits): Địa chỉ IP nguồn.
- Destination IP Address (32 bits): Địa chỉ IP đích.
- IP Option (24 bits): Phần các tùy chọn dùng cho mục đích đặc biệt. Khai báo các tuỳ chọn do nơi gửi yêu cầu. Trường option không bắt buộc phải có trong mọi gói IP datagram và chủ yếu dùng để kiểm tra lỗi trên mạng. Option là một phần quan trọng của giao thức IP nên mọi tiêu chuẩn thực hiện dựa trên IP phải bao gồm tiến trình xử lý trường này. Độ dài của trường Option thay đổi tuỳ thuộc vào các tham số đi kèm. Khi các Option xuất hiện trong datagram, nó sẽ kéo dài liên tục mà không có sự ngắt quãng.
- Padding (8 bits): Chứa các bit 0.

5) ICMP (*Internet Control Message Protocol*).

Giao thức ICMP cung cấp cơ chế thông báo lỗi và các tình huống không mong muốn cũng như điều khiển các thông báo trong bộ giao thức TCP/IP. Giao thức này được tạo ra để thông báo các lỗi dẫn đường cho trạm nguồn. ICMP phụ thuộc vào IP để có thể hoạt động và là một phần không thể thiếu của bộ giao thức TCP/IP, tuy nhiên nó không phải giao thức dùng để truyền tải dữ liệu nên thường được coi nằm trong tầng Internet mà không phải là tầng giao vận.

Chức năng của ICMP như sau:

- Cung cấp thông báo phản hồi và trả lời để kiểm tra độ tin cậy của kết nối giữa hai trạm. Điều này được thiết lập bởi câu lệnh PING (Packet internet gopher).
- Định hướng lại lưu lượng để cung cấp việc dẫn đường hiệu quả hơn khi một bộ dẫn đường quá tải dõ lưu lượng qua nó quá lớn.
- Gửi thông báo về thời gian quá khi datagram của trạm nguồn đã vượt quá TTL và bị loại bỏ.
- Gửi quảng cáo dẫn đường để xác định địa chỉ của các bộ dẫn đường trên đoạn mạng.
- Cung cấp các thông báo quá hạn thời gian.

- Xác định subnet mask nào được sử dụng trên đoạn mạng.

  Cấu trúc của gói tin ICMP như sau:

0 - 7	8 - 15	16 - 31
Type	Code	Checksum
Identifier		Sequence number
Data		

Bảng 2.4. Cấu trúc tổng quát của gói tin ICMP.

- Type (1 byte): Kiểu của thông điệp ICMP
- Code (1 byte): Loại thông điệp ICMP
- Checksum (2 bytes): Mã kiểm tra gói tin ICMP.
- Identifier (2 bytes): Mã số của gói tin ICMP. Nếu chỉ gửi đi 1 gói tin ICMP thì trường này có thể bỏ qua.
- Sequence number (2 bytes): Số thứ tự của gói tin ICMP

Type	Code	Description	Loại ICMP
0	0	Echo reply	Echo reply
3		Destination unreachable	Đích không đến được
	0	Network unreachable	Mạng không đến được
	1	Host unreachable	Host không đến được

	2	Protocol unreachable	Host không hỗ trợ giao thức mà bên gửi dùng
	3	Port unreachable	Không thể thiết lập kết nối với cổng đã chọn
	4	Fragmentation needed and DF set.	Gói tin cần phải được phân mảng nhưng bit DF (Cấm phân mảnh) được bật.
	5	Source route failed	Không thể định tuyến cho gói tin.
4		Source quench	Giảm tốc độ nguồn phát
5		Redirect	Gửi lại (chọn tuyến đường khác)
	0	Redirect datagrams for the network	Gửi lại gói tin cho mạng
	1	Redirect datagrams for the host	Gửi lại gói tin cho host
	11	Redirect datagrams for the type of service and network	Gửi lại gói tin cho ToS và mạng
	12	Redirect datagrams for the type of service and host	Gửi lại gói tin cho ToS và host

8	0	Echo request	Echo request
11		Time exceeded	Time exceeded
	0	Time to Live exceeded in transmit	Time exceeded do gói tin hết TTL
	1	Fragment re-assemble time exceeded	Time exceeded do một fragment của gói tin bị time out.
12		Parameter Problem	Tham số không đúng
13		Timestamp	Timestamp
14		Timestamp reply	Timestamp Reply
15		Information request	Information request
16		Information reply	Information reply

Bảng 2.5. Chi tiết kiểu và loại gói tin ICMP.

6) TCP (*Transmission Control Protocol*).

Giao thức điều khiển truyền TCP là một giao thức hoạt động theo phương thức có kết nối (connection – oriented). Trong bộ giao thức TCP/IP, nó là giao thức trung gian giữa IP và một ứng dụng phía trên, đảm bảo dữ liệu được trao đổi một cách tin cậy và đúng thứ tự. Các ứng dụng sẽ gửi các dòng gồm các byte 8 bit tới TCP để gửi qua mạng. TCP sẽ phân chia các dòng này thành các đoạn (segment) có kích thước thích hợp (thường dựa theo kích thước của đơn vị truyền dẫn tối đa MTU của tầng liên kết của mạng mà máy tính đang nằm trong đó). Sau đó TCP chuyển các gói tin thu được xuống tầng dưới. Trong quá trình này, nó sẽ có cơ chế bắt tay, điều khiển truyền, đánh số thứ tự và sửa lỗi để việc truyền dẫn diễn ra đúng đắn và chính xác.

Đơn vị dữ liệu của TCP được gọi là Segment bao gồm 2 phần: Header và Data, được miêu tả dưới hình sau:

<table>
<tr><th>0 - 3</th><th>4 - 9</th><th>10 - 15</th><th>16 - 23</th><th>24 - 31</th></tr>
<tr><td colspan="3">Source Port</td><td colspan="2">Destination Port</td></tr>
<tr><td colspan="5">Sequence Number</td></tr>
<tr><td colspan="5">Acknowledgement Number</td></tr>
<tr><td>HLEN</td><td>Reserved</td><td>Code Bits</td><td colspan="2">Window</td></tr>
<tr><td colspan="3">Checksum</td><td colspan="2">Urgent Pointer</td></tr>
<tr><td colspan="4">Options (if any)</td><td>Padding</td></tr>
<tr><td colspan="5">Data</td></tr>
<tr><td colspan="5">…</td></tr>
</table>

Bảng 2.6. Cấu trúc gói tin trong giao thức TCP.

- Source Port (16 bits): Cổng gửi.
- Destination Port (16 bits): Cổng nhận.
- Sequence Number (32 bits): Có hai vai trò:

  + Nếu cờ SYN được bật (1), thì đây là số thứ tự ban đầu (init sequence number).Số thứ tự của byte dữ liệu tiếp theo bằng với số xác nhận vừa nhận được, và bằng với số thứ tự trước đó cộng thêm 1.

  + Nếu cờ SYN tắt (0), thì đây là số thứ tự tích lũy của byte dữ liệu đầu tiên của segment này cho đến phiên hiện tại.

- Acknowledgement Number (32 bits):

    + Cờ ACK được bật thì giá trị này là số thứ tự tiếp theo mà bên nhận mong đợi.

    + Các ACK đầu tiên dùng để xác nhận số thứ tự ban đầu của bên vừa gửi, không chứa dữ liệu.

- HLEN (4 bits): Xác định kích thước của phần header. (Tối thiểu là 160 bits -> Tối đa là 480 bits)
- Reserved (6 bits): Dự phòng.
- Code Bits (6 bits): Xác định chi tiết kiểu segment.

URG	ACK	PSH	RST	SYN	FIN

    + URG: Thông báo đây là gói tin khẩn cấp cần gửi đi ngay.

    + ACK: Để báo đây là 1 segment ACK.

    + PSH: Để thông báo gói tin ở trạng thái đang được gủi đi.

    + RST: Thiết lập lại đường truyền.

    + SYN: Để bắt tay thiết lập chế độ đồng bộ.

    + FIN: Báo kết thúc kết nối, giải phóng các vùng nhớ đệm.

- Window (16 bits): Kích thước cửa sổ trượt.
- Checksum (16 bits): Kiểm tra lỗi của header và data
- Urgent Pointer (16 bits): Là Sequence Number dịch trái nếu cờ URG được bật.
- Options: Tùy chọn (nếu có).
- Padding (8 bits): Chứa các bit 0.

7) UDP (*User Datagram Protocol*).

Đây là một giao thức không kết nối. Khác với TCP, UDP không có các chức năng thiết lập và giải phóng kết nối. Nó cũng không cung cấp các cơ chế báo nhận, không sắp xếp tuần tự các đơn vị dữ liệu đến và có thể dẫn tới tình trạng dữ liệu mất hoặc trùng mà không hề có thông báo lỗi cho người gửi. Tóm lại nó cung cấp các dịch vụ giao vận không tin cậy. Do ít chức năng phức tạp nên UDP có xu thế hoạt động nhanh hơn so với TCP. Nó thường được dùng cho các ứng dụng không đòi hỏi độ tin cậy cao trong giao vận.

Cấu trúc của một đơn vị dữ liệu UDP như sau:

0 – 15	16 – 31
Source Port	Destination Port
Length	Checksum
Data (if any)	

Bảng 2.7. Cấu trúc gói tin trong giao thức UDP.

- Source Port (16 bits): Cổng gửi
- Destination Port (16 bits): Cổng nhận
- Length (16 bits): Độ dài của UDP Datagram tính theo bytes.
- Checksum (16 bits): Mã sửa sai, dùng để máy đích kiểm tra rằng UDP Datagram được giữ nguyên vẹn trên đường đi.

2.2. Kỹ thuật giám sát gói tin trên mạng

2.2.1. Các thành phần của chương trình giám sát gói tin

- **Phần cứng:** Để có thể chặn bắt các gói tin vào/ra một mạng gián tiếp từ một nút mạng thì card mạng của nút mạng đó phải hỗ trợ chế độ promiscuous. Hầu hết các card mạng hiện nay đều hỗ trợ chế độ này. Tuy nhiên, các mạng hiện nay đang dần chuyển sang sử dụng switch thay vì broadcast gói tin như hub, vậy nên để chặn bắt gói tin trong một mạng không còn đơn giản như trước đây nữa.
- **Capture driver:** Là phần quan trọng nhất. Nó có nhiệm vụ bắt lấy network traffic trên đường dây, lưu trữ dữ liệu vào buffer và lọc lấy thông tin cần thiết.
- **Buffer**: Dữ liệu sau khi được lấy về sẽ được lưu trữ tạm thời tại buffer. Thường có 2 phương pháp sử dụng buffer: ghi vào cho tới khi buffer đầy, hoặc ghi theo phương pháp vòng tròn khi mà dữ liệu mới nhất sẽ thay thế dữ liệu cũ nhất.
- **Real-time analysis**: Phân tích traffic về protocol, kiểm tra lỗi khi capture packet.
- **Decode**: Giải mã và hiển thị nội dung của network traffic dưới dạng phù hợp tùy thuộc vào yêu cầu.

2.2.2. Cách hoạt động

- **Theo dõi Network Traffic:** Ethernet được xây dựng dựa trên khái niệm chia sẻ. Tất cả các máy trong một mạng nội bộ đều được chia sẻ chung một đường dây. Điều đó chỉ ra rằng tất cả các máy trong mạng đều có thể "nhìn thấy" traffic trong đường dây đó. Do đó, phần cứng Ethernet sẽ có một bộ lọc ("filter") bỏ qua tất cả những traffic không phải dành cho nó (bằng cách bỏ qua tất cả các frame có địa chỉ MAC không phù hợp). Để nhìn thấy tất cả traffic, sniffer phải có cơ chế tắt "filter" ở trên, đưa phần cứng Ethernet vào chế độ promiscuous.

- **Phân tích Network Traffic**: Khi dữ liệu được gửi trên đường dây, nó sẽ được chia nhỏ, đóng gói thành nhiều packet và được gửi đi một cách riêng biệt. Sniffer là chương trình sẽ chặn bắt các packet này. Sau khi đã tiến hành chặn bắt thành công các gói tin, chúng ta sẽ có được các packet mang thông tin. Tuy nhiên, để lấy được thông tin cần thiết phục vụ cho các mục đích khác nhau, chúng ta phải thực hiện việc phân tích các gói tin đó (Packet Analysis), trích xuất thông tin và chuyển thành các trường thông tin có thể dễ dàng đọc bởi con người. Các giao thức có thể sniffing như: Ethernet, IPv4, IPv6, ARP/RARP, TCP, UDP, hoặc ICMPv4, telnet, rlogin, http, SNMP, NNTP, POP, FTP, IMAP…

2.2.3. Phương pháp chặn bắt gói tin

Phương pháp bắt gói tin được sử dụng trong sniffer có thể được tóm tắt thành hai loại: một là cơ chế bắt gói được cung cấp bởi nhân hệ điều hành (socket); phần còn lại được điều khiển bởi phần mềm ứng dụng (pcap).

Socket là một phương pháp để thiết lập kết nối truyền thông giữa một chương trình yêu cầu dịch vụ (client) và một chương trình cung cấp dịch vụ (server) trên mạng LAN, WAN hay Internet và đôi lúc là giữa những quá trình ngay bên trong máy tính.

Các đặc tính của Socket bao gồm:

- Giao thức (TCP, UDP hay raw IP).
- Số hiệu cổng.
- Địa chỉ IP.

Phân loại: Có vài loại Socket thông dụng như:

- Datagram Socket hay còn gọi là connectionless socket sử dụng UDP.

- Stream Socket hay còn gọi là connection – oriented socket sử dụng TCP.
- Raw Socket (hay là Raw IP Socket) là một socket cho phép truy nhập trực tiếp tới header của một packet. Nói một cách khác, Raw socket là một cách bỏ qua toàn bộ Network Stack và đưa packet tới thẳng tầng ứng dụng. Raw Socket có thể thực hiện một trong hai tác vụ:
    - Packet Sniffing: nhận các packet từ raw socket.
    - Packet Injection: gửi các packet tới raw socket.

Tóm lại, sử dụng Socket ta có thể chặn bắt và truy nhập các thông tin từ tầng giao vận trở lên (TCP và UDP) và có thể truy nhập tới tầng Internet (IP) nếu sử dụng Raw Socket. Tuy nhiên hiện nay trên hệ điều hành window chỉ có thư viện winsock dành cho Visual C++ và Socket trong .Net hỗ trợ Raw Socket.

Pcap (packet capture) bao gồm những giao diện lập trình ứng dụng (API) dùng để chặn bắt network traffic. Đối với các hệ thống thuộc họ Unix ta có thư viện libpcap, còn đối với Window ta có thư viện được port từ libpcap là winpcap. Libpcap và Winpcap được sử dụng từ link layer trở lên. Chúng cung cấp cơ chế packet capture và packet filter, có thể lưu trữ packet thu được vào file hay đọc file đó... Ngoài ra chúng còn cho phép tạo các custom packet và injection chúng trên mạng. Tuy nhiên nhược điểm của việc sử dụng các thư viện này là chúng chỉ có thể sử dụng để chặn bắt gói tin mà không thể block một địa chỉ hay một cổng hay một tiến trình truy nhập mạng như socket. API của libpcap và winpcap được viết bằng C hoặc C++ nên để có thể xây dựng ứng dụng bằng các ngôn ngữ khác như .NET, Java ta cần có wrapper. Trong đó trên nền tảng .NET thì thư viện Sharppcap cung cấp khá đầy đủ các chức năng phục vụ cho Sniffer.

Raw socket và Pcap đều có thể được sử dụng để viết chương trình sniffer. Tuy nhiên khi sử dụng Socket có một số hạn chế hơn so với Pcap như sau:

- Socket chỉ có thể làm việc từ tầng thứ 4 trong mô hình OSI trở lên (transport layer trong TCP/IP) và Raw Socket có thể làm việc được với tầng thứ 3 trong mô hình OSI trở lên (network layer trong TCP/IP) còn pcap có thể làm việc với tầng thứ 2 trở lên trong mô hình OSI (link layer trong TCP/IP).
- Ngoài ra Raw Socket trên Window đã không còn được Microsoft hỗ trợ cũng như tính năng bị giới hạn như:

    + Dữ liệu TCP không thể được gửi qua Raw Socket.

+ UDP datagram với địa chỉ nguồn không hợp lệ sẽ không thể gửi qua Raw Socket.

Do vậy, nếu sử dụng Socket để giám sát toàn bộ lưu lượng thông tin vào/ra một hệ thống hay một trạm thì sẽ dẫn tới kết quả có thể không chính xác do nó chỉ có thể chặn bắt một số loại packet nhất định (TCP và UDP) (IP nếu như sử dụng raw IP socket). Các giao thức với các gói dữ liệu khác như ARP, RARP, ICMP ta sẽ không thể chặn bắt khi sử dụng socket.

3. Nền tảng công nghệ được sử dụng - SharpPcap:

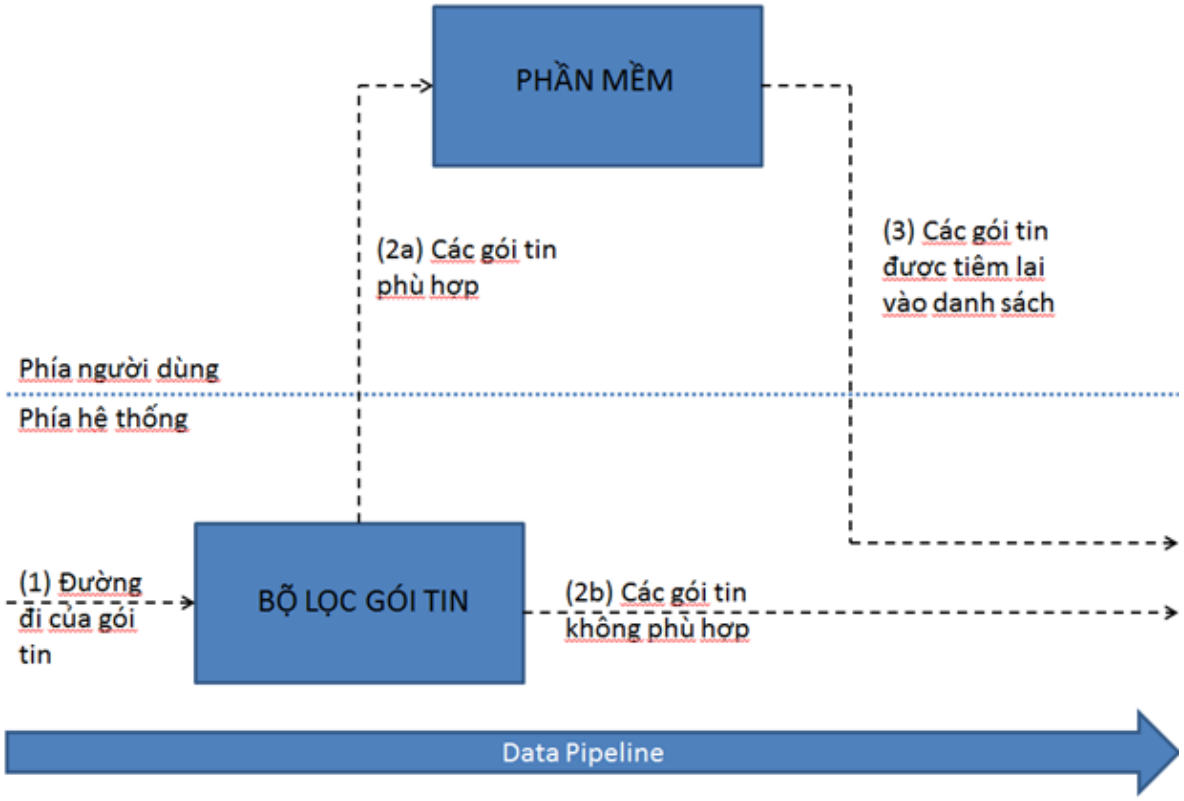
3.1. Khái niệm:

- Là một framework chặn bắt gói tin đa nền tảng cho môi trường .NET, được xây dựng dựa trên thư viện pcap/WinPcap nổi tiếng. SharpPcap cung cấp API cho việc chặn bắt, tiêm, phân tích và xây dựng các gói tin sử dụng các ngôn ngữ .NET như C# và VB.NET.

3.2. Đặc điểm chính

- Kết hợp với Packet.NET để phân tách và xây dựng danh sách gói tin. (phân tích Packet.NET rõ hơn ở phần phụ lục 1)
- Hỗ trợ cho nhiều hệ điều hành:
  - Trên Linux, hỗ trợ thư viện libpcap.
  - Trên Windows, hỗ trợ thư viện Npcap hoặc WinDivert.
- Hỗ trợ lập danh sách các thiết bị/network interface đang hoạt động.
- Hỗ trợ thống kê dữ liệu, thông tin các packets chặn bắt được trên đường truyền mạng.
- Hỗ trợ đọc sâu các gói tin từ các thiết bị/network interface trực tiếp hoặc từ các thiết bị/file chặn bắt đã được chuẩn bị từ trước.
- Hỗ trợ lưu danh sách các gói tin đã bắt được dưới file dạng Pcap hoặc Pcap-ng.
- Hỗ trợ ReadOnlySpan<>: được sử dụng để tránh các hoạt động cấp phát bộ nhớ và sao chép bất thường bên trong SharpPcap; từ đó cung cấp hiệu suất tốt nhất. (để duy trì danh sách các gói tin chặn bắt trong bộ nhớ, SharpPcap cung cấp ‘Helper methods’ để chuyển đổi thành các đối tượng cụ thể)
- Hỗ trợ NativeLibrary:
  - Hoạt động xuyên suốt giữa các hệ điều hành Linux, OSX và Windows.
  - Hỗ trợ đọc các file libpcap trên Linux trên mọi phiên bản.
- Hỗ trợ .NET Core 3 và .NET Framework.

3.3. Kiến trúc cơ bản của hệ thống SharpPcap



Biểu đồ 3.1. Kiến trúc cơ bản của hệ thống SharpPcap

(1) Các gói tin trên đường truyền mạng đi qua bộ lọc gói tin.

(2a) Nếu các gói tin phù hợp, một hàm sẽ được gọi để đọc gói tin.

(2b) Nếu gói tin không phù hợp, gói tin tin sẽ tiếp tục di chuyển trên đường truyền mạng như bình thường.

(3) Các gói tin phù hợp được phần mềm thu bắt rồi được tiêm lại vào danh sách.

3.4. Các hàm chức năng chính

- **Thu thập danh sách các thiết bị:**

```
/* Thu thập danh sách các thiết bị có sẵn */
PcapDeviceList devices = SharpPcap.GetAllDevices();

/* Nếu không có thiết bị nào khả dụng, hiện ra thông báo lỗi trên màn hình */
if (devices.Count < 1)
{
    Console.WriteLine("No device found on this machine");
    return;
}

int i = 0;

/* Quét danh sách và in ra thông tin thiết bị khả dụng */
foreach (PcapDevice dev in devices)
{
    /* Mô tả */
    Console.WriteLine("{0}) {1}", i, dev.PcapDescription);
    Console.WriteLine();
    /* Tên thiết bị */
    Console.WriteLine("\tName:\t{0}", dev.PcapName);
    /* Địa chỉ IP */
    Console.WriteLine("\tIP Address: \t\t{0}", dev.PcapIpAddress);
    /* Hỗ trợ Is Loopback */
    Console.WriteLine("\tLoopback: \t\t{0}", dev.PcapLoopback);

    Console.WriteLine();
    i++;
}
```

> **Output**:

```
SharpPcap 1.0.2.0
The following devices are available on this machine:
----------------------------------------------------
0) Generic dialup adapter
Name:   \Device\NPF_GenericDialupAdapter
IP Address:             0.0.0.0
Loopback:               False
1) Intel(R) PRO/1000 MT Mobile Connection (Microsoft's Packet Scheduler)
Name:   \Device\NPF_{355BF264-B768-454A-84BC-096A44F0ADA9}
IP Address:             10.10.10.100
Loopback:               False
Hit 'Enter' to exit...
```

- **Thu thập các thông tin nâng cao về các thiết bị khả dụng:**

```
/* Quét danh sách và in ra thông tin thiết bị khả dụng */
foreach (PcapDevice dev in devices)
{
    /* Mô tả */
    Console.WriteLine("{0}) {1}", i, dev.PcapDescription);
    Console.WriteLine();
    /* Tên thiết bị */
    Console.WriteLine("\tName:\t\t{0}", dev.PcapName);
    /* Hỗ trợ Is Loopback */
    Console.WriteLine("\tLoopback:\t\t{0}", dev.PcapLoopback);
    /* Nếu người dùng đang sử dụng thiết bị mạng vật lý, in thêm các thông tin nâng cao */
    if (dev is NetworkDevice)
    {//Then..
        /* Gọi tới NetworkDevice để thu thập thêm thông tin */
        NetworkDevice netDev = (NetworkDevice)dev;
        /* Các thông tin nâng cao */
        Console.WriteLine("\tIP Address:\t\t{0}", netDev.IpAddress);
        Console.WriteLine("\tSubnet Mask:\t\t{0}", netDev.SubnetMask);
        Console.WriteLine("\tMAC Address:\t\t{0}", netDev.MacAddress);
        Console.WriteLine("\tDefault Gateway:\t{0}", netDev.DefaultGateway);
        Console.WriteLine("\tPrimary WINS:\t\t{0}", netDev.WinsServerPrimary);
        Console.WriteLine("\tSecondary WINS:\t\t{0}", netDev.WinsServerSecondary);
        Console.WriteLine("\tDHCP Enabled:\t\t{0}", netDev.DhcpEnabled);
        Console.WriteLine("\tDHCP Server:\t\t{0}", netDev.DhcpServer);
        Console.WriteLine("\tDHCP Lease Obtained:\t{0}", netDev.DhcpLeaseObtained);
        Console.WriteLine("\tDHCP Lease Expires:\t{0}", netDev.DhcpLeaseExpires);
    }
    Console.WriteLine();
    i++;
}
```

> **Output**:

```
0) Generic dialup adapter
Name:           \Device\NPF_GenericDialupAdapter
Loopback:               False
1) Intel(R) PRO/1000 MT Mobile Connection (Microsoft's Packet Scheduler)
Name:           \Device\NPF_{355BF264-B768-454A-84BC-096A44F0ADA9}
Loopback:               False
IP Address:             10.10.10.100
Subnet Mask:            255.255.255.0
MAC Address:            000D60CDB2A5
Default Gateway:        10.10.10.1
Primary WINS:           1.1.1.1
Secondary WINS:         2.2.2.2
DHCP Enabled:           True
DHCP Server:            10.10.10.1
DHCP Lease Obtained:    27-Nov-05 11:13:55
DHCP Lease Expires:     30-Nov-05 11:13:55
Hit 'Enter' to exit...
```

- **Mở adapter và bắt đầu chặn bắt gói tin trên đường truyền mạng:**

```
/* Trích xuất một thiết bị từ danh sách */
PcapDevice device = devices[i];
/* Gọi hàm xử lý khi có gói tin đến */
device.PcapOnPacketArrival +=
new SharpPcap.PacketArrivalEvent(device_PcapOnPacketArrival);
/* Mở thiết bị để bắt đầu chặn bắt gói tin
true -- chế độ bắt tất cả gói tin trên lưu lượng mạng
//default -- chế độ bắt tất cả gói tin tới máy
1000 -- tốc độ đọc = 1000ms */
device.PcapOpen(true, 1000);
//device.PcapOpen(1000) - chế độ default
Console.WriteLine("-- Listenning on {0}, hit 'Enter' to stop...",
device.PcapDescription);
/* Bắt đầu bắt gói tin */
device.PcapStartCapture();
/* Đợi người dùng ấn nút "Enter" để dừng lại */
Console.ReadLine();
/* Dừng việc bắt gói tin */
device.PcapStopCapture();
/* Đóng thiết bị pcap */
device.PcapClose();
```

- **Chặn bắt gói tin liên tục:**

```
/* Trích xuất một thiết bị từ danh sách */
PcapDevice device = devices[i];
/* Mở thiết bị để bắt đầu chặn bắt gói tin
true -- chế độ bắt tất cả gói tin trên lưu lượng mạng
//default -- chế độ bắt tất cả gói tin tới máy
1000 -- tốc độ đọc = 1000ms */
device.PcapOpen(true, 1000);
Console.WriteLine();
Console.WriteLine("-- Listenning on {0}...",
device.PcapDescription);
Packet packet = null;
/* Sử dụng hàm PcapGetNextPacket() để liên tục bắt gói tin */
while ((packet = device.PcapGetNextPacket()) != null)
{
    /* In ra thời gian và độ dài các gói tin bắt được */
    DateTime time = packet.PcapHeader.Date;
    int len = packet.PcapHeader.PacketLength;
    Console.WriteLine("{0}:{1}:{2},{3} Len={4}",
    time.Hour, time.Minute, time.Second, time.Millisecond, len);
}
/* Đóng thiết bị pcap */
device.PcapClose();
Console.WriteLine("-- Capture stopped, device closed.");
```

- **Lọc các gói tin phù hợp với cấu hình:**

```
/* Mở thiết bị để bắt đầu chặn bắt gói tin
true -- chế độ bắt tất cả gói tin trên lưu lượng mạng
//default -- chế độ bắt tất cả gói tin tới máy
1000 -- tốc độ đọc = 1000ms */
device.PcapOpen(true, 1000);
/* Thiết lập bộ lọc tcpdump để bắt các gói tin TCP/IP */
string filter = "ip and tcp";
/* Liên kết với bộ lọc trên */
device.PcapSetFilter(filter);
Console.WriteLine();
Console.WriteLine("-- The following tcpdump filter will be applied: \"{0}\"", filter);
Console.WriteLine("-- Listenning on {0}, hit 'Enter' to stop...",device.PcapDescription);
/* Bắt đầu bắt gói tin */
device.PcapCapture(SharpPcap.INFINITE);
/* Đóng thiết bị pcap */
device.PcapClose();
```

- **Phân tích các gói tin:**

```
/* In ra các thông tin về gói tin TCP/IP thu thập được trên lưu lượng mạng:
    thời gian chặn bắt, độ dài gói tin, địa chỉ IP nguồn, địa chỉ IP đích và cổng */
private static void device_PcapOnPacketArrival(object sender, Packet packet)
{
    if (packet is TCPPacket)
    {
        DateTime time = packet.Timeval.Date;
        int len = packet.PcapHeader.len;
        TCPPacket tcp = (TCPPacket)packet;
        string srcIp = tcp.SourceAddress;
        string dstIp = tcp.DestinationAddress;
        int srcPort = tcp.SourcePort;
        int dstPort = tcp.DestinationPort;
        Console.WriteLine("{0}:{1}:{2},{3} Len={4} {5}:{6} -> {7}:{8}",
        time.Hour, time.Minute, time.Second, time.Millisecond, len,
        srcIp, srcPort, dstIp, dstPort);
    }
}
```

> **Output**:

```
Available devices:
------------------1) Intel(R) PRO/1000 MT Mobile Connection (Microsoft's Packet Scheduler)
--Please choose a device to capture:
1-- Listenning on Intel(R) PRO/1000 MT Mobile Connection (Microsoft's Packet Scheduler)
...1:18:17,675 Len=123 66.102.7.147:80 -> 10.21.98.21:435011:18:17,675 Len=80
10.21.98.21:43501 -> 66.102.7.147:801:18:17,919 Len=54 66.102.7.147:80 -> 10.21.98.21:43501
```

- **Xử lý các file dump ngoại tuyến:**
    - **Lưu thông tin gói tin dưới dạng file libpcap:**

```
Console.Write("-- Please enter the output file name: ");
string capFile = Console.ReadLine();
PcapDevice device = devices[i];
/* Gọi hàm xử lý khi có gói tin đến */
eventdevice.PcapOnPacketArrival +=
new SharpPcap.PacketArrivalEvent( device_PcapOnPacketArrival );
/* Mở thiết bị để bắt đầu chặn bắt gói tin */
device.PcapOpen(true, 1000);
/* Mở hoặc tạo file output */
device.PcapDumpOpen( capFile );
Console.WriteLine();
Console.WriteLine("-- Listenning on {0}, hit 'Ctrl-C' to exit...",device.PcapDescription);
/* Bắt đầu chặn bắt các gói tin liên tục */
device.PcapCapture( SharpPcap.INFINITE );
/* Đóng thiết bị pcap */
device.PcapClose();
```

```
/* Dump mỗi gói tin vào file pcap */
private static void device_PcapOnPacketArrival(object sender, Packet packet)
{
    PcapDevice device = (PcapDevice)sender;
    /* Nếu thiết bị đang mở file dump */
    if( device.PcapDumpOpened )
    {
        /* Dump các gói tin thu thập được vào file đang mở */
        device.PcapDump( packet );
        Console.WriteLine("Packet dumped to file.");
    }
    /* Nếu không, dump các gói tin vào file pcap mới */
}
```

- **Đọc gói tin từ file dump:**

```
Console.Write("-- Please enter an input capture file name: ");
string capFile = Console.ReadLine();
PcapDevice device;
try
{
    /* Thu thập file ngoại tuyến từ thiết bị pcap */
    device = SharpPcap.GetPcapOfflineDevice( capFile );
    /* Mở thiết bị để bắt đầu chặn bắt gói tin */
    device.PcapOpen();
}
catch(Exception e)
{
    Console.WriteLine(e.Message);
    return;
}
/* Gọi hàm xử lý khi có gói tin đến */
device.PcapOnPacketArrival +=
new SharpPcap.PacketArrivalEvent( device_PcapOnPacketArrival );
Console.WriteLine();
Console.WriteLine("-- Capturing from '{0}', hit 'Ctrl-C' to exit...",capFile);
/* Bắt đầu chặn bắt các gói tin liên tục - kết thúc khi đi tới cuối file */
device.PcapCapture( SharpPcap.INFINITE );
/* Đóng thiết bị pcap */
device.PcapClose();
Console.WriteLine("-- End of file reached.");
```

- **Gửi các gói tin trên đường truyền mạng:**
    - **Gửi một gói tin đơn bằng SharpPcap:**

```
/* Mở thiết bị để bắt đầu chặn bắt gói tin */
device.PcapOpen();
/* Thu thập và tạo ra một gói tin bất kỳ */
byte[] bytes = GetRandomPacket();
try
{
    /* Gửi gói tin đó ra ngoài thiết bị mạng */
    device.PcapSendPacket( bytes );
    Console.WriteLine("-- Packet sent successfuly.");
}
catch(Exception e)
{
    Console.WriteLine("-- "+ e.Message );
}
/* Đóng thiết bị pcap */
device.PcapClose();
Console.WriteLine("-- Device closed.");
```

- **Gửi các gói tin theo hàng chờ:**

```
PcapDevice device;
try
{
    /* Thu thập file ngoại tuyến từ thiết bị pcap */
    device = SharpPcap.GetPcapOfflineDevice( capFile );
    /* Mở thiết bị để bắt đầu chặn bắt */
    device.PcapOpen();
}
catch(Exception e)
{
    Console.WriteLine(e.Message);
    return;
}
Console.Write("Queueing packets...");
/* Cấp phát một hàng chờ để gửi các gói tin */
PcapSendQueue squeue = new PcapSendQueue( (int)((PcapOfflineDevice)device).PcapFileSize );
Packet packet;
try
{
    /* Duyệt qua các gói tin và thêm vào hàng chờ */
    while( (packet=device.PcapGetNextPacket()) != null )
    {
        if( !squeue.Add( packet ) )
        {
            Console.WriteLine("Warning: packet buffer too small, "+"not all the packets will be sent.");
            break;
        }
    }
}
```

```
/* Quét danh sách và in ra thông tin thiết bị khả dụng */
foreach (PcapDevice dev in devices)
{
    /* Mô tả */
    Console.WriteLine("{0}) {1}",i,dev.PcapDescription);
    i++;
}
Console.WriteLine();
Console.Write("-- Please choose a device to transmit on: ");
i = int.Parse( Console.ReadLine() );
devices[i].PcapOpen();
string resp;
if(devices[i].PcapDataLink != device.PcapDataLink)
{
    Console.Write("Warning: the datalink of the capture"+" differs from the one of the selected interface, continue? [YES|no]");
    resp = Console.ReadLine().ToLower();
    if((resp!="")&&( !resp.StartsWith("y")))
    {
        Console.WriteLine("Cancelled by user!");
        devices[i].PcapClose();
        return;
    }
}
device.PcapClose();
```

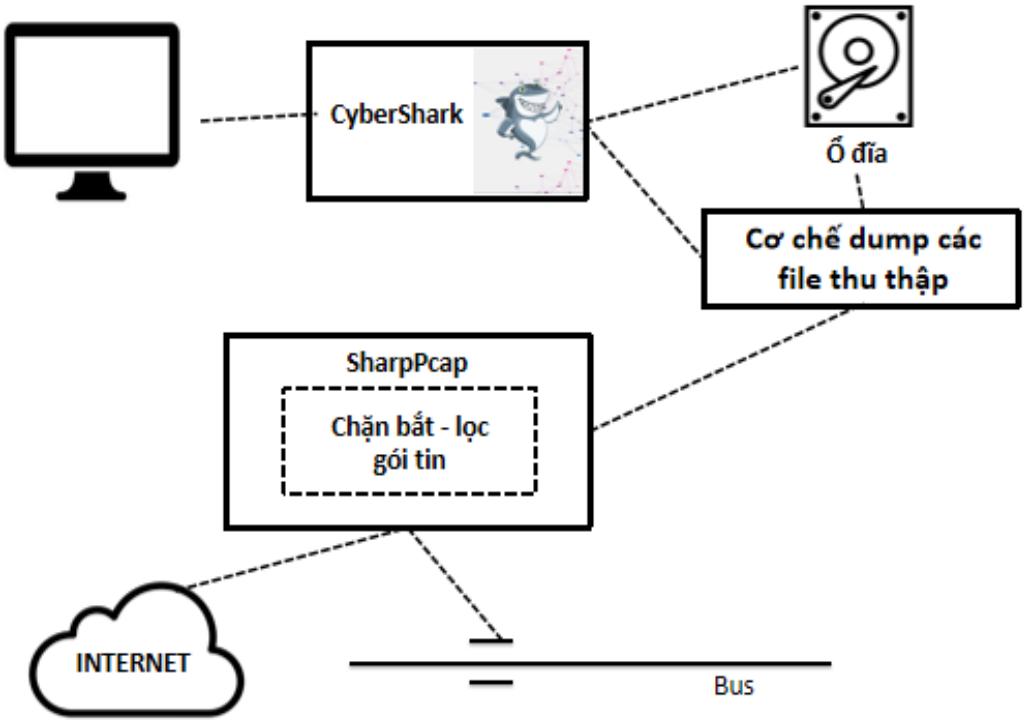
```
device = devices[i];
Console.Write("This will transmit all queued packets through"+" this device, continue? [YES|no]");
resp = Console.ReadLine().ToLower();
if((resp!="")&&( !resp.StartsWith("y")))
{
    Console.WriteLine("Cancelled by user!");
    return;
}
try
{
    Console.Write("Sending packets...");
    int sent = device.PcapSendQueue( squeue, true );
    Console.WriteLine("Done!");
    if( sent < squeue.CurrentLength )
    {
        Console.WriteLine("An error occurred sending the packets: {0}. "+"Only {1} bytes were sent\n", device.PcapLastError, sent);
    }
}
catch(Exception e)
{
    Console.WriteLine( "Error: "+e.Message );
}
/* Giải phóng hàng chờ */
queue.Dispose();
Console.WriteLine("-- Queue is disposed.");
/* Đóng thiết bị pcap */
device.PcapClose();
Console.WriteLine("-- Device closed.");
```

- **Thu thập thông tin thống kê trên lưu lượng mạng:**

```
/* Gọi hàm xử lý khi có sự kiện thống kê */
device.PcapOnPcapStatistics +=
new Tamir.IPLib.SharpPcap.PcapStatisticsEvent( device_PcapOnPcapStatistics );
/* Mở thiết bị để bắt đầu chặn bắt */
device.PcapOpen(true, 1000);
/* Chỉ thu thập các gói tin TCP */
device.PcapSetFilter( "tcp" );
 /* Đặt thiết bị về chế độ thống kê */
device.PcapMode = PcapMode.Statistics;
Console.WriteLine();
Console.WriteLine("-- Gathering statistics on \"{0}\", hit 'Enter' to stop...",device.PcapDescription);
/* Bắt đầu chặn bắt gói tin */
device.PcapStartCapture();
/* Đợi người dùng ấn nút "Enter" để dừng việc chặn bắt */
Console.ReadLine();
 /* Dừng chặn bắt gói tin */
device.PcapStopCapture();
/* Đóng thiết bị pcap */
device.PcapClose();
Console.WriteLine("Capture stopped, device closed.");
```

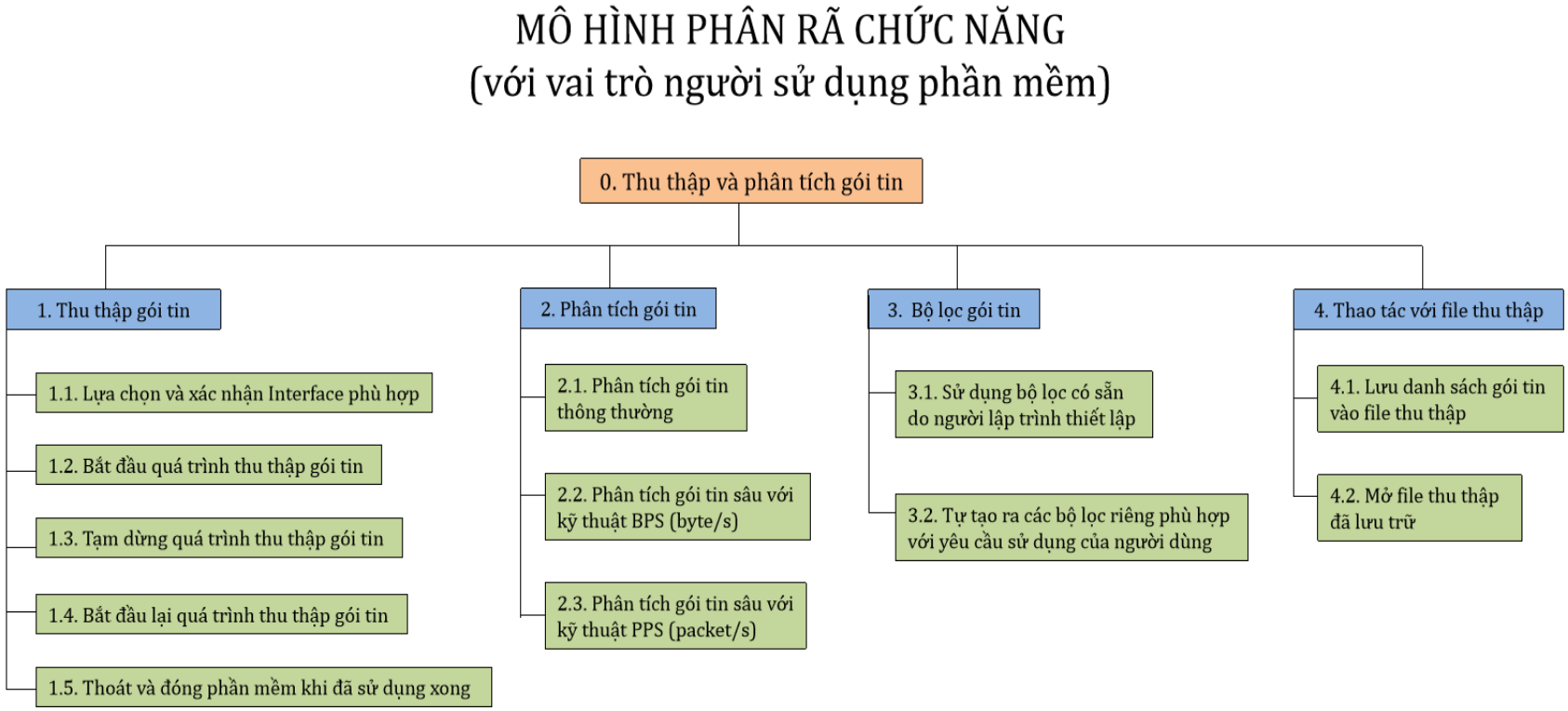
4. Xây dựng chương trình

4.1. Mô hình kiến trúc hệ thống



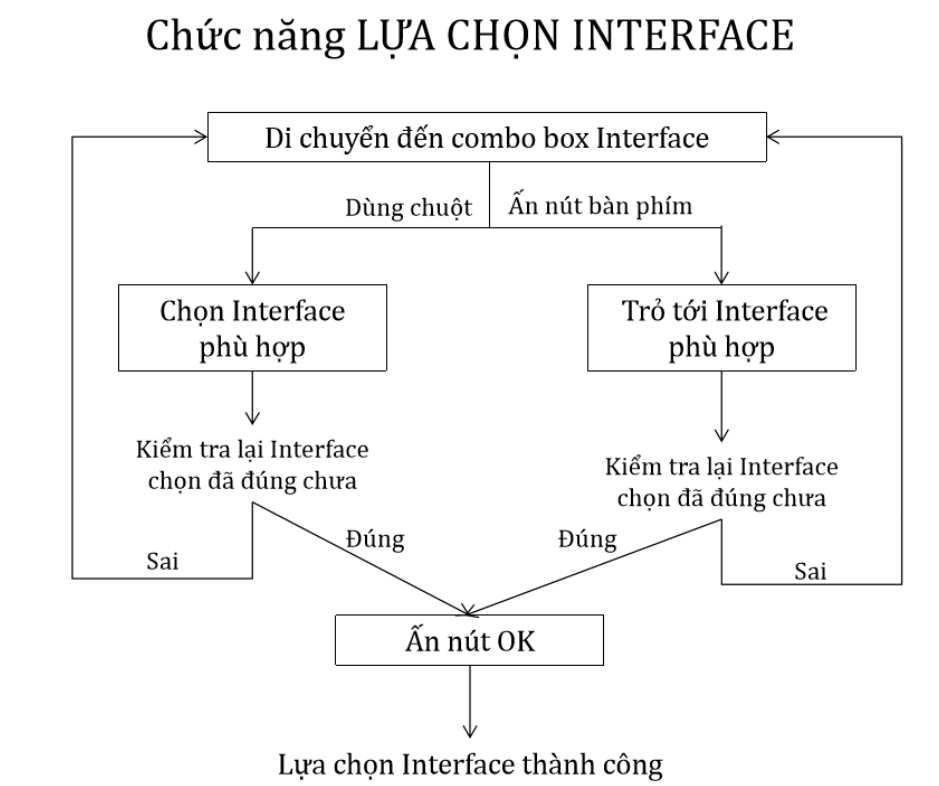
Biểu đồ 4.1. Mô hình kiến trúc hệ thống chặn bắt và phân tích gói tin CyberShark

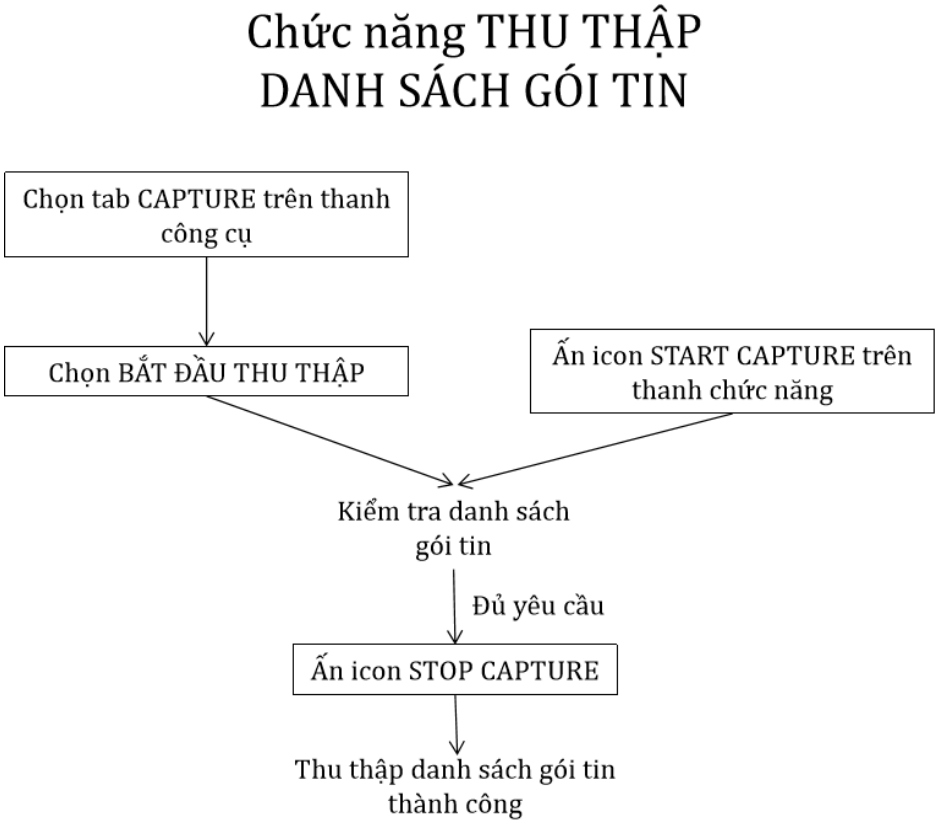
4.2. Mô hình phân rã chức năng



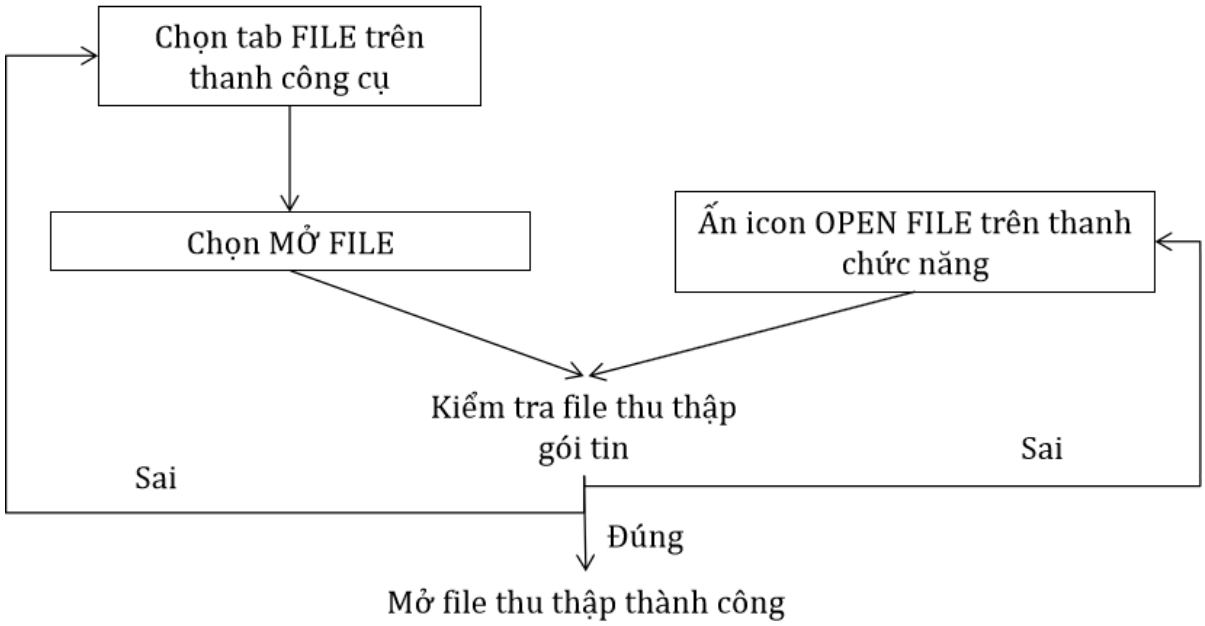
Biểu đồ 4.2. Mô hình phân rã chức năng phần mềm CyberShark

4.3. Mô hình giao diện người dùng (mô tả theo biểu đồ GOMS)

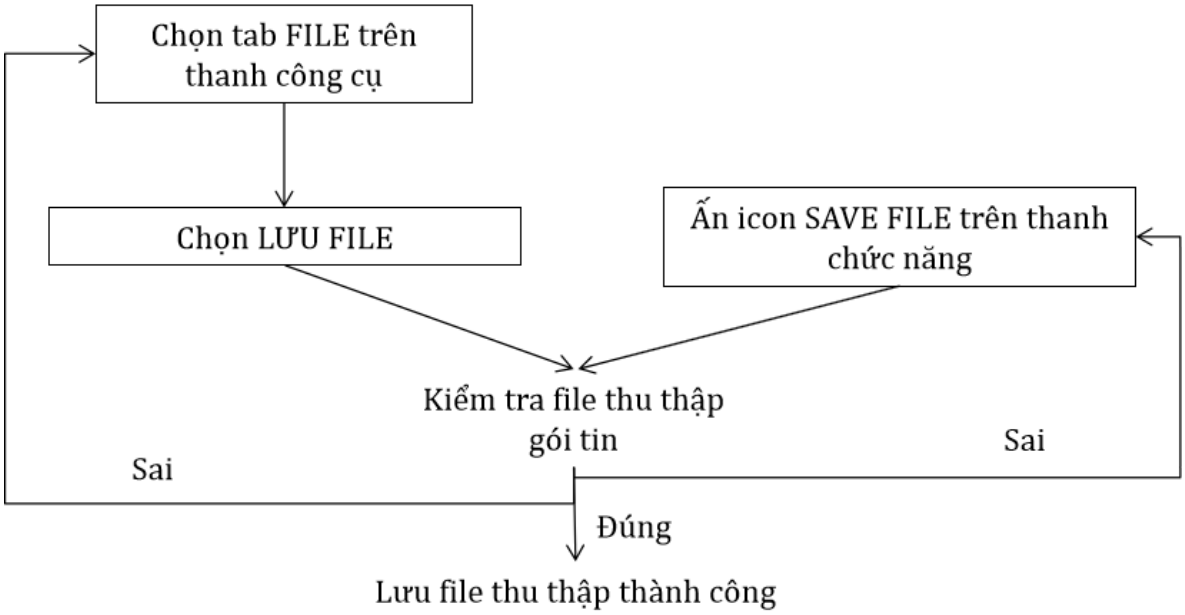




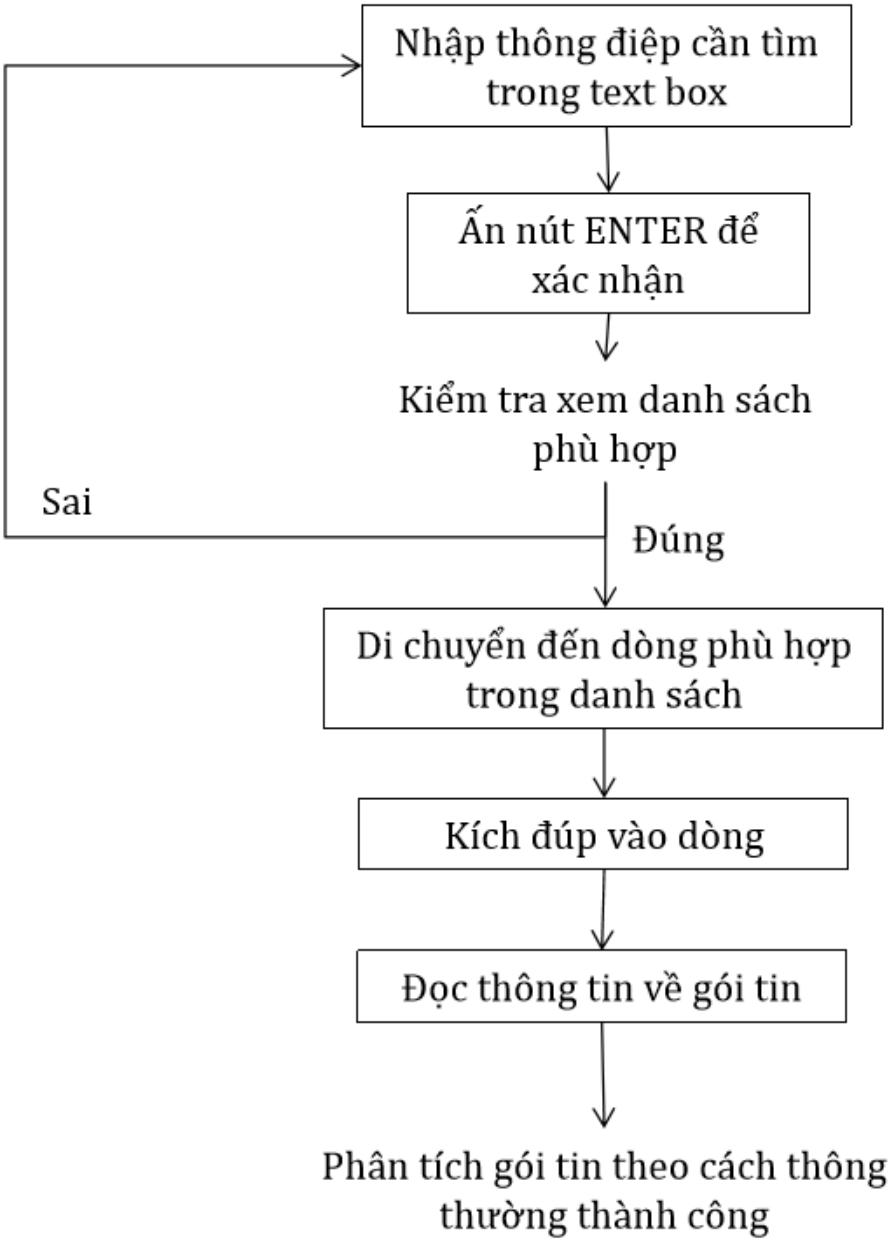
Chức năng MỞ FILE THU THẬP LƯU TRỮ



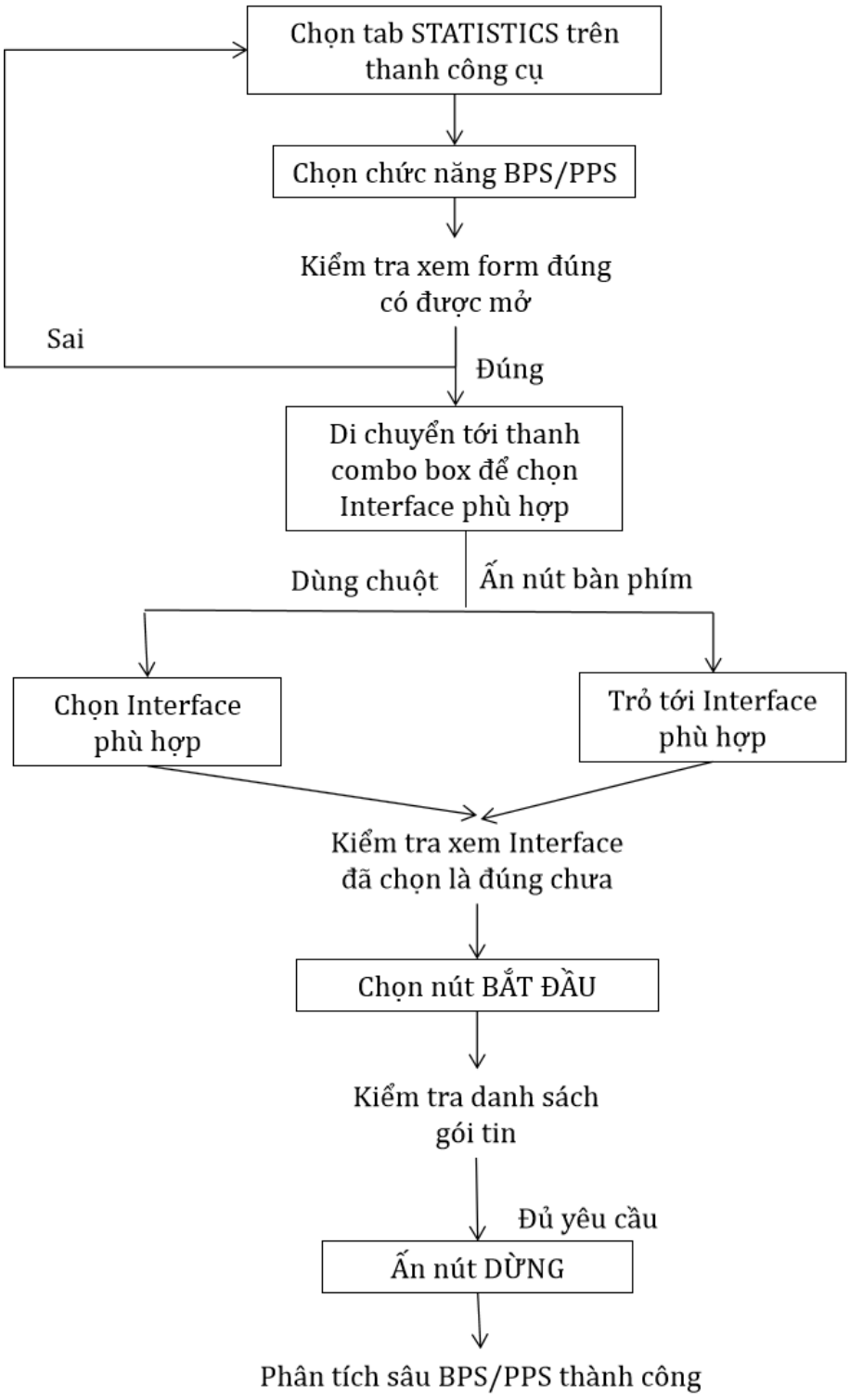
Chức năng LƯU FILE THU THẬP LƯU TRỮ



Chức năng PHÂN TÍCH GÓI TIN THÔNG THƯỜNG



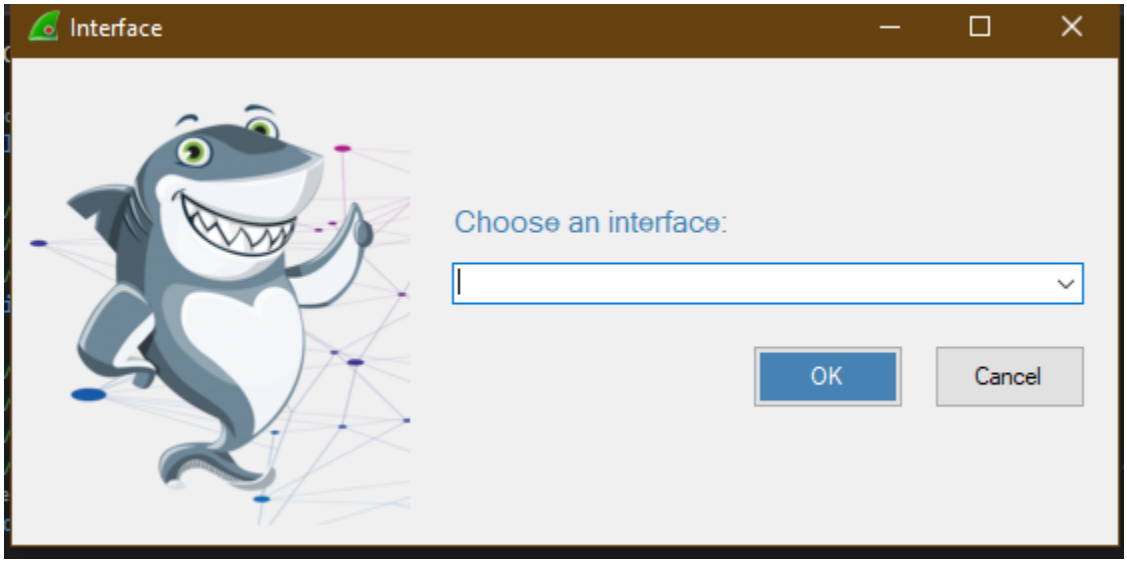
Chức năng PHÂN TÍCH GÓI TIN SÂU BPS/PPS



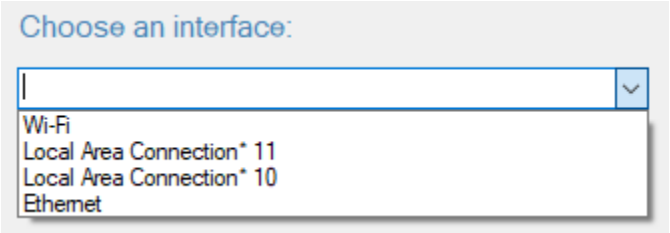
5. Giao diện chương trình

5.1. Giao diện mở đầu:

Dưới đây là giao diện chào mừng của phần mềm; cung cấp cho người dùng chức năng lựa chọn Interface để thu thập gói tin.



Người dùng lựa chọn Interface để thu thập gói tin. (Wifi, Ethernet, ...)

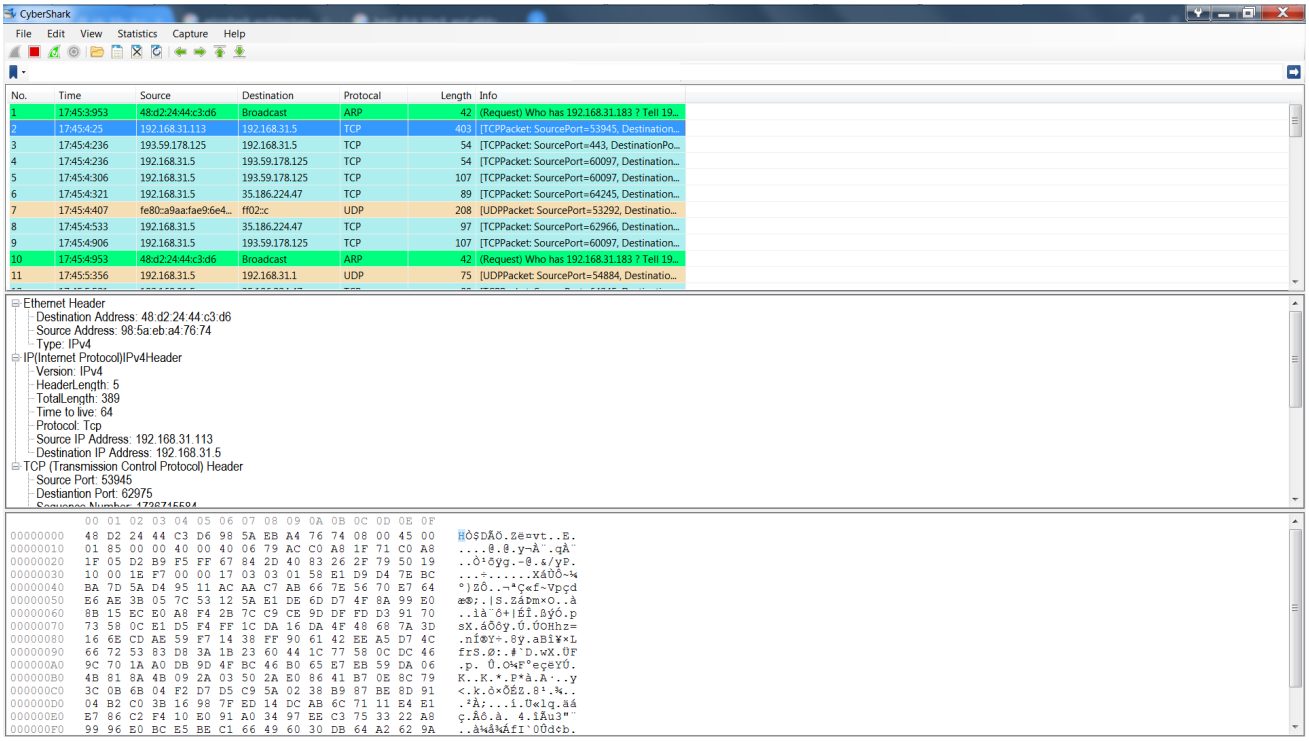


Đặc tả giao diện:

Interface - giao diện lựa chọn interface sử dụng			
STT	**Tên thành phần**	**Định dạng**	**Mô tả chức năng**
1	mInterfaceCombo	combo box	Danh sách các Interface có sẵn
2	button1	button	Xác nhận Interface và bắt đầu sử dụng
3	button2	button	Đóng phần mềm

5.2. Các giao diện chính:

5.2.1. Giao diện sử dụng chính:



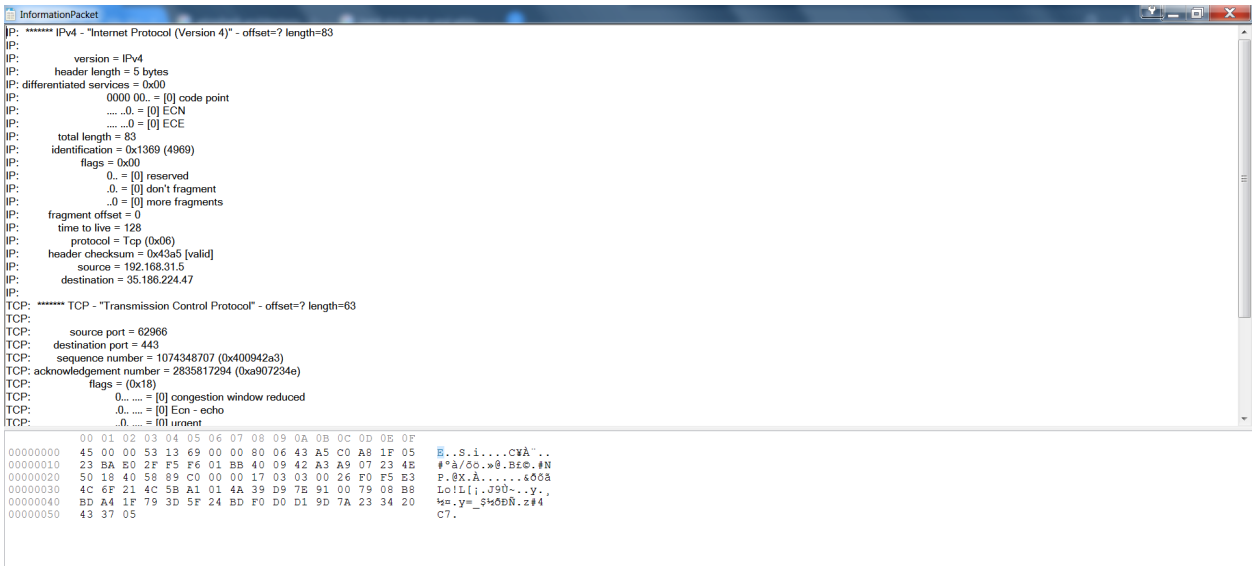
Đặc tả giao diện:

Giao diện chính sử dụng phần mềm			
STT	**Tên thành phần**	**Định dạng**	**Mô tả chức năng**
1	fileToolStripMenuItem	tool strip menu item	Lưu trữ các lựa chọn để trích xuất file (mở, lưu, lưu với định dạng khác và thoát phần mềm)
2	editToolStripMenuItem	tool strip menu item	Lưu trữ các lựa chọn để chỉnh sửa file
3	viewToolStripMenuItem	tool strip menu item	Lưu trữ các lựa chọn về cách trình bày giao diện
4	goToolStripMenuItem	tool strip menu item	Lưu trữ các lựa chọn về các thao tác phân tích sâu (BPS/PPS)

5	captureToolStripMenuItem	tool strip menu item	Lưu trữ các lựa chọn để thao tác thu thập gói tin (bắt đầu thu thập, dừng, bắt đầu lại và thiết lập bộ lọc gói tin)
6	helpToolStripMenuItem	tool strip menu item	Lưu trữ các lựa chọn để hỗ trợ người sử dụng phần mềm
7	toolStripBtnStartCapturin	button	Bắt đầu quá trình thu thập gói tin
8	toolStripbtnStopCap	button	Tạm dừng quá trình thu thập gói tin
9	toolStripbtnRestartCap	button	Bắt đầu lại quá trình thu thập gói tin
10	toolStripCaptureOpt	button	Lưu trữ các lựa chọn khác trong quá trình thu thập gói tin
11	toolStripbtnOpenFile	button	Mở file đã thu thập được từ trước
12	toolStripbtnSaveFile	button	Lưu file đang thu thập được
13	toolStripbtnCloseFile	button	Đóng file đang mở
14	toolStripbtnReloadFile	button	Mở lại file đang được mở
15	toolStripbtnPreviusPacket	button	Di chuyển tới gói tin ngay trước gói tin đang được lựa chọn
16	toolStripbtnNextPacket	button	Di chuyển tới gói tin ngay sau gói tin đang được lựa chọn
17	toolStripbtnFirstPacket	button	Di chuyển tới gói tin ở cuối danh sách
18	toolStripbtnLastPacket	button	Di chuyển tới gói tin ở đầu danh sách
19	toolStripDropDownButton1	drop down button	Lưu trữ lịch sử các thông điệp đã tìm kiếm từ trước
20	toolStripTbFilter	text box	Nhập thông điệp cần tìm kiếm trong danh sách

21	toolStripBtnFillter	button	Lưu trữ các lựa chọn về bộ lọc gói tin
22	listView1	list view	Danh sách các gói tin thu thập được
23	treeView1	tree view	Thông tin chi tiết về một gói tin đang được lựa chọn (dưới dạng người dùng có thể đọc được)
24	hexbox1	hex box	Nội dung dump gói tin (dưới dạng mã hex)

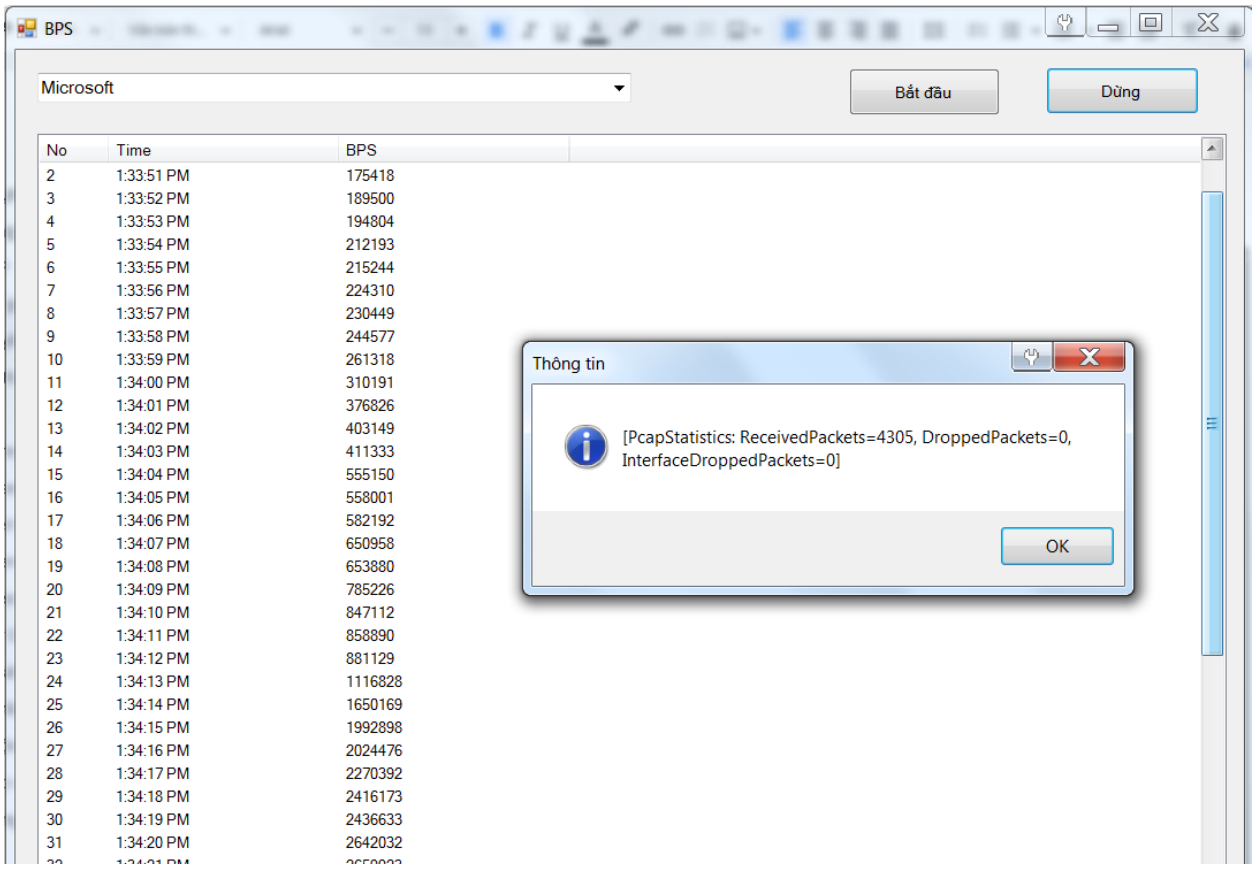
5.2.2. Giao diện mô tả chi tiết gói tin:



Đặc tả giao diện:

Giao diện đưa ra thông tin chi tiết gói tin			
STT	**Tên thành phần**	**Định dạng**	**Mô tả chức năng**
1	richTextBox1	text box	Thông tin chi tiết về một gói tin đang được lựa chọn (dưới dạng người dùng có thể đọc được)
2	hexbox1	hex box	Nội dung gói tin (dưới dạng mã hex)

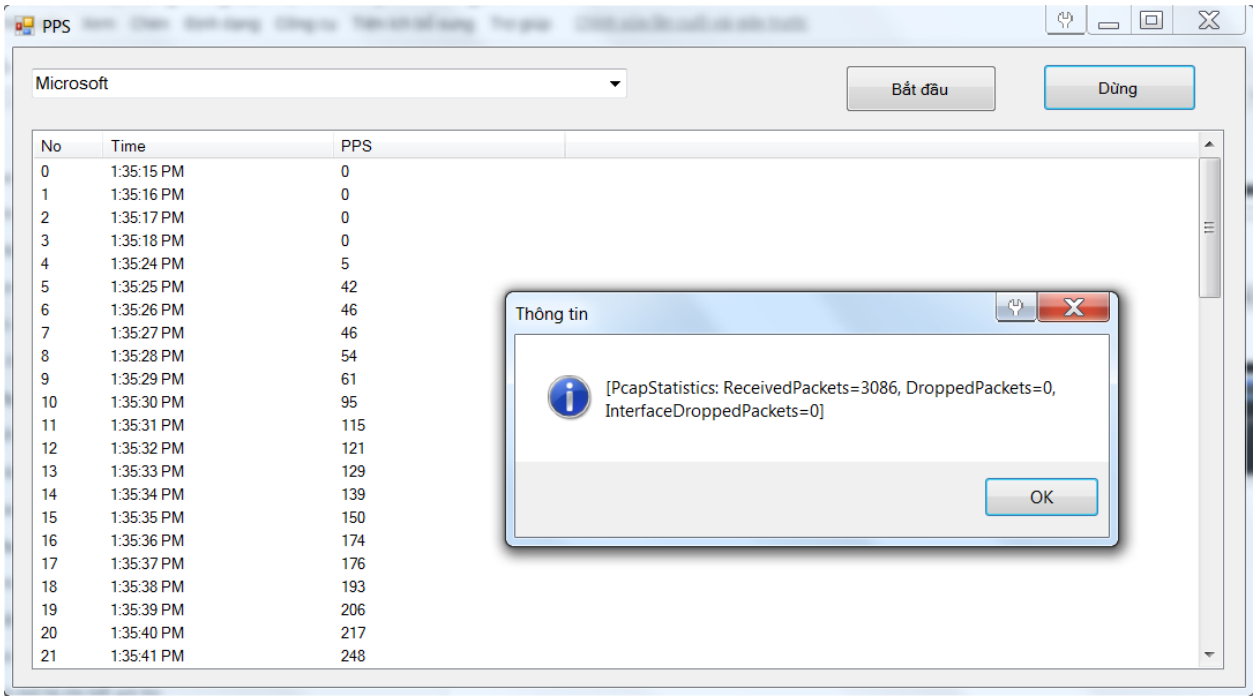
5.2.3. Giao diện phân tích sâu BPS (Byte Per Second):



Đặc tả giao diện:

Giao diện BPS			
STT	**Tên thành phần**	**Định dạng**	**Mô tả chức năng**
1	combo_Interface	combo box	Danh sách các Interface có sẵn
2	btn_start	button	Bắt đầu quá trình thu thập gói tin theo byte/s
3	btn_cancel	button	Tạm dừng quá trình thu thập gói tin theo byte/s
4	listView1	list view	Danh sách các gói tin thu thập được

5.2.4. Giao diện phân tích sâu PPS (Packet Per Second):



Đặc tả giao diện:

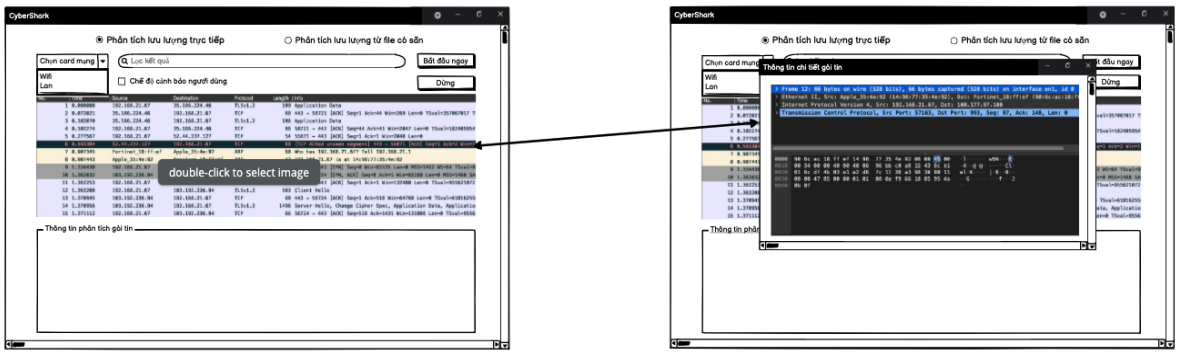
Giao diện PPS			
STT	**Tên thành phần**	**Định dạng**	**Mô tả chức năng**
1	combo_Interface	combo box	Danh sách các Interface có sẵn
2	btn_start	button	Bắt đầu quá trình thu thập gói tin theo gói tin/s
3	btn_cancel	button	Tạm dừng quá trình thu thập gói tin theo gói tin/s
4	listView1	list view	Danh sách các gói tin thu thập được

5.3. Module chính:

Phần mềm CyberShark được xây dựng gồm 2 module chính

5.3.1. Phân tích các gói tin một cách trực tiếp (realtime)

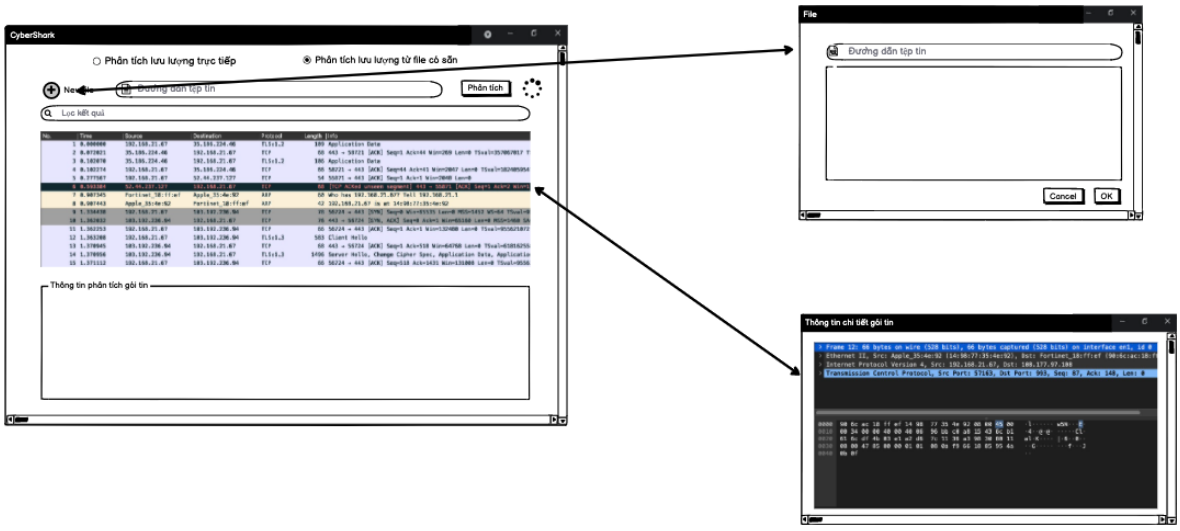
- phân tích sâu ở phần sau



Biểu đồ 5.1. Flow giao diện Phân tích lưu lượng trực tiếp.

5.3.2. Phân tich các gói tin ở một tệp dữ liệu có sẵn

- phân tích sâu ở phần sau

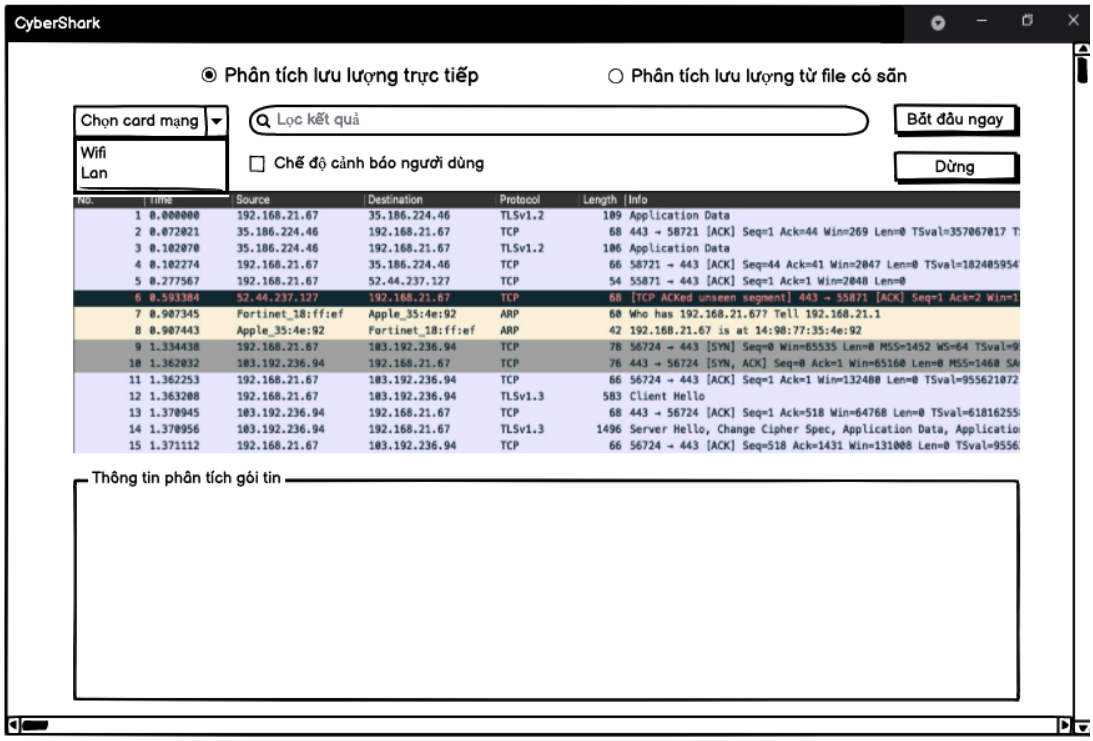


Biểu đồ 5.2. Flow giao diện Phân tích lưu lượng từ file có sẵn.

6. Phân tích nghiệp vụ hệ thống

6.1. Phân tích các gói tin một cách trực tiếp (real-time)

Khi người dùng chọn radio button “Phân tích lưu lượng trực tiếp”. Giao diện sẽ hiển thị như hình.



Hình 6.1. Giao diện Phân tích lưu lượng trực tiếp

Người dùng bắt buộc phải chọn card mạng tùy ý trên combobox, sau đó nhấn nút “Bắt đầu ngay”, quá trình thu thập và phân tích gói tin đi qua card mạng đã chọn bắt đầu, kết quả phân tích sẽ được hiển thị bên dưới theo các trường No (STT), Time (Thời gian), Source (IP nguồn), Destination (IP đích), Protocol (Giao thức), Length (Độ dài gói tin), Info (Thông tin gói tin).

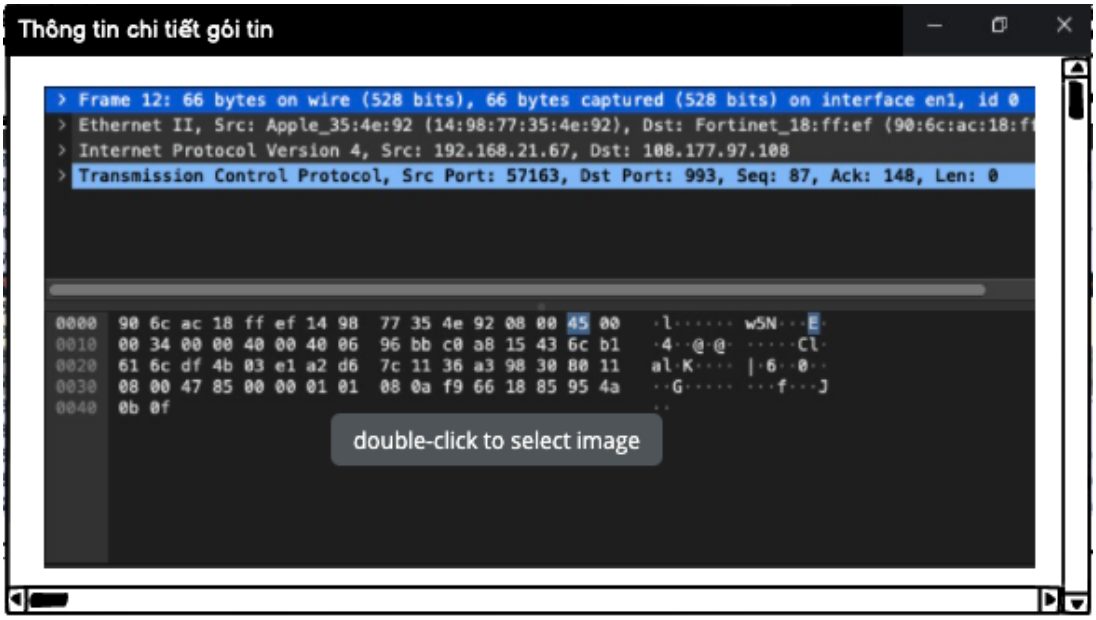
Khi người dùng kích hoạt “Chế độ cảnh báo người dùng”, chương trình sẽ vào chế độ cảnh báo người dùng thông qua notification. Người dùng sẽ nhận được thông báo khi phân tích gói tin, thông báo sẽ đưa ra theo các trường hợp:

+ Cảnh báo người dùng khi truy cập vào một IP botnet, C&C.
+ Cảnh báo người dùng khi gói tin chứa domain web phising.

+ Cảnh báo người dùng khi gói tin chứa protocol là trang web dễ bị lộ lọt thông tin người dùng.

Người dùng có thể lọc kết quả tìm kiếm theo các trường: Protocol, IP source, IP Destination,...

Thông tin phân tích gói tin sẽ được thể hiện ở ô Thông tin chi tiết gói tin, nếu người dùng muốn xem chi tiết hơn gói tin thì kích đúp chuột vào dòng thể hiện từng gói tin thì sẽ hiển thị popup.



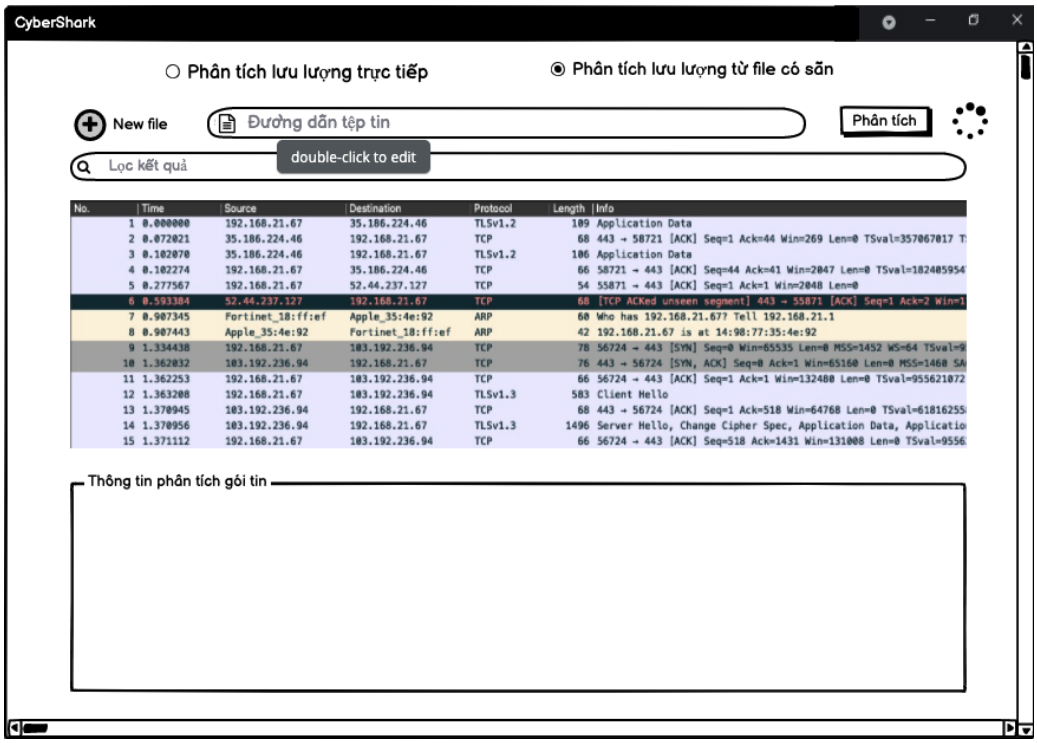
Hình 6.2. Giao diện chi tiết từng gói tin

Người dùng muốn dừng quá trình thu thập và phân tích gói tin thì nhấn nút “Dừng” trên màn hình.

Khi người dùng thoát khỏi chương trình, popup xuất hiện để hỏi ý kiến người dùng về việc có muốn lưu lại lịch sử thu thập và phân tích gói tin hay không.

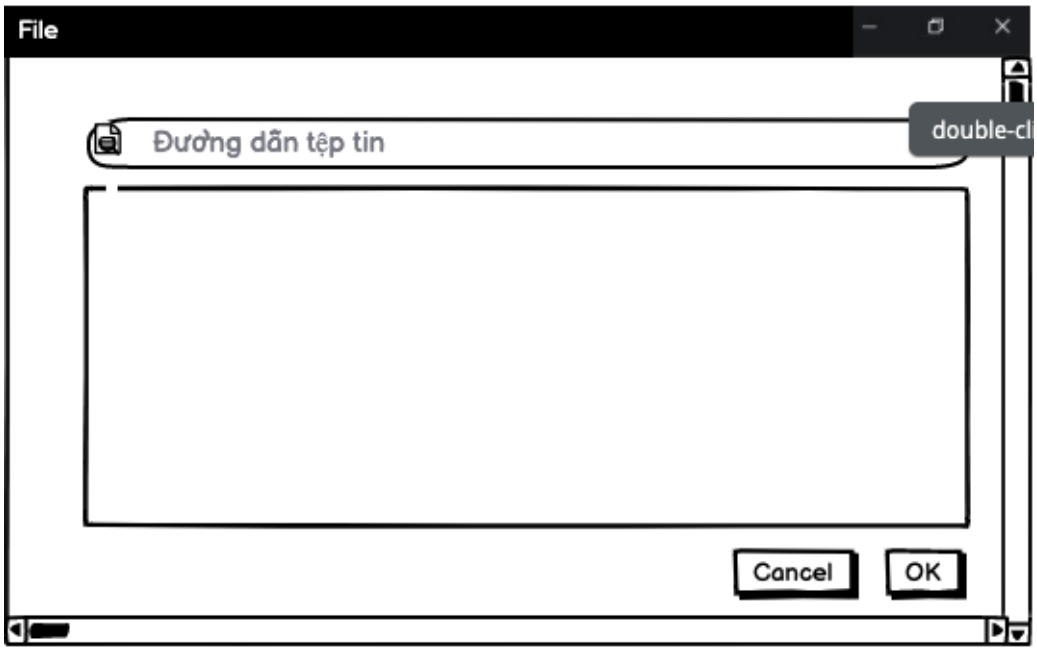
6.2. Phân tích các gói tin ở một tệp dữ liệu có sẵn

Khi người dùng chọn Radio button “Phân tích lưu lượng từ file có sẵn”, giao diện sẽ hiển thị như hình.



Hình 6.3. Giao diện phân tích lưu lượng từ file có sẵn

Khi người dùng chọn “New file”, sẽ hiển thị popup cho phép người dùng chọn tệp có định dạng .pcap



Hình 6.4. Popup chọn tệp

Sau khi người dùng chọn và xác nhận tệp muốn phân tích, người dùng nhấn chọn nút “Bắt đầu ngay”, quá trình phân tích gói tin bắt đầu, và mọi gói tin sau khi phân tích sẽ được hiển thị bên dưới theo các trường No (STT), Time (Thời gian), Source (IP nguồn), Destination (IP đích), Protocol (Giao thức), Length (Độ dài gói tin), Info (Thông tin gói tin).

Người dùng có thể lọc kết quả tìm kiếm theo các trường: Protocol, IP source, IP Destination,... ở ô Lọc kết quả tìm kiếm

Thông tin phân tích gói tin sẽ được thể hiện ở ô Thông tin chi tiết gói tin, nếu người dùng muốn xem chi tiết hơn gói tin thì kích đúp chuột vào dòng thể hiện từng gói tin thì sẽ hiển thị popup (Hình 1.5)

7. Phụ lục

7.1. Chú ý

- Phiên bản nâng cấp mới nhất 6.0 cung cấp hiệu suất sử dụng tốt hơn, do:
  - Hạn chế việc cấp phát bộ nhớ trong lúc chặn bắt gói tin trên đường truyền mạng. Từ đó, hiệu suất sử dụng của các cá nhân tăng lên 130%.
  - Chức năng chặn bắt được cải thiện nhờ sử dụng NativeLibrary.

7.2. Packet.NET

7.2.1. Khái niệm:

- là hợp ngữ .NET hiệu suất cao được sử dụng để tách và xây dựng các gói tin mạng (Ethernet, ip, tcp, udp, …).

7.2.2. Hiệu suất:

- Packet.Net được thiết kế để hiệu suất thiết bị hoạt động tốt nhất. Do đó, với mục đích hỗ trợ việc xác định rõ ràng các gói dữ liệu, Packet.Net thực hiện xử lý dữ liệu thấp nhất.
- Ví dụ: một gói tin TCP sẽ được phân tích cú pháp thành một loạt các đối tượng được liên kết với nhau (Ethernet -> IPv4 -> TCP), nhưng sẽ không có quá trình xử lý dữ liệu nào được thực hiện cho đến khi các trường cụ thể được xác định.
- Ngoài ra, các đối tượng sẽ được trỏ ngay tới bộ nhớ gói tin ngay lập tức, từ đó tránh việc cấp phát cũng như sao chép nội dung gói tin (trừ khi cần thiết).

7.2.3. Đặc điểm chính:

- Hỗ trợ SharpPcap trong việc chặn bắt và phân tích gói tin.
- Hỗ trợ đa định dạng (Ethernet, ip, tcp, udp, ...).
- Cung cấp phần mở rộng hỗ trợ việc thu thập định dạng/loại gói tin raw.

7.3. PacketDotNET.Connections

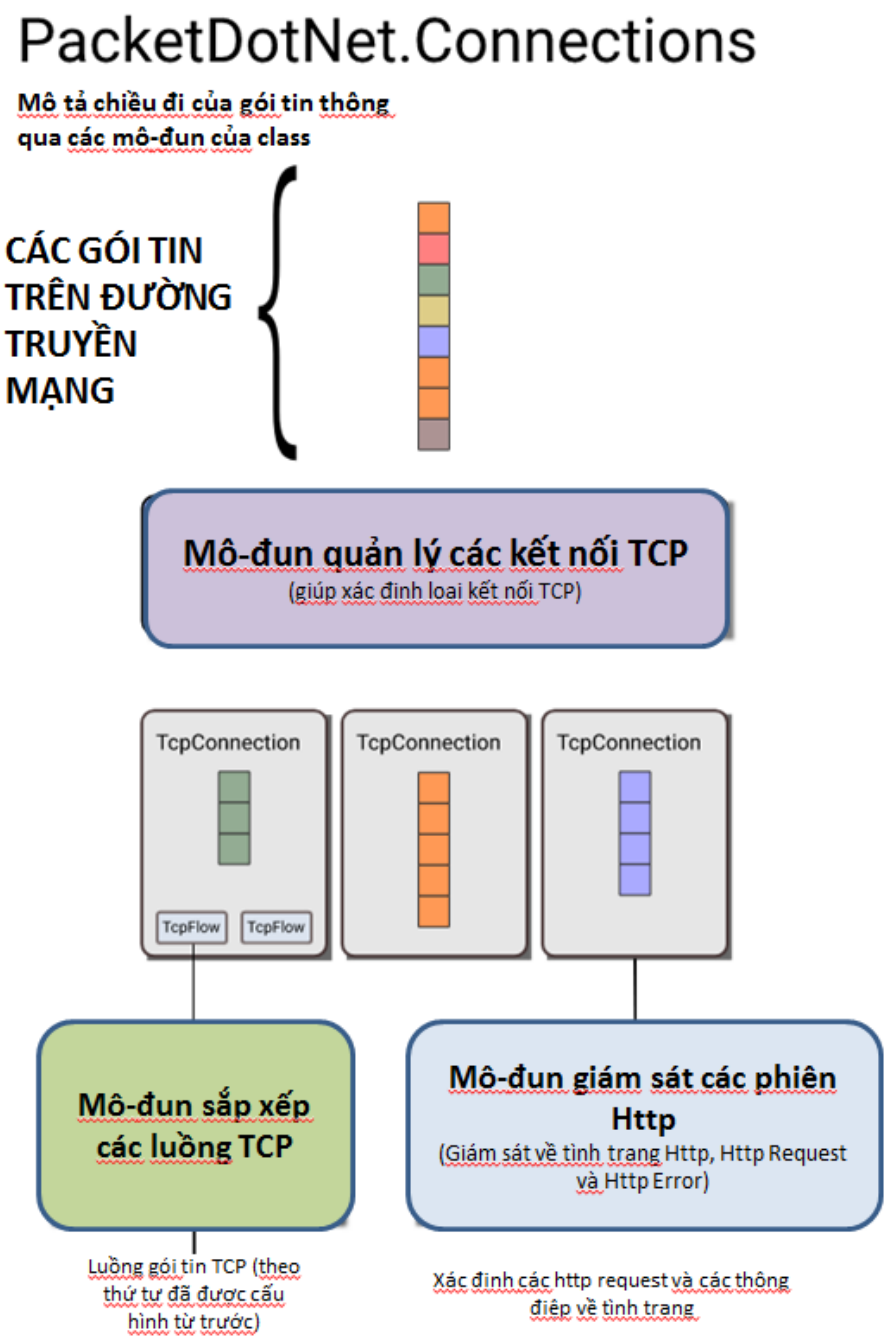
7.3.1. Khái niệm:

- là thư viện bổ sung cho Packet.NET; cung cấp các class để theo dõi các kết nối trên đường truyền mạng (được hỗ trợ cho các kết nối TCP và UDP).

7.3.2. Bối cảnh sử dụng:

- Thư viện được phát triển cho các sản phẩm thương mại để làm việc với SharpPcap và Packet.NET; cho phép hành vi xử lý các luồng dữ liệu TCP.
- Thư viện này là một nền tảng cho việc xây dựng và cải thiện cho những cá nhân muốn triển khai các quy trình xử lý theo hướng kết nối.

7.3.3. Mô hình hệ thống PacketDotNet.Connections



Biểu đồ 7.1. Mô hình hệ thống PacketDotNet.Connections

a) **Đặc điểm chính:**

- Giám sát băng thông.
- Quản lý các kết nối, luồng TCP thu thập được trên lưu lượng mạng.
- Quản lý và giám sát các phiên Http.

8. Những việc đã làm được

- Hoàn thành việc phát triển các giao diện màn hình chính; bao gồm:
    - Giao diện hiện lên danh sách gói tin.
    - Giao diện hiện lên thông tin chi tiết gói tin.
    - Giao diện hiện lên thông tin hex view của gói tin.
    - Thanh công cụ của phần mềm.
    - Giao diện thu thập và chọn interface thích hợp để chặn bắt và phân tích gói tin.
- Hoàn thành việc phát triển các chức năng; bao gồm:
    - Chức năng chặn bắt các gói tin.
    - Chức năng lọc gói tin đã và đang thu thập được.
    - Chức năng phân tích từng gói tin theo từng giao thức tương ứng từ layer 2->4 trong mô hình TCP/IP.
    - Chức năng phân tích sâu BPS và PPS các gói tin.
    - Chức năng mở và lưu file thu thập dạng .pcap.
    - Chức năng hoạt động đa luồng.

9. Tài liệu tham khảo

- https://www.wireshark.org/
- https://github.com/chmorgan/sharppcap/
- https://www.tcpdump.org/
- https://nmap.org/
- https://www.reqrypt.org/windivert.html/
- https://csharpdoc.hotexamples.com/namespace/SharpPcap/
- http://sharppcap.sourceforge.net/htmldocs/SharpPcap/index.html/
- https://www.cnblogs.com/armyao/archive/2010/11/05/1870163.html/